

# LTl Model Checking for Security Protocols

Alessandro Armando  
AI-Lab, DIST – Università di Genova  
Viale Causa 13, 16145 Genova, Italy  
armando@dist.unige.it

Roberto Carbone  
AI-Lab, DIST – Università di Genova  
Viale Causa 13, 16145 Genova, Italy  
carbone@dist.unige.it

Luca Compagna  
SAP Research  
805 Av. du Dr M. Donat, 06250 Mougins, France  
luca.compagna@sap.com

## Abstract

*Most model checking techniques for security protocols make a number of simplifying assumptions on the protocol and/or on its execution environment that prevent their applicability in some important cases. For instance, most techniques assume that communication between honest principals is controlled by a Dolev-Yao intruder, i.e. a malicious agent capable to overhear, divert, and fake messages. Yet we might be interested in establishing the security of a protocol that relies on a less unsecure channel (e.g. a confidential channel provided by some other protocol sitting lower in the protocol stack). In this paper we propose a general model for security protocols based on the set-rewriting formalism that, coupled with the use of LTL, allows for the specification of assumptions on principals and communication channels as well as complex security properties that are normally not handled by state-of-the-art protocol analysers. By using our approach we have been able to formalise all the assumptions required by the ASW protocol for optimistic fair exchange as well as some of its security properties. Besides the previously reported attacks on the protocol, we report a new attack on a patched version of the protocol.*

## 1. Introduction

During the last decade we have witnessed to the development of a new generation of model checking techniques specifically tailored for security protocols [17, 5, 15, 18]. As a result of this endeavour, a wide range of industrial strength security protocols can now be analysed automatically by state-of-the-art tools [2]. This is undoubtedly a remarkable result, but it must be noted that most techniques rely on (and thus are tailored to work with) a number of

simplifying assumptions:

- (A1) An adversary cannot learn anything from an encrypted message unless he knows the corresponding key. This is called the *perfect cryptography assumption*.
- (A2) Communication between honest principals is controlled by a Dolev-Yao (DY) intruder [13], a malicious agent capable to overhear, divert, and fake messages. When this assumption applies, we say that communication takes place over a *DY channel*.
- (A3) Honest principals are simple processes which are only required to react to messages of a specified form by sending other messages.
- (A4) Security properties are expressed as state properties (i.e. invariants).

While these assumptions are acceptable in many cases, there are a number of situations in which they prevent or greatly complicate the direct application of the analysis techniques based on them.

The importance of weakening assumption (A1) has already been recognised by many authors and extensions to the existing model checking techniques taking into account properties of cryptographic primitives have been put forward (see, e.g., [9, 11, 6]).

Assumptions (A2), (A3), and (A4) have received less attention. Yet, their importance cannot be overestimated as it is not uncommon to encounter verification problems for which they do not hold. For instance, DY channels are not appropriate to model the behaviour of an attacker against over-the-air protocols because message interception is unfeasible over broadcast communication media. But the best example is probably the optimistic fair exchange protocol

proposed by Asokan, Shoup, and Waidner (ASW) [4]. The protocol assumes for its proper functioning that communication between principals is carried over confidential and/or resilient channels. Thus assumption (A2) is obviously violated. Moreover, all protocol participants are assumed to make progress during execution of the protocol. This means that also assumption (A3) is violated. Finally, the protocol is expected to enjoy a security property (namely, fair exchange) that cannot be directly expressed as a reachability property. This means that assumption (A4) is violated as well.

In previous work we proposed a bounded model checking technique for security protocols that reduces the problem of determining whether a security protocol violates a security property in  $k > 0$  steps to the problem of checking the satisfiability of a propositional formula (the SAT problem). We have implemented our technique in a tool, called SATMC, that by leveraging on state-of-the-art SAT solvers can compete and in some cases outperform other state-of-the-art protocol analysers [3]. We have recently extended SATMC to support model checking of LTL formulae as described in [7].

In this paper we propose a general model for security protocols based on the set-rewriting formalism that, coupled with the use of LTL, allow us to relieve assumptions (A2), (A3), and (A4). We will consider model checking problems of the form:

$$M \models (C_I \wedge C_H) \supset G \quad (1)$$

where  $M$  is a labelled transition system modelling the behaviours of the honest agents and of the intruder,  $C_I$  and  $C_H$  are LTL formulae (henceforth called *LTL constraints*) that specify the allowed behaviours of the intruder and of the honest principals respectively, and  $G$  is an LTL formula stating the security property that the protocol is expected to enjoy.

In order to assess the effectiveness of our approach we have carried out a thorough analysis of the ASW protocol. By using our approach we have been able to formalise all the assumptions required by the protocol as well as its security properties. We have then analysed the protocol by using SATMC. Besides the previously reported attacks on the protocol, SATMC finds an unknown attack on the modified version of the ASW protocol proposed in [20]. We believe that the ability to detect this attack, which has eluded previous formal analyses of the protocol [20, 14], is a clear indication of the effectiveness of our approach.

**Structure of the paper.** In the next section we give a brief introduction of the ASW protocol. In Section 3 we describe our set-rewriting specification formalism and show how to use LTL to specify different types of channels, assumptions on the honest principals, and complex security properties. In Section 4 we discuss the results of our analysis of the

ASW with SATMC. In Section 5 we discuss some of the related work.

## 2. The ASW Protocol

The ASW protocol consists of three subprotocols *exchange*, *abort* and *resolve* shown in Figure 1.

Three roles take part in the subprotocols, namely the *originator* ( $O$ ), the *responder* ( $R$ ) and a *trusted third party* ( $T$ ), corresponding to the communicating processes of Figure 2.

The exchange subprotocol is played by  $O$  and  $R$  and—if successful—it allows the participating agents to achieve a mutual, non repudiable commitment on a previously agreed contractual text  $Text$  without the involvement of the TTP. If during execution of the exchange subprotocols either  $O$  or  $R$  do not receive the expected replies by the corresponding party within an acceptable time frame, then they initiate the abort or the resolve subprotocols with the TTP to force the resolution of the contract. The TTP keeps a permanent database of the contracts  $DB$  that he has already arbitrated.

In the following we use  $A$  to denote a principal that can play either the role of the originator or of the responder and write  $Sig_A(e_1, \dots, e_n)$  as an abbreviation of  $\{\langle e_1, \dots, e_n \rangle\}_{Ka^{-1}}$ , where  $Ka^{-1}$  is the private key of  $A$ .

**The exchange subprotocol.** The exchange subprotocol starts with  $O$  sending  $R$  a signed message  $me_1 = Sig_O(v(O), v(R), T, Text, h(N_O))$  containing its own public key  $v(O)$ , the public key of the responder  $v(R)$ , the identity of the TTP  $T$ , the contractual text  $Text$ , and the hash of a nonce  $h(N_O)$ . Notice that  $me_1$  does not constitute in itself a commitment by  $O$  on  $Text$  until  $N_O$  (i.e.  $O$ 's secret commitment) is revealed. Upon receipt of  $me_1$ ,  $R$  replies with the signed message  $me_2$  which, besides  $me_1$ , contains the hash of a new nonce ( $R$ 's secret commitment). When  $O$  receives  $me_2$  he replies by revealing his secret commitment  $N_O$  and the subprotocol concludes with  $R$  sending his own secret commitment to  $O$ . We say that *a principal has a standard contract* if and only if he knows the messages  $me_1$ ,  $me_2$ ,  $N_O$ , and  $N_R$ .

However things can go wrong either because of a network failure between  $O$  and  $R$  or because a malicious agent may divert messages. The abort and resolve subprotocols are meant to protect the protocol in these situations.

**The abort subprotocol.** If  $O$  does not receive  $me_2$  from  $R$  after a reasonable amount of time, then he may start the abort subprotocol. This amounts to requesting  $T$  to abort the previously initiated exchange by sending  $T$  the message  $ma_1 = Sig_O(aborted, me_1)$ . Upon receipt of an abort request from  $O$ ,  $T$  looks up its permanent database  $DB$

The *exchange* subprotocol:

- E1.  $O \rightarrow R$  :  $me_1 = Sig_O(v(O), v(R), T, Text, h(N_O))$
- E2.  $R \rightarrow O$  :  $me_2 = Sig_R(me_1, h(N_R))$
- E3.  $O \rightarrow R$  :  $N_O$
- E4.  $R \rightarrow O$  :  $N_R$

The *abort* subprotocol:

- A1.  $O \rightarrow T$  :  $ma_1 = Sig_O(aborted, me_1)$
- A2.  $T \rightarrow O$  : **if**  $resolved(me_1, me_2) \in DB$  **then**  $Sig_T(me_1, me_2)$   
**else**  $Sig_T(aborted, ma_1)$ ;  $DB := DB \cup \{aborted(me_1)\}$

The *resolve* subprotocol:

- R1.  $A \rightarrow T$  :  $mr_1 = \langle me_1, me_2 \rangle$
- R2.  $T \rightarrow A$  : **if**  $aborted(me_1) \in DB$  **then**  $Sig_T(aborted, ma_1)$   
**else**  $Sig_T(me_1, me_2)$ ;  $DB := DB \cup \{resolved(me_1, me_2)\}$

**Figure 1. The ASW protocol**

and determines whether this instance of the protocol has already been resolved. If this is the case, then  $T$  sends back a *replacement contract*, i.e. a message of the form  $Sig_T(me_1, me_2)$ . Otherwise,  $T$  aborts this instance of the protocol by sending an *abort token*, i.e. a message of the form  $Sig_T(aborted, ma_1)$  and updates  $DB$  accordingly.

**The resolve subprotocol.** The resolve subprotocol can be initiated both by  $O$  and  $R$  when they are waiting for the other party's secret commitment but suspect something has gone wrong. The initiating party (say  $A$ ) starts the subprotocol by sending  $T$  a message  $mr_1 = \langle me_1, me_2 \rangle$  thereby requesting to resolve this instance of the protocol. Upon receipt of this message,  $T$  looks up  $DB$  to determine whether this instance of the protocol has been already aborted. If this the case, then  $T$  replies with an abort token, otherwise it returns a replacement contract.

We say that *principal a has a valid contract* if and only if  $a$  has a standard contract or a replacement contract.

The proper functioning of the ASW protocol relies on a number of assumptions on the environment as well as on the behaviour of the honest participants.

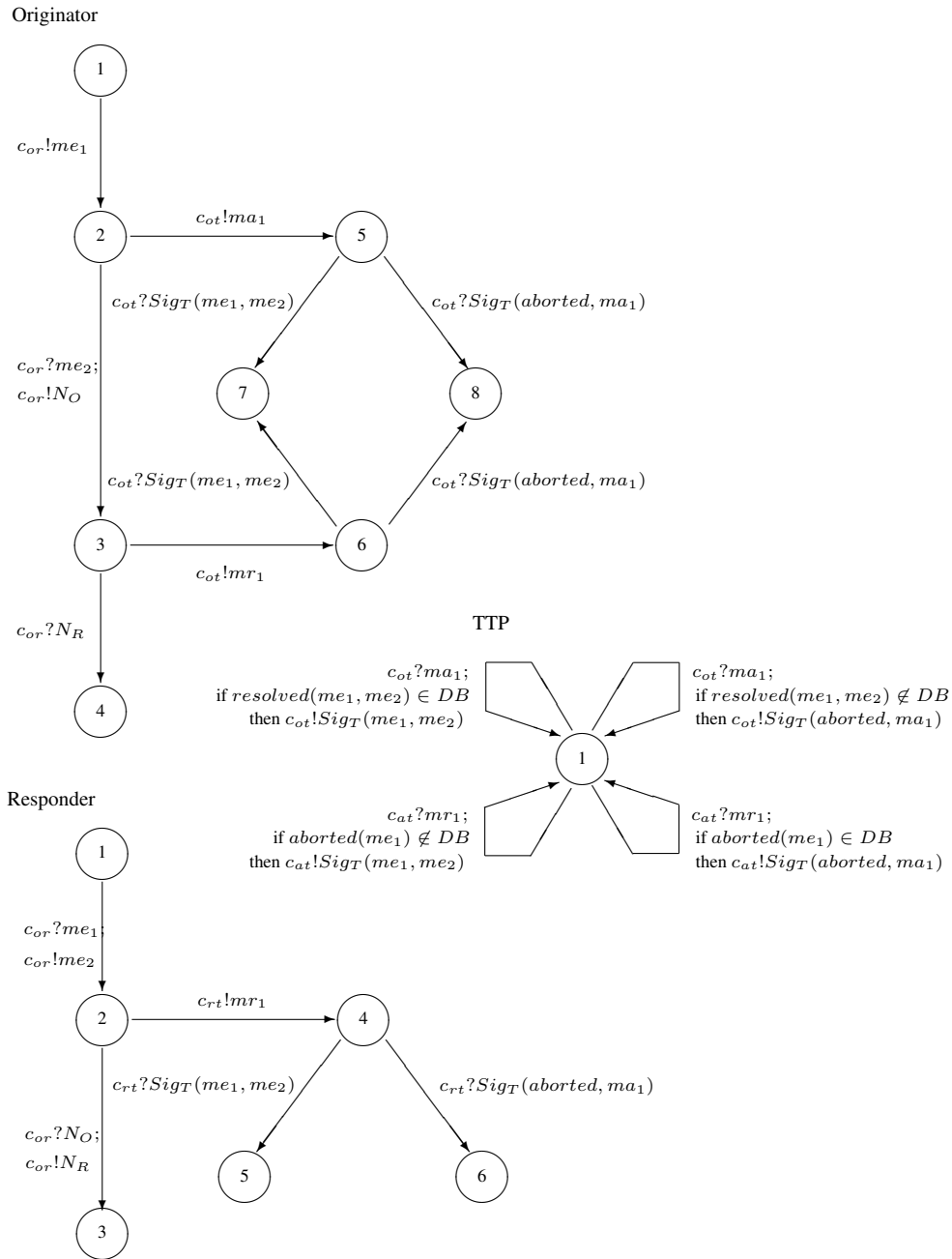
**Assumptions on the channels.** The communication channels between any two protocol participants are assumed to be confidential, i.e. eavesdroppers do not have (or have limited) access to the information travelling through these channels. Moreover it is also assumed that the channels between each participant and the TTP are resilient, i.e. any message deposited into these channels will be eventually delivered to its intended recipient.

**Assumptions on the Honest Principals.** It is assumed that during execution of the exchange sub-protocol, both the originator and the recipient will not indefinitely wait for a reply from the corresponding party and will eventually start either the abort (only the originator) or resolve subprotocol. With reference to Figure 2, when the originator is in state 2 or 3 he will eventually time out and choose the alternative transition leading to state 5 and 6 respectively. Similarly, when the responder is in state 2 he will eventually time out and choose the alternative transition leading to state 4.

A further assumption dictates that TTP must be always available, i.e. he must eventually process all the messages received by replying to requests by the originator and the responder.

**Security Properties.** The ASW protocol is expected to meet a number of security objectives. Here we focus on fair exchange. Fair exchange can be expressed as the conjunction of the following two properties (slightly adapted from [20]):

- (A) It is impossible for a corrupt principal to obtain a valid contract without allowing the remaining principal to also obtain a valid contract.
- (B) Once an honest principal obtains an abort token, it is impossible for any other principal to obtain a valid contract.



Legenda:

- $c_{or}$ ,  $c_{ot}$ , and  $c_{rt}$  channels used to support the communication between  $O$  and  $R$ ,  $O$  and  $T$ , and  $R$  and  $T$ .
- $c!m$  means that message  $m$  is sent over channel  $c$ .
- $c?x$  means that a message, say  $m$ , is read from  $c$  and variable  $x$  is set to  $m$ .

Figure 2. Process view of the ASW Protocol

### 3. Modelling Security Protocols using Set-rewriting and LTL

We use set-rewriting (in the line of [8]) to specify the transition system associated with the concurrent execution of a number of sessions of the protocol and LTL to constrain the allowed behaviours of the channels and of the honest principals as well as to define the security property that the protocol is expected to enjoy. Both the set-rewriting formalism and logic are defined in terms of a first-order language with sorts.

Let  $V$  be the set of variables occurring in the specification of the protocol. A *session of the protocol* is a triple  $\langle i, \pi, \sigma \rangle$ , where  $i \in \mathbb{N}$  is the *session identifier*,  $\pi$  is a function that associates each role in the protocol with individual constants of sort AGENT, and  $\sigma$  is a substitution mapping each variable in  $V$  of sort different from NONCE into an expression of the same sort. A *protocol scenario* is a finite set of protocol sessions.

With reference to our specification of the ASW protocol given in Figure 1,  $V = \{O, R, T, Txt, N_O, N_R\}$ , where  $O, R$ , and  $T$  are of sort AGENT,  $Txt$  is of sort TEXT, and  $N_R$  and  $N_O$  are of sort NONCE. A possible session of the ASW protocol is given by  $ses_1 = \langle 1, \pi_1, \sigma_1 \rangle$ , where  $\pi_1(\text{Originator}) = o$ ,  $\pi_1(\text{Responder}) = r$ , and  $\pi_1(\text{TTP}) = t$  and  $\sigma_1 = \{o/O, r/O, t/T, txt/Txt\}$ . Another possible session of the ASW protocol is given by  $ses_2 = \langle 2, \pi_2, \sigma_2 \rangle$ , where  $\pi_2(\text{Originator}) = o$ ,  $\pi_2(\text{Responder}) = i$ , and  $\pi_2(\text{TTP}) = t$  and  $\sigma_2 = \{o/O, i/R, t/T, txt/Txt\}$ . A possible scenario for the ASW protocol is  $\mathcal{S} = \{ses_1, ses_2\}$ .

Thus we focus on the problem of determining whether the current execution of the sessions of the protocol included in a given protocol scenario enjoy the expected security properties. In this section we show how this problem can be recast into a model checking problem of the form (1). In Section 3.1 we show how (a superset of) the behaviours of the honest principals involved in the protocol scenario and of the intruder can be specified using a set-rewriting formalism. This amounts to specifying the model  $M$  of (1). In Section 3.2 and in Section 3.3 we show how assumptions on the behaviour of the intruder and of the honest principals respectively can be specified by means of LTL constraints corresponding to the  $C_H$  and  $C_I$  constraints of (1) respectively. Finally in Section 3.4 we show how the security property that the protocol is expected to enjoy can be specified by means of LTL formulae corresponding to the LTL formula  $G$  in (1).

#### 3.1 Specifying the Protocol Scenario and the Intruder

The model  $M$  consists of a labelled transition system  $T$  modelling the behaviours of the honest agents and of the

intruder, and of the initial state  $I$  of the honest agents and of the intruder.

States are represented as sets of *facts* and transitions as *set-rewriting rules* that define mappings between sets of facts. Facts are expressions of the form given in the left column of Table 1; their informal meaning is explained in the right column. If  $S$  is a set of facts, then we interpret the facts in  $S$  as the propositions holding in the state represented by  $S$ , all other facts being false in that state (closed-world assumption). If  $S$  is a set of facts representing a state, then the state of honest principal  $a$  is represented by the facts of the form  $\text{state}_{Role}(j, a, es, s)$  (called *state-facts*) and  $\text{ak}(a, m)$  occurring in  $S$ . (We assume that for each session  $s$  and for each principal  $a$  there exists at most one fact of the form  $\text{state}_{Role}(j, a, es, s)$  in  $S$ .) No state-fact is included in the state of the intruder.

Besides the facts representing the initial knowledge of the intruder and of the honest agents, the initial state  $I$  contains a state-fact  $\text{state}_{Role}(1, a, es, s)$  for each session  $\langle s, \pi, \sigma \rangle \in \mathcal{S}$  such that  $\pi(\text{Role}) = a$  and  $a$  is not the intruder. For the sake of simplicity we do not discuss here how  $es$  can be automatically computed given the protocol and its scenario.

Rewriting rules specify the evolution of the system. The behaviour of honest participants is specified by rules of the form:

$$\text{sent}(\text{RS}, \text{B}, \text{A}, \text{M}, \text{C}) \xrightarrow{\text{receive}(\text{A}, \text{B}, \text{RS}, \text{M}, \text{C})} \text{rcvd}(\text{A}, \text{B}, \text{M}, \text{C}) \cdot \text{ak}(\text{A}, \text{M}) \quad (2)$$

$$\text{rcvd}(\text{A}, \text{B}, \text{M}, \text{C}) \cdot \text{state}_{Role}(j, \text{A}, es, \text{S}) \xrightarrow{\text{send}_j(\text{A}, \text{B}, \text{B1}, \dots, \text{S})} \text{sent}(\text{A}, \text{A}, \text{B1}, \text{M1}, \text{C1}) \cdot \text{state}_{Role}(l, \text{A}, es', \text{S}) \quad (3)$$

Rule (2) models the reception of a message by an honest agent, whereas rule (3) models the processing of a previously received message. More in detail, rule (3) states that if principal A is at step  $j$  in session S of the protocol and she has received message M on channel C (supposedly) by B, then she can send message M1 to B1 on channel C1 and change her internal state accordingly. Notice that rule (3) may take slightly different forms depending on the type of protocol step to be modelled. For instance, if  $j = 1$  and A plays the role of initiator (i.e. the principal that sends the first message of the protocol), the fact  $\text{rcvd}(\text{A}, \text{B}, \text{M}, \text{C})$  is not included in the left hand side of the rule. Similarly, if a fresh term must be generated and sent, then  $c(\text{N})$  and  $c(s(\text{N}))$  need to be added to the left and right hand sides of the rule respectively. A further variant is necessary when the step involves either a membership test or an update of a set of elements. In this case the fact *contains* needs to be taken properly into account. In general the above rules can be automatically generated from the process view of the

**Table 1. Facts and their informal meaning**

Fact	Meaning
$\text{state}_{\text{Role}}(j, a, es, s)$	Principal $a$ , playing role $\text{Role}$ , is ready to execute step $j$ in session $s$ of the protocol, and $es$ is a list of expressions representing the internal state of $a$ and thus affecting her future behaviour.
$\text{ak}(a, m)$	Principal $a$ knows message $m$ .
$\text{sent}(rs, b, a, m, c)$	Principal $rs$ has sent message $m$ on channel $c$ to principal $a$ pretending to be principal $b$ .
$\text{rcvd}(a, b, m, c)$	Message $m$ (supposedly sent by principal $b$ ) has been received on channel $c$ by principal $a$ , but $a$ has not processed it yet.
$c(n)$	Term $n$ is the current value of the counter used to construct fresh terms. This value is incremented every time a fresh term is built.
$\text{contains}(db, m)$	Message $m$ is contained into set $db$ . Sets are used to share data between principals.

protocol. For instance, the transition of the responder from node 1 to node 2 in Figure 2 is modelled by the rule (2) and by the following rewrite rule:

$$\begin{aligned} & c(N) \cdot \text{rcvd}(R, O, me_1(O, R, \text{Txt}, N_0, T), C_{OR}) \cdot \\ & \text{state}_{\text{resp}}(1, R, [O, T, \text{Txt}, C_{RT}, C_{OR}], S) \\ & \xrightarrow{\text{send}(R, \dots, S)} \\ & c(s(N)) \cdot \text{sent}(R, R, O, me_2(O, R, \text{Txt}, N_0, N, T), C_{OR}) \cdot \\ & \text{ak}(R, N) \cdot \text{state}_{\text{resp}}(2, R, [O, T, \text{Txt}, C_{RT}, C_{OR}, h(N_0), N], S) \end{aligned}$$

The abilities of the intruder are modelled by:

$$\begin{aligned} & \text{ik}(M) \cdot \text{ik}(A) \cdot \text{ik}(B) \xrightarrow{\text{fake}(A, B, M, C)} \\ & \text{sent}(i, A, B, M, C) \cdot \text{ik}(M) \cdot \text{ik}(A) \cdot \text{ik}(B) \\ & \text{sent}(A, A, B, M, C) \xrightarrow{\text{intercept}(A, B, M, C)} \text{rcvd}(i, A, M, C) \cdot \text{ik}(M) \\ & \text{sent}(A, A, B, M, C) \xrightarrow{\text{overhear}(A, B, M, C)} \\ & \text{sent}(A, A, B, M, C) \cdot \text{rcvd}(i, A, M, C) \cdot \text{ik}(M) \end{aligned}$$

Finally, the inferential capabilities of the agents are modelled by the following rules (where  $\bar{K}$  is  $K$  if  $K = K^{-1}$  and  $\bar{K}$  is  $K^{-1}$  otherwise):

$$\begin{aligned} & \text{ak}(A, M) \cdot \text{ak}(A, K) \xrightarrow{\text{encrypt}(A, K, M)} \\ & \text{ak}(A, M) \cdot \text{ak}(A, K) \cdot \text{ak}(A, \{M\}_K) \\ & \text{ak}(A, \{M\}_K) \cdot \text{ak}(A, \bar{K}) \xrightarrow{\text{decrypt}(A, K, M)} \\ & \text{ak}(A, \{M\}_K) \cdot \text{ak}(A, \bar{K}) \cdot \text{ak}(A, M) \\ & \text{ak}(A, M_1) \cdot \text{ak}(A, M_2) \xrightarrow{\text{pairing}(A, M_1, M_2)} \\ & \text{ak}(A, M_1) \cdot \text{ak}(A, M_2) \cdot \text{ak}(A, \langle M_1, M_2 \rangle) \end{aligned}$$

$$\text{ak}(A, \langle M_1, M_2 \rangle) \xrightarrow{\text{decompose}(A, M_1, M_2)} \text{ak}(A, \langle M_1, M_2 \rangle) \cdot \text{ak}(A, M_1) \cdot \text{ak}(A, M_2)$$

If  $T$  is a sort, then *the domain of  $T$*  is a set of ground (i.e. variable-free) terms of sort  $T$ . Since our model checking technique carries out a bounded analysis of the protocols, we assume that the domains of all sorts are finite. (The computation of finite over-approximation of the domains can be done in polynomial time by carrying out a static analysis of the protocol [3].) If  $x$  is a variable of sort  $T$ , then with  $\text{dom}(x)$  we denote the domain of  $T$  and use  $\forall x.w$  and  $\exists x.w$  as an abbreviation of the formulae  $\bigwedge_{t \in \text{dom}(x)} w[t/x]$  and  $\bigvee_{t \in \text{dom}(x)} w[t/x]$  respectively, where  $w[t/x]$  is the formula obtained by substituting all free occurrences of  $x$  in  $w$  with the term  $t$ .

Preliminary to the definition of the semantics of LTL we provide an (excerpt of) the operational semantics of our set-rewriting formalism. (The full definition is available in [3].) Let  $(L \xrightarrow{\rho} R)$  be an instance of a rewrite rule and  $S$  be a set of facts, if  $L \subseteq S$ , then we say that  $\rho$  is applicable in  $S$  and that  $S' = \text{app}_{\rho}(S) = (S \setminus L) \cup R$  is the state resulting from the execution of  $\rho$  in  $S$ . A *path*  $\pi$  is an alternating sequence of states and rules  $S_0 \rho_1 S_1 \dots S_{n-1} \rho_n S_n$  such that  $S_i = \text{app}_{\rho_i}(S_{i-1})$  (i.e.  $S_i$  is a state resulting from the execution of  $\rho_i$  in  $S_{i-1}$ ), for  $i = 1, \dots, n$ . If, additionally,  $S_0 \subseteq \mathcal{I}$ , then we say that the path is *initialised*. Let  $\pi = S_0 \rho_1 S_1 \dots S_{n-1} \rho_n S_n$  be a path, we define  $\pi(i) = S_i$  and  $\pi_i = S_i \rho_{i+1} S_{i+1} \dots S_{n-1} \rho_n S_n$ .  $\pi(i)$  and  $\pi_i$  are the  $i$ -th state of the path and the suffix of the path starting with the  $i$ -th state respectively. We also assume that paths have infinite length. This can be always obtained by adding stuttering transitions to the transition system.

The language of LTL we consider uses facts as atomic propositions, the usual propositional connectives (namely,  $\neg, \vee, \wedge, \supset$ ), and the temporal operators **F** (eventually) and **G** (globally). Additional temporal operators (including past operators) can be easily added to the definition. If  $\pi$  is an

initialised path of  $M$ , the validity of an LTL formula  $f$  along  $\pi$ , denoted as  $\pi \models f$ , is inductively defined in Figure 3.

$\pi \models f$	iff	$f \in \pi(0)$
$\pi \models \neg f$	iff	$\pi \not\models f$
$\pi \models (f \vee g)$	iff	$\pi \models f$ or $\pi \models g$
$\pi \models (f \wedge g)$	iff	$\pi \models f$ and $\pi \models g$
$\pi \models (f \supset g)$	iff	$\pi \not\models f$ or $\pi \models g$
$\pi \models \mathbf{G}f$	iff	$\forall i. \pi_i \models f$
$\pi \models \mathbf{F}f$	iff	$\exists i. \pi_i \models f$

**Figure 3. Semantics of LTL**

### 3.2. Constraining the Behaviour of the Intruder

In this section we show how confidential and resilient channels can be obtained by introducing suitably defined LTL constraints to  $C_I$ .

**Confidential channels.** As stated in [16]: “A channel provides confidentiality if its *output* is exclusively accessible to a specified receiver [...]” In our model this amounts to requiring that in every state  $S$  if a fact  $\text{rcvd}(b, a, m, c) \in S$ , then  $b$  is one of the principals that have exclusive access to the channel. Thus, the condition that *channel  $c$  is confidential to a set of principals  $PS$*  can be formalised by the following formula:

$$\text{confidential}(c, PS) := \mathbf{G} \forall (\text{rcvd}(B, A, M, c) \supset \bigvee_{p \in PS} B = p)$$

where here and in the sequel we use  $\forall w$  as an abbreviation of the formula  $\forall x_1. \dots \forall x_n. w$ , where  $x_1, \dots, x_n$  are the free variables of  $w$ .

**Resilient channels.** As stated in [4]: “A communication channel is *resilient* if it is normally operational but an attacker can succeed in delaying messages by an arbitrary, but finite amount of time. In other words, a message inserted into a resilient channel will eventually be delivered.” In our model this amounts to requiring that every message sent over the channel will be eventually delivered to the intended recipient. Thus the condition that *channel  $c$  is resilient* can be formalised as follows:

$$\text{resilient}(c) := \mathbf{G} \forall (\text{sent}(A', A, B, M, c) \supset \mathbf{F} \text{rcvd}(B, A, M, c))$$

### 3.3. Constraining the Behaviour of Honest Principals

In order to ensure that the originator and the responder will not indefinitely wait for an answer from the other party it suffices to assume that whenever they are in a state from which a time-out can occur, eventually they will not be in that state any more. This can be formalised by the following LTL constraints:

$$\mathbf{G} \forall (\text{state}_{\text{Role}}(j, \dots) \supset \mathbf{F} \neg \text{state}_{\text{Role}}(j, \dots))$$

for  $j \in \{2, 3\}$  if  $\text{Role} = \text{orig}$  and  $j = 2$  if  $\text{Role} = \text{resp}$ . In the sequel we will use *progress\_or* to denote the conjunction of the above constraints.

The availability of the TTP can be imposed by requiring that all the messages received will be eventually processed by the TTP. This is formalised by the following LTL constraint:

$$\text{progress\_ttp} := \mathbf{G} \forall (\text{rcvd}(t, P, M, C) \supset \mathbf{F} \neg \text{rcvd}(t, P, M, C))$$

### 3.4. Specifying Security Properties

In the sequel, free variables (that—we recall—are denoted by capital letters) occurring in formulae are to be considered as universally quantified.

**Goal A.** The goal amounts to requiring that if some protocol participant obtains a valid contract binding the other participant to some contractual text  $\text{txt}$  using some secret commitments ( $N_O, N_R$ ), then eventually the other participant will also obtain a valid contract relative to the same contractual text and secret commitments. We recall that a principal has a valid contract if and only if he has a standard contract (i.e. he knows  $me_1, me_2, N_O$ , and  $N_R$ ) or a replacement contract (i.e. he knows  $\text{Sig}_T(me_1, me_2)$ ). More precisely, the fact that principal  $P$  has a valid contract binding  $O$  and  $R$  (as originator and responder respectively) to contractual text  $\text{Txt}$  using  $N_O$  and  $N_R$  as secret commitments and  $T$  as TTP is formalised by the formula

$$\begin{aligned} & [\text{ak}(P, me_1(O, R, \text{Txt}, N_O, T)) \wedge \\ & \quad \text{ak}(P, me_2(O, R, \text{Txt}, N_O, N_R, T)) \wedge \\ & \quad \text{ak}(P, N_O) \wedge \text{ak}(P, N_R)] \vee \\ & \quad \text{ak}(P, \text{Sig}_T(me_1(O, R, \text{Txt}, N_O, T), \\ & \quad \quad me_2(O, R, \text{Txt}, N_O, N_R, T))) \end{aligned}$$

which we abbreviate with  $\text{has\_vc}(P, O, R, \text{Txt}, N_O, N_R, T)$ .

Goal  $A$  can then be expressed by the formula, say  $G_A$ , obtained by taking the conjunction of all the formulae of the form

$$\mathbf{G} \forall (has\_vc(o, o, r, txt, N_O, N_R, t) \supset \mathbf{F} has\_vc(r, o, r, txt, N_O, N_R, t)) \quad (4)$$

$$\mathbf{G} \forall (has\_vc(r, o, r, txt, N_O, N_R, t) \supset \mathbf{F} has\_vc(o, o, r, txt, N_O, N_R, t)) \quad (5)$$

for all  $\langle s, \pi, \{o/O, r/R, t/T, txt/Txt\} \rangle \in \mathcal{S}$ . Formula (4) states that if the originator  $o$  has a valid contract binding  $r$  to  $txt$  using  $N_O$  and  $N_R$  as secret commitments and  $t$  as TTP then eventually  $r$  will have a corresponding valid contract. Formula (5) is dual.

**Goal  $B$ .** The goal amounts to requiring that if some protocol participant obtains an abort token for a contract binding the other participant to some contractual text  $txt$  using a secret commitment  $N_O$ , then the other participant will never obtain a corresponding valid contract. Here “to obtain an abort token” does not only mean the act of receiving the abort token, but also that of processing it. By looking at Figure 2 it is easy to see that the originator (responder) has obtained an abort token if and only if he is in state 8 (state 6, resp.). Goal  $B$  can then be expressed by the formula, say  $G_B$ , obtained by taking the conjunction of all the formulae of the form

$$\mathbf{G} \forall (has\_at(a, o, r, txt, N_O, N_R, t, s_a) \supset \mathbf{G} \neg has\_vc(b, o, r, txt, N_O, N'_R, t))$$

for all  $\langle s_a, \pi, \{o/O, r/R, t/T, txt/Txt\} \rangle \in \mathcal{S}$  and  $\langle a, b \rangle \in \{\langle o, r \rangle, \langle r, o \rangle\}$ , where  $has\_at(a, o, r, txt, N_O, N_R, t, s_a)$  is  $state_{orig}(8, o, [r, txt, N_O, N_R, t], s_a)$  if  $a = o$  and  $state_{resp}(6, r, [o, txt, N_O, N_R, t], s_a)$  if  $a = r$ .

## 4. Analysis of the ASW Protocol

We have analysed the ASW protocol by defining and feeding SATMC with model checking problems of the form (1), where

- $M$  is a state transition system modelling a protocol scenario; in our analysis we considered a variety of protocol scenarios. The attacks described in the sequel are obtained by considering the following three protocol scenarios:

*Scenario 1:* one single session in which the intruder plays the responder,

*Scenario 2:* two sessions in which the intruder plays the originator, and

*Scenario 3:* two sessions in which the intruder does not play any role;

- $C_I$  is the conjunction of the constraints imposing the resilience of the channels  $c_{ot}$  and  $c_{rt}$  (namely  $resilient(c_{ot})$  and  $resilient(c_{rt})$ ) and possibly other constraints imposing the confidentiality of all the channels (see below);
- $C_H$  is the conjunction of the constraints  $progress\_or$  and  $progress\_ttp$ ; and
- $G$  is either  $G_A$  or  $G_B$  as defined in Section 3.4.

Besides the definition of confidential channel given in Section 3.2, it is possible to give a weaker definition that was proposed and used in [20] for their analysis of the ASW protocol. This alternative definition assumes that the intruder cannot learn anything from the messages in the channel, but he can store and replay the messages later. Notice that these abilities are prevented to intruder if the definition of Section 3.2 is assumed as he cannot even receive the message in transit on the channel. We call *weak confidentiality* the weak form of confidentiality and call *strong confidentiality* the one defined in Section 3.2.

It must be noted that both forms of confidentiality are relevant in practice. A weakly confidential channel can be obtained by encrypting all the messages sent over the channel with a key only known to (or shared with) the receiver. In this way the intruder (or any other agent) can easily identify, store and replay the encrypted messages but he cannot access to their content. A strongly confidential channel can be obtained by establishing a communication link implementing a stream cipher preliminary to the execution of the protocol. For instance, this can be obtained by running the ASW protocol on top of TLS connections. In this way the intruder cannot extract the individual messages from the observed traffic.

In the light of the above considerations, we considered both definitions of confidentiality in our analysis.

### 4.1. Protocol Analysis with Strongly Confidential Channels

We first analysed the protocol by assuming that all the channels are confidential according to the definition given in Section 3.2. This means that, besides the constraints stating the resilience of  $c_{ot}$  and  $c_{rt}$ ,  $C_I$  contains the conjunction of all the formulae of the form  $confidential(c_{or}, \{o, r\})$ ,  $confidential(c_{ot}, \{o, t\})$  and  $confidential(c_{rt}, \{r, t\})$ .

The analysis of the protocol w.r.t. Goal  $A$ , i.e. when  $G = G_A$ , revealed the attack of Figure 4. Here the intruder, playing the responder, executes the exchange subprotocol with  $O$  and in the meantime he also computes a different response  $me'_2$  using a different nonce  $N'_R$ . Thus at the end

- E1.  $O \rightarrow I$  :  $me_1$   
E2.  $I \rightarrow O$  :  $me_2$   
    $I$  computes new random  $N'_I$  and then  $me'_2$   
E3.  $O \rightarrow I$  :  $N_O$   
E4.  $I \rightarrow O$  :  $N_I$

**Figure 4. Attack on the ASW protocol violating Goal A**

- E1.  $O \rightarrow I$  :  $me_1$   
E2.  $I \rightarrow O$  :  $me_2$   
    $I$  computes new random  $N'_I$  and then  $me'_2$   
E3'.  $O \rightarrow I$  :  $Sig_O(N_O, h(N_I))$   
E4'.  $I \rightarrow O$  :  $Sig_I(N_I, h(N_O))$   
R1.  $I \rightarrow T$  :  $mr_1 = \langle me_1, me'_2 \rangle$   
R2.  $T \rightarrow I$  :  $mr_2 = Sig_T(me_1, me'_2)$

**Figure 5. Attack on the patched version of the ASW protocol violating Goal A**

of the protocol, the intruder possesses a standard contract consisting of the messages  $me_1$ ,  $me'_2$ ,  $N_O$ ,  $N'_R$  while the originator possesses only the standard contract consisting of  $me_1$ ,  $me_2$ ,  $N_O$ ,  $N_R$ . This attack is similar in spirit to the second attack in Section 6.1 of [20], but it must be noted that the attack in Figure 4 is simpler as the TTP is not involved.

In [20] Shmatikov and Mitchell propose to repair the protocol by replacing steps E1 and E2 of Figure 1 with the following two:

- E3'.  $O \rightarrow R$  :  $Sig_O(N_O, h(N_R))$   
E4'.  $R \rightarrow O$  :  $Sig_R(N_R, h(N_O))$

SATMC confirms that the improved version of the protocol does not suffer from the attack of Figure 4, but it detects a new (i.e. previously unknown) attack which is shown in Figure 5. This attack shows that Goal A can be violated because the intruder, here playing the responder, obtains a replacement contract relative to the secret commitments  $N_O$  and  $N'_I$ , while the originator obtains only a valid contract relative to the secret commitments  $N_O$  and  $N_I$ .

Thus the patch to the exchange subprotocol proposed in [20] does not solve the problem with the ASW protocol. We believe this is due to the asymmetrical nature of the protocol and cannot be circumvented without carrying out more significant changes on the protocol. This is due to the fact that  $R$  has the ability to generate (infinitely many) variants of the contract by using newly generated nonces. As an alternative, we propose (as already done in [19]) to accept the

asymmetry in the protocol and we weaken the goal. The new formula  $G'_A$  can be obtained by replacing (5) in the definition of  $G_A$  with the following formula:

$$\mathbf{G} \forall (has\_vc(r, o, r, txt, N_O, N_R, t) \supset \\ \mathbf{F} \exists N_R. has\_vc(o, o, r, txt, N_O, N_R, t))$$

As before, if  $R$  has a valid contract we still ask  $O$  to possess a valid contract relative to the same contractual text  $txt$  and secret commitment  $N_O$ , but we no longer insist on requiring that the contract is relative to the same secret commitment  $N_R$ .

As expected, when checking the protocol w.r.t. the formula  $G'_A$  SATMC does not find any attack on the protocol.

We then turned our attention to Goal B, i.e. we set  $G = G_B$ . A spurious attack soon revealed that the formula erroneously considers as problematic the situation in which a principal obtains the abort token while the other principal has already a corresponding valid contract. We therefore weakened the goal in the following way:

- (B') If an honest principal obtains an abort token, then any other principal has already a corresponding valid contract or will not be able to obtain one in the future.

This property can be expressed by the formula  $G_{B'}$  obtained by conjoining all the formulae of the form:

$$\mathbf{G} \forall (has\_at(a, o, r, txt, N_O, N_R, t, s_a) \supset \\ (has\_vc(b, o, r, txt, N_O, N'_R, t) \vee \\ \mathbf{G} \neg has\_vc(b, o, r, txt, N_O, N'_R, t)))$$

for all  $\langle s_a, \pi, \{o/O, r/R, t/T, txt/Txt\} \rangle \in \mathcal{S}$  and  $\langle a, b \rangle \in \{\langle o, r \rangle, \langle r, o \rangle\}$ .

By checking the protocol w.r.t. the formula  $G_{B'}$  SATMC found the attack shown in Figure 6. Here the responder obtains an abort token relative to the secret commitment  $N_I$  and eventually the intruder, here playing the originator, obtains a standard contract relative to the same contractual text and secret commitment  $N_I$ . But also in this case the problem lies with the formulation of the security property and not with the protocol. In fact the responder eventually obtains the same standard contract obtained by the intruder.

A similar problem was already recognised in [14]. In the same paper it is also proposed to relax the objective to “If an honest agent has an abort token, then he also possesses a valid contract or nobody else can obtain one.”. However, this solution does not solve our problem because in our attack the originator obtains the valid contract *after* the reception of the abort token.

We propose to rectify the problem by reformulating goal in the following way:

- (B'') If an honest principal, say  $A$ , obtains an abort token, then any other principal has already a corresponding

E1.  $I \rightarrow R$  :  $me_1$   
 E2.  $R \rightarrow I$  :  $me_2$   
 A1.  $I \rightarrow T$  :  $ma_1 = Sig_I(aborted, me_1)$   
 A2.  $T \rightarrow I$  :  $ma_2 = Sig_T(aborted, ma_1)$   
 R1.  $R \rightarrow T$  :  $mr_1 = \langle me_1, me_2 \rangle$   
 R2.  $T \rightarrow R$  :  $mr_2 = Sig_T(aborted, ma_1)$   
 E1.  $I \rightarrow R$  :  $me_1$   
 E2.  $R \rightarrow I$  :  $me'_2$   
 E3.  $I \rightarrow R$  :  $N_I$   
 E4.  $R \rightarrow I$  :  $N'_R$

**Figure 6. Attack on the ASW protocol violating Goal  $B'$**

valid contract or will not be able to obtain one in the future, or  $A$  already possesses or eventually will obtain a corresponding valid contract.

This property can be expressed by the formula  $G_{B''}$  obtained by conjoining all the formulae of the form:

$$\begin{aligned}
 & \mathbf{G} \forall ((has\_at(a, o, r, txt, N_O, N_R, t, s_a) \supset \\
 & \quad (has\_vc(b, o, r, txt, N_O, N'_R, t) \vee \\
 & \quad \mathbf{G} \neg has\_vc(b, o, r, txt, N_O, N'_R, t) \vee \\
 & \quad \mathbf{F} has\_vc(a, o, r, txt, N_O, N'_R, t)))
 \end{aligned}$$

for all  $\langle s_a, \pi, \{o/O, r/R, t/T, txt/Txt\} \rangle \in \mathcal{S}$  and  $\langle a, b \rangle \in \{\langle o, r \rangle, \langle r, o \rangle\}$ .

SATMC does not find any attack while by checking the protocol w.r.t. Goal  $B''$ .

## 4.2. Protocol Analysis with Weakly Confidential Channels

We finally analysed the protocol by assuming that all channels are weakly confidential, that is the intruder cannot learn anything from the messages in transit on these channels, but he can store and replay the messages later. As pointed out in [20], since no signing keys are transmitted during the execution of the protocol, the intruder does not have the possibility to sign messages (unless that it is a participant of the protocol). Therefore in order to model this form of confidentiality we did not add any constraint for confidentiality to  $C_I$  which therefore simply contains the constraints stating the resilience of  $c_{ot}$  and  $c_{rt}$ .

As in the previous case, we started our analysis by considering the original version of the protocol. SATMC found a replay attack on the stronger version of Goal  $A$  (i.e. when  $G = G_A$ ) which is shown in Figure 7. Here the intruder overhears a session of the exchange protocol played by  $O$

E1.  $O \rightarrow R$  :  $me_1$   
 E2.  $R \rightarrow O$  :  $me_2$   
 E3.  $O \rightarrow R$  :  $N_O$   
 E4.  $R \rightarrow O$  :  $N_R$   
 E1.  $I(O) \rightarrow R$  :  $me_1$   
 E2.  $R \rightarrow O$  :  $me'_2$  //  $I$  intercepts  
 E3.  $I(O) \rightarrow R$  :  $N_O$   
 E4.  $R \rightarrow O$  :  $N'_R$  //  $I$  intercepts

**Figure 7. Replay attack on the ASW protocol violating Goal  $A$**

E1.  $O \rightarrow R$  :  $me_1$   
 E2.  $R \rightarrow O$  :  $me_2$   
 E3'.  $O \rightarrow R$  :  $Sig_O(N_O, h(N_R))$   
 E4'.  $R \rightarrow O$  :  $Sig_I(N_R, h(N_O))$   
 R1.  $I(R) \rightarrow T$  :  $mr_1 = \langle me_1, me'_2 \rangle$   
 R2.  $T \rightarrow R$  :  $mr_2 = Sig_T(me_1, me'_2)$

**Figure 8. Replay attack on the patched version of the ASW protocol violating Goal  $A$**

and  $R$  and then he initiates a new session with  $R$  pretending to be  $O$ . At the end of this second session of the protocol,  $R$  possesses a standard contract consisting of the messages  $me_1, me'_2, N_O, N'_R$  whereas  $O$  possesses only a standard contract consisting of  $me_1, me_2, N_O, N_R$ . This is the same attack as the first of the two attacks described in Section 6.1 of [20].

SATMC also found a replay attack on Goal  $A$  (shown in Figure 8), which is similar to that of Figure 5. This attack can be staged by the intruder as soon as a normal run of the protocol (played by two honest principals  $O$  and  $R$ ) is completed. The intruder simply asks the TTP for a replacement contract pretending to be  $R$  and the TTP replies by sending a replacement contract to  $R$ . Thus at the end  $R$  has two valid contracts: one consisting of  $me_1, me_2, Sig_O(N_O, h(N_R))$ , and  $Sig_I(N_R, h(N_O))$  and the other consisting of  $Sig_T(me_1, me'_2)$ .

As in the previous case with strong confidential channels, SATMC does not find any attack by checking the protocol w.r.t. the weaker version of Goal  $A$  (i.e. when  $G = G'_A$ ).

## 5. Related Work

Shmatikov and Mitchell [20] and, more recently, Drielsma and Mödersheim [14] have carried out a detailed analysis of the ASW protocol using two different model

checking techniques for security protocols. Since both techniques can only check invariants, these approaches require a transformation of the original model checking problem into a new one of the form  $M' \models \mathbf{G} P'$  (where  $P'$  is a formula without temporal operators). If, on the one hand, these approaches proved effective as they unveiled unknown attacks on the ASW protocol, on the other hand the complexity of the transformation—for which the proposed techniques provide no support—makes these approaches both error prone and time consuming. This is confirmed by the existence of the attack of Figure 5 on the “patched” version of the ASW protocol proposed in [20] that was not detected by using these techniques. The approach we have presented in this paper provides the protocol designer with much more flexibility as the assumptions on the channels, the assumptions on the behaviour of honest principals, and the security properties can be readily specified in LTL and directly fed to the model checker.

Kremer and Raskin [19] use the formal model of alternating transition systems to specify the ASW protocol and alternating-time temporal logic to state the property of abuse freeness. In their model of the protocol they do not include an independent playing the intruder, but create an honest and a malicious version for each principal involved in the protocol. Like in our approach they use formulae of the logic to state the assumptions on the communication channels, the assumptions on the behaviour of honest agents, and the secure properties. However their model is tailored to the analysis of abuse-freeness in contract-signing protocols, whereas our approach supports the treatment of a variety of protocols, including all the standard authentication protocols included in the Clark-Jacob library [10] and the industrial-scale security protocols of the AVISPA Library [2].

Corin, Etalle and Saptawijaya [12] propose a linear-time temporal logic with past operators for the specification of security protocols and their properties. They also provide a model checking procedure that combines an enumerator of the symbolic execution traces allowed by the protocol with a decision procedure capable to determine whether the given security properties hold on any given symbolic execution trace. Their approach, like ours, allows for the specification of a number of security properties of interest. However the lack of future operators complicates the application of their approach in all the cases (as the one discussed in this paper) in which security assumptions and goals are naturally expressed by means of future operators. More importantly, the DY intruder model is hardwired in their model and they do not allow for different channel types as we do in our approach.

Abstractions for secure channels are provided by process calculi through scoping rules, but these are usually limited to confidential and to authentic channels [1]. The usage of

LTL as specification language allows us to readily model channels that are confidential, authentic and resilient (or a combination thereof), but more work is needed to assess the relative strengths of the two approaches.

## 6. Conclusions

We have presented a general model for security protocols that by combining the set-rewriting formalism and LTL allows for the specification of assumptions on principals and communication channels as well as complex security properties that are normally not handled by state-of-the-art protocol analysers. We have demonstrated the effectiveness of the approach through a thorough analysis of the ASW protocol. Our analysis unveiled a previously unknown attack on the protocol that was not detected by other tools. Our analysis also shows that the use of an expressive logic such as LTL is not only useful for specifying and supporting the mechanical verification of security protocols, but also to support the understanding and the specification of their security requirements.

## Acknowledgements

We are grateful to Jorge Cuellar and Sebastian Mödersheim for stimulating discussions on these topics. This work was partially funded by the FIRB-MIUR Project “Verifica Automatica di Protocolli di Sicurezza per Internet” (RBAU01P5SS).

## References

- [1] M. Abadi, C. Fournet, and G. Gonthier. Secure implementation of channel abstractions. In *LICS*, pages 105–116, 1998.
- [2] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Héam, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In *Proceedings of the 17th International Conference on Computer Aided Verification (CAV'05)*. Springer-Verlag, 2005. Available at [www.avispa-project.org](http://www.avispa-project.org).
- [3] A. Armando and L. Compagna. SAT-based model-checking for security protocols analysis. *International Journal of Information Security (to appear on)*, 2007. Available at <http://www.ai-lab.it/compagna/papers/IJIS-2006/paper.pdf>.
- [4] 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, 1998.
- [5] D. Basin. Lazy infinite-state analysis of security protocols. In R. Baumgart, editor, *Secure Networking — CORE (Secure)'99*, LNCS 1740, pages 30–42. Springer-Verlag, 1999.

- [6] D. Basin, S. Mödersheim, and L. Viganò. Algebraic intruder deductions. In G. Sutcliffe and A. Voronkov, editors, *Proceedings of LPAR'05*, LNAI 3835, pages 549–564. Springer, 2005.
- [7] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic Model Checking without BDDs. In *Proceedings of TACAS'99*, LNCS 1579, pages 193–207. Springer-Verlag, 1999.
- [8] I. Cervesato, N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop: CSFW'99*, pages 55–69. IEEE Computer Society Press, 1999.
- [9] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turunani. An NP Decision Procedure for Protocol Insecurity with XOR. In *Proceedings of the Logic In Computer Science Conference, LICS'03*, pages 261–270, 2003. Available at <http://www.avispa-project.org>.
- [10] J. Clark and J. Jacob. A Survey of Authentication Protocol Literature: Version 1.0, 17. Nov. 1997. URL: [www.cs.york.ac.uk/~jac/papers/drareview.ps.gz](http://www.cs.york.ac.uk/~jac/papers/drareview.ps.gz).
- [11] H. Comon-Lundh and S. Delaune. The finite variant property: How to get rid of some algebraic properties. In J. Giesl, editor, *Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA'05)*, volume 3467 of *Lecture Notes in Computer Science*, pages 294–307, Nara, Japan, Apr. 2005. Springer.
- [12] R. Corin, S. Etalle, and A. Saptawijaya. A logic for constraint-based security protocol analysis. In *IEEE Symposium on Security and Privacy*, 2006.
- [13] D. Dolev and A. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 2(29), 1983.
- [14] 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>.
- [15] F. Jacquemard, M. Rusinowitch, and L. Vigneron. Compiling and Verifying Security Protocols. In M. Parigot and A. Voronkov, editors, *Proceedings of LPAR 2000*, LNCS 1955, pages 131–160. Springer-Verlag, 2000.
- [16] U. Maurer and P. Schmid. A calculus for security bootstrapping in distributed systems. *Journal of Computer Security*, 4(1):55–80, 1996.
- [17] C. Meadows. The NRL Protocol Analyzer: An Overview. *Journal of Logic Programming*, 26(2):113–131, 1996. See <http://chacs.nrl.navy.mil/projects/crypto.html>.
- [18] J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proceedings of the ACM Conference on Computer and Communications Security CCS'01*, pages 166–175, 2001.
- [19] J. Raskin and S. Kremer. Game analysis of abuse-free contract signing. In *15th IEEE Computer Security Foundations Workshop*, pages 206–220. IEEE Computer Society Press, June 2002.
- [20] V. Shmatikov and J. C. Mitchell. Finite-state analysis of two contract signing protocols. *Theoretical Computer Science*, 283(2):419–450, 2002.