Optimistic Non-Repudiation Protocol Analysis

Judson Santiago and Laurent Vigneron*

LORIA – Nancy Université
{laurent.vigneron}@loria.fr

Abstract. Non-repudiation protocols with session labels have a number of vulnerabilities. Recently Cederquist, Corin and Dashti have proposed an optimistic non-repudiation protocol that avoids altogether the use of session labels. We have specified and analysed this protocol using an extended version of the AVISPA Tool and one important fault has been discovered. We describe the protocol, the analysis method, show two attack traces that exploit the fault and propose a correction to the protocol.

1 Introduction

While security issues such as secrecy and authentication have been studied intensively [11], most interest in non-repudiation protocols has only come in recent years, notably in the yearly 1990s with the explosion of Internet services and electronic transactions.[1]

Non-repudiation protocols must ensure that when two parties exchange information over a network, neither one nor the other can deny having participated to this communication. Consequently a non-repudiation protocol must generate evidences of participation to be used in case of a dispute. With the advent of digital signatures and public key cryptography, the base for non-repudiation services was created. Given an adequate public key infrastructure, one having a signed message has an evidence of the participation and the identity of his party [7].

While non-repudiation can be provided by standard cryptographic mechanisms like digital signatures, fairness is more difficult to achieve: no party should be able to reach a point where they have the evidence or the message they require without the other party also having their 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 [7]. However this simultaneous secret exchange is troublesome for actual implementations because fairness is based on the assumption of equal computational power on 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 it is impossible to achieve fair exchange without a TTP [10, 9]. The TTP can be used as a delivery agent to provide simultaneous share

* This work is supported by the ACI Sécurité SATIN and the RNTL project 03V360 Prouvé.

[1] See http://www.lsv.ens-cachan.fr/~kremer/FXbib/references.php for a detailed list of publications.

of evidences. The Fair Zhou-Gollmann protocol [16] is the most known example of non-repudiation protocol, using a TTP as a delivery agent of a key for decrypting the message sent by one agent to another agent; a significant amount of work has been done over this protocol and its derivations [2, 6, 13, 17]. However, instead of passing the complete message through the TTP and thus creating a possible bottleneck, recent evolution of these protocols resulted in efficient, *optimistic* versions, in which the TTP is only involved in case anything 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 (timeliness).

One of these recent protocols, which we describe in the following section, is the optimistic Cederquist-Corin-Dashti (CCD) non-repudiation protocol [3]. The CCD protocol has the advantage of not using session labels, contrariwise to many others in the literature [7, 8, 16, 13]. A session label typically consists of a hash of all message components. Gürgens et al. [6] 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.

In this paper we describe the CCD non-repudiation protocol, present the analysis method and explain two attack traces of an important flaw discovered in this protocol. The attack has been found after the specification and analysis of the protocol in the AVISPA Tool [1][2], using an extended version of the AtSe engine [15] that supports non-repudiation analysis. We propose a correction for the CCD protocol that have been successfully analysed for many scenarios.

2 The CCD Protocol

The CCD non-repudiation protocol has been created for permitting an agent A to send a message M to 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.

2.1 Definition of the Main Protocol

This main protocol 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 a problem happens to one of the agents, in order to finish properly the protocol, the agents can exchange messages with a trusted third party (TTP) by executing the *abort* or the *resolve* sub-protocol.

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

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

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 assure 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 can check his identity, learns the name of the TTP, can check that the hash code is the result of hashing the first part of the message, but 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 message M.
Finally, B sends to A another evidence of receipt, permitting A to check that the symmetric key has been received by B.

2.2 The *Abort* Sub-Protocol

The *abort* sub-protocol is executed by agent A in case 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: the hash of the encrypted message, the name of the other agent (B), and the key used for encrypting M.
According to what happened before, the TTP 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, 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.

2.3 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. 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.

2.4 Agents' Evidences

Non-repudiation protocols require evidence of receipt (EOR) and evidence of origin (EOO). All parties have to agree that these evidences constitute a valid proof of participation in the protocol. In the case of a dispute, the parties should present their evidences to an external judge. Ideally the judge should be capable of deciding the matter by executing a verification algorithm over the evidences presented by each party.

For the CCD protocol, the evidence of receipt for A is $\{M\}_K$ and EOR_M, plus either EOR_K or E_{TTP}. The evidence of origin for B is $\{M\}_K$, EOO_M and K. At the end of the protocol execution, each agent must have all the parts that compose his evidence. The choice of these evidences is not discussed here, see [3] for more information.

3 Analysis of the CCD Protocol

The CCD protocol was formally analysed by its authors in [3] and no attack has been found for the following scenarios: A and B honest; A honest, B dishonest; and B dishonest, A honest.

But our analysis shows that there is a serious flaw in the protocol, even when the agents act honestly. The attack occurs because one agent does not get all the required information for building its evidence when the protocol finishes by the intervention of the TTP. We describe in Sections 3.3 and 3.4 two scenarios that lead to an unfair situation for the agent playing the role A, thus contradicting the result of [3] for the same fairness property. But before presenting the attacks, we describe in the next sections the AVISPA Tool analysis method and the representation of the non-repudiation properties in the AVISPA Tool.

3.1 Analysis Method

Our analysis method is based on the technology build into the AVISPA Tool [1]: the protocol is specified in the High Level Protocol Specification Language (HLPSL) [4], translated into a state transition system called the intermediate format (IF) and fed

to one of the four analysis engines available with the tool. In this work, the Attack Searcher (AtSe) engine [15] has been used. The AtSe analysis engine implements the so-called lazy intruder model [5], which greatly increases the performance of the searching process. Previously only used to analyse secrecy and authentication properties, we have extended this engine to support a subset of Linear Temporal Logics (LTL) formulae, allowing the specification and analysis of a broader spectrum of properties, including the fairness property for non-repudiation.

3.2 Description of Non-repudiation Properties

The AVISPA Tool was designed to analyse complex Internet security protocols, like the protocols described by the Internet Engineering Task Force (IETF). Even though the tool has support for the specification of arbitrarily complex properties by the use of LTL formulae, no analysis engine of the AVISPA Tool actually uses this power. Natively, properties are specified by the use of macros and only secrecy and authentication properties are supported.

In a previous work [12], we have represented non-repudiation properties as a combination of authentication properties. This representation has been applied to the Fair Zhou-Gollmann protocol [16] and has given good results, raising a problem in the protocol. But because of the implementation of the intruder strategy in the AVISPA Tool, the notion of dishonest agent could not be fully expressed (see [12] for more details). This is the reason why we have decided to use LTL formulae for describing non-repudiation properties in HLPSL, and to extend AtSe for considering this kind of formulae.

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

To analyse non-repudiation in the AVISPA Tool, we have to find a way to express the knowledge of the agents by a predicate added in some protocol transitions, and to find a way to express the non-repudiation properties by the use of these predicates. We have then introduced the predicates `aknows` (for agent knowledge) and `iknows` (for intruder knowledge) in all the levels of the AVISPA Tool, namely in the specification language (HLPSL), in the intermediate format (IF) and in the analysis engine (AtSe). Note that `iknows` was already used in the IF and in AtSe. As with the other predicates, `aknows` and `iknows` are used in the LTL description of the properties (non-repudiation properties in our case) and to mark the protocol specification.

Definition 1 (aknows). *Le $\mathcal{A}$ be a set of agents playing a finite number of sessions $\mathcal{S}$ of a protocol, $\mathcal{T}$ a set of terms sent in the messages of this protocol and $\mathcal{E}$ the subset of terms $t \in \mathcal{T}$ that are part of the evidences of non-repudiation in the protocol. For an agent $a \in \mathcal{A}$, $\mathcal{E}_a$ is the set of terms $t \in \mathcal{E}$ that constitute the evidence of non-repudiation for the agent a. The predicate* $\texttt{aknows}(a, b, s, t)$ *with $a, b \in \mathcal{A}$, $s \in \mathcal{S}$ and $t \in \mathcal{T}$, express that the agent a, playing with agent b in the session s, knows the term t.*

Definition 2 (Non-repudiation of origin or receipt). *If at the end of the execution of agent a in protocol session s, the predicate* $\texttt{aknows}(a, b, s, t)$ *is true for all $t \in \mathcal{E}_a$, then the non-repudiation property (of origin or receipt, according to the role of a in the protocol) is satisfied. Otherwise, the property of non-repudiation for agent a is false.*

The fairness of the non-repudiation property is true only when both agents know their non-repudiation evidences, or when neither one nor the other knows his evidence. But for the properties of non-repudiation of origin and non-repudiation of receipt, the knowledge of one agent is enough to decide if the property is true or not.

With the predicates `aknows` and `iknows`, we know exactly when an agent learns a term t and thus we can automatically verify the non-repudiation properties using the knowledge of the agents. If at the end of the execution of an agent, there is no `aknows` for the non-repudiation evidences of that agent, then we have a non-repudiation of origin or non-repudiation of receipt attack.

Definition 3 (Fairness). *If at the end of the execution of agent a in session s, the predicate* $\texttt{aknows}(a, b, s, t)$ *is true for all $t \in \mathcal{E}_a$, then the fairness property is true from the point of view of a. And if the fairness property is true from the point of view of the other agent, say b, the protocol session is said to be fair. The protocol is also fair if none of the agents knows all his evidences. Otherwise, the fairness property is false.*

Even if the fairness property needs data from both agents, when the predicate `aknows` is true for one agent, agent a for example, we can guarantee that the property is satisfied from the point of view of a and concentrate the analysis on the property by the point of view of agent b at the end of his execution. If one agent is dishonest or personified by the intruder, say b for example, the predicate $\texttt{aknows}(b, a, s, u)$ must be replaced by $iknows(u)$ and the agent name is written i (the intruder name). This last predicate is satisfied if the intruder knows (or can build from his knowledge) the term u.

The AtSe analysis engine has been extended to analyse properties described as LTL formulae using `aknows` and `iknows` predicates. The non-repudiation fairness for the CCD protocol is described by the following LTL formula:

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

Basically the property states that if A knows the EOR evidence ($\{M\}_K$, EOR_M, and EOR_K or E_{TTP}) or if the intruder, playing the role A, knows this evidence, then B

must know the EOO evidence. There is a similar property for B: if B knows the EOO evidence ($\{M\}_K$, EOO_M, and K or E_{TTP}) or if the intruder knows it, then A must know the EOR evidence:

$$\Box\left(\begin{pmatrix}(\ \mathtt{aknows}(B,A,s,\{M\}_K)\wedge\mathtt{aknows}(B,A,s,EOO_M)\wedge\\ \ (\mathtt{aknows}(B,A,s,K)\vee\mathtt{aknows}(B,TTP,s,E_{TTP}))\ )\vee\\ (\ \mathtt{iknows}(\{M\}_K)\wedge\mathtt{iknows}(EOO_M)\ \wedge B=i\wedge\\ \ (\mathtt{iknows}(K)\vee\mathtt{iknows}(E_{TTP}))\ )\end{pmatrix}\Rightarrow\begin{pmatrix}\mathtt{aknows}(A,B,s,\{M\}_K)\wedge\\ \mathtt{aknows}(A,B,s,EOR_M)\wedge\\ (\ \mathtt{aknows}(A,B,s,EOR_K)\vee\\ \ \mathtt{aknows}(A,TTP,s,E_{TTP})\ )\end{pmatrix}\right)$$

The protocol was specified in the HLPSL language and analysed with the new version of the AtSe engine. The attacks found in the analysis are described in the following sections.

3.3 Delayed Abort Request Attack

When A does not receive EOR_M from B, the *abort* sub-protocol is invoked. When B does not receive K from A, the *resolve* sub-protocol is invoked. So, if the messages EOR_M and K are not sent or delayed in the insecure channel between A and B (either because of a network problem, or intercepted by the intruder), both agents will query the TTP, A trying to abort and B trying to resolve the protocol.

The problem arises if the abort request does not reach the TTP before the resolve request. In this case, the TTP will resolve the protocol, permitting B to get all the knowledge for building the evidence of origin. Because of this previous resolve request by B, the abort request by A will not lead to the abortion of the protocol. If the TTP receives this abort request, he will send E_{TTP} to A, but as A does not (and cannot) know EOR_M, he cannot build the evidence of receipt. So, at the end of the execution, there is a fairness attack, as B can prove that A has sent M, but A cannot prove that B has received it.

The attack trace given below, automatically found by AtSe, is even more surprising, as explained hereafter. In this trace, $i(G)$ means that the intruder impersonated agent G; and for a better clarity, the detailed contents of messages have been replaced by more explicit names.

```
1. A -> i(B)   : {M}_K.EOOM
*** timeout for A ***
2. A -> i(TTP) : ABORT
3. i(A) -> B   : {M}_K.EOOM
4. B -> i(A)   : EORM
5. i(A) -> TTP : RESOLVE  (=EORM)
6. TTP -> i(A) : ETTP
*** timeout for B ***
7. B -> i(TTP) : RESOLVE
8. i(TTP) -> A : ETTP
9. i(TTP) -> B : ETTP
```

The first step is the standard one, but the intruder intercepts the message before it reaches B. Without any answer to his message, A decides to abort the protocol, message

also intercepted by i (step 2). In step 3, i impersonating A forwards the message 1 to B, who answers with EOR_M (step 4). The intruder uses this last message for pretending to the TTP that A wants to resolve the protocol (step 5). As the TTP has not received the abort request of A, he answers by sending E_{TTP} (step 6). B not having any answer to his EOR_M message, he decides to ask the TTP for resolving the protocol (step 7). Then the intruder sends the TTP resolve answer to A and B (steps 8 and 9).

The originality of this attack trace 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 will think that A has asked for the protocol to be resolved, followed by B; so for him, both A and B can build their evidences.

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

3.4 Dishonest Agent Attack

A variant of the previous attack has also been discovered by AtSe. It happens when agent A plays the protocol with a dishonest agent B (called the intruder and names i). As soon as i has received the first message from A, he builds EOR_M and sends it to the TTP as resolve request. When A 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.

```
1. A -> i   : {M}_K.EOOM
2. i -> TTP : RESOLVE
3. TTP -> i : ETTP
*** timeout for A ***
4. A -> TTP : ABORT
5. TTP -> A : ETTP
```

4 Correction of the CCD Protocol

In this section, we first discuss the role of the trusted third party for trying to solve the problems raised by the attacks found. Then we describe a correction of the *abort* sub-protocol and report the new analyses done, in which no attack has been found.

4.1 About the TTP Role

Both attacks described in the previous section come from the same flaw: the TTP does not give EOR_M to agent A when the protocol is already resolved and A tries to abort it. However, the TTP has received EOR_M in the resolve request, so one can argue that A only needs to know E_{TTP} to prove that B knows the message M: A knowing E_{TTP} means that TTP knows EOR_M, and consequently A could know EOR_M by asking it to the TTP, in case of a dispute.

From B's side, if B resolves the protocol and gets the message E_{TTP}, this means that B knows EOR_M, and according to the protocol, owning EOR_M means owning EOO_M and M_K. If the TTP stores EOR_M in its database for every resolved transaction, A could try to prove that B knows M by requesting to the TTP a proof that EOR_M is known by B.

If we consider this situation acceptable, and if we prove that A knowing E_{TTP} implies B also knowing E_{TTP} and M_K, we can say that the protocol is fair even when A only receives E_{TTP} as evidence of receipt.

But this situation is not acceptable, first because accepting E_{TTP} as an evidence of receipt puts extra importance on the TTP. The evidences should be strong enough to prove participation in the protocol without the need of using TTP's knowledge as part of the proof. Second, the TTP would need to store all EOR_M messages for all resolved sessions of the protocol. And last, without EOR_M we cannot prove that B has agreed on the use of the agent TTP as the trusted third party: there is no message signed by B that contains the name of the TTP. So E_{TTP} cannot be a proof of receipt without EOR_M.

This is why we propose some changes to correct this flaw in the protocol.

4.2 Correction of the *abort* Sub-protocol

To correct the protocol, we need to change the *abort* sub-protocol to provide the complete EOR evidence to A, no matter the sequence of abort and resolve requests in the session of the protocol. Below we present the new version of the *abort* sub-protocol.

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

Messages E_{TTP}, EOR_M and AB_{TTP} are the same as in the original protocol. The only change is the addition of EOR_M message in the TTP's answer to A when the sub-protocol is invoked and the TTP has already resolved the session (and stored EOR_M together with the `resolved` predicate in its database).

We have specified and analysed the corrected protocol. An extended number of scenarios has been checked, compared to the original work of Cederquist et al. [3], including two-sessions scenarios where the sessions are run in parallel.

One-session scenarios. We have analysed the common one-session scenarios: A and B honest, A honest and B dishonest, A dishonest and B honest. In our analysis approach, the intruder impersonates the dishonest agents. For all three scenarios the fairness property could not be falsified.

Two-sessions scenarios. We have also analysed some critical two-sessions scenarios: A and B honest in parallel with A honest and B dishonest; A and B honest in parallel with A dishonest and B honest; A honest and B dishonest in parallel with A dishonest and B honest. When running sessions in parallel, the intruder has an improved knowledge and he can try, for example, to use knowledge/messages from one session in the other session. Again, for those scenarios AtSe has found no fairness attack.

5 Conclusion

Non-repudiation protocols have an important role in many areas where secured transactions with proofs of participation are necessary. The evidences of origin and receipt of a message are the elements that the parties should have at the end of the communication. The CCD protocol is a 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 analyse 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 extended the AVISPA Tool and one of its analysis engines, AtSe, to implement our analysis method for the non-repudiation properties. Our method is based on the knowledge of agents and can be used to automatically analyse non-repudiation protocols as well as contract signing protocols [14]. Second, with this method, we have specified and analysed the CDD protocol and a serious flaw has been found. We have proposed a correction that has been further analysed by additional scenarios and no attack has been found.

Our representation of the non-repudiation properties has also been applied successfully to the Fair Zhou-Gollmann protocol [12]. We have tested other specifications of the CCD protocol, for example with secure communication channels between agents and the TTP, and for the original definition for the *abort* sub-protocol: no attack has been found; but using such channels is not considered as acceptable, because it requires too much work for the TTP.

The AVISPA Tool has proved its efficiency for analysing secrecy and authentication properties of protocols. We have extended it to handle non-repudiation properties, but by this extension, adding `aknows` and `iknows` predicates and using LTL formulae as goal, we have open a highway to the specification of many other properties, without any more change in the specification languages and the analysis engines. And for the analysis of the CCD protocol, the use of LTL formulae did not have any impact on the speed of AtSe for finding attacks (or for not finding attacks concerning the fixed version of the protocol).

References

1. Alessandro Armando, David A. Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, Paul Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, Sebastian Mödersheim, David von Oheimb, Michaël Rusinowitch, Judson Santiago, Mathieu Turuani, Luca Viganò, and Laurent Vigneron. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In Kousha Etessami and Sriram 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.
2. Giampaolo Bella and Lawrence C. Paulson. Mechanical Proofs about a Non-repudiation Protocol. In Richard J. Boulton and Paul 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.
3. Jan Cederquist, Ricardo Corin, and Muhammad Torabi Dashti. On the Quest for Impartiality: Design and Analysis of a Fair Non-repudiation Protocol. In Sihan Qing, Wenbo Mao, Javier Lopez, and Guilin 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.
4. Yannick Chevalier, Luca Compagna, Jorge Cuéllar, Paul Hankes Drielsma, Jacopo Mantovani, Sebastian Mödersheim, and Laurent Vigneron. A High Level Protocol Specification Language for Industrial Security-Sensitive Protocols. In *Automated Software Engineering. Proceedings of the Workshop on Specification and Automated Processing of Security Requirements, SAPS'04*, pages 193–205, Austria, September 2004. Austrian Computer Society.
5. Yannick Chevalier and Laurent Vigneron. A Tool for Lazy Verification of Security Protocols. In *16th IEEE International Conference on Automated Software Engineering (ASE 2001)*, pages 373–376, San Diego, CA, USA, 2001. IEEE Computer Society.
6. Sigrid Gürgens, Carsten Rudolph, and Holger Vogt. On the Security of Fair Non-repudiation Protocols. In Colin Boyd and Wenbo Mao, editors, *Information Security, 6th International Conference, ISC 2003*, volume 2851 of *Lecture Notes in Computer Science*, pages 193–207, Bristol, UK, 2003. Springer.
7. Steve Kremer, Olivier Markowitch, and Jianying Zhou. An Intensive Survey of Fair Non-repudiation Protocols. *Computer Communications Journal*, 25(17):1606–1621, 2002.
8. Olivier Markowitch and Steve Kremer. An Optimistic Non-repudiation Protocol with Transparent Trusted Third Party. In George I. Davida and Yair Frankel, editors, *Information Security, 4th International Conference, ISC 2001*, volume 2200 of *Lecture Notes in Computer Science*, pages 363–378, Malaga, Spain, 2001. Springer.
9. Olivier Markowitch and Yves Roggeman. Probabilistic Non-Repudiation without Trusted Third Party. In *Second Workshop on Security in Communication Networks'99*, Amalfi, Italy, 1999.
10. Henning Pagnia and Felix 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.
11. Peter Ryan, Michael Goldsmith, Gavin Lowe, Bill Roscoe, and Steve Schneider. *Modelling & Analysis of Security Protocols*. Addison Wesley, 2000.
12. Judson Santiago and Laurent Vigneron. Automatically Analysing Non-repudiation with Authentication. In *Proceedings of 3rd Taiwanese-French Conference on Information Technology (TFIT)*, pages 541–554, Nancy, France, March 2006.
13. Steve 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.
14. Vitaly Shmatikov and John C. Mitchell. Analysis of Abuse-Free Contract Signing. In Yair 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.
15. Mathieu Turuani. The CL-Atse Protocol Analyser. In Frank Pfenning, editor, *Term Rewriting and Applications, 17th International Conference, RTA 2006*, volume 4098 of *Lecture Notes in Computer Science*, pages 277–286, Seattle, WA, USA, 2006. Springer.

16. Jianying Zhou and Dieter Gollmann. A Fair Non-repudiation Protocol. In *1996 IEEE Symposium on Security and Privacy*, pages 55–61, Oakland, CA, USA, 1996. IEEE Computer Society.
17. Jianying Zhou and Dieter 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.