LTL model checking for security protocols

Alessandro Armando* — **Roberto Carbone*** — **Luca Compagna****

*AI-Lab, DIST – Università di Genova
Viale Causa 13
16145 Genova (Italy)
armando@dist.unige.it
carbone@dist.unige.it

** 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 greatly complicate or even 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 of complex security properties that are normally not handled by state-of-the-art security protocol analysers. By using our approach we have been able to formalise all the assumptions required by the ASW protocol for optimistic fair exchange and some of its key security properties. Besides the previously reported attacks on the protocol, we report a new attack on a patched version of the protocol.*

KEYWORDS: *security protocols, bounded model checking, SAT-based model checking, multi-set rewriting, LTL, abstract communication channels, ASW.*

DOI:10.3166/JANCL.18.1–25

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 (Meadows, 1996; Basin, 1999; Jacquemard *et al.*, 2000; Millen *et al.*, 2001). 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 (Armando *et al.*, 2005). 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. In other words protocols are assumed to be resistant against a number of common vulnerabilities due, e.g., to weaknesses of the cryptography or to the choice of poorly chosen passwords. This assumption (called the *perfect cryptography assumption*) is strong but it is justified by the observation that many protocols have been found vulnerable to severe attacks that do not rely on these kind of weaknesses. These attacks exploit logical flaws that occur when complex and/or unexpected interleavings of different protocol sessions are considered.

(A2) Communication between honest principals is controlled by a Dolev-Yao (DY) intruder (Dolev *et al.*, 1983), 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 input pattern by sending out messages matching a given output pattern.

(A4) Security properties are expressed as state properties (i.e. invariants). For instance, violation of the secrecy of a message M is expressed as reachability of a state in which the intruder knows M.

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 or the ability to guess and verify secrets have been put forward (see, e.g., (Chevalier *et al.*, 2003; Comon-Lundh *et al.*, 2005; Basin *et al.*, 2005; Lowe, 2002; Cohen, 2002; Hankes Drielsma *et al.*, 2005)).

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. Also, browser-based protocols (Gross *et al.*, 2005) often assume for their proper functioning that communication between the browser and the server is carried over a unilateral SSL 3.0 or TLS 1.0 channel, established through the exchange of a valid certificate. This means that messages exchanged over these channels cannot be intercepted nor forged by the intruder. But the best example is probably the optimistic fair exchange protocol proposed by Asokan, Shoup, and Waidner (ASW) (Asokan *et al.*, 1998). 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 principals 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 (Armando *et al.*, 2002; Armando *et al.*, 2003; Armando *et al.*, 2007) 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 (Armando *et al.*, 2004), that by leveraging on state-of-the-art SAT solvers can compete and in some cases outperform other state-of-the-art protocol analysers. We have recently extended SATMC to support model checking of LTL formulae as described in (Biere *et al.*, 1999).

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

$$M \models (C_I \wedge C_H) \Rightarrow G \tag{1}$$

where M is a labelled transition system modelling the behaviours of the honest principals and of the DY intruder, C_I and C_H are LTL formulae (henceforth called *LTL constraints*) that constrain the allowed behaviours of the intruder and of the honest principals respectively, and G is an LTL formula stating the security properties 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 the property of *fair exchange*, one of the key security properties that the protocol is expected to meet. 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 (Shmatikov *et al.*, 2002). We believe that the ability to detect this attack, which has eluded previous formal analyses of the protocol (Shmatikov *et*

al., 2002; Hankes Drielsma *et al.*, 2004), 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 Txt without the involvement of the TTP. If during the execution of the exchange subprotocol O does not receive the expected reply by R within an acceptable time frame, then he initiates the abort or the resolve subprotocols with the TTP to force the resolution of the contract. Similarly, if during execution of the exchange subprotocol R does not receive the expected reply by O within an acceptable time frame, then he initiates the resolve subprotocol with the TTP to force the resolution of the contract. The TTP keeps a permanent database DB of the contracts 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.

THE EXCHANGE SUBPROTOCOL. The exchange subprotocol starts with O sending R a signed message $me_1 = Sig_O(v(O), v(R), T, Txt, h(N_O))$ containing its own public key $v(O)$, the public key of the responder $v(R)$, the identity of the TTP T, the contractual text Txt, and the hash of a nonce $h(N_O)$. (Without loss of generality in this paper we assume that $Sig_A(e_1, \ldots, e_n)$ stands for $\{\langle e_1, \ldots, e_n \rangle\}_{Ka^{-1}}$, where Ka^{-1} is the private key of A.) Notice that me_1 does not constitute in itself a commitment by O on Txt 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.

The *exchange* subprotocol:

$$\begin{array}{lll} E1.\ O \to R & : & me_1(O, R, Txt, N_O, T) \\ E2.\ R \to O & : & me_2(O, R, Txt, N_O, N_R, T) \\ E3.\ O \to R & : & N_O \\ E4.\ R \to O & : & N_R \end{array}$$

The *abort* subprotocol:

$$\begin{array}{lll} A1.\ O \to T & : & ma_1(O, R, Txt, N_O, T) \\ A2.\ T \to O & : & \textbf{if}\ \text{resolved}(me_1, me_2) \in DB \\ & & \textbf{then}\ Sig_T(me_1, me_2) \\ & & \textbf{else}\ Sig_T(\textit{aborted}, ma_1); \\ & & \quad DB := DB \cup \{\text{aborted}(me_1)\} \end{array}$$

The *resolve* subprotocol:

$$\begin{array}{lll} R1.\ A \to T & : & mr_1 = \langle me_1, me_2 \rangle \\ R2.\ T \to A & : & \textbf{if}\ \text{aborted}(me_1) \in DB \\ & & \textbf{then}\ Sig_T(\textit{aborted}, ma_1) \\ & & \textbf{else}\ Sig_T(me_1, me_2); \\ & & \quad DB := DB \cup \{\text{resolved}(me_1, me_2)\} \end{array}$$

Legenda:

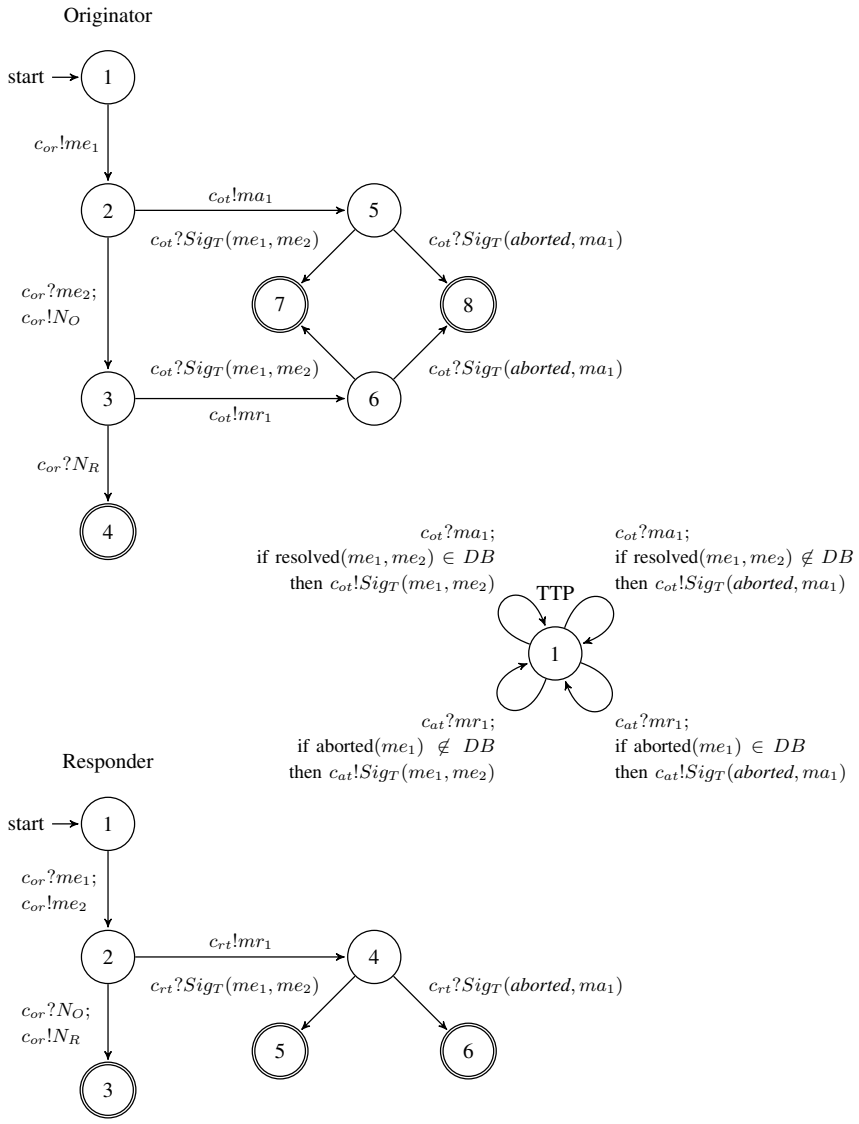
$$\begin{array}{lcl} me_1(O, R, Txt, N_O, T) & := & Sig_O(v(O), v(R), T, Txt, h(N_O)) \\ me_2(O, R, Txt, N_O, N_R, T) & := & Sig_R(me_1(O, R, Txt, N_O, T), h(N_R)) \\ ma_1(O, R, Txt, N_O, T) & := & Sig_O(\textit{aborted}, me_1(O, R, Txt, N_O, T)) \\ mr_1(O, R, Txt, N_O, N_R, T) & := & \langle me_1(O, R, Txt, N_O, T), me_2(O, R, Txt, N_O, N_R, T) \rangle \end{array}$$

For the sake of brevity, we write $me_1, \ldots$ in place of $me_1(O, R, Txt, N_O, T), \ldots$ respectively, whenever the actual arguments can be inferred from the context.

Figure 1. *The ASW protocol*

However things can go wrong either because of a network failure or because a malicious agent diverts 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 initiate the abort subprotocol by sending T the message $ma_1 = Sig_O(\textit{aborted}, me_1)$. Upon receipt of an abort request from O, T looks up its permanent database DB 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



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*

protocol by sending an *abort token*, i.e. a message of the form $Sig_T(\textit{aborted}, ma_1)$ 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 principals.

ASSUMPTIONS ON THE CHANNELS. The communication channels between any two protocol principals 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 principal 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 goals. Here we focus on fair exchange. Fair exchange can be expressed as the conjunction of the following two properties (slightly adapted from (Shmatikov *et al.*, 2002)):

(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.

A protocol session is a run of the protocol played by individual agents and using specific data. As an example consider the protocol session in which O, R and T are

played by Alice, Bob and Charlie respectively using 347282 and 998231 for N_O and N_R respectively, and the string *"Alice will pay €100,00 for Bob's butterfly collection."* for the contractual text Txt. Another protocol session is obtained by letting Alice, Bob and Charlie play the same roles, using the same contractual text, but different numbers for N_O and N_R. A further example is given by the session in which the roles of Alice and Bob are exchanged (i.e. Alice plays R and Bob plays O, while Charlie plays T) using different numbers for N_O and N_R, and the same contractual text.

A *protocol scenario* is a finite set of protocol sessions. Given a protocol scenario Σ and assumptions on the security offered by the communication channels as well as on the expected behaviours of the involved agents, we are interested in the problem of determining whether the concurrent execution of the sessions in Σ enjoys the expected security properties. In the next section we show how this problem can be recast into a model checking problem of the form (1), where M is a labelled transition system modelling the concurrent execution of the sessions in Σ under the assumption that communication channels are controlled by a DY intruder, C_I and C_H are LTL formulae specifying the allowed behaviours of the intruder and of the honest principals respectively, and G is an LTL formula specifying the security properties that the protocol is expected to enjoy.

3. Specifying security protocols with set-rewriting and LTL

In this section we propose a framework based on a rewrite-based formalism (in the line of (Cervesato *et al.*, 1999)) and LTL that allows for the formal specification of the transition system associated with the concurrent execution of the sessions in a given protocol scenario and the associated security properties. We also show that the proposed framework supports the modelling of a number of relevance intruder models as well as the straightforward specification of important security properties.

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 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 properties 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 behaviour of principals*

The model M consists of a labelled transition system modelling the behaviours of the honest principals and of the intruder and their initial state I. The states of the transition system are represented by sets of ground (i.e. variable-free) *facts*, i.e. atomic formulae of the form given in the left column of Table 1 and whose informal

Table 1. *Facts and their informal meaning*

Fact	Meaning
$\texttt{state}_r(j, a, es, s)$	Principal a, playing role r, 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.
$\texttt{ak}(a, m)$	Principal a knows message m. (Therefore, the DY intruder, referred to as $\texttt{i}$, knows m if and only if $\texttt{ak}(\texttt{i}, m)$ holds.
$\texttt{sent}(rs, b, a, m, c)$	Principal rs has sent message m on channel c to principal a pretending to be principal b.
$\texttt{rcvd}(a, b, m, c)$	Message m (supposedly sent by principal b) has been received on channel c by principal a, but a has not processed it yet.
$\texttt{c}(n)$	Term n is the current value of the counter used to issue fresh terms, and is incremented as $\texttt{s}(n)$ every time a fresh term is issued.
$\texttt{contains}(db, m)$	Message m is contained into set db. Sets are used, e.g., to share data between principals.

meaning is explained in the right column. Transitions are represented by *rewrite rules* of the form $(L \xrightarrow{rn(v_1,\ldots,v_n)} R)$, where L and R are finite sets of facts, rn is a *rule name*, i.e. a function symbol uniquely associated with the rule, and $v_1, \ldots, v_n$ are the variables occurring in L. It is required that the variables occurring in R also occur in L. Here and in the sequel we use typewriter font to denote states and rewrite rules with the additional convention that variables are capitalized (e.g. `A`, `Txt`), while constants and function symbols begin with a lower-case letter (e.g. `a`, `txt`). Our rewrite-based formalism is inspired by the Intermediate Format (IF) developed in the context of the AVISPA Project (Armando *et al.*, 2005).

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 local state of the honest principal a is represented by the facts of the form $\texttt{state}_r(j, a, es, s)$ (called *state-facts*) and $\texttt{ak}(a, m)$ occurring in S. (We assume that for each session s and for each principal a there exists at most one fact of the form $\texttt{state}_r(j, a, es, s)$ 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 principals, the initial state I contains a state-fact $\texttt{state}_r(1, a, es, s)$ for each protocol session defined in the protocol scenario Σ, where role r is played by a (with a different from the intruder). Basically, each state-fact in I models the state of a honest principal ready to get involved in a specific protocol session.

Let S be a set of facts, $(L \xrightarrow{rn(v_1,\ldots,v_n)} R)$ a rewrite rule and σ a substitution of the variables $v_1, \ldots, v_n$. We say that *rule (instance)* $\rho = rn(v_1\sigma, \ldots, v_n\sigma)$ *is applicable in* S if and only if $L\sigma \subseteq S$. If $\rho = rn(v_1\sigma, \ldots, v_n\sigma)$ is applicable in S, then $S' = \mathrm{app}_\rho(S) = (S \setminus L\sigma) \cup R\sigma$ is the state resulting from the execution of ρ in S. A *path* π is an alternating sequence of states and rules $S_0\rho_1S_1 \ldots S_{n-1}\rho_nS_n$ such that $S_i = \mathrm{app}_{\rho_i}(S_{i-1})$ (i.e. S_i is a state resulting from the execution of ρ_i in S_{i-1}), for $i = 1, \ldots, n$. If, additionally, $S_0 \subseteq I$, then we say that the path is *initialised*. Let $\pi = S_0\rho_1S_1 \ldots S_{n-1}\rho_nS_n$ be a path, we define $\pi(i) = S_i$ and $\pi_i = S_i\rho_{i+1}S_{i+1} \ldots S_{n-1}\rho_nS_n$; $\pi(i)$ and π_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 behaviour of honest principals is specified by rules of the following form:

$$\mathtt{sent}(\mathtt{RS}, \mathtt{B}, a, \mathtt{M}, \mathtt{C}) \xrightarrow{\mathtt{receive(RS,B,M,C)}} \mathtt{rcvd}(a, \mathtt{B}, \mathtt{M}, \mathtt{C}) \,.\, \mathtt{ak}(a, \mathtt{M}) \tag{2}$$

$$\mathtt{rcvd}(a, b_1, m_1, c_1) \,.\, \mathtt{state}_r(j, a, es_1, \mathtt{S}) \xrightarrow{\mathtt{send}_i(\mathtt{S},\ldots)} \mathtt{sent}(a, a, b_2, m_2, c_2) \,.\, \mathtt{state}_r(l, a, es_2, \mathtt{S}) \tag{3}$$

for all honest principals a and suitable terms b_1, b_2, c_1, c_2, es_1, es_2, m_1, and m_2. Rule (2) models the reception of a message by an honest principal, 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_1 on channel c_1 (supposedly) by b_1, then she can send message m_2 to b_2 on channel c_2 and change her internal state accordingly preparing for step l. 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 that initiates the protocol (i.e. the principal that sends the first message of the protocol), the fact $\mathtt{rcvd}(a, b_1, m_1, c_1)$ is not included in the left hand side of the rule. A similar rule is used to let the principal continue execution when a time out occurs. Also, if the protocol step uses a nonce, then $\mathtt{c}(n)$ and $\mathtt{c}(\mathtt{s}(n))$ are 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 protocol (cf. Figure 2) where a rule represents a role transition from node j to node l. For instance, the transition of the responder from node 1 to node 2 in Figure 2 is modelled by rule (2) and by the following rewrite rule:

$$\mathtt{c}(\mathtt{N}) \,.\, \mathtt{rcvd}(\mathtt{R}, \mathtt{O}, me_1(\mathtt{O}, \mathtt{R}, \mathtt{Txt}, \mathtt{N_O}, \mathtt{T}), \mathtt{C_{OR}}) \,.\, \mathtt{state_{resp}}(1, \mathtt{R}, [\mathtt{O}, \mathtt{T}, \mathtt{Txt}, \mathtt{C_{RT}}, \mathtt{C_{OR}}], \mathtt{S})$$
$$\xrightarrow{\mathtt{send}_2(\mathtt{R},\mathtt{O},\mathtt{N},\mathtt{Txt},\mathtt{N_O},\mathtt{T},\mathtt{C_{OR}},\mathtt{S})}$$
$$\mathtt{c}(\mathtt{s}(\mathtt{N})) \,.\, \mathtt{sent}(\mathtt{R}, \mathtt{R}, \mathtt{O}, me_2(\mathtt{O}, \mathtt{R}, \mathtt{Txt}, \mathtt{N_O}, \mathtt{N}, \mathtt{T}), \mathtt{C_{OR}}) \,.\, \mathtt{ak}(\mathtt{R}, \mathtt{N}) \,.$$
$$\mathtt{state_{resp}}(2, \mathtt{R}, [\mathtt{O}, \mathtt{T}, \mathtt{Txt}, \mathtt{C_{RT}}, \mathtt{C_{OR}}, \mathtt{h}(\mathtt{N_O}), \mathtt{N}], \mathtt{S})$$

The abilities of the DY intruder are modelled by the following rules:

$$\mathtt{ak(i,M)}\,.\,\mathtt{ak(i,A)}\,.\,\mathtt{ak(i,B)}\,.\,\mathtt{ak(i,C)} \xrightarrow{\mathtt{fake(A,B,M,C)}} \mathtt{sent(i,A,B,M,C)}\,.\,\mathit{LHS} \quad (4)$$

$$\mathtt{sent(A,A,B,M,C)} \xrightarrow{\mathtt{intercept(A,B,M,C)}} \mathtt{rcvd(i,A,M,C)}\,.\,\mathtt{ak(i,M)}$$

$$\mathtt{sent(A,A,B,M,C)} \xrightarrow{\mathtt{overhear(A,B,M,C)}} \mathtt{rcvd(i,A,M,C)}\,.\,\mathtt{ak(i,M)}\,.\,\mathit{LHS}$$

where *LHS* abbreviates the left hand side of each rule.

Finally, the inferential capabilities of the principals are modelled by the following rules (where k and $\overline{k}$ are the inverse keys of one another):

$$\mathtt{ak(A,M)}\,.\,\mathtt{ak(A,K)} \xrightarrow{\mathtt{encrypt(A,K,M)}} \mathtt{ak(A,\{M\}_K)}\,.\,\mathit{LHS}$$

$$\mathtt{ak(A,\{M\}}_k\mathtt{)}\,.\,\mathtt{ak(A,}\overline{k}\mathtt{)} \xrightarrow{\mathtt{decrypt(A,\ldots,M)}} \mathtt{ak(A,M)}\,.\,\mathit{LHS}$$

$$\mathtt{ak(A,M_1)}\,.\,\mathtt{ak(A,M_2)} \xrightarrow{\mathtt{pairing(A,M_1,M_2)}} \mathtt{ak(A,\langle M_1,M_2\rangle)}\,.\,\mathit{LHS}$$

$$\mathtt{ak(A,\langle M_1,M_2\rangle)} \xrightarrow{\mathtt{decompose(A,M_1,M_2)}} \mathtt{ak(A,M_1)}\,.\,\mathtt{ak(A,M_2)}\,.\,\mathit{LHS}$$

The language of LTL we consider uses facts and equalities as atomic propositions, the usual propositional connectives (namely, $\neg$, $\vee$, $\wedge$, $\Rightarrow$), the first-order quantifiers $\forall$ and $\exists$, and the temporal operators $\mathbf{F}$ (eventually) and $\mathbf{G}$ (globally). Let $\mathcal{V}$ be the set of variables used in the language and let $\mathcal{T}$ be the set of all ground terms that can be built by using the individual constants and the function symbols occurring in the specification of M. An *assignment over* $\mathcal{V}$ is a total function from $\mathcal{V}$ into $\mathcal{T}$, i.e. $\alpha : \mathcal{V} \to \mathcal{T}$. Assignments are extended to the set of facts in the obvious way. Let π be an initialised path of M and α be an assignment over $\mathcal{V}$, an LTL formula ϕ *is satisfied by* α *in* π, written $\pi \models_\alpha \phi$, if and only if $\pi_0 \models_\alpha \phi$, where $\pi_i \models_\alpha \phi$, with $i \geq 0$, is inductively defined as follows:

$\pi_i \models_\alpha f$	$\alpha(f) \in \pi(i)$ (f is a fact)
$\pi_i \models_\alpha (t_1 = t_2)$	$\alpha(t_1)$ and $\alpha(t_2)$ are the same term
$\pi_i \models_\alpha \neg\phi$	$\pi_i \not\models_\alpha \phi$
$\pi_i \models_\alpha (\phi_1 \vee \phi_2)$	$\pi_i \models_\alpha \phi_1$ or $\pi_i \models_\alpha \phi_2$
$\pi_i \models_\alpha (\phi_1 \wedge \phi_2)$	$\pi_i \models_\alpha \phi_1$ and $\pi_i \models_\alpha \phi_2$
$\pi_i \models_\alpha (\phi_1 \Rightarrow \phi_2)$	$\pi_i \not\models_\alpha \phi_1$ or $\pi_i \models_\alpha \phi_2$
$\pi_i \models_\alpha \mathbf{G}\phi$	$\forall j \geq i.\pi_j \models_\alpha \phi$
$\pi_i \models_\alpha \mathbf{F}\phi$	$\exists j \geq i.\pi_j \models_\alpha \phi$
$\pi_i \models_\alpha \forall x.\phi$	$\pi_i \models_{\alpha[t/x]} \phi$ for all $t \in \mathcal{T}$
$\pi_i \models_\alpha \exists x.\phi$	$\pi_i \models_{\alpha[t/x]} \phi$ for some $t \in \mathcal{T}$

where $\alpha[t/x]$ is the assignment that associate x with t and all other variables y with $\alpha(y)$. We say that ϕ *is valid in* M, in symbols $M \models \phi$, if and only if $\pi \models_\alpha \phi$ for all initialised paths π of M and all assignments α.

3.2. *Constraining the behaviour of the intruder*

Security protocols and services often rely on transport protocols used to establish communication channels enjoying some given security properties. For instance TLS is often used to provide unilateral or bilateral authentic and/or confidential channels. In this section we show how security-relevant properties of communication channels can be specified in our framework by adding suitable LTL formulae to C_I. To illustrate, we focus on confidential and resilient channels, the ones required by our worked-out example.

CONFIDENTIAL CHANNELS. As stated in (Maurer *et al.*, 1996): "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 $\texttt{rcvd}(b, a, m, c) \in S$, then b is one of the principals that have exclusive access to channel c. Thus, the condition that *channel c is confidential to a set of principals ps* can be formalised by the following formula:

$$\mathit{confidential}(c, ps) := \mathbf{G}\,\forall b.\forall a.\forall m.(\texttt{rcvd}(b, a, m, c) \Rightarrow \bigvee_{p \in ps} b = p)$$

RESILIENT CHANNELS. As stated in (Asokan *et al.*, 1998): "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:

$$\mathit{resilient}(c) := \mathbf{G}\,\forall a.\forall b.\forall rs.\forall m.(\texttt{sent}(rs, a, b, m, c) \Rightarrow \mathbf{F}\,\texttt{rcvd}(b, a, m, c))$$

3.3. *Constraining the behaviour of honest principals*

Security protocols and services often rely on assumptions on the behaviour of the principals involved. Principals are more than processes that simply react to predefined messages by sending messages of a predefined form. They are normally assumed to make progress during the execution of the protocol using, e.g., time-out mechanisms to continue execution when specified event is delayed for too long. Some of them are assumed to be always available, others to be reliable, etc. Our set-rewriting formalism coupled with the use of LTL, allows for the specification of these type of assumptions on principals. Hereafter we show how the assumptions required by the ASW protocol can be specified by adding suitably defined LTL formulae to C_H.

PROGRESS. In order to ensure that a principal will not indefinitely wait for a reply from another principal it suffices to assume that whenever it is in a state from which

a time-out can occur, it will change state eventually. Thus, the condition that a *honest principal a, playing role r, cannot stay indefinitely at step j of the protocol session s* can be formalised by the following formula:

$$progress(a, r, j, s) := \mathbf{G}\,\forall es.(\mathtt{state}_r(j, a, es, s) \Rightarrow \mathbf{F}\,\neg\mathtt{state}_r(j, a, es, s)) \quad (5)$$

The ASW protocol assumes that the originator and the responder will not indefinitely wait for an answer from the other party. In our case study this can be specified by means of a conjunction of formulae of the form (5) where $j \in \{2, 3\}$ if $r = \mathtt{orig}$, $j = 2$ if $r = \mathtt{resp}$, and the other parameters are properly instantiated according to the protocol scenario considered. In the sequel we will use $progress\_or$ to denote this conjunction.

AVAILABILITY. The availability of a principal on a certain channel can be imposed by requiring that all the messages received by this principal on that channel will be processed eventually. The condition that *principal a is always available on channel c* can be formalised as:

$$availability(a, c) := \mathbf{G}\,\forall p.\forall m.(\mathtt{rcvd}(a, p, m, c) \Rightarrow \mathbf{F}\,\neg\,\mathtt{rcvd}(a, p, m, c))$$

In the case of the ASW protocol, the TTP is assumed to be always available. This can be captured by adding to C_H a conjunction of availability constraints properly instantiated on Σ, the specific protocol scenario considered. In the sequel we will use $availability\_ttp$ to denote this conjunction.

3.4. *Specifying security properties*

In this section we show how the use of LTL allows for the specification of the security properties that the protocol is expected to enjoy. We focus on the property of fair exchange required by the ASW protocol, and we show how it can be specified in LTL. In the following we consider goals A and B, corresponding to the properties (A) and (B) respectively described in Section 2.

GOAL A. The goal amounts to requiring that if some protocol participant obtains a valid contract binding the other participant to some contractual 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 $Sig_t(me_1, me_2)$ where t is a TTP). More precisely, the fact that principal p has a valid contract binding o and r (as originator and responder respectively) to contractual

text txt using n_O and n_R as secret commitments and t as TTP is formalised by the formula:

$$has\_vc(p, o, r, txt, n_O, n_R, t) :=$$
$$(\mathtt{ak}(p, me_1(o, r, txt, n_O, t)) \wedge \mathtt{ak}(p, me_2(o, r, txt, n_O, n_R, t)) \wedge \mathtt{ak}(p, n_O) \wedge \mathtt{ak}(p, n_R))$$
$$\vee\ \mathtt{ak}(p, Sig_t(me_1(o, r, txt, n_O, t), me_2(o, r, txt, n_O, n_R, t)))$$

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

$$\mathbf{G}\, \forall n_O. \forall n_R. (has\_vc(o, o, r, txt, n_O, n_R, t) \Rightarrow \mathbf{F}\, has\_vc(r, o, r, txt, n_O, n_R, t)) \quad (6)$$
$$\mathbf{G}\, \forall n_O. \forall n_R. (has\_vc(r, o, r, txt, n_O, n_R, t) \Rightarrow \mathbf{F}\, has\_vc(o, o, r, txt, n_O, n_R, t)) \quad (7)$$

for all sessions in the protocol scenario Σ where o, r, t play O, R, T resp., using txt as contractual text Txt. Formula (6) 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 (7) 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. In fact, in order to avoid spurious attacks, the abort token must be the one really expected by the agent. In our model, the state-fact of the agent (and in particular its knowledge) is taken into account during the processing rule (3) of a previously received message. Among all the abort tokens received, only the one containing a nonce that matches the one generated by the agent can be processed. 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

$$\mathbf{G}\, \forall n_O. \forall n_R. \forall n'_R. (has\_at(a, o, r, txt, n_O, n_R, t, s_a) \Rightarrow \mathbf{G} \neg has\_vc(b, o, r, txt, n_O, n'_R, t))$$

for all sessions s_a in the protocol scenario Σ where o, r, t play O, R, T resp., using txt as contractual text Txt, 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 $\mathtt{state}_{\mathtt{orig}}(8, o, [r, txt, n_O, n_R, t], s_a)$ if $a = o$ and $\mathtt{state}_{\mathtt{resp}}(6, r, [o, txt, n_O, n_R, t], s_a)$ if $a = r$.

4. Analysis of the ASW protocol

We have extended SATMC, our SAT-based Model Checker for automatic analysis of security protocols, to support model checking of LTL formulae (Carbone, 2009).

At the core of SATMC lies an automatic procedure that, given an integer k, first computes a finite over-approximation of the states that are reachable in up to k steps and then uses this over-approximation to generate a propositional formula whose satisfying assignments (if any) correspond to counterexamples of (1) (i.e. execution traces of M that falsify $(C_I \wedge C_H) \Rightarrow G$) of length bounded by k. Finding attacks (of length k) on the protocol therefore boils down to solving propositional satisfiability problems. SATMC relies on state-of-the-art SAT solvers for this task which can handle propositional satisfiability problems with hundreds of thousands of variables and clauses or more. SATMC can be instructed to perform an iterative deepening on k. (More details can be found in (Armando *et al.*, 2007).)

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: a 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 stating the resilience of 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 $availability\_ttp$; 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 corresponding to that used in (Shmatikov *et al.*, 2002). 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 the intruder if our definition (cf. 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 we 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 messages sent over the channel with a key known only to (or shared with) the receiver. In this way the intruder (or any other agent) can easily identify, store and replay 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 used 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 $\mathit{confidential}(c_{or}, \{o, r\})$, $\mathit{confidential}(c_{ot}, \{o, t\})$ and $\mathit{confidential}(c_{rt}, \{r, t\})$.

The analysis of the protocol w.r.t. Goal A, i.e. when $G = G_A$, revealed the attack of Figure 3. 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 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 (Shmatikov *et al.*, 2002), but it must be noted that the attack in Figure 3 is simpler as the TTP is not involved.

$$
\begin{array}{lllll}
E1. & O \to I & : & me_1 \\
E2. & I \to O & : & me_2 \\
 & \multicolumn{3}{l}{I \text{ computes new random } N'_I \text{ and then } me'_2} \\
E3. & O \to I & : & N_O \\
E4. & I \to O & : & N_I
\end{array}
$$

Figure 3. *Attack on the ASW protocol violating Goal A*

In (Shmatikov *et al.*, 2002) Shmatikov and Mitchell propose to repair the protocol by replacing steps E1 and E2 of Figure 1 with the following two:

$$
\begin{array}{lllll}
E3'. & O \to R & : & Sig_O(N_O, h(N_R)) \\
E4'. & R \to O & : & Sig_R(N_R, h(N_O))
\end{array}
$$

SATMC confirms that the improved version of the protocol does not suffer from the attack of Figure 3, but it detects a new (i.e. previously unknown) attack which is shown in Figure 4. 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 (Shmatikov *et al.*, 2002) 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. In fact 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 (Raskin *et al.*, 2002)) to accept the asymmetry in the

$$
\begin{array}{rlcl}
E1. & O \to I & : & me_1 \\
E2. & I \to O & : & me_2 \\
 & I \text{ computes new random } N'_I \text{ and then } me'_2 & & \\
E3'. & O \to I & : & Sig_O(N_O, h(N_I)) \\
E4'. & I \to O & : & Sig_I(N_I, h(N_O)) \\
R1. & I \to T & : & mr_1 = \langle me_1, me'_2 \rangle \\
R2. & T \to I & : & mr_2 = Sig_T(me_1, me'_2)
\end{array}
$$

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

protocol and weaken the goal. The new formula G'_A can be obtained by replacing (7) in the definition of G_A with the following formula:

$$\mathbf{G}\,\forall n_O.\forall n_R.\exists n'_R.(has\_vc(r, o, r, txt, n_O, n_R, t) \Rightarrow \mathbf{F}\; has\_vc(o, o, r, txt, n_O, n'_R, t))$$

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:

$$\begin{aligned}\mathbf{G}\,\forall n_O.\forall n_R.\forall n'_R.(has\_at(a, o, r, txt, n_O, n_R, t, s_a) \Rightarrow \\ (has\_vc(b, o, r, txt, n_O, n'_R, t) \vee \mathbf{G} \neg has\_vc(b, o, r, txt, n_O, n'_R, t)))\end{aligned}$$

for all sessions s_a in the protocol scenario Σ where o, r, t play O, R, T resp., using txt as contractual text Txt, 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 4.1. 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 (Hankes Drielsma *et al.*, 2004). 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.

$$
\begin{array}{llcl}
E1. & I \to R & : & me_1 \\
E2. & R \to I & : & me_2 \\
A1. & I \to T & : & ma_1 = Sig_I(\textit{aborted}, me_1) \\
A2. & T \to I & : & ma_2 = Sig_T(\textit{aborted}, ma_1) \\
R1. & R \to T & : & mr_1 = \langle me_1, me_2 \rangle \\
R2. & T \to R & : & mr_2 = Sig_T(\textit{aborted}, ma_1) \\
E1. & I \to R & : & me_1 \\
E2. & R \to I & : & me'_2 \\
E3. & I \to R & : & N_I \\
E4. & R \to I & : & N'_R
\end{array}
$$

Figure 5. *Attack on the ASW protocol violating Goal B'*

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 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:

$$
\begin{aligned}
\mathbf{G}\, \forall n_O.\forall n_R.\forall n'_R.((has\_at(a,o,r,txt,n_O,n_R,t,s_a) \Rightarrow \\
(has\_vc(b,o,r,txt,n_O,n'_R,t) \vee \\
\mathbf{G} \neg\, has\_vc(b,o,r,txt,n_O,n'_R,t) \vee \\
\mathbf{F}\, has\_vc(a,o,r,txt,n_O,n'_R,t))))
\end{aligned}
$$

for all sessions s_a in the protocol scenario Σ where o, r, t play O, R, T resp., using txt as contractual text Txt, 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 (Shmatikov *et al.*, 2002), 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 6. Here the intruder overhears a session of the exchange protocol played by O 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 (Shmatikov *et al.*, 2002).

$$
\begin{array}{llll}
E1. & O \rightarrow R & : & me_1 \\
E2. & R \rightarrow O & : & me_2 \\
E3. & O \rightarrow R & : & N_O \\
E4. & R \rightarrow O & : & N_R \\
E1. & I(O) \rightarrow R & : & me_1 \\
E2. & R \rightarrow O & : & me'_2 \quad \textit{// } I \textit{ intercepts} \\
E3. & I(O) \rightarrow R & : & N_O \\
E4. & R \rightarrow O & : & N'_R \quad \textit{// } I \textit{ intercepts}
\end{array}
$$

Figure 6. *Replay attack on the ASW protocol violating Goal A*

Considering the improved version of the protocol, SATMC also found a replay attack on Goal A (shown in Figure 7), which is similar to that of Figure 4. 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 $G = G'_A$).

$$
\begin{array}{rlcl}
E1. & O \to R & : & me_1 \\
E2. & R \to O & : & me_2 \\
E3'. & O \to R & : & Sig_O(N_O, h(N_R)) \\
E4'. & R \to O & : & Sig_I(N_R, h(N_O)) \\
R1. & I(R) \to T & : & mr_1 = \langle me_1, me_2' \rangle \\
R2. & T \to R & : & mr_2 = Sig_T(me_1, me_2')
\end{array}
$$

Figure 7. *Replay attack on the patched version of the ASW protocol violating Goal A*

5. Related work

In this section we compare our work with related ones from the area of model checking of contract signing protocols, modelling and reasoning about secure channels, and LTL model checking for security protocols analysis.

5.1. *Model checking of contract signing protocols*

Shmatikov and Mitchell (Shmatikov *et al.*, 2002) and, more recently, Drielsma and Mödersheim (Hankes Drielsma *et al.*, 2004) 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 4 on the "patched" version of the ASW protocol proposed in (Shmatikov *et al.*, 2002) 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 (Raskin *et al.*, 2002) 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 there is not an independent principal playing the intruder, but all the principals can play either honestly or maliciously according to a weaker threat model than DY (e.g., no inferential capabilities on knowledge). Moreover they do not consider parallel executions of the protocol, using the strong hypotheses that different parallel executions cannot interfere, and—as pointed out in (Kähler *et al.*, 2007a)—the formalisation of the fairness property they provide is too weak requiring the agent involved to use the protocol in a "smart" way.

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 principals, 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 (Clark *et al.*, n.d.) and the industrial-scale security protocols of the AVISPA Library (Armando *et al.*, 2005).

Kähler et al. (Kähler *et al.*, 2007b) provide important theoretical results for the Alternating-time μ-Calculus-model checking problem over infinite-state concurrent games structures induced by protocols and the Dolev-Yao intruder. In particular, they show that, when making reasonable assumptions on the behaviour of honest principals and on secure channels, the model checking problem is decidable for a fragment of Alternating-time μ-Calculus, which contains, for example, all properties formulated in (Raskin *et al.*, 2002) and presumably the ones in (Corin *et al.*, 2006). However no implementation of the techniques proposed is provided.

Klay and Vigneron (Klay *et al.*, 2008) propose a framework for analysing non-repudiation protocols in presence of an active intruder. Similarly to our approach, they define new predicates permitting to access the knowledge of protocol participants. Thus, they can specify properties based on the knowledge of participants, even though they are limited to state invariants.

5.2. *Modelling and reasoning about secure channels*

Dilloway and Lowe (Dilloway *et al.*, 2007) provide a fine-grained hierarchy of secure channels. Their definition of confidential channels is similar to our definition of weak confidentiality (see Section 4), while the notion of strong confidentiality we describe does not seem to be taken into account.

Mödersheim and Viganò (Mödersheim *et al.*, 2008) provide a framework for the specification of secure channels. They consider three basic kinds of channels — authentic, confidential, and secure (i.e. channels that are both authentic and confidential) — defining the specification of their properties either as assumptions or as goals. Besides this channels, they allow the specification of pseudonymous channels, guaranteeing the receiver that the messages come from the same source, whose real identity is not known. This concept is also referred to as sender invariance (Drielsma *et al.*, 2006). This type of channel is expressed by identifying a participant with a pseudonym, instead of using its real name. A similar notion can be modelled in our framework by using LTL, as shown in (Armando *et al.*, 2008), where a unilateral TLS channel is formalised. Nevertheless, a detailed understanding of the peculiarity of these two approaches needs further investigation. Our formalism allows also for the specification of other channel types, which can be characterized by restricting the traces that are allowed. For instance, this is the case of resilient channels obtained by excluding traces where sent messages are never received.

Cederquist and Dashti (Cederquist *et al.*, 2006) investigate the suitability of the DY intruder model for automatic verification of liveness properties under the resilient communication channels assumption. They demonstrate that their definition of resilient channels cannot be expressed in LTL. In particular, they require that a resilient network may destroy a message when it would not be accepted by its recipient. In our modelling of resilient channels we impose that all messages are eventually received by the intended receiver, as required in (Asokan *et al.*, 1998).

Abstractions for secure channels are provided by process calculi through scoping rules, but these are usually limited to confidential and to authentic channels (Abadi *et al.*, 1998). 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.

5.3. *LTL model checking for security protocols analysis*

Corin, Etalle and Saptawijaya (Corin *et al.*, 2006) 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.

6. Conclusions

We have presented a general model for security protocols that by combining a 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 supported by the FP7-ICT-2007-1 Project no. 216471, "AVANTSSAR: Automated Validation of Trust and Security of Service-oriented Architectures" (www.avantssar.eu).

7. References

Abadi M., Fournet C., Gonthier G., "Secure Implementation of Channel Abstractions", *LICS*, pp. 105–116, 1998.

Armando A., Basin D., Boichut Y., Chevalier Y., Compagna L., Cuellar J., Hankes Drielsma P., Heám P.-C., Kouchnarenko O., Mantovani J., Mödersheim S., von Oheimb D., Rusinowitch M., Santiago J., Turuani M., Viganò L., Vigneron L., "The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications", *Proceedings of the 17th International Conference on Computer Aided Verification (CAV'05)*, Springer-Verlag, 2005. Available at www.avispa-project.org.

Armando A., Carbone R., Compagna L., Cuellar J., Abad L. T., "Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps", *in* V. Shmatikov (ed.), *Proceedings of the 6th ACM Workshop on Formal Methods in Security Engineering (FMSE 2008)*, ACM Press, pp. 1–10, 2008.

Armando A., Compagna L., "Automatic SAT-Compilation of Protocol Insecurity Problems via Reduction to Planning", *in* D. Peled, M. Vardi (eds), *Proceedings of 22nd IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE)*, LNCS 2529, Springer-Verlag, pp. 210–225, 2002. Also presented at the FCS & Verify Workshops, Copenhagen, Denmark, July 2002. Available at www.avispa-project.org.

Armando A., Compagna L., "SATMC: a SAT-based Model Checker for Security Protocols", *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA'04)*, vol. 3229 of *LNAI*, Springer-Verlag, Lisbon, Portugal, pp. 730-733, 2004.

Armando A., Compagna L., "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.

Armando A., Compagna L., Ganty P., "SAT-based Model-Checking of Security Protocols using Planning Graph Analysis", *in* K. Araki, S. Gnesi, D. Mandrioli (eds), *Proceedings of the 12th International Symposium of Formal Methods Europe (FME)*, LNCS 2805, Springer-Verlag, pp. 875–893, 2003. Available at www.avispa-project.org.

Asokan N., Shoup V., Waidner M., "Asynchronous protocols for optimistic fair exchange", *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pp. 86–99, 1998.

Basin D., "Lazy Infinite-State Analysis of Security Protocols", *in* R. Baumgart (ed.), *Secure Networking — CQRE (Secure)'99*, LNCS 1740, Springer-Verlag, pp. 30–42, 1999.

Basin D., Mödersheim S., Viganò L., "Algebraic Intruder Deductions", *in* G. Sutcliffe, A. Voronkov (eds), *Proceedings of LPAR'05*, LNAI 3835, Springer, pp. 549–564, 2005.

Biere A., Cimatti A., Clarke E., Zhu Y., "Symbolic Model Checking without BDDs", *Proceedings of TACAS'99*, LNCS 1579, Springer-Verlag, pp. 193–207, 1999.

Carbone R., LTL Model-Checking for Security Protocols, Phd, Università degli Studi di Genova, Italy, 2009.

Cederquist J., Dashti M. T., "An intruder model for verifying liveness in security protocols", *in* M. Winslett, A. D. Gordon, D. Sands (eds), *FMSE*, ACM, pp. 23-32, 2006.

Cervesato I., Durgin N., Lincoln P., Mitchell J., Scedrov A., "A meta-notation for protocol analysis", *Proceedings of the 12th IEEE Computer Security Foundations Workshop: CSFW'99*, IEEE Computer Society Press, pp. 55–69, 1999.

Chevalier Y., Küsters R., Rusinowitch M., Turuani M., "An NP Decision Procedure for Protocol Insecurity with XOR", *Proceedings of the Logic In Computer Science Conference, LICS'03*, pp. 261–270, 2003. Available at `http://www.avispa-project.org`.

Clark J., Jacob J., "A Survey of Authentication Protocol Literature: Version 1.0, 17. Nov. 1997", n.d., URL: `www.cs.york.ac.uk/~jac/papers/drareview.ps.gz`.

Cohen E., "Proving Protocols Safe from Guessing", *Proceedings of Foundations of Computer Security 2002*, pp. 85–92, 2002.

Comon-Lundh H., Delaune S., "The finite variant property: How to get rid of some algebraic properties", *in* J. Giesl (ed.), *Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA'05)*, vol. 3467 of *Lecture Notes in Computer Science*, Springer, Nara, Japan, pp. 294-307, April, 2005.

Corin R., Etalle S., Saptawijaya A., "A Logic for Constraint-based Security Protocol Analysis", *IEEE Symposium on Security and Privacy*, 2006.

Dilloway C., Lowe G., "On the Specification of Secure Channels", *Proceedings of the Workshop on Issues in the Theory of Security (WITS '07)*, 2007.

Dolev D., Yao A., "On the Security of Public-Key Protocols", *IEEE Transactions on Information Theory*, 1983.

Drielsma P. H., Mödersheim S., Viganò L., Basin D. A., "Formalizing and Analyzing Sender Invariance", *in* T. Dimitrakos, F. Martinelli, P. Y. A. Ryan, S. A. Schneider (eds), *Formal Aspects in Security and Trust*, vol. 4691 of *Lecture Notes in Computer Science*, Springer, pp. 80-95, 2006.

Gross T., Pfitzmann B., Sadeghi A.-R., "Browser Model for Security Analysis of Browser-Based Protocols", *Proceedings of the Eight ESORICS*, vol. 3679 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin Germany, October, 2005.

Hankes Drielsma P., Mödersheim S., "The ASW Protocol Revisited: A Unified View", *Proceedings of the IJCAR04 Workshop ARSPA*, 2004. To appear in ENTCS, available at `http://www.avispa-project.org`.

Hankes Drielsma P., Mödersheim S., Viganò L., "A Formalization of Off-Line Guessing for Security Protocol Analysis", *in* F. Baader, A. Voronkov (eds), *Proceedings of LPAR'04*, vol. 3452 of *LNAI*, Springer, pp. 363–379, 2005.

Jacquemard F., Rusinowitch M., Vigneron L., "Compiling and Verifying Security Protocols", *in* M. Parigot, A. Voronkov (eds), *Proceedings of LPAR 2000*, LNCS 1955, Springer-Verlag, pp. 131–160, 2000.

Kähler D., Küsters R., Truderung T., Infinite State AMC-Model Checking for Cryptographic Protocols, Technical Report num. 0702, Institut für Informatik, CAU Kiel, Germany, 2007a.

Kähler D., Küsters R., Truderung T., "Infinite State AMC-Model Checking for Cryptographic Protocols", *Proceedings of the Twenty-Second Annual IEEE Symposium on Logic in Computer Science (LICS 2007)*, IEEE, Computer Society Press, pp. 181-190, 2007b.

Klay F., Vigneron L., "Automatic Methods for Analyzing Non-Repudiation Protocols with an Active Intruder", *in* P. Degano, J. Guttman, F. Martinelli (eds), *5th International Workshop on Formal Aspects in Security and Trust (FAST)*, pp. 165–180, 2008. To appear as LNCS.

Lowe G., "Analysing protocols subject to guessing attacks", *in* J. Guttman (ed.), *Workshop on Issues in the Theory of Security (WITS'02)*, Portland, Oregon, USA, January, 2002.

Maurer U. M., Schmid P. E., "A Calculus for Security Bootstrapping in Distributed Systems", *Journal of Computer Security*, vol. 4, num. 1, pp. 55-80, 1996.

Meadows C., "The NRL Protocol Analyzer: An Overview", *Journal of Logic Programming*, vol. 26, num. 2, pp. 113–131, 1996. See `http://chacs.nrl.navy.mil/projects/crypto.html`.

Millen J. K., Shmatikov V., "Constraint solving for bounded-process cryptographic protocol analysis", *Proceedings of the ACM Conference on Computer and Communications Security CCS'01*, pp. 166-175, 2001.

Mödersheim S., Viganò L., Secure Pseudonymous Channels, Technical Report num. RZ3724, IBM Zurich Research Lab, 2008. `domino.research.ibm.com/library/cyberdig.nsf`.

Raskin J., Kremer S., "Game Analysis of Abuse-free Contract Signing", *15th IEEE Computer Security Foundations Workshop*, IEEE Computer Society Press, pp. 206–220, June, 2002.

Shmatikov V., Mitchell J. C., "Finite-state analysis of two contract signing protocols", *Theoretical Computer Science*, vol. 283, num. 2, pp. 419–450, 2002.