Available online at www.sciencedirect.com
SCIENCE DIRECT
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 135 (2005) 95–114
www.elsevier.com/locate/entcs

Formalizing and Analyzing the Needham-Schroeder Symmetric-Key Protocol by Rewriting

Monica Nesi[1,2] and Giuseppina Rucci

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

Abstract

This paper reports on work in progress on using rewriting techniques for the specification and the verification of communication protocols. As in Genet and Klay's approach to formalizing protocols, a rewrite system $\mathcal{R}$ describes the steps of the protocol and an intruder's ability of decomposing and decrypting messages, and a tree automaton $\mathcal{A}$ encodes the initial set of communication requests and an intruder's initial knowledge. In a previous work we have defined a rewriting strategy that, given a term t that represents a property of the protocol to be proved, 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. In this paper we present a formalization of the Needham-Schroeder symmetric-key protocol and use the rewriting strategy for deriving two well-known authentication attacks.

Keywords: Protocol verification, term rewriting, tree automata, rewriting strategy.

1 Introduction

In the past few years several approaches have been applied to protocol specifications in order to formally verify various properties of interest, such as authentication, secrecy or confidentiality, freshness, etc. These approaches range from model checking [24,27,4] to theorem proving [26,35,36,37,22] through process calculi [1,8,9], Horn clauses [6], multiset rewriting and strand spaces [5,10],

[1] Work partially supported by MIUR ex-40% project "Infrastrutture e architetture software per applicazioni globali, adattabili e attendibili (GHIBLI)".
[2] Email: monica@di.univaq.it

1571-0661 © 2005 Elsevier B.V. Open access under CC BY-NC-ND license.
doi:10.1016/j.entcs.2005.06.002

rewriting techniques and strategies [11,14,23] using tree automata and abstract interpretation [18,19,28]. Most of these verification approaches have also been implemented using either specific-purpose tools, such as AVISPA [2], CASRUL [23], NRL [26] and Timbuk [19], or general-purpose tools, such as ELAN [11,18], FDR [24], Isabelle [35,36], Maude [14] and SPASS [37]. There has also been some work on comparing and combining different approaches, e.g. the combination of Genet and Klay's approximation technique with Paulson's inductive method [33,34].

We are interested in the use of rewriting based techniques for the formalization and the verification of communication protocols. Rewrite systems provide a very natural approach to operationally describe the behaviour of a protocol. In particular, rewrite systems and tree automata are used in [17,18,19] to specify and verify properties of security protocols by developing an approximation technique that aims at finding that there are no attacks on a protocol, rather than at discovering attacks. The protocol is specified through a rewrite system $\mathcal{R}$, while the initial set E of communication requests and an intruder's initial knowledge are 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. The quality of the approximation depends on an approximation function γ which defines the subterms that can be approximated. The approximation technique can be seen as a particular completion process between $\mathcal{R}$ and $\mathcal{A}$, as critical pairs are computed between the rules in $\mathcal{R}$ and the transitions in $\mathcal{A}$. The rules derived from the critical pairs are new transitions that are normalized using γ and then added to $\mathcal{A}$. Thus, 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.

In developing our approach to verifying security protocols, we have been borrowing Genet and Klay's formalization of a protocol, i.e. a rewrite system $\mathcal{R}$ and a tree automaton $\mathcal{A}$. Then, given a term t that describes a property to be proved, we apply a rewriting strategy (defined in [31]) 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 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. 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. Thus, while the formalization of the protocol is similar to the one given by Genet and Klay, our verification approach is different from their approximation technique, specially no approximation is carried out on the transitions when applying the bottom-up derivation process. This results in our strategy being able to derive whether a property is satisfied or not.

In [31] the strategy has been applied to the typical first case study of protocol verification, i.e. the Needham-Schroeder Public-Key protocol [29] (NSPK for short), for reasoning about the authentication and secrecy properties. Our experimentation 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 NSPK.

The strategy has then been tested on a classic example of symmetric-key authentication protocols, i.e. the Needham-Schroeder Symmetric-Key protocol [29] (NSSK from now on). In this respect, besides the authentication and secrecy properties, it is important to be able to specify and verify the freshness of shared (symmetric) keys. This paper presents a formalization of the NSSK, based on Genet and Klay's approach, which is a revised and enriched version of their rewrite system and tree automaton. In particular, session identifiers are added in the relevant messages of the protocol, in order to be able to express and reason on the freshness property. The verification strategy, unchanged with respect to its definition in [31], is applied to the proposed formalization of the NSSK and two authentication attacks are derived.

The paper is organized as follows. Some basic definitions about term rewriting and tree automata are recalled in Section 2. Section 3 briefly describes the NSSK and Denning and Sacco's attack. Next, our formalization of the NSSK and the ingredients of the strategy are presented in Section 4. The rewriting strategy is then illustrated in Section 5 and finally applied to the specification of the NSSK for deriving the attacks in Section 6. The paper ends with some concluding remarks and directions for future work.

2 Basic Definitions

Below we summarize the most relevant definitions of term rewriting and tree automata, and refer to [3,16,13] for more details.

A signature is a set of function symbols $\mathcal{F} = \bigcup_n \mathcal{F}^n$ where $\mathcal{F}^n$ is the set of

symbols of arity n. Let $\mathcal{T}(\mathcal{F},\mathcal{V})$ be the set of finite and first-order terms with function symbols $\mathcal{F}$ and variables $\mathcal{V}$. The root symbol of a term $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$ is $root(t) = t$ if $t \in \mathcal{V} \cup \mathcal{F}^0$ and $root(t) = f$ if $t = f(t_1,\ldots,t_n)$ for some $f \in \mathcal{F}^n$ and $t_1,\ldots,t_n \in \mathcal{T}(\mathcal{F},\mathcal{V})$. A *context* $\mathcal{C}$ is a term in $\mathcal{T}(\mathcal{F} \cup \{\Box\},\mathcal{V})$, where $\Box \notin \mathcal{F}$ is a new symbol representing "holes". Given a context $\mathcal{C}$ with n occurrences of $\Box$ and $t_1,\ldots,t_n \in \mathcal{T}(\mathcal{F},\mathcal{V})$, the term $\mathcal{C}[t_1,\ldots,t_n]$ is obtained by replacing, from left to right, the n occurrences of $\Box$ with $t_1,\ldots,t_n$.

A *rewrite system* or *term rewriting system* (TRS) $\mathcal{R}$ is any set $\{(l_i,r_i) \mid l_i, r_i \in \mathcal{T}(\mathcal{F},\mathcal{V}), l_i \notin \mathcal{V},\ Var(r_i) \subseteq Var(l_i)\}$. The pairs (l_i,r_i) are called *rewrite rules* and written $l_i \to r_i$. The *rewriting relation* $\to_{\mathcal{R}}$ over $\mathcal{T}(\mathcal{F},\mathcal{V})$ is defined as the least relation containing $\mathcal{R}$ that is closed under context application and substitution. A term t *rewrites* to a term t', written $t \to_{\mathcal{R}} t'$ (or simply $t \to t'$), if there exists a rule $l \to r$ in $\mathcal{R}$, a substitution σ and a subterm $t|_p$ (called *redex*) at the position p, such that $t|_p = \sigma l$ and $t' = t[\sigma r]_p$. A term t is said to *overlap* a term t' if t unifies with a non-variable subterm of t' (after renaming the variables in t so as not to conflict with those in t'). Let $Pos'(t)$ be the set of positions of non-variable subterms of t. If $l_i \to r_i$ and $l_j \to r_j$ are two rewrite rules (with distinct variables), $p \in Pos'(l_i)$ and $\sigma = mgu(l_i|_p, l_j)$, then the equation $(\sigma l_i)[\sigma r_j]_p = \sigma r_i$ is a *critical pair* formed from those rules.

Let $\xrightarrow{+}$ and $\xrightarrow{*}$ denote the transitive and transitive-reflexive closure of $\to$, respectively. A TRS $\mathcal{R}$ is *terminating* if there is no infinite derivation $t_1 \to t_2 \to \ldots$ in $\mathcal{R}$. A term t is *in $\mathcal{R}$-normal form* if there is no term s such that $t \to s$. A term s is an *$\mathcal{R}$-normal form* of t if $t \xrightarrow{*} s$ and s is in $\mathcal{R}$-normal form. A TRS $\mathcal{R}$ is *confluent* if whenever $s \xleftarrow{*} t \xrightarrow{*} u$, there exists a term t' such that $s \xrightarrow{*} t' \xleftarrow{*} u$. A TRS $\mathcal{R}$ is *canonical* (or *convergent*) if it is terminating and confluent.

A term t *reduces via narrowing* to a term t', written $t \leadsto t'$, if there exists a rule $l \to r$ in $\mathcal{R}$, a position $p \in Pos'(t)$ and a substitution $\sigma = mgu(t|_p, l)$, and $t' = \sigma(t[r]_p)$.

Let $\mathcal{Q}$ be a finite set of symbols, with arity 0, called *states*. A *transition* is a rewrite rule $c \to q$, where $c \in \mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ and $q \in \mathcal{Q}$. A *normalized transition* is a transition $c \to q$ where $c = q' \in \mathcal{Q}$ or $c = f(q_1,\ldots,q_n)$, $f \in \mathcal{F}^n$ and $q_1,\ldots,q_n \in \mathcal{Q}$. A bottom-up non-deterministic finite tree automaton is a quadruple $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$, where $\mathcal{Q}_f \subseteq \mathcal{Q}$ and Δ is a set of normalized transitions. A tree automaton is *deterministic* if there are no two rules with the same left hand side. The rewrite relation induced by Δ is denoted by either $\to_\Delta$ or $\to_{\mathcal{A}}$. The tree language recognized by $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = \{t \in \mathcal{T}(\mathcal{F}) \mid \exists q \in \mathcal{Q}_f : t \xrightarrow{*}_{\mathcal{A}} q\}$. A tree language (or a set of terms) E is *regular* if there exists a bottom-up tree automaton $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = E$.

3 The NSSK Protocol

The NSSK [29] is a well-known example of symmetric-key communication protocol that aims at the mutual authentication of two agents communicating through an insecure network. In the NSSK there are typically three principals: agents A and B and a server S acting as a key distribution centre. The NSSK protocol is described in Figure 1.

$$
\begin{aligned}
&1.\ A \longrightarrow S \ :\ A, B, N_A \\
&2.\ S \longrightarrow A \ :\ \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}} \\
&3.\ A \longrightarrow B \ :\ \{K_{AB}, A\}_{K_{BS}} \\
&4.\ B \longrightarrow A \ :\ \{N_B\}_{K_{AB}} \\
&5.\ A \longrightarrow B \ :\ \{N_B - 1\}_{K_{AB}}
\end{aligned}
$$

Fig. 1. The Needham-Schroeder Symmetric-Key protocol.

The protocol must guarantee the secrecy and the freshness of K_{AB} as follows: at the end of a protocol session, K_{AB} should be known only to A, B and S, and A (resp. B) should believe that B (resp. A) has the same key K_{AB} created by S in the current session of the protocol. If B accepts the message at step (5), then K_{AB} has been sent by A in step (3) (authentication).

The NSSK has been found insecure by Denning and Sacco [15] who have shown an authentication attack involving two sessions (i) and (ii) of the protocol, one before the other. Assume that an intruder has recorded the session (i) and the key K'_{AB}, created in session (i), has been compromised and is known to the intruder. The session (ii) can develop as depicted in Figure 2.

$$
\begin{aligned}
&ii.1.\ A \longrightarrow S \ :\ A, B, N_A \\
&ii.2.\ S \longrightarrow A \ :\ \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}} \\
&ii.3.\ I(A) \longrightarrow B \ :\ \{K'_{AB}, A\}_{K_{BS}} \\
&ii.4.\ B \longrightarrow I(A) \ :\ \{N_B\}_{K'_{AB}} \\
&ii.5.\ I(A) \longrightarrow B \ :\ \{N_B - 1\}_{K'_{AB}}
\end{aligned}
$$

Fig. 2. Denning and Sacco's authentication attack.

At the end of session (ii), B thinks that (s)he has established a communication with A and is sharing a secret fresh key with A, while (s)he has been communicating with I (authentication attack) and has received an old compromised session key (freshness and secrecy attacks). The corrected version

of the NSSK proposed by Denning and Sacco [15] is based on adding timestamps to messages in steps (2)÷(3) and removing all nonces. Needham and Schroeder [30] have proposed an amended version of the NSSK, where they introduce a handshake between A and B at the beginning of the protocol. Lowe [25] has later modified Denning and Sacco's version to prevent multiplicity attacks by adding a nonce handshake between B and A at the end of the protocol.

4 The Formalization of the NSSK

As in the approach developed by Genet and Klay [17,18,19] and also used by Oehl [7,32], a protocol is formalized through a rewrite system $\mathcal{R} = \mathcal{R}_P \cup \mathcal{R}_I$, where $\mathcal{R}_P$ describes the steps of the protocol and the properties to be verified, and $\mathcal{R}_I$ defines an intruder's ability of decomposing and decrypting messages. This section introduces the ingredients of our rewriting strategy for protocol verification.

4.1 The Protocol

The signature $\mathcal{F}$ of $\mathcal{R}$ is defined as follows. Let L_{agt} be an infinite set of agent labels. In the NSSK we are interested in the behaviour of three principals, i.e. two agents A and B and a server S. 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 $l \in L_{agt}$. The server S is a distinguished principal built using the constructor $serv$. [3] Natural numbers are also used for denoting the protocol session or run: $r(i)$ represents the run i, $i \in \mathbb{N}$. $mesg(x, y, c, w)$ denotes a message from principal x to principal y with contents c in protocol run w. $sk(x, y, w)$ represents a key which is shared between x and y for communicating in protocol run w. $ltk(x, y)$ represents a long-term key which is shared between x and y. $encr(k, x, c)$ denotes the result of encrypting c with the key k (x is a flag that stores the principal that did the encryption). A term $N(x, y, w)$ denotes a nonce generated by x for communicating with y in protocol run w. A list made of x and y is represented by $cons(x, y)$. $goal(x, y, w)$ denotes that x tries to establish a communication with y in protocol run w. $c\_init(x, y, z, w)$ represents the fact that x thinks of communicating with y in protocol run w, but in fact x has established a communication with z. $c\_resp(x, y, z, w)$ denotes that x thinks (s)he has replied to a communication requested by y in protocol run w, but in fact (s)he has

[3] Note that an infinite set of servers can be defined in a way similar to the infinite set of agents.

replied to a communication requested by z.

$$goal(agt(a), agt(b), r(j)) \tag{1}$$
$$\rightarrow mesg(agt(a), serv(S), cons(N(agt(a), serv(S), r(j)), cons(agt(a), agt(b))), r(j))$$

$$mesg(a_2, a_3, cons(N(agt(a), serv(S), r(j)), cons(agt(a), agt(b))), r(j)) \tag{2}$$
$$\rightarrow mesg(serv(S), agt(a), encr(ltk(agt(a), serv(S)), serv(S), cons(N(agt(a), serv(S), r(j)),$$
$$cons(agt(b), cons(sk(agt(a), agt(b), r(j)), encr(ltk(agt(b), serv(S)), serv(S),$$
$$cons(sk(agt(a), agt(b), r(j)), agt(a)))))), r(j))$$

$$mesg(a_4, a_5, encr(ltk(agt(a), serv(S)), a_3, cons(N(agt(a), serv(S), r(j)), cons(agt(b),$$
$$cons(sk(agt(a), agt(b), r(i_1)), encr(ltk(agt(b), serv(S)), a_1,$$
$$cons(sk(agt(a), agt(b), r(i_2)), agt(a)))))), r(j)) \tag{3}$$
$$\rightarrow mesg(agt(a), agt(b), encr(ltk(agt(b), serv(S)), a_1,$$
$$cons(sk(agt(a), agt(b), r(i_2)), agt(a))), r(j))$$

$$mesg(a_6, a_7, encr(ltk(agt(b), serv(S)), a_5, cons(sk(agt(a), agt(b), r(i)), agt(a))), r(j)) \tag{4}$$
$$\rightarrow mesg(a_7, a_6, encr(sk(agt(a), agt(b), r(i)), a_7, N(agt(b), agt(a), r(j))), r(j))$$

$$mesg(a_8, a_6, encr(sk(agt(a), agt(b), r(i)), a_7, N(agt(b), agt(a), r(j))), r(j)) \tag{5}$$
$$\rightarrow mesg(a_6, a_8, encr(sk(agt(a), agt(b), r(i)), a_6, N(agt(b), agt(a), r(j))), r(j))$$

$$mesg(a_8, a_6, encr(sk(agt(a), agt(b), r(i)), a_7, N(agt(b), agt(a), r(j))), r(j)) \tag{6}$$
$$\rightarrow c\_init(agt(a), agt(b), a_7, r(j))$$

$$mesg(a_{10}, a_6, encr(sk(agt(a), agt(b), r(i)), a_9, N(agt(b), agt(a), r(j))), r(j)) \tag{7}$$
$$\rightarrow c\_resp(agt(b), agt(a), a_9, r(j))$$

$$cons(x, y) \rightarrow x \tag{8}$$

$$cons(x, y) \rightarrow y \tag{9}$$

$$encr(sk(agt(0), agt(x), w), y, z) \rightarrow z \tag{10}$$

$$encr(sk(agt(x), agt(0), w), y, z) \rightarrow z \tag{11}$$

$$encr(sk(agt(s(x_1)), agt(x), w), y, z) \rightarrow z \tag{12}$$

$$encr(sk(agt(x), agt(s(x_1)), w), y, z) \rightarrow z \tag{13}$$

$$encr(ltk(agt(0), serv(S)), y, z) \rightarrow z \tag{14}$$

$$encr(ltk(agt(s(x_1)), serv(S)), y, z) \rightarrow z \tag{15}$$

$$mesg(x, y, z, w) \rightarrow z \tag{16}$$

Fig. 3. $\mathcal{R} = \mathcal{R}_P \cup \mathcal{R}_I$.

The TRS $\mathcal{R} = \mathcal{R}_P \cup \mathcal{R}_I$ we have been using for experimenting our strategy on the NSSK is given in Figure 3, where $\mathcal{R}_P = (1) \div (7)$ and $\mathcal{R}_I = (8) \div (16)$. Rules $(1) \div (5)$ encode the five steps of the protocol. Every protocol step is for-

malized 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. Rules (6)÷(7) formalize the properties under consideration, i.e. the authentication properties for the initiator and the responder of the communication. The rules in $\mathcal{R}_I$ define an intruder's ability of decomposing and decrypting messages. In particular, an intruder can decompose a list (rules (8)÷(9)) and a message (rule (16)) thus learning their contents, can decrypt the contents of a message encrypted with a session key the intruder is sharing with another agent (rules (10)÷(13)) or with a long-term key the intruder is sharing with the server (rules (14)÷(15)). By considering the equational theory defined by $\mathcal{R}_I$, we also get an intruder's ability of composing lists and messages and of encrypting messages with her/his shared keys. According to the Dolev-Yao model, an intruder has further abilities, such as encrypting/decrypting messages using keys that (s)he is not supposed to know. This is due to the fact that an intruder can learn new messages, nonces, keys, etc., through the execution of the protocol sessions and use such knowledge for building attacks on the protocol. In the approach by Genet and Klay, this ability is embedded in the tree automaton $\mathcal{A}$ (see Section 4.2).

Although our TRS $\mathcal{R}$ is based on Genet and Klay's method to formalizing protocols, there are a few differences with respect to their rewrite system, and two new notions are introduced in order to express and verify the freshness property. First, the LHS operator, that keeps track of the steps applied to yield a certain term, and the *add* and $\cup$ operators are omitted. Then, two function symbols are used to distinguish between shared session keys (sk) and shared long-term keys (ltk), as session keys and long-term keys have different properties. Finally, a number denoting the protocol run is added as a further parameter to some function symbols, e.g. $goal$, $mesg$, sk, N, c_init and c_resp. This information is embedded in the model to be able, for example, to distinguish shared keys between the same agents used in different sessions of the protocol, in order to reason on the freshness of keys. These last two questions about different kinds of keys and how to specify freshness have also been addressed by Oehl [32] who, independently of our work, has enriched his formalization of the NSSK by introducing a distinction among keys (similar to ours) and a notion of *freshness level* to check the freshness of information.

In fact, when defining the rewrite system $\mathcal{R}_P$, we have tried to single out some general guidelines to formalizing the protocol steps. They can be summarized as follows. The left hand side of the first rule is of the form $goal(agt(a), agt(b), r(j))$. The left hand side of a rule, whose root is $mesg$,

is such that (i) the sender and the recipient of the message are encoded as variables; (ii) if there is an encryption, the principal that did it is a variable, possibly different from the sender. The left hand side of a rule (except the first one) is a message whose contents is equal (modulo replacing some terms with variables, e.g. (ii) above) to the contents of the message in the right hand side of the previous rule.

As far as the right hand sides are concerned, we have the following: (iii) agents, server and messages are explicitly specified using the corresponding function symbols, except in rules (4)÷(5) that encode the steps of the protocol where the communication is between agents; (iv) the principal that does an encryption coincides with the sender, except when the sender simply forwards a message encrypted by another principal, as in rule (3).

These hints simply result from our experience on formalizing the usual description of the NSSK into a rewrite system. They need to be tested and applied to other symmetric-key protocols and possibly extended to other classes of protocols.

4.2 The Intruder's Knowledge

We now need to formalize the intruder's initial knowledge and give rules for deducing his/her incremental knowledge obtained while running the protocol sessions.

The intruder's initial knowledge is encoded by means of a tree automaton $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$, which is a simple extension of Oehl's version [32], where $\mathcal{Q}_f = \{q_f\}$. The transitions in Δ are listed in Figure 4. In particular, the following transitions allow one to encode an intruder's abilities of building messages with what is in his/her knowledge, according to the Dolev-Yao model: $mesg(q_f, q_f, q_f, q_f) \rightarrow q_f$ asserts that an intruder can build a message with what (s)he knows; $cons(q_f, q_f) \rightarrow q_f$ expresses an intruder's ability of composing a list, and $encr(q_f, q_{agtI}, q_f) \rightarrow q_f$ asserts that an intruder can encrypt/decrypt information with any key (s)he knows.

The rules for deriving the intruder's incremental knowledge are given by means of a proof system that checks whether a term can be recognized by the intruder. Let t be a term that describes a property to be proved or disproved. In the approximation technique, checking whether t can be recognized by the approximation automaton $\mathcal{T}_{\mathcal{R}}\uparrow(\mathcal{A})$ means checking whether a transition $t \rightarrow q_f$ can be generated from critical pairs. Our verification strategy does not build $\mathcal{T}_{\mathcal{R}}\uparrow(\mathcal{A})$, but starting from t simulates 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

$0 \to q_0$	$0 \to q_{int}$	
$r(q_0) \to q_{r_0}$	$s(q_{int}) \to q_{int}$	
$s(q_0) \to q_1$		
$r(q_1) \to q_{r_1}$	$agt(q_{int}) \to q_{agtI}$	
$A \to q_A$	$agt(q_A) \to q_{agtA}$	
$B \to q_B$	$agt(q_B) \to q_{agtB}$	
$S \to q_S$	$serv(q_S) \to q_{serv}$	
$goal(q_{agtA}, q_{agtB}, q_f) \to q_f$	$goal(q_{agtA}, q_{agtA}, q_f) \to q_f$	*communication requests*
$goal(q_{agtB}, q_{agtA}, q_f) \to q_f$	$goal(q_{agtB}, q_{agtB}, q_f) \to q_f$	
$goal(q_{agtA}, q_{agtI}, q_f) \to q_f$	$goal(q_{agtI}, q_{agtA}, q_f) \to q_f$	
$goal(q_{agtB}, q_{agtI}, q_f) \to q_f$	$goal(q_{agtI}, q_{agtB}, q_f) \to q_f$	
$goal(q_{agtI}, q_{agtI}, q_f) \to q_f$		
$r(q_0) \to q_f$		*intruder's initial knowledge*
$r(q_1) \to q_f$	$ltk(q_{agtI}, q_{serv}) \to q_f$	
$agt(q_{int}) \to q_f$	$sk(q_{agtI}, q_{agtI}, q_f) \to q_f$	
$agt(q_A) \to q_f$	$sk(q_{agtI}, q_{agtA}, q_f) \to q_f$	
$agt(q_B) \to q_f$	$sk(q_{agtI}, q_{agtB}, q_f) \to q_f$	
$serv(q_S) \to q_f$	$N(q_{agtI}, q_{agtI}, q_f) \to q_f$	
$mesg(q_f, q_f, q_f, q_f) \to q_f$	$N(q_{agtI}, q_{agtA}, q_f) \to q_f$	
$cons(q_f, q_f) \to q_f$	$N(q_{agtI}, q_{agtB}, q_f) \to q_f$	
$encr(q_f, q_{agtI}, q_f) \to q_f$	$N(q_{agtI}, q_{serv}, q_f) \to q_f$	

Fig. 4. The set of transitions Δ.

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 [20,21] 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 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. Section 4.3 below illustrates the expansion process and how to ensure its termination. We now explain what we mean by recognizability.

A term t is *recognizable* by an intruder if q_f can be derived from t using the transitions in Δ. As the strategy simulates a completion process that would produce an 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 proof system $\vdash_{\mathcal{A}}$ (directly built from the transitions in Δ) shown in Figure 5, 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.

$$\frac{t \xrightarrow{*}_{\Delta} q \quad q \in \{q_f, q_{agtI}\}}{t \vdash_{\mathcal{A}} q} \qquad \frac{t_1 \vdash_{\mathcal{A}} q_f \quad t_2 \vdash_{\mathcal{A}} q_f \quad t_3 \vdash_{\mathcal{A}} q_f \quad t_4 \vdash_{\mathcal{A}} q_f}{mesg(t_1,t_2,t_3,t_4) \vdash_{\mathcal{A}} q_f} \qquad \frac{t_1 \vdash_{\mathcal{A}} q_f \quad t_2 \vdash_{\mathcal{A}} q_f}{cons(t_1,t_2) \vdash_{\mathcal{A}} q_f}$$

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

Fig. 5. The proof system $\vdash_{\mathcal{A}}$.

Note that the proof system $\vdash_{\mathcal{A}}$ for the NSSK suitably extends the one for the NSPK defined in [31] by simply adding the inference rule for recognizability of shared session keys.

4.3 Expanding Terms

A term t is expanded with $\mathcal{R}$ if a subterm of t unifies with the right hand side of a rule of $\mathcal{R}$:

$$expansion(t, \mathcal{R}) = \{s = \sigma(t[l]_p) \mid \exists\, l \to r \in \mathcal{R}, p \in Pos'(t) \text{ and } \sigma = mgu(t|_p, r)\}.$$

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 $Inst = \{c_1, \ldots, c_k\}$, thus getting the *instance set* $\mathcal{I}(t, Inst) = \{\sigma t \mid \sigma : Var(t) \to Inst\}$.

Inst is a parameter of the strategy and is built as follows. The terms that can instantiate the variables in the TRS for the NSSK are constructed based on the function symbols in the given signature, the names of principals (the server and the agents with their labels), the numbers for denoting the intruders and the protocol runs. We are interested in the behaviour of two agents A and B, a server S and an intruder (represented by $agt(0)$) while running two protocol sessions, typically chosen as $r(0)$ and $r(s(0))$. Thus, it is sufficient to take $Inst = \{A, B, agt(A), agt(B), serv(S), agt(0), 0, s(0)\}$.

A notion of *well-formedness* of terms is used for ensuring the termination of the expansion process. This notion 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 TRS $\mathcal{R}$ for the insecure NSSK, $t_1 = N(agt(a_1), agt(a_2), w)$ is well-formed for any protocol run w and agent labels a_1, a_2, while the term $t_2 = N(agt(a_1), sk(agt(a_2), agt(a_3), w'), w)$

is not, as there is no term t in $\mathcal{R}$ such that $root(t) = N$ and the second argument of t starts with sk. 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 principal and one term of type protocol run, thus the type-checking of t_2 will fail.

We assume that the properties to be proved or disproved on a protocol are described by well-formed terms 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 might be not enough for ensuring the termination of the expansion process (see, for example, the expansion process with the TRS for the NSPK in [31]). 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 NSSK we give the following definition.

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

- $f = agt$ and $t_1 \in L_{agt}$;
- $f = serv$ and $t_1 = S$;
- $f = r$ and $t_1 \in \mathbb{N}$;
- $f = goal$, $root(t_1) = root(t_2) = agt$, $root(t_3) = r$ and $wf(t_i)$ for $i = 1, 2, 3$;
- $f = mesg$, $root(t_1), root(t_2) \in \{agt, serv\}$, $root(t_3) \in \{encr, cons\}$, $root(t_4) = r$ and $wf(t_i)$ for $i = 1, 2, 3, 4$;
- $f = encr$, $root(t_1) \in \{sk, ltk\}$, $root(t_2) \in \{agt, serv\}$, $root(t_3) \in \{cons, N\}$ and $wf(t_i)$ for $i = 1, 2, 3$;
- $f = ltk$, $root(t_1) = agt$, $root(t_2) = serv$ and $wf(t_i)$ for $i = 1, 2$;
- $f = sk$, $root(t_1) = root(t_2) = agt$, $root(t_3) = r$ and $wf(t_i)$ for $i = 1, 2, 3$;
- $f = cons$, $root(t_i) \in \{N, agt, sk, cons, encr\}$ and $wf(t_i)$ for $i = 1, 2$;
- $f = N$, $root(t_1), root(t_2) \in \{agt, serv\}$, $root(t_3) = r$ and $wf(t_i)$ for $i = 1, 2, 3$;
- $f \in \{c\_init, c\_resp\}$, $root(t_1) = root(t_2) = root(t_3) = agt$, $root(t_4) = r$ and $wf(t_i)$ for $i = 1, 2, 3, 4$.

5 The Rewriting Strategy

The input to the strategy is given by the TRS specifying the protocol $\mathcal{R} = \mathcal{R}_P \cup \mathcal{R}_I$, the predicate *wf*, the instantiation set *Inst*, the proof system $\vdash_{\mathcal{A}}$ based on the intruder's initial knowledge Δ, and the well-formed term t_{in} describing the property under consideration. The strategy is defined through the set of inference rules given in Figure 6. 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. The results about the correctness, termination and completeness of the strategy have been proved in [31].

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. 6. 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). Given instances t_i of t that 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 TRS $\mathcal{R}$, the predicate *wf*, the instantiation set *Inst*, the proof system $\vdash_{\mathcal{A}}$ and the well-formed term $t_{in} \in \mathcal{L}(\mathcal{A}_{\overline{p}})$, where $\mathcal{A}_{\overline{p}}$ is the negation automaton for the property to be checked, the correctness, termination and completeness of the strategy are formalized as follows [31].

Proposition 5.1 *(correctness)*
Let $t_{in} \in \mathcal{L}(\mathcal{A}_{\overline{p}})$.
(i) If $\{t_{in}\} \vdash$ success*, then the transition* $t_{in} \to q_f$ *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.*

Proposition 5.2 *(termination)*

The rewriting strategy terminates on any well-formed input term $t_{in} \in \mathcal{L}(\mathcal{A}_{\overline{p}})$.

Note that, by definition of the strategy, 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 $\mathcal{R}_P$. In particular, this guarantees that the execution of the protocol, either forward (usual rewriting) or backward (well-formed expansion), will not loop on a subset of the rules in $\mathcal{R}$. Moreover, by construction of rules $(1) \div (5)$ that describe the steps of the protocol, it cannot happen that a step is skipped, as the left hand side of each rule (except the first one) is a more general term than the right hand side of the previous rule, but is not matching with any other right hand side of rules $(1) \div (5)$.

Corollary 5.3 *(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.

6 Applying the Strategy on the NSSK

This section illustrates the derivation of Denning and Sacco's authentication attack. We consider two protocol sessions $r(0)$ and $r(s(0))$ between agents A and B, where $r(0)$ and $r(s(0))$ correspond to sessions (i) and (ii) of Section 3 respectively. The assumptions are that an intruder, here represented by $agt(0)$, has recorded the run $r(0)$ and the shared key $skey(agt(A), agt(B), r(0))$ has been compromised and is known to the intruder. Thus, the intruder's initial knowledge at the beginning of run $r(s(0))$ also includes all messages exchanged during session $r(0)$ and the axiom $skey(agt(A), agt(B), r(0)) \vdash_{\mathcal{A}} q_f$.

The initial term for the proof of the authentication attack is

$$t_{in} = c\_resp(agt(B), agt(A), agt(0), r(s(0))) \in \mathcal{L}(\mathcal{A}_{\overline{a}})$$

where $\mathcal{A}_{\overline{a}}$ denotes the negation automaton for the property of authentication. [4] The derivation develops as follows. By applying the inference rule Well-formed

[4] Due to the definition of tree automaton, t_{in} is actually $c\_resp(q_{agtB}, q_{agtA}, q_{agtI}, q_{r_1})$, that gets expanded into $c\_resp(agt(B), agt(A), agt(0), r(s(0)))$. Here we abstract from these details.

Expansion on $\{t_{in}\}$ using rules (7), (5), (4) in $\mathcal{R}_P$, the last three steps of session (ii) are performed backward, thus yielding the following configurations:

$$\{c\_resp(agt(B), agt(A), agt(0), r(s(0)))\}$$
$$\vdash \{mesg(a_{10}, a_6, encr(sk(agt(A), agt(B), r(i)), agt(0), N(agt(B), agt(A), r(s(0)))), r(s(0)))\}$$
$$\vdash \{mesg(a_6, agt(0), encr(sk(agt(A), agt(B), r(i)), a_7, N(agt(B), agt(A), r(s(0)))), r(s(0)))\}$$
$$\vdash \{mesg(agt(0), a_6, encr(ltk(agt(B), serv(S)), a_5, cons(sk(agt(A), agt(B), r(i)), agt(A))), r(s(0)))\}$$

The term, say t, in the last configuration cannot be further expanded and does not contain t_{in} as subterm. Thus, the variables $\{a_6, a_5, i\}$ of t are instantiated through *Inst*. Among the various possible instances, let us choose the substitution $\sigma = \{agt(B)/a_6, serv(S)/a_5, 0/i\}$ and compute the function $\overline{rec}$ on σt. We have that the term

$$t_1 = encr(ltk(agt(B), serv(S)), serv(S), cons(sk(agt(A), agt(B), r(0)), agt(A)))$$

is a non-recognizable subterm of σt, i.e. it does not belong to the intruder's basic knowledge. Note that t_1 is part of a message, say m, sent from A to B in session $r(0)$, that is known to the intruder by hypothesis. Using this assumption on m and the TRS $\mathcal{R}_I$, the recognizability of t_1 can be derived and added to the proof system $\vdash_{\mathcal{A}}$. This is not yet embedded in the strategy that checks the recognizability of t_1 by executing backward the steps that have generated m. In fact, the strategy applies rule Split that adds t_1 to the set E and then expands t_1 using rules (18), (3), (2), (1) in $\mathcal{R}$. This yields the following derivation (among possible others) that performs the first three steps of session (i):

$$t_1$$
$$\vdash mesg(x, y, encr(ltk(agt(B), serv(S)), serv(S), cons(sk(agt(A), agt(B), r(0)), agt(A))), w)$$
$$\vdash mesg(a_4, a_5, encr(ltk(agt(A), serv(S)), a_3, cons(N(agt(A), serv(S), r(j)), cons(agt(B),$$
$$cons(sk(agt(A), agt(B), r(i_1)), encr(ltk(agt(B), serv(S)), serv(S),$$
$$cons(sk(agt(A), agt(B), r(0)), agt(A)))))), r(j))$$
$$\vdash mesg(a_2, serv(S), cons(N(agt(A), serv(S), r(0)), cons(agt(A), agt(B))), r(0))$$
$$\vdash goal(agt(A), agt(B), r(0))$$

By rule Success1 we have *success*, the property represented by the term t_{in} is satisfied and Denning and Sacco's authentication attack is thus derived.

Note that our strategy builds another derivation path leading to *success*, when choosing the substitution $\sigma' = \{agt(B)/a_6, serv(S)/a_5, s(0)/i\}$ in rule Split. This represents another authentication attack on the NSSK due to the fact that in run $s(0)$ the intruder replays the message in step (3), thus fooling agent B into thinking that A is trying to establish two protocol sessions with B (multiplicity attack [25]).

7 Concluding Remarks and Future Work

This paper has presented a formalization of the NSSK based on rewrite systems, tree automata and proof systems, and an approach to the analysis and the verification of protocol specifications by means of a bottom-up rewriting strategy. Although there are some notions in common with Rewriting Logic, we are not using such a framework.

Our formalization of the NSSK introduces two new elements in the approach given by Genet and Klay for specifying security protocols and verifying the secrecy and authentication properties. In order to deal with freshness of information, we make use of a notion of protocol run (i.e. a kind of timestamp) and distinguish between shared session keys and shared long-term keys. In his thesis Oehl [32] has developed a similar approach to formalizing the freshness property. The use of session identifiers is not new in protocol specifications and it follows the lines of Denning and Sacco's correction to the NSSK [15]. However, our aim is not, for the moment, formalizing the amended version of the NSSK, but formally deriving the attacks on the insecure version. Based on this formalization of the NSSK, the strategy is able to formally derive two authentication attacks, including Denning and Sacco's one.

Our strategy is inspired by a previously defined rewriting strategy [20,21] for dealing with the problem of divergence in the completion of equational theories and by the work on the approximation technique and its application to protocol verification [17,18,19,33,34]. We have borrowed the formalization of a protocol as a combination of rewrite systems and tree automata, but our strategy does not depend on any approximation and is able to derive whether a property is satisfied or not. Moreover, in the approximation technique, the resulting automaton $\mathcal{T}_{\mathcal{R}}\uparrow(\mathcal{A})$ is usually 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. Finally, whenever attacks are found on a protocol, feedback on error location can be obtained by going back along the critical pairs in the bottom-up search and using the substitutions applied to derive the steps of the attack.

The task of formalizing a protocol by means of a rewrite system may be difficult and error-prone. Choosing variables rather than non-variable terms in some positions of the left (right) hand side of a rule may prevent from deriving certain behaviours of a protocol. Our formalization for the NSSK might appear as an ad-hoc encoding. Actually, we have tried to abstract some criteria to help defining, in particular, the rewrite system $\mathcal{R}_P$ and improving our understanding of a protocol. Our aim is now to check whether these criteria can be applied to other symmetric-key protocols and possibly to other classes

of protocols, thus developing a general (and possibly automated) approach to the formalization of security protocols based on rewriting techniques.

We also need to test our strategy on other protocols and properties, and show its generality and independence of the security protocol under consideration. Note that, when moving from the NSPK to the NSSK, the verification strategy has remained unchanged with respect to its definition in [31]. We are currently working on the Otway-Rees protocol, its formalization and the application of the strategy on it. We are also considering the implementation of the strategy using a more expressive strategy language, e.g. the one of the ELAN system [12], or a theorem proving environment. In general, our strategy can be implemented in any verification tool, which is based on the symbolic manipulation of the representation of systems and properties and is provided with a language of tactics and strategies. [5]

Finally, more study is needed to better characterize well-formedness and the relationship between the rules describing a protocol and the languages characterizing the properties under consideration, by providing more general criteria for ensuring the correctness and the termination of the strategy.

References

[1] Abadi, M., and A. D. Gordon, *A Calculus for Cryptographic Protocols*, Journal of the ACM **148(1)** (1999), 1–70.

[2] Armando, A., D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. H. Drielsma. P.-C. Héam, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò and L. Vigneron, *The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications*, in Proceedings CAV 2005, Lecture Notes in Computer Science, Springer-Verlag, to appear.

[3] Baader, F., and T. Nipkow, "Term Rewriting and All That", Cambridge University Press, Cambridge, 1998.

[4] Basin, D., S. Mödersheim and L. Viganò, *An On-The-Fly Model-Checker for Security Protocol Analysis*, in Proceedings ESORICS 2003, Lecture Notes in Computer Science, Vol. 2808, Springer-Verlag, 253–270.

[5] Bistarelli, S., I. Cervesato, G. Lenzini and F. Martinelli, *Relating Multiset Rewriting and Process Algebras for Security Protocol Analysis*, Journal of Computer Security **13(1)** (2005), 3–47.

[6] Blanchet, B., *From Secrecy to Authenticity in Security Protocols*, in Proceedings SAS'02, Lecture Notes in Computer Science, Vol. 2477, Springer-Verlag, 342–359.

[7] Boichut, Y., P.-C. Héam, O. Kouchnarenko and F. Oehl, *Improvements on the Genet and Klay Technique to Automatically Verify Security Protocols*, in Proceedings 3rd International Workshop on Automated Verification of Infinite-State Systems (AVIS 2004), Barcelona, April 2004.

[5] Note that the rewriting strategy for dealing with the divergence in the completion of equational theories has been implemented as a derived rule of inference in the HOL proof assistant [21].

[8] Boreale, M., R. De Nicola and R. Pugliese, *Proof Techniques for Cryptographic Processes*, SIAM Journal on Computing **31(3)** (2002), 947–986.

[9] Buchholtz, M., H. Riis Nielson and F. Nielson, *A Calculus for Control Flow Analysis of Security Protocols*, International Journal of Information Security **2(3-4)** (2004), 145–167.

[10] Cervesato, I., N. Durgin, P.D. Lincoln, J.C. Mitchell and A. Scedrov, *A Comparison between Strand Spaces and Multiset Rewriting for Security Protocol Analysis*, Journal of Computer Security **13(2)** (2005), 265–316.

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

[12] Cirstea, H., C. Kirchner, L. Liquori and B. Wack, *Rewrite Strategies in the Rewriting Calculus*, in Final Proceedings WRS'03, Electronic Notes in Theoretical Computer Science **86(4)** (2003), URL: http://www.sciencedirect.com/science/journal/15710661.

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

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

[15] Denning, D., and G. Sacco, *Timestamps in key distributed protocols*, Communications of the ACM **24(8)** (1981), 533–535.

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

[17] Genet, T., *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.

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

[19] Genet, T., 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.

[20] Inverardi, P., 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.

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

[22] Jacobs, B., *Semantics and Logic for Security Protocols*, Manuscript, September 2004, URL: http://www.cs.ru.nl/B.Jacobs/PAPERS/protsemlog.pdf.

[23] Jacquemard, F., 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.

[24] Lowe, G., *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.

[25] Lowe, G., *A Family of Attacks upon Authentication Protocols*, Technical Report 1997/5, Dept. of Mathematics and Computer Science, University of Leicester, January 1997.

[26] Meadows, C. A., *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.

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

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

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

[30] Needham, R. M., and M. D. Schroeder, *Authentication revisited*, Operating Systems Review **21(7)** (1987).

[31] Nesi, M., G. Rucci and M. Verdesca, *A Rewriting Strategy for Protocol Verification*, in Final Proceedings WRS'03, Electronic Notes in Theoretical Computer Science **86(4)** (2003), URL: http://www.sciencedirect.com/science/journal/15710661.

[32] Oehl, F., "Formal Verification of Cryptographic Protocols", Ph.D. Thesis, Dublin City University, School of Computing, 2005 (in preparation).

[33] Oehl, F., 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, December 2001.

[34] Oehl, F., 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.

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

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

[37] Weidenbach, C., *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.