Automatic Methods for Analyzing Non-repudiation Protocole with an Active Intruder

Francis Klay, Laurent Vigneron

Automatic Methods for Analyzing Non-Repudiation Protocols with an Active Intruder

Francis Klay[1] and Laurent Vigneron[2] *

[1] France Telecom R&D, Lannion, France — francis.klay@orange-ftgroup.com
[2] LORIA - Nancy Université, Vandoeuvre-lès-Nancy, France — laurent.vigneron@loria.fr

Abstract. Non-repudiation protocols have an important role in many areas where secured transactions with proofs of participation are necessary. Formal methods are clever and without error, therefore using them for verifying such protocols is crucial. In this purpose, we show how to partially represent non-repudiation as a combination of authentications on the Fair Zhou-Gollmann protocol. After discussing the limitations of this method, we define a new one, based on the handling of the knowledge of protocol participants. This second method is general and of natural use, as it consists in adding simple annotations in the protocol specification. It is very easy to implement in tools able to handle participants knowledge. We have implemented it in the AVISPA Tool and analyzed the Fair Zhou-Gollmann protocol and the optimistic Cederquist-Corin-Dashti protocol, discovering attacks in each. This extension of the AVISPA Tool for handling non-repudiation opens a highway to the specification of many other properties, without any more change in the tool itself.
Keywords: cryptographic protocols, non-repudiation, fairness, authentication, automatic analysis, AVISPA Tool.

1 Introduction

Authentication and secrecy properties of security protocols have been intensively studied for years [23], but the interest of other properties such as non-repudiation and fairness has been raised only in the 1990s with the explosion of Internet services and electronic transactions.[3]

Non-repudiation protocols are designed for verifying that, when two parties exchange information over a network, neither one nor the other can deny having participated to this communication. Such a protocol must therefore generate evidences of participation to be used in case of a dispute. The basic tools for non-repudiation services have been digital signatures and public key cryptography. Indeed, a signed message is an evidence of participation and identity of the other party [14].

The majority of the non-repudiation property analysis efforts in the literature are manually driven though. One of the first efforts to apply formal methods to the verification of non-repudiation protocols has been presented by Zhou et al. in [31], where they have used SVO logic. In [25] Schneider uses process algebra CSP to prove the correctness of a non-repudiation protocol, the well-known Fair Zhou-Gollmann protocol. With the same goal, Bella et al. have used the theorem prover Isabelle [4]. Schneider has defined a rank function for encoding that in an execution trace, an event happens before another event. The verification is done by analyzing traces in the stable failures models of CSP. Among the automatic analysis attempts, we can cite Shmatikov

* This work is supported by the ANR AVOTÉ, http://www.lsv.ens-cachan.fr/anr-avote/.
[3] See [1] for a detailed list of publications related to the analysis of non-repudiation protocols.

and Mitchell [26] with Murφ, a finite state model-checker, to analyze a fair exchange and two contract signing protocols, Kremer and Raskin [15] with a game-based model, Armando et al. [3] using LTL for encoding resilient channels in particular, the work of Gürgens and Rudolph [9] based on the asynchronous product automata (APA) and the simple homomorphism verification tool (SHVT) [19], raising flaws in three variants of the Fair Zhou-Gollmann protocol and in two other optimistic fair non-repudiation protocols [13, 29]. Wei and Heather [27] have used FDR, with an approach similar to Schneider, for a variant of the Fair Zhou-Gollmann protocol with timestamps.

The common point between all those works is that they use rich logics, with a classical bad consequence for model checkers, the difficulty to consider large protocols. For avoiding this problem, Wei and Heather [28] have used PVS [22], but some of the proofs still had to be done by hand.

Fairness is a property that is more difficult to achieve: no party should be able to reach a point where he has the evidence or the message he requires, without the other party also having his required evidence. Fairness is not always required for non-repudiation protocols, but it is usually desirable.
A variety of protocols has been proposed in the literature to solve the problem of fair message exchange with non-repudiation. The first solutions were based on a gradual exchange of the expected information [14]. However this simultaneous secret exchange is troublesome for actual implementations because fairness is based on the assumption of equal computational power for both parties, which is very unlikely in a real world scenario. A possible solution to this problem is the use of a trusted third party (TTP), and in fact it has been shown that this is impossible to achieve fair exchange without a TTP [18, 20]. The TTP can be used as a delivery agent to provide simultaneous share of evidences. The Fair Zhou-Gollmann protocol [30] is a well known example using a TTP as a delivery agent; a significant amount of work has been done over this protocol and its derivations [4, 10, 21, 25, 31]. However, instead of passing the complete message through the TTP and thus creating a possible bottleneck, recent evolution of protocols resulted in efficient, *optimistic* versions, in which the TTP is only involved in case something goes wrong. Resolve and abort sub-protocols must guarantee that every party can complete the protocol in a fair manner and without waiting for actions of the other party.
One of these recent protocols is the optimistic Cederquist-Corin-Dashti (CCD) non-repudiation protocol [6]. The CCD protocol has the advantage of not using session labels, unlike many others in the literature [14, 17, 30, 25]. A session label typically consists of a hash of all message components. Gürgens et al. [10] have shown a number of vulnerabilities associated to the use of session labels and, to our knowledge, the CCD protocol is the only optimistic non-repudiation protocol that avoids altogether the use of session labels.

This paper presents a method for automatically verifying non-repudiation protocols in presence of an active intruder. Our method has been implemented in the AVISPA Tool [2][4] and we illustrate it with examples. This tool, intensively used for defining Internet security protocols and automatically analyzing their authentication and secrecy properties, did not provide any help for considering non-repudiation properties.
We first consider non-repudiation analysis as a combination of authentication problems, applied to the Fair Zhou-Gollmann protocol. We show the limitations of this representation and the difficulties for proving non-repudiation properties using only authentications. Then, we define a method based on the analysis of agents knowledge, permitting to handle non-repudiation and fairness properties in a uniform framework. Our approach allows one to specify the logical prop-

[4] `http://www.avispa-project.org`

erties in a natural way: they correspond to state invariants that are convincing properties for the user. This method is easy to integrate in lazy verification systems, such as the AVISPA Tool, and can also be integrated in any system able to handle agents (or intruder) knowledge. This should permit, contrarily to more complex logics like LTL, to set up abstractions more easily for considering unbounded cases. This should also permit to get a more efficient verification for bounded cases. We illustrate this fact with the analysis of the optimistic Cederquist-Corin-Dashti protocol.
In this paper, the defined techniques are based on the formal semantics presented in [7, 8] for the AVISPA Tool.

2 Non-Repudiation Properties

Non-repudiation (NR) is a general property that is usually not clearly defined. It is described by protocols designers as a set of required services, depending on the protocol and the required security level. In particular, non-repudiation properties may differ whether a trusted third party (TTP) is used or not in the protocol.

In the following, we recall the classical model independent definitions of non-repudiation services required by most of the existing security applications (for e-commerce for example). All these services are defined for a message sent by an originator agent to a recipient agent, possibly via a delivery agent, a TTP.

Definition 1. *The service of* ***non-repudiation of origin****, denoted* $NRO_B(A)$*, provides the recipient* B *with a set of evidences which ensures that the originator* A *has sent the message. The evidence of origin is generated by the originator and held by the recipient. This property protects the recipient against a dishonest originator.*

Definition 2. *The service of* ***non-repudiation of receipt****, denoted* $NRR_A(B)$*, provides the originator* A *a set of evidences which ensures that the recipient* B *has received the message. The evidence of receipt is generated by the recipient and held by the originator. This property protects the originator against a dishonest recipient.*

Definition 3. *The service of* ***non-repudiation of submission****, denoted* $NRS_A(B)$*, provides the originator* A *a set of evidences which ensures that he has submitted the message for delivery to* B*. This service only applies when the protocol uses a TTP. Evidence of submission is generated by the delivery agent, and will be held by the originator. This property protects the originator against a dishonest recipient.*

Definition 4. *The service of* ***non-repudiation of delivery****, denoted* $NRD_A(B)$*, provides the originator* A *a set of evidences which ensures that the recipient* B *has received the message. This service only applies when the protocol uses a TTP. Evidence of delivery is generated by the delivery agent, and will be held by the originator. This property protects the originator against a dishonest recipient.*

Definition 5. *A service of* ***fairness*** *(also called* strong fairness*) for a non-repudiation protocol provides evidences that, at the end of the protocol execution, either the originator has the evidence of receipt of the message and the recipient has the evidence of origin of the corresponding message, or none of them has any valuable information. This property protects the originator and the recipient.*

Definition 6. *A service of* ***timeliness*** *for a non-repudiation protocol guarantees that, whatever happens during the protocol run, all participants can reach a state that preserves fairness, in a finite time.*

Note that in general, sets of evidences such as $\mathcal{NRO}$, $\mathcal{NRR}$, $\mathcal{NRS}$ and $\mathcal{NRD}$ are composed with messages signed by an agent.

After this informal use of the notion of evidence, let us consider for the sequel of this paper the following definition.

Definition 7. *An **evidence** for an agent A and a non-repudiation property P is a message, a part of a message, or a combination of both, received by A that is necessary for guaranteeing property P.*

We will also consider the following definition of a valid service.

Definition 8. *A non-repudiation service is **valid** if is satisfies the corresponding property.*

Remark: In this paper, we consider the evidences given by the protocol designer as valid: without intervention of an intruder, those evidences are sufficient to guarantee the non-repudiation service; and in case of a dispute, a judge analyzing them will always be able to protect honest agents. Thus, we suppose that evidences are correctly chosen, so that a judge can use them for building proofs protecting honest agents.

3 Non-Repudiation as Authentication

It is well known that non-repudiation is a form of authentication [23]. In this section we use the Fair Zhou-Gollmann protocol to demonstrate that properties like NRO, NRR,... can be at least partially represented by authentication properties. However we show some strong limitations of this approach, motivating the introduction of a new approach in the next section.

3.1 Running Example: the FairZG Protocol

In this section we describe the Fair Zhou-Gollmann protocol (FairZG) [31], a fair non-repudiation protocol that uses a TTP. We have chosen this protocol as a case study to demonstrate our analysis approach because of the existence of significant related work [4, 10, 21, 25]. The protocol is presented below in Alice&Bob notation, where fNRO, fNRR, fSUB and fCON are labels used to identify the purpose of messages.

1. A → B: fNRO.B.L.C.NRO
2. B → A: fNRR.A.L.NRR
3. A → TTP: fSUB.B.L.K.SubK
4. B ↔ TTP: fCON.A.B.L.K.ConK
5. A ↔ TTP: fCON.A.B.L.K.ConK

where A (for Alice) is the originator of the message M, B (for Bob) is the recipient of the message M, TTP is the trusted third party, M is the message to be sent from Alice to Bob, C is a commitment (the message M encrypted by a key K), L is a unique session identifier (also called label), K is a symmetric key defined by Alice, NRO is a message used for non-repudiation of origin (the message fNRO.B.L.C signed by Alice), NRR is a message used for non-repudiation of receipt (the message fNRR.A.L.C signed by Bob), SubK is a proof of submission of K (the message fSUB.B.L.K signed by Alice), ConK is a confirmation of K (the message fCON.A.B.L.K signed by the TTP).

Non-repudiation properties of origin and receipt are defined by the protocol designers by the following sets of terms:

$\mathcal{NRO}_B(A) = \{\mathsf{NRO}, \mathsf{ConK}\}$
$\mathcal{NRR}_A(B) = \{\mathsf{NRR}, \mathsf{ConK}\}$

The main idea of this FairZG protocol is to split the delivery of a message into two parts. First a commitment C, containing the message M encrypted by a key K, is exchanged between Alice and Bob (message fNRO). Once Alice has an evidence of commitment from Bob (message fNRR), the key K is sent to a trusted third party (message fSUB). Once the TTP has received the key, both Alice and Bob can retrieve the evidence ConK and the key K from the TTP (messages fCON). This last step is represented by a double direction arrow in the Alice&Bob notation because it is implementation specific and may be composed by several message exchanges between the agents and the TTP. In this scenario we assume that the network will not be down forever and both Alice and Bob have access to the TTP's shared repository where it stores the evidences and the key. This means that the agents will be able to retrieve the key and evidences from the TTP even in case of network failures.

3.2 Non-Repudiation of Origin as Authentication

In our example, the FairZG protocol, non-repudiation of origin should provide the guarantee that if Bob owns $\mathcal{NRO}$ then Alice has sent M to Bob. Proposition 1 shows how this can be partially ensured with a set of authentications.

Definition 9. *auth(X,Y,D) is the non injective authentication, and means agent X authenticates agent Y on data D.*

The semantics of such a predicate is standard and can be found in [16]. The next two lemmas present standard properties of authentication.

Lemma 1 (Subterm property). *Given agents A and B, and message M, if auth(A,B,M), then for each subterm s of M, accessible by composition/decomposition of M by both agents, auth(A,B,s) is true.*

Lemma 2 (Transitivity of authentication). *Given agents A, B and C, and message M, if auth(A,B,M) and auth(B,C,M), then auth(A,C,M).*

Proposition 1. *Given the FairZG protocol, if auth(B,A,NRO), auth(B,TTP,ConK) and auth(TTP,A,SubK) are valid, then the non-repudiation service of origin $NRO_B(A)$ is valid.*

Proof. For the two evidences of $\mathcal{NRO}_B(A) = \{\mathsf{NRO}, \mathsf{ConK}\}$, we have:

- $\mathsf{NRO} = \mathsf{Sig_A}(\mathsf{fNRO.B.L.}\{\mathsf{M}\}_\mathsf{K})$: since auth(B,A,NRO) is valid, there is an agreement between B and A on $\mathsf{Sig_A}(\mathsf{fNRO.B.L.C})$. From the subterm property, this also means an agreement on $\{\mathsf{M}\}_\mathsf{K}$, thus A has sent the $\{\mathsf{M}\}_\mathsf{K}$ that B holds.
- $\mathsf{ConK} = \mathsf{Sig_{TTP}}(\mathsf{fCON.A.B.L.K})$: as above auth(B,TTP,ConK) implies an agreement on K between B and TTP. Furthermore $\mathsf{SubK} = \mathsf{Sig_A}(\mathsf{fSUB}, \mathsf{B}, \mathsf{L}, \mathsf{K})$, thus auth(TTP,A,SubK) implies an agreement on K between TTP and A. By transitivity we have an agreement on K between B and A which means that A has sent K to TTP, that same K that B got from TTP.

As A has sent $\{\mathsf{M}\}_\mathsf{K}$ and K, it means that he has generated M and run the protocol in order to transmit it to B.
Non-injective authentication is only required for auth(B,TTP,ConK) because B can ask many times ConK. However since all authentications imply an agreement on the unique session identifier L, this protects from authentication across different sessions. □

3.3 Non-Repudiation of Receipt as Authentication

In our example, the FairZG protocol, non-repudiation of receipt should provide the guarantee that if Alice owns $\mathcal{NRR}$ then Bob has received M from Alice. Proposition 2 shows how this can be partially done with a set of authentications.

Proposition 2. *Given the FairZG protocol, if* auth(A,B,NRR), auth(A,TTP,ConK) *and* auth(B,TTP,ConK) *are valid, then the non-repudiation service of receipt* $NRR_A(B)$ *is valid.*

Proof. For the two evidences of $\mathcal{NRR}_A(B) = \{\mathsf{NRR}, \mathsf{ConK}\}$, we have:

- $\mathsf{NRR} = \mathsf{Sig_B}(\mathsf{fNRR.A.L.}\{\mathsf{M}\}_\mathsf{K})$: a reasoning as for NRO in Proposition 1 ensures that B has received $\{\mathsf{M}\}_\mathsf{K}$.
- $\mathsf{ConK} = \mathsf{Sig_{TTP}}(\mathsf{fCON.A.B.L.K})$: auth(A,TTP,ConK) implies an agreement on K between A and TTP. Furthermore auth(B,TTP,ConK) implies an agreement on K between B and TTP. This means that there is an agreement on K between A and B, thus when A holds ConK, B has received or will be able to receive K.

The proof end is similar to the one of Proposition 1. □

3.4 Limitations and Difficulties

We have just illustrated on the FairZG protocol how to represent some non-repudiation properties using authentication. This shows that non-repudiation can be handled by most existing protocol analyzers, as most of them can handle authentication.

However, this only permits to partially handle non-repudiation:

1. The main problem is to apply Propositions 1 and 2 in automatic tools, since the authentication property is usually encoded by an annotation pair (for example "witness"/"request" in AVISPA). In such a situation we cannot handle dishonest agents since for example with the $NRO_B(A)$ service, a dishonest Bob could forge a fake evidences set without executing the "request" annotation. In such a case there is no authentication failure but the service is not valid.
   More generally dishonest agents can always act so that authentications in which they are involved fail or not, by generating wrong authentication "requests", or wrong "witnesses". This is the reason why tools like AVISPA do not handle authentications involving the intruder. This is also why with our representation of non-repudiation, the AVISPA tool does not find any error in the FairZG protocol, while this is possible to prove that the protocol is not fair when agent A is dishonest [9] (see Section 4.4 for details of this attack).
   In order to avoid this kind of problems we need to prove that Bob could only own $\mathcal{NRO}$ if Alice has actually sent the correct protocol messages. This may be done as for example in [25], [27] or [10] but this is not trivial.
2. Another problem with the handling of non-repudiation as authentications is that it is difficult to apply to optimistic non-repudiation protocols that include sub-protocols like *abort* and *resolve* as presented in the next section. One of the main difficulties is that such protocols are non-deterministic.

As a conclusion, proving non-repudiation with the help of authentications does not seem to be the best way; this is why in the next section we propose another simple and complete approach for handling non-repudiation.

4 Non-Repudiation based on Agent Knowledge

In this section, we present a new method for considering non-repudiation services and fairness in a uniform framework: we introduce a logic permitting to describe states invariants. This logic is a very classical one, except that we define two new predicates, `deduce` and `aknows` that permit to consider agents knowledge in the description of goals. The `aknows` predicate is also used as a protocol annotation, with the following semantics: *agent X knows (or can deduce) term t.*

All our work is based on the standard formal semantics described in [7, 8] for the AVISPA Tool.

4.1 Description of Non-Repudiation Properties

The main role of a non-repudiation protocol is to give evidences of non-repudiation to the parties involved in the protocol. To analyze this kind of protocol, one must verify which participants have their non-repudiation evidences at the end of the protocol execution. For example, if the originator has all its evidences for non-repudiation of receipt, then the service of non-repudiation of receipt is guaranteed. If the recipient has all its evidences for non-repudiation of origin, then the service of non-repudiation of origin is guaranteed. If both parties (or none of them) have their evidences, fairness is guaranteed. In other words, to analyze non-repudiation, we need to verify if a set of terms is known by an agent at the end of the protocol execution.

And for considering a large class of non-repudiation protocols, we shall not restrict evidences to a set of terms, but we have to consider them as a combination of terms using standard logical connectors (conjunction, disjunction, negation).

For considering non-repudiation and fairness properties involving honest and dishonest agents, we have defined a new predicate that permits to access the knowledge of protocol participants. This predicate, named `aknows` (for *agent knows*), is used in protocols specifications for annotating transitions and for defining properties.

Definition 10 ($\mathcal{NR}_{\_X}(Y)$). *Let $\mathcal{A}$ be a set of agents playing a finite number of sessions of a protocol, $\mathcal{T}$ a set of terms sent in the messages of this protocol and $\mathcal{E}$ the subset of terms in $\mathcal{T}$ that are part of the evidences of non-repudiation in the protocol. For agents $X, Y \in \mathcal{A}$, $\mathcal{NR}_{\_X}(Y)$ is a logical combination of terms $t \in \mathcal{E}$ that constitute the evidence for a service of non-repudiation $NR_{\_}$ for agent X wrt. agent Y.*

Definition 11 (`aknows`). *Let $\mathcal{A}$ be a set of agents playing a finite number of sessions of a protocol, $\mathcal{P}$ the set of processes (ie. instances of protocol roles) involved in those sessions, and $\mathcal{T}$ a set of terms. The protocol annotation $\texttt{aknows}(X, p, t)$ is a predicate with $X \in \mathcal{A}$, $p \in \mathcal{P}$ and $t \in \mathcal{T}$, asserting that agent X, playing a role of the protocol as process p, knows (or can deduce) the term t.*

The semantics of predicate $\texttt{aknows}(X, p, t)$ is that the term t can be composed by agent X, according to its current knowledge in process p of the protocol, whether this agent is honest or not. This composability test can be easily done by any tool that is able to manage agents knowledge or intruder knowledge.

By abuse of notation, we may write $\texttt{aknows}(X, p, L)$, for a logical formula L combining evidences ($\mathcal{NR}_{\_X}(Y)$ for example), considering that the predicate `aknows` is an homomorphism:

$$\begin{aligned}
\texttt{aknows}(X, p, L_1 \wedge L_2) &= \texttt{aknows}(X, p, L_1) \wedge \texttt{aknows}(X, p, L_2) \\
\texttt{aknows}(X, p, L_1 \vee L_2) &= \texttt{aknows}(X, p, L_1) \vee \texttt{aknows}(X, p, L_2) \\
\texttt{aknows}(X, p, \neg L) &= \neg\texttt{aknows}(X, p, L)
\end{aligned}$$

Definition 12 (`deduce`). *Let $\mathcal{A}$ be a set of agents playing a finite number of sessions of a protocol and $\mathcal{T}$ a set of terms. We define* $\texttt{deduce}(X, t)$*, with $X \in \mathcal{A}$ and $t \in \mathcal{T}$, as the predicate which means that* `X` *can deduce* `t` *from its knowledge.*

We will use the same abuse of notation for `deduce` as for `aknows`.

The `aknows` predicate is used in protocol transitions for indicating that an agent knows an important information; it corresponds to a fact; it has the same meaning when used in the description of a property, but also indicates that protocol transitions have really been run.
The `deduce` predicate is used in properties description for indicating a deducible knowledge.
As a consequence, we can assume that each `aknows` annotation in protocols transitions corresponds to a valid `deduce` predicate on the same information; this assumption permits to avoid bad annotations.

Definition 13 (well-formedness). *The evidence $\mathcal{NR}_{\_X}(Y)$ is well-formed if it contains information that uniquely identifies processes of X and Y involved in the protocol session, and an injective function of the message M for which $NR_{\_}$ acts as a protection against a dishonest agent.*

We now give the results obtained by this representation.

Proposition 3. *Given a non-repudiation service of B against A about a message M with the well-formed evidence $\mathcal{NR}_{\_B}(A)$ for processes p_B and p_A of B and A respectively. If the following formulae are true at the end of process p_B of B, then the non-repudiation service is valid.*

$$\begin{aligned} &\texttt{aknows}(B, p_B, \mathcal{NR}_{\_B}(A)) \Rightarrow \texttt{aknows}(A, p_A, M) \\ &\texttt{deduce}(B, \mathcal{NR}_{\_B}(A)) \quad\;\; \Rightarrow \texttt{aknows}(B, p_B, \mathcal{NR}_{\_B}(A)) \end{aligned}$$

Proof. A sketch of proof is as follows: by the second implication if B is able to deduce $\mathcal{NR}_{\_B}(A)$ then $\texttt{aknows}(B, p_B, \mathcal{NR}_{\_B}(A))$ is included in its knowledge, since by well-formedness of $\mathcal{NR}_{\_B}(A)$, $\mathcal{NR}_{\_B}(A)$ and $\texttt{aknows}(B, p_B, \mathcal{NR}_{\_B}(A))$ are related to the same process p_B.

And again by well-formedness of $\mathcal{NR}_{\_B}(A)$, it includes all the information uniquely identifying M, thus the first implication implies an agreement on M between B and A. Finally as $\texttt{aknows}(A, p_A, M)$ is an annotation, this means that A has followed the protocol, thus he has done what he must do with M. □

Remark: Verifying formulae given in the above Proposition is not a problem, because a priori any theorem prover (able to consider secrecy) can compute whatever can be deduced by an agent at a given step of the protocol, especially concerning the `deduce` predicate [12].

Corollary 1. *Given a non-repudiation service of origin for B against A about message M, involving processes p_B and p_A of B and A respectively. If $\mathcal{NRO}_B(A))$ is well-formed and the following formulae are true at the end of process p_B, then the service is valid.*

$$\begin{aligned} &\texttt{aknows}(B, p_B, \mathcal{NRO}_B(A)) \Rightarrow \texttt{aknows}(A, p_A, M) \\ &\texttt{deduce}(B, \mathcal{NRO}_B(A)) \quad\;\; \Rightarrow \texttt{aknows}(B, p_B, \mathcal{NRO}_B(A)) \end{aligned}$$

Corollary 2. *Given a non-repudiation service of receipt for A against B about message M, involving processes p_A and p_B of A and B respectively. If $\mathcal{NRR}_A(B))$ is well-formed and the following formulae are true at the end of process p_A, then the service is valid.*

$$\begin{aligned} &\texttt{aknows}(A, p_A, \mathcal{NRR}_A(B)) \Rightarrow \texttt{aknows}(B, p_B, M) \\ &\texttt{deduce}(A, \mathcal{NRR}_A(B)) \quad\;\; \Rightarrow \texttt{aknows}(A, p_A, \mathcal{NRR}_A(B)) \end{aligned}$$

4.2 Description of Fairness

In the literature, authors often give different definitions of fairness for non-repudiation protocols. In some definitions none of the parties should have more evidences than the others at any given point in time. Others have a more flexible definition in which none of them should have more evidences than the others at the end of the protocol run. In many works it is also not very clear if only successful protocol runs are taken into account, or partial protocol runs are valid as well.

In this paper we consider the flexible definition of fairness, taking into account complete protocol runs. By complete protocol runs we mean a run where, even though the protocol could not have reached its last transition for all agents, there is no executable transition left, i.e. all possible protocol steps have been executed, but this does not mean that all agents are in a final state.

We define this standard notion of fairness as a function of non-repudiation of origin and of non-repudiation of receipt. If both properties, NRO and NRR, are ensured or both are not valid for a given message M, then we have fairness.

Proposition 4. *Given a protocol whose purpose is to send a message from Alice to Bob, we have the following equivalence concerning the standard definition of fairness for processes p_A and p_B of Alice and Bob respectively. If the non-repudiation is valid for the NRO and NRR services then:*

$$\mathit{Fairness} \equiv (\ \mathtt{aknows}(Bob, p_B, \mathcal{NRO}_{Bob}(Alice)) \textit{ iff } \mathtt{aknows}(Alice, p_A, \mathcal{NRR}_{Alice}(Bob))\ )$$

This result can be generalized to fairness wrt. a set of non-repudiation services as follows.

Theorem 1. *Given a protocol involving a finite number of agents, given a finite set of valid non-repudiation services NR, the protocol is fair wrt. NR iff*

$$\forall \mathcal{NRS}_{1X_1}(Y_1), \mathcal{NRS}_{2X_2}(Y_2) \in \mathcal{NR},$$
$$\mathtt{aknows}(X_1, p_1, \mathcal{NRS}_{1X_1}(Y_1)) \textit{ iff } \mathtt{aknows}(X_2, p_2, \mathcal{NRS}_{2X_2}(Y_2))$$

4.3 Running Example: CCD

For illustrating the analysis method described above, we use in this section a recent protocol, the Cederquist-Corin-Dashti (CCD) optimistic non-repudiation protocol [6]. The CCD protocol has been created for permitting an agent A to send a message M to an agent B in a fair manner. This means that agent A should get an evidence of receipt of M by B (EOR) if and only if B has really received M and the evidence of origin from A (EOO). EOR permits A to prove that B has received M, while EOO permits B to prove that M has been sent by A. The protocol is divided into three sub-protocols: the main protocol, an *abort* sub-protocol and a *resolve* sub-protocol.

The Main Protocol. It describes the sending of M by A to B and the exchange of evidences in the case where both agents can complete the entire protocol. If this direct communication cannot be completed, in order to finish properly the protocol, the agents execute the *abort* or the *resolve* sub-protocol with a trusted third party (TTP).

The main protocol is therefore composed of the following messages exchanges, described in the Alice&Bob notation:

1. $A \rightarrow B:\ \{M\}_K.EOO_M$ where $EOO_M = \{B.TTP.H(\{M\}_K).\{K.A\}_{Kttp}\}_{inv(Ka)}$
2. $B \rightarrow A:\ EOR_M$ where $EOR_M = \{EOO_M\}_{inv(Kb)}$
3. $A \rightarrow B:\ K$
4. $B \rightarrow A:\ EOR_K$ where $EOR_K = \{A.H(\{M\}_K).K\}_{inv(Kb)}$

where K is a symmetric key freshly generated by A, H is a one-way hash function, Kg is the public key of agent g and $inv(Kg)$ is the private key of agent g (used for signing messages). Note that we assume that all public keys are known by all agents (including dishonest agents).

In the first message, A sends the message M encrypted by K and the evidence of origin for B (message signed by A, so decryptable by B). In this evidence, B checks his identity, learns the name of the TTP, checks that the hash code is the result of hashing the first part of the message, but he cannot decrypt the last part of the evidence; this last part may be useful if any of the other sub-protocols is used.
B answers by sending the evidence of receipt for A, A checking that EOR_M is EOO_M signed by B.
In the third message, A sends the key K, permitting B to discover the plaintext message M.
Finally, B sends to A another evidence of receipt, permitting A to check that the symmetric key has been received by B.

The *Abort* Sub-Protocol. The *abort* sub-protocol is executed by agent A if he does not receive the message EOR_M at step 2 of the main protocol. The purpose of this sub-protocol is to cancel the messages exchange.

1. $A \rightarrow TTP : \{\texttt{abort}.H(\{M\}_K).B.\{K.A\}_{Kttp}\}_{inv(Ka)}$
2. $TTP \rightarrow A : \begin{cases} E_{TTP} & \text{where } E_{TTP} = \{A.B.K.H(\{M\}_K)\}_{inv(Kttp)} \\ & \text{if } \texttt{resolved}(A.B.K.H(\{M\}_K)) \\ AB_{TTP} & \text{where } AB_{TTP} = \{A.B.H(\{M\}_K).\{K.A\}_{Kttp}\}_{inv(Kttp)} \\ & \text{otherwise} \end{cases}$

In this sub-protocol, A sends to the TTP an abort request, containing the `abort` label and some information about the protocol session to be aborted.
According to what the TTP knows about this protocol session, he has two possible answers: if this is the first problem received by the TTP for this protocol session, the TTP sends a confirmation of abortion, AB_{TTP}, and stores in its database that this protocol session has been aborted; but if the TTP has already received a request for resolving this protocol session, he sends to A the information for completing his evidence of receipt by B, E_{TTP}.

The *Resolve* Sub-Protocol. The role of this second sub-protocol is to permit agents A and B to finish the protocol in a fair manner, if the main protocol cannot be run until its end by some of the parties. For example, if B does not get K or if A does not get EOR_K, they can invoke the *resolve* sub-protocol.

1. $G \rightarrow TTP : EOR_M$
2. $TTP \rightarrow G : \begin{cases} AB_{TTP} & \text{if } \texttt{aborted}(A.B.K.H(\{M\}_K)) \\ E_{TTP} & \text{otherwise} \end{cases}$

where G stands for A or B.

A resolve request is done by sending EOR_M to the TTP. If the protocol session has already been aborted, the TTP answers by the abortion confirmation, AB_{TTP}. If this is not the case, the TTP sends E_{TTP} so that the user could complete its evidence of receipt (if G is A) or of origin (if G is B). Then the TTP stores in its database that this protocol session has been resolved.

Agents' Evidences. For this protocol, according to [6], the logical expressions of evidences are:

$$\mathcal{NRO}_B(A) = \{M\}_K \wedge EOO_M \wedge K$$
$$\mathcal{NRR}_A(B) = \{M\}_K \wedge EOR_M \wedge (EOR_K \vee E_{TTP})$$

Note that there are two possibilities of evidences for non-repudiation of receipt, according to the way the protocol is run.

According to our method, we simply have to annotate protocol steps with `aknows` predicates, and then write the logical formula to be verified.

Non-repudiation of Origin. The following table shows where those annotations take place in the three CCD sub-protocols, for considering non-repudiation of origin.

$\mathcal{NRO}_B(A)$	Protocol - step
$\texttt{aknows}(B, p_B, \{M\}_K)$	Main - 1.
$\texttt{aknows}(B, p_B, EOO_M)$	Main - 1.
$\texttt{aknows}(B, p_B, K)$	Main - 3.
$\texttt{aknows}(B, p_B, K)$	Resolve - 2.

Note that the key K can be obtained either by the third message of the main protocol, or by the second message of the resolve sub-protocol. One annotation has to be put in each of those protocol steps.

By Corollary 1, **non-repudiation of origin** for the CCD protocol is represented by the following invariant formulae:

$$\texttt{aknows}(B, p_B, \{M\}_K \wedge EOO_M \wedge K) \Rightarrow \texttt{aknows}(A, p_A, M)$$
$$\texttt{deduce}(B, \{M\}_K \wedge EOO_M \wedge K) \Rightarrow \texttt{aknows}(B, p_B, \{M\}_K \wedge EOO_M \wedge K)$$

Non-repudiation of Receipt. The following table shows where those annotations take place in the three CCD sub-protocols, for considering non-repudiation of receipt.

$\mathcal{NRR}_A(B)$	Protocol - step
$\texttt{aknows}(A, p_A, \{M\}_K)$	Main - 1.
$\texttt{aknows}(A, p_A, EOR_M)$	Main - 2.
$\texttt{aknows}(A, p_A, EOR_K)$	Main - 4.
$\texttt{aknows}(A, p_A, E_{TTP})$	Abort - 2.
$\texttt{aknows}(A, p_A, E_{TTP})$	Resolve - 2.

For this property, E_{TTP} can be obtained from the second message of the abort sub-protocol or of the resolve sub-protocol.

According to Corollary 2, **non-repudiation of receipt** for the CCD protocol is represented by the following invariant formulae:

$$\texttt{aknows}(A, p_A, \{M\}_K \wedge EOR_M \wedge (EOR_K \vee E_{TTP})) \Rightarrow \texttt{aknows}(B, p_B, M)$$
$$\texttt{deduce}(A, p_A, \{M\}_K \wedge EOR_M \wedge (EOR_K \vee E_{TTP})) \Rightarrow \texttt{aknows}(A, p_A, \{M\}_K \wedge EOR_M \wedge (EOR_K \vee E_{TTP}))$$

Fairness. For analyzing **fairness**, this protocol requires timeliness, that is each participant should reach a final state before testing fairness. Fairness for the CCD protocol is described by the following logical formula, a very simple application of Theorem 1:

$$\texttt{aknows}(A, p_A, \mathcal{NRR}_A(B)) \Leftrightarrow \texttt{aknows}(B, p_B, \mathcal{NRO}_B(A))$$

Basically the property states that if A knows the EOR evidence ($\{M\}_K$, EOR_M, and EOR_K or E_{TTP}), then B knows the EOO evidence. And symmetrically for B, if B knows the EOO evidence ($\{M\}_K$, EOO_M and K), then A knows the EOR evidence.

Experiments. The CCD protocol has been specified in the AVISPA Tool, with the description of the fairness property given above. The detailed formulae used in the AVISPA Tool, with an LTL syntax, are:

$$\Box\left(\begin{pmatrix}\texttt{aknows}(A,p_A,\{M\}_K)\wedge\\ \texttt{aknows}(A,p_A,EOR_M)\wedge\\ (\texttt{aknows}(A,p_A,EOR_K)\vee\texttt{aknows}(A,p_A,E_{TTP}))\end{pmatrix}\Rightarrow\begin{pmatrix}\texttt{aknows}(B,p_B,\{M\}_K)\wedge\\ \texttt{aknows}(B,p_B,EOO_M)\wedge\\ \texttt{aknows}(B,p_B,K)\end{pmatrix}\right)$$

$$\Box\left(\begin{pmatrix}\texttt{aknows}(B,p_B,\{M\}_K)\wedge\\ \texttt{aknows}(B,p_B,EOO_M)\wedge\\ \texttt{aknows}(B,p_B,K)\end{pmatrix}\Rightarrow\begin{pmatrix}\texttt{aknows}(A,p_A,\{M\}_K)\wedge\\ \texttt{aknows}(A,p_A,EOR_M)\wedge\\ (\texttt{aknows}(A,p_A,EOR_K)\vee\texttt{aknows}(A,p_A,E_{TTP}))\end{pmatrix}\right)$$

Several scenarios have been run, and two of them have raised an attack, showing that the CCD protocol does not provide the fairness property for which it has been designed.

The ***first attack*** has been found for a scenario with only one protocol session where A, an honest agent, plays the protocol with a dishonest agent B (named i, for *intruder*). As soon as i has received the first message from A, he builds EOR_M and sends it to the TTP as resolve request. Later, when A, not receiving EOR_M, decides to abort the protocol, this is too late: the protocol has already been resolved, the intruder can get M and build the proof that A has sent M, and A cannot build the evidence of receipt, as he will never get EOR_M.
The trace of this attack is the following:

1. $A \rightarrow i:$ $\{M\}_K.EOO_M$
2. $i \rightarrow TTP:$ `RESOLVE`
3. $TTP \rightarrow i:$ E_{TTP}

`*** timeout for A ***`

4. $A \rightarrow TTP:$ `ABORT`
5. $TTP \rightarrow A:$ E_{TTP}

The ***second attack*** is a variant where both A and B are honest agents. The only difference is that B sends EOR_M to A, but this message is intercepted by the intruder and never delivered to A. At this point, the protocol is blocked, both agents waiting for a message. So, each agent will ask the help of the TTP for concluding the protocol: A will invoke the *abort* sub-protocol and B will invoke the *resolve* sub-protocol. And if the resolve request reaches the TTP before the abort request [5], B will get all his necessary evidences from the TTP, while A, having asked for an abort, will not be able to get all his evidences even with the help of the TTP.
The originality of this attack is that, at the end:

- A will guess (according to the answer received to his abort request) that the protocol has been resolved by B, so he will assume that B knows M and can build the proof that A has sent it; but A cannot prove this;
- B has resolved the protocol and has received from the TTP the information for getting M and building the proof that A has sent M; but he does not know that A does not have his proof;
- the TTP cannot know that A has not received EOR_M; so he knows that B can build its evidences, but he cannot know if A can or not.

[5] Note that this is possible even if channels are protected or pervasive, as agents use different channels; this is also possible if B has a shorter timeout than A; this notion of timeout is essential in the implementation of protocols, as demonstrated by Carbonell et al. in [5].

So, those attacks show that the CCD protocol is not fair, even if both agents A and B are honest. The attack is due to a malicious intruder or a network problem, and the TTP is of no help for detecting the problem.

Correcting the protocol is not difficult, for example by sending EOR_M together with E_{TTP} in the *abort* sub-protocol, when the protocol is already resolved. The numerous scenarios that have been tried for this new version have not raised any attack. This experiment on the CCD protocol is detailed in [24].

4.4 Back to the FairZG Protocol

We have illustrated in Section 3 the representation of non-repudiation properties by authentications with the FairZG protocol, raising some limitations and difficulties for an automatic analysis. We have also analyzed this protocol with our second method, based on agents knowledge.

This protocol is known for having an attack when agent A is dishonest [9]. Indeed in [31], it is not specified whether or not the TTP should store ConK forever. And from the TTP point of view, a transaction is closed once both A and B have retrieved ConK, so he could delete all the information about this transaction.

When the TTP acts in that way, Gürgens and Rudolph have described an attack: a first session is run until its end between A and B; then, A starts a second session with B, using the same K and L as in the first session, but with a different message M_2; if B does not remark the similarity of the sessions, he will answer to A; but once A has got NRR, he can stop the session, not sending the third message of the protocol; at that point, A owns NRR from the second session and ConK from the first session, and this constitutes the evidences of receipt of M_2 by B; on his side, B will never be able to get ConK from the TTP and will never know how to decrypt M_2.

So, this attack is due to the hypothesis that the TTP does not keep information on closed sessions. We have modeled this hypothesis by using two parallel processes for the TTP, one for each session. And we have found the same attack.

5 Conclusion

Non-repudiation protocols have an important role in many areas where secure transactions with proofs of participation are necessary. The evidences of origin and receipt of a message are two examples of elements that the parties should own at the end of a communication. We have given two very different examples of such protocols. The FairZG protocol is an intensively studied protocol in which the role of the trusted third party is essential. The CCD protocol is a more recent non-repudiation protocol that avoids the use of session labels and distinguishes itself by the use of an optimistic approach, the trusted third party being used only in case of a problem in the execution of the main protocol.

The fairness of a non-repudiation protocol is a property difficult to analyze and there are very few tools that can handle the automatic analysis of this property.

The contribution of this work is twofold. First, we have illustrated with the FairZG protocol how difficult it is to consider full non-repudiation properties using only a combination of authentications.

Second, we have defined a new method that permits to handle in a very easy way non-repudiation properties and fairness in a uniform framework. This method is based on the handling of agents knowledge and can be used to automatically analyze non-repudiation protocols as well as contract signing protocols [26]. We have implemented it in the AVISPA Tool and have successfully applied it to the CCD and FairZG protocols, proving that they are not fair. We have also tested other specifications of the CCD protocol, for example with secure communication

channels between agents and the TTP, no attack has been found; but using such channels is not considered as acceptable, because it generates an overload of the TTP activity.

Our method, based on the writing of simple state invariants, is of easy use, and can be implemented in any tool handling agents (or intruder) knowledge. It should be very helpful for setting abstractions for handling unbounded scenarios, and it should be very efficient for bounded verifications, as it has been the case in our implementation. We hope that this work will open a highway to the specification of many other properties, without any more change in the specification languages and the analysis engines.

Our work has been done for analyzing non-repudiation protocols. A complementary approach has been defined by Guttman in [11], where he describes a protocol design process, based on authentication tests, permitting to guarantee some security properties, including some non-repudiation properties. Note that in the example presented by Guttman, fairness is not considered.

References

1. `http://www.lsv.ens-cachan.fr/~kremer/FXbib/references.php`.
2. A. Armando, D. A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In K. Etessami and S. K. Rajamani, editors, *Computer Aided Verification, 17th International Conference, CAV 2005*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285, Edinburgh, Scotland, UK, 2005. Springer.
3. A. Armando, R. Carbone, and L. Compagna. LTL Model Checking for Security Protocols. In *20th IEEE Computer Security Foundations Symposium (CSF'07)*, pages 385–396, Los Alamitos, CA, USA, 2007. IEEE Computer Society.
4. G. Bella and L. C. Paulson. Mechanical Proofs about a Non-repudiation Protocol. In R. J. Boulton and P. B. Jackson, editors, *Theorem Proving in Higher Order Logics, 14th International Conference, TPHOLs 2001*, volume 2152 of *Lecture Notes in Computer Science*, pages 91–104, Edinburgh, Scotland, UK, 2001. Springer.
5. M. Carbonell, J. M. Sierra, J. A. Onieva, J. Lopez, and J. Zhou. Estimation of TTP Features in Non-repudiation Service. In O. Gervasi and M. L. Gavrilova, editors, *International Conference on Computational Science and Its Applications - ICCSA*, volume 4706 of *Lecture Notes in Computer Science*, pages 549–558, Kuala Lumpur, Malaysia, Aug. 2007. Springer.
6. J. Cederquist, R. Corin, and M. T. Dashti. On the Quest for Impartiality: Design and Analysis of a Fair Non-repudiation Protocol. In S. Qing, W. Mao, J. Lopez, and G. Wang, editors, *Information and Communications Security, 7th International Conference, ICICS 2005*, volume 3783 of *Lecture Notes in Computer Science*, pages 27–39, Beijing, China, 2005. Springer.
7. Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, J. Mantovani, S. Mödersheim, and L. Vigneron. A High Level Protocol Specification Language for Industrial Security-Sensitive Protocols. In *Proceedings of Workshop on Specification and Automated Processing of Security Requirements (SAPS)*, volume 180, Linz, Austria, Sept. 2004. Oesterreichische Computer Gesellschaft (Austrian Computer Society).
8. L. Compagna. *SAT-based Model-Checking of Security Protocols*. PhD thesis, PhD Thesis, Università degli Studi di Genova and the University of Edinburgh, 2005.
9. S. Gürgens and C. Rudolph. Security Analysis of Efficient (Un-)fair Non-repudiation Protocols. *Formal Aspects of Computing*, 17(3):260–276, 2005.
10. S. Gürgens, C. Rudolph, and H. Vogt. On the Security of Fair Non-repudiation Protocols. *Int. Journal of Information Security*, 4:253–262, 2005.
11. J. D. Guttman. Authentication tests and disjoint encryption: A design method for security protocols. *Journal of Computer Security*, 12(3-4):409–433, 2004.

12. F. Jacquemard, M. Rusinowitch, and L. Vigneron. Compiling and Verifying Security Protocols. In M. Parigot and A. Voronkov, editors, *Logic for Programming and Automated Reasoning*, volume 1955 of *Lecture Notes in Computer Science*, pages 131–160, St Gilles, Réunion, France, Nov. 2000. Springer.
13. S. Kremer and O. Markowitch. Optimistic Non-repudiable Information Exchange. In J. Biemond, editor, *21st Symp. on Information Theory in the Benelux*, pages 139–146, Wassenaar (NL), 2000. Werkgemeenschap Informatieen Communicatietheorie, Enschede (NL).
14. S. Kremer, O. Markowitch, and J. Zhou. An Intensive Survey of Fair Non-repudiation Protocols. *Computer Communications*, 25(17):1606–1621, 2002.
15. S. Kremer and J.-F. Raskin. A Game-Based Verification of Non-repudiation and Fair Exchange Protocols. In K. G. Larsen and M. Nielsen, editors, *CONCUR 2001 - Concurrency Theory, 12th International Conference*, volume 2154 of *Lecture Notes in Computer Science*, pages 551–565, Aalborg, Denmark, 2001. Springer.
16. G. Lowe. A Hierarchy of Authentication Specification. In *10th Computer Security Foundations Workshop (CSFW'97)*, pages 31–44, Rockport, Massachusetts, USA, 1997. IEEE Computer Society.
17. O. Markowitch and S. Kremer. An Optimistic Non-repudiation Protocol with Transparent Trusted Third Party. In G. I. Davida and Y. Frankel, editors, *Information Security, 4th International Conference, ISC 2001*, volume 2200 of *Lecture Notes in Computer Science*, pages 363–378, Malaga, Spain, 2001. Springer.
18. O. Markowitch and Y. Roggeman. Probabilistic Non-Repudiation without Trusted Third Party. In *Second Workshop on Security in Communication Networks'99*, Amalfi, Italy, 1999.
19. P. Ochsenschläger, J. Repp, R. Rieke, and U. Nitsche. The SH-Verification Tool - Abstraction-Based Verification of Co-operating Systems. *Formal Aspects of Computing*, 10(4):381–404, 1998.
20. H. Pagnia and F. C. Gärtner. On the Impossibility of Fair Exchange without a Trusted Third Party. Technical Report TUD-BS-1999-02, Darmstadt University of Technology, Darmstadt, Germany, 1999.
21. S. Pancho-Festin and D. Gollmann. On the Formal Analyses of the Zhou-Gollmann Non-repudiation Protocol. In T. Dimitrakos, F. Martinelli, P. Ryan, and S. Schneider, editors, *Formal Aspects in Security and Trust, FAST'05*, volume 3866 of *Lecture Notes in Computer Science*, pages 5–15, Newcastle, UK, 2006. Springer.
22. A. W. Roscoe, C. A. R. Hoare, and R. Bird. *The Theory and Practice of Concurrency*. Prentice Hall PTR, 1997.
23. P. Ryan, M. Goldsmith, G. Lowe, B. Roscoe, and S. Schneider. *Modelling and Analysis of Security Protocols*. Addison Wesley, 2000.
24. J. Santiago and L. Vigneron. Optimistic Non-repudiation Protocol Analysis. In D. S. et al., editor, *Proceedings of the Workshop in Information Security Theory and Practices (WISTP'2007), Smart Cards, Mobile and Ubiquitous Computing Systems*, volume 4462 of *Lecture Notes in Computer Science*, pages 90–101, Heraklion (Greece), May 2007. Springer.
25. S. Schneider. Formal Analysis of a Non-Repudiation Protocol. In *Proceedings of The 11th Computer Security Foundations Workshop*, pages 54–65. IEEE Computer Society Press, 1998.
26. V. Shmatikov and J. C. Mitchell. Analysis of Abuse-Free Contract Signing. In Y. Frankel, editor, *Financial Cryptography, 4th International Conference, FC 2000*, volume 1962 of *Lecture Notes in Computer Science*, pages 174–191, Anguilla, British West Indies, 2000. Springer.
27. K. Wei and J. Heather. Towards Verification of Timed Non-repudiation Protocols. In T. Dimitrakos, F. Martinelli, P. Y. A. Ryan, and S. A. Schneider, editors, *Formal Aspects in Security and Trust, 3rd international Workshop, FAST 2005, Revised selected papers*, volume 3866 of *Lecture Notes in Computer Science*, pages 244–257, Newcastle, UK, July 2006. Springer.
28. K. Wei and J. Heather. A Theorem-Proving Approach to Verification of Fair Non-repudiation Protocols. In T. Dimitrakos, F. Martinelli, P. Y. A. Ryan, and S. A. Schneider, editors, *Formal Aspects in Security and Trust, 4th international Workshop, FAST 2006, Revised selected papers*, volume 4691 of *Lecture Notes in Computer Science*, pages 202–219, Hamilton, Ontario, Canada, Aug. 2007. Springer.
29. J. Zhou, R. H. Deng, and F. Bao. Evolution of Fair Non-repudiation with TTP. In *ACISP '99: Proceedings of the 4th Australasian Conference on Information Security and Privacy*, volume 1587 of *Lecture Notes in Computer Science*, pages 258–269. Springer-Verlag, 1999.

30. J. Zhou and D. Gollmann. A Fair Non-repudiation Protocol. In *1996 IEEE Symposium on Security and Privacy*, pages 55–61, Oakland, CA, USA, 1996. IEEE Computer Society.
31. J. Zhou and D. Gollmann. Towards verification of non-repudiation protocols. In *Proceedings of 1998 International Refinement Workshop and Formal Methods Pacific*, pages 370–380, Canberra, Australia, September 1998.