www.avispa-project.org

IST-2001-39252

Automated Validation of Internet Security Protocols and Applications

Deliverable D3.1: Security Properties

Abstract

In this deliverable, we investigate the specification and formalisation of a number of security properties, such as different forms of authentication, secrecy, anonymity, and non-repudiation. Our current methods are well suited to automatically – and very efficiently – check for violations of security properties goals like authentication and secrecy. Furthermore, many other security properties closely resemble authentication and secrecy. It is, therefore, particularly desirable to find reductions from complex properties into Boolean or temporal combinations of these simpler properties for which efficient analysis techniques already exist. We adopt this approach wherever possible.

Deliverable details

Deliverable version: *v1.0* | Person-months required: *19*
Date of delivery: *18.02.2005* | Due on: *31.12.2004*
Classification: *public* | Total pages: *30*

Project details

Start date: *January 1st, 2003*
Duration: *30 months*
Project Coordinator: *Alessandro Armando*
Partners: *Università di Genova, INRIA Lorraine, ETH Zürich, Siemens AG*

Project funded by the European Community under the *Information Society Technologies* Programme (1998-2002)

Contents

1 Introduction

Security protocols are designed to ensure certain *security properties*; that is, the *goals* which the protocols should achieve.[1] Authentication of principals and secrecy of confidential data are classical examples of such goals, though many others exist. Deliverable 6.1 [7] lists the security properties relevant for the protocols of the AVISPA List of Problems, as well as a selection of interesting properties currently under discussion at the Internet Engineering Task Force (IETF) but, as yet, beyond the scope of formal analysis. The properties we study here all fall in the class of properties which, in the general framework of verification, are known as *safety properties*; that is, they express that something "bad" should not happen.

To protocol designers, security properties represent guarantees: specifically, those that should be assured after execution of a protocol. To those in the formal methods community, security properties represent the correctness specifications against which we attempt to analyse formal protocol models. Often, characterising these correctness specifications can be the most challenging aspect of modelling a protocol, as one must extract — usually from an informal description such as a Request For Comments (RFC) — the formal specification of what was intended by the protocol designers. It is, for instance, not sufficient to say, informally, that two agents should authenticate one another. In order to build a formal model, one must be more precise: the modeller must know (or infer) where to place the so-called goal events (see below) and what they should contain. This entails knowing precisely at what point an agent accepts the authentication of his communication partner as having been successful, and upon what data the two agents should agree upon when authenticating one another.

Goal Reductions. Our current methods are well suited to automatically — and very efficiently — check for violations of security properties like authentication and secrecy. Furthermore, many other security properties closely resemble authentication and secrecy. More specifically, authentication in particular has the classical form of a so-called *safety temporal property* [7], expressing that one given event should always be preceded by another. Many other properties share this general form. It is, therefore, particularly desirable to find reductions from complex properties into Boolean or temporal combinations of these simpler properties for which efficient analysis techniques already exist. We adopt this approach wherever possible.

Specifying Security Properties. HLPSL [4] and IF [5] are based on different formalisms: the former on temporal logic, and the latter on term rewriting. Each language therefore lends itself to expressing security properties in a manner suited to its respective formalism: namely, HLPSL goals are expressed as temporal formulae, while IF goals take

[1]We will often use the terms "security property" and "goal" interchangeably.

the form of a description of states that represent violations of one or more of the goals.[2]

Specifications in both of these languages include so-called *goal facts* or *goal events*. Intuitively, these represent flags indicating that an agent has reached a certain significant point in the execution of the protocol. For instance, the `request` (see, e.g., §4) event marks the point when an agent accepts the authentication of his communication partner.

Semantically, a goal event in HLPSL translates to a simple Boolean variable, and a goal fact in IF is simply a fact like any other. There is, however, an important semantic difference between the two: namely, the Boolean variables to which HLPSL goal events translate are true only in the transitions in which the goal events appear and false otherwise; in contrast, IF goal facts are terms that, once introduced, are persistent (unless removed by the application of another rewrite rule). It is over these goal events/facts that we define the temporal formulae (in HLPSL) and goal state descriptions (in IF) that specify the security properties we wish to analyse. This is discussed in more detail in [4, 5], and we will illustrate with concrete property specifications in the coming sections.

We note briefly that one could avoid the specification of goal facts within a HLPSL model and seek alternate ways of expressing security properties. For example, given HLPSL's temporal logic semantics, one could model goals as temporal formulae over the protocol execution trace itself, for instance formulae describing the values that have passed over individual channels. We find it desirable, however, to hide details like this from the protocol modeller. Having goal facts allows us to be more explicit and decouples the modelling of the goal from the modelling of the protocol behaviour.

Structure of This Deliverable. In §2, we describe a manner of thinking and reasoning about goals that we find useful and which we use in some of the subsequent discussions. In the sections that follow, we discuss individual security properties, the challenges they pose, and our approaches to modelling them in HLPSL: §3 discusses secrecy properties; §4 treats authentication; §5 discusses anonymity; and §6 considers non-repudiation. We conclude in §7.

2 Understanding Security Properties

We find it helpful to adopt an intuitive common manner of thinking about goals which we present here. This method of understanding goals is informal but can effectively guide the process of formalising security properties. Essentially, we propose to think about security properties as they can be understood on three levels.

Message Analysis and Synthesis. At the lowest level, we consider the message analysis and synthesis abilities of the intruder. Here, the question is really one of which messages can the intruder indeed obtain and/or generate himself? Secrecy is a property that can

[2]Other formalisms, of course, have different approaches to the modelling of security properties; however, we do not discuss these alternate approaches here.

be understood almost entirely on this lowest level. Even secrecy, however, has subtler aspects: what if, for instance, we wish to model short-term secrecy? If, after a certain time, secrets are to be revealed to the intruder, then the question of whether or not the intruder can obtain them is no longer sufficient. One must further ask when he obtained them: before they were revealed, or after? This leads us to the second level of our common understanding of goals.

Temporal Aspects. The second level involves the temporal aspects of the security property. Here, we are reasoning not about individual messages, but about what traces the intruder can cause to happen. By reasoning on both of these two levels, one can thus pose questions like "Can the intruder synthesise this data which should remain secret temporarily, and can he do so before the data is revealed?"

Meta-Reasoning. The third level we propose is one which we call the level of meta-reasoning. One can sometimes justify that, in a given scenario, a particular security property can be reduced to another or has an equivalent alternative formulation. Reasoning done at this level is not necessarily formal but can nonetheless be very helpful. The "inputs" to this reasoning process include, for instance, our intuition about the goal itself, the properties of the model we have built of a particular protocol that should ensure this goal, and any modelling assumptions made.

Consider, as an example, the case of the ASW fair exchange protocol [3] for contract signing.[3] A primary security property that this protocol should ensure is fairness: intuitively, that either both parties obtain valid contracts after execution of the protocol, or neither party does. This goal, as stated, would not currently fall within the scope of the AVISPA tool. The authors of [25], however, show via meta-reasoning about the final states of their model of ASW that the fairness goal could be reduced to a secrecy property. They analyse the resulting model with the OFMC back-end of the AVISPA tool and, indeed, uncover a subtle flaw that was previously unknown.

It is important to note that this result is specific to fairness not only as it is provided by ASW, but specifically in the context of the formalisation of ASW found in [25]. Thus, we can see that this meta-reasoning process need not be specific to a single goal, but can indeed involve reasoning about the goal at hand, a particular protocol, and a specific model of that protocol.

Reasoning about security properties as one understands them on these three distinct levels can be helpful in many respects. In addition to the examples we have already seen, this framework of understanding can also indicate if certain goals may be particularly challenging to model. Note that the three levels presented here nicely encapsulate much of the expressiveness of our formalism for modelling protocols. If modelling a particular security property requires one to reason about aspects of the goal that fit into none of these

[3]While ASW is designed to enable the exchange of multiple different types of items, we consider here only the contract signing case.

three levels, then this may be an indicator that this goal – or least certain characterisations of it – may be challenging to capture in our formalism. For example, as we discuss in §5, anonymity is a property that is often characterised epistemically; that is, by reasoning about what the intruder knows and believes. This notion of belief does not seem to fit into any of the three levels we have defined here, and so this may be an indicator that such characterisations of anonymity will be very challenging to bring into our formalism. That said, as we shall see, other characterisations of variants of anonymity exist which we can indeed capture.

3 Secrecy

3.1 Defining Secrecy

Secrecy is perhaps the most studied property in the security protocol analysis community. Informally, the secrecy of some data (in practise, generally a key or a nonce) is guaranteed if this data cannot be obtained by the intruder unless he is explicitly allowed to know it.

Considering a protocol as a transitions system `TS` and a protocol run as a trace of `TS`, one can easily check a safety property like secrecy. A protocol trace satisfies a secrecy property if

- either the intruder is allowed to know the secret,
- or the intruder is not allowed to know the secret and cannot deduce it during the protocol run.

Obviously, secrecy depends on the abilities of the intruder. Practically, the secrecy of one trace of a protocol is not significant. The objective is either to find a trace that does not satisfy the property (we call it an *attack trace*) or to guarantee that all traces satisfy the property. Since this problem is well known to be undecidable, we either bound the number of traces by bounding the number of parallel sessions or we use abstractions (see [6]) and semi-decision procedures (see [9]).

In the following subsection, we illustrate how to express these notions in our high-level specification language HLPSL.

3.2 Secrecy Goal in HLPSL

In HLPSL, a protocol designer can declare a secret and the agents allowed to know this secret by using facts as illustrated in the next subsection, while the modelling in HLPSL and IF is described in the two following subsections.

3.2.1 Secret Facts

As we pointed out in Section 3.1, a term can be declared to be a secret shared by several agents. In a role description, we use conjunctions of predicates of type `"secret(T,A)"` standing for ”the honest player of this role claims that the term T is a secret that the agent A is allowed to know”. A secret fact "secret(T,A)" is considered as a signal emitted during a transition firing. It is often useful to declare that the intruder is indeed allowed to know a term that might otherwise be secret. To this end, a secret term "secret(T,i)" is perfectly legal and can be interpreted as “publication” of the term T. That is, declaring that T is shared with the intruder i is tantamount to declaring it as public. In this way, we can model situations in which an agent communicates with the intruder under his own name; short-term secrets (such as session keys), which are initially declared secret and later divulged; and the explicit publication of some data.

Let us illustrate this by considering a simple example: the Encrypted Key Exchange (EKE) protocol (see Deliverable [10] for more details):

```
role EKE_Resp (B,A: agent,
               Kab: symmetric_key,
               Snd,Rcv: channel(dy)) played_by B def=

  local State : nat,
        Nb,K : text (fresh),
          Na : text,
          Ea : public_key

  init State = 0
  knowledge(B) = {A,B,Kab}

  transition

   1. State = 0 /\ Rcv({Ea'}_Kab) =|>
      State'= 1 /\ Snd({{K'}_Ea'}_Kab)
                /\ secret(K',A) /\ secret(K',B)

   2. State = 1 /\ Rcv({Na'}_K) =|>
      State'= 2 /\ Snd({Na'.Nb'}_K)
                /\ witness(B,A,nb,Nb')

   3. State = 2 /\ Rcv({Nb}_K) =|>
      State'= 3 /\ request(B,A,na,Na)

end role
```

In the EKE_Resp role, there are two secret terms specifying that A and B are allowed to know the key K. Later on the verification, tools will compose a list of secret terms that have to be verified for each state of the protocol.

3.2.2 HLPSL Secrecy Specification

Let us consider an execution of a protocol and a term declared to be secret, indicating that the intruder is not supposed to know it. Informally, we then say that the secrecy of the term is satisfied if, at the end of the run the intruder does not know and is not able to synthesise this term. As explained in Section 3.2.1, when a datum `m` is claimed to be secret, a corresponding predicate is set to true in a given transition. Thus, considering a transition system `TS` representing a protocol and a trace `tr` of `TS`, the secrecy of `m` is satisfied if for all states of `tr`,

- a signal concerning `m` has not yet been emitted in `tr`;
- `m` was claimed secret sometime in the past by a signal and the intruder knows `m` only if he is allowed to know it.

Formally, we say that a run of the protocol satisfies the secrecy of `X` if its trace satisfies the following linear temporal formula

```
[](<->secret(X,A) --> (<->secret(X,i) \/ not(iknows(X))))
```

The negative version representing a violated secrecy goal is written as follows:

```
<>(<->(secret(X,A))/\ iknows(X) /\ not(<->(secret(X,i))))
```

In natural language, this formula expresses that the secrecy of `X` is violated if the intruder knows or is able to build the secret `X` when he is not supposed to know it.

Notice that following the translation introduced in [19], it is possible to translate an HLPSL specification into a set of Temporal Logic of Actions formulae. In this context, for all terms M, the predicate iknows(M) is satisfied in a state if

$$\text{iknows}(M) \triangleq M \in \text{IK}\,,$$

where IK represents the intruder knowledge, in the sense that IK is the set of all terms that the intruder may generate from his knowledge (and according to his abilities). Thus, if iknows(m) is true in a state p, then iknows(m) is true in all states accessible from p.

3.3 IF Translation

The goal section of an IF specification expresses the violation of security properties by describing goal states. We have seen in §1 that, while goal events on the HLPSL level are true only during the transition in which they appear, IF goal facts are persistent unless explicitly removed from the current state by a rewrite rule. We can therefore provide the IF translation of the temporal formula given above, without loss of attacks, by simply removing the temporal operators. Thus, the IF translation is direct:

```
secrecy_of (M,X):=iknows(M).secret(M,X) & not(secret(M,i))
```

4 Authentication

4.1 Defining Authentication

As pointed out in [16], authentication is essentially assurance of who you are talking to. This can be made more specific in any number of ways. For example, you may want to make sure that those obtaining a session key are who they say they are, make sure that someone who has the key is currently on line, make sure that the principal you think has the key does have it, make sure that the principal with whom you think you are sharing the key also thinks he is sharing it with you, and so on.

There is not a unique definition of authentication that all secure protocols satisfy, and considerable effort, e.g. [16, 20, 21, 22, 29], has been devoted to specifying and classifying, semi-formally or formally, different forms of authentication such as: ping authentication, entity authentication, aliveness, weak agreement, non-injective agreement, injective agreement, and matching histories.

In Deliverable 6.1 [7], we identified different forms of authentication according to the following classification:

- **Authentication (unicast)**: Verifying an identity (distinguishing identifier) claimed by or for a system entity, which may be a peer in a communication or the source of some data. This assured identity may be well known (a real name, telephone number, mailing address, phone number, social security number, IP- or email address) or it can be an unlinkable identifier (like a pseudonym). The verification is achieved presenting authentication information (credentials) that corroborates the binding between the entity and the identifier. Authentication is usually divided into entity and message (or data) authentication. The main difference between the two is that message authentication provides no timeliness guarantee (the authenticated message may be old), while entity authentication implies actual communication with an associated verifier during execution of the current run of the protocol. Authentication is usually unilateral (e.g. "Alice authenticates Bob"). *Mutual Authentication* refers to Authentication in both directions.

    1. **Entity authentication (Peer Entity Authentication)**: The protocol must provide means of assuring one party, through presentation of evidence and/or credentials of the identity of a second party involved in a protocol, and that the second has actually participated during execution of the current run of the protocol. Usually this is done by presenting a piece of data that could only have been generated by the second party in question (as a response to a challenge, for instance). Thus, usually entity authentication implies that some data can be unequivocally traced back to a certain entity, which implies Data Origin Authentication.
    2. **Message authentication (Data Origin Authentication)**: The protocol must provide means to ensure confidence that a received message or piece of data has been created by a certain party at some (typically unspecified) time

    in the past, and that this data has not been corrupted or tampered with, but without giving uniqueness or timeliness guarantees. The confidence that data has been created by a certain party, but without the assurance that it has not been modified, is of no interest for us. Thus Message authentication implies integrity.

    3. **Replay Protection**: Some IETF documents define Replay Protection as "The protocol must provide means to ensure confidence that a received message has not been recorded and played back by an adversary". As such, this property is not verifiable. We define it rather as: Assuring one party that an authenticated message is not old. Depending on the context, this could have different meanings:
        - that the message was generated during this session, or
        - that the message was generated during a known recent time window, or
        - that the message has not been accepted before.

- **Authentication in Multicast or via a Subscribe / Notify Service**: These are the authentication requirements for groups with a single source and a very large number of potential recipients (multicast), or a source and a service which posts the information to subscribed (and authorised) users. The basic requirements for the solution are:
    1. **Implicit Destination Authentication** The protocol must provide means to ensure that a sent message is only readable by entities that the sender allows. That is, only legitimate authorised members will have access to the current information, multicast message or group communication. This includes groups with highly dynamic membership.
    2. **Source Authentication** Legitimate group members will be able to authenticate the source and contents of the information or group communication. This includes cases where group members do not trust each other.

- **Authorisation (by a Trusted Third Party)**: In some protocols, a Trusted Third Party **T3P** introduces one principal **B** to another principal **A** and **A** is assured that **B** is "trusted" by the **T3P** and is "authorised", in the required sense of the protocol. When the protocol is run between **A**, **B** and **T3P**, then **A** is perhaps not able to use local access control lists or other mechanisms to authorise **B**, (because the name of **B** is unknown to **A**, or may even be a pseudonym), but **A** is assured that **B** is authorised by **T3P**.

- **Key Agreement Properties**:
    1. **Key authentication** is the property whereby one party is assured that no other party aside from a specifically identified second party (and possibly additional identified trusted parties) may gain access to a particular secret key.

2. **Key confirmation (Key Proof of Possession)**: one party is assured that a second (possibly unidentified) party actually has possession of a particular secret key (or of all keying material needed to calculate it).
3. **Perfect Forward Secrecy (PFS)**: A protocol has this property if compromise of long-term keys does not compromise past session keys.
4. **Fresh Key Derivation** The protocol uses dynamic key management in order to derive fresh session keys.
5. **Secure capabilities negotiation (Resistance against Downgrading and Negotiation Attacks)**. When a key agreement protocol also discovers the cryptographic capabilities and preferences of the peers and negotiates the security parameters (such as security association identifiers, key strength, and ciphersuites), it is important to ensure that the announced capabilities and negotiated parameters have not been forged by an attacker.

We will now briefly summarise which kinds of authentication goals we have focused on and how we can specify them in HLPSL.

4.2 Authentication Goals in HLPSL

4.2.1 Goal Events: `witness` and `request`

We focus here on authentication in a unicast setting and on key agreement properties, which cover a large subset of the authentication properties relevant for the protocols of the AVISPA List of Problems and of the AVISPA Library [7, 10]. As described in detail in [4, 19], we call our goal events for authentication properties `witness` and `request`.

We illustrate their use with a simple example based on Microsoft's Challenge/Response Authentication Protocol, version 2 (CHAPv2 [33]). CHAPv2 is the authentication mechanism for the Point-to-Point Tunnelling Protocol (PPTP [24]), which itself is used to secure PPP connections over TCP/IP. Figure 1 shows the initiator role from a HLPSL model of CHAPv2. In this protocol, an initiator `A` and a responder `B` should authenticate one another based on a shared key (or password) `Kab` which both know in advance. To ensure freshness and preclude replay attacks, `A` and `B` exchange nonces called `Na` and `Nb`, respectively.

The second and third transitions are the most interesting when considering authentication. In the second, `A` generates a new value `Na'` for her nonce and sends it. She also flags a goal event `witness(A,B,na,Na')` to, in a sense, certify that she has created fresh data for `B`. Intuitively, we can read this event as follows: "agent `A` wants to execute the protocol with agent `B` and use value `Na'` her nonce." In order to specify for which variable of the protocol a particular value was meant (in this case the protocol variable `Na`), we specify a unique identifier for this variable (a so-called `protocol_id`), in this case `na`.

Similarly, in the third transition, the `request(A,B,nb,Nb)` goal event can be read, intuitively, "agent `A` accepts the value `Nb` and now relies on the guarantee that agent `B`

```
role CHAP_Init (A,B   : agent,
                Kab   : symmetric_key,
                H     : function,
                Snd,Rcv: channel(dy)) played_by A def=

  local State: nat,
        Na: text (fresh),
        Nb: text

  init   State = 0
  knowledge(A) = {A,B,Kab,Na}

  transition
   1. State=0 /\ Rcv(start) =|> Snd(A) /\ State'=1

   2. State=1 /\ Rcv(Nb') =|>
        Snd(Na'.H(Kab.Na'.Nb'.A)) /\
        State'=2 /\
        witness(A,B,na,Na')

   3. State=2 /\ Rcv(H(Kab.Na)) =|>
        State'=3 /\
        request(A,B,nb,Nb)

end role
```

Figure 1: The initiator role of CHAPv2

exists and agrees with her on this value for `protocol_id nb`." Thus, as of the point at which she raises this `request` event, `A` accepts `B`'s identity as authentic.

The placement of these goal events is crucial: in particular, `witness` facts issued too late or `request` facts raised too early can lead to trivial attacks. For instance, note that `A` could, in principal, issue her `request` event immediately upon receiving `B`'s nonce `Nb` in the second transition, as she has enough information to generate it. The success of the authentication, however, hinges on the exchange of proof of possession of `Kab`, which `A` cannot be certain of until she receives the message in the third transition. Were she to issue both her `witness` and `request` facts in the second transition, one can easily see that the intruder could produce trivial attacks by sending any value to `A`, and she would accept it as proof of authentication from `B`.

Thus, we can see that the modelling of security properties requires a precise understanding of the goals at hand: in this case, not merely what authentication means, but also the protocol-specific question of at what point does an agent accept his communication partner as "authenticated."

4.2.2 Temporal Formulae

CHAPv2 should provide entity authentication with replay protection, a combination which we often refer to as *strong authentication* and which Lowe calls *injective agreement* in [29]. In HLPSL, we write this using the following temporal formula.[4]

```
[](request(B,A,na,Na) --> (<->witness(A,B,na,Na) /\
                          not((-)<->request(B,A,na,Na))))
```

In natural language, this formula expresses that it is always true that a `request` event has been preceded by an accompanying `witness` event. Moreover, no agent should accept the same value twice from the same communication partner: that is, as of one time point before a `request` event, the same value had never been previously requested.

If replay protection is not guaranteed, we speak of *weak authentication*. This we express simply by removing the second conjunct of the consequent in the above formula as follows.

```
[](request(B,A,na,Na) --> <->witness(A,B,na,Na))
```

That is, each `request` event has been preceded by an accompanying `witness` event as before, but there is no guarantee that an intruder has not replayed values from a previous protocol run.

During the past months, we have analysed a number of protocols that have been designed to guarantee different forms of authentication and we have described the results of our analysis in various deliverables. A current list is given in the year 2 assessment of the AVISPA Tool [11].

5 Anonymity

Anonymity, in its intuitive understanding, is a quality of someone or something whose identity or origin is unknown. Protocols providing anonymity thus seek to hide the identities of communicating principals or sets of principals. In an increasingly privacy-aware Internet community, anonymity is of real practical interest.

Anonymity, as we will describe, exists in many variants. At its core, however, the primary concept – that of information hiding – bears an appealing resemblance to a secrecy property. Modelling anonymity in a very general sense (that is, attempting to capture all or even a large subset of the possible variants) poses several interesting research challenges, many of which remain open. Nevertheless, as we will see, the prospect of reducing anonymity to forms of secrecy goals holds promise for capturing many of the variants of anonymity that are relevant for the protocols of the AVISPA List of Problems [7].

5.1 The Challenge of Anonymity

Modelling and analysing anonymity presents several considerable challenges, a selection of which we outline here.

[4] In this formula, we introduce the symbol `(-)`, which should be read "one time point in the past."

Definitions As is the case for several security properties, it is difficult to provide an all-encompassing formal definition of anonymity itself. Rather, anonymity exists in several forms. Examples include the IKEv2 protocol [27], which hides the identities of communicating participants until an initial security association has been established, and Chaum's mix networks [17], which endeavour to obfuscate the origin and destination of network traffic from an outside observer.

Syverson and Stubblebine [31] observe that different types of anonymity can be differentiated on two primary criteria: the piece of information that should be protected and the nature of the protection provided. Examples of the former include protecting the identity of a principal involved in a protocol run, but also more complicated cases such as hiding from an intruder that two messages originate from the same sender.

The ability to reason about the origin of messages and the agent who performs a given action is not captured by the current version of the HLPSL/IF formalism. This relates to a richer notion of what an intruder knows that we describe below.

Characterisations Anonymity is a property that, as [31] shows, can be characterised elegantly using epistemic logic in which one can reason about the knowledge and beliefs of principals in a system, including the intruder. We note that "knowledge" in this context is used differently than we usually employ the term, which is in the sense of "intruder knowledge." Here, the term is used more broadly, and it includes the knowledge of, for instance, who performed a particular action or who said a given message. While the HLPSL/IF formalism can reason about intruder knowledge in a powerful way, it is yet unclear how one would incorporate epistemic statements about what the intruder "knows" (in this broader sense of knowledge) and believes.

The informal, three-layered understanding of goals that we describe in §2 indeed indicates that such epistemic characterisations of anonymity may pose significant challenges. In particular, epistemic notions of knowledge and belief can be captured neither on the first layer (that of the concrete messages an intruder can derive) nor on the second layer (that of the temporal properties of a given protocol) of our understanding. Thus, we can conclude that either a meta-reasoning step is required, or a consequential extension of our formalism and our understanding of goals — in a sense, the addition of a fourth layer of understanding — may be required in order to capture such epistemic aspects of anonymity. Indeed, the former may prove sufficient to model certain types of anonymity for certain protocols. We believe that the results of [1, 2, 14, 15] will provide a useful first step in the direction of such a meta-reasoning step.

Mechanisms Many mechanisms for ensuring anonymity are based on combining cryptography with a network topology that obscures identity information. Such mechanisms are difficult to model in a Dolev-Yao setting where one makes very drastic worst-case assumptions about the intruder's abilities. For instance, the seminal work of Chaum [17] is intended to solve the so-called traffic analysis problem; that is, "the problem of keeping confidential who converses with whom, and when they converse" [17]. Chaum proposed the use of mixes, or intermediary computers that route cryptographically-protected traffic before it is delivered to its final destination. Mixes employ various techniques to obscure the correlation between traffic entering the mix network and traffic leaving the mix network.

The modelling of such concrete mechanisms (and their associated anonymity goals) is beyond the scope of standard Dolev-Yao analysis. For example, the commonly used assumption that "the intruder is the network" essentially builds in a very strong form of traffic analysis. So strong, in fact, that one might call it "traffic direction," as *all* messages that are sent go directly to the intruder, and all messages that are received come directly from the intruder. Thus, messages need not even be addressed in the standard IP sense, because the intruder decides where each one goes. Moreover, the notion of dummy traffic, which is often used in mix networks to further obscure which parties are communicating, is also not applicable in a Dolev-Yao setting, given this notion of traffic direction. Any dummy messages would merely serve to increase the knowledge of a standard Dolev-Yao intruder.

We can see that the notion of traffic analysis is of questionable relevance in a setting in which the intruder controls the entire network.

5.2 Anonymity Protocols and AVISPA

Anonymity is thus a non-trivial property to model. In addition to there being many differing definitions of the goal itself, we have seen that (at least parts of) an important characterisation of the property and an important mechanism for its implementation are both currently beyond the scope of our formalism. That being said, we observe that many protocols of the AVISPA List of Problems [7] do not provide anonymity, since a principal has to know with whom he is speaking in order to determine which cryptographic key to use. Those protocols that do offer identity protection offer it in the following two variants, as described in [7].

1. **Identity Protection against Eavesdroppers**: An attacker (eavesdropper) should not be able to link the communication exchanged by one party to the real identity of the party.

2. **Identity Protection against Peer**: The peer in a communication should not be able to link the communication exchanged by one party to the real identity of the party, but rather to an unlinked pseudonym or private identifier.

The former variant reduces elegantly to a secrecy goal if we assume all sessions of a protocol are between honest agents.[5] In this setting, we can analyse identity protection against 1. by simply declaring the names of the honest agents as secret. Assuming these agent names are not in the initial intruder knowledge, then this secrecy goal will be violated

[5]Note that this does not imply that the intruder is passive, merely that none of the honest agents wish to participate in a session with the intruder under his real name.

if and only if there is a violation of the identity protection goal. The latter variant, which can be seen as a stronger version of the former, can also be captured in this way.

The the case of the former variant in which the intruder participates in sessions with honest agents under his real name is, however, more challenging and cannot be reduced to secrecy goals in a straightforward way. The reasons for this relate to the rather restricted notion of "knowledge" inherent in the Dolev-Yao intruder model. Consider, for instance, two sessions of a given protocol in parallel: one between honest agents Alice and Bob, and one between Alice and the intruder, who plays under his real name. If we wish to analyse 1., then the conflicting situation arises in which the identity of Alice should remain hidden from the intruder in the first session but may be revealed to the intruder in the second. We can thus see that, under the standard Dolev-Yao notion of knowledge, the intruder learning Alice's identity in the second session (which is not a violation of Alice's anonymity against eavesdroppers, as she intended to communicate with the intruder) immediately violates Alice's anonymity in all other sessions if we attempt a straightforward reduction to secrecy. A helpful feature here would be the ability to reason not only about whether the intruder knows some data or not, but also correlations between the data and the sessions to which they belong.

5.3 Promising Avenues

Despite these challenges, work to incorporate the analysis of at least certain forms of anonymity shows promise.

Pseudonymity. In practice, anonymity is often achieved through the use of digital pseudonyms (see, e.g., [17]). Here, a party, say P, who wishes to remain anonymous generates an asymmetric key pair. The public key, which he then publishes, is the digital pseudonym via which others will refer to P, using it to encrypt messages which only P can then decrypt and also to verify signatures on messages purporting to originate from the pseudonymous party. P's anonymity is ensured by the fact that there should be no correlation between his identity and the key pair generated. Indeed, in frameworks like Purpose-Built Keys (PBK [13]), parties generate fresh key pairs for each entity with whom they communicate.

HLPSL already offers elegant means of modelling precisely such situations. Namely, functions can serve to map agent names to pseudonyms, as they are not invertible by the intruder. Thus, so long as the intruder does not know the pseudonym function, say `pn`, itself, he cannot find any relation between an agent name `a` and its pseudonym `pn(a)`. There still arises a problem of how the intruder can correlate names and pseudonyms (again related to the restricted Dolev-Yao notion of intruder knowledge), but we are confident that pseudonymity mechanisms should be elegantly expressible in HLPSL.

Anonymity and Guessing. Other areas of protocol analysis are similarly restricted by the "all or nothing" notion of intruder knowledge in the Dolev-Yao model that we described in the previous section. For instance, the modelling of a guessing intruder [8, 26] requires an extended notion of intruder knowledge to capture messages that have been guessed but not yet verified: that is, messages built up of guesses where the intruder knows that the values for the guesses are in his dictionary but has not yet verified which entries are the correct values of the guesses. This is quite similar in characterisation to some of the condens, or types of anonymity protection, described in [31]: for instance those that state that an intruder attempting to discover who performed a given action may be able to narrow it down to a set of candidate agents (a dictionary of agents, so to speak), but should not be able to determine which of those agents it was. Anonymity thus bears some appealing similarity to guessing, but we leave the investigation of these similarities for future work.

6 Non-Repudiation

6.1 Introduction

While security issues such as secrecy and authentication have been studied intensively, non-repudiation protocols began raising interest only in recent years, most notably in the early 1990s with the explosion of Internet services and electronic transactions. Non-repudiation services must ensure that when two parties exchange informations over a network, neither one nor the other can later deny having participated in this communication. Consequently, a non-repudiation protocol has to generate evidence of participation to be used in the 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 irrefutable evidence of the participation and the identity of his party.

In the cases where the evidence is provided to both parties, the protocol might also aim to provide *fairness*, i.e. 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 required for non-repudiation, but it is usually desirable.

The first solutions providing fairness in exchange protocols were based on a gradual exchange of the expected information [28]. 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 unlikely in a real world scenario. Therefore, the solution we will focus on here is the adoption of a *Trusted Third Party* (TTP). The TTP can be used as a delivery agent to provide simultaneous sharing of evidence.

A typical non-repudiation protocol can provide a number of different non-repudiation services, like non-repudiation of origin, non-repudiation of receipt and fairness, but the actual non-repudiation services provided by a protocol depend mainly on its application. From the existing applications, we can distinguish the following non-repudiation services:

Non-repudiation of origin (NRO) provides the recipient with the evidence NRO which ensures that the originator will not be able to deny having sent the message. The evidence of origin is generated by the originator and held by the recipient.

Non-repudiation of receipt (NRR) provides the originator with the evidence NRR which ensures that the recipient will not be able to deny having received the message. The evidence of receipt is generated by the recipient and held by the originator.

Non-repudiation of submission (NRS) is intended to provide evidence that the originator submitted the message for delivery. 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.

Non-repudiation of delivery (NRD) is intended to provide evidence that the recipient received the message. This service also only applies when the protocol uses a TTP. Evidence of delivery is generated by the delivery agent, and will be held by the originator.

Fairness is achieved for a non-repudiation protocol if at the end of the protocol execution either the originator has the evidence of receipt for the message M and the recipient has the evidence of origin and the corresponding message M or neither of them have any valuable information.

The evidence NRO, NRR, NRS and NRD are messages signed by an agent. In the following subsection, we will describe a fair non-repudiation protocol that uses a TTP. This case study will show more precisely what are the evidence. Moreover we will show how the non-repudiation goals of the protocol can be modelled using authentication. This will permit us to specify the protocol in HLPSL and analyse it using the AVISPA Tool.

6.2 A Fair Non-repudiation Protocol

We haven chosen the Fair Zhou-Gollmann protocol (FairZG) as a case study to demonstrate our analysis approach of non-repudiation. (One of the motivations for our choice is the existence of significant related work [12, 23, 30, 32].)

The protocol is presented below in Alice&Bob-style notation, where fNRO,fNRR,fSUB and fCON are labels used to identify the purpose of the messages (a graphical representation is given in Figure 2):

1.	$\mathrm{A} \rightarrow \mathrm{B}$:	$\mathrm{fNRO}, \mathrm{B}, \mathrm{L}, \mathrm{C}, \mathrm{NRO}$
2.	$\mathrm{B} \rightarrow \mathrm{A}$:	$\mathrm{fNRR}, \mathrm{A}, \mathrm{L}, \mathrm{NRR}$
3.	$\mathrm{A} \rightarrow \mathrm{TTP}$:	$\mathrm{fSUB}, \mathrm{B}, \mathrm{L}, \mathrm{K}, \mathrm{subK}$
4.	$\mathrm{B} \leftrightarrow \mathrm{TTP}$:	$\mathrm{fCON}, \mathrm{A}, \mathrm{B}, \mathrm{L}, \mathrm{K}, \mathrm{conK}$
5.	$\mathrm{A} \leftrightarrow \mathrm{TTP}$:	$\mathrm{fCON}, \mathrm{A}, \mathrm{B}, \mathrm{L}, \mathrm{K}, \mathrm{conK}$

where

- A denotes the originator of the message M;

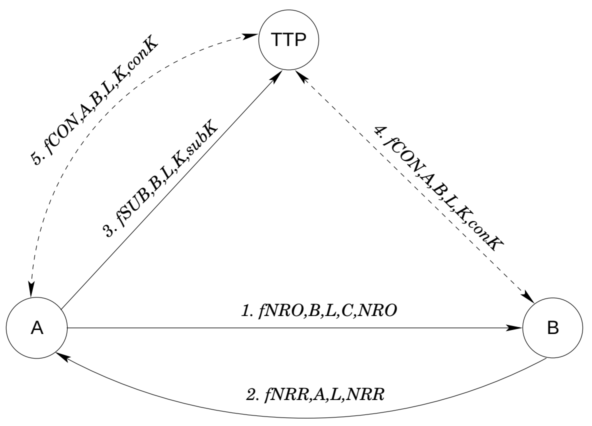

Figure 2: Fair Zhou Gollmann Non-repudiation Protocol

- B denotes the recipient of the message M;
- TTP denotes the trusted third party;
- M is the message sent from A to B;
- C is the commitment, the message $\{M\}_K$;
- L is a connection identifier, a nonce;
- K is a key defined by A;
- NRO is the evidence for non-repudiation of origin, the message $\{fNRO, B, L, C\}_{inv(KA)}$;
- NRR is the evidence for non-repudiation of receipt, the message $\{fNRR, A, L, C\}_{inv(KB)}$;
- subK is proof of submission of K, the message $\{fSUB, B, L, K\}_{inv(KA)}$; and
- conK represents confirmation of K, the message $\{fCON, A, B, L, K\}_{inv(KTTP)}$.

The main idea of the 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 A and B. Once A has an evidence of commitment from Bob (NRR), the key K is sent to a trusted third party. Once the TTP has received the key, both A and B can retrieve the evidence ConK and the key K from the TTP. This last step is represented by a double direction arrow in the Alice&Bob-style notation because it is implementation specific and may be composed by several message exchanges. In this scenario, we assume that the network will not be down forever and that both A and B have access to the TTP's shared repository where the evidence ConK and the key K are stored. This means that the agents will eventually be able to retrieve the key and the evidence from the TTP even in case of network failures.

6.3 Non-repudiation Goals

6.3.1 Non-repudiation of Origin

The property of non-repudiation of origin requires a guarantee that an agent sent some particular message. In the FairZG protocol, non-repudiation of origin should provide the guarantee that A sent a message to B containing the label L and the ciphertext C, and another message containing the same label L and the key K. If these two messages are signed by A and successfully received by B, they provide irrefutable evidence that A sent M to B.

Therefore to verify that the protocol FairZG ensures the non-repudiation of origin property, one needs to be sure that the agent B received L, C and L, K from A. We can have this assurance by doing three authentications:

- B authenticates A on NRO: NRO is the message $\{fNRO, B, L, C\}_{inv(KA)}$. The signature will assure A's identity and serve as evidence to B that A sent L and the commitment C. The authentication guarantees that L and C came from A and not from an intruder.
- TTP authenticates A on SubK: SubK is the message $\{fSUB, B, L, K\}_{inv(KA)}$. The authentication guarantees that the key K being sent in this message and stored in the TTP's directory came from A and no one else.
- B authenticates TTP on ConK: ConK is the message $\{fCON, A, B, L, K\}_{inv(KTTP)}$. The authentication guarantees that the key K received by B really came from the TTP.

From the above authentications, B can show NRO and ConK as evidence that A sent M and an external judge can do the following reasoning to testify:

1. The messages L and C that are part of the signed message NRO came from A.
2. The key K that is part of the message SubK and that was stored in the TTP's directory came from A.
3. The key K retrieved by B on the message ConK was the one stored previously on the TTP's directory.
4. Therefore, $M = \{C\}_K$ came from A

To illustrate these authentication goals, we adopt an informal notation reminiscent of a belief logic to express statements about the originators and recipients of messages. As we have seen in §5, epistemic statements like those found in belief logics are difficult to integrate into the HLPSL framework. This, therefore, is not meant as a formal definition, but rather for illustrative purposes. In this notation, we can write the above authentications at a lower level as a predicate comprising witnesses and requests and deduce from it that B has all the messages and evidence he needs to invalidate any repudiation from A, namely NRO and ConK.

Definition 1. *Given a trace of the FairZG protocol execution tr,*

Non-Repudiation of Origin(tr) =

$$
\begin{aligned}
&(\ \textit{witness}\ (\mathrm{A}, \mathrm{B}, \mathrm{nor}, \{\mathrm{fNRO.B.L.C}\}_{\mathrm{inv(K_A)}}) \subset \textit{tr} \wedge \\
&\ \ \textit{request}\ (\mathrm{B}, \mathrm{A}, \mathrm{nor}, \{\mathrm{fNRO.B.L.C}\}_{\mathrm{inv(K_A)}}) \subset \textit{tr} \wedge \\
&\ \ \textit{witness}\ (\mathrm{A}, \mathrm{TTP}, \mathrm{sub}, \{\mathrm{fSUB.B.L.K}\}_{\mathrm{inv(K_A)}}) \subset \textit{tr} \wedge \\
&\ \ \textit{request}\ (\mathrm{TTP}, \mathrm{A}, \mathrm{sub}, \{\mathrm{fSUB.B.L.K}\}_{\mathrm{inv(K_A)}}) \subset \textit{tr} \wedge \\
&\ \ \textit{witness}\ (\mathrm{TTP}, \mathrm{B}, \mathrm{con}, \{\mathrm{fCON.A.B.L.K}\}_{\mathrm{inv(K_{TTP})}}) \subset \textit{tr} \wedge \\
&\ \ \textit{request}\ (\mathrm{B}, \mathrm{TTP}, \mathrm{con}, \{\mathrm{fCON.A.B.L.K}\}_{\mathrm{inv(K_{TTP})}}) \subset \textit{tr}\ ) \\
&\ \ \Rightarrow (\mathrm{A}\ \textit{sent}\ \{\mathrm{fNRO.B.L.C}\}_{\mathrm{inv(K_A)}} \wedge \\
&\qquad \mathrm{B}\ \textit{received}\ \{\mathrm{fNRO.B.L.C}\}_{\mathrm{inv(K_A)}} \wedge \\
&\qquad \mathrm{A}\ \textit{sent}\ \{\mathrm{fSUB.B.L.K}\}_{\mathrm{inv(K_A)}} \wedge \\
&\qquad \mathrm{TTP}\ \textit{received}\ \{\mathrm{fSUB.B.L.K}\}_{\mathrm{inv(K_A)}} \wedge \\
&\qquad \mathrm{TTP}\ \textit{sent}\ \{\mathrm{fCON.A.B.L.K}\}_{\mathrm{inv(K_{TTP})}} \wedge \\
&\qquad \mathrm{B}\ \textit{received}\ \{\mathrm{fCON.A.B.L.K}\}_{\mathrm{inv(K_{TTP})}}\ )
\end{aligned}
$$

6.3.2 Non-repudiation of Receipt

The non-repudiation of receipt property requires a guarantee that an agent received some particular message. In the FairZG protocol, non-repudiation of receipt should provide the guarantee that B received one message with the label L and the ciphertext C, and another message containing the same label L and the key K. If A has evidence that B received these messages, than B can not deny having received them. The evidence for A are NRR (the commitment from B) and ConK (the certificate from the TTP).

To verify that the protocol FairZG ensures the non-repudiation of receipt, one needs to be sure that the agent A has both NRR and ConK. This task can be accomplished by the analysis of three authentication problems:

- A authenticates B on NRR: NRR is the message $\{\mathrm{fNRR}, \mathrm{A}, \mathrm{L}, \mathrm{C}\}_{\mathrm{inv(KB)}}$. The authentication guarantees that the label L and the commitment C were received by B and B's signature provides the required evidence of B's identity.
- TTP authenticates A on SubK: SubK is the message $\{\mathrm{fSUB}, \mathrm{B}, \mathrm{L}, \mathrm{K}\}_{\mathrm{inv(KA)}}$ that will provide the certitude that the key K stored in the TTP came from A, i.e, the channel with the TTP is secure.
- A authenticates TTP on ConK: ConK is the message $\{\mathrm{fCON}, \mathrm{A}, \mathrm{B}, \mathrm{L}, \mathrm{K}\}_{\mathrm{inv(KTTP)}}$. If A can retrieve the message ConK then B received or has access to the same message and consequently knows the key K to decrypt the message C.

Here, even if B did not take the key from TTP's repository, the non-repudiation of receipt is assured thanks to our assumption of the agents eventually having access to the repository even in case of network failures. The key is available, it is up to B to retrieve it or not.

From the above authentications an external judge can conclude:

1. The messages L and C that are part of the signed message NRR have been received by B.
2. The key K that is part of the message SubK and that was stored in the TTP's repository came from A.
3. The key K is available to B on the message ConK stored in the TTP's repository.
4. B can deduce $M = \{C\}_K$

As before, we can write the above authentications at a lower level and deduce using our informal notation, inspired by belief logic, that A has all the messages and evidence he needs to invalidate any repudiation from B, namely NRR and ConK.

Definition 2. *Given a trace of the FairZG protocol execution tr,*

Non-Repudiation of Origin(tr) =

$$\begin{aligned}
&(\ \textit{witness}\ (B, A, nrr, \{fNRR.A.L.C\}_{inv(K_B)}) \subset \textit{tr} \land \\
&\ \ \textit{request}\ (A, B, nrr, \{fNRR.A.L.C\}_{inv(K_B)}) \subset \textit{tr} \land \\
&\ \ \textit{witness}\ (A, TTP, sub, \{fSUB.B.L.K\}_{inv(K_A)}) \subset \textit{tr} \land \\
&\ \ \textit{request}\ (TTP, A, sub, \{fSUB.B.L.K\}_{inv(K_A)}) \subset \textit{tr} \land \\
&\ \ \textit{witness}\ (TTP, A, con, \{fCON.A.B.L.K\}_{inv(K_{TTP})}) \subset \textit{tr} \land \\
&\ \ \textit{request}\ (A, TTP, con, \{fCON.A.B.L.K\}_{inv(K_{TTP})}) \subset \textit{tr}\ ) \\
&\ \ \Rightarrow (B\ \textit{sent}\ \{fNRR.A.L.C\}_{inv(K_B)} \land \\
&\ \ \quad A\ \textit{received}\ \{fNRR.A.L.C\}_{inv(K_B)} \land \\
&\ \ \quad A\ \textit{sent}\ \{fSUB.B.L.K\}_{inv(K_A)} \land \\
&\ \ \quad TTP\ \textit{received}\ \{fSUB.B.L.K\}_{inv(K_A)} \land \\
&\ \ \quad TTP\ \textit{sent}\ \{fCON.A.B.L.K\}_{inv(K_{TTP})} \land \\
&\ \ \quad A\ \textit{received}\ \{fCON.A.B.L.K\}_{inv(K_{TTP})}\ )
\end{aligned}$$

6.3.3 Fairness

In the literature, different definitions of fairness for non-repudiation protocols are considered. In some definitions, none of the parties should have more evidence than the other at any given point in time. Other definitions are more flexible and require that none of the parties should have more evidence than the others at the end of a protocol run (that was the definition used in Section 6.1). It is also often left unclear if only complete protocol runs are taken into account or whether partial protocol runs are to be considered as well. As this distinction has an impact in the automatic analysis methods used by the AVISPA tools, we make the following distinction between the definitions:

Weak Fairness is achieved for a non-repudiation protocol if all the obligatory steps of the protocol are executed and at the end of the protocol execution either the originator has the evidence NRR for the message M and the recipient has the evidence NRO and the corresponding message M or none of then have any valuable information.

Strong Fairness is achieved for a non-repudiation protocol if at any given time, independently of how the protocol is executed, either the originator has the evidence NRR for the message M and the recipient has the evidence NRO and the corresponding message M or none of then have any valuable information.

We can define weak fairness for the FairZG protocol as a function of non-repudiation of origin and of non-repudiation of receipt. If we have both non-repudiations for a given trace *tr*, then we have weak fairness, i.e. at the end of the protocol run, A and B have their required evidence and they can not deny having participated of any message exchange.

We can also write this definition as follows:

Definition 3. *Given a trace of the protocol execution tr,*

Weak Fairness(tr) =
Non-Repudiation of Origin(tr) ∧
Non-Repudiation of Receipt(tr)

Unfortunately, using our current state-based model of the protocol we cannot reason about strong fairness. The goal, as an invariant of the state system, has to be evaluated at every system state, and there will always be a state in which, for example, A has the evidence she needs and B is yet to get the key K from the TTP. In this scenario, we do not have strong fairness because A has the evidence ConK stating that B received K but B do not have the key. Since the above scenario will eventually happen the strong fairness property will always be false regardless of the protocol.

6.4 Specification of a Non-repudiation Protocol

In this subsection, we discuss the HLPSL specification of the FairZG protocol. Alice is the sender and Bob is the receiver of the message. The part of the protocol responsible for retrieving the key K from the TTP was modelled by system states without events (step 3); Alice sends the key K to the TTP in the second protocol rule and from this point on she can ask for her evidence ConK at any time.

```
role Alice ( A,B,S : agent,
             Ka,Kb,Ks : public_key,
             K: symmetric_key,
             Hash: function,
             Snd, Rcv : channel(dy)) played_by A def=
  local State : nat,
        M, L  : text(fresh),
        C, NRO, NRR, SubK, ConK : message
  owns Snd
  init State = 0
  knowledge(A) = {A,B,S,Ka,Kb,Ks,inv(Ka),fNRO,fNRR,fSUB,fCON}

  transition
   1. State=0 /\
      Rcv(start)
      =|>
      C' = {M'}_K /\
      L' = Hash(M',K) /\
      NRO' = {fNRO.B.L'.C'}_inv(Ka) /\
      Snd(fNRO.B.Hash(M',K).{M'}_K.NRO') /\
      State'=1 /\
      % Non-repudiation of Origin:
      witness(A,B,nro,NRO')

   2. State=1 /\
      Rcv(fNRR.A.L.NRR') /\
      NRR' = {fNRR.A.L.C}_inv(Kb)
      =|>
      SubK' = {fSUB.B.L.K}_inv(Ka) /\
      Snd(fSUB.B.L.K.SubK') /\
      State'=2 /\
      % Non-repudiation of Submission:
      witness(A,S,sub,SubK')

   3. State=2
      --|>
      Snd(fREQ.A.B.L) /\
      State'=3

   4. State=3 /\
      Rcv(fCON.A.B.L.K.ConK') /\
      ConK'={fCON.A.B.L.K}_inv(Ks)
      =|>
      State'=4 /\
      % Non-repudiation of Delivery:
      request(A,S,con,ConK') /\
      % Non-repudiation of Receipt:
      request (A,B,nrr,NRR)
end role
```

The role Bob works in a similar way as the role Alice. Bob can request ConK to the TTP as soon as he exchanged the commitment C with Alice. The TTP, however, will only be able to answer when he receives the key K from Alice.

```
role Bob (B,A,S : agent,
          Kb,Ka,Ks : public_key,
          Snd, Rcv : channel (dy)) played_by B def=
  local State: nat, M, L: text,
        K: symmetric_key,
        C, NRO, NRR, ConK: message
  owns Snd
  init State = 0
  knowledge(B) = {B,A,S,Ka,Kb,Ks,inv(Kb),fNRO,fNRR,fSUB,fCON}

  transition
   1. State=0 /\
      Rcv(fNRO.B.L'.C'.NRO') /\
      NRO'={fNRO.B.L'.C'}_inv(Ka)
      =|>
      NRR'={fNRR.A.L'.C'}_inv(Kb) /\
      Snd(fNRR.A.L'.NRR')  /\
      State'=1  /\
      % Non-repudiation of Receipt:
      witness (B,A,nrr,NRR')

   2. State=1
      --|>
      Snd(fREQ.A.B.L) /\
      State'=2

   3. State=2 /\
      Rcv(fCON.A.B.L.K'.ConK') /\
      ConK'={fCON.A.B.L.K'}_inv(Ks) /\
      C = {M'}_K'
      =|>
      State'=3 /\
      % Non-repudiation of Delivery:
      request(B,S,con,ConK') /\
      % Non-repudiation of Origin:
      request(B,A,nro,NRO)
end role
```

The task of the role Server (TTP) is to receive the key K from Alice and send the evidence ConK to both parties upon request.

```
role Server (S,A,B : agent,
             Ks,Ka,Kb : public_key,
             Snd, Rcv : channel (dy)) played_by S def=
  local State: nat,
        L: text,
        K: symmetric_key,
        SubK, ConK : message
  owns Snd
  init State = 0
  knowledge(S) = {S,A,B,Ks,Ka,Kb,inv(Ks),fREQ,fNRO,fNRR,fSUB,fCON}

  transition
   1. State=0 /\
      Rcv(fSUB.B.L'.K'.SubK') /\
      SubK'={fSUB.B.L'.K'}_inv(Ka)
      =|>
      State'=1

   2. State=1 /\
      Rcv(fREQ.A.B.L)
      =|>
      ConK'={fCON.A.B.L.K}_inv(Ks) /\
      Snd(fCON.A.B.L.K.ConK') /\
      State'=2 /\
      % Non-repudiation of Delivery:
      witness (S,A,con,ConK') /\
      witness (S,B,con,ConK') /\
      % Non-repudiation of Submission:
      request (S,A,sub,SubK)
end role
```

6.5 Protocol Analysis

The analysis was made with 4 parallel sessions of the protocol. The intruder plays the role of Alice, Bob and the TTP in the last three sessions as shown below. This represents all the combinations of how the intruder can step in the protocol.

```
role Environment() def=

 knowledge(i) = {a,b,s,ka,kb,ks,ki,i,inv(ki),fNRO,fNRR,fSUB,fCON}

 composition
       Session(a,b,s,ka,kb,ks,k,h,sa1,ra1,sb1,rb1,ss1,rs1)
    /\ Session(a,i,s,ka,ki,ks,k,h,sa2,ra2,si2,ri2,ss2,rs2)
    /\ Session(i,b,s,ki,kb,ks,k,h,si3,ri3,sb3,rb3,ss3,rs3)
    /\ Session(a,b,i,ka,kb,ki,k,h,sa4,ra4,sb4,rb4,si4,ri4)
end role
```

Each session is composed by an instance of Alice, Bob and the TTP.

```
role Session(A,B,S: agent,
             Ka,Kb,Ks: public_key,
             K: symmetric_key,
             H: function,
             SA,RA,SB,RB,SS,RS: channel (dy)) def=

  composition
    Alice(A,B,S,Ka,Kb,Ks,K,H,SA,RA) /\
    Bob(B,A,S,Kb,Ka,Ks,SB,RB) /\
    Server(S,A,B,Ks,Ka,Kb,SS,RS)
end role
```

We were able to automatically verify the FairZG protocol for non-repudiation of origin and non-repudiation of receipt. Since no attacks were found on the specified scenario we can also guarantee the weak fairness property (according to Definition 3). The specified scenario consisted of a typed analysis of 4 parallel sessions of the protocol over the Dolev-Yao intruder model.

6.6 Discussion

To analyse a non-repudiation protocol the modeller only needs to add a pair *witness/request* for the events that characterise the different non-repudiation events, i.e. add a *witness* when Alice sends the NRO evidence, add a *request* in the end of Bob's role, when he is supposed to have the evidence NRO, add a *witness* when Bob sends the NRR evidence and a corresponding *request* in Alice's role, and so forth. This schema was used to analyse the FairZG protocol and it looks to be easily carried over to other non-repudiation protocols.

Although the Dolev-Yao intruder model is not the ideal model to analyse non-repudiation protocols, it can be used to automatically verify the properties of non-repudiation of origin, non-repudiation of receipt and weak fairness using an authentication-like model of the protocol goals.

Some restrictions apply to this model: all protocols traces are traces where the protocol finishes. It is not possible to simulate an unresponsive agent using this model and then we cannot find attacks in which some agent does not respond to a message when he is supposed to, like the attack described by [23].

The main results we obtained are the modelling of non-repudiation properties as authentication properties and the complete automatic analysis of the FairZG protocol. Further work should incorporate a dishonest agent model or unresponsive agent model to be able to completely analyse all aspects of non-repudiation protocols.

7 Conclusion

As we have seen, security properties are the goals against which we check our formal models of security protocols. Indeed, modelling the goals of a protocol can be a very challenging aspect of building a formal specification. Sometimes, a generally agreed-upon formal definition of the security property at hand may not even exist.

As we strive to develop efficient techniques for the automated analysis of increasingly complex security properties, it is advantageous to reduce goals to Boolean or temporal combinations of other goals for which efficient analysis methods already exist. We have seen examples of such reductions that show that this technique is quite effective, particularly where there is a strong similarity between the intuitions behind two security properties, as is, for instance, the case between secrecy and anonymity.

Challenging open problems remain in the formalisation of complex security properties. We have touched on some of these in relation to the security properties already discussed, and we now briefly describe two more interesting properties that we intend to study in future work.

Sender Invariance: In mobile settings in which communicating parties share no previous knowledge of each other's identities and no cryptographically authenticated data (i.e. public keys), classical authentication is an infeasible goal. In protocols like the binding update subprotocol of the Mobile IP suite, one therefore tries to ensure a weaker goal: namely, that after an initial setup phase between a sender and a receiver, the receiver can be sure that all subsequent communication originates from the same sender. This property is called *sender invariance.* To our knowledge, no formal definition of sender invariance has been formulated. Here, a definition based on a reduction to some form of authentication [29] seems promising, as sender invariance is closely related to a weak form of authentication.

Denial of Service: Denial of Service (DoS) attacks can arise if an attacker can bring a server to exhaust some resource (memory or CPU time, for instance), leading to an interruption of service to the machine's legitimate clients. DoS vulnerabilities are frequently exploited, often at great cost to the individual or corporation who is attacked, and are therefore of significant practical interest. Mounting such an attack is only attractive if the attacker must invest few computational resources himself (at least relative to those that bring the server to exhaust its capacity). To analyse denial of service, we therefore plan to develop methods based on assigning a computational expense to operations: for instance, calculating a cryptographic hash is certainly "less expensive" than performing a public-key encryption. Variants of DoS should then be expressible in terms of the computational costs to an intruder relative to those of an agent under attack. If the intruder can, for low computational expense, bring the agent to perform actions of high computational expense, then this may represent a DoS vulnerability.

Here again, particularly as the case of DoS, we can see that our framework for understanding goals gives us a clue that modelling DoS may require significant extensions to our formalism, as the notion of computational cost fits on none of our three levels of reasoning.

References

[1] R. Accorsi, D. Basin, and L. Viganò. Towards an awareness-based semantics for security protocol analysis. In J. Goubault-Larrecq, editor, *Proceedings of LACPV'01*, ENTCS 55(1). Elsevier Science Publishers, Amsterdam, 2001.

[2] R. Accorsi, D. Basin, and L. Viganò. Modal specifications of trace-based security properties. In *Proceedings of the Second International Workshop on Security of Mobile Multiagent Systems (SEMAS-2002)*, pages 1–11. Research Report DFKI-RR-02-03, DFKI Kaiserslautern/Saarbrücken, Germany, 2002.

[3] N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 86–99, Oakland, CA, USA, May 1998. IEEE Computer Society Press.

[4] AVISPA. Deliverable 2.1: The High-Level Protocol Specification Language. Available at `http://www.avispa-project.org`, 2003.

[5] AVISPA. Deliverable 2.3: The Intermediate Format. Available at `http://www.avispa-project.org`, 2003.

[6] AVISPA. Deliverable 5.1: Abstractions. Available at `http://www.avispa-project.org`, 2003.

[7] AVISPA. Deliverable 6.1: List of selected problems. Available at `http://www.avispa-project.org`, 2003.

[8] AVISPA. Deliverable 3.2: Assumptions on Environment. Available at `http://www.avispa-project.org`, 2004.

[9] AVISPA. Deliverable 5.2: Infinite-State Model-Checking. Available at `http://www.avispa-project.org`, 2004.

[10] AVISPA. Deliverable 6.2: Specification of the Problems in the High Level Protocol Specification Language. Available at `http://www.avispa-project.org`, 2004.

[11] AVISPA. Deliverable 7.3: Assessment of the AVISPA tool v.2. Available at `http://www.avispa-project.org`, 2004.

[12] G. Bella and L. C. Paulson. Mechanical proofs about a non-repudiation protocol. *Lecture Notes in Computer Science*, 2152:91+, 2001.

[13] S. Bradner, A. Mankin, and J. I. Schiller. A framework for purpose built keys (PBK), June 2003. Work in Progress (Internet Draft).

[14] C. Caleiro, L. Viganò, and D. Basin. Towards a metalogic for security protocol analysis. In *Proc. Workshop on the Combination of Logics: Theory and Applications (Comblog'04)*. Instituto Superior Técnico, Lisbon, 2004. Available at `http://www.cs.math.ist.utl.pt/comblog04/`.

[15] C. Caleiro, L. Viganò, and D. Basin. Metareasoning about Security Protocols using Distributed Temporal Logic. In *Proc. IJCAR'04 Workshop on Automated Reasoning for Security Protocol Analysis (ARSPA'04)*. ENTCS, to appear. Available at `http://www.avispa-project.org/arspa/workshop-index.html`.

[16] I. Cervesato and P. F. Syverson. The logic of authentication protocols. In *Foundations of Security Analysis and Design*, LNCS 2171, pages 63–136. Springer-Verlag, 2001.

[17] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *CACM*, 24(2):84–88, Feb. 1981.

[18] 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*, volume 180 of *Automated Software Engineering*, pages 193–205. Austrian Computer Society, Austria, September 2004.

[19] 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 *Proc. SAPS'04*. Austrian Computer Society, 2004.

[20] J. Clark and J. Jacob. A survey of authentication protocol literature : Version 1.0., November 1997.

[21] D. Gollmann. What do we mean by Entity Authentication? In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 46–54. IEEE Computer Society Press, May 1996.

[22] D. Gollmann. Authentication – myths and misconceptions. In *Progress in Computer Science and Applied Logic*, volume 20. Birkhäuser Verlag, 2001.

[23] S. Gürgens, C. Rudolph, and H. Vogt. On the security of fair non-repudiation protocols. In *ISC*, pages 193–207, 2003.

[24] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, and G. Zorn. RFC 2637: Point-to-Point Tunneling Protocol, July 1999. Status: Informational.

[25] P. Hankes Drielsma and S. Mödersheim. The asw protocol revisited: A unified view. In *Proceedings of the IJCAR04 Workshop ARSPA*, 2004. To appear in ENTCS, available at `http://www.avispa-project.org`.

[26] P. Hankes Drielsma, S. Mödersheim, and L. Viganò. A formalization of off-line guessing for security protocol analysis. In *Proceedings of the 11th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, to appear.

[27] C. Kaufman. Internet Key Exchange (IKEv2) Protocol, Oct. 2003. Work in Progress.

[28] S. Kremer, O. Markowitch, and J. Zhou. An intensive survey of non-repudiation protocols. *Computer Communications Journal*, 25(17):1606–1621, 2002.

[29] G. Lowe. A hierarchy of authentication specifications. In *Proceedings of the 10th IEEE Computer Security Foundations Workshop (CSFW'97)*, pages 31–43. IEEE Computer Society Press, 1997.

[30] S. Schneider. Formal analysis of a non-repudiation protocol. In *PCSFW: Proceedings of The 11th Computer Security Foundations Workshop*, pages 54–65. IEEE Computer Society Press, 1998.

[31] P. F. Syverson and S. G. Stubblebine. Group principals and the formalization of anonymity. In *World Congress on Formal Methods (1)*, pages 814–833, 1999.

[32] J. Zhou and D. Gollmann. Towards verification of non-repudiation protocols, 1998.

[33] G. Zorn. RFC 2759: Microsoft PPP CHAP Extensions, Version 2, Jan. 2000. Status: Informational.