URL: http://www.elsevier.nl/locate/entcs/volume86.html 16 pages

A Rewriting Strategy for Protocol Verification [1]

Monica Nesi [2], Giuseppina Rucci and Massimo Verdesca

Dipartimento di Informatica
Università degli Studi di L'Aquila
via Vetoio, 67010 L'Aquila, Italy

Abstract

This paper presents a rewriting strategy for the analysis and the verification of communication protocols. In a way similar to the approximation technique defined by Genet and Klay, a rewrite system $\mathcal{R}$ specifies the protocol and a tree automaton $\mathcal{A}$ describes the initial set of communication requests. Given a term t that represents a property to be proved, a rewriting strategy is defined that suitably expands and reduces t using the rules in $\mathcal{R}$ and the transitions in $\mathcal{A}$ to derive whether or not t is recognized by an intruder. This is done by simulating a completion process in a bottom-up manner starting from t and trying to derive a transition $t \rightarrow q_f$ from critical pairs, where q_f is a final state of the tree automaton. The rewriting strategy is defined through a set of inference rules and used for reasoning about the authentication and secrecy properties of the Needham-Schroeder public-key protocol.

1 Introduction

The analysis and the verification of communication and security protocols have recently become an important subject of research, mostly due to the development of the Internet and related electronic commerce services. Several approaches have been applied to protocol specifications in order to formally prove various properties of interest, such as authentication, secrecy or confidentiality, freshness of nonces, etc. These approaches range from model checking [17,19] to theorem proving [18,25,26,28] through rewriting techniques and strategies [2,6,14] combined with tree automata and abstract interpretation [9,11,20]. There is also ongoing work on combining different approaches, such as the combination of Genet and Klay's approximation technique with Paulson's inductive method in Isabelle [22,23].

[1] Work partially supported by MURST 40% "Architetture Software per Infrastrutture di Rete ad Accesso Eterogeneo (SAHARA)".

[2] Email: monica@di.univaq.it

We are interested in the approach based on the approximation technique developed in [8,9,11] that uses rewrite systems and tree automata to specify and verify properties of security protocols. This technique aims at finding that there are no attacks on a protocol, rather than at discovering attacks. The protocol is operationally specified through a rewrite system $\mathcal{R}$, while the initial set E of communication requests is described through a tree automaton $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) \supseteq E$. Starting from $\mathcal{R}$ and $\mathcal{A}$, the approximation technique by Genet and Klay builds a tree automaton which over-approximates the set of the messages exchanged among the protocol agents. In other words, the language recognized by the resulting approximation automaton $\mathcal{T}_{\mathcal{R}}\uparrow(\mathcal{A})$ includes all $\mathcal{R}$-descendants of E. In this way, in order to prove whether a property p is satisfied, it is sufficient to consider the intersection between the language of $\mathcal{T}_{\mathcal{R}}\uparrow(\mathcal{A})$ and the language of a tree automaton $\mathcal{A}_{\overline{p}}$ which models the negation of p and thus contains the "prohibited" terms. If such intersection is empty, then p is satisfied.

Like in the approach based on the approximation technique, we consider a rewrite system $\mathcal{R}$ to specify the protocol and a tree automaton $\mathcal{A}$ to represent the initial set E of communication requests. Given a term t that describes a property to be proved, a rewriting strategy is defined that suitably expands and reduces t using the rules in $\mathcal{R}$ and the transitions in $\mathcal{A}$ to derive whether or not t is recognized by the intruder. This is done by simulating a completion process in a bottom-up manner starting from t and trying to derive if a transition $t \rightarrow q_f$ can be generated from critical pairs, where q_f is a final state of $\mathcal{A}$. If the transition $t \rightarrow q_f$ is derived by the strategy, this means that the term t is recognized by the intruder and thus the property represented by t is not satisfied by the protocol. If $t \rightarrow q_f$ is not derived, then the property given by t is true. Note that no approximation is carried out on the transitions when applying the bottom-up derivation process.

The strategy is inspired by a previously defined rewriting strategy to deal with the problem of divergence of the completion process [12,13]. The automaton $\mathcal{T}_{\mathcal{R}}\uparrow(\mathcal{A})$ resulting from the approximation technique is characterized by a finite set of transitions, but their number can be very high. We think that also in this case it can be worth applying the bottom-up strategy, as it generates a small subset of transitions for any given input term t. Moreover, the strategy can help finding attacks when the language intersection is not empty and there may be an attack.

The protocol specification under consideration is the Needham-Schroeder Public-Key protocol [21] (NSPK from now on), which is the typical first case study of protocol verification. This simple protocol, developed in 1978, was proved insecure only many years later by Lowe [16] and Meadows [18].

The paper is organized as follows. Section 2 describes the NSPK and its formalization as a rewrite system. In Section 3 the approximation technique is briefly recalled and discussed. The rewriting strategy is introduced and defined by means of inference rules in Section 4 and then shown to be correct,

terminating and complete. In Section 5 the strategy is applied to the specification of the NSPK for proving the properties of secrecy and authentication. The paper ends with some concluding remarks and directions for future work.

2 The NSPK Protocol

The NSPK [21] aims at the mutual authentication of two agents communicating through an insecure network. Every agent A has a public key K_A, that is known to any agent and can be obtained from a key server, and a secret key K_A^{-1}, the inverse of K_A, that is supposed to be known only to A. A message m crypted with a key K is denoted $\{m\}_K$. Thus, any agent can encrypt a message m with the public key of any other agent A by producing $\{m\}_{K_A}$, but only A can decrypt such a message using her/his private key K_A^{-1} (if the secrecy of the private key is guaranteed). The protocol is based on the exchange of messages made of *nonces* $N_A, N_B, \ldots$ (random numbers freshly generated) where the index denotes the agent that has created the nonce. Given two agents A and B, the NSPK protocol (without key server) can be described as follows:

$$
\begin{aligned}
&1.\ A \longrightarrow B\ :\ \{N_A, A\}_{K_B}\\
&2.\ B \longrightarrow A\ :\ \{N_A, N_B\}_{K_A}\\
&3.\ A \longrightarrow B\ :\ \{N_B\}_{K_B}
\end{aligned}
$$

The agent A (initiator of the communication) sends a message (crypted with the public key of B) to the agent B. The message of A contains a nonce N_A and her/his identity. The responder B can decrypt the message and sends back to A a message encrypted with the public key of A, containing the nonce N_A that B has received and a nonce N_B. In the last step of the protocol, A returns the nonce N_B to B.

This version of the NSPK has been proved insecure by Lowe [16] and Meadows [18]. In particular, the property of authentication (the form of authentication called *entity authentication* in [27]) fails. As shown by Lowe in [16], the attack involves two parallel interleaving executions of the protocol, where in the first session the agent A establishes a communication with the intruder I, and in the second session I impersonates A and establishes a valid communication with the agent B. At the end of the executions of the protocol, B thinks (s)he has established a communication with A, while (s)he is communicating with I that has impersonated A.

The corrected version of the NSPK proposed by Lowe [17] is the following:

$$
\begin{aligned}
&1.\ A \longrightarrow B\ :\ \{N_A, A\}_{K_B}\\
&2.\ B \longrightarrow A\ :\ \{N_A, N_B, B\}_{K_A}\\
&3.\ A \longrightarrow B\ :\ \{N_B\}_{K_B}
\end{aligned}
$$

It introduces the identity of the responder B in the message exchanged in the second step of the protocol, and the initiator A checks this identity against that of the agent with whom A has started the communication. In this way an intruder I cannot send along a message from B to A as if it was her/his own.

The NSPK can be operationally described by a rewrite system $\mathcal{R}$ [9]. [3] Every protocol step is formalized by means of a rewrite rule whose left hand side expresses a condition on the current state of the protocol (received messages and communication requests), and the right hand side is the message to be sent if the condition is satisfied. The rewrite system $\mathcal{R}$ we have been using for experimenting our strategy on the insecure NSPK is given below. We have $\mathcal{R} = \mathcal{R}_P \cup \mathcal{R}_I$, where the rules in $\mathcal{R}_P = (1) \div (5)$ describe the steps of the protocol, and the rules in $\mathcal{R}_I = (6) \div (10)$ define the intruder's ability of decomposing and decrypting messages.

$$goal(agt(a), agt(b)) \tag{1}$$
$$\rightarrow mesg(agt(a), agt(b), encr(pubkey(agt(b)), agt(a), cons(N(agt(a), agt(b)), agt(a))))$$

$$mesg(agt(a_4), agt(b), encr(pubkey(agt(b)), agt(a_3), cons(a_1, agt(a)))) \tag{2}$$
$$\rightarrow mesg(agt(b), agt(a), encr(pubkey(agt(a)), agt(b), cons(a_1, N(agt(b), agt(a)))))$$

$$mesg(agt(a_6), agt(a), encr(pubkey(agt(a)), agt(a_5), cons(N(agt(a), agt(b)), a_2))) \tag{3}$$
$$\rightarrow mesg(agt(a), agt(b), encr(pubkey(agt(b)), agt(a), a_2))$$

$$mesg(agt(a_6), agt(a), encr(pubkey(agt(a)), agt(a_5), cons(N(agt(a), agt(b)), a_2))) \tag{4}$$
$$\rightarrow c\_init(agt(a), agt(b), agt(a_5))$$

$$mesg(agt(a_8), agt(b), encr(pubkey(agt(b)), agt(a_7), N(agt(b), agt(a)))) \tag{5}$$
$$\rightarrow c\_resp(agt(b), agt(a), agt(a_7))$$

$$\cup(cons(x, y), z) \rightarrow x \tag{6}$$

$$\cup(cons(x, y), z) \rightarrow y \tag{7}$$

$$encr(pubkey(agt(0)), y, z) \rightarrow z \tag{8}$$

$$encr(pubkey(agt(s(x))), y, z) \rightarrow z \tag{9}$$

$$mesg(x, y, z) \rightarrow z \tag{10}$$

Let $\mathcal{F}$ denote the signature of $\mathcal{R}$. Let L_{agt} be an infinite set of agent labels. We are interested in the behaviour of two agents A and B, and all other agents are coded through natural numbers built using the constructors 0 and s. Thus, $L_{agt} = \{A, B\} \cup \mathbb{N}$. $agt(l)$ denotes an agent whose label is $l \in L_{agt}$. $mesg(x, y, c)$ denotes a message from x to y with contents c. $pubkey(a)$ represents the public key of an agent a and $encr(k, a, c)$ denotes the result of encrypting c with the key k (a is a flag that stores the agent that

[3] We refer to [1,7] for more details about rewrite systems and to [5] for the theory of tree automata.

did the encryption). A term $N(x, y)$ denotes a nonce generated by agent x for communicating with agent y. $goal(x, y)$ denotes that x tries to establish a communication with y. $c\_init(x, y, z)$ represents the fact that x thinks of communicating with y but in fact x has established a communication with z. $c\_resp(x, y, z)$ denotes that x thinks (s)he has replied to a communication requested by y but in fact (s)he has replied to a communication requested by z. A list made of x and y is represented by $cons(x, y)$.

Note that $\mathcal{R}$ is slightly different from the system given in [9,22,23]. In particular, the LHS operator, that keeps track of the steps applied to yield a certain term, and the *add* operator are omitted, and the $\cup$ operator is removed from some rewrite rules, except for rules (6) and (7).

3 The Approximation Technique

Many approaches to the verification of security protocols have proved the authentication property (besides other properties) for the corrected version of the NSPK. The approach based on the approximation technique [8,9,11] is one of them. This technique allows one to automatically compute an over-approximation of the set $\mathcal{R}^*(E)$ of the $\mathcal{R}$-descendants of a set of terms E, where E is described through a tree automaton $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) \supseteq E$. The approximation automaton $\mathcal{T}_{\mathcal{R}}\!\uparrow\!(\mathcal{A})$ is such that $\mathcal{L}(\mathcal{T}_{\mathcal{R}}\!\uparrow\!(\mathcal{A})) \supseteq \mathcal{R}^*(E)$. The quality of the approximation depends on an *approximation function* γ which defines the so-called folding positions, i.e. subterms that can be approximated.

We recall the idea of the approximation technique very briefly. Starting from $\mathcal{A}_0 = \mathcal{A}$, the technique builds a finite number of tree automata $\mathcal{A}_i$ with transitions Δ_i such that for all $i \geq 0$ we have $\mathcal{L}(\mathcal{A}_i) \subset \mathcal{L}(\mathcal{A}_{i+1})$ until an automaton $\mathcal{A}_k$ is obtained such that $\mathcal{L}(\mathcal{A}_k) \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}_0))$, i.e. $\mathcal{L}(\mathcal{A}_k) \supseteq \mathcal{R}^*(E)$. The approximation technique can be seen as a particular completion process between the rules in $\mathcal{R}$ and the transitions Δ_i of $\mathcal{A}_i$. In fact, in order to build $\mathcal{A}_{i+1}$ from $\mathcal{A}_i$, a critical pair is computed between a rewrite rule in $\mathcal{R}$ and the transitions in Δ_i. The rule derived from the critical pair is a new transition that is normalized using the approximation function γ and then added to Δ_i, thus yielding Δ_{i+1}.

The approximation technique, initially given for left linear rewrite systems [8], has then been extended to AC non-left linear rewrite systems in [9]. Given $\mathcal{T}_{\mathcal{R}}\!\uparrow\!(\mathcal{A})$, reachability properties on $\mathcal{R}$ and E can be proved by checking whether the intersection between $\mathcal{L}(\mathcal{T}_{\mathcal{R}}\!\uparrow\!(\mathcal{A}))$ and a set of terms, that must not be reached from E through $\mathcal{R}$, is empty. In the case of protocol verification, the negation of the property p to be proved is described through a tree automaton $\mathcal{A}_{\overline{p}}$, whose language contains those terms that must not be recognized by the intruder. If the intersection $\mathcal{L}(\mathcal{T}_{\mathcal{R}}\!\uparrow\!(\mathcal{A})) \cap \mathcal{L}(\mathcal{A}_{\overline{p}})$ is empty, then the property p is satisfied. On the contrary, whenever such intersection is not empty, we cannot say that there is an attack. Actually, this approach has been designed for proving that there are no attacks on a protocol, rather

than for detecting them. The fact is that the quality of the approximation automaton depends on the approximation function γ and, in spite of further studies on γ and its automation [24], whenever the intersection is not empty, it is not possible to distinguish whether there is an attack on the protocol or the function γ is too "large" and $\mathcal{T}_{\mathcal{R}}\uparrow(\mathcal{A})$ defines a language that also includes terms of $\mathcal{L}(\mathcal{A}_{\overline{p}})$. Hence, the approximation technique is not able to discover the attack on the insecure version of the NSPK, while it demonstrates the properties of authentication and secrecy on the corrected version [9,11].

4 The Rewriting Strategy

The implementation of the approximation technique we have been using for our experimentation is the one in Timbuk [10,11], a library of Objective Caml [15] for manipulating tree automata. Oehl has kindly provided us with his Timbuk files for both the insecure and the corrected versions of the NSPK, together with his approximation functions γ. We were primarily interested in running the approximation technique on the insecure version of the NSPK and understanding why the language intersection is not empty for both secrecy (nonces must be kept secret) and authentication of agents.

The rewrite system $\mathcal{R}$ specifying the insecure NSPK has been given in Section 2. The tree automaton $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$ describing the initial set E of communication requests is taken from [9], where $\mathcal{Q}_f = \{q_f\}$ and the transitions in Δ are as follows:

$0 \to q_{int}$		
$s(q_{int}) \to q_{int}$	$agt(q_{int}) \to q_{agtI}$	
$A \to q_A$	$agt(q_A) \to q_{agtA}$	
$B \to q_B$	$agt(q_B) \to q_{agtB}$	
$goal(q_{agtA}, q_{agtB}) \to q_f$	$goal(q_{agtA}, q_{agtA}) \to q_f$	*communication requests*
$goal(q_{agtB}, q_{agtA}) \to q_f$	$goal(q_{agtB}, q_{agtB}) \to q_f$	
$goal(q_{agtA}, q_{agtI}) \to q_f$	$goal(q_{agtI}, q_{agtA}) \to q_f$	
$goal(q_{agtB}, q_{agtI}) \to q_f$	$goal(q_{agtI}, q_{agtB}) \to q_f$	
$goal(q_{agtI}, q_{agtI}) \to q_f$	$\cup(q_f, q_f) \to q_f$	
$agt(q_{int}) \to q_f$	$pubkey(q_{agtI}) \to q_f$	*intruder's initial knowledge*
$agt(q_A) \to q_f$	$pubkey(q_{agtA}) \to q_f$	
$agt(q_B) \to q_f$	$pubkey(q_{agtB}) \to q_f$	
$mesg(q_f, q_f, q_f) \to q_f$	$N(q_{agtI}, q_{agtI}) \to q_f$	
$cons(q_f, q_f) \to q_f$	$N(q_{agtI}, q_{agtA}) \to q_f$	
$encr(q_f, q_{agtI}, q_f) \to q_f$	$N(q_{agtI}, q_{agtB}) \to q_f$	

Let t be a term that describes a property to be proved or disproved. We consider the problem of checking whether t can be recognized by the approximation automaton $\mathcal{T}_{\mathcal{R}}\uparrow(\mathcal{A})$. This means checking whether a transition $t \to q_f$

can be generated from critical pairs, where q_f is a final state of $\mathcal{A}$. We do not build $\mathcal{T}_{\mathcal{R}}\uparrow(\mathcal{A})$, but starting from t we simulate a completion process in a bottom-up manner guided by the critical pairs, thus reconstructing the rewriting path that has led to the intruder's knowledge of t, if any. If $t \rightarrow q_f$ can be generated during this process, the property represented by t is not satisfied. Moreover, by going up along the critical pairs we get to know which terms have been previously (in the completion process) recognized by the intruder, thus getting some feedback on the error location. This strategy is similar to a rewriting strategy defined in [12,13] to deal with the problem of divergence of the completion process, where the bottom-up strategy allows one to compute the normal form of a term with respect to the infinite canonical rewrite system.

The critical pairs between the rules in $\mathcal{R}$ and the transitions in Δ computed by the approximation technique are generated by the strategy in a bottom-up manner, by applying an expansion process on terms with respect to $\mathcal{R}$ and then checking the (instances of the) resulting terms for recognizability using only the intruder's initial knowledge Δ. Moreover, a notion of well-formedness of terms is used for ensuring the termination of the expansion process.

A term t is expanded with $\mathcal{R}$ if t unifies with the right hand side of a rule of $\mathcal{R}$: $\mathit{expansion}(t,\mathcal{R}) = \{s = \sigma(t[l]_p) \mid \exists\, l \rightarrow r \in \mathcal{R},\ p \in Pos'(t)$ and $\sigma = \mathrm{mgu}(t|_p, r)\}$. Thus, an expansion step is a narrowing step with a reversed rule of $\mathcal{R}$. The expansion process may introduce occurrences of "new" variables in s. These variables are considered as implicitly universally quantified and will be then instantiated by means of a finite set of ground terms $\mathit{Inst} = \{c_1, \ldots, c_k\}$, thus getting the *instance set* $\mathcal{I}(t, \mathit{Inst}) = \{\sigma(t) \mid \sigma : \mathit{Var}(t) \rightarrow \mathit{Inst}\}$. *Inst* is a parameter of the strategy and can be typically chosen as a (finite) set of agent labels needed for instantiating the agent variables in $\mathcal{R}$. For the NSPK it will be sufficient to take $\mathit{Inst} = \{A, B, 0\}$.

A term t is *recognizable* by the intruder if q_f can be derived from t using the transitions in Δ. As the strategy simulates a completion process that would produce the intruder's incremental knowledge, whenever t is not directly recognizable using Δ, the strategy is applied to those subterms of t that are not recognizable in Δ. Based on the following proof system $\vdash_{\mathcal{A}}$, we define a function $\overline{rec}(t) = \emptyset$ if $t \vdash_{\mathcal{A}} q_f$, otherwise $\overline{rec}(t) = \{t_i \mid t = \mathcal{C}[t_i]$ for some context $\mathcal{C}$ and $t_i \not\vdash_{\mathcal{A}} q_f\}$, thus yielding those subterms of t labelling the leaves of the proof tree of t in $\vdash_{\mathcal{A}}$ that remain unsolved.

$$\frac{t \xrightarrow{*}_{\Delta} q \quad q \in \{q_f, q_{agtI}\}}{t \vdash_{\mathcal{A}} q} \qquad \frac{t_1 \vdash_{\mathcal{A}} q_f \quad t_2 \vdash_{\mathcal{A}} q_f}{\cup(t_1,t_2) \vdash_{\mathcal{A}} q_f} \qquad \frac{t_1 \vdash_{\mathcal{A}} q_f \quad t_2 \vdash_{\mathcal{A}} q_f \quad t_3 \vdash_{\mathcal{A}} q_f}{\mathit{mesg}(t_1,t_2,t_3) \vdash_{\mathcal{A}} q_f}$$

$$\frac{t_1 \vdash_{\mathcal{A}} q_f \quad t_2 \vdash_{\mathcal{A}} q_f}{\mathit{cons}(t_1,t_2) \vdash_{\mathcal{A}} q_f} \qquad \frac{t_1 \vdash_{\mathcal{A}} q_{agtI} \quad t_2 \vdash_{\mathcal{A}} q_f}{N(t_1,t_2) \vdash_{\mathcal{A}} q_f} \qquad \frac{t_1 \vdash_{\mathcal{A}} q_f \quad t_2 \vdash_{\mathcal{A}} q_{agtI} \quad t_3 \vdash_{\mathcal{A}} q_f}{\mathit{encr}(t_1,t_2,t_3) \vdash_{\mathcal{A}} q_f}$$

The notion of *well-formedness* of terms is based on the following intuition. A term t is well-formed if it "agrees" with the syntactic structure of the rewrite

system that specifies the protocol. For example, given the system $\mathcal{R}$ for the insecure NSPK, $t_1 = N(agt(a_1), agt(a_2))$ is well-formed for any variables or agent labels a_1, a_2, while the term $t_2 = N(agt(a_1), pubkey(agt(a_2)))$ is not, as there is no term t in $\mathcal{R}$ such that $root(t) = N$ and the second argument of t starts with *pubkey*. One could also talk of well-sorted or well-typed terms, in the sense that when considering the function associated to, for example, the symbol N, this function is expected to take as input two terms of type agent, thus the type-checking of t_2 will fail.

The terms describing the authentication and secrecy properties are well-formed in the sense above. Moreover, during the expansion phase of the strategy, only well-formed terms will be considered and the non-well-formed ones will be cut out of the search space. This is not enough for ensuring the termination of the expansion process. In fact, rule (3) of $\mathcal{R}$ can be applied as a reversed rule infinitely many times to produce well-typed terms that contain subterms of the form $t = cons(x_1, cons(y_1, cons(z_1, \ldots)))$. Due to the structure of the contents of messages in the NSPK and of the terms in the language describing the properties to be checked, it is not needed to derive such terms by expansion as they cannot be the vertex of any interesting critical peak. Hence, expansion by rule (3) is stopped by defining that terms containing instances of t above are not well-formed. The predicate of well-formedness is a parameter for the strategy: whenever other protocols and/or different properties are considered, the definition of the well-formedness of terms might have to be changed accordingly. For the insecure NSPK we give the following definition.

A term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is *well-formed*, written $wf(t)$, if (i) $t \in \mathcal{X} \cup \mathcal{F}^0$ or (ii) $t = f(t_1, \ldots, t_n)$ with $f \in \mathcal{F}^n$ $(n > 0)$ and either $t_i \in \mathcal{X}$ or t_i satisfies the following conditions based on the value of f $(i = 1, \ldots, n)$:

- $f = agt$ and $t_1 \in L_{agt}$;
- $f = goal$, $root(t_i) = agt$ and $wf(t_i)$ for $i = 1, 2$;
- $f = mesg$, $root(t_i) = agt$ and $wf(t_i)$ for $i = 1, 2$, $root(t_3) = encr$ and $wf(t_3)$;
- $f = encr$, $root(t_1) = pubkey$, $root(t_2) = agt$, $root(t_3) \in \{cons, N, agt\}$ and $wf(t_i)$ for $i = 1, 2, 3$;
- $f = pubkey$, $root(t_1) = agt$ and $wf(t_1)$;
- $f = cons$, $root(t_i) \in \{N, agt\}$ and $wf(t_i)$ for $i = 1, 2$;
- $f = N$, $root(t_i) = agt$ and $wf(t_i)$ for $i = 1, 2$;
- $f \in \{c\_init, c\_resp\}$, $root(t_i) = agt$ and $wf(t_i)$ for $i = 1, 2, 3$;
- $f = \cup$ and $wf(t_i)$ for $i = 1, 2$.

The input to the strategy is given by the rewrite system specifying the protocol $\mathcal{R} = \mathcal{R}_P \cup \mathcal{R}_I$, the predicate *wf*, the instantiation set *Inst*, the intruder's initial knowledge Δ in $\mathcal{A}$, and the well-formed term t_{in} describing the property under consideration. The strategy is defined through the set of in-

ference rules given in Figure 1. An inference rule is a binary relation between *configurations*, which are either (finite) sets of well-formed terms or elements of the set $\{success, failure\}$. Thus, the inference rules either map a set of well-formed terms into another such set ($E \vdash E'$) or terminate the derivation process ($E \vdash success$ or $E \vdash failure$). The initial configuration is $E_0 = \{t_{in}\}$. The predicate $subterm(t, t')$ is true if t' is a subterm of t.

Well-formed Expansion:

$$\frac{t \in E \quad expansion(t, \mathcal{R}) = E'}{E \setminus \{t\} \cup \{t' \in E' \mid wf(t')\}}$$

Failure:

$$\frac{E = \emptyset}{failure}$$

Success1:

$$\frac{t \in E \quad \exists t'.subterm(t, t') \ \wedge \ root(t') = goal}{success}$$

Cut:

$$\frac{\begin{array}{c} t \in E \quad expansion(t, \mathcal{R}_P) = \emptyset \quad subterm(t, t_{in}) \\ \exists t'.subterm(t, t') \ \wedge \ root(t') = mesg \end{array}}{E \setminus \{t\}}$$

Success2:

$$\frac{\begin{array}{c} t \in E \quad expansion(t, \mathcal{R}_P) = \emptyset \quad not(subterm(t, t_{in})) \\ \exists t'.subterm(t, t') \ \wedge \ root(t') = mesg \\ \mathcal{I}(t, Inst) = E_1 \quad \exists t_1 \in E_1.\, \overline{rec}(t_1) = \emptyset \end{array}}{success}$$

Split:

$$\frac{\begin{array}{c} t \in E \quad expansion(t, \mathcal{R}_P) = \emptyset \quad not(subterm(t, t_{in})) \\ \exists t'.subterm(t, t') \ \wedge \ root(t') = mesg \\ \mathcal{I}(t, Inst) = \{t_1, \ldots, t_k\} \quad \forall i.\overline{rec}(t_i) \neq \emptyset \end{array}}{E \setminus \{t\} \cup \overline{rec}(t_1) \cup \ldots \cup \overline{rec}(t_k)}$$

Fig. 1. The inference rules of the strategy.

The successful termination of the strategy means that an attack has been detected, thus the property represented by t_{in} is not satisfied. Termination with failure means that the strategy has failed in finding an attack, thus the property represented by t_{in} is satisfied. The rule Well-formed Expansion replaces a term in E with its well-formed expansions. Whenever there are no terms left in E, the strategy fails without finding an attack (rule Failure). If there exists a subterm of t the root of which is *goal*, then the strategy terminates with success, because every communication request is in the intruder's basic knowledge (rule Success1). The remaining inference rules work under the condition that a term $t \in E$ is selected that cannot be further expanded with the rules in $\mathcal{R}_P$. If there exists a subterm of t the root of which is $mesg$, we distinguish on whether the input term t_{in} occurs as a subterm of t. If t_{in}

occurs in t, then t is deleted from E (rule Cut) as expanding t will loop without adding information on the intruder's knowledge. Otherwise, the instances of t are checked for recognizability. It is sufficient to have a recognizable instance of t to terminate with success (rule Success2). Whenever all instances t_i of t are not recognizable, rule Split replaces t in E with those subterms of all t_i that are not in the intruder's basic knowledge, thus looking for possible further critical peaks in the bottom-up search.

The (non-deterministic) rewriting strategy for protocol verification is then defined as a regular expression over the names of the inference rules:

$$((\textit{Well-formed Expansion} + \textit{Cut})^*.(\textit{Failure} + \textit{Success1} + \textit{Success2} + \textit{Split}))^*$$

where r^* means iteration of the inference rule r, $r.r'$ means sequencing of r and r', and $r + r'$ means non-deterministic choice between r and r'. Thus, the rewriting strategy applies well-formed expansions of terms and prunes the derivation paths (whenever possible) in a non-deterministic way, and then checks if either conditions for failure/success are satisfied or the inference steps must be iterated from the terms added to E by rule Split.

Given the rewrite system $\mathcal{R}$, the predicate *wf*, the instantiation set *Inst*, the tree automaton $\mathcal{A}$ with transitions Δ and the negation automaton $\mathcal{A}_{\overline{p}}$ for the property to be checked, the correctness, termination and completeness of the strategy are formalized as follows.

Proposition 4.1 *(correctness) Let* $t_{in} \in \mathcal{L}(\mathcal{A}_{\overline{p}})$.
(i) If $\{t_{in}\} \vdash$ success*, then* $t_{in} \to q_f$ *can be generated from critical pairs.*
(ii) If $\{t_{in}\} \vdash$ failure*, then the transition* $t_{in} \to q_f$ *cannot be generated from critical pairs.*

Proof. (Sketch)
(i) Let $\{t_{in}\} \vdash \textit{success}$ be derived by means of a derivation $\{t_{in}\} \vdash E_1 \vdash \ldots \vdash E_n \vdash \textit{success}$. If *success* is obtained by applying the inference rule Success1, this means that t_{in} has been expanded using $\mathcal{R}_I$ and $\mathcal{R}_P$ until a term $t'[t'']_p \in E_n$ has been obtained such that $\textit{root}(t'') = \textit{goal}$. As all communication requests $\textit{goal}(x, y)$ are in the intruder's initial knowledge, the critical peak $t_{in} \overset{*}{\leftarrow}_{\mathcal{R}} t' \overset{*}{\rightarrow}_{\Delta} q_f$ produces the transition $t_{in} \to q_f$.
If *success* is obtained by applying the inference rule Success2, then the situation is as follows. The term t_{in} has been expanded using $\mathcal{R}_I$ and $\mathcal{R}_P$ until a term $t' \in E_k$ for some $k \leq n$ is derived such that either Success2 or Split can be applied. The instance set $\mathcal{I}(t', \textit{Inst})$ is thus computed using *Inst*. If rule Success2 can be applied, this means that there exists an instance t'' of t' which is a vertex of a critical peak as t'' can be reduced to t_{in} in $\mathcal{R}$ and is recognizable by the intruder, i.e. $t'' \overset{*}{\rightarrow}_{\Delta} q_f$. Hence, the transition $t_{in} \to q_f$ is generated. The application of rule Split means that all instances of t' cannot be recognized based on the intruder's initial knowledge, thus further critical pairs must be searched in a bottom-up manner by iterating the application of the strategy.

(ii) Let $\{t_{in}\} \vdash \mathit{failure}$ be derived by means of a derivation $\{t_{in}\} \vdash E_1 \vdash \ldots \vdash E_n \vdash \mathit{failure}$, where $E_n = \emptyset$ (rule Failure). Terms have been expanded and possibly instantiated using *Inst* and split, until terms have been derived that are not well-formed (no terms are added to the expansion set E and the expanded terms are removed from E by the rule Well-formed Expansion) or contain the input term t_{in} (they can be deleted using the rule Cut without losing information about the intruder's knowledge). Any such term cannot be the vertex of a critical peak, thus the transition $t_{in} \rightarrow q_f$ cannot be generated from critical pairs. □

Proposition 4.2 *(termination) The rewriting strategy terminates on any input term* $t_{in} \in \mathcal{L}(\mathcal{A}_{\overline{p}})$.

Proof. (Sketch) The set E only contains well-formed terms. Given any well-formed $t_{in} \in \mathcal{L}(\mathcal{A}_{\overline{p}})$, the well-formed expansion process terminates because the repeated application of the rules of $\mathcal{R}$ as expansion rules will eventually produce only terms that do not satisfy the well-formed predicate. In fact, well-formedness ensures that there cannot be infinite expansions by $\mathcal{R}_I$, while infinite expansions by $\mathcal{R}_P$ cannot occur because terms will eventually not unify any more with any of the right hand sides of $\mathcal{R}_P$. Thus, the expansion process along the several different derivation paths terminates because the terms on each path are all well-formed and each path ends with a term that cannot be further expanded in a well-formed way. If the strategy does not derive *success*, then the set E decreases till it becomes empty (rule Failure), because the rule Cut is repeatedly applied along the different paths or the expanded terms are removed from E and not replaced by other well-formed terms (rule Well-formed Expansion). □

Corollary 4.3 *(completeness) Let* $t_{in} \in \mathcal{L}(\mathcal{A}_{\overline{p}})$.
(i) If the transition $t_{in} \rightarrow q_f$ *can be generated from critical pairs, then* $\{t_{in}\} \vdash$ success.
(ii) If the transition $t_{in} \rightarrow q_f$ *cannot be generated from critical pairs, then* $\{t_{in}\} \vdash$ failure.

Proof. It follows from the termination (Proposition 4.2) and the correctness (Proposition 4.1) of the strategy. □

5 A Verification Example

Let us consider the property of secrecy for the NSPK. Given the system $\mathcal{R}$ (Section 2), let $t = N(agt(B), agt(A)) \in \mathcal{L}(\mathcal{A}_{\overline{s}})$, where $\mathcal{L}(\mathcal{A}_{\overline{s}})$ denotes the negation automaton for the property of secrecy of nonces. [4] We want to prove whether the nonce t is recognizable by the intruder, thus attacking the se-

[4] Due to the definition of tree automaton, t is actually the term $N(q_{agtB}, q_{agtA})$ that gets expanded into $N(agt(B), agt(A))$. In the example we will abstract from these details.

crecy of the NSPK. By abstracting from some details, we get the following derivation steps:

$\{N(agt(B), agt(A))\}$

$\vdash \{\cup(cons(N(agt(B), agt(A)), y_1), z_1), \cup(cons(x_1, N(agt(B), agt(A))), z_1), encr(pubkey(agt(0)), y_1,$

$N(agt(B), agt(A))), encr(pubkey(agt(s(x))), y_1, N(agt(B), agt(A))), mesg(x_1, y_1, N(agt(B), agt(A)))\}.$

By expanding the third term in the current configuration, we obtain:

$encr(pubkey(agt(0)), y_1, N(agt(B), agt(A)))$

$\vdash mesg(x_2, y_2, encr(pubkey(agt(0)), y_1, N(agt(B), agt(A))))$

$\vdash mesg(agt(a_6), agt(a), encr(pubkey(agt(a)), agt(a_5), cons(N(agt(a), agt(0)), N(agt(B), agt(A)))))$

$\vdash mesg(agt(a_4), agt(B), encr(pubkey(agt(B)), agt(a_3), cons(N(agt(A), agt(0)), agt(A)))).$

This term cannot be further expanded and does not have the initial term as subterm. Thus, its variables are instantiated through *Inst*. Among the various (finite) possible instances, let us consider the substitution $\sigma = \{A/a_4, 0/a_3\}$, thus yielding the term

$$mesg(agt(A), agt(B), encr(pubkey(agt(B)), agt(0), cons(N(agt(A), agt(0)), agt(A)))).$$

By computing the function $\overline{rec}$ on t, we have that $N(agt(A), agt(0))$ is a non-recognizable subterm of t, i.e. $N(agt(A), agt(0))$ does not belong to the intruder's basic knowledge. Thus, the rule Split adds $N(agt(A), agt(0))$ to the expansion set E and the following derivation is obtained:

$N(agt(A), agt(0)) \vdash encr(pubkey(agt(0)), y_1', N(agt(A), agt(0))))$

$\vdash mesg(x_2', y_2', encr(pubkey(agt(0)), y_1', N(agt(A), agt(0))))$

$\vdash mesg(agt(a_6'), agt(a'), encr(pubkey(agt(a')), agt(a_5'), cons(N(agt(A), agt(0)), N(agt(A), agt(0)))))$

$\vdash mesg(agt(a_4'), agt(A), encr(pubkey(agt(A)), agt(a_3'), cons(N(agt(0), agt(0)), agt(0)))).$

The last term cannot be further expanded. Using the function $\overline{rec}$ on an instance of such a term (for example, $\sigma' = \{0/a_4', 0/a_3'\}$), q_f is derived and the strategy terminates with *success*. Hence, we have $N(agt(A), agt(0)) \to q_f$ and the transition $N(agt(B), agt(A)) \to q_f$ can also be generated from critical pairs. Thus, t is recognizable by the intruder and the property of secrecy of nonces is not true for the NSPK. It is worth noting that the transition $N(agt(B), agt(A)) \to q_f$ derives from a critical pair involving the transition $N(agt(A), agt(0)) \to q_f$ which is, in turn, generated by a critical pair with the basic transition $N(agt(0), agt(0)) \to q_f$.

If we take as $\mathcal{R}$ the rewrite system for the corrected version of the NSPK and run the strategy on the term $t = N(agt(B), agt(A))$, the strategy terminates with *failure*. In fact, the expansion set E becomes empty without deriving any transition $t \to q_f$. In order to prove that the secrecy of nonces is true for the protocol, the strategy must be applied to all terms in the (finite) language of $\mathcal{A}_{\overline{s}}$.

In a similar way it can be proved that the property of authentication is true for the corrected version of the NSPK, while it fails on the insecure version. In fact, given $t = c\_init(agt(B), agt(A), agt(0))$, the strategy derives *success*. Note that, when verifying the authentication and secrecy properties for the corrected version of the NSPK, the predicate *wf* needs to be slightly modified

in order to allow more expansion steps and preserve the completeness of the strategy.

6 Concluding Remarks and Further Research

This paper has presented an approach to the analysis and the verification of protocol specifications based on a bottom-up rewriting strategy. Our experimentation with the authentication and secrecy properties on both the insecure and the corrected versions of the NSPK has shown that the strategy is able to detect the attacks in the insecure version, and derive that the properties hold for the corrected version of the protocol. The correctness, termination and completeness of the strategy depend on a notion of well-formedness on terms, which allows one to reduce the search space of the strategy. More study is needed on the relationship between the rules describing a protocol and the languages characterizing the properties under consideration. We need to better characterize well-formedness, extend it to other protocols and provide more general criteria for ensuring the correctness and the termination of the strategy.

With respect to the approach based on the approximation technique, the strategy carries out a bottom-up completion driven by the term representing the property to be proved, without generating the whole approximation automaton. Moreover, the strategy is able to say whether a property is satisfied or not, as it does not depend on any approximation function, and feedback on error location can be obtained in case of attacks by going back along the critical pairs in the bottom-up search.

The approximation technique provides an automatic manner for checking whether there are no attacks on a protocol specification, but has limitations on the set of properties that can be checked and is not able to assert that a property is not satisfied by a protocol due to the approximation function. On the contrary, using induction in Isabelle, Paulson has proved (or disproved) many properties of several kinds of security protocols using both a certain degree of automation and some interaction with the system. The knowledge and experience that a user needs to have in order to carry out some verification tasks on protocol specifications is much more relevant when using the inductive approach in Isabelle, than using the approximation technique in Timbuk. This evidence is the starting point for the combination of the two approaches presented in [22,23]. This simply consists in first applying the approximation technique to prove a property. If the property is satisfied, then this result is used as an axiom in Paulson's inductive method. Whenever the intersection between the languages of the tree automata is not empty, then the logical theories developed in Isabelle are used to prove or disprove the property. To make easier the link between the two systems, a translation is given from the Isabelle files to the input files for Timbuk. With respect to these two verification methods, the strategy illustrated in this paper can be seen as a

compromise between the full efficiency of the approximation technique and the full expressive power of Paulson's approach based on theorem proving.

The authentication and secrecy properties of the NSPK have already been extensively studied in the literature. However, by applying our approach to the NSPK we have considered what is called in [27] (page 101) "a major test bed, sometimes even a rite of passage, for every new protocol verification tool". We are currently working on applying our strategy to other protocols, e.g. the Woo-Lam and Otway-Rees protocols, that can be formalized in a way similar to the NSPK. Future work includes the application of the approach to other classes of protocols and the extension of the strategy to checking other properties of interest. This will mean to enrich the signature and the rules of the protocol specification. Furthermore, we are interested in implementing the strategy by defining it in a more expressive strategy language, e.g. the one of the systems ELAN [3] and Maude [4].

Acknowledgements

We would like to thank Thomas Genet for kindly and promptly replying to all our questions and doubts on the approximation technique, Frédéric Oehl for sharing his Timbuk files with us, and the anonymous referees for many helpful comments on the preliminary version of the paper.

References

[1] F. Baader and T. Nipkow, *Term Rewriting and All That*, (Cambridge University Press, Cambridge, 1998).

[2] H. Cirstea, Specifying Authentication Protocols Using Rewriting and Strategies, in: *Proceedings PADL 2001*, Lecture Notes in Computer Science, Vol. 1990, Springer-Verlag, 138–152.

[3] H. Cirstea, C. Kirchner, L. Liquori and B. Wack, Rewrite Strategies in the Rewriting Calculus, in: *Proceedings WRS 2003*, (Valencia, Spain, June 2003).

[4] M. Clavel, Strategies and User Interfaces in Maude (Extended Abstract), in: *Proceedings WRS 2003*, (Valencia, Spain, June 2003).

[5] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison and M. Tommasi, *Tree Automata Techniques and Applications*, `http://www.grappa.univ-lille3.fr/tata/`.

[6] G. Denker, J. Meseguer and C. Talcott, Protocol Specification and Analysis in Maude, in: *Proceedings 2nd WRLA Workshop*, (Pont-à-Mousson, France, 1998).

[7] N. Dershowitz and J.-P. Jouannaud, Rewrite Systems, in: J. van Leeuwen ed., *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, (North-Holland, Amsterdam, 1990), 243-320.

[8] T. Genet, Decidable Approximations of Sets of Descendants and Sets of Normal Forms, in: *Proceedings RTA 1998*, Lecture Notes in Computer Science, Vol. 1379, Springer-Verlag, 151–165.

[9] T. Genet and F. Klay, Rewriting for Cryptographic Protocol Verification, in: *Proceedings CADE 2000*, Lecture Notes in Artificial Intelligence, Vol. 1831, Springer-Verlag, 271–290.

[10] T. Genet and V. Viet Triem Tong, Timbuk - A tree automata library, `http://www.irisa.fr/lande/genet/timbuk`.

[11] T. Genet and V. Viet Triem Tong, Reachability Analysis of Term Rewriting Systems with Timbuk, in: *Proceedings LPAR 2001*, Lecture Notes in Artificial Intelligence, Vol. 2250, Springer-Verlag, 695–706.

[12] P. Inverardi and M. Nesi, A Strategy to Deal with Divergent Rewrite Systems, in: *Proceedings CTRS 1992*, Lecture Notes in Computer Science, Vol. 656, Springer-Verlag, 458–467.

[13] P. Inverardi and M. Nesi, Semi-equational Rewriting for Divergent Rewrite Systems, Technical Report No. 113, Department of Pure and Applied Mathematics, University of L'Aquila, July 1996.

[14] F. Jacquemard, M. Rusinowitch and L. Vigneron, Compiling and Verifying Security Protocols, in: *Proceedings LPAR 2000*, Lecture Notes in Artificial Intelligence, Vol. 1955, Springer-Verlag, 131–160.

[15] X. Leroy, D. Doligez, J. Garrigue, D. Rémy and J. Vouillon, The Objective Caml system, `http://caml.inria.fr/ocaml/`.

[16] G. Lowe, An Attack on the Needham-Schroeder Public-Key Authentication Protocol, *Information Processing Letters* **56** (1995) 131–133.

[17] G. Lowe, Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR, in: *Proceedings TACAS 1996*, Lecture Notes in Computer Science, Vol. 1055, Springer-Verlag, 147-166.

[18] C. A. Meadows, Analyzing the Needham-Schroeder Public-Key Protocol: A Comparison of Two Approaches, in: *Proceedings ESORICS 1996*, Lecture Notes in Computer Science, Vol. 1146, Springer-Verlag, 351-364.

[19] J. C. Mitchell, M. Mitchell and U. Stern, Automated analysis of cryptographic protocols using Murphi, in: *Proceedings IEEE Symposium on Security and Privacy 1997*, 141–153.

[20] D. Monniaux, Abstracting Cryptographic Protocols with Tree Automata, in: *Proceedings SAS 1999*, Lecture Notes in Computer Science, Vol. 1694, Springer-Verlag, 149–163.

[21] R. M. Needham and M. D. Schroeder, Using Encryption for Authentication in Large Networks of Computers, *Communications of the ACM* **21(12)** (1978) 993–999.

[22] F. Oehl and D. Sinclair, Combining two approaches for the verification of cryptographic protocols, in: *Proceedings Workshop on Specification, Analysis and Validation for Emerging Technologies in Computational Logic (SAVE 2001)*, (Paphos, Cyprus, December 2001).

[23] F. Oehl and D. Sinclair, Combining Isabelle and Timbuk for Cryptographic Protocol Verification, in: *Proceedings Workshop on Securité de la Communication sur Internet (SECI 2002)*, (Tunis, September 2002), 57–67.

[24] F. Oehl, G. Cécé, O. Kouchnarenko and D. Sinclair, Automatic Approximation for the Verification of Cryptographic Protocols, INRIA Technical Report RR-4599 (October 2002).

[25] L. Paulson, Proving Properties of Security Protocols by Induction, in: *Proceedings 10th Computer Security Foundations Workshop*, (IEEE Computer Society Press, 1997), 70–83.

[26] L. C. Paulson, The Inductive Approach to Verifying Cryptographic Protocols, *Journal on Computer Security* **6** (1998) 85–128.

[27] P. Syverson and I. Cervesato, The Logic of Authentication Protocols, in: *Proceedings Foundations of Security Analysis and Design 2000*, Lecture Notes in Computer Science, Vol. 2171, Springer-Verlag, 63–136.

[28] C. Weidenbach, Towards an Automatic Analysis of Security Protocols in First-Order Logic, in: *Proceedings CADE 1999*, Lecture Notes in Artificial Intelligence, Vol. 1632, Springer-Verlag, 314–328.