# Mechanical Proofs about a Non-Repudiation Protocol

Giampaolo Bella[1,2] Lawrence C Paulson[1]

[1] Computer Laboratory, University of Cambridge
Pembroke Street, Cambridge CB2 3QG (UK)
{gb221,lcp}@cl.cam.ac.uk

[2] Dipartimento di Matematica e Informatica, Università di Catania
Viale A. Doria 6, I-95125 Catania (ITALY)
giamp@dmi.unict.it

**Abstract.** A non-repudiation protocol of Zhou and Gollmann [18] has been mechanically verified. A non-repudiation protocol gives each party evidence that the other party indeed participated, evidence sufficient to present to a judge in the event of a dispute. We use the theorem-prover Isabelle [10] and model the security protocol by an inductive definition, as described elsewhere [1, 12]. We prove the protocol goals of *validity of evidence* and of *fairness* using simple strategies. A typical theorem states that a given piece of evidence can only exist if a specific event took place involving the other party.

## 1 Introduction

A wide variety of techniques are available for verifying cryptographic protocols [3, 8, 12, 14]. Past work has focused largely on two security goals: *confidentiality* (who can read the message?) and *authenticity* (who originated the message?). One direction for further research is to attempt proofs of more esoteric security goals. Traditional protocols help a pair of honest agents to communicate in the presence of an attacker, but in some situations agents may act unfairly and abandon protocol sessions before these terminate. The present work concerns *non-repudiation*, which seeks to prevent a party from abandoning an agreement. Non-repudiation would provide us with a reliable means of making contracts over a network.

The primary goal of a non-repudiation protocol is *validity of evidence*. It must provide each peer with convincing evidence of the other's participation in a protocol session. If one peer falsely denies participating in a session, then the other peer can present his evidence to a judge, who can safely conclude that the other peer did participate. Crucially, the judge does not have to monitor the network traffic, but can make his judgement on the basis of the evidence alone. Some of the evidence is usually referred to as *non-repudiation of origin*, other as *non-repudiation of receipt*. The initiator of a session typically seeks evidence of the first form, and the responder typically looks for evidence of the second form.

An additional goal of some non-repudiation protocols is *fairness*: at no time should one peer hold more evidence than the other does. Although fairness is not indispensable in all situations, it may be needed for certain e-commerce transactions. For example, if a client $C$ holds evidence that a merchant $M$ received $C$'s request for goods, fairness means that $C$ cannot deny sending the request: $M$ holds the corresponding evidence. Similarly, if $M$ holds evidence that $C$ received the goods, fairness means that $C$ holds evidence that it was $M$ who sent them. In the latter case for example, $M$ could claim the payment for the goods but, should the goods be unsatisfactory, $C$ could demand a refund. Resolving such disputes becomes a matter of cyberlaw; the judge referred to above could be a real judge sitting in a traditional courtroom.

A number of protocols have been designed to achieve non-repudiation, but they are not yet deployed [7, 9, 18]. Verifying them formally [16, 17] might increase their credibility. Proving non-repudiation was one of the the first author's reasons for extending the Inductive Approach to verifying cryptographic protocols [12] with message reception [1] and agents' knowledge [2]. This paper shows that we have now achieved that purpose through the development of simple strategies to prove validity of evidence and fairness. We were pleased to observe that these strategies differ little from those for proving authentication goals [5, 6], and that the approach required no extensions. Our proofs were conducted on a popular non-repudiation protocol due to Zhou and Gollmann [18] using the second author's original modelling of agents. An unlimited population of agents can only send messages of the form that the protocol prescribes but can quit a protocol session at any time; the *spy* can send messages of arbitrary form.

This paper is organised as follows. A brief overview of the Inductive Approach (§2) precedes the description of our strategies to proving the non-repudiation goals (§3). The Zhou-Gollmann protocol is described (§4), modelled (§5) and verified without the spy (§6). Then, the verification is repeated in the presence of the spy (§7). Finally, the related work is discussed (§8), and some conclusions are given (§9).

## 2 The Inductive Approach

The Inductive Approach has been used successfully to analyse Local Area Network protocols [5], Internet protocols [13], e-commerce protocols [4] and smart card protocols [3]. Here, we recall only its main concepts, but a full treatment may be found elsewhere [1, 12].

The approach draws from the observation that the goals of security protocols are invariants of the protocol execution, so proving the goals means showing that they are preserved by all protocol steps. The inductive model of a protocol is the set of all possible histories (*traces*) of the network that the protocol execution may produce. There is no limit to the number of agents who may participate. They may also interleave sessions at will.

A trace is a list of network events of the following form:

– $\mathsf{Says}\, A\, B\, X$, indicating that agent $A$ sends a message $X$ to agent $B$;

– Gets $A\,X$, indicating that $A$ receives $X$;
– Notes $A\,X$, indicating that $A$ notes down $X$ for future use.

The last event can model an agent's storing a message component or the result of a computation [13].

There are traces in which some events have not taken place although the necessary preconditions are met. Therefore, the protocol model does not force events to happen; messages may be sent but may not be received, and agents may abandon a protocol execution. This captures the unreliability of the communications, and a degree of unfairness of agents. However, the only agent who can build messages other than those prescribed by the protocol is the spy. Also, messages cannot alter during transmission.

Given a trace *evs*, we model the knowledge that agent $A$ acquires during the network history denoted by *evs* as $\mathsf{knows}\,A\ evs$ [2]. This is the set of messages that $A$ sent or received or noted on *evs*. In particular, $\mathsf{knows\,Spy}\ evs$ contains the private signature keys of a set bad of agents and all messages that were sent or received by anyone or noted by bad agents on *evs*.

Three operators can be applied to a message set $H$:

– parts, yielding all components of messages in $H$, except encryption keys;
– analz, yielding those components of messages in $H$ whose encryption key is recursively available;
– synth, yielding all messages constructed by concatenation or encryption from messages recursively obtained from $H$.

The special set $\mathsf{synth}(\mathsf{analz}(\mathsf{knows\,Spy}\ evs))$ contains all messages that the spy can synthesize using the components obtained from the analysis of the traffic with the help of bad agents' private keys. The spy can send in the traffic any message derived from that set (§7).

## 3 Strategies to Proving Non-Repudiation

How can a judge who is off-line evaluate the non-repudiation evidence presented by a peer? The judge could perhaps make a decision given a full log of the network traffic, but that would not be practical. Our proofs can help the judge by establishing that a given piece of evidence guarantees that certain critical events occurred. Our proofs are subject to human error (for instance, our model could be too abstract), but they add credibility to the protocol. The judge weighs up these points in making his decision; a Guilty verdict requires the absence of a reasonable doubt.

Each trace of the protocol model is in fact a full log of a network history. So, scanning the trace tells what events have occurred. These observations inspire our strategy to proving validity of evidence, which must be applied for each piece of evidence that an agent presents to the judge. If $A$ presents evidence $X$, then certainly $A$ holds $X$; formally: $X \in \mathsf{knows}\,A\ evs$ for some trace *evs*. Our strategy rests on an assumption of that form, and develops through two main types of result:

1. If $A$ holds some evidence $X$, then $A$ got the evidence from the network, perhaps from the spy.
2. If $A$ got evidence $X$, then $B$ sent some other evidence $Y$. Alternatively —
2′. If $A$ got evidence $X$, then $B$ was entitled to receive some other evidence $Y$.

Proving a theorem of the form (1) and a theorem of the form (2) typically serves to establish validity of the evidence for non-repudiation of origin. Proving a theorem of the form (1) and a theorem of the form (2′) typically serves to establish validity of the evidence for non-repudiation of receipt.

Proving theorems of the form (1) is novel as it involves reasoning on the knowledge of friendly agents. Since a friendly agent *only* knows what she sends or receives or notes [2], these proofs generate longer case splits than previous proofs [6, 12] based on the spy's knowledge, which includes *everything* that is sent or received by anyone or noted by bad agents. By contrast, theorems of the form (2) or (2′) can be proved conventionally, as they resemble authentication theorems [5].

The strategy for proving fairness is simple once the theorems assessing validity of evidence are available. We need to establish is that, if some evidence is available to a peer, then other evidence is available to the other peer. This is in fact a possible way to read the theorems on validity of evidence. Simple lemmas stating that an agent performs an event only if he has performed another one may be sometimes required.

## 4 A Fair Non-Repudiation Protocol

We choose a recent non-repudiation protocol, shown in Fig. 1, that also aims at fairness [18] as a case study to demonstrate our approach. The protocol was designed by Zhou and Gollmann, who also published a version aiming at efficient implementation [19]. One of the motivations for our choice was the existence of significant related work [16, 17], which is discussed in the next section. It is useful to outline the syntax we use:

- $A$ is the initiator of a protocol session with $B$;
- $B$ is the responder of the session initiated by $A$;
- $\mathsf{TTP}$ is the trusted third party;
- $M$ is the message that $A$ wants to transmit to $B$;
- $K$ is the key that $A$ chooses to transmit $M$;
- $C$ is $M$ encrypted with $K$ (the $C$ refers to $A$'s Commitment to $B$);
- $L$ is a unique label identifying the session between $A$ and $B$;
- $f_\star$ are the non-repudiation flags;
- $sK_X$ is the private signature key of agent $X$ (no syntax is needed for the public verification key of $X$);
- $\{\!|m|\!\}_{sK_X}$ is the signature of message $m$ by key $sK_X$.

The protocol intends to transmit a message $M$ from $A$ to $B$, giving $A$ evidence for non-repudiation of receipt, giving $B$ evidence for non-repudiation of

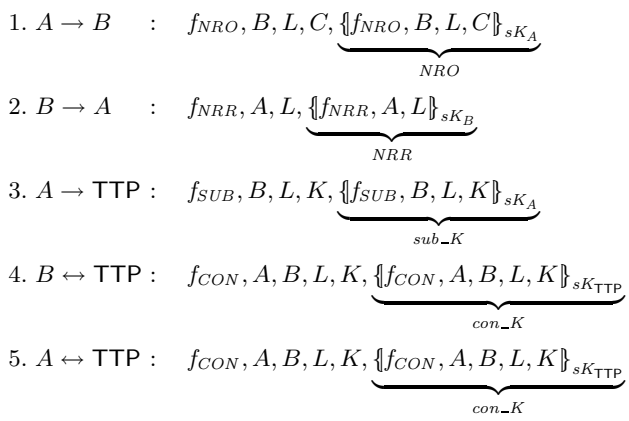


**Fig. 1.** The Fair Zhou-Gollmann Protocol

origin, and ensuring fairness. The first protocol step prescribes that $A$ pick a cryptographic key $K$ and a random label $L$. Then, $A$ uses symmetric cryptography to build $C$ out of $M$ and $K$, signs $\{\!|f_{NRO}, B, L, C|\!\}$ and sends the result to $B$ along with the unsigned message, which is in general needed for signature verification. Note that $A$ sends $M$ encrypted, so $B$ will not be able to obtain $M$ until he gets $K$. Upon reception of the first message, $B$ verifies $A$'s signature, signs $\{\!|f_{NRR}, A, L|\!\}$ and sends the result to $A$. Upon reception of the second message, $A$ lodges $K$ with $\mathsf{TTP}$ by sending her signature on $\{\!|f_{SUB}, B, L, K|\!\}$.

Once $\mathsf{TTP}$ has successfully verified $A$'s signature on the received message, $\mathsf{TTP}$ signs $\{\!|f_{CON}, A, B, L, K|\!\}$ and makes it available in its public directory. This message confirms that the key $K$ concerns the session between $A$ and $B$ that is identified by label $L$. The last two steps, which are interchangeable, see the peers *ftp get* the message available from $\mathsf{TTP}$. The protocol assumes that nobody can interfere with an *ftp get* operation, but we will relax this assumption below (§7).

Zhou and Gollmann [18] observe that, even if the peers do not want to play fair, they must complete a session in order to get sufficient evidence to win any disputes with each other. Let us informally analyse how to resolve disputes. From $B$'s standpoint, it appears that obtaining *con_K* signifies that $A$ submitted $K$, bound to label $L$, to $\mathsf{TTP}$; obtaining *NRO* should signify that $A$ sent $C$ as a commitment bound to label $L$. In consequence, message $M$, obtained by decrypting $C$ with $K$, should have originated with $A$. From $A$'s standpoint, a similar reasoning seems feasible. If $A$ holds *con_K*, this should guarantee that $A$ lodged $K$ and $L$ with $\mathsf{TTP}$, and so $B$ should be able to get it via *ftp*. If $A$ also holds *NRR*, it should be the case that $B$ accepted commitment $C$. In consequence, $B$ would be able to obtain $M$.

This reasoning might resemble that of a judge who is provided with evidence *NRO*, *con_K*, $M$, $C$, $K$, $L$ by agent $B$, or with a similar evidence (but *NRR* rather

than *NRO*) by agent $A$. It is not trivial to verify such reasoning for all possible network histories. The communication means is unreliable, and the protocol is executed by an unlimited number of agents, each entitled to interleave or quit sessions at will.

## 5 Modelling a Fair Non-Repudiation Protocol

We build our protocol model on the Isabelle theory Public [11] for cryptographic protocols based on asymmetric encryption. The theory models three kinds of network agents: a spy, whom we discuss later (§7), a trusted server, which is renamed as TTP here, and an unlimited population of legitimate agents. Each agent $X$ is endowed with a key pair. His private signature key, priK $X$, he keeps secret; his public verification key, pubK $X$, is known to all. The theory also provides a primitive for encryption, Crypt, which we use where the protocol requires a digital signature.

The protocol model is the set of traces zg, whose inductive definition is in Fig. 2. A rule for the base of the induction, stating that the empty trace belongs to zg, is omitted from the figure. Rules ZG1, ZG2 and ZG3 respectively model the first three steps of the protocol. Note that agent $A$ chooses a fresh nonce in rule ZG1 to initiate the protocol with $B$. Recall that $A$ runs the protocol because she wants to transmit some message $M$ to $B$. All these messages and the ciphertexts obtained from them by any key constitute the set targetmsgs. We reasonably assume that this set contains none of the other messages (either atomic or compound) exchanged by the protocol. Also, $A$ is free to choose any key to encrypt $M$, even an old one — we merely assume that she cannot pick private signature keys.

We highlight the important certificates by defining them in the premises, using equations; we use the names so defined in the conclusions. When a certificate is defined in the premises of a rule, then the rule only applies for a certificate of the specified form: informally, the agent verifies it. For example, $B$ must check that *NRO* in rule ZG2 is signed by $A$ in order to learn the sender of the message just received and address *NRR* to her.

By contrast, $A$ does not need to check that *NRR* in rule ZG3 is signed by $B$ because *NRR* is associated to label $L$, which $A$ knows to be associated with $B$. Clearly, the check becomes mandatory in the presence of the spy, who can actively intercept and fake the messages that are in the traffic (§7).

Rule TTP_prepare_ftp models TTP's preparation of the key confirmation *con_K*. Note that TTP verifies the signature on *sub_K* to learn the identities of the peers of $K$. All the components needed to verify that signature are available. The installation of *con_K* in TTP's public directory can be modelled by a Notes event.

Rules A_ftp and B_ftp model the peers' retrieval of *con_K*. The two rules are not forced to fire simultaneously, since each peer decides independently whether to terminate the protocol. Rather than introducing a new event to express the *ftp get* operation, we again adopt Notes. Using a Gets event instead would violate

ZG1

```
[| evs1 ∈ zg; Nonce L ∉ used evs1; C = Crypt K M;
   M ∈ targetmsgs; K ∉ range priK;
   NRO = Crypt (priK A) {|Number f_nro, Agent B, Nonce L, C|} |]
⟹ Says A B {|Number f_nro, Agent B, Nonce L, C, NRO|} # evs1 ∈ zg
```

ZG2

```
[| evs2 ∈ zg; C ∈ targetmsgs;
   Gets B {|Number f_nro, Agent B, Nonce L, C, NRO|} ∈ set evs2;
   NRO = Crypt (priK A) {|Number f_nro, Agent B, Nonce L, C|};
   NRR = Crypt (priK B) {|Number f_nrr, Agent A, Nonce L, C|} |]
⟹ Says B A {|Number f_nrr, Agent A, Nonce L, NRR|} # evs2  ∈  zg
```

ZG3

```
[| evs3 ∈ zg; C = Crypt K M;
   Says A B {|Number f_nro, Agent B, Nonce L, C, NRO|} ∈ set evs3;
   Gets A {|Number f_nrr, Agent A, Nonce L, NRR|} ∈ set evs3;
   sub_K = Crypt (priK A) {|Number f_sub, Agent B, Nonce L, Key K|} |]
⟹ Says A TTP {|Number f_sub, Agent B, Nonce L, Key K, sub_K|}
      # evs3 ∈ zg
```

TTP_prepare_ftp

```
[| evsT ∈ zg;
   Gets TTP {|Number f_sub, Agent B, Nonce L, Key K, sub_K|} ∈ set evsT;
   sub_K = Crypt (priK A) {|Number f_sub, Agent B, Nonce L, Key K|};
   con_K = Crypt (priK TTP) {|Number f_con, Agent A, Agent B,
                              Nonce L, Key K|} |]
⟹ Notes TTP {|Number f_con, Agent A, Agent B, Nonce L, Key K, con_K|}
      # evsT ∈ zg
```

A_ftp

```
[| evsA ∈ zg;
   Notes TTP {|Number f_con, Agent A, Agent B, Nonce L, Key K, con_K|}
     ∈ set evsA |]
⟹ Notes A {|Number f_con, Agent A, Agent B, Nonce L, Key K, con_K|}
      # evsA ∈ zg
```

B_ftp

```
[| evsB ∈ zg;
   Notes TTP {|Number f_con, Agent A, Agent B, Nonce L, Key K, con_K|}
     ∈ set evsB |]
⟹ Notes B {|Number f_con, Agent A, Agent B, Nonce L, Key K, con_K|}
      # evsB ∈ zg
```

Reception

```
[| evsr ∈ zg; Says A B X ∈ set evsr |] ⟹ Gets B X # evsr ∈ zg
```

**Fig. 2.** Modelling the Fair Zhou-Gollmann Protocol

the conventions of the message reception model: each Gets event must follow a matching Says event, as established by rule Reception.

## 6 Verifying a Fair Non-Repudiation Protocol

For the sake of clarity, this section discusses the guarantees proved of the Zhou-Gollmann protocol in a model that allows no spy. The influence of the spy on these guarantees will be the topic of the next section.

### 6.1 Proving Validity of Evidence

**Guarantees for *B*.** Let us verify that, at the end of a session, $B$ holds sufficient evidence to refute a denial by $A$. We prove that, if $B$ holds *con_K*, *NRO* and all other atomic messages, then $A$ cannot deny having sent $M$.

According to the general strategy (§3), we establish that the only way for $B$ to get hold of *con_K* is via *ftp*, namely completing the protocol, as stated by Theorem 1.

**Theorem 1.**

```
[| evs ∈ zg; con_K ∈ parts (knows B evs);
   con_K = Crypt (priK TTP) {|Number f_con, Agent A, Agent B,
                              Nonce L, Key K|} |]
⟹ Notes B {|Number f_con, Agent A, Agent B, Nonce L, Key K, con_K|}
     ∈ set evs
```

The proof is non-trivial in the Reception case, where Isabelle's simplifier leaves us with the possibility that $B$ knows *con_K* because he has received it from the network (rather than noted it). In this sub-case, someone must have sent it by a Says event, but we appeal to a lemma stating that nobody ever sends *con_K*.

Again following the general strategy, we can routinely prove Theorem 2 by induction, which states that if $B$ has *con_K* then $A$ indeed lodged $K$ with TTP, bound to label $L$. The proof initially deduces that TTP made *con_K* available, and then concludes that $A$ sent *sub_K*.

**Theorem 2.**

```
[| evs ∈ zg;
   Notes B {|Number f_con, Agent A, Agent B, Nonce L, Key K, con_K|}
     ∈ set evs;
   con_K = Crypt (priK TTP) {|Number f_con, Agent A, Agent B,
                              Nonce L, Key K|};
   sub_K = Crypt (priK A) {|Number f_sub, Agent B, Nonce L, Key K|} |]
⟹ Says A TTP {|Number f_sub, Agent B, Nonce L, Key K, sub_K|} ∈ set evs
```

Theorems 1 and 2 together state that, if $B$ has *con_K*, then $A$ certainly sent *sub_K*, binding the key $K$ to the label $L$. However, some extra evidence is needed to $B$ to refute a denial from $A$. The evidence is *NRO*, which (by Theorem 3) $B$ holds only if $A$ sent it to him.

**Theorem 3.**

```
[| evs ∈ zg; NRO ∈ parts (knows B evs);
   NRO = Crypt (priK A) {|Number f_nro, Agent B, Nonce L, C|} |]
⟹ Says A B {|Number f_nro, Agent B, Nonce L, C, NRO|} ∈ set evs
```

Proving this theorem requires a lemma concluding that $B$ could only receive $NRO$ inside the first message of the protocol, namely that the event

$$\mathsf{Gets}\ B\ \{\!|\mathsf{Number}\ \mathsf{f\_nro}, \mathsf{Agent}\ B, \mathsf{Nonce}\ L, C, NRO|\!\}\} \tag{1}$$

occurred. In the Reception case, the simplifier tries to establish whether $B$ might learn $NRO$ by receiving a message of any form, but another lemma states that this is impossible. Event (1) implies that $NRO$ is in the network traffic, and then an authenticity theorem derives that it certainly originated with $A$, thus concluding the proof. Theorem 3 states that a judge may safely conclude from $B$'s presenting $NRO$ that $A$ sent it, binding the ciphertext $C$ to the label $L$.

These theorems show that $C$ is bound to the key $K$ via the label $L$. Hence, if $B$ presents $NRO$, $con\_K$, $C$, $L$ and $K$ to a judge, then the judge can conclude that $A$ sent $B$ the message $M$ that is obtained decrypting $C$ by $K$.

**Guarantees for *A*.** Analogous theorems justify $A$'s evidence that $B$ received the plaintext, $M$. In particular, Theorem 1 can be proved analogously for $A$, stating that, when $con\_K$ is known to $A$, then $A$ certainly noted it. On this assumption, we easily prove that TTP made $con\_K$ publicly available. Combining these two results, we get Theorem 4.

**Theorem 4.**

```
[| evs ∈ zg; con_K ∈ parts (knows A evs);
   con_K = Crypt (priK TTP) {|Number f_con, Agent A, Agent B,
                              Nonce L, Key K|};
⟹ Notes TTP {|Number f_con, Agent A, Agent B, Nonce L, Key K, con_K|}
     ∈ set evs
```

Following this theorem, when $A$ presents $con\_K$ to a judge, she also proves that she has bound the key $K$ to $L$ via TTP. Theorem 5 also states that, if $A$ has $NRR$, then $B$ certainly received $NRO$ confirming that $A$ associated the ciphertext $C$ to the label $L$.

**Theorem 5.**

```
[| evs ∈ zg; NRR ∈ parts (knows A evs);
   NRR = Crypt (priK B) {|Number f_nrr, Agent A, Nonce L, C|};
   NRO = Crypt (priK A) {|Number f_nro, Agent B, Nonce L, C|} |]
⟹ Gets B {|Number f_nro, Agent B, Nonce L, C, NRO|} ∈ set evs
```

The proof resembles that of Theorem 3 to derive that $B$ sent $NRR$ inside an instance of message 2. This, in turn, requires an appeal to an authenticity theorem: once $NRR$ is in the traffic, then it certainly originated with $B$. A subsidiary lemma stating that $B$ only sends $NRR$ upon reception of $NRO$ concludes.

Theorems 4 and 5 guarantee to a judge that, if $A$ presents $NRR$, $con\_K$, $C$, $L$ and $K$, then $B$ can decrypt $C$ using the key $K$, which was available with $con\_K$

via *ftp*. It is up to $B$ to get *con_K*, so this may appear a weaker guarantee than the corresponding one for $B$. However, the protocol authors observe that $B$ is interested in getting *con_K* in order to win a dispute over $A$, as confirmed by Theorems 1 and 2.

### 6.2 Proving Fairness

**Guarantees for *B*.** Theorem 4 may be read as a guarantee of fairness for $B$ because it says that, should *con_K* be known to $A$, then $B$ too would be able to obtain it via *ftp* from TTP's public directory. Similarly, Theorem 5 guarantees to $B$ that, in case $A$ knows *NRR*, then $B$ has received the corresponding *NRO*.

Certainly TTP makes *con_K* available only in case it receives a valid instance of message 3. So, on the conclusion of Theorem 4, we can prove that the event

$$\mathsf{Gets}\ \mathsf{TTP}\ \{\!|\mathsf{Number}\,\mathsf{f\_sub}, \mathsf{Agent}\,B, \mathsf{Nonce}\,L, \mathsf{Key}\,K, \mathit{sub\_K}|\!\}$$

occurred. This implies that *sub_K* is in the traffic and therefore, via a suitable authenticity theorem for *sub_K*, that the event

$$\mathsf{Says}\,A\ \mathsf{TTP}\ \{\!|\mathsf{Number}\,\mathsf{f\_sub}, \mathsf{Agent}\,B, \mathsf{Nonce}\,L, \mathsf{Key}\,K, \mathit{sub\_K}|\!\}$$

also occurred. We can now conclude that $A$ received a valid instance of message 2, thus learning *NRR*; this verifies the main condition of Theorem 5. The reasoning above is the proof of Theorem 6, which states that $A$'s knowledge of *con_K* enables $B$ to retrieve the same certificate from TTP and guarantees $B$ to have received *NRO*. The ciphertext $C$ being existentially quantified does not weaken the theorem because $C$ is also bound to label $L$ by the message structure.

**Theorem 6.**

```
[| evs ∈ zg; con_K ∈ parts (knows A evs);
   con_K = Crypt (priK TTP) {|Number f_con, Agent A, Agent B,
                              Nonce L, Key K|} |]
⟹ Notes TTP {|Number f_con, Agent A, Agent B, Nonce L, Key K, con_K|}
     ∈ set evs ∧
   (EX NRO C.
       Gets B {|Number f_nro, Agent B, Nonce L, C, NRO|} ∈ set evs ∧
       NRO = Crypt (priK A) {|Number f_nro, Agent B, Nonce L, C|})
```

**Guarantees for *A*.** If $B$ holds *con_K*, and it names $A$ as the originator, then *con_K* is available to $A$ too, who has also received *NRR*. Theorem 7 guarantees this.

**Theorem 7.**

```
[| evs ∈ zg; con_K ∈ parts (knows B evs);
   con_K = Crypt (priK TTP) {|Number f_con, Agent A, Agent B,
                              Nonce L, Key K|};
   NRR = Crypt (priK B) {|Number f_nrr, Agent A, Nonce L, C|} |]
⟹ Notes TTP {|Number f_con, Agent A, Agent B, Nonce L, Key K, con_K|}
     ∈ set evs ∧
    Gets A {|Number f_nrr, Agent A, Nonce L, NRR|} ∈ set evs
```

The first part of the conclusion derives from proving Theorem 4 on the assumption that *con_K* is known to $B$. The second part derives from an appeal to Theorem 2 and a lemma saying that $A$ only sends message 3 upon reception of message 2.

Note that there is no analogue of Theorem 5 for $A$: $B$'s possession of *NRO* does not imply $A$'s possession of *NRR*. Although this suggests that, upon reception of *NRO*, $B$ has an advantage over $A$, who holds no evidence, our theorems on validity of the evidence held by $B$ indicate that $B$ cannot win any disputes until he also gets *con_K*. Theorem 7 concludes that, at that stage, $A$ will hold equivalent evidence.

## 7 Modelling and Verifying with a Spy

This section discusses how the presence of the spy influences the protocol goals. The protocol definition must be extended by the rule Fake given in Fig. 3. That rule allows the spy to send any of the messages that can be built from the analysis of the traffic using the private signature keys of bad agents [12]. Note that the spy also sees the messages that bad agents retrieve via *ftp*, for those events were modelled in terms of Notes (§2, §5). By contrast, the spy cannot exploit TTP's creation of *con_K* because TTP is assumed not to be bad.

Fake
```
[| evsF ∈ zg;  X ∈ synth (analz (knows Spy evsF)) |]
⟹ Says Spy B X # evsF ∈ zg
```

**Fig. 3.** Modelling a Spy

The rest of the protocol definition requires minor changes. Rule ZG3 must check that *NRR* truly is $B$'s signature on the expected components, because the spy might have replaced it with a fake signature. Rule TTP_prepare_ftp must check that $K$ is a symmetric key, as the spy might have inserted some bad agent's private key. This attack could not take place in the real world, since private keys are asymmetric, with a typical length of 1024 bits, while a symmetric key is typically no longer that 128 bits. So it is realistic to assume that TTP can reject such substitutions.

Our strategies to proving the non-repudiation goals work as before. However, when proving validity of evidence, the exact form of the message whereby an agent learnt a certificate cannot be stated. The spy could have prevented the delivery of the legal message containing the certificate, extracted the certificate and then forwarded it inside a fake message of unpredictable form.

So, the theorems presented above receive minor variations. For example, Theorem 1 now has the form of Theorem $1'$, which shows that an agent other than the spy who knows *con_K* has either got it via *ftp* or received it from the network inside some larger message.

**Theorem 1′.**

```
[| evs ∈ zg; con_K ∈ parts (knows B evs); B ≠ Spy;
   con_K = Crypt (priK TTP) {|Number f_con, Agent A, Agent B,
                              Nonce L, Key K|} |]
⟹ Notes B {|Number f_con, Agent A, Agent B, Nonce L, Key K, con_K|}
     ∈ set evs ∨
   Gets B X ∈ set evs ∧ con_K ∈ parts {X}
```

What was Theorem 2 can now be proved on the conclusion of Theorem 1′ via the observation that *con_K* was certainly in the traffic, and therefore originated with TTP. In consequence, combining the two new theorems produces the same guarantee as before, but only for an agent who is not the spy.

Other minor changes concern the authenticity theorems that have been mentioned along the treatment. For example, *NRO* is encrypted by $A$'s private key, so proving that it originated with $A$ requires assuming that $A$ does not belong to the set bad, otherwise the spy would know $A$'s private signature key priK $A$ and could forge the certificate. The same extra condition is needed on $A$ when proving that *sub_K* originated with $A$, or on $B$ when proving that *NRR* originated with $B$. In consequence, for Theorem 3 to continue to hold, $B$ must not be bad. Likewise, Theorem 5 now needs $A$ not to be bad, and can only state that $B$ gets *NRO* inside a message of some form. Theorem 4 remains unchanged.

The fairness theorems do not need extra assumptions because they rest on the message signed by TTP, which cannot be forged. However, their conclusions cannot state the exact form of the messages that deliver *NRO* and *NRR* respectively. Those messages are now existentially quantified.

Following these considerations, it seems fair to conclude that the Zhou-Gollmann protocol achieves its goals even in the presence of a spy who is allowed to monitor the traffic and to exploit bad agent's private signature keys.

## 8 Related Work

Schneider was the first to analyse the Zhou-Gollmann protocol formally [16]. He uses the theory of Communicating Sequential Processes (*CSP*) to extend an existing model previously used for authentication protocols with extra channels whereby the peers present their evidence to a judge. His excellent account on validity of evidence and fairness is carried out by pen and paper. The proof strategies are significantly different from ours. Events are enforced by proving that they do not belong to a CSP refusal set. He writes,

> 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 [15, §5]

Schneider allows any agent to send messages of any form using components from the agent's knowledge, but obviously prevents this when proving fairness. By contrast, in our model, all agents must send messages of the form that the

protocol prescribes and can arbitrarily decide to quit the protocol; but we have also considered the influence of a powerful spy. This difference between the two models is superficial: it would be trivial to define a set of unfair agents and allow each agent $A$ of the set to send messages from $\mathsf{synth}(\mathsf{analz}(\mathsf{knows}\, A\ evs))$ on any trace $evs$.

Zhou and Gollmann analyse their protocol using a *belief logic* [17]. The approach allows for simple proofs on validity of evidence that only require four axioms and two inference rules. They formalise a judge $J$ and reach the conclusions that, at the end of a session between agents $A$ and $B$, the following predicates hold:

- $J$ *believes* ($A$ *said* $M$)
- $J$ *believes* ($B$ *received* $M$)

These predicates do not highlight what evidence the peers have to present to convince the judge, but the protocol verifier could understand this from following the proofs more closely. Fairness properties are not considered, and the difficulty in tackling them appears to be a limitation of the approach. The philosophical differences between reasoning on beliefs and reasoning on traces of events are well known. However, it may be interesting to note that also these proofs on validity of evidence closely follow the events of the session. The protocol in fact requires each agent to send certain evidence only upon reception of some other, specific evidence.

Both works discussed here model a judge explicitly. We have chosen not to model a judge because his functioning as well as the peers' interaction with him are external to the non-repudiation protocol. A peer's presenting some evidence to the judge in fact implies that the peer holds the evidence, which our function $\mathsf{knows}$ concisely expresses. A desirable consequence is that the current Inductive Approach can be used *with no extensions* to verify non-repudiation goals.

## 9 Conclusions

A non-repudiation protocol differs from the protocols traditionally studied in that the protocol participants do not trust each other. Every agent is a potential enemy. This change affects our models and our theorems slightly, but not drastically. We plan to enrich the model further, to model more precisely an agent who is trying to subvert the protocol. (The spy behaves too arbitrarily; for instance, he might give away his private keys, as no real villain would). To the best of our knowledge, no one else has proved non-repudiation properties using verification tools.

## Acknowledgements

This work was funded by the EPSRC grant GR/R01156/01 *Verifying Electronic Commerce Protocols*.

## References


1. G. Bella. Message Reception in the Inductive Approach. Research Report 460, University of Cambridge — Computer Laboratory, 1999.
2. G. Bella. Modelling Agents' Knowledge Inductively. In *Proc. of the 7th International Workshop on Security Protocols*, LNCS 1796. Springer-Verlag, 1999.
3. G. Bella. Mechanising a protocol for smart cards. In *Proc. of International Conference on Research in Smart Cards (e-Smart'01)*, LNCS. Springer-Verlag, 2001. In Press.
4. G. Bella, F. Massacci, L. C. Paulson, and P. Tramontano. Formal Verification of Cardholder Registration in SET. In F. Cuppens, Y. Deswarte, D. Gollmann, and M. Waidner, editors, *Proc. of the 6th European Symposium on Research in Computer Security (ESORICS 2000)*, LNCS 1895, pages 159–174. Springer-Verlag, 2000.
5. G. Bella and L. C. Paulson. Kerberos Version IV: Inductive Analysis of the Secrecy Goals. In J.-J. Quisquater, Y. Deswarte, C. Meadows, and D. Gollmann, editors, *Proc. of the 5th European Symposium on Research in Computer Security (ESORICS'98)*, LNCS 1485, pages 361–375. Springer-Verlag, 1998.
6. G. Bella and L. C. Paulson. Mechanising BAN Kerberos by the Inductive Method. In A. J. Hu and M. Y. Vardi, editors, *Proc. of the International Conference on Computer-Aided Verification (CAV'98)*, LNCS 1427, pages 416–427. Springer-Verlag, 1998.
7. M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest. A Fair Protocol for Signing Contracts. *IEEE Transactions on Information Theory*, 36(1):40–46, 1990.
8. C. A. Meadows. The NRL Protocol Analyzer: An Overview. *Journal of Logic Programming*, 26(2):113–131, 1996.
9. T. Okamoto and K. Ohta. How to Simultaneously Exchange Secrets by General Assumptions. In *Proc. of the 2nd ACM Conference on Computer and Communication Security (CCS'94)*, pages 184–192, 1994.
10. L. C. Paulson. *Isabelle: A Generic Theorem Prover*. LNCS 828. Springer-Verlag, 1994.
11. L. C. Paulson. *Theory for public-key protocols*, 1996. `http://www4.informatik.tu-muenchen.de/~isabelle/library/HOL/Auth/Public.html`.
12. L. C. Paulson. The Inductive Approach to Verifying Cryptographic Protocols. *Journal of Computer Security*, 6:85–128, 1998.
13. L. C. Paulson. Inductive Analysis of the Internet protocol TLS. *ACM Transactions on Computer and System Security*, 1999. In press.
14. P. Y. A. Ryan and S. A. Schneider. *The Modelling and Analysis of Security Protocols: the CSP Approach*. Addison-Wesley, 2000.
15. S. Schneider. Verifying Authentication Protocols with CSP. In *Proc. of the 10th IEEE Computer Security Foundations Workshop*, pages 3–17. IEEE Computer Society Press, 1997.
16. S. Schneider. Formal Analysis of a Non-Repudiation Protocol. In *Proc. of the 11th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 1998.
17. G. Zhou and D. Gollmann. Towards Verification of Non-Repudiation Protocols. In *Proc. of the 1998 International Refinement Workshop and Formal Methods Pacific*, pages 370–380. Springer-Verlag, 1998.

18. J. Zhou and D. Gollmann. A Fair Non-Repudiation Protocol. In *Proc. of the 15th IEEE Symposium on Security and Privacy*, pages 55–61. IEEE Computer Society Press, 1996.
19. J. Zhou and D. Gollmann. An Efficient Non-Repudiation Protocol. In *Proc. of the 10th IEEE Computer Security Foundations Workshop*, pages 126–132. IEEE Computer Society Press, 1996.