Formal analysis of a non-repudiation protocol*

Steve Schneider
Department of Computer Science
Royal Holloway, University of London
Egham, Surrey, TW20 0EX, UK

Abstract

This paper applies the theory of Communicating Sequential Processes (CSP) to the modelling and analysis of a non-repudiation protocol. Non-repudiation protocols differ from authentication and key-exchange protocols in that the participants require protection from each other, rather than from an external hostile agent. This means that the kinds of properties that are required of such a protocol, and the way it needs to be modelled to enable analysis, are different to the standard approaches taken to the more widely studied class of protocols and properties. A non-repudiation protocol proposed by Zhou and Gollmann is analysed within this framework, and this highlights some novel considerations that are required for this kind of protocol.

1. Introduction

Over the past few years, formal methods have been successfully applied to the analysis of security protocols. The bulk of the effort has been concerned with authentication and confidentiality properties, and there are now a range of maturing techniques and approaches for such analysis, as exemplified in [6], and in [1, 3, 4, 5, 7, 11, 12]. Non-repudiation [2] has not been addressed to the same degree by these techniques, and it is the aim of this paper to consider how the CSP approach presented in [9] extends or adapts to the analysis of this property.

Non-repudiation protocols are used to enable agents to send and receive messages, and provide them each with *evidence* so that neither of them can successfully deny at a later time that the message was transmitted. Each participant aims to collect evidence that could later be shown to a judge to prove that the other party did send or receive the message (as appropriate). A protocol designed to achieve this is generally required to provide the property of correctness of the evidence: that the evidence really is strong enough to guarantee what the holder requires of it.

In some cases, the protocol might also aim to provide fairness: that no party should be able to reach a point where they have the evidence or the message that they require, without the other party also having their required evidence. Fairness is not required for non-repudiation [2], but it may be desirable in some cases. The protocol considered in this paper aims to provide fairness.

Evidence is generally in the form of signed messages, which provide guarantees concerning their originator. In the design of such protocols, fairness is the more difficult property to achieve, and various schemes have been proposed to try to achieve this. The problems and proposed solutions are discussed in [13]. Firstly, the sender and the recipient do not involve any other parties, and gradually release information to each other over many rounds of a protocol so that they effectively obtain the evidence and the message together as a gradual process. Secondly, a trusted third party can be involved in a protocol to handle some of the evidence. The problems and proposed solutions are discussed in more detail in [13].

In contrast to authentication and key-exchange protocols, non-repudiation protocols are not concerned with communication in the presence of a hostile agent between two parties who trust each other. Instead they are employed when a communication is required between two agents who require protection from each other and who do not entirely trust each other to behave honourably in the future. They are typically proposed in the context of a passive communication medium which cannot be manipulated by either party or by other agents, but which may nevertheless have some unreliable behaviour.

In analysis, the system must be modelled from the

*Copyright 1998 IEEE. Published in *Proceedings, 1998 IEEE Computer Security Foundations Workshop*, Rockport Massachusetts, June, 1998.

point of view of a judge who would be used to arbitrate in the case of a dispute. Correctness is concerned with whether a judge, who cannot know *a priori* which agents are honest, must accept evidence as guaranteeing that the message was sent. This concerns the nature of *evidence*: an agent might himself know that a message was sent, and yet not be in a position to prove this to the judge.

The CSP modelling reveals an aspect of non-repudiation unusual for a security property. Most security properties are safety (trace) properties, essentially that nothing bad (a breach of security) should happen at any stage. In the case of the protocol considered in this paper, some of the aspects of non-repudiation involve liveness as well as safety. For example, the evidence that A collects does not guarantee that B has in fact received the message, but it does guarantee that the message must be *available* to B. Non-repudiation can require that certain additional activities ought to be possible.

This paper is organised as follows: the CSP notation is briefly introduced, and the Zhou-Gollmann protocol is then introduced. This protocol and the system required for analysing it are modelled in CSP. The modelling is similar to the approach taken in [10], though the descriptions of the component processes are different to reflect the different property that is being analysed. The CSP specification and verification of the system description are then introduced. The results concerning the required properties of the system are all presented. The proofs were carried out by hand; they are not all included here for reasons of space and readability; there are essentially two kinds of property: safety (achieved via rank functions) and liveness (achieved by considering liveness of the components). A sketch is provided for each kind of proof.

2. CSP notation

CSP is an abstract language designed specifically for the description of communication patterns of concurrent system components that interact through message passing. It is underpinned by a theory which supports analysis of systems described in CSP. It is therefore well suited to the description and analysis of network protocols. For a fuller introduction to the language and the semantic models, the reader is referred to [8].

In CSP, systems are modelled in terms of the events that they can perform. The set of all possible events (fixed at the beginning of the analysis) is denoted Σ. Events may be atomic in structure or may consist of a number of distinct components or fields. An example of events used in this paper are those of the form $c.i.j.m$ consisting of a channel c, a source i, a destination j and a message m.

Processes are the entities that are described by CSP expressions, and they are described in terms of the possible events that they may engage in. The output $c!v \rightarrow P$ is able initially to perform only $c.v$, the output of v on channel c, after which it behaves as P. The input $c?x : T \rightarrow P(x)$ can accept any input x of type T along channel c, following which it behaves as $P(x)$. Its first event will be any event of the form $c.t$ where $t \in T$. The process $P \,\Box\, Q$ (pronounced 'P choice Q') can behave either as P or as Q: its possible communications are those of P and those of Q. An indexed form of choice $\Box_{i \in I} P_i$ is able to behave as any of its arguments P_i.

Processes may also be composed in parallel. If D is a set of events then the process $P \,|[\, D \,]|\, Q$ behaves as P and Q acting concurrently, with the requirement that they have to synchronise on any event in the synchronisation set D; events not in D may be performed by either process independently of the other. Interleaving is a special form of parallel operator in which the two components do not interact on any events: it is written $P \,|||\, Q$, and is equivalent to $P \,|[\{\}]|\, Q$. There is also an indexed form $|||_{i \in I} P_i$.

Processes may also be recursively defined by means of equational definitions.

The *traces* of a process P, *traces*(P), is defined to be the set of finite sequences of events from Σ that P may possibly perform. Examples of traces include the empty trace $\langle\rangle$, and $\langle in.3, out.3, in.5\rangle$ which is a possible trace of the recursive process $COPY = in?x \rightarrow out!x \rightarrow COPY$. If a is an event and tr is a trace, then a in tr means that a appears in the trace tr.

The *failures* of a process P, *failures*(P), is defined to be the set of trace/refusal pairs (tr, X) that P can exhibit, where tr is a trace and X is a set of events that P can *refuse* to participate in after some execution of the sequence of events tr. Examples of failures include the empty failure $(\langle\rangle, \varnothing)$ which is possible for any process, and $(\langle in.3, out.3, in.5\rangle, \{out.3, out.4\})$ which is a possible failure of $COPY$.

Availability or liveness on events can be deduced from the set of failures of a process: for example, if tr is a trace of a process P, and $a \notin X$ for any failure (tr, X) of P, then a cannot be refused after performance of tr, and so it must be available.

Safety specifications are given as predicates on traces, and a process P satisfies a specification $S(tr)$ if all of its traces satisfy $S(tr)$:

$$P \ \mathbf{sat}\ S(tr) \quad \Leftrightarrow \quad \forall\, tr \in traces(P) \bullet S(tr)$$

Liveness specifications are given as predicates on failures, and a process P satisfies a specification $S(tr, X)$ if all of its failures meet that predicate:

$$P \textbf{ sat } S(tr, X) \quad \Leftrightarrow \quad \forall (tr, X) \in \mathit{failures}(P) \bullet S(tr, X)$$

3. The Zhou-Gollmann protocol

The full Zhou-Gollmann non-repudiation protocol is described in [13]. The aim is for A to send a message M to B, and for the parties to obtain evidence that the message was sent and received. The message M is transferred in two stages: an encrypted form is first sent directly to B under some key K, and after A has received evidence of receipt from B, the key K itself is sent via a trusted third party (TTP). The trusted third party makes the key available via *ftp*, and both A and B have the responsibility to retrieve the key and the evidence that it was deposited by A.

Agent B should not be able to extract M until both of these messages have been received.

A cut down version of the protocol, with the unsigned parts of the message omitted, is described as follows:

1. $A \rightarrow B :$ $s_A(f_{NRO}, B, L, C)$
2. $B \rightarrow A :$ $s_B(f_{NRR}, A, L, C)$
3. $A \rightarrow TTP :$ $s_A(f_{SUB}, B, L, K)$
4. $B \leftrightarrow TTP :$ $s_T(f_{CON}, A, B, L, K)$
5. $A \leftrightarrow TTP :$ $s_T(f_{CON}, A, B, L, K)$

Zhou and Gollmann explain the elements of the protocol as follows:

- 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 which is to be sent from A to B.
- C: commitment (ciphertext) for message M, e.g. M encrypted under a key K. The point is that C in itself is not enough to identify the message M, but that C together with K is.
- K: message key defined by A.
- L is a label used to identify a particular protocol run. It should be unique to a single protocol run.
- f_{NRO}, f_{NRR}, f_{SUB} and f_{CON} are flags used to identify the step of the protocol in which a particular message was generated.
- s_i is a private signature key known only to its owner i; and s_T is TTP's private signature key.

The steps of the protocol are explained as follows:

1. With the first message, A sends a signed combination of $C = K(M)$, a label L, and the recipient's name B. B will use this as evidence that $K(M)$ was sent in a run identified with L;
2. B responds with a signed record that C has been received in run L. This will provide evidence for A that $K(M)$ was received;
3. A then sends the key K to the trusted third party together with the label L. If A tries to cheat by sending the wrong key, then he will not obtain the evidence he requires, since $K(M)$ and K' will not convince the judge that M was sent;

4 & 5. Each of A and B can retrieve by means of an *ftp-get* a signed record from TTP that the key K associated with protocol run L has been deposited. Responsibility for retrieving this information rests with the agents themselves, to nullify a possible future claim that 'the message was never received'. Thus both A and B can obtain evidence that the key K was made available to B.

The TTP only needs to handle relatively small messages, and make them available by *ftp*, so this protocol is appropriate even if the messages themselves are extremely large, since TTP never has to handle them directly.

At the end of the protocol run, if A wishes to prove that the message has been received, he presents $s_B(f_{NRR}, A, L, C)$ and $s_T(f_{CON}, A, B, L, K)$ to the judge: the first piece of evidence confirms that B received C, and the second piece confirms that the key was deposited with the TTP, which means that B has access to it, and hence to the message. The label L in both pieces of evidence connects the two items K and C as being associated with the same protocol run.

If B wishes to prove that the message was sent, he presents both pieces of evidence $s_A(f_{NRO}, B, L, C)$ and $s_T(f_{CON}, A, B, L, K)$ to the judge: the first provides evidence that C was sent, and the second provides evidence that K was also sent, to the TTP.

In [13] there is a detailed informal analysis of the protocol with regard to both its correctness properties (that the evidence guarantees what it is supposed to) and its fairness properties (that no party has an advantage at any stage). This paper is concerned with providing a more formal analysis.

Throughout this paper the protocol will be referred to as the ZG protocol.

3.1. CSP modelling

CSP will be used to model and analyse this protocol. This forces the assumptions underlying the protocol, and its expected properties, to be clarified.

Different properties will be associated with different points of view, and these may require alternative models of the system for their analysis. In particular, correctness of the evidence is from the point of view of the judge: it concerns the conclusions that a judge can draw from the particular evidence presented before him, even though he has not witnessed the purported run of the protocol himself.

On the other hand, fairness with respect to obtaining evidence will be the concern of the individual agents involved in the run, and they are only entitled to expect fairness if they follow the protocol faithfully.

Hence an analysis of correctness of the evidence must be considered from the point of view of the judge who may be presented with evidence from some party. The judge is entitled to make some assumptions concerning each of the parties (in particular, that they do not divulge their secret keys), but cannot assume that they have accurately followed the steps of the protocol.

On the other hand, an analysis of fairness for any particular agent will need to model that agent as correctly following the protocol. The judge is not directly concerned with fairness; that is more the concern of the agents themselves, and agents can know that they have followed the protocol even if they are unable to convince the judge of this.

3.2. The architecture

Any CSP model of the system will have to include the two participants in the protocol, who will be labelled A and B, and the trusted third party TTP. It is also reasonable to allow the presence of other agents who are potential protocol participants, since the protocol is expected to be correct even in the presence of other users of the network.

Communication between the agents is generally achieved by sending and receiving messages. The messages are not guaranteed to arrive, and they can arrive in any order. This is best modelled by an explicit process *MEDIUM* whose description contains all of the behaviour expected of it. The transmission of messages from agents will be modelled by a CSP channel *trans*: the event $trans.i.j.m$ means that agent i transmits a message m to agent j. Similarly, the receipt of a message is modelled by use of the CSP channel *rec*: the event $rec.j.i.m$ indicates receipt by j of message m from i.

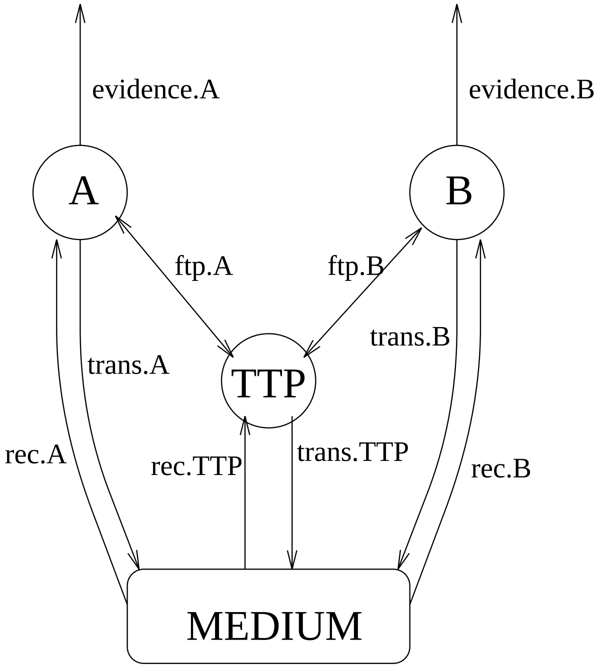

Figure 1. Network for a non-repudiation protocol

Communication is also possible via *ftp* between the agents and the trusted third party. This is a synchronisation between the two participants, and is modelled by the channel *ftp*: the event $ftp.i.TTP.m$ indicates that i receives m from TTP by means of an *ftp-get*.

Finally, the agents have an *evidence* channel which they use to present evidence to a judge.

The entire network is the parallel combination of these components:

$$
\begin{aligned}
&NETWORK = \\
&\quad (|||_{i \in USER}\ AGENT_i(INIT_i)\ [|\,ftp\,|]\ TTP) \\
&\quad [|\,trans, rec\,|] \\
&\quad MEDIUM(\varnothing)
\end{aligned}
$$

This is illustrated in Figure 1.

Having established the architecture of the system, it is now necessary to model the behaviour of the various components.

3.3. The medium

The medium provides an unreliable message delivery service: sending a message does not guarantee that it is received, and messages might be lost.

The attempt to explicitly model the medium raises a number of issues concerning the degree to which the medium is unreliable: it must be decided whether messages can be delivered to the wrong destination, whether they can arrive apparently from someone other than the genuine sender, and whether messages can become altered in transit. It is also necessary to consider whether messages can be delivered to more than one destination, and whether they are removed from the medium once they are delivered. Finally, the potential loss of messages should be considered.

We will firstly assume that messages cannot alter in transit. This amounts to the assumption that any corrupted messages will be detected and disposed of by the medium—such messages will be treated as if they had become lost. Deliberate altering of messages in order to attack the protocol must be carried out by other agents.

We will also have to assume that messages cannot be delivered to the wrong address. This assumption will be discussed later—it is needed for one of the fairness properties ($FAIR1$).

If the protocol can be verified with such a medium, then it is equally correct over a better behaved communications network. It means that agents can have confidence in the protocol even if they do not have confidence in the medium to deliver messages accurately.

This medium is defined most naturally in two clauses:

- If the medium is empty, then it can do nothing but accept messages:

  $$\begin{aligned} &\mathit{MEDIUM}(\varnothing) = \\ &\qquad \mathit{trans}?i?j?m \rightarrow \mathit{MEDIUM}(\{(i,j,m)\}) \end{aligned}$$

- If the medium is not empty ($M \neq \varnothing$) then it can either accept messages, deliver them, or nondeterministically lose them.

  $$\begin{aligned} &\mathit{MEDIUM}(S) = \\ &\qquad \mathit{trans}?i?j?m \rightarrow \mathit{MEDIUM}(S \cup \{(i,j,m)\}) \\ &\qquad \Box \\ &\qquad \Box_{(i,j,m \in S)} \qquad\qquad \mathit{ec}.j!i!m \rightarrow \\ &\qquad\qquad\qquad\qquad \mathit{MEDIUM}(S \setminus \{(i,j,m)\}) \\ &\qquad \sqcap \\ &\qquad \bigsqcap_{(i,j,m) \in S} \mathit{MEDIUM}(S \setminus \{(i,j,m)\}) \end{aligned}$$

  The argument S is the set of messages in the medium.

Messages

The medium itself is ready to accept and pass on any kinds of messages. However, in order to model the agents themselves and what they can do, it is necessary to tie down more precisely what kinds of messages can circulate in the system. This approach was used in [10], and is explained more fully there.

The original informal description of the protocol indicates that the message space contains at least flags, labels, names of users, keys, text, and combinations of these. The set of messages $\mathit{MESSAGE}$ can be given by the following context-free grammar:

$$\begin{aligned} \mathit{RAW} \quad &::= \quad \mathit{FLAG} \mid \mathit{LABEL} \mid \mathit{USER} \\ &\qquad\; \mid \mathit{TEXT} \mid \mathit{KEY} \\ \mathit{MESSAGE} \quad &= \quad \mathit{RAW} \mid \mathit{KEY}(\mathit{MESSAGE}) \\ &\qquad\; \mid \mathit{MESSAGE}.\mathit{MESSAGE} \end{aligned}$$

- The set FLAG contains f_{NRO}, f_{NRR}, f_{SUB}, and f_{CON}.
- The set USER contains A, B, and TTP, as well as other users.
- The set $\mathit{KEY} = \mathit{SECRET} \mid \mathit{PUBLIC} \mid \mathit{SHARED}$ contains a secret key $s_i \in \mathit{SECRET}$ for each $i \in \mathit{USER}$, and $s_T \in \mathit{SECRET}$ which is TTP's secret key; these are used for signing messages in the protocol in this paper. For each secret key s_i it also contains a corresponding public key $p_i \in \mathit{PUBLIC}$. Furthermore, KEY contains other keys SHARED used in the protocol to encrypt text.
- Elements of $\mathit{MESSAGE}$ are either raw, or else encrypted messages, or else concatenated messages.

A 'generates' relation $\vdash$ indicates when new messages m may be generated from a set of already known messages S. It is defined by the following clauses:

- $m \in S \Rightarrow S \vdash m$
- $S \vdash m \wedge S \subseteq S' \Rightarrow S' \vdash m$
- $f \in \mathit{FLAG} \Rightarrow \varnothing \vdash f$
- $l \in \mathit{LABEL} \Rightarrow \varnothing \vdash l$
- $i \in \mathit{USER} \Rightarrow \varnothing \vdash i$
- $k \in \mathit{KEY}, m \in \mathit{MESSAGE} \Rightarrow \{k, m\} \vdash k(m)$
- $s_i \in \mathit{SECRET}, p_i \in \mathit{PUBLIC}, m \in \mathit{MESSAGE} \Rightarrow \{s_i(m), p_i\} \vdash m$
- $s_i \in \mathit{SECRET}, p_i \in \mathit{PUBLIC}, m \in \mathit{MESSAGE} \Rightarrow \{p_i(m), s_i\} \vdash m$
- $sh \in \mathit{SHARED}, m \in \mathit{MESSAGE} \Rightarrow \{sh, sh(m)\} \vdash m$

See [10] for a discussion of this relation.

3.4. The protocol participants

We are now in a position to model the agents potentially involved in the protocol run.

The judge aims to verify that the evidence presented is strong enough to establish non-repudiation, even under the possibility that either or both of the participants have not behaved in line with the protocol, and also that some other agents may have become involved.

In general, an agent is able to send anything over the network that can be generated from the information already in that user's possession. However, it is important to assume that the agents do not divulge their secret signing keys. The agent i can send out all messages that can be generated through $\vdash$, and can also sign messages with s_i.

The 'i-generates' relation $\vdash_i$ is thus defined by the following two clauses:

$$\begin{array}{lcl} S \vdash m & \Rightarrow & S \vdash_i m \\ S \vdash_i m & \Rightarrow & S \vdash_i s_i(m) \end{array}$$

From the point of view of the judge, who can directly observe only the evidence that appears on the *evidence* channels, all the possibilities of an agents behaviour should be considered.

The behaviour of an arbitrary user of the network is therefore described by the CSP process description $AGENT_i$:

$$\begin{array}{l} AGENT_i(S) = \\ \qquad \Box_{j \in USER, S \vdash_i m} \; trans.i!j!m \rightarrow AGENT_i(S) \\ \qquad \Box \; rec.i?j?m \rightarrow AGENT_i(S \cup \{m\}) \\ \qquad \Box \; ftp.i.TTP?m \rightarrow AGENT_i(S \cup \{m\}) \\ \qquad \Box \; \Box_{S \vdash_i m} \; evidence.i!m \rightarrow AGENT_i(S) \end{array}$$

An agent with information S is able to send any message that can be generated from S, and can also present any such information as evidence. It can also receive any message m (which will augment S) either from the medium, or else by an *ftp-get*. The *ftp* channel is distinct from the medium. It models direct, synchronous communication between the TTP and an agent.

Observe that the way we have modelled $AGENT_i$ means that it is always ready to accept messages along $ftp.i.TTP$:

Lemma 3.1

$$AGENT_i(S) \quad \textbf{sat} \quad ftp.i.TTP.m \notin X$$

□

This corresponds to the assumption that the judge must make makes, that any agent is always able to retrieve messages along the channel *ftp*.

For definiteness, the originator of a protocol run will be $AGENT_A$, and a responder will be $AGENT_B$. In other words, a judge might be faced with some evidence on *evidence*$.A$ claiming that B received a message, or with evidence on *evidence*$.B$ claiming that A sent a message. The descriptions of both of these agents are instances of the generic $AGENT_i$.

The definition of $AGENT_A$ allows for the possibility of A executing the protocol correctly, provided $INIT_A \vdash_A L$, $INIT_A \vdash_A K$, and $INIT_A \vdash_A M$. In other words, the ZG protocol is contained in $AGENT_A$'s possible executions, and the protocol need not be given explicitly.

Similarly, the process $AGENT_B(INIT_B)$ is able to execute the responder's part of the protocol. From the point of view of appropriateness of the evidence, no assumptions are required concerning what $AGENT_B$ is or is not able to generate.

Other agents may also be present in the network.

3.5. The trusted third party

The trusted third party described by process TTP accepts signed messages of the form of step 3 of the protocol, and makes them available via ftp. The judge has to assume that the trusted third party acts in accordance with its role in the protocol. It is therefore modelled as follows:

$$\begin{array}{lcl} TTP(S) & = & rec.T?j?s_j(f_{SUB}.b.l.k) \\ & & \qquad \rightarrow TTP(S \cup \{s_T(f_{CON}.j.b.l.k)\} \\ & & \Box \\ & & \Box_{j \in USER, m \in M} \quad ftp.j.TTP.m \\ & & \qquad\qquad \rightarrow TTP(S) \end{array}$$

The trusted third party guarantees that any messages retrieved from it via ftp correspond to receipt of an appropriately signed f_{SUB} message in accordance with the protocol. This is formalised in the following lemma:

Lemma 3.2

$$\begin{array}{lll} TTP(\varnothing) & \textbf{sat} & ftp.i.TTP.(s_T(f_{CON}.j.b.l.k)) \text{ in } tr \\ & & \quad \Rightarrow rec.TTP.j.(s_j(f_{SUB}.b.l.k)) \text{ in } tr \end{array}$$

□

Secondly, TTP meets a liveness property: once a message has been provided by *ftp* to some agent i then it will always be available to any agent i'.

LEMMA 3.3 For any i and i',

$$\begin{array}{lll} \mathit{TTP}(S) & \textbf{sat} & \mathit{ftp}.i.\mathit{TTP}.(s_T(f_{CON}.j.b.l.k)) \text{ in } \mathit{tr} \\ & & \Rightarrow \mathit{ftp}.i'.\mathit{TTP}.(s_T(f_{CON}.j.b.l.k)) \notin X \end{array}$$

□

4. Specification and verification

Specification of Non-repudiation of Origin (NRO)

The non-repudiation of origin property requires that B's evidence provides a guarantee that A sent some particular message. In particular, it should provide the guarantee that A sent a message to B containing the label L and the ciphertext $C = K(M)$, and another message intended for B containing the same label L and the key K; these two messages are taken to establish that A sent M to B.

Expressed in terms of CSP traces, we require that if both $s_A(f_{NRO}.B.L.C)$ and $s_T(f_{CON}.A.B.L.K)$ appear in the trace on $\mathit{evidence}.B$, then both messages $s_A(f_{NRO}.B.L.C)$ and $s_A(f_{SUB}.B.L.K)$ must have been sent by A along $\mathit{trans}.A$.

The evidence cannot guarantee that A transmitted those messages in accordance with the protocol—they might have been sent as components of other messages. Hence the formal trace specification is given by:

$$\begin{array}{lll} \mathit{NRO}(\mathit{tr}) & = & \mathit{evidence}.B.s_A(f_{NRO}.B.L.C) \text{ in } \mathit{tr} \\ & & \wedge\ \mathit{evidence}.B.s_T(f_{CON}.A.B.L.K) \text{ in } \mathit{tr} \\ & & \quad \Rightarrow\ A \text{ sent } s_A(f_{NRO}.B.L.C) \wedge \\ & & \qquad A \text{ sent } s_A(f_{SUB}, B, L, K) \end{array}$$

and the requirement on the system is that

$$\mathit{NETWORK} \quad \textbf{sat} \quad \mathit{NRO}(\mathit{tr})$$

In the definition of NRO, A sent m allows for the possibility that m is contained within some other message that was transmitted by A:

DEFINITION 4.1

$$\begin{array}{lll} i \text{ sent } m & = & \exists\, M : \mathit{MESSAGE};\ j : \mathit{USER} \bullet \\ & & \quad \mathit{trans}.i.j.M \text{ in } \mathit{tr} \\ & & \quad \wedge\ M \text{ contains } m \end{array}$$

□

where the contains relation is defined as follows:

DEFINITION 4.2 For all messages m, m', and m'', and keys k:

- m contains m
- m' contains $m \Rightarrow m''.m'$ contains m
- m' contains $m \Rightarrow m'.m''$ contains m
- m' contains $m \Rightarrow k(m')$ contains m

□

4.1. General properties of the network

In order to establish particular non-repudiation properties, it is beneficial first to establish some general properties of $\mathit{NETWORK}$ which will be useful.

Many properties are of the form 'R precedes T' for sets of events R and T, in the sense that if some event from T occurs in a trace, then some element from R must appear earlier in the trace. Such specifications have been studied in the form of authentication properties, and there is a well-developed theory using 'rank functions' for establishing such properties for systems such as $\mathit{NETWORK}$. Informally, for the network given in this paper, we aim to find a rank function $\rho : \mathit{MESSAGE} \rightarrow \mathbb{Z}$ such that

- every component of the network (the agents, the trusted third party, and the medium), when prevented from outputting R, maintains positive rank (i.e. if only messages of positive rank are input, then any output message must have positive rank). In order to check this for the agents, each 'generates' relation $\vdash_i$ must be checked to establish that if every member of S has positive rank, and $S \vdash_i m$, then m has positive rank.
- Every message in T has rank 0 or less.

If such a rank function can be found, then nothing in T can occur unless something in R occurs previously. The rank function approach is discussed more fully in [10], and is used in this paper without further discussion. It will be illustrated in Lemma 4.3; the other results established using rank functions will be presented without proof for reasons of space.

The key property is that signing provides the required assurances—that if a message is signed by s_i then agent i must have sent it. This is of the form R precedes T, where T is the set of messages in which a message signed with s_i is received, and R is the set of messages in which it is sent by i.

The property is used in two forms: one for when evidence is presented, and one for when a message is received by another agent.

$$
\begin{aligned}
\rho_0(u) &= 1\\
\rho_0(t) &= 1\\
\rho_0(p_i) &= 1\\
\rho_0(s_i) &= \begin{cases} 0 & \text{if } i = i_0\\ 1 & \text{otherwise}\end{cases}\\[1em]
\rho(r) &= \rho_0(r)\\
\rho(s_j(m)) &= \begin{cases} 0 & \text{if } j = i_0 \text{ and } m = m_0\\ \rho(m) & \text{otherwise}\end{cases}\\
\rho(p_j(m)) &= \begin{cases} \rho(m_0) & \text{if } j = i_0 \text{ and } m = m_0\\ \rho(m) & \text{otherwise}\end{cases}\\
\rho(m_1.m_2) &= \min\{\rho(m_1), \rho(m_2)\}
\end{aligned}
$$

Figure 2. Rank function for Lemma 4.3

LEMMA 4.3 For any message m, if $i \neq j$ and $i \neq TTP$, then

$$
\begin{aligned}
&\mathit{NETWORK}\ \textbf{sat}\\
&\quad \mathit{evidence}.j.s_i(m) \text{ in } tr \Rightarrow i \text{ sent } s_i(m)
\end{aligned}
$$

□

Proof Fix the message as m_0 and the signing agent as i_0. If i_0 is blocked on sending any message containing m_0 on *trans* (and $\mathit{evidence}.i_0$), then the rank function in Figure 2 has the required properties. □

DEFINITION 4.4

$$
\begin{aligned}
j \text{ received } m \quad = \quad & \exists\, M : \mathit{MESSAGE};\ i : \mathit{USER} \bullet\\
& \mathit{rec}.j.i.M \text{ in } tr\\
& \wedge M \text{ contains } m
\end{aligned}
$$

□

LEMMA 4.5 For any users i and j, and any message m:

$$
\mathit{NETWORK} \quad \textbf{sat} \quad j \text{ received } s_i(m) \Rightarrow i \text{ sent } s_i(m)
$$

□

This is also proved using the rank function in Figure 4.3.

The trusted third party is used to provide evidence to the various parties. The only signed evidence TTP provides is via ftp:

LEMMA 4.6 For any message m, if $j \neq TTP$, then

$$
\begin{aligned}
&\mathit{NETWORK}\ \textbf{sat}\\
&\quad \mathit{evidence}.j.s_T(m) \text{ in } tr \Rightarrow\\
&\qquad \exists\, h \bullet \mathit{ftp}.h.TTP.s_T(m) \text{ in } tr
\end{aligned}
$$

□

This is also established with an appropriate rank function.

It need not be j himself that retrieved the message directly from TTP, since some other agent h might have retrieved the message and passed it on to j. But some party must have retrieved the message from TTP.

COROLLARY 4.7

$$
\begin{aligned}
&\mathit{NETWORK}\ \textbf{sat}\\
&\quad \mathit{ftp}.i.TTP.s_T(f_{CON}.i.j.l.k)) \text{ in } tr\\
&\qquad \Rightarrow i \text{ sent } s_i(f_{SUB}.j.l.k)
\end{aligned}
$$

□

Proof This follows from Lemmas 3.2 and 4.5 (with TTP as the receiving agent). □

TTP provides a guarantee that i sent the appropriate signed message.

Lemma 4.6 and Corollary 4.7 are needed for both non-repudiation of origin and non-repudiation of receipt: both parties need evidence that the key was deposited, and hence that the other party had access to it.

4.2. Correctness of evidence

Verification of Non-repudiation of Origin (NRO)

Each piece of evidence that B obtains corresponds to a different message that A can be proved to have sent.

Lemma 4.3 with a particular instantiation for m yields for the first piece of evidence that

$$
\begin{aligned}
&\mathit{NETWORK}\ \textbf{sat}\\
&\quad \mathit{evidence}.B.s_A(f_{NRO}.B.L.C) \text{ in } tr\\
&\qquad \Rightarrow A \text{ sent } s_A(f_{NRO}.B.L.C)
\end{aligned}
$$

Lemma 4.6 and Corollary 4.7 together establish for the second piece of evidence that

$$
\begin{aligned}
&\mathit{NETWORK}\ \textbf{sat}\\
&\quad \mathit{evidence}.B.s_T(f_{CON}.A.B.L.K) \text{ in } tr\\
&\qquad \Rightarrow A \text{ sent } s_A(f_{SUB}.B.L.K)
\end{aligned}
$$

These two results together mean that $NETWORK$ satisfies the conjunction of the specifications, which together imply $NRO(tr)$. Hence as required:

$$NETWORK \quad \mathbf{sat} \quad NRO(tr)$$

Verification of Non-repudiation of receipt (NRR)

Non-repudiation of receipt states that if the messages $S_T(f_{CON}.A.B.L.K)$ and $S_B(f_{NRR}.A.L.C)$ appear on $evidence.A$ then B must have received some message containing C, and also K is made available by the TTP. Thus B has effectively received K and C, and knows them to be linked because of the label L.

Unlike the case of NRO (which is concerned with guaranteeing that messages have been sent), there is no guarantee that all of the messages have actually been received by B by the time A presents the evidence. We therefore formulate NRR in part as a liveness specification, requiring that the messages must at least be guaranteed to be available to B. In fact, the evidence does guarantee that B's first message was sent, so the liveness is concerned only with the availability via *ftp* of the message deposited with the TTP.

$$\begin{aligned}
&NRR(tr, X) = \\
&\quad evidence.A.s_B(f_{NRR}.A.L.C) \text{ in } tr \\
&\quad \wedge\ evidence.A.s_T(f_{CON}.A.B.L.K) \text{ in } tr \\
&\qquad \Rightarrow\ B \text{ sent } s_B(f_{NRR}.A.L.C) \\
&\qquad\quad \wedge\ ftp.B.TTP.(s_T(f_{CON}.A.B.L.K)) \notin X
\end{aligned}$$

In order for $NETWORK$ to guarantee that some event $e \notin X$, all the participants in the event e must be willing to perform it. In the case of the event $ftp.B.TTP.(s_T(f_{CON}.A.B.L.K))$, the participants are B and TTP.

The assumptions built into the modelling of $AGENT_B$ yielded the result that $AGENT_B$ **sat** $ftp.i.TTP.m \notin X$, as given in Lemma 3.1.

Lemmas 3.3 and 4.6 together with the fact that $AGENT_B$ is live on the channel $ftp.i.TTP$ all establish that

$$\begin{aligned}
&NETWORK\ \mathbf{sat} \\
&\quad evidence.A.s_T(f_{CON}.A.B.L.K) \text{ in } tr \\
&\quad \Rightarrow ftp.B.TTP.(s_T(f_{CON}.A.B.L.K)) \notin X
\end{aligned}$$

Furthermore, similarly to its use in the verification of NRO, Lemma 4.3 with a particular m establishes that

$$\begin{aligned}
NETWORK \quad \mathbf{sat} \quad & evidence.A.s_B(f_{NRR}.A.L.C) \text{ in } tr \\
& \Rightarrow B \text{ sent } s_B(f_{NRR}.A.L.C)
\end{aligned}$$

These two results together combine to yield

$$NETWORK \quad \mathbf{sat} \quad NRR(tr, X)$$

4.3. Fairness

Having established that the evidence does achieve what is intended, we can now address fairness considerations—each party's access to the evidence.

Fairness in non-repudiation protocols is concerned with the relationship between the gathering of evidence by the involved parties. A protocol is unfair if one party can obtain the evidence he requires before the other party is able to do so. Such an imbalance makes it possible for the party in the advantage to stop participating in the protocol at that stage. Furthermore, the party receiving the message must not be able to access it and know what it contains until the sender has the evidence of receipt: if B has M, then A has the NRR evidence.

An agent is only entitled to expect fairness if he behaves in accordance with the protocol. For example, agent A could send the key K to B along with the first message. In this case, B will be able to access the message before A has the NRR evidence, but A has forfeited any right to complain by failing to behave in accordance with the protocol.

Thus the fairness requirements for agent A require a different modelling of $AGENT_A$, one in which he behaves in accordance with the protocol. Modelling of other agents remains as before, since A has no guarantees about their behaviour, and wishes to be assured of fairness even if they misbehave. Similarly, the fairness requirements for $AGENT_B$ require that agent to be modelled in accordance with the protocol, with the other agents as before.

The agent A running the protocol will then be described as follows:

$$\begin{aligned}
&PROT\_AGENT_A = \\
&\quad trans.A!B!(s_A(f_{NRO}.B.L.C)) \\
&\quad \rightarrow rec.A.B.(s_B(f_{NRR}.A.L.C)) \\
&\quad \rightarrow trans.A!TTP!(s_A(f_{SUB}.B.L.K)) \\
&\quad \rightarrow ftp.A.TTP.(s_T(f_{CON}.A.B.L.K)) \\
&\quad \rightarrow FINISHED_A(\ s_B(f_{NRR}.A.L)), \\
&\qquad\qquad\qquad\qquad s_T(f_{CON}.A.B.L.K))
\end{aligned}$$

The process $FINISHED_i$ describes the result of running the protocol: the two pieces of evidence are ready to be presented.

$$\begin{aligned}
&FINISHED_i(e, f) = \\
&\quad evidence.i!e \rightarrow FINISHED_i(e, f) \\
&\quad \Box\ evidence.i!f \rightarrow FINISHED_A(e, f)
\end{aligned}$$

Once A has run through the protocol and reached $FINISHED_A$, then the two pieces of evidence are ready to be presented.

Similarly, agent B running the protocol is described as follows:

$$\begin{aligned}
&PROT\_AGENT_B = \\
&\quad rec.B?i?(s_i(f_{NRO}.B.L.C)) \\
&\quad \rightarrow trans.B!i!(s_B(f_{NRR}.i.L)) \\
&\quad \rightarrow ftp.B.TTP?(s_T(f_{CON}.i.B.L.K)) \\
&\quad \rightarrow FINISHED_B(s_i(\ f_{NRO}.B.L.C)), \\
&\qquad\qquad\qquad\qquad s_T(f_{CON}.i.B.L.K))
\end{aligned}$$

It is clear that each party might not have all the evidence when the other does, since they might not yet have obtained the last piece of evidence from the TTP via ftp. What we require is that they have unhampered *access* to the evidence. Each $AGENT_i$, once they have performed their *ftp-get*, will be in a position to offer this final piece of evidence. This is an assumption rather than a requirement of the network, and it may be confirmed to hold of the individual agent descriptions as follows:

$$\begin{aligned}
&AGENT_i \ \mathbf{sat} \\
&\quad ftp.i.TTP.m \ \mathsf{in} \ tr \Rightarrow evidence.i.m \notin X
\end{aligned}$$

and hence

$$\begin{aligned}
&NETWORK \ \mathbf{sat} \\
&\quad ftp.i.TTP.m \ \mathsf{in} \ tr \Rightarrow evidence.i.m \notin X
\end{aligned}$$

In other words, the users in the network are able to use whatever they obtain via *ftp* on their evidence channel.

Hence if a particular piece of information m is *available* via ftp to user i in the description $NETWORK$, then this provides that user with the access to m. Thus for fairness it is sufficient to require only that m is available via *ftp*:

$$ftp.i.TTP.m \notin X$$

The communication $ftp.i.TTP.m$ should not appear in the refusal set X.

A user is thus considered to have access to a piece of evidence either if it is already in their possession, or else if it available via ftp. The description of the protocol indicates that the NRR and NRO evidence should already be in each participants' possession by the time the other has finished the run, and that the evidence provided by the TTP should be made available to each of them.

Fairness for A concerning message receipt

Firstly we consider the case where B should not know what the message M is until proof of receipt has been provided to A. If B is able to provide the message M (along its *evidence* channel, say), then A must have proof of receipt. This can be expressed as a liveness requirement:

$$\begin{aligned}
&FAIR1(tr, X) = \\
&\quad evidence.B.M \ \mathsf{in} \ tr \\
&\quad \Rightarrow \\
&\quad\ ftp.A.TTP.(s_T(f_{CON}.A.B.L.K)) \notin X \\
&\quad\ \vee \ \ (evidence.A.s_B(f_{NRR}.A.L.C) \notin X \\
&\quad\qquad \wedge \ evidence.A.s_T(f_{CON}.A.B.L.K) \notin X)
\end{aligned}$$

A may not actually have obtained the evidence via ftp, but must at least be in a position to do so. The way $PROT\_AGENT_A$ is defined, A is not ready to provide any evidence until the *ftp* event has occurred.

The proof obligation is that

$$\begin{aligned}
&((PROT\_AGENT_A \ ||| \ (|||_{i \neq A} AGENT_i)) \\
&[|\, ftp \,|]\ TTP) \\
&[|\, trans, rec \,|]\ MEDIUM \\
&\qquad \mathbf{sat} \ FAIR1(tr, X)
\end{aligned}$$

and this is established along the lines of earlier proofs. The crux of the proof is that B cannot obtain the key K until it is provided by TTP. Thus for this property it is necessary to assume that none of the agents apart from A initially knows the key K. The key is sent out by A exactly once, to TTP, so no other party will receive that message until TTP gives it out. This relies on the model of the medium as delivering messages accurately; this is discussed further in Section 5.

Fairness for B obtaining evidence

Secondly, if A has proof of receipt then B must be in a position to obtain proof of origin.

$$\begin{aligned}
&FAIR2(tr, X) = \\
&\quad evidence.A.s_T(f_{CON}.A.B.L.K) \ \mathsf{in} \ tr \\
&\quad \wedge \ evidence.A.s_B(f_{NRR}.A.L.C) \ \mathsf{in} \ tr \\
&\quad \Rightarrow \\
&\quad\ ftp.B.TTP.(s_T(f_{CON}.A.B.L.K)) \notin X \\
&\quad\ \vee \ \ (evidence.B.s_A(f_{NRO}.B.L.C) \notin X \\
&\quad\qquad \wedge \ evidence.B.s_T(f_{CON}.A.B.L.K) \notin X)
\end{aligned}$$

This specification states that if A is able to present the evidence concerning NRR, then B must either be able to provide the evidence concerning *NRO* or be in a position to obtain it.

The proof obligation here is that

$$\begin{aligned}&((PROT\_AGENT_B \;|||\; (|||_{i \neq B} AGENT_i))\\ &|[\,ftp\,]|\; TTP)\\ &|[\,trans, rec\,]|\; MEDIUM\\ &\quad \textbf{sat}\; FAIR2(tr, X)\end{aligned}$$

and this is straightforward to establish.

Fairness for A obtaining evidence

Conversely, if B has proof of origin then A should have proof of receipt.

$$\begin{aligned}&FAIR3(tr, X) =\\ &\quad evidence.B.s_T(f_{CON}.A.B.L.K) \text{ in } tr\\ &\quad \wedge\ evidence.B.s_A(f_{NRO}.B.L.C) \text{ in } tr\\ &\quad \Rightarrow\\ &\quad\ ftp.A.TTP.(s_T(f_{CON}.A.B.L.K)) \notin X\\ &\quad\ \vee\ \ (evidence.A.s_B(f_{NRR}.A.L.C) \notin X\\ &\qquad\ \wedge\ evidence.B.s_T(f_{CON}.A.B.L.K) \notin X)\end{aligned}$$

The proof obligation here is that the network with A executing the protocol should satisfy $FAIR3(tr, X)$.

5. Discussion

In this paper we have considered a particular non-repudiation protocol and analysed it both with respect to correctness of the evidence, and with respect to fairness to the participants. The hope has been to extract some general understanding of how to model and analyse non-repudiation protocols from this particular example. The specifications that were formulated were necessarily influenced by the protocol itself:

1. **Correctness of evidence:** A kind of 'authentication' property that requires that if various pieces of evidence $e_1 \ldots e_m$ are in the possession of an agent, then some other messages $m_1 \ldots m_n$ must have been sent (in the case of NRO), or received or been made available (in the case of NRR) to some other agents.

2. **Fairness:** A property requiring that if the message being sent, or the various pieces of evidence $e_1 \ldots e_m$ appearing in an 'authentication' requirement of type 1, are in the possession of the appropriate agent, then the other party should also have access to the evidence that he requires.

The first of these properties is a concern of every non-repudiation protocol; the second is a property that is in desirable in some cases, though it is not an essential aspect of non-repudiation protocols.

Unusually for security properties, some of these properties are formulated in terms of liveness, and so not only traces but also refusals of the system need to be considered for such properties.

Curiously, the verifications of the correctness of evidence properties are carried out without reference to the protocol at all, but only with respect to the capabilities and assumptions concerning the participating agents. On reflection this is appropriate, since the judge cannot know that either party has carried out the protocol. These properties are concerned with the nature of evidence rather than with how the parties distribute it. The verification also means that the parties cannot collude to fool the judge (though in any case it is not clear why they would wish to) since such behaviour has already been considered within the general description of the processes.

The process of modelling and verifying the protocol and the network in CSP revealed a number of issues that were not immediately obvious. For example, I first attempted to verify the *FAIR* properties with the original general descriptions of the agents. This attempt failed because there was no guarantee that the agents would not send some message to undermine their own fairness requirement. One possibility is that agent A could send the first and the third message of the protocol without waiting for the response from B. In this case, B will receive the message and the evidence, but A will not. It thus became clear that the fairness properties should only be verified for agents that faithfully follow the protocol, which is obvious in hindsight but it is comforting to know that the analysis process forces this point to become explicit.

The other issue that became revealed by the modelling process concerned the medium: my original description of the medium did allow for the possibility of messages being delivered to parties other than the intended recipient: a message $trans.i.j.m$ put onto the medium could result in a delivery $rec.k.l.m$ to a completely different agent k, and apparently from another agent l (though this last aspect does not cause a problem). All the properties concerning the evidence: *NRO*, *NRR*, *FAIR*2, and *FAIR*3, remain true even with this less reliable medium, but the property *FAIR*1 does not hold, because the third step of the protocol, which reveals key K to TTP, can be misdelivered to B and never reach TTP. This will allow B to read the message M without A obtaining the evidence he requires. If the protocol is required over this kind of medium, then it would make sense for A to provide the third message of the protocol to TTP by means of an *ftp-put*. In fact, it does not matter if B can listen in on such a communication, provided it can be guaranteed that TTP also receives it.

The use of an *ftp* server in a non-repudiation protocol is a novel idea introduced by Zhou and Gollmann, and it is not clear how easily the CSP analysis of this protocol would generalise to other non-repudiation protocols. It seems likely that the CSP properties which capture non-repudiation would also apply naturally to other protocols which involve trusted third parties, but it is less clear whether an analysis of a two party multi-pass non-repudiation protocol would show up different issues and perhaps require a different approach to specification. This is a topic for future research.

In summary, modelling and analysing this protocol in CSP has helped to clarify issues concerning the protocol and its context, and has enabled a formal statement of the specification claimed for the protocol, and corresponding verification.

Acknowledgements

I am grateful to Peter Ryan for comments and discussion on an earlier draft of this paper, and to the anonymous referees for their comments.

References

[1] M. Abadi and A. Gordon. A calculus for cryptographic protocols: the spi calculus. *Information and Computation*, 1998. (to appear).

[2] I. JTC1. Information technology - open systems interconnection - security frameworks in open system, part 4: Non-repudiation, 1995. ISO/IEC DIS 10181-4, 1995.

[3] R. Kemmerer, C. Meadows, and J. Millen. Three systems for cryptographic protocol analysis. *Journal of Cryptology*, 7(2), 1994.

[4] G. Lowe. Towards a completeness result for model checking of security protocols. University of Leicester, 1998. draft.

[5] J. Millen. The interrogator model. In *IEEE Computer Society Symposium on Research in Security and Privacy*, 1995.

[6] H. Orman and C. Meadows, editors. *DIMACS Workshop on Design and Formal Verification of Security Protocols*. Rutgers University, 1997.

[7] L. C. Paulson. Proving properties of security protocols by induction. In *CSFW10*. IEEE Press, 1997.

[8] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice-Hall, 1997.

[9] S. A. Schneider. Security properties and CSP. In *IEEE Computer Society Symposium on Research in Security and Privacy*, 1996.

[10] S. A. Schneider. Verifying authentication protocols with CSP. In *CSFW10*. IEEE Press, 1997.

[11] P. Syverson and C. Meadows. A formal language for cryptographic protocol requirements. *Designs, Codes and Cryptography*, 7, 1996.

[12] T. Woo and S. Lam. A semantic model for authentication protocols. In *IEEE Computer Society Symposium on Research in Security and Privacy*, 1993.

[13] J. Zhou and D. Gollmann. A fair non-repudiation protocol. In *IEEE Computer Society Symposium on Research in Security and Privacy*, 1996.