

Alessandro Armando · Luca Compagna

SAT-based Model-Checking for Security Protocols Analysis

Abstract We present a model checking technique for security protocols based on a reduction to propositional logic. At the core of our approach is a procedure that, given a description of the protocol in a multi-set rewriting formalism and a positive integer k , builds a propositional formula whose models (if any) correspond to attacks on the protocol. Thus, finding attacks on protocols boils down to checking a propositional formula for satisfiability, problem that is usually solved very efficiently by modern SAT solvers. Experimental results indicate that the approach scales up to industrial strength security protocols with performance comparable with (and in some cases superior to) that of other state-of-the-art protocol analysers.

Keywords Security Protocols · Bounded Model Checking · SAT-based Model Checking · Multi-set Rewriting

1 Introduction

In a world strongly dependent on distributed data communication, the design of secure IT infrastructures is a crucial task. At the core of computer security-sensitive applications are security protocols i.e. communication protocols aiming at providing security guarantees (such as authentication of principals or secrecy of some piece

of information) through the application of cryptographic primitives. The problem is that—in spite of their apparent simplicity—the design of security protocols is notoriously error-prone. Many published protocols have been implemented and deployed in real applications only to be found flawed years later. (See, e.g., [27] for a survey.) Quite interestingly, many attacks can be carried out without breaking cryptography. These attacks exploit weaknesses in the protocol that are due to the complex and unexpected interleavings of different protocol sessions as well as to the possible interference of malicious agents. Since these weaknesses are very difficult to spot by simple inspection of the protocol specification, security protocols are considered a new, promising application domain for formal methods.

In the last decade we have witnessed a dramatic speed-up of SAT solvers: problems with several thousands variables are now solved routinely in milliseconds by state-of-the-art SAT solvers. This has led to breakthroughs in important areas such as planning and model-checking of hardware. An interesting question is whether SAT solvers can lead to similar results also in security protocol analysis. In this paper we try to answer this question and we do this by presenting and evaluating (both theoretically and experimentally) a model checking technique for security protocols based on a reduction to propositional logic.

We model the problem of determining whether a security protocol fails to meet the security properties it is expected to enjoy as a reachability problem (called protocol insecurity problem) of a transition system specifying all the possible actions of the participating agents. Among the participating agents we include an active intruder that, besides overhearing and intercepting messages exchanged by honest participants, can also forge and send new messages. As in [24,6] we specify protocol insecurity problems using a rule-based specification framework based on (order-sorted) multi-set rewriting. The state of the transition system is represented by a set of ground (i.e. variable-free) first-order atomic formulae called facts, while the allowed transitions are specified by

Alessandro Armando
AI-Lab, DIST – Università di Genova.
Viale Causa 13, 16145 Genova, Italy.
Tel.: +39-010-3532216
Fax: +39-010-3532948
E-mail: armando@dist.unige.it

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

and
AI-Lab, DIST – Università di Genova.
Viale Causa 13, 16145 Genova, Italy.
E-mail: compa@dist.unige.it

rewrite rules modelling the abilities of the agents. Each rewrite rule is a declarative specification of the applicability conditions of the associated action (in terms of the facts that must hold in a state for the action to be applicable in it) and of its effects (namely, the facts that must be replaced from and added to the current state in order to obtain the next state).

It is well-known that solving protocol insecurity problems is undecidable in the general case [36]. However, as shown in [61], solving protocol insecurity problems with a bounded number of acyclic sessions (BNAS problems, for short) is NP-complete and hence decidable. In this paper we focus on the bounded protocol insecurity problem, i.e. the problem of determining whether any given security protocol is vulnerable to attacks of length k , for any given bound $k > 0$. This problem is obviously decidable. Building decision procedures for the bounded protocol insecurity problem is important. In fact, any decision procedure for this problem can be used to build a semi-decision procedure for (unrestricted) protocol insecurity problems by simply doing iterative deepening on the value of k . Moreover it can be shown (by using the polynomial bounds on normal attacks discussed in [61]) that any given BNAS problem can be safely reduced to a bounded version of the problem with a value of k quadratic in the size of the protocol specification. Thus any decision procedure for the bounded protocol insecurity problem can be used as a decision procedure for BNAS problems.

Given a protocol insecurity problem and a positive integer k , i.e. given a bounded protocol insecurity problem, we show how to build a propositional formula whose models (if any) correspond to sequences R_1, \dots, R_k of sets of rules representing partial order attacks (of length k) on the protocol. (The idea is that rules in the same set R_i , for $i = 1, \dots, k$, can be executed in any order, but all the rules in R_i must be executed after the rules in R_h , for all $h < i$, and before the rules in R_j , for all $j > i$. Thus a partial order attack is a concise representation of all the attacks obtained by imposing a total ordering on the rules in R_i for $i = 1, \dots, k$.) Thus, by using the approach described in this paper, finding attacks of bounded length boils down to generating and checking for satisfiability a propositional formula. Since for the solving phase we can rely on the efficiency of modern SAT solvers, in this paper we will focus on the issue of generating a propositional formula corresponding (in the sense specified above) to the bounded protocol insecurity problem given as input. This activity requires some ingenuity, as a naive approach to the problem leads to propositional formulae of unmanageable size even for relatively simple problems. The following techniques make our approach not only viable but also effective of many problems of practical interest:

Optimisations. We apply a number of optimising transformations on the input problem. The effect of these transformations is to reduce considerably the number

of messages and the number of actions performed by the participating agents that need to be considered during the analysis.

Quantifier-elimination. Existential quantifiers are used in our multi-set specification formalism to model the generation of fresh data. We provide two methods, that by eliminating the quantifiers from the protocol specification, remove the non-determinism associated with the generation of fresh data.

Propositional Encoding. We generate a propositional formula that encodes all possible partial order attacks of bounded length on the resulting protocol insecurity problem. We have adapted two encoding techniques originally developed for planning to the domain of security protocol analysis.

-
1. Given a protocol insecurity problem Ξ and a bound $k > 0$;
 2. $i := 0$;
 3. Apply the optimising transformations to Ξ ;
 4. Remove the quantifiers from Ξ ;
 5. **while** $i \leq k$
 6. Generate the propositional formula $[[\Xi]]_i$ encoding all the possible partial order attacks on Ξ of length $\leq i$;
 7. Check $[[\Xi]]_i$ by invoking a SAT solver;
 8. If $[[\Xi]]_i$ is satisfiable, then build and print the partial order attack corresponding to the model found for the formula and **halt**;
 9. $i := i + 1$;
 10. **Print** “No partial order attack of length $\leq k$ exists” and then **halt**;
-

Fig. 1 SAT-based model checking of security protocols

We have built a SAT-based model-checker (SATMC) for security protocols based on the above ideas. An abstract account of the internal functioning of SATMC is given by the procedure of Figure 1. Given as input a protocol insecurity problem Ξ specified in our multi-set rewriting specification formalism and a bound $k > 0$ (which can be set also to ∞), the procedure starts by optimising and removing the existential quantifiers from Ξ (lines 3 and 4 respectively) and then enters a loop (with increasing values of a counter i , initially set to 0) in which it builds a propositional formula $[[\Xi]]_i$ representing all possible partial order attacks on Ξ of length $\leq i$ and stops as soon as (i) an attack is found (line 8) or (ii) the bound is reached (lines 5 and 10). If k is set to a positive number, then the procedure is a decision procedure for the corresponding bounded protocol insecurity problem. If k is set to ∞ , then the procedure may not terminate and is a semi-decision procedure for the corresponding protocol insecurity problem.

The approach neatly separates the encoding phase from the search activity that (if we neglect the external loop implementing the iterative deepening) is completely delegated to the SAT solver. This is a very attractive feature as state-of-the-art SAT solvers are very powerful

search engines that combine the sheer speed ensured by finely tuned data structures and algorithms with the high sophistication of advanced search strategies which often lead to a dramatic pruning of the search space. Moreover, virtually all state-of-the-art SAT solvers accept propositional formulae in the same common format. This makes their integration immediate and their replacement (as soon as new, faster SAT solvers become available) almost effortless.

The encoding techniques we present in this paper are based on the idea to add a time-index to the facts and rules to indicate the time step in which the facts hold and the rules apply respectively. (Rules are ground first-order atomic formulae used to unambiguously refer to the ground instances of the rewrite rules.) Facts are thus indexed by 0 through k and rules by 0 through $k - 1$. The propositional formula $\llbracket \Xi \rrbracket_i$ is obtained by conjoining (i) a set of 0-indexed facts specifying the initial state, (ii) a propositional combination of $(k + 1)$ -indexed facts representing the set of states whose reachability implies the violation of the security property that the protocol is expected to guarantee, (iii) a set of formulae specifying the applicability conditions of the actions at time step i as well as their effects on the state at time step $i + 1$, for $i = 0, \dots, k$, (iv) a set of frame axioms stating that actions are the only cause of change, and (v) some further formulae whose form and meaning depend on the specific encoding technique employed.

As we will see, the size of the resulting propositional formula grows linearly in the number of facts and quadratically in the number of rules in the worst case. Unfortunately both the number of facts and rules (and hence also the size of the propositional formula) are in the worst case exponential in the size of the input security protocol specification. This result seems to limit the applicability of the approach to small protocols. However computer experiments, obtained by running SATMC on complex, industrial strength security protocols, show that our approach generates formulae of manageable size in most cases and exhibits performance comparable with (and in some cases better than) that of other state-of-the-art protocol analysers. We believe that this result is due to (i) the proper use of sort declarations in the protocol specifications, (ii) the effectiveness of the optimising transformations and of the encoding techniques, and (iii) the ability of the SAT solver to quickly solve complex combinatorial problems. The combined effect of (i) and (ii) makes the approach generate remarkably compact encodings, while (iii) allows SATMC to scale up smoothly as the number of sessions in the problem considered increases.

The technical contribution of the paper is fourfold:

- We provide an operational semantics for multi-set rewriting systems that takes into account the non-determinism associated with the generation of fresh data and show that this form of non-determinism can

be safely eliminated. In most accounts (e.g. [24,6]) this is either taken for granted or abstracted away.

- In previous work [7,10] we showed how encoding techniques originally developed for planning can be adapted and successfully applied to security protocol analysis. In this paper we provide a detailed account of these techniques and prove their soundness and completeness.
- We provide a detailed and comprehensive account of our SAT-based model checking procedure for security protocols. We show that our procedure can be turned into a decision procedure for BNAS problems.
- We provide a thorough experimental assessment of our approach obtained by running SATMC and two other state-of-the-art tools against a large set of protocol insecurity problems drawn from two publicly available libraries of security protocols. The experiments indicate that SATMC is more sensitive to the size of messages exchanged by the protocols than the other two tools considered but also that it scales better as the number of sessions increases. This is an interesting result that indicates that our approach is complementary to other state-of-the-art techniques.

Structure of the paper. In Section 2 we give a brief presentation of the Needham-Schroeder Public-Key Protocol that will be used as a running example throughout the rest of the paper. (This section can be skipped by readers who are familiar with this protocol.) In Section 3 we show how the notion of protocol insecurity problem can be expressed as a reachability problem in a multi-set rewriting formalism. In Section 4, we focus on the problem of encoding bounded protocol insecurity problems into propositional logic. In Section 5 we discuss how the previously presented techniques can be put together to obtain a decision procedure for BNAS problems and, more in general, a semi-decision procedure for protocol insecurity problems. In Section 6 we present and analyse the experimental results. The related work is discussed in Section 7. We conclude in Section 8 with some final remarks. Proofs of the main results are given in the appendix.

2 An Example: the Needham-Schroeder Public-Key Protocol

We will use the well-known Needham-Schroeder Public-Key (NSPK) protocol [57] as a running example throughout the paper. The NSPK protocol is intended to ensure the mutual authentication of the participating agents Alice and Bob through a challenge-response mechanism which is succinctly represented by following three protocol steps:

- (1) $A \rightarrow B : \{A, Na\}_{Kb}$
- (2) $B \rightarrow A : \{Na, Nb\}_{Ka}$
- (3) $A \rightarrow B : \{Nb\}_{Kb}$

Alice starts the protocol by sending Bob her own identity and a nonce Na encrypted with Bob's public key. (A *nonce* is a number used only once, i.e. a number chosen in an unpredictable way from a large set of candidates. This assumption ensures that it is extremely difficult for an intruder to guess the nonces generated by honest agents during a run of the protocol.) In this way Alice is guaranteed that only Bob can access the content of the sent message and to Nb in particular. Bob replies by sending a new nonce Nb and the nonce Na received by Alice encrypted by Alice's public key. Again, this ensures that only Alice can access the content of this message. Upon receipt of this message Alice can conclude that this message has been generated and sent by Bob (since nobody else can possibly have generated a message containing Na) and that Bob agrees with her on the values of Na and Nb . Alice then concludes her participation in the protocol by sending the nonce Nb encrypted by Bob's public key. Upon receipt of this message Bob concludes that this message has been generated and sent by Alice (since nobody else can possibly have generated a message containing Nb) and that Alice agrees with him on the values of Na and Nb .

Unfortunately the above argumentation (and hence the protocol) is flawed. While it is correct for Bob to conclude at the end of the protocol that Na has been generated by Alice, it might be the case that Na has been generated by Alice with a different *purpose* than authenticating with Bob. In the scenario of Figure 2 (corresponding to Lowe's attack on the NSPK protocol [49]) nonce na is generated by Alice with the purpose of authenticating with Ives, but Ives mischievously reuses na by forging and sending the message $\{a, na\}_{kb}$. In this way Ives tricks Bob into believing that Alice wishes to initiate the protocol with him. As a consequence of this, at the end Bob concludes he has successfully authenticated Alice and that they both agree with values of Na and Nb , but this is obviously not the case.

3 Protocol Insecurity Problems as Reachability Problems in Multi-Set Rewriting

We model protocol insecurity problems as reachability problems associated with an order-sorted multi-set rewriting system specifying the behaviour of the honest agents as well as that of the intruder.

3.1 Order-sorted Language

An *order-sorted signature* is a triple $\langle \mathcal{S}, \leq, \Sigma \rangle$ such that $\langle \mathcal{S}, \leq \rangle$ is a partially ordered set of sorts and Σ is a set of function symbols, each equipped with a set of *arities*, i.e. a set of sequences of sorts $\langle s_1, \dots, s_n, s \rangle$ for some $n \geq 0$. (We call *individual constants* the function symbols with $n = 0$.) We write $f : s_1, \dots, s_n \rightarrow s$ to indicate that

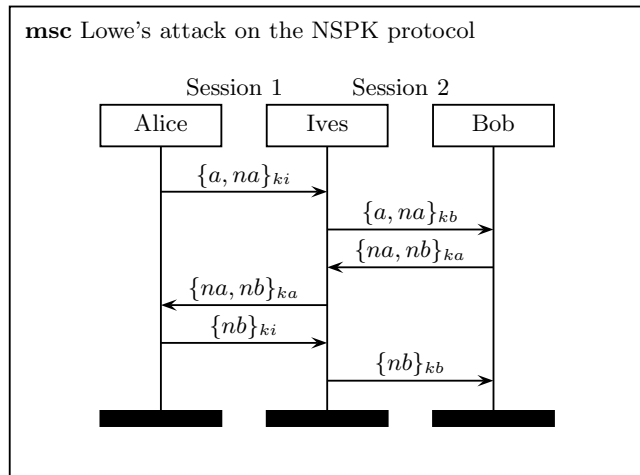


Fig. 2 Message sequence chart of Lowe's Attack on the NSPK protocol

$\langle s_1, \dots, s_n, s \rangle$ is an arity of f . Let $V = \langle V^s \rangle_{s \in \mathcal{S}}$ be an \mathcal{S} -indexed family of variables and let $s \in \mathcal{S}$. We assume the usual inductive definition of Σ -terms of sort s over V (see, e.g., [63]). By $T_s(\Sigma, V)$ we indicate the set of all Σ -terms of sort s over V . A term without variables is said to be *ground* and we abbreviate $T_s(\Sigma, \emptyset)$ with $T_s(\Sigma)$. Let X be a set, by $|X|$ we denote the cardinality of X . If s is a sort, then we write $|s|$ in place of $|T_s(\Sigma, V)|$ and we say that the cardinality of s is $|T_s(\Sigma, V)|$.

We assume that \mathcal{S} has sorts for messages (MSG), agent names (AGENT), (a subset of the) natural numbers (NUM), asymmetric cryptographic keys (KEY), protocol steps (STEPID), data identifiers (ID), and session identifiers (SESSID). Furthermore, $\text{AGENT} \leq \text{MSG}$, $\text{NUM} \leq \text{MSG}$, $\text{KEY} \leq \text{MSG}$, $\text{STEPID} \leq \text{NUM}$, and $\text{SESSID} \leq \text{NUM}$. (Numbers are used to represent nonces, protocol steps, and protocol sessions.) Moreover we assume that Σ has

- individual constants for agents (e.g. a, b, i), for cryptographic keys (e.g. ka, kb, ki), for numbers, protocol step identifiers and session identifiers (e.g. $0, 1, 2, \dots$);
- function symbols for pairing ($\langle _, _ \rangle : \text{MSG}, \text{MSG} \rightarrow \text{MSG}$), for symmetric encryption ($\{_ \}_\#^s : \text{MSG}, \text{MSG} \rightarrow \text{MSG}$), for asymmetric encryption ($\{_ \}_\# : \text{MSG}, \text{KEY} \rightarrow \text{MSG}$), and for denoting the inverse key ($_^{-1} : \text{KEY} \rightarrow \text{KEY}$). (Symbols for other cryptographic operators, e.g. hash functions, can be also included if needed.)

Notice that any message can be used as a symmetric key: this allows us to model protocols of real-world complexity (e.g. TLS) in which symmetric keys are built out of several pieces of information.

We also assume that \mathcal{S} contains two sort symbols FACT and RULE. We call *fact terms* and *rule terms* the terms in $T_{\text{FACT}}(\Sigma, V)$ and $T_{\text{RULE}}(\Sigma, V)$ respectively. (We call *facts* and *rules* the terms in $T_{\text{FACT}}(\Sigma)$ and $T_{\text{RULE}}(\Sigma)$)

respectively.) Finally we call *rule names* the function symbols f such that $f : s_1, \dots, s_n \rightarrow \text{RULE}$ for some s_1, \dots, s_n and $n \geq 0$. We assume that fact terms are of the forms shown in the left column of Table 1, where j is a term of sort `STEPID`, s, s_1, \dots, s_h , and r are terms of sort `AGENT`, o is a term of sort `ID`, c is a term of sort `SESSID`, and m, m_1, \dots, m_h are terms of sort `MSG`. The informal meaning of these facts is explained in the right column. (Notice that, the last four rows describe fact terms used to express security properties and they will thus be discussed again only at the end of Section 3.4.)

Finally we assume that \mathcal{S} has sorts for freshly generated entities. For instance, `FRESH(NUM)` is the sort for nonces and `FRESH(KEY)` is the sort for session keys. In general if s a basic sort, then `FRESH(s)` is the corresponding sort of fresh elements and `FRESH(s) ≤ s`. We assume that Σ is extended with an enumerable set of individual constants (called *fresh constants*) for each sort of type `FRESH(s)`. We require that if c is a constant of sort `FRESH(s)` for some $s \in \mathcal{S}$, then `FRESH(s)` is the only sort of c . If $v \in V^{\text{FRESH}(s)}$ for some $s \in \mathcal{S}$, then we say that v is a *fresh variable*. By Σ^- and V^- we denote the result of dropping from Σ and V the fresh constants and the fresh variables respectively.

3.2 Multi-set Rewriting Systems

Let O be a set of terms. By $\text{vars}(O)$ we denote the set of variables occurring in O . Moreover we write \mathbf{x} to denote a sequence of expressions $\langle x_1, \dots, x_n \rangle$. With abuse of notation in the sequel we use \mathbf{x} also to denote the set of the expressions in \mathbf{x} . Context will disambiguate what is meant.

A *multi-set rewriting system of signature* $\langle \mathcal{S}, \leq, \Sigma \rangle$ over V is a pair $\Gamma = \langle \mathcal{I}, \mathcal{R} \rangle$ where $\mathcal{I} \subseteq T_{\text{FACT}}(\Sigma^-)$ is a multi-set of facts representing the initial state of the system and \mathcal{R} is a function mapping rule names into triples of the form $\langle L, \mathbf{x}, \exists \mathbf{y}.R \rangle$ called *rewrite rules*, where \mathbf{x} and \mathbf{y} is a sequence of the variables occurring in L , \mathbf{y} is a sequence of fresh variables occurring in R , and L and R are multi-sets of terms in $T_{\text{FACT}}(\Sigma^-, V^-)$ and $T_{\text{FACT}}(\Sigma^-, V)$ respectively with $(\text{vars}(R) \setminus \mathbf{y}) \subseteq \text{vars}(L)$. In this paper we will restrict our attention to the simpler class of multi-set rewriting systems in which all the multi-sets are just sets. We will write $(L \xrightarrow{r(\mathbf{x})} \exists \mathbf{y}.R) \in \mathcal{R}$ (or even $(L \xrightarrow{r(\mathbf{x})} \exists \mathbf{y}.R)$, when \mathcal{R} can be inferred from the context) in place of $\mathcal{R}(r) = \langle L, \mathbf{x}, \exists \mathbf{y}.R \rangle$. If \mathbf{y} is the empty sequence of variables, then we will abbreviate $\exists \mathbf{y}.R$ with R . Moreover if f and fs are a fact term and a set of fact terms respectively, then we write $f \cdot fs$ in place of $\{f\} \cup fs$ and f in place of $\{f\}$.

A possible initial state S_0 for the intruder \mathbf{i} and two honest agents \mathbf{a} and \mathbf{b} involved in two concurrent sessions

of the NSPK is:

$$\text{state}(0, \mathbf{a}, \mathbf{a}, [\mathbf{a}, \mathbf{i}, \mathbf{ka}, \mathbf{ka}^{-1}, \mathbf{ki}], 1). \quad (1)$$

$$\begin{aligned} &\text{state}(0, \mathbf{a}, \mathbf{a}, [\mathbf{a}, \mathbf{b}, \mathbf{ka}, \mathbf{ka}^{-1}, \mathbf{kb}], 2). \\ &\text{state}(1, \mathbf{a}, \mathbf{b}, [\mathbf{b}, \mathbf{a}, \mathbf{kb}, \mathbf{kb}^{-1}, \mathbf{ka}], 2). \end{aligned} \quad (2)$$

$$\begin{aligned} &\text{ik}(\mathbf{i}) \cdot \text{ik}(\mathbf{a}) \cdot \text{ik}(\mathbf{b}) \cdot \text{ik}(\mathbf{ki}) \cdot \text{ik}(\mathbf{ki}^{-1}). \\ &\text{ik}(\mathbf{ka}) \cdot \text{ik}(\mathbf{kb}) \end{aligned} \quad (3)$$

Fact (1) represents the initial state of the honest agent \mathbf{a} playing the role of initiator in session 1 and knowing her own identity, the identity of intruder (the agent she would like to talk with), her own public and private keys, and the public key of the intruder. Facts (2) represent the initial state of the honest agents \mathbf{a} and \mathbf{b} involved as initiator and responder (respectively) in session 2. Facts (3) represent the terms initially known by the intruder.

Rewrite rules modelling the behaviour of the honest agents in the NSPK protocol are given in Figure 3. Rewrite rule (4) is applicable to the state comprising facts (1) and (2) when in the substitution the variables $\mathbf{A}, \mathbf{B}, \mathbf{Ka}, \mathbf{Kb}$, and \mathbf{S} are instantiated with $\mathbf{a}, \mathbf{i}, \mathbf{ka}, \mathbf{ki}$, and 1 respectively, possibly leading to the state:

$$\begin{aligned} &\text{state}(2, \mathbf{i}, \mathbf{a}, [\mathbf{na}, \mathbf{a}, \mathbf{i}, \mathbf{ka}, \mathbf{ka}^{-1}, \mathbf{ki}], 1) \\ &\cdot \text{msg}(1, \mathbf{a}, \mathbf{i}, \{\langle \mathbf{a}, \mathbf{na} \rangle\}_{\mathbf{ki}}) \\ &\cdot \text{state}(0, \mathbf{a}, \mathbf{a}, [\mathbf{a}, \mathbf{b}, \mathbf{ka}, \mathbf{ka}^{-1}, \mathbf{kb}], 2) \\ &\cdot \text{state}(1, \mathbf{a}, \mathbf{b}, [\mathbf{b}, \mathbf{a}, \mathbf{kb}, \mathbf{kb}^{-1}, \mathbf{ka}], 2) \end{aligned}$$

where \mathbf{na} is an individual constant of sort `FRESH(NUM)`. As effect of the application of the rule, agent \mathbf{a} (*i*) has sent message $\{\langle \mathbf{a}, \mathbf{na} \rangle\}_{\mathbf{ki}}$ to the intruder \mathbf{i} , (*ii*) has extended her knowledge with the nonce \mathbf{na} , and (*iii*) is ready to execute step 2 of the protocol as witnessed by the fact `state(2, i, a, [na, a, i, ka, ka-1, ki], 1)` occurring in the state. Notice that all the other facts occurring in the initial state are automatically inherited by the new state. The second step of the NSPK protocol in which the responder receives the challenge generated by the initiator and replies with his own nonce, is modelled by (5). The state of the responder is updated by incrementing the protocol step and by extending the knowledge with the acquired information, namely the nonce sent by the initiator and the nonce freshly generated by the responder itself. Moreover, the message $\{\langle \mathbf{Na}, \mathbf{Nb} \rangle\}_{\mathbf{ka}}$ is sent to \mathbf{A} . The remaining steps of the protocol are modelled by (6) and (7).

We consider the Dolev-Yao model of the intruder [31]. The Dolev-Yao intruder has complete control of the network: he can overhear and divert messages, and by using the knowledge gleaned from the observed traffic he forge and send fraudulent messages to the honest participants. These abilities are modelled by the rules of Figure 4. Rewrite rule (8) models the ability of the intruder of diverting the information exchanged by the honest participants. It states that, if a message has been sent on the communication channel, then the intruder can

Table 1 Facts and their informal meaning

Fact	Meaning
$ik(m)$	The intruder knows message m .
$msg(j, s, r, m)$	Principal s has supposedly sent message m to principal r at protocol step j .
$state(j, s, r, [m_1, \dots, m_h], c)$	Principal r is ready to execute step j of session c of the protocol, knows the messages m_1, \dots, m_h , and—if $j \neq 0$ —a message from s to r is awaited for the step to be executed.
$secret(m, [s_1, \dots, s_h])$	Message m is a secret shared among the group of agents s_1, \dots, s_h .
$witness(s, r, o, m)$	Value m of the data item o has been generated by agent s for the purpose of authentication with agent r .
$wrequest(r, s, o, m)$	Agent r accepts the value m of the data item o as originated from s for the purpose of authentication with him.
$request(r, s, o, m, c)$	Agent r accepts the value m of the data item o as originated from s for the purpose of authentication with him in session c .

$$\begin{array}{l} state(0, A, A, [A, B, Ka, Ka^{-1}, Kb], S) \\ \xrightarrow{\text{step}_1(A, B, Ka, Kb, S)} \end{array} \quad (4)$$

$$\exists Na. state(2, B, A, [Na, A, B, Ka, Ka^{-1}, Kb], S) \cdot msg(1, A, B, \{A, Na\}_{Kb})$$

$$\begin{array}{l} msg(1, A, B, \{A, Na\}_{Kb}) \cdot state(1, A, B, [B, A, Kb, Kb^{-1}, Ka], S) \\ \xrightarrow{\text{step}_2(A, B, Ka, Kb, Na, S)} \end{array} \quad (5)$$

$$\exists Nb. state(3, A, B, [Na, Nb, B, A, Kb, Kb^{-1}, Ka], S) \cdot msg(2, B, A, \{Na, Nb\}_{Ka})$$

$$\begin{array}{l} msg(2, B, A, \{Na, Nb\}_{Ka}) \cdot state(2, B, A, [Na, A, B, Ka, Ka^{-1}, Kb], S) \\ \xrightarrow{\text{step}_3(A, B, Ka, Kb, Na, Nb, S)} \end{array} \quad (6)$$

$$state(4, B, A, [Nb, Na, A, B, Ka, Ka^{-1}, Kb], S) \cdot msg(3, A, B, \{Nb\}_{Kb})$$

$$\begin{array}{l} msg(3, A, B, \{Nb\}_{Kb}) \cdot state(3, A, B, [Na, Nb, B, A, Kb, Kb^{-1}, Ka], S) \\ \xrightarrow{\text{step}_4(A, B, Ka, Kb, Na, Nb, S)} \end{array} \quad (7)$$

$$state(5, A, B, [Na, Nb, B, A, Kb, Kb^{-1}, Ka], S)$$

Fig. 3 Rewrite rules for the honest agents in the NSPK protocol

read its content and remove it from the channel. The rules (9), (10), and (11) model the ability of composing messages by encrypting with a known key and by pairing known messages. The deduction of new knowledge by decryption (via known keys) and pair decomposition is modelled by the rules (12), (13), and (14). Finally, the ability to send arbitrary messages (possibly faking somebody else's identity in doing so) is modelled by the rewrite rules of the form (15), for all terms j of sort STEPID:

Notice that the decryption key must be known in order to extract the plaintext from the cyphertext. This is commonly known as the *perfect cryptography* assumption. This assumption is important as it allows us to abstract from the cryptosystem and focus on the logic of the protocol.

We also make the so called *strong typing* assumption that rules out *type-flaw attacks*, i.e. attacks where a field that was originally intended to have one type is subsequently interpreted as having another type. A tagging mechanism that allows for the implementation of protocols enjoying the strong typing assumption is proposed in [43].

3.3 Operational Semantics

A multi-set rewriting system defines a state transition systems whose states are represented by sets of facts. If S is a set of facts, then we interpret the facts in S as the propositions holding in the state represented by S , all other facts being false in that state (closed-world assumption). A (*ground*) *substitution* is a total function mapping the variables in V in (ground, resp.) terms of the same sort. Let $(L \xrightarrow{\rho} \exists \mathbf{y}.R) \in \mathcal{R}$, σ be a ground substitution, and S be a set of facts. If $L\sigma \subseteq S$, then we say that *rule instance* $\langle \rho, \sigma \rangle$ is *applicable in* S and that $S' = \text{app}_{\langle \rho, \sigma \rangle}(S) = (S \setminus L\sigma) \cup R\sigma$ is the *state resulting from the execution of* $\langle \rho, \sigma \rangle$ *in* S . Under the same hypotheses we say that *rule* ρ is *applicable in* S and that S' is a *state resulting from an execution of* ρ *in* S .

Thus a rule models an action that may non-deterministically lead to many (possibly infinite) successor states. For example, all the rule instances corresponding to the rewrite rule (4) are of the form $\langle \rho, \sigma \rangle$ with $\sigma = \{a/A, b/B, ka/Ka, kb/Kb, 1/S, n/Na\}$ for any constant n of sort FRESH(NUM) and all of them correspond to

$$\text{msg}(J, A, B, M) \xrightarrow{\text{divert}(A, B, J, M)} \text{ik}(M) \quad (8)$$

$$\text{ik}(M) \cdot \text{ik}(K) \xrightarrow{\text{encrypt}(K, M)} \text{ik}(M) \cdot \text{ik}(K) \cdot \text{ik}(\{M\}_K) \quad (9)$$

$$\text{ik}(M) \cdot \text{ik}(K) \xrightarrow{\text{sencrypt}(K, M)} \text{ik}(M) \cdot \text{ik}(K) \cdot \text{ik}(\{M\}_K^s) \quad (10)$$

$$\text{ik}(M_1) \cdot \text{ik}(M_2) \xrightarrow{\text{pairing}(M_1, M_2)} \text{ik}(\langle M_1, M_2 \rangle) \quad (11)$$

$$\text{ik}(\{M\}_K) \cdot \text{ik}(K^{-1}) \xrightarrow{\text{decrypt}(K, M)} \text{ik}(\{M\}_K) \cdot \text{ik}(K^{-1}) \cdot \text{ik}(M) \quad (12)$$

$$\text{ik}(\{M\}_K^s) \cdot \text{ik}(K) \xrightarrow{\text{sdecrypt}(K, M)} \text{ik}(\{M\}_K^s) \cdot \text{ik}(K) \cdot \text{ik}(M) \quad (13)$$

$$\text{ik}(\langle M_1, M_2 \rangle) \xrightarrow{\text{decompose}(M_1, M_2)} \text{ik}(M_1) \cdot \text{ik}(M_2) \quad (14)$$

$$\text{ik}(M) \cdot \text{ik}(A) \cdot \text{ik}(B) \xrightarrow{\text{fake}_j(A, B, M)} \text{ik}(M) \cdot \text{ik}(A) \cdot \text{ik}(B) \cdot \text{msg}(j, A, B, M) \quad (15)$$

Fig. 4 Rewrite rules for the intruder

the rule $\text{step}_1(a, b, ka, kb, 1)$. Thus starting from a given state, the execution of a rule may non-deterministically lead the system into different (possibly infinitely many) successor states. At a first sight this seems a serious problem for automatic analysis as each rule application seems to lead to an infinite branching. However, as we will see below, this is a form of “don’t care non-determinism”. This means that no branching at all is needed when considering the application of a rule during the analysis as any (arbitrarily chosen) rule instance will serve as representative of all the others.

A *path* is an alternating sequence of states and rule instances $S_0 \langle \rho_1, \sigma_1 \rangle S_1 \dots S_{n-1} \langle \rho_n, \sigma_n \rangle S_n$ such that, for $i = 1, \dots, n$,

1. $S_i = \text{app}_{\langle \rho_i, \sigma_i \rangle}(S_{i-1})$, i.e. S_i is a state resulting from the execution of $\rho_i \sigma_i$ in S_{i-1} ;
2. if y and y' are existentially quantified variables of ρ_i and ρ_j (resp.) and $\sigma_i(y) = \sigma_j(y')$, then $i = j$ and $y = y'$ for all $i, j = 1, \dots, n$, i.e. the substitutions $\sigma_1, \dots, \sigma_n$ not only associate different existentially quantified variables with different terms but also associate occurrences at different execution steps of the same existentially quantified variable with different fresh constants.

If, additionally, $S_0 \subseteq \mathcal{I}$, then we say that the path is *initialised*. Condition (2) ensures that existentially quantified variables are replaced by fresh terms representing newly generated data (e.g. nonces and session keys). Notice also that the above definition—by excluding the simultaneous execution of two (or more) rules—enforces an *interleaving semantics*. This is a common assumption in security protocol analysis as it rules out execution paths (and hence attacks) that are difficult to reproduce in practice.

Let $\chi = \langle \rho_1, \sigma_1 \rangle, \dots, \langle \rho_n, \sigma_n \rangle$ and $\chi' = \langle \rho'_1, \sigma'_1 \rangle, \dots, \langle \rho'_n, \sigma'_n \rangle$ be two sequence of rule instances of Γ . We say that χ and χ' are *rule equivalent* if and only if $\rho'_i \sigma'_i = \rho_i \sigma_i$ for $i = 1, \dots, n$. A sequence of rule instances $\chi = \langle \rho_1, \sigma_1 \rangle, \dots, \langle \rho_n, \sigma_n \rangle$ is a *run of Γ* if and only if there ex-

ist a sequence of states S_0, S_1, \dots, S_n such that $S_0 \langle \rho_1, \sigma_1 \rangle S_1 \dots S_{n-1} \langle \rho_n, \sigma_n \rangle S_n$ is an initialised path of Γ . When this is the case, we write $S_n = \text{app}_\chi(S_0)$ and we say that S_n is *reachable in n steps from S_0 by applying in sequence the rules $\rho_1 \sigma_1, \dots, \rho_n \sigma_n$* . We say that two states S and S' are *equivalent*, in symbols $S \sim S'$, if and only if they are equal modulo renaming of the fresh constants.

Lemma 1 *Let χ be a run of $\Gamma = \langle \mathcal{I}, \mathcal{R} \rangle$ and χ' be a sequence of rule instances of Γ . If χ' is rule equivalent to χ then also χ' is a run of Γ and $\text{app}_{\chi'}(\mathcal{I}) \sim \text{app}_\chi(\mathcal{I})$.*

3.4 Protocol Insecurity Problems

Given a multi-set rewriting system of signature $\langle \mathcal{S}, \leq, \Sigma \rangle$ over V , we are interested in the reachability problem over it. The bad states (i.e. the states of the system whose reachability implies a violation of a security property that the protocol is expected to enjoy) are represented by *goals*, i.e. pairs of the forms $\mathcal{B} = \langle \mathcal{G}, \mathcal{C} \rangle$ where \mathcal{G} is a quantifier-free formula built out of fact terms $T_{\text{FACT}}(\Sigma^-, V^-)$ using the standard propositional connectives, and \mathcal{C} is a set of equality and inequality constraints of the form $t_1 \simeq t_2$ and $t_1 \not\simeq t_2$ respectively where $t_1, t_2 \in T_s(\Sigma^-, V)$ for some sort s . A *solution of \mathcal{C}* is a ground substitution σ , in symbols $\sigma \models_c \mathcal{C}$, if and only if $t_1 \sigma = t_2 \sigma$ for all $t_1 \simeq t_2 \in \mathcal{C}$ and $t_1 \sigma \neq t_2 \sigma$ for all $t_1 \not\simeq t_2 \in \mathcal{C}$.

A *reachability problem of signature $\langle \mathcal{S}, \leq, \Sigma \rangle$ over V* is a pair $\Xi = \langle \Gamma, \mathcal{B} \rangle$ where Γ is a multi-set rewriting system of signature $\langle \mathcal{S}, \leq, \Sigma \rangle$ over V and \mathcal{B} is a goal.

A state S is *bad*, in symbols $S \models \mathcal{B}$, if and only if there exists a substitution σ such that $S \models \mathcal{G}\sigma$ and $\sigma \models_c \mathcal{C}$, where \models is the entailment relation in propositional logic. A *solution to $\Xi = \langle \Gamma, \mathcal{B} \rangle$* is a run χ of Γ such that $\text{app}_\chi(\mathcal{I}) \models \mathcal{B}$.

Theorem 1 *Let χ be a solution to $\Xi = \langle \Gamma, \mathcal{B} \rangle$ and χ' be a sequence of rule instances of Γ . If χ' is rule equivalent to χ then also χ' is a solution to Ξ .*

A *protocol insecurity problem* is a reachability problem $\Xi = \langle \Gamma, \mathcal{B} \rangle$ where Γ is a multi-set rewriting system specifying the behaviour of the honest agents and of the intruder and \mathcal{B} is a goal specifying the set of states whose reachability implies the violation of one of the security properties that the protocol is expected to enjoy. Any solution to Ξ is an attack on the protocol specified by Ξ .

We now see how common security properties like secrecy and authentication can be specified in our formalism.

Secrecy. If the protocol is intended to ensure the secrecy of a message m among a set of principals $P = \{p_1, \dots, p_k\}$, with $i \notin P$, then we extend either the initial state (if m is present from the outset) or the right hand side of the rewrite rule whose application introduces m with the fact $\mathbf{secret}(m, [p_1, \dots, p_k])$ and we set $\mathcal{G} = \mathbf{secret}(m, [p_1, \dots, p_k]) \wedge \mathbf{ik}(m)$ and $\mathcal{C} = \{p \neq i : p \in P\}$. Intuitively, the fact $\mathbf{secret}(m, [p_1, \dots, p_k])$ expresses that m is to be kept confidential among the principals in P and thus any state in which the intruder knows m is a bad state. In our NSPK example to indicate that the nonces must be kept secret between initiator and responder, we can add the fact terms $\mathbf{secret}(\mathbf{Na}, [\mathbf{A}, \mathbf{B}])$ and $\mathbf{secret}(\mathbf{Nb}, [\mathbf{A}, \mathbf{B}])$ to the right hand sides of the rewrite rules (4) and (5) respectively.

Authentication. A form of authentication, that we call *weak authentication*, is defined in [50] (under the name of *non-injective agreement*) as follows: an initiator p_1 weakly authenticates a responder p_2 on a data item d if, whenever p_1 (acting as initiator) completes a run of the protocol, apparently with p_2 (acting as responder), then (i) p_2 has previously been running the protocol apparently with p_1 and (ii) the two agents agreed on the values v of d .

In order to check for this property we need to carry out a few, minor modifications to the rewrite rules for the honest agents (cf. Figure 3). As a first step we add a fact term of the form $\mathbf{witness}(p_2, p_1, d, v)$ to the right hand side of the rewrite rules that generate data values for the purpose of the authentication. Intuitively a fact of the form $\mathbf{witness}(p_2, p_1, d, v)$ means that value v of the data item d has been generated by agent p_2 for the purpose of authentication with agent p_1 . In our example this extension amounts to adding fact terms $\mathbf{witness}(\mathbf{A}, \mathbf{B}, \mathbf{na}, \mathbf{Na})$ and $\mathbf{witness}(\mathbf{B}, \mathbf{A}, \mathbf{nb}, \mathbf{Nb})$ to the right hand sides of the rewrite rules (4) and (5) respectively. As a second step we add a fact term of the form $\mathbf{wrequest}(p_1, p_2, d, v)$ to the right hand side of the rewrite rules modelling the step of the protocol in which p_1 accepts the value v of the data item d as originated from p_2 for the purpose of authentication with p_1 . In our example this extension amounts to adding fact terms $\mathbf{wrequest}(\mathbf{A}, \mathbf{B}, \mathbf{nb}, \mathbf{Nb})$ and $\mathbf{wrequest}(\mathbf{B}, \mathbf{A}, \mathbf{na}, \mathbf{Na})$ to the right hand sides of the rewrite rules (6) and (7) respectively. Thus, any

state in which $\mathbf{wrequest}(p_1, p_2, d, v)$ holds and the corresponding witness fact (i.e. $\mathbf{witness}(p_2, p_1, d, v)$) does not hold represents a violation of the authentication property and we concisely represent this set of bad states with the formula $\mathcal{G} = \mathbf{wrequest}(p_1, p_2, d, v) \wedge \neg \mathbf{witness}(p_2, p_1, d, v)$ and by the set constraints $\mathcal{C} = \{p_2 \neq i\}$.

It is easy to see that weak authentication does not guarantee protection against replay attacks. To rectify this problem we must ensure that there exists a one-one relationship between the sessions of p_1 and the sessions of p_2 . We can specify a stronger form of authentication, called *strong authentication* in this paper and *injective agreement* in [50], by augmenting the request fact with a fifth parameter identifying the session and by setting $\mathcal{G} = (\mathbf{request}(p_1, p_2, d, v, s) \wedge \neg \mathbf{witness}(p_2, p_1, d, v)) \vee (\mathbf{request}(p_1, p_2, d, v, s_1) \wedge \mathbf{request}(p_1, p_2, d, v, s_2))$ and $\mathcal{C} = \{p_2 \neq i, s_1 \neq s_2\}$. Basically, besides the satisfaction of weak authentication, strong authentication requires that no principal should accept the same value twice from the same communication partner in two different sessions.

The definition of protocol insecurity problem is very general and allows for the specification of a wide variety of application scenarios, including those in which an unbounded number of sessions are allowed (e.g., see Example 1), infinitely-many fresh terms are generated, the number of steps that an honest agent can perform in each session is not fixed in advance, and/or infinitely many messages are exchanged by honest agents.

Example 1 The protocol insecurity problem associated with the NSPK protocol that we presented in Section 3.2 consists of two sessions only: one between agent **a** and the intruder with session identifier 1 (cf. fact (1) in the initial state) and one between agents **a** and **b** with session identifier 2 (cf. fact (2) in the initial state). It is possible to add the ability to generate an unbounded number of sessions, each involving two new agents, by extending the protocol insecurity problem with the following rewrite rule:

$$\begin{array}{l} \xrightarrow{\mathbf{new_session}(X, Y, KX, KY, NS)} \\ \exists X. Y. KX. KY. NS. \\ \mathbf{state}(0, X, X, [X, Y, KX, KX^{-1}, KY], NS) \\ \cdot \mathbf{state}(1, X, Y, [Y, X, KY, KY^{-1}, KX], NS) \end{array}$$

Notice that this rewrite rule has an empty left-hand-side and therefore it can be applied in any state. For each execution of the rewrite rule two new agents (X and Y), their public (KX, KY) and private keys (KX^{-1}, KY^{-1}), and a session identifier (NS) are freshly generated. These values define a new session of the NSPK protocol played by agents X and Y .

Not surprisingly, because of this expressiveness solving a protocol insecurity problem is undecidable in the general case. (As shown in [29] the Post Correspondence

Problem can be recast as a protocol insecurity problem.) However, as shown in [51], in some cases it is possible to reduce the general protocol insecurity problem to a protocol insecurity problem for the same protocol but comprising a small number of acyclic sessions (i.e. it can be reduced to a BNAS problem). Moreover violations of the security properties often exploit only a small number of sessions. (This claim is substantiated by the empirical observation that all flawed protocols in the Clark-Jacob's library [27] suffers from attacks that exploit at most two sessions.)

Since BNAS problems are an important, decidable class of protocol insecurity problems we now provide a syntactic characterisation of this class of problems inspired by the definition of *bounded role theory* given in [34]. Let Ξ be a protocol insecurity problem. We say that Ξ is a *protocol insecurity problem with a bounded number of acyclic sessions* (*BNAS problem*, for short) if, and only if, there exists an anti-reflexive and transitive relation \prec over a finite set $\{S_0, S_1, \dots, S_t\}$ of **state** facts such that (i) if $S_i \prec S_j$, then $S_i\sigma \prec S_j\sigma$ for each substitution σ and (ii) for each rewrite rule $(L \xrightarrow{r(\mathbf{x})} \exists \mathbf{y}.R)$ of an honest agent there is exactly one occurrence of a **state** fact in L and R , say S_i and S_j resp., and it must be the case that $S_i \prec S_j$. The definition basically states that in each execution path the number of applications of the rules of the honest agents is finite and that each rule instance can be applied at most once. This justifies the name BNAS since it guarantees that the number of sessions associated to a BNAS problem is finite and that the sessions are acyclic. Notice that the number of sessions (in general defined as the number of different session identifiers occurring in the state facts of the reachable states) can be easily computed for BNAS problems by a static inspection of the input specification. (This usually amounts to counting the number of different session identifiers occurring in the state facts of the initial state.)

It is immediate to see that the protocol insecurity problem previously specified for the NSPK protocol is a BNAS problem with two sessions. To show this it suffices to collect the **state** facts occurring in the rules of Figure 3 and define $\mathbf{state}(i, _, A, _, S) \prec \mathbf{state}(j, _, A, _, S)$ if, and only if, $i < j$. Moreover, since the application of every rule of Figure 3 just advances the state of an agent by leaving him in the same session he was involved in, then the number of sessions is given by the number of different session identifiers (namely 1 and 2) occurring in the state facts of the initial state.

3.5 Optimising Transformations on Protocol Insecurity Problems

The notion of protocol insecurity problem we have outlined so far is accurate but not adequate to carry out automatic analysis in an efficient way. The main problem rests with the specification of the intruder that allows to

forge messages that cannot be possibly used to mount an attack on the protocol. This can be avoided by providing a refined model of the intruder that forges a message only if it can be accepted by a honest participant. Moreover it is possible to drastically reduce the number of transitions (while retaining all the possible attacks on the protocol) by merging some intruder rules with protocols steps carried out by the honest participants. We now discuss in more details these optimisations.

Step compression. An effective optimisation, called *step compression* has been proposed in [12]. The key idea is to replace the rewrite rules of the honest agents and those modelling the ability of the intruder to fake and divert messages with a new set of rewrite rules obtained by merging these rewrite rules. This amounts to changing the model of the problem by regarding the intruder and the network as a single entity and by assuming that (i) every message sent by an honest agent is received by the intruder and that (ii) every message received by a honest agents has been generated by the intruder. The step compression optimisation removes the intruder rules (8) and (15) and replaces every rule of the honest agents, i.e.,

$$\begin{aligned} & \mathbf{msg}(j, s, r, m) \cdot \mathbf{state}(j, s, r, ms, c) \xrightarrow{r(\mathbf{x})} \\ & \exists \mathbf{y}. \mathbf{msg}(j', r, r', m') \cdot \mathbf{state}(j'', s', r, ms', c) \end{aligned} \quad (16)$$

with

$$\begin{aligned} & \mathbf{ik}(m) \cdot \mathbf{state}(j, s, r, ms, c) \xrightarrow{\mathbf{sc}.r(\mathbf{x})} \\ & \exists \mathbf{y}. \mathbf{ik}(m) \cdot \mathbf{ik}(m') \cdot \mathbf{state}(j'', s', r, ms', c) \end{aligned} \quad (17)$$

Notice that the above transformation has to be slightly adapted for those rules that do not have a message fact in either the left hand side or the right hand side (cf. rules (4) and (7) of Figure 3). More specifically: when $\mathbf{msg}(j, s, r, m)$ does not occur in the left hand side of (16), then $\mathbf{ik}(m)$ will not occur in (17); when $\mathbf{msg}(j', r, r', m')$ does not occur in the right hand side of (16), then $\mathbf{ik}(m)$ will not occur in the right hand side of (17). We say that m is the *reception pattern* of the rule. For instance, the step compressed rewrite rule corresponding to (6) is:

$$\begin{aligned} & \mathbf{ik}(\{\langle \text{Na}, \text{Nb} \rangle_{\text{Ka}}\}) \cdot \mathbf{state}(2, \text{B}, \text{A}, [\text{Na}, \text{A}, \text{B}, \text{Ka}, \text{Ka}^{-1}, \text{Kb}], \text{S}) \\ & \xrightarrow{\mathbf{sc_step}_3(\text{A}, \text{B}, \text{Ka}, \text{Kb}, \text{Na}, \text{Nb}, \text{S})} \\ & \mathbf{ik}(\{\langle \text{Na}, \text{Nb} \rangle_{\text{Ka}}\}) \cdot \mathbf{ik}(\{\langle \text{Nb} \rangle_{\text{Kb}}\}) \cdot \\ & \mathbf{state}(4, \text{B}, \text{A}, [\text{Nb}, \text{Na}, \text{A}, \text{B}, \text{Ka}, \text{Ka}^{-1}, \text{Kb}], \text{S}) \end{aligned} \quad (18)$$

Specialisation of the composition rules. Let us assume that the step compression optimisation has been applied to the input protocol insecurity problem. The intuition behind our second optimisation is to build in the left hand side of the rewrite rules of the honest agents the ability to compose all the possible messages matching their reception pattern. This gives us a new version of these

rules and allows us to remove the composition rules of the intruder, namely (9), (10), and (11). The rationale for this optimising transformation rests on the following fact borrowed from [52]: we can safely restrict our attention to those intruder knowledge derivations in which all the decomposition rules are applied before all the composition rules.

Let $m \in T_{\text{MSG}}(\Sigma, V)$, \mathcal{M} be a set of subsets of $T_{\text{MSG}}(\Sigma, V)$, and $\xrightarrow{c^*}$ ($\xrightarrow{d^*}$) be the reflexive and transitive closure of the rewrite relation defined by the rewrite rules (9), (10), and (11) ((12), (13), and (14), resp.). If $M = \{m_1, \dots, m_n\} \subseteq T_{\text{MSG}}(\Sigma, V)$, then $\text{ik}(M)$ abbreviates $\text{ik}(m_1) \dots \text{ik}(m_n)$. We say that \mathcal{M} is a generator of m if and only if for each grounding substitution σ and for all $M_0 \subseteq T_{\text{MSG}}(\Sigma, V)$ such that $\text{ik}(M_0) \xrightarrow{c^*} \text{ik}(m\sigma)$ there exists $M \subseteq T_{\text{MSG}}(\Sigma, V)$ such that $\text{ik}(M) \in \mathcal{M}$ and $\text{ik}(M_0) \xrightarrow{d^*} \text{ik}(M\sigma) \xrightarrow{c^*} \text{ik}(m\sigma)$. For instance, both $\mathcal{M}_1 = \{\text{ik}(\{\langle \text{Na}, \text{Nb} \rangle_{\text{Ka}}\}), \text{ik}(\text{Na}) \cdot \text{ik}(\text{Nb}) \cdot \text{ik}(\text{Ka})\}$ and $\mathcal{M}_2 = \{\text{ik}(\{\langle \text{Na}, \text{Nb} \rangle_{\text{Ka}}\}), \text{ik}(\text{Na}) \cdot \text{ik}(\text{Nb}) \cdot \text{ik}(\text{Ka}), \text{ik}(\langle \text{Na}, \text{Nb} \rangle \cdot \text{ik}(\text{Ka}))\}$ are generators of $\{\langle \text{Na}, \text{Nb} \rangle_{\text{Ka}}\}$, while $\mathcal{M}_3 = \{\text{ik}(\text{Na}) \cdot \text{ik}(\text{Nb}) \cdot \text{ik}(\text{Ka})\}$ is not, as $\text{ik}(\{\langle \text{na}, \text{nb} \rangle_{\text{ka}}\}) \xrightarrow{c^*} \text{ik}(\{\langle \text{na}, \text{nb} \rangle_{\text{ka}}\})$ but $\text{ik}(\{\langle \text{na}, \text{nb} \rangle_{\text{ka}}\}) \not\xrightarrow{d^*} \text{ik}(\text{na}) \cdot \text{ik}(\text{nb}) \cdot \text{ik}(\text{ka})$.

The optimisation amounts to removing all the composition rules of the intruder and then to replacing each step compressed rewrite rule of the form (17) by a new set, say $\mathcal{RC}(\text{sc}_r)$, of rewrite rules

$$\text{ik}(M) \cdot \text{state}(j, s, r, ms, c) \xrightarrow{\text{sc}_r^t(\mathbf{x})} \exists \mathbf{y}. \text{ik}(M) \cdot \text{ik}(m') \cdot \text{state}(j'', s', r, ms', c)$$

for each $\text{ik}(M) \in \mathcal{M}$, where \mathcal{M} is a generator of the reception pattern m . Notice that each path of the resulting protocol insecurity problem contains at most one application of the rules in $\mathcal{RC}(\text{sc}_r)$. In fact, all the rules in $\mathcal{RC}(\text{sc}_r)$ have the same state fact in the left hand side, namely $\text{state}(j, s, r, ms, c)$, and this very same fact is deleted from the state by applying any such rule.

A simple way to compute a generator of m of a given reception pattern m is given by $\mathcal{M} = \{\text{ik}(M) \mid M \subseteq T_{\text{MSG}}(\Sigma, V) \text{ and } \text{ik}(m) \rightarrow^* \text{ik}(M)\}$, where \rightarrow^* is the reflexive and transitive closure of the rewrite relation defined by the rewrite rules of Figure 5. (Notice that these rewrite rules model the backward representation of the composition rules of the intruder.) It must be noted that in the worst case, i.e. when the tree representation of m is a complete binary tree with n nodes all labelled by the pairing operator, the cardinality of \mathcal{M} is exponential in n . (This follows from (i) the observation that there exists a bijection between the elements of \mathcal{M} and the set of strongly binary trees of depth $d = \log_2 n$ and (ii) the fact that the number of strongly binary trees of depth d is equal to c^{2^d} , where $c \simeq 1.5$ [2].) However a dramatic reduction of the cardinality of \mathcal{M} may be obtained by dropping from \mathcal{M} all the sets containing messages whose top-level function symbol is a pairing operator. It can be

shown that the set simplified in this way is still a generator of m . For instance, if $m = \langle \langle \text{A}, \text{B} \rangle, \{\langle \langle \text{A}, \text{B} \rangle, \text{N} \rangle_{\langle \text{Ka}, \text{Kb} \rangle} \rangle$, then $|\mathcal{M}| = 15$ but the above simplification leaves us with a generator comprising two elements only, namely $\text{ik}(\text{A}) \cdot \text{ik}(\text{B}) \cdot \text{ik}(\text{A}) \cdot \text{ik}(\text{B}) \cdot \text{ik}(\text{N}) \cdot \text{ik}(\text{Ka}) \cdot \text{ik}(\text{Kb})$ and $\text{ik}(\text{A}) \cdot \text{ik}(\text{B}) \cdot \text{ik}(\langle \langle \text{A}, \text{B} \rangle, \text{N} \rangle_{\langle \text{Ka}, \text{Kb} \rangle})$.

Let us now consider the application of the optimisation to the rewrite rule (18). The reception pattern is $m = \{\langle \text{Na}, \text{Nb} \rangle_{\text{Ka}}\}$. By applying the above method, we obtain $\mathcal{M}_1 = \{\text{ik}(\{\langle \text{Na}, \text{Nb} \rangle_{\text{Ka}}\}), \text{ik}(\text{Na}) \cdot \text{ik}(\text{Nb}) \cdot \text{ik}(\text{Ka})\}$ as generator of m and thus we can generate the following two specialised rewrite rules:

$$\begin{aligned} & \text{ik}(\{\langle \text{Na}, \text{Nb} \rangle_{\text{Ka}}\}) \cdot \text{state}(2, \text{B}, \text{A}, [\text{Na}, \text{A}, \text{B}, \text{Ka}, \text{Ka}^{-1}, \text{Kb}], \text{S}) \\ & \xrightarrow{\text{sc_step}_3^1(\text{A}, \text{B}, \text{Ka}, \text{Kb}, \text{Na}, \text{Nb}, \text{S})} \\ & \text{ik}(\{\langle \text{Na}, \text{Nb} \rangle_{\text{Ka}}\}) \cdot \text{ik}(\{\text{Nb}\}_{\text{Kb}}) \\ & \cdot \text{state}(4, \text{B}, \text{A}, [\text{Nb}, \text{Na}, \text{A}, \text{B}, \text{Ka}, \text{Ka}^{-1}, \text{Kb}], \text{S}) \\ \\ & \text{ik}(\text{Na}) \cdot \text{ik}(\text{Nb}) \cdot \text{ik}(\text{Ka}) \\ & \cdot \text{state}(2, \text{B}, \text{A}, [\text{Na}, \text{A}, \text{B}, \text{Ka}, \text{Ka}^{-1}, \text{Kb}], \text{S}) \\ & \xrightarrow{\text{sc_step}_3^2(\text{A}, \text{B}, \text{Ka}, \text{Kb}, \text{Na}, \text{Nb}, \text{S})} \\ & \text{ik}(\text{Na}) \cdot \text{ik}(\text{Nb}) \cdot \text{ik}(\text{Ka}) \cdot \text{ik}(\{\text{Nb}\}_{\text{Kb}}) \\ & \cdot \text{state}(4, \text{B}, \text{A}, [\text{Nb}, \text{Na}, \text{A}, \text{B}, \text{Ka}, \text{Ka}^{-1}, \text{Kb}], \text{S}) \end{aligned}$$

Since each rewrite rule is replaced by $|\mathcal{M}|$ new rewrite rules, for the effectiveness of the optimisation it is necessary that the generators have finite cardinality. For this reason we apply this optimisation only to those problems in which the size of all the ground instances of the reception patterns is bounded. Although this requirement is apparently quite restrictive, many complex protocols (e.g. Kerberos, AAA Mobile IP, and Fair ZG) meet this condition provided that their implementation adds enough redundancy to the messages to allow the agents to check the well-formedness of the received messages. (A tagging mechanism that achieves this is described in [43].) The following example illustrates the point.

Example 2 Let us consider the following protocol fragment where a server S generates and then secretly distributes a nonce N to agents A and B . Secrecy is ensured by encrypting the messages with the public keys of the intended recipients.

$$\begin{aligned} (1) \quad S & \rightarrow A : \{S, A, B, N\}_{K_a}, \{S, A, B, N\}_{K_b} \\ (2) \quad A & \rightarrow B : \{S, A, B, N\}_{K_b} \end{aligned}$$

Notice that since A does not know B 's private key, she cannot decrypt $\{S, A, B, N\}_{K_b}$ and then she cannot check its structure and content. She will thus accept any message comprising two components and will forward the second component provided that the first component has the expected content. This behaviour can be modelled by the two (step compressed) rewrite rules of the following

$$\begin{aligned} \text{ik}(\langle M_1, M_2 \rangle) &\rightarrow \text{ik}(M_1) \cdot \text{ik}(M_2) \\ \text{ik}(\{M_1\}_{M_2}) &\rightarrow \text{ik}(M_1) \cdot \text{ik}(M_2) \\ \text{ik}(\{M_1\}_{M_2}^s) &\rightarrow \text{ik}(M_1) \cdot \text{ik}(M_2) \end{aligned}$$

Fig. 5 Rewrite rules for specialised composition

form:

$$\begin{aligned} \dots &\xrightarrow{\text{step}_1(S, A, B, \dots)} \exists N. \text{ik}(\langle \{S, A, B, N\}_{\text{Ka}}, \{S, A, B, N\}_{\text{Kb}} \rangle) \dots \\ \text{ik}(\langle \{S, A, B, N\}_{\text{Ka}}, X \rangle) \cdot \text{state}(1, S, A, [\dots], -) & \\ \xrightarrow{\text{step}_2(A, B, S, \text{Ka}, N, X, \dots)} &\text{ik}(X) \dots \end{aligned} \quad (19)$$

where X is a variable of sort MSG. Since the size of the terms of sort MSG is unbounded, this protocol falls outside the scope of our optimisation. However, if we assume that A can check the well-formedness of the messages she receives, then we can replace rule (19) with the following, less liberal, rule:

$$\begin{aligned} \text{ik}(\langle \{S, A, B, N\}_{\text{Ka}}, \{S', A', B', N'\}_{\text{Kb}'} \rangle) \cdot \text{state}(1, S, A, [\dots], -) & \\ \xrightarrow{\text{step}_2(A, B, S, \text{Ka}, N, S', A', B', N', \text{Kb}', \dots)} &\text{ik}(\{S', A', B', N'\}_{\text{Kb}'} \dots \end{aligned}$$

where the size of all the ground instances of the reception pattern is bounded and thus our optimisation becomes applicable.

4 Reducing Bounded Protocol Insecurity Problems to SAT

We now focus on the problem of reducing of a bounded protocol insecurity problem to SAT. Given a protocol insecurity problem Ξ and an integer $k \geq 0$ we build a propositional formula $\llbracket \Xi \rrbracket_k$ whose models correspond (in a sense that will be defined shortly) to the partial order attacks of Ξ of length k .

Preliminarily to this we show how a protocol insecurity problem Ξ can be turned into an quantifier-free, attack-equivalent protocol insecurity problem $\tilde{\Xi}$. (We say that $\tilde{\Xi}$ is *attack-equivalent* to Ξ if, and only if, there exists a one-to-one mapping between the solutions of Ξ and those of $\tilde{\Xi}$.) How this can be done will be discussed in Section 4.1. In Section 4.2 we introduce the notion of partial order solution of protocol security problems that will play a fundamental role in stating and proving the fundamental properties of the encoding techniques that we present in Section 4.3.

4.1 Quantifier Elimination

A multi-set rewriting system $\Gamma = \langle \mathcal{I}, \mathcal{R} \rangle$ is *quantifier-free* if, and only if, for all $(L \xrightarrow{r(\mathbf{x})} \exists \mathbf{y}. R) \in \mathcal{R}$ we have that $\mathbf{y} = \emptyset$.

A first, general approach to quantifier elimination amounts to extending the state of the system with a counter whose value is used and incremented every time a fresh term is built. More in detail, a fact of the form $c(0)$ is added to the initial state of the system and then every rule $(L \xrightarrow{r(\mathbf{x})} \exists \mathbf{y}. R)$ where $\mathbf{y} = \langle y_1, \dots, y_n \rangle$ is replaced with

$$c(z) \cdot L \xrightarrow{r(\mathbf{x}, z)} c(\mathbf{s}(z)) \cdot R'$$

where z is a variable of sort NUM that does not occur in R and R' is obtained from R by substituting every occurrence of y_i with $f_i(z)$ where f_i is a new function symbol (of the appropriate arity) for $i = 1, \dots, n$. Intuitively the term $f_i(z)$ denotes the fresh value generated by the execution of the rule and associated with the existentially quantified variable y_i . For example, the rewrite rule (4) is transformed into:

$$\begin{aligned} c(C) \cdot \text{state}(0, A, A, [A, B, \text{Ka}, \text{Ka}^{-1}, \text{Kb}], S) & \\ \xrightarrow{\text{step}_1(A, B, C, \text{Ka}, \text{Kb}, S)} & \\ c(\mathbf{s}(C)) \cdot \text{state}(2, B, A, [f_1(C), A, B, \text{Ka}, \text{Ka}^{-1}, \text{Kb}], S) & \\ \cdot \text{msg}(1, A, B, \{\langle A, f_1(C) \rangle\}_{\text{Kb}}) & \end{aligned}$$

It is easy to see that this simple transformation guarantees the uniqueness of fresh terms and hence that the resulting protocol insecurity problems is attack-equivalent to the original one.

An alternative approach to quantifier elimination, which can be applied effectively to BNAS protocol insecurity problems, amounts to determining in advance a set of ground terms U representing all fresh values that can possibly be generated during execution and transforming the protocol insecurity problem in the following way. (Notice that for BNAS protocol insecurity problems the set U has necessarily finite size.) The initial state is extended with a fact of the form $\text{fresh}(t)$ (stating that the value denoted by t has not been used before) for each $t \in U$. Every rule of the form $(L \cdot \text{state}(j, s, r, w, c) \xrightarrow{\rho} \exists \mathbf{y}. R)$, where $\mathbf{y} = \langle y_1, \dots, y_n \rangle$ and w is a term representing the messages known by agent r , is then replaced with

$$L \cdot \text{state}(j, s, r, w, c) \cdot \text{fresh}(t_1) \cdot \dots \cdot \text{fresh}(t_n) \xrightarrow{\rho} R'$$

where $t_i = f_i(r, j, c)$ (with f_i a new function symbol of the appropriate arity) and R' is obtained from R by substituting y_i with t_i for $i = 1, \dots, n$. For instance, the rewrite rule (4) becomes:

$$\begin{aligned} \text{state}(0, A, A, [A, B, \text{Ka}, \text{Ka}^{-1}, \text{Kb}], S) \cdot \text{fresh}(f_1(A, 0, S)) & \\ \xrightarrow{\text{step}_1(A, B, \text{Ka}, \text{Kb}, S)} & \\ \cdot \text{state}(2, B, A, [f_1(A, 0, S), A, B, \text{Ka}, \text{Ka}^{-1}, \text{Kb}], S) & \\ \cdot \text{msg}(1, A, B, \{\langle A, f_1(A, 0, S) \rangle\}_{\text{Kb}}) & \end{aligned}$$

Notice that, as a result of the application of an instance of the above rule, the facts occurring on the left hand side

will be removed from the state and therefore the same rule instance will not be applicable in the subsequent steps.

4.2 Partial Order Solutions

Even though we are assuming an interleaving semantics for protocol insecurity problems, considering the parallel execution of rewrite rules may have significant advantages. In fact the parallel execution of two (or more) rules often leads to a state which would be anyway reachable by applying the same rules sequentially in some order. To illustrate, consider the situation in which you want to execute the rewrite rules in $\{\text{off}_i \xrightarrow{\text{act}_i} \text{on}_i : i = 1, \dots, n\}$ so to activate n alarm devices through n different switches. It is immediate to see that the parallel execution of $\text{act}_1, \dots, \text{act}_n$ leads (in a single step) to the same state obtained by applying the rules in sequence and in any order. Thus by enabling parallelism we may obtain shorter solutions and, as it will be clear in the next section, smaller encodings. However, in some cases the parallel execution of two (or more) rules leads to a state which is not reachable by applying the same rules sequentially in any order. For instance, the parallel execution of the rewrite rules $\mathbf{a} \xrightarrow{\mathbf{r}_1} \mathbf{b} \cdot \mathbf{c}$ and $\mathbf{b} \xrightarrow{\mathbf{r}_2} \mathbf{a} \cdot \mathbf{c}$ corresponds to the rule $\mathbf{a} \cdot \mathbf{b} \xrightarrow{\mathbf{r}_1 \parallel \mathbf{r}_2} \mathbf{a} \cdot \mathbf{b} \cdot \mathbf{c}$ whose behaviour cannot be reproduced by the sequential application of \mathbf{r}_1 and \mathbf{r}_2 in any order.

We are thus interested in enabling a restricted form of parallelism which ensures that all the parallel executions of the rules lead to a state which is reachable by applying the same individual rules sequentially in some order.

For the sake of simplicity from here on we focus on *ground* reachability problems, i.e. reachability problems of a given signature $\langle \mathcal{S}, \Sigma \rangle$ over the empty set of variables. This is without loss of generality as any given quantifier-free protocol insecurity problem $\Xi = \langle \Gamma, \langle \mathcal{G}, \mathcal{C} \rangle \rangle$ can be thought as a concise representation of the protocol insecurity problem obtained from Ξ by replacing (i) the rewrite rules of Γ with the corresponding ground instances, (ii) \mathcal{G} with the disjunction of all its instances satisfying the constraints in \mathcal{C} , and (iii) \mathcal{C} with the empty set. Moreover, if $(L \xrightarrow{\rho} R) \in \mathcal{R}$, then we define $\text{pre}(\rho) = L$, $\text{add}(\rho) = R$, and $\text{del}(\rho) = (L \setminus R)$ to be the *preconditions*, the *positive effects*, and the *negative effects* of ρ respectively.

Definition 1 (Parallel Composition) Let $\rho_i \in \text{dom}(\mathcal{R})$ for $i = 1, \dots, n$. We say that ρ_1, \dots, ρ_n are *composable* if and only if $\text{add}(\rho_i) \cap \text{del}(\rho_j) = \emptyset$ for all $i, j = 1, \dots, n$ such that $i \neq j$, i.e. the effects of the rules in ρ_1, \dots, ρ_n do not contradict each other. If ρ_1, \dots, ρ_n are composable rules, then the (*parallel*) *composition* of ρ_1, \dots, ρ_n , in symbols $\rho_1 \parallel \dots \parallel \rho_n$, is the rule label ρ associated with the rewrite rule $\bigcup_{i=1}^n L_i \xrightarrow{\rho} \bigcup_{i=1}^n R_i$.

If the parallel execution of a set of composable rules leads to a state which is reachable by applying the same rules sequentially in some order, then we say that this set of rule is *linearisable*.

Definition 2 (Linearisation) Let ρ_1, \dots, ρ_n be composable rules. A *linearisation* of $\rho_1 \parallel \dots \parallel \rho_n$ is a permutation $\rho = \rho_{k_1}, \dots, \rho_{k_n}$ of ρ_1, \dots, ρ_n such that for all $S_0 \supseteq \bigcup_{i=1}^n \text{pre}(\rho_i)$ there exists an initialised path $\pi_{\rho_1, \dots, \rho_n}(S_0)$. We say that $\rho_1 \parallel \dots \parallel \rho_n$ is *linearisable* if and only if there exists a linearisation of $\rho_1 \parallel \dots \parallel \rho_n$.

Lemma 2 Let ρ_1, \dots, ρ_n be composable rules, let $\rho_{k_1}, \dots, \rho_{k_n}$ be a linearisation of $\rho_1 \parallel \dots \parallel \rho_n$, and let $S_0 \supseteq \bigcup_{i=1}^n \text{pre}(\rho_i)$, then $\text{app}_{\rho_{k_1}; \dots; \rho_{k_n}}(S_0) = \text{app}_{\rho_1 \parallel \dots \parallel \rho_n}(S_0)$.

Let $\pi = S_0 \rho_1 S_1 \dots S_{m-1} \rho_m S_m$ and $\pi' = S'_0 \rho'_1 S'_1 \dots S'_{n-1} \rho'_n S'_n$ be initialised paths such that $S'_0 = S_m$. (From here on we write $S_0 \rho_1 S_1 \dots S_{m-1} \rho_m S_m$ in place of $S_0 \langle \rho_1, \emptyset \rangle S_1 \dots S_{n-1} \langle \rho_n, \emptyset \rangle S_n$.) The *concatenation* of π and π' , in symbols $\pi \pi'$, is the initialised path $S_0 \rho_1 S_1 \dots S_{m-1} \rho_m S_m \rho'_1 S'_1 \dots S'_{n-1} \rho'_n S'_n$. The following result easily follows from Lemma 2.

Lemma 3 Let $\Gamma = \langle \mathcal{I}, \mathcal{R} \rangle$ and $\hat{\Gamma} = \langle \mathcal{I}, \hat{\mathcal{R}} \rangle$, where $\hat{\mathcal{R}}$ is obtained from \mathcal{R} by adding all linearisable compound rules. If $S_0 \rho_1 S_1 \dots S_{m-1} \rho_m S_m$ is an initialised path of $\hat{\Gamma}$, then $\pi_{\bar{\rho}_1}(S_0) \pi_{\bar{\rho}_2}(S_1) \dots \pi_{\bar{\rho}_m}(S_m)$ is an initialised path of Γ where $\bar{\rho}_1, \dots, \bar{\rho}_m$ are linearisations of ρ_1, \dots, ρ_m respectively, and we say that the initialised path $S_0 \rho_1 S_1 \dots S_{m-1} \rho_m S_m$ is *linearisable*.

It is thus important to identify conditions that guarantee the linearisability of parallel paths. A simple approach is based on the idea of requiring that all component rules must be executed in any order and must lead to the same result.

Definition 3 (Interference) We say that ρ_1 *interferes* with ρ_2 , in symbols $\rho_1 \oplus \rho_2$, if and only if (i) $\rho_1 \neq \rho_2$ and (ii) $\text{del}(\rho_1) \cap \text{pre}(\rho_2) \neq \emptyset$ or $\text{del}(\rho_2) \cap \text{pre}(\rho_1) \neq \emptyset$, i.e. ρ_1 is different from ρ_2 and the execution of ρ_1 prevents the execution of ρ_2 or vice versa.

Lemma 4 Let ρ_1, \dots, ρ_n be composable rules. If ρ_i does not interfere with ρ_j for $i, j = 1, \dots, n$, then all permutations of ρ_1, \dots, ρ_n are linearisations of $\rho_1 \parallel \dots \parallel \rho_n$.

The following result readily follows from Lemma 2 and Lemma 4.

Theorem 2 Let $\Gamma = \langle \mathcal{I}, \mathcal{R} \rangle$ and let $\hat{\Gamma} = \langle \mathcal{I}, \hat{\mathcal{R}} \rangle$ where $\hat{\mathcal{R}}$ is obtained from \mathcal{R} by adding all the compound rules $\rho_1 \parallel \dots \parallel \rho_n$ for all pairwise non-interfering set of rules $\{\rho_1, \dots, \rho_n\} \subseteq \mathcal{R}$. If $S_0 \rho_1 S_1 \dots S_{m-1} \rho_m S_m$ is an initialised path of $\hat{\Gamma}$, then $\pi_{\bar{\rho}_1}(S_0) \pi_{\bar{\rho}_2}(S_1) \dots \pi_{\bar{\rho}_m}(S_m)$ is an initialised path of Γ where $\bar{\rho}_1, \dots, \bar{\rho}_m$ are linearisations of ρ_1, \dots, ρ_m respectively, and we say that $S_0 \rho_1 S_1 \dots S_{m-1} \rho_m S_m$ is a *partial-order path* of Γ .

An immediate consequence of Theorem 2 is that any solution to $\hat{\Xi} = \langle \hat{\Gamma}, \mathcal{B} \rangle$ compactly represents a finite set of solutions to $\Xi = \langle \Gamma, \mathcal{B} \rangle$ and it is thus referred to as a *partial-order solution* of Ξ . Moreover, since by definition the paths of $\hat{\Gamma}$ are a superset of those of Γ , then for each solution to Ξ there exists a corresponding (possibly shorter) solution to $\hat{\Xi}$ and vice versa.

4.3 Reducing Bounded Quantifier-free Protocol Insecurity Problems to SAT

Given a ground protocol insecurity problem $\Xi = \langle \Gamma, \mathcal{B} \rangle$ of signature $\langle \Sigma, \mathcal{S} \rangle$ and an integer $k > 0$, we now consider the problem of building a propositional formula $\llbracket \Xi \rrbracket_k$ such that every model of $\llbracket \Xi \rrbracket_k$ corresponds to a (linearisable) partial-order solution of Ξ of length k and vice versa. The starting point is to add a time-index to the rules and facts to indicate the state at which the rules apply or the facts hold. Facts are thus indexed by 0 through k and rules by 0 through $k-1$. If p is a fact or a rule and i is an index, then p^i is the corresponding time-indexed propositional variable. If $\mathbf{p} = p_1, \dots, p_n$ is a tuple of facts or rules and i is an index, then $\mathbf{p}^i = p_1^i, \dots, p_n^i$ is the corresponding time-indexed tuple of propositional variables. To simplify notation from here on we will abbreviate $T_{\text{FACT}}(\Sigma)$ and $T_{\text{RULE}}(\Sigma)$ with \mathcal{F} and \mathcal{L} respectively. Moreover, if $\Gamma = \langle \mathcal{I}, \mathcal{R} \rangle$, then $\hat{\Gamma} = \langle \mathcal{I}, \hat{\mathcal{R}} \rangle$ where $\hat{\mathcal{R}}$ is obtained from \mathcal{R} by adding all the compound rules $\rho_1 \parallel \dots \parallel \rho_n$ for all pairwise non-interfering set of rules $\{\rho_1, \dots, \rho_n\} \subseteq \mathcal{R}$. Similarly, if $\Xi = \langle \Gamma, \mathcal{B} \rangle$, then $\hat{\Xi} = \langle \hat{\Gamma}, \mathcal{B} \rangle$.

The formula $\llbracket \Xi \rrbracket_k$ is defined by

$$\llbracket \Xi \rrbracket_k = \llbracket \Gamma \rrbracket_k \wedge \llbracket \mathcal{B} \rrbracket_k \quad (20)$$

where $\llbracket \Gamma \rrbracket_k$ is a propositional formula encoding the behaviour of the (multi)-set rewriting system $\hat{\Gamma}$ and $\llbracket \mathcal{B} \rrbracket_k$ is a propositional formula encoding the set of bad states represented by \mathcal{B} and reachable in k steps. The propositional formula $\llbracket \Gamma \rrbracket_k$ is of the form:

$$\llbracket \Gamma \rrbracket_k = I(\mathbf{f}^0) \wedge \bigwedge_{i=0}^{k-1} T_i(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1}) \quad (21)$$

where \mathbf{f} and $\boldsymbol{\rho}$ are tuples of elements of \mathcal{F} and \mathcal{L} respectively and $\bigwedge_{i=0}^{k-1} T_i(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$ stands for the propositional constant *true*. The formula $I(\mathbf{f}^0)$ encodes the initial state whereas the formula $T_i(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$ encodes all the possible evolutions of the system from step i to step $i+1$. These formulae can be defined in a variety of ways but here we provide a high level characterisation of the encoding techniques in terms of a set of abstract requirements on $I(\mathbf{f}^0)$ and $T_i(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$.

(R1) $I(\mathbf{f}^0)$ is such that if M is a model of $I(\mathbf{f}^0)$ then $\{f \in \mathcal{F} : f^0 \in M\} \in \mathcal{I}$ and, conversely, if $S \in \mathcal{I}$ then $\{f^0 : f \in S\}$ is a model of $I(\mathbf{f}^0)$.

(R2) $T_i(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$ is such that for any $i = 0, \dots, k-1$

- (a) if M is a model of $T_i(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$ then the rules in $\Lambda = \{\rho \in \mathcal{L} : \rho^i \in M\}$ are composable and their parallel composition, say $\parallel \Lambda$, is linearisable and such that $\text{app}_{\parallel \Lambda}(S) = S'$ for $S = \{f \in \mathcal{F} : f^i \in M\}$ and $S' = \{f \in \mathcal{F} : f^{i+1} \in M\}$;
- (b) if $S \subseteq \mathcal{F}$ is reachable in i steps in $\hat{\Gamma}$, the set of rules in $\Lambda \subseteq \mathcal{L}$ is linearisable and $\parallel \Lambda$ is applicable in S , and $S' = \text{app}_{\parallel \Lambda}(S)$, then $M = \{f^i : f \in S\} \cup \{f^{i+1} : f \in S'\} \cup \{\rho^i : \rho \in \Lambda\}$ is a model of $T_i(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$.

The following two theorems state that any encoding satisfying the above requirements is both sound and complete (respectively).

Theorem 3 (Soundness) *If M is a model of $\llbracket \Gamma \rrbracket_k$ for $k \geq 0$, $S_i = \{f \in \mathcal{F} : f^i \in M\}$ for $i = 0, \dots, k$, and $\rho_i = \parallel \{\rho \in \mathcal{L} : \rho^i \in M\}$, for $i = 0, \dots, k-1$ then $S_0 \rho_0 S_1 \dots S_{k-1} \rho_{k-1} S_k$ is an initialised path of $\hat{\Gamma}$.*

Let M be a model of $\llbracket \Gamma \rrbracket_k$. We define $\pi(M) = S_0 \rho_0 S_1 \dots S_{k-1} \rho_{k-1} S_k$, where $S_i = \{f : f^i \in M\}$ for $i = 0, \dots, k$ and ρ_i is the parallel composition of the rules in $\{\rho : \rho^i \in M\}$ for $i = 0, \dots, k-1$.

Theorem 4 (Completeness) *If $\pi_k = S_0 \rho_0 S_1 \dots S_{k-1} \rho_{k-1} S_k$ is an initialised path of $\hat{\Gamma}$, then there exists a model M of $\llbracket \Gamma \rrbracket_k$ such that $\pi(M) = \pi_k$.*

We now present two approaches to building the formulae $I(\mathbf{f}^0)$, $T_i(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$, and $\llbracket \mathcal{B} \rrbracket_k$ adapted from two encoding techniques originally introduced for AI planning, namely the linear encoding and graphplan-based encoding [45,35].

4.3.1 The Linear Encoding

We assume that \mathcal{F} and \mathcal{L} have finite size. This is the case for BNAS problems for which it is possible to precompute these sets by looking at the signature of the language and at the rewrite rules of the protocol insecurity problem. Notice that although the cardinality of \mathcal{F} and \mathcal{L} is in the worst case exponential in the size of the original specification, the proper use of sort declarations and of the optimising transformations of Section 3.5 allows us to keep the size of \mathcal{F} and \mathcal{L} reasonably small for many protocols of interests.

The initial state is encoded by the formula:

$$I(\mathbf{f}^0) = \bigwedge \{f^0 : f \in \mathcal{I}\} \wedge \bigwedge \{\neg f^0 : f \in (\mathcal{F} \setminus \mathcal{I})\} \quad (22)$$

and the formula $T_i(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$, encoding the i -th step of the transition relation, is given by the conjunction of the following formulae:

Universal Axioms: for each $\rho \in \mathcal{L}$

$$\rho^i \supset \bigwedge \{f^i \mid f \in \text{pre}(\rho)\} \quad (23)$$

$$\rho^i \supset \bigwedge \{f^{i+1} \mid f \in \text{add}(\rho)\} \quad (24)$$

$$\rho^i \supset \bigwedge \{\neg f^{i+1} \mid f \in \text{del}(\rho)\} \quad (25)$$

Explanatory Frame Axioms: for all $f \in \mathcal{F}$

$$(f^i \wedge \neg f^{i+1}) \supset \bigvee \{\rho^i \mid \rho \in \mathcal{L}, f \in \text{del}(\rho)\} \quad (26)$$

$$(\neg f^i \wedge f^{i+1}) \supset \bigvee \{\rho^i \mid \rho \in \mathcal{L}, f \in \text{add}(\rho)\} \quad (27)$$

Conflict Exclusion Axioms (CEA): for all pairs of rules $\rho_1, \rho_2 \in \mathcal{L}$ such that $\rho_1 \oplus \rho_2$

$$\neg(\rho_1^i \wedge \rho_2^i) \quad (28)$$

Intuitively, the universal axioms state that for a rule ρ to be applicable at the time-step i its preconditions must hold at the same time-index—cf. formula (23)—and its effects must hold at the next time-index—cf. formulae (24) and (25) for positive and negative effects respectively. The explanatory frame axioms express that if a fact changes value from true to false (from false to true) between the time-indexes i and $i+1$, then one of the rules that deletes (adds, resp.) the fact must have been applied at time-index i —cf. formula (26) (formula (27), resp.). In other words, the explanatory frame axioms state that the rules are the only cause of change. The CEA, by imposing that two interfering rules cannot be executed at the same time, ensure that the interleaving semantics is respected. In the following example we describe a fragment of the encoding associated with our NSPK protocol insecurity problem.

Example 3 Let us consider the quantifier-free protocol insecurity problem resulting from the application of the second variant of the quantifier elimination method described in Section 4.1 and of the optimising transformations of Section 3.5. Besides others, the problem contains the following two rewrite rules:

$$\begin{aligned} & \text{ik}(\{\langle A, \text{Na} \rangle\}_{\text{Kb}}) \cdot \text{state}(1, A, B, [\text{B}, A, \text{Kb}, \text{Kb}^{-1}, \text{Ka}], S) \\ & \cdot \text{fresh}(f_2(B, 1, S)) \\ & \xrightarrow{\text{sc_step}_{2a}(A, B, \text{Ka}, \text{Kb}, \text{Na}, S)} \\ & \text{state}(3, A, B, [\text{Na}, f_2(B, 1, S), B, A, \text{Kb}, \text{Kb}^{-1}, \text{Ka}], S) \\ & \cdot \text{ik}(\{\langle \text{Na}, f_2(B, 1, S) \rangle\}_{\text{Ka}}) \cdot \text{witness}(B, A, \text{nb}, f_2(B, 1, S)) \\ & \cdot \text{ik}(\{\langle A, \text{Na} \rangle\}_{\text{Kb}}) \\ & \text{ik}(A) \cdot \text{ik}(B) \cdot \text{ik}(\text{Kb}) \cdot \text{state}(1, A, B, [\text{B}, A, \text{Kb}, \text{Kb}^{-1}, \text{Ka}], S) \\ & \cdot \text{fresh}(f_2(B, 1, S)) \\ & \xrightarrow{\text{sc_step}_{2b}(A, B, \text{Ka}, \text{Kb}, \text{Na}, S)} \\ & \text{state}(3, A, B, [\text{Na}, f_2(B, 1, S), B, A, \text{Kb}, \text{Kb}^{-1}, \text{Ka}], S) \\ & \cdot \text{ik}(\{\langle \text{Na}, f_2(B, 1, S) \rangle\}_{\text{Ka}}) \cdot \text{witness}(B, A, \text{nb}, f_2(B, 1, S)) \\ & \cdot \text{ik}(\{\langle A, \text{Na} \rangle\}_{\text{Kb}}) \end{aligned}$$

The universal axioms associated with the rule $\text{sc_step}_{2a}(a, b, \text{ka}, \text{kb}, f_1(a, 0, 2), 2)$ are as follow:

$$\begin{aligned} & \text{sc_step}_{2a}(a, b, \text{ka}, \text{kb}, f_1(a, 0, 2), 2)^i \supset \\ & \text{ik}(\{\langle a, f_1(a, 0, 2) \rangle\}_{\text{kb}})^i \wedge \text{fresh}(f_2(b, 1, 2))^i \wedge \\ & \text{state}(1, a, b, [b, a, \text{kb}, \text{kb}^{-1}, \text{ka}], 2)^i \quad (29) \end{aligned}$$

$$\begin{aligned} & \text{sc_step}_{2a}(a, b, \text{ka}, \text{kb}, f_1(a, 0, 2), 2)^i \supset \\ & \text{ik}(\{\langle f_1(a, 0, 2), f_2(b, 1, 2) \rangle\}_{\text{ka}})^{i+1} \wedge \\ & \text{witness}(b, a, \text{nb}, f_2(b, 1, 2))^{i+1} \wedge \\ & \text{state}(3, a, b, [f_1(a, 0, 2), f_2(b, 1, 2), b, a, \text{kb}, \text{kb}^{-1}, \text{ka}], 2)^{i+1} \quad (30) \end{aligned}$$

$$\begin{aligned} & \text{sc_step}_{2a}(a, b, \text{ka}, \text{kb}, f_1(a, 0, 2), 2)^i \supset \\ & \neg \text{fresh}(f_2(b, 1, 2))^{i+1} \wedge \\ & \neg \text{state}(1, a, b, [b, a, \text{kb}, \text{kb}^{-1}, \text{ka}], 2)^{i+1} \quad (31) \end{aligned}$$

where (29), (30), and (31) are instances of (23), (24), and (25) respectively.

The explanatory frame axioms associated with the fact $\text{fresh}(f_2(b, 1, 2))$ are:

$$\begin{aligned} & \text{fresh}(f_2(b, 1, 2))^i \wedge \neg \text{fresh}(f_2(b, 1, 2))^{i+1} \supset \\ & (\text{sc_step}_{2a}(a, b, \text{ka}, \text{kb}, f_1(a, 0, 2), 2) \vee \\ & \text{sc_step}_{2b}(a, b, \text{ka}, \text{kb}, f_1(a, 0, 2), 2)) \quad (32) \\ & \neg \text{fresh}(f_2(b, 1, 2))^i \wedge \text{fresh}(f_2(b, 1, 2))^{i+1} \supset \text{false} \quad (33) \end{aligned}$$

where (32) and (33) are instances of (26) and (27) respectively. Formula (32) states that there is no other way to change the value of $\text{fresh}(f_2(b, 1, 2))$ from true to false other than executing either $\text{sc_step}_{2a}(a, b, \text{ka}, \text{kb}, f_1(a, 0, 2), 2)$ or $\text{sc_step}_{2b}(a, b, \text{ka}, \text{kb}, f_1(a, 0, 2), 2)$, while (33) states that the freshness of $f_2(b, 1, 2)$ cannot be restored once it has been used.

The rules $\text{sc_step}_{2a}(a, b, \text{ka}, \text{kb}, f_1(a, 0, 2), 2)$ and $\text{sc_step}_{2b}(a, b, \text{ka}, \text{kb}, f_1(a, 0, 2), 2)$ are obviously interfering since the execution of one prevents the execution of the other. The following CEA is thus generated:

$$\begin{aligned} & \neg(\text{sc_step}_{2a}(a, b, \text{ka}, \text{kb}, f_1(a, 0, 2), 2)^i \wedge \\ & \text{sc_step}_{2b}(a, b, \text{ka}, \text{kb}, f_1(a, 0, 2), 2)^i) \quad (34) \end{aligned}$$

where (34) is an instance of (28).

Theorem 5 states the soundness and completeness of the linear encoding. The proof, given in Appendix, amounts to showing that $I(\mathbf{f}^0)$ enjoys requirement R1 and $T_i(\mathbf{f}^i, \rho^i, \mathbf{f}^{i+1})$ enjoys part (a) and part (b) of requirement R2. From this and from Theorem 3 it readily

follows that the formulae (22)-(28) faithfully encode the behaviours of \hat{F} of length k and hence the main result. Since the structure of the formula $T_i(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$ is independent from i , in the rest of this section we write $\overline{T}(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$ in place of $T_i(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$.

Since Ξ is ground, we assume that $\mathcal{B} = \langle \mathcal{G}, \emptyset \rangle$ (with \mathcal{G} ground) and we define $\llbracket \mathcal{B} \rrbracket_k = \mathcal{G}^k$ where \mathcal{G}^k is obtained from \mathcal{G} by replacing every fact p with p^k .

Theorem 5 (Soundness and Completeness of the Linear Encoding) *If M is a model of $\llbracket \Xi \rrbracket_k$ for $k \geq 0$, then $S_0 \rho_0 S_1 \cdots S_{k-1} \rho_{k-1} S_k$ is a (partial-order) solution of $\hat{\Xi}$, where $S_i = \{f \in \mathcal{F} : f^i \in M\}$ for $i = 0, \dots, k$ and ρ_i is the parallel composition of the rules in $\{\rho \in \mathcal{L} : \rho^i \in M\}$ for $i = 0, \dots, k-1$. Conversely, if $\pi_k = S_0 \rho_0 S_1 \cdots S_{k-1} \rho_{k-1} S_k$ is a (partial-order) solution of $\hat{\Xi}$, then there exists a model M of $\llbracket \Xi \rrbracket_k$ such that $\pi(M) = \pi_k$.*

It is immediate to see that the number of atoms in $\overline{T}(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$ is in $O(|\mathcal{F}| + |\mathcal{L}|)$, i.e. a propositional variable for each fact and rule to denote their truth-value at step i . Moreover the Universal Axioms lead to $(|\text{pre}(\rho)| + |\text{add}(\rho)| + |\text{del}(\rho)|)$ binary clauses for each rule $\rho \in \mathcal{L}$. As a consequence, the number of clauses in $\overline{T}(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$ associated with the Universal Axioms is in $O(P_0 |\mathcal{L}|)$ where $P_0 = \max_{\rho \in \mathcal{L}} (|\text{pre}(\rho)|, |\text{add}(\rho)|)$ is the maximal number of facts mentioned in one side of a rule, usually a small number, e.g. $P_0 = 4$ for our NSPK protocol insecurity problem. (Notice that since $\text{del}(\rho) = \text{pre}(\rho) \setminus \text{add}(\rho)$ we have that $|\text{del}(\rho)| \leq |\text{pre}(\rho)|$ for all $\rho \in \mathcal{L}$.) The number of clauses in $\overline{T}(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$ associated with the Explanatory Frame Axioms is in $O(|\mathcal{F}|)$ as two clauses are generated for each fact. Finally, the number of (binary) clauses in $\overline{T}(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$ associated with the CEA is in $O(|\mathcal{L}|^2)$, as in the worst case all rules are pairwise interfering. Thus, the overall number of clauses is in $\overline{T}(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$ is in $O(|\mathcal{F}| + |\mathcal{L}|^2)$.

The quadratic growth of the number of the CEA can produce encodings of unmanageable size when the considered security protocol involves the exchange of complex messages.

It is possible to dispense with the CEA by using a bitwise representation of the rules as proposed in [35]. This amounts to replacing the CEA with the Bitwise Rule Representation Axioms shown below. Let Q be a set of $\lceil \log_2 |\mathcal{L}| \rceil$ new propositional variables and let β be any injective function mapping every $\rho \in \mathcal{L}$ into a set of literals such that $q \in \beta(\rho)$ if and only if $(\neg q) \notin \beta(\rho)$ for all $q \in Q$ and all $\rho \in \mathcal{L}$. Intuitively $\beta(\rho)$ provides a binary representation of ρ in terms of the propositional variables in Q .

Bitwise Rule Representation Axioms: for each $\rho \in \mathcal{L}$:

$$\rho^i \equiv \bigwedge \beta(\rho)^i \quad (35)$$

By using this encoding the number of propositional atoms in $\overline{T}(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$ is still in $O(|\mathcal{F}| + |\mathcal{L}|)$ as $\lceil \log_2 |\mathcal{L}| \rceil$ is

dominated by $|\mathcal{L}|$, while the number of clauses is in $O(|\mathcal{F}| + |\mathcal{L}| \lceil \log_2 |\mathcal{L}| \rceil)$ as each axiom of the form (35) leads to $\lceil \log_2 |\mathcal{L}| \rceil + 1$ clauses. This is a considerable saving, but it must be noted that the bitwise rule representation axioms rule out all parallelism from the solutions as they impose that at most one rule is executed at a time. As a consequence, attacks are usually found at greater values of k than if the CEA are employed.

A further alternative is to use the iterative abstraction/refinement strategy we proposed in [8]. The basic idea is to avoid generating the CEA a priori. By doing this we may get spurious attacks, i.e. attacks that not executable. So, whenever a solution is found by the solver we check whether it is spurious or not. If it is not spurious, then the procedure halts reporting the attack. If the attack is spurious, then a set of pairs of interfering rules are extracted from the attack, a set of clauses excluding the simultaneous execution of interfering rules is added to the formula, and the whole procedure is iterated. In the worst case the procedure detects a single pair of interfering rules at each iteration and stops when all the possible pairs are generated. Hence in the worst case the overall number of clauses generated by the procedure is still in $O(|\mathcal{F}| + |\mathcal{L}|^2)$, but in many cases of interest, a small number of iterations is sufficient to discover an attack or to conclude that no attack exists.

We have thoroughly experimented with the variants of the linear encoding we have just presented: on all the problems considered the abstraction/refinement strategy outperforms both the approach based on the standard linear encoding and that based on the bitwise representation of the actions. Details of the experimental analysis can be found in [29].

4.3.2 The Graphplan-based Encoding

We recall that by using the linear encoding the formula representing the transition relation, namely $\overline{T}(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$, is independent from the time step i . From this it follows that important simplifications are possible on this formula. For instance, not all the rules are applicable at time step 0, yet the formula

$$\overline{T}(\mathbf{f}^0, \boldsymbol{\rho}^0, \mathbf{f}^1) \quad (36)$$

encodes the effects of all possible rules, also those that are not applicable in the initial state. By looking at the initial states it is possible to build a simpler version of (36), say $T_0(\mathbf{f}^0, \boldsymbol{\rho}^0, \mathbf{f}^1)$, specifying the effect of only those rules that are applicable in the initial state. The same line of reasoning can be applied at the subsequent steps: by computing an over-approximation of the reachable states at time step i we can then determine a simplified encoding of the transition relation at time step i , say $T_i(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$, for $i = 0, \dots, k-1$. Formally, the simplified encoding of such an over-approximation will be such that for every $j = 0, \dots, k$ the formulae encoding the (multi)-set rewriting system by means of

the linear encoding, i.e. $I(\mathbf{f}^0) \wedge \bigwedge_{i=0}^{j-1} \overline{T}(\mathbf{f}^i, \rho^i, \mathbf{f}^{i+1})$, and by means of the graphplan-based encoding, i.e. $I(\mathbf{f}^0) \wedge \bigwedge_{i=0}^{j-1} T_i(\mathbf{f}^i, \rho^i, \mathbf{f}^{i+1})$, are logically equivalent.

The graphplan-based encoding is defined on the basis of the above ideas. Preliminary to the generation of the encoding is the construction of a data structure, called *planning graph*, used to determine for each time step an over-approximation of the reachable states. More in general, the planning graph concisely represents an over-approximation of the forward search tree and the encoding of the transition relation can be drastically simplified by exploiting this information.

In the rest of this section we will first describe how to construct a planning graph representing the over-approximation of the reachable states for the time step k , and then we will present how to exploit such an over-approximation while building the propositional formula $I(\mathbf{f}^0) \wedge \bigwedge_{i=0}^{k-1} T_i(\mathbf{f}^i, \rho^i, \mathbf{f}^{i+1})$ that encodes the unfolding of the transition relation up to k steps.

The k -*planning graph* for Γ is a directed acyclic graph

$$([\Gamma])_k = \langle N_f, N_\rho, \xrightarrow[pre]{}, \xrightarrow[add]{}, \xrightarrow[del]{}, \oplus_f, \oplus_\rho \rangle$$

where N_f is a time-indexed family of sets of *fact nodes*, i.e. $N_f^i \subseteq \mathcal{F}$ is the set of fact nodes of *layer* i , for $i = 0, \dots, k$; similarly $N_\rho \subseteq \mathcal{L}$ is a time-indexed family of sets of *rule nodes*, i.e. N_ρ^i is the set of rule nodes of *layer* i for $i = 1, \dots, k$; $\xrightarrow[pre]{}$, $\xrightarrow[add]{}$, and $\xrightarrow[del]{}$ are time-indexed binary relations between between rule nodes and fact nodes, i.e. $\xrightarrow[pre]{i} \subseteq N_\rho^i \times N_f^i$, $\xrightarrow[add]{i} \subseteq N_\rho^i \times N_f^{i+1}$, and $\xrightarrow[del]{i} \subseteq N_\rho^i \times N_f^{i+1}$, whose instances are called *preconditions*, *add*, and *delete* edges respectively, for $i = 0, \dots, k-1$; finally \oplus_f and \oplus_ρ are time-indexed (irreflexive and commutative) relations of mutual exclusion (*mutex* for short) between facts (i.e. $\oplus_f^i \subseteq N_f^i \times N_f^i$) and rules (i.e. $\oplus_\rho^i \subseteq N_\rho^i \times N_\rho^i$) respectively, for $i = 1, \dots, k$.

As a preliminary step to the construction of the k -planning graph for Γ the sets \mathcal{L} and \mathcal{R} are extended with the *no-operation* rule labels and rules of the form $nop(f)$ and $(f \xrightarrow{nop(f)} f)$ respectively, for all $f \in \mathcal{F}$. The k -planning graph for Γ is then inductively defined as follows:

(PG1) $N_f^0 = \mathcal{I}$; moreover, $f \in N_f^i$ if and only if there exists $\rho \in N_\rho^{i-1}$ such that either $\rho \xrightarrow[add]{i-1} f$ or $\rho \xrightarrow[del]{i-1} f$, for $i = 1, \dots, k$;

(PG2) $\rho \in N_\rho^i$ if and only if $\text{pre}(\rho) \subseteq N_f^i$ and for all facts $f, f' \in \text{pre}(\rho)$ not $f \oplus_f^i f'$, for $i = 0, \dots, k-1$;

(PG3) for each $\rho \in N_\rho^i$ and for $i = 0, \dots, k-1$:

$\rho \xrightarrow[pre]{i} f$ if and only if $f \in \text{pre}(\rho)$,

$\rho \xrightarrow[add]{i} f$ if and only if $f \in \text{add}(\rho)$, and

$\rho \xrightarrow[del]{i} f$ if and only if $f \in \text{del}(\rho)$;

(PG4) $\rho \oplus_\rho^i \rho'$ if and only if $\rho \neq \rho'$ and either

(a) there exists $f \in \mathcal{F}$ s.t. $\rho \xrightarrow[del]{i} f$, and $\rho' \xrightarrow[add]{i} f$ or $\rho' \xrightarrow[pre]{i} f$, or

(b) there exist $f, f' \in \mathcal{F}$ s.t. $\rho \xrightarrow[pre]{i} f$, $\rho' \xrightarrow[pre]{i} f'$, and $f \oplus_f^i f'$, for $i = 0, \dots, k-1$;

(PG5) $f \oplus_f^i f'$ if and only if $f \neq f'$ and for all $\rho, \rho' \in N_\rho^{i-1}$ such that $\rho \xrightarrow[add]{i-1} f$ and $\rho' \xrightarrow[add]{i-1} f'$ we have $\rho \oplus_\rho^{i-1} \rho'$, for $i = 1, \dots, k$.

It can be shown that the time needed to build the k -planning graph for Γ is polynomial in the size of Γ . Notice that unlike in the linear encoding it is not necessary to assume that \mathcal{L} and \mathcal{R} have finite size: finite over-approximations of the facts reachable in k steps and of the rules executable in one of the k steps (resp.) are automatically built during the construction of the planning graph. These over-approximations are still in the worst case exponential in the size of the original specification, but are usually dramatically smaller than those computed for the linear encoding.

Example 4 Let us consider the quantifier-free (multi)-set rewriting system Γ such that $\mathcal{F} = \{v, x, y, w, z\}$, $\mathcal{L} = \{a, b, c, d\}$, $\mathcal{I} = x \cdot y$, and $\mathcal{R} = \{(x \xrightarrow{a} x \cdot y \cdot z), (x \cdot z \xrightarrow{b} w \cdot z), (w \xrightarrow{c} w \cdot z), (v \xrightarrow{d} v \cdot x)\}$.

The forward search tree of depth 3 associated with Γ is depicted in Figure 6. By applying rule *a* the system moves from the initial state $x \cdot y$ to the state $x \cdot y \cdot z$ (i.e. the fact z is added by *a*); by executing any of the applicable no-operation rules—denoted by $\text{nop}(_)$ —the system remains in the current state $x \cdot y$ and the same branch of the forward search tree can be repeated; no other rules can be applied from the initial state.

A graphical representation of $([\Gamma])_3$ is given in Figure 7. The set of fact nodes are built starting with layer 0 ($N_f^0 = \{x, y\}$) and the mutex relation on the same layer is initialised to the empty set ($\oplus_f^0 = \emptyset$). The rest of the planning graph is built by repeating the following steps for $i = 0, 1, 2$: (i) the set of rule nodes at layer i is computed, e.g. $N_\rho^0 = \{a, \text{nop}(x), \text{nop}(y)\}$; (ii) the mutex relation on the rules at layer i is calculated, e.g. $a \oplus_\rho^1 b$ and $b \oplus_\rho^1 \text{nop}(x)$ at layer 1; (iii) the set of fact nodes at layer $i+1$ is computed, e.g. $N_f^1 = \{x, y, z\}$; and (iv) the mutex relation on the facts at layer $i+1$ is calculated, e.g. $x \oplus_f^2 w$ at layer 2.

The over-approximation of the set of states reachable at step $i \leq k$ associated with $([\Gamma])_k$ is given by:

$$\text{MBR}([\Gamma]_k, i) = \{S \subseteq N_f^i : \text{not } f_1 \oplus_f^i f_2 \text{ for all } f_1, f_2 \in S\}$$

Thus if S is reachable in i steps, then $S \in \text{MBR}([\Gamma]_k, i)$, for $i = 0, \dots, k$, while the converse does not necessarily hold as illustrated by the following example.

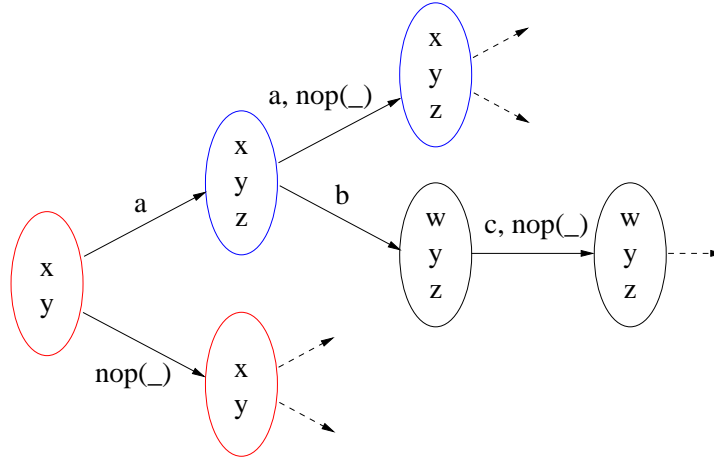


Fig. 6 Forward search tree

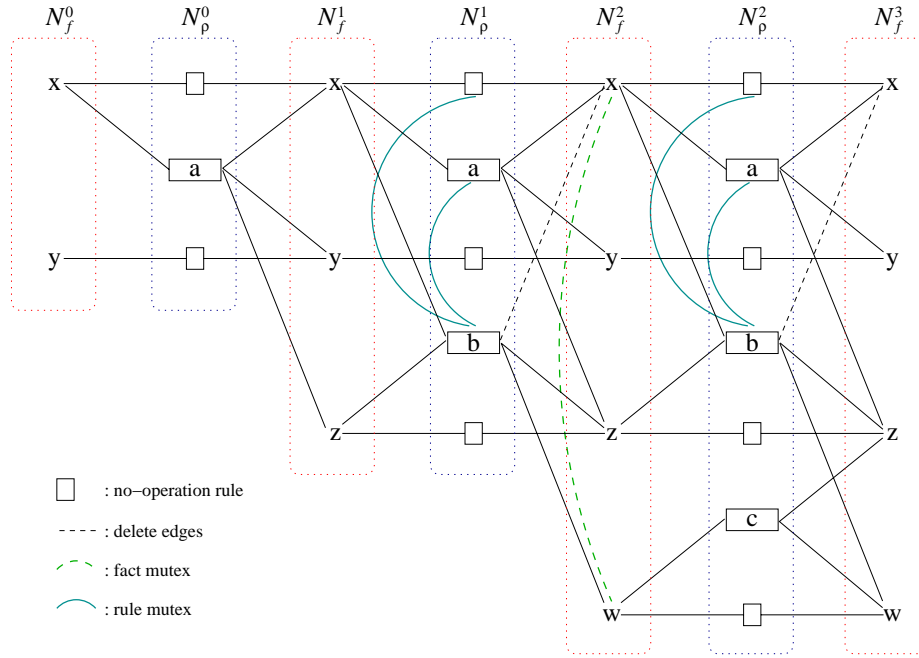


Fig. 7 Planning graph

Example 5 With reference to the multi-set rewriting system Γ presented in Example 4 and the associated planning graph of Figure 7, it is immediate to see that

$$\text{MBR}(\langle \Gamma \rangle_3, 1) = \{x, y, z, x \cdot y, x \cdot z, y \cdot z, x \cdot y \cdot z\}$$

This ensures that all the states in $2^{\mathcal{F}} \setminus \text{MBR}(\langle \Gamma \rangle_3, 1)$, i.e. all the states containing the fact w , are unreachable at depth 1. However we cannot guarantee that the states in $\text{MBR}(\langle \Gamma \rangle_3, 1)$ are indeed reachable. To show this it suffices to consider the state x . This state is in

$\text{MBR}(\langle \Gamma \rangle_3, 1)$, but it is not reachable at depth 1 as shown by the forward search tree of Figure 6.

Given a quantifier-free reachability problem $\Xi = \langle \Gamma, \mathcal{B} \rangle$, if $\text{MBR}(\langle \Gamma \rangle_k, k) \cap \{B : B \models \mathcal{B}\} = \emptyset$, then no bad state represented by \mathcal{B} can be reached in k steps. However if there exists a goal state $S \in \text{MBR}(\langle \Gamma \rangle_k, k) \cap \{B : B \models \mathcal{B}\}$, we cannot conclude that S is truly reachable.

The problem of establishing if a state S included in the over-approximation of the forward search tree at time step k (i.e. if $S \in \text{MBR}(\langle \Gamma \rangle_k, k)$) is indeed reachable at

time step k can be reduced to checking if $M = \{f^k : f \in S\}$ is a model of $\llbracket \Gamma \rrbracket_k = I(\mathbf{f}^0) \wedge \bigwedge_{i=0}^{k-1} T_i(\mathbf{f}^i, \rho^i, \mathbf{f}^{i+1})$ where the initial state is encoded by

$$I(\mathbf{f}^0) = \bigwedge \{f^0 : f \in N_f^0\} \quad (37)$$

and $T_i(\mathbf{f}^i, \rho^i, \mathbf{f}^{i+1})$ for $i = 0, \dots, k-1$ is the conjunction of the following formulae.

Universal Axioms: for each $\rho \in N_\rho^i$

$$\rho^i \supset \bigwedge \{f^i \mid \rho \xrightarrow[pre]{i} f\} \quad (38)$$

$$\rho^i \supset \bigwedge \{f^{i+1} \mid \rho \xrightarrow[add]{i} f\} \quad (39)$$

$$\rho^i \supset \bigwedge \{\neg f^{i+1} \mid \rho \xrightarrow[del]{i} f\} \quad (40)$$

Explanatory Frame Axioms (EFA): for all $f \in N_f^i$

$$(f^i \wedge \neg f^{i+1}) \supset \bigvee \left\{ \rho^i \mid \rho \xrightarrow[del]{i} f \right\} \quad (41)$$

$$(\neg f^i \wedge f^{i+1}) \supset \bigvee \left\{ \rho^i \mid \rho \xrightarrow[add]{i} f \right\} \quad (42)$$

Additionally, for all $f \in (N_f^{i+1} \setminus N_f^i)$

$$f^{i+1} \supset \bigvee \left\{ \rho^i \mid \rho \xrightarrow[add]{i} f \right\} \quad (43)$$

Mutex Axioms: for all $p_1, p_2 \in (N_\rho^i \cup N_f^i)$ such that $p_1 \oplus p_2$:

$$\neg(p_1^i \wedge p_2^i) \quad (44)$$

The additional explanatory frame axioms (44) simply state that if a fact holds at time step $i+1$ and it occurs in the $(i+1)$ -th layer N_f^{i+1} but it does not occur in the preceding one (i.e. N_f^i), then at least one of the rules occurring in the i -th rule layer and adding f must hold at the time step i . Basically the extra explanatory frame axiom for the fact f is just a simplification of the explanatory frame axiom for f where f is imposed to be false at time step i . Notice that this follows from the observation that if f is introduced only at the $(i+1)$ -th fact layer of the planning graph, then there is no state reachable in i steps in which f holds.

We now state the soundness and completeness of the graphplan-based encoding. As for the case of the linear encoding, Ξ is ground and therefore $\mathcal{B} = \langle \mathcal{G}, \emptyset \rangle$ (with \mathcal{G} ground). We then define $\llbracket \mathcal{B} \rrbracket_k = \tilde{\mathcal{G}}^k$ where $\tilde{\mathcal{G}}^k$ is obtained from \mathcal{G} by first replacing every fact $f \notin N_f^k$ by *false* and every other fact p with p^k .

Theorem 6 (Soundness and Completeness of the Graphplan-based Encoding) *If M is a model of $\llbracket \Xi \rrbracket_k$ for $k \geq 0$, then $S_0 \rho_0 S_1 \dots S_{k-1} \rho_{k-1} S_k$ is a (partial-order) solution of $\hat{\Xi}$, where $S_i = \{f \in \mathcal{F} : f^i \in M\}$ for $i = 0, \dots, k$ and ρ_i is the parallel composition of the rules in $\{\rho \in \mathcal{L} : \rho^i \in M\}$ for $i = 0, \dots, k-1$. Conversely, if $\pi_k = S_0 \rho_0 S_1 \dots S_{k-1} \rho_{k-1} S_k$ is a (partial-order) solution of $\hat{\Xi}$, then there exists a model M of $\llbracket \Xi \rrbracket_k$ such that $\pi(M) = \pi_k$.*

In the worst case the number of atoms and clauses generated by the graphplan-based encoding are equal to those generated with the linear encoding. However, in most cases $|N_f^i| \ll |\mathcal{F}|$ (for $i = 0, \dots, k$) and $|N_\rho^i| \ll |\mathcal{L}|$ (for $i = 0, \dots, k-1$) and therefore the formulae generated by the graphplan-based encoding are usually considerably smaller than those built with linear encoding.

5 SAT-based model checking for security protocol analysis

The techniques presented in the previous sections are the fundamental building blocks of the SAT-based model checking procedure outlined in Figure 1. In particular

- the optimising transformations on the input protocol insecurity problem described in Section 3.5 are applied at line 3 of the procedure,
- the techniques for the elimination of quantifiers presented in Section 4.1 are applied at line 4, and
- the encoding techniques described in Section 4.3 are applied at line 6. Notice that $\llbracket \Xi \rrbracket_i$ need not be computed from scratch at each iteration. When building $\llbracket \Xi \rrbracket_i$ it is in fact possible to reuse (selected parts of) the formula generated at the previous step thereby speeding up significantly the whole procedure.

We recall that (by setting k to ∞) the procedure in Figure 1 can be used as a semi-decision procedure for the input protocol insecurity problem Ξ , i.e. the procedure halts with a solution when it exists, but it may not terminate otherwise. In the rest of this section we discuss the termination and the complexity of the procedure when applied to a BNAS protocol insecurity problems Ξ . Let n be the DAG size of Ξ .

Termination. We know from [61] that for every attack on Ξ there exists an attack on Ξ such that between the consecutive application of two rules of the honest agents there are at most $3n^2$ applications of the intruder rules. Since sessions are acyclic, the maximal number of rules of the honest agents occurring in an attack is $r * s$, where r and s are the number of rules of the honest agents and the number of sessions in Ξ . Therefore we can restrict our attention to attacks of length $\leq k_{max}$, where $k_{max} = 3n^2(r*s+1)$. Thus the procedure of Figure 1 is a decision procedure for BNAS problems provided that we add the following line between lines 8 and 9:

- 8'. If $\llbracket \Xi \rrbracket_i$ is unsatisfiable and $i \geq k_{max}$, then print “No attack exists for the BNAS input problem” and then **halt**.

A smaller value for k_{max} can be computed whenever the specialisation of the composition rules of the intruder described in Section 3.5 is applicable. Let Ξ' be the result of applying this optimisation to Ξ . Since the composition rules of the intruder are built into the rules of

the honest agents, the only possible activity that the intruder can do in between two applications of rules of the honest agents is to apply a sequence of decomposition rules. It is easy to see that this sequence can comprise at most $d - 1$ applications of decomposition rules, where d is the maximal depth of the messages occurring in the right hand side of the rewrite rules of Ξ . Thus $k_{max} = (d - 1)(r * s + 1)$ suffices. In fact the application of the optimising transformation to Ξ does not change the number of sessions and, although the number of rules in Ξ' is usually greater than that of Ξ , each path will contain at most r application of the (specialised) rules of the honest agents. (We recall from Section 3.5 that each path of Ξ' contains at most one application of the rules in $\mathcal{RC}(sc.r)$ for each step compressed rule $sc.r$ of Ξ .) For the two-sessions variant of the NSPK protocol with the Lowe's fix [49], well-known to be not flawed when type-flaw attacks are neglected, we can thus safely bound our analysis to $k_{max} = (3 - 1)(4 * 2 + 1) = 18$ steps.

Under the same hypotheses, there is another, usually very effective approach to determine whether $\llbracket \Xi \rrbracket_i$ covers all possible attacks of Ξ , and thus to enforce an early (but still safe) termination of the procedure when the graphplan-based encoding is applied. Following [20] we say that a planning graph $\llbracket \Gamma \rrbracket_i$ has *levelled-off* when two adjacent fact levels are identical. The following result, which applies to BNAS problems obtained by specialising the composition rules of the intruder as described in Section 3.5, ensures that when the planning graph has levelled off, if the current formula $\llbracket \Xi \rrbracket_i$ is unsatisfiable then there is no attack on Ξ . (Notice that for BNAS problems obtained by specialising the composition rules of the intruder, the planning graph is always guaranteed to reach the levelled-off.)

Theorem 7 *Let $\Xi = \langle \Gamma, \mathcal{B} \rangle$ be a BNAS problem obtained by specialising the composition rules of the intruder as described in Section 3.5. When $\llbracket \Gamma \rrbracket_i$ has levelled-off, if $\llbracket \Xi \rrbracket_i$ is unsatisfiable, then there is no attack on Ξ .*

By using the above result we can thus further improve the procedure of Figure 1 by adding the following line just after line 8:

8". If the planning graph has levelled off and $\llbracket \Xi \rrbracket_i$ is unsatisfiable, then print "No attack exists for the BNAS input problem" and then **halt**.

Notice that in this way the procedure can conclude that the protocol is secure without reaching k_{max} . For the two-sessions variant of the NSPK protocol with the Lowe's fix the procedure halts at $i = 7$, thereby saving 11 iterations.

Complexity. We now discuss the complexity of our model checking procedure when used as a decision procedures for BNAS problems. We start by first looking at the complexity of the individual steps of the procedure and then

we discuss the complexity of the overall procedure. We consider both variants of the procedure we have just outlined.

- *Optimising transformations* (line 3). The optimising transformations turn Ξ into a new problem Ξ' with a number of rewrite rules that in the worst case is exponential in n , i.e. Ξ' is in $O(2^n)$.
- *Quantifier elimination* (line 4). It is easy to see that quantifier elimination does not have any impact on the overall complexity of the procedure.
- *Generation of the encodings* (line 6). We recall from Section 4.3 that the number of atoms and the number of clauses in $\llbracket \Xi' \rrbracket_i$ are in $O(|\mathcal{F}| + |\mathcal{L}|)$ and $O(|\mathcal{F}| + |\mathcal{L}|^2)$ respectively. We know from [61] that we can safely restrict the instantiation of variables with ground terms of size $\leq n$. We observe that the number of terms of size $\leq n$ is in $O(n^{n+1})$. (In fact each term can be regarded as a tree with n nodes together with a mapping from the nodes to the function symbols of Σ' . Since the number of function symbols is $\leq n$, it is easy to conclude that the number of terms of size n is in $O(n^{n+1})$. By summing the number of terms of size $\leq n$ we obtain $O(n^{n+1})$.) Since the number of variables occurring in a term is in $O(n)$, the size of \mathcal{F} and \mathcal{L} are thus in $O(n^{n+2})$ and $O(2^n * n^{n+2})$ respectively. As a consequence the number of atoms and clauses of $\llbracket \Xi' \rrbracket_i$ are in $O(i * n^{n+2})$ and $O(i * 2^{2n} * n^{2(n+2)})$ and thus in $O(i * 2^{(n+2) \log_2 n})$ and $O(i * 2^{2n+2(n+2) \log_2 n})$ respectively.

As we previously said, the iterations of the **while** loop of the procedure is bound by k_{max} , where k_{max} is polynomial in n .

To summarise, in the worst case, our decision procedure generates and solves polynomially many propositional formulae whose size is at most exponential in the size of the input specification. Though this complexity result seems to limit the applicability of the approach to small size protocols, in the next section we will see that our approach generates formulae of manageable size which are quickly solved by the SAT solver in many cases of interest.

6 Experimental Results

We have implemented the above ideas in SATMC [9], a SAT-based Model Checker for the automatic analysis of security protocols. SATMC is available at <http://www.ai.dist.unige.it/satmc>. SATMC is also integrated in the AVISPA Tool [5], a push-button tool for the automated validation of Internet security-sensitive protocols and applications, and can be used on-line via the AVISPA graphical user interface available at <http://www.avispa-project.org>.

We thoroughly assessed our approach by running SATMC on a large set of protocols drawn from so-called

Clark-Jacob library and the AVISPA Library. The Clark-Jacob library [27] is a collection of small and medium-sized authentication protocols. The AVISPA Library (available at <http://www.avispa-project.org>) is a selection of practically relevant, industrial protocols that have recently been standardised or are currently undergoing standardisation, especially at the Internet Engineering Task Force (IETF). For each protocol we consider a BNAS problem modelling a scenario consisting of a bounded number of sessions in which the involved principals exchange messages over a channel controlled by a Dolev-Yao intruder. These BNAS problems are automatically generated by the AVISPA Tool starting from a protocol specification in a higher level specification language called HLPSL [25]. All the experimental results presented in this section have been obtained by setting a resource limit of 1 hour CPU time and 1GB memory, on a Pentium IV 2.4GHz under Linux. SATMC is invoked with k set to ∞ on all the problem considered and the tool always terminates by either reporting an attack, by positively concluding that no attack is possible on the given problem, or by exhausting the available resources.

In order to fine tune our techniques and quantitatively assess the relative merits of the encoding techniques discussed in Section 4 we run the encoding techniques (including some variants thereof) implemented in SATMC against a selection of (flawed) security protocols drawn from the Clark-Jacob library. We found that the linear encoding performs best when used in conjunction with the abstraction/refinement strategy discussed in Section 4.3.1. We also found that the graphplan-based encoding performs best if we weaken the definition of \oplus by enforcing only static mutexes (i.e., by neglecting PG5 and by considering only case (a) of clause PG4 in the definition of the planning graph given in Section 4.3.2). By using this weaker version of the mutex relation we obtain a coarser over-approximation of the set of the reachable states. Our experimental results indicate that this loss of precision in the construction of the planning graph is largely compensated by a drastic reduction in the time spent to build the planning graph.

The best results obtained with each encoding are given in Table 2. For each protocol we give the smallest value of k at which the attack is found (**K**) and for each encoding technique we give the number of propositional variables (**A**) and clauses (**CL**) in the SAT formula, the time spent to generate the SAT formulae (**EncT**) and the total time spent by the SAT solver to solve the SAT formulae (**SolT**). The comparison shows that, by using the graphplan-based encoding technique, we obtain (i) SAT instances whose size is up to 3 orders of magnitude smaller and (ii) significantly better encoding and solving times. Moreover by using the graphplan-based encoding SATMC succeeds in analysing protocols (e.g., the *KLS rep.* protocol) that otherwise are not solved with the given resources.

We have then compared SATMC against two other state-of-the-art protocol analysers integrated in the AVISPA Tool, namely CL-AtSe [26,64] and OFMC [12], by running them against the security protocols in the AVISPA Library. The results are given in Table 3. These results are obtained by assuming that all the messages exchanged are well-typed and that they are freely generated.¹ For each protocol we give the number of security problems analysed (“#Pb”) and the number of problems for which attacks are detected (“#Att”), and for each back-end, the average time spent by the back-end to analyse (i.e. either to find the attacks or to report that no attack exists in the scenario considered) a single problem of the protocol. For SATMC we report only the time spent by the back-end to generate the SAT formula, since in this case the time spent by the SAT-solver—we used the Chaff solver [56] for these experiments—to solve the formula is negligible.

The boxed number associated with ISO-PK-3 (corresponding to the ISO Public Key Two-Pass Mutual Authentication protocol [44]) indicates that the back-ends have found a new (i.e. previously unknown) attack on the protocol. It was already known that ISO-PK3 is vulnerable to replay attacks and hence it does not provide strong authentication [32] as nothing in the messages ensures the freshness of the messages for the responder. Our analysis, however, shows that the ISO-PK3 protocol does not even guarantee weak authentication, i.e. after successfully executing the protocol, neither the initiator nor the responder can be sure about the authenticity of the exchanged messages.

In general the experiments indicate that the time spent by SATMC and by the other tools is small for all the problems. More in particular, OFMC and CL-AtSe are very fast and performs better than SATMC in many cases whereas SATMC performs slightly better on some instances (namely FairZG and PBK-fix-weak-auth) and on a few protocols (namely Simple, Pervasive, and QoS-NSLP) SATMC significantly outperforms the other tools.

By inspecting the problems used for the experiments of Table 3 it is possible to conclude that OFMC and CL-AtSe are less sensitive than SATMC to the size of messages exchanged by the protocols. This is not surprising since we know that the complexity of our approach depends on the size of the messages exchanged. We also noticed that SATMC is less sensitive than OFMC and CL-AtSe to the number of concurrent sessions considered. To verify this we then run all the three tools against problems modelling both the flawed and the fixed variants of the NSPK-KS protocol (i.e. the Needham-Schroeder Public Key protocol including the key-server [57]) for increasing numbers of concurrent sessions. The experimental results, summarised in Table 4, indicate that OFMC

¹ It is worth pointing out that OFMC and CL-AtSe can also deal with some properties of cryptographic operators such as, e.g., exponentiation.

Table 2 Experimental data on protocols from the Clark/Jacob library

Protocol	K	Linear Encoding				Graphplan-based Encoding			
		A	CL	EncT	SolT	A	CL	EncT	SolT
<i>Andrew</i>	9	145,500	2,256,256	111.4	12.1	447	1,368	0.17	0.00
<i>EKE</i>	5	61,508	783,484	74.1	3.7	396	1,340	0.20	0.00
<i>ISO-CCF-1 U</i>	4	283	839	0.1	0.0	107	300	0.03	0.00
<i>ISO-CCF-2 M</i>	4	933	4,289	0.2	0.1	118	314	0.04	0.00
<i>ISO-PK-1 U</i>	4	501	1,573	0.1	0.0	154	423	0.06	0.00
<i>ISO-PK-2 M</i>	4	1,741	9,633	0.6	0.1	132	366	0.06	0.00
<i>ISO-SK-1 U</i>	4	295	859	0.1	0.0	98	270	0.02	0.00
<i>ISO-SK-2 M</i>	4	848	3,320	0.4	0.0	122	319	0.06	0.00
<i>KaoChow 1</i>	7	36,318	130,917	8.0	0.7	436	1,778	0.21	0.00
<i>KaoChow 2</i>	9	530,726	1,804,005	140.4	15.6	736	3,397	0.38	0.01
<i>KaoChow 3</i>	9	995,323	5,736,662	585.5	41.6	1,000	6,122	0.80	0.01
<i>KLS rep.</i>	7			MO		1,658	23,219	5.86	0.03
<i>NSCK</i>	9	114,530	334,086	17.1	1.3	448	1,403	0.17	0.00
<i>NSPK</i>	7	6,612	33,326	1.5	0.1	423	1,276	0.14	0.00
<i>NSPK-server</i>	8	9,157	53,741	2.8	0.1	848	2,704	0.32	0.00
<i>SPLICE</i>	9	14,142	62,392	3.6	0.2	963	3,166	0.50	0.00
<i>Swick 1</i>	5	4,266	12,947	0.8	0.1	196	559	0.08	0.00
<i>Swick 2</i>	6	7,551	28,573	1.7	0.1	261	843	0.14	0.00
<i>Swick 3</i>	4	4,558	11,394	0.7	0.1	175	503	0.05	0.00
<i>Swick 4</i>	5	14,557	57,111	10.2	0.3	223	657	0.08	0.00
<i>Stubblebine rep</i>	3	12,860	94,714	6.3	0.1	147	481	0.07	0.01
<i>Woo-Lam M</i>	6	481,394	2,498,382	304.4	7.7	485	1,600	0.27	0.00

MO means that a memory out has been reached

and CL-AtSe fail to analyse the problem within the given resources when more than three sessions are considered, whereas SATMC successfully analyse all the problems considered and scales very smoothly as the number of sessions increases.

7 Related Work

We broadly classify the related approaches in SAT-based automated reasoning techniques and in techniques for the automatic analysis of security protocol.

7.1 SAT-based Automated Reasoning Techniques

Planning as Satisfiability. We have greatly benefited from the experience matured in the area of “Planning as Satisfiability” initiated by the seminal paper by Kautz and Selman [46]. While the encoding techniques we use are adapted from those proposed for planning [45, 35], their application to the automatic analysis of security protocols is new as new are the proofs of soundness and correctness of the encodings of Section 4. Proofs of similar results have been recently and independently developed for planning by J. Rintanen, K. Heljanko, and I. Niemelä [60]. However their results apply to linear encodings only, whereas our approach is more general as it applies to graphplan-based encodings as well. In the same paper the authors propose encodings that may lead to shorter

solutions than that allowed by the encodings used in our paper. This is an interesting feature, but its application may lead a number of additional clauses in the formula that grows cubically in the size of the input problem. While this may be acceptable for small problems, it does not seem very promising for our application domain.

We have also written a translator from protocol insecurity problems to planning problems and we have generated a set of planning problems corresponding to protocol insecurity problems taken from the Clark-Jacob library. We have then fed these problem to *blackbox* v.4.1, a state-of-the-art planner based on the planning as satisfiability paradigm. The experimental results reported in [29] indicate that SATMC performs uniformly and significantly better than *blackbox* on these problems. One of the problem with *blackbox* is the limited expressiveness of the PDDL input specification language [39]. For instance only individual constants are allowed by PDDL and this can be circumvented only by generating very large PDDL specifications.

Answer Set Programming. It is well-known that logic programming with answer sets semantics is closely related to propositional logic [58, 48] and AI planning [47]. In [3] the authors present an executable specification language for security protocols based on logic programming and use an efficient model generators for logic programs to find attacks on protocols. Experimental results reported in the same paper show the feasibility of the approach on protocols in the Clark-Jacob library. SATMC performs

Table 3 Experiments on the AVISPA library

Protocol	#Pb	#Att	CL	OF	SATMC
UMTS-AKA	4	0	0.01	0.03	0.01
ISO1	1	1	0.02	0.02	0.04
ISO2	1	0	0.02	0.07	0.63
ISO3	2	2	0.03	0.03	0.39
ISO4	2	0	0.03	0.38	208.31
ChapV2	4	0	0.02	0.18	0.10
EKE	4	2	0.03	0.10	0.09
TLS	4	0	0.05	0.29	1018.28
LPD-MSR	2	2	0.02	0.02	0.06
LPD-IMSR	2	0	0.04	0.04	0.10
Kerb-basic	10	0	0.07	0.61	6.24
Kerb-Cross-Realm	18	0	0.52	2.22	5.70
Kerb-Ticket-Cache	11	0	0.08	0.60	28.90
Kerb-PKINIT	12	0	0.06	0.47	27.21
Kerb-Forwardable	12	0	0.16	7.12	TO
Kerb-PreAuth	12	0	0.12	0.39	20.54
CRAM-MD5	2	0	0.04	0.23	0.17
PBK	1	1	0.01	0.34	0.25
PBK-fix	1	1	0.03	0.14	0.09
PBK-fix-weak-auth	1	0	0.49	3.47	0.33
DHCP-delayed-auth	2	0	0.02	0.06	0.12
TSIG	2	0	0.05	0.19	0.38
ASW	3	0	0.10	0.35	TO
ASW-abort	4	1	0.20	1.98	65.75
FairZG	5	0	0.37	8.76	0.28
SET-purchase	4	2	23.66	1.24	TO
SET-p.-hon.-payment-gw	4	0	0.61	0.83	TO
AAAMobileIP	9	0	0.03	0.14	0.11
Simple	3	0	100.17	78.50	0.50
CTP-non_predictive-fix	3	0	0.06	0.23	TO
geopriv	5	0	0.04	0.25	0.08
pervasive	2	0	47.67	30.52	4.00
two_pseudonyms	5	0	0.06	0.30	0.07
QoS-NSLP	2	0	32.16	16.01	0.21
sip	1	0	0.05	1.86	810.01

\boxed{n} : n new attacks has been found

TO : time-out

Table 4 Experiments on the NSPK-KS with increasing number of sessions

Protocol	#S	Attack	CL	OF	SATMC (EncT / SolT)
NSPK-KS	2	YES	0.06	3.56	2.05 / 0.03
	3	YES	185.90	1,038.40	3.89 / 0.04
	4	YES	TO	TO	6.54 / 0.03
	5	YES	TO	TO	8.55 / 0.04
	6	YES	MO	TO	12.95 / 0.05
NSPK-KS-fix	2	NO	2,616.40	10.74	2.07 / 0.02
	3	NO	TO	1,035.29	4.66 / 0.03
	4	NO	TO	TO	7.19 / 0.06
	5	NO	TO	TO	10.72 / 0.07
	6	NO	MO	TO	14.69 / 0.06

#S : number of sessions

YES : an attack has been found

NO : the protocol is safe under the analysed scenario

MO : memory-out

TO : time-out

significantly better on the same problems and—as we showed in Section 6—it also successfully analyses protocols of industrial complexity. A further attempt to use logic programming with answer sets semantics to analyse security protocols is described in [11].

Bounded Model Checking. The approach described in this paper belongs to the wider area of bounded model checking [15] and as such it can benefit from results obtained in this context. We are currently extending SATMC in order to support the analysis of protocols with respect to goals expressed as generic LTL formulae. This will allow for the specification of (i) different, more sophisticated intruder models (e.g. resilient and location-limited communication channels) by means of fairness constraints and (ii) more complex security properties.

7.2 Other techniques for security protocols analysis

Process Calculi. Casper/FDR has been applied to analyse security protocols of the Clark-Jacob library in [32]. Protocol specifications, written in the high level Casper specification language, are translated into CSP specifications which are then fed to FDR. This approach has been very successful to discover new flaws in protocols. However, some preliminary experiments indicate that SATMC performs better than Casper/FDR. Besides this, Casper/FDR is not able to handle non-atomic keys and this prevents the application of the approach to the analysis of many real-world protocols. Our approach does not suffer from this limitation.

A technique for the automatic analysis of security protocols based Non-Interference is proposed in [38]. The approach, which is based on a slight extension of CCS, has been proved effective a large subset of protocols from the Clark-Jacob library [33]. The Non-Interference check provides a sufficient condition for the absence of attacks, but, whenever the Non-Interference check fails, the set of interferences returned must be (manually) inspected by the user in order to determine those corresponding to attacks. Another variant of CCS is at the core of the partial model checking technique for security protocols proposed in [53]. Here the protocol is modelled as an open system and therefore the analysis does not rely on a specific intruder model. The approach is currently limited to the analysis of security protocols with a bounded number of acyclic sessions and bounded message size. A decidability result has been proved when the analysis is restricted to secrecy properties. The techniques has been successfully applied against several protocol instances including NSPK and SSL, but since a systematic experimental analysis is not publicly available it is difficult to assess its scalability.

A novel static analysis technique for security protocols is proposed in [22]. The technique, which is based on LySa (a process algebra inspired by the Spi-calculus),

is fully automatic and always terminating in polynomial time. The tool builds an over-approximation of the behaviour of the protocol (including also the actions performed by a Dolev-Yao intruder). If no traces representing attacks on the protocol are in the over-approximation, then the protocol is guaranteed to satisfy the expected security property. Otherwise the over-approximation is explored in order to detect an attack on the protocol. If the search is successful then the found attack is reported to the user, otherwise the result is inconclusive. However this does not happen so frequently and the approach has been shown to be efficient several security protocols of interest [21, 42]. (A related static analysis technique for authentication protocols is presented in [23].) Our work is complementary to this as it is capable to detect all the attacks on the protocol for the given scenario but may fail to establish their absence in the general case.

The idea of type-checking secrecy properties of cryptographic protocols has been put forward in [1]. The approach has been later extended to support the verification of authentication properties by using both types and effects and by annotating protocol specifications (in the Spi-calculus) with correspondence assertions [40, 41]. As in the previous case, these techniques always terminate and can check the absence of attacks, but can report spurious attacks.

Blanchet's Logic Programming Approach. Bruno Blanchet has developed a technique and a tool (called ProVerif) in which protocols and security properties are expressed as sets of Horn clauses which are then saturated by means of different strategies [17–19]. ProVerif allows one to prove security properties for an unbounded number of protocol sessions, in particular strong secrecy (which means that an intruder cannot see any difference when the value of the secret changes). The tool may raise some false attacks since nonces are abstracted by constants or function symbols, so that attacks have to be constructed by the user itself. (Recently, however, ProVerif has been enhanced to reconstruct automatically the attacks in some cases [4].) Termination of the technique (which is not guaranteed in the general case) has been proved for the restricted class of protocols in which encryption primitives are distinguished by means of a tagging mechanism [16].

Explicit State Model Checking. The general purpose state enumeration based model-checker Murphi has been applied to analyse some small cryptographic protocols such as the NSPK protocol [55]. Experimental results indicate that Murphi suffers from state-space explosion. To cope with this problem the user must restrict the model in several ways. For instance, the size of the network channel as well as the size of the intruder knowledge are fixed a priori. Moreover the intruder model is not provided by the tool, but the burden is on the user to come out with an appropriate finite state machine describing the

intruder behaviour. As a result, an attack on the protocol can be missed because either the bound imposed for some parameter is too small (e.g. the attack may require an intruder with more memory than that specified) or an ability of the intruder that has not been specified.

BRUTUS combines depth-first search with a message derivation mechanism modelling the capabilities of the intruder in the spirit of the Paulson's synthesis and analyse operators [59]. A powerful specification logic is also provided to describe a variety of security properties including secrecy, authentication, non-repudiation, and a weak form of anonymity. Similarly to other explicit state model checkers, BRUTUS suffers from the state explosion problem, partially mitigated by some partial-order reduction techniques [28]. However the approach has been experimented only on a few protocols. Furthermore BRUTUS supports atomic keys only.

Lazy evaluation-based approaches for Protocol Insecurity Problems. Both OFMC [14] and CL-AtSe [26] combine forward search exploration with constraint solving using a symbolic representation of the search space. OFMC also features a novel partial order reduction technique, called *constraint differentiation* [13], that leads to considerable pruning in the search space. Both tools are also capable to exploit some algebraic properties of the cryptographic operators.

Thus both OFMC and CL-AtSe use symbolic techniques to explore the search space in a demand-driven way. SATMC is dual in this respect as it eagerly expands all the terms while building the propositional formula and delegates the search to the SAT solver. It is thus not surprising that OFMC and CL-AtSe are less sensitive than SATMC to the size of messages exchanged in the protocol. On the other hand SATMC is less sensitive than OFMC and CL-AtSe to the combinatorial explosion associated with the interleaved execution of multiple sessions of the protocol.

Strand Spaces approaches. Strand spaces [37] provide an alternative approach to state-based analysis of protocol by emphasising causal interactions among protocol participants. The approach has been implemented in the Athena protocol analyser [62] which combines model checking and theorem proving techniques with the (parametric) strand space model to reduce the search space and automatically prove the correctness of or find an attack on security protocols (if it terminates). Besides the strands for honest roles, Athena uses quite prolific strands for modelling the behaviour of the Dolev-Yao intruder. Moreover, Athena supports atomic keys only. In [54] and, later, in [30] these limitations have been uplifted.

These strands-based tools has been successfully experimented against the protocols in the Clark-Jacob library [27]. Since their input language is different from that of SATMC, a direct comparison is not possible, but according to the published material [62, 54, 30] the results

obtained with these tools are very good and promising. However, no results are available on protocols of industrial complexity like those in the AVISPA library.

8 Conclusions

We have presented a model checking procedure for security protocols based on a reduction to propositional logic. We have described a number of specialised techniques that are essential to make the approach not only viable but also effective on many problems of practical interest. In particular we have adapted and applied encoding techniques originally developed for planning to security protocol analysis, proving their soundness and completeness. Moreover, we have showed that our procedure can be turned into a decision procedure for BNAS problems. To demonstrate the feasibility and effectiveness of our approach we have run our tool against a large set of complex, industrial strength security protocols and compared it with two state-of-the-art protocol analysers. The experiments indicate that our tool is more sensitive to the size of the messages exchanged by the protocol but it scales better than other state-of-the-art tools as the number of sessions increases.

In the future we would like to extend our techniques so to provide support for the specification of security properties by means of generic LTL formulae. This will provide the user considerably more flexibility in the specification of the security properties that the protocol is expected to enjoy as well as of the properties of the environment (e.g. the communication channels) in which the protocol is executed.

Acknowledgements This work was partially funded by FET Open EC Project "AVISPA: Automated Validation of Internet Security Protocols and Applications" (IST-2001-39252) and by the FIRB Project no. RBAU01P5SS.

References

1. Abadi: Secrecy by typing in security protocols. JACM: Journal of the ACM **46** (1999)
2. Aho, A.V., Sloane, N.J.A.: Some doubly exponential sequences. Fibonacci Quarterly **11**, 429–437 (1973)
3. Aiello, L.C., Massacci, F.: Verifying security protocols as planning in logic programming. ACM Trans. on Computational Logic **2**(4), 542–580 (2001)
4. Allamigeon, X., Blanchet, B.: Reconstruction of attacks against cryptographic protocols. In: 18th IEEE Computer Security Foundations Workshop, (CSFW-18 2005), 20-22 June 2005, Aix-en-Provence, France, pp. 140–154 (2005)
5. Armando, A., Basin, D., Boichut, Y., Chevalier, Y., Compagna, L., Cuellar, J., Drielsma, P.H., Heam, P., Kouchnarenko, O., Mantovani, J., Moedersheim, S., von Oheimb, D., Rusinowitch, M., Santiago, J., Turuani, M., Viganò, L., Vigneron, L.: The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In: Proceedings of the 17th International Conference on Computer-Aided Verification (CAV'05) (2005)

6. Armando, A., Basin, D., Bouallagui, M., Chevalier, Y., Compagna, L., Mödersheim, S., Rusinowitch, M., Turuani, M., Viganò, L., Vigneron, L.: The AVISS Security Protocol Analysis Tool. In: Proceedings of CAV'02, LNCS 2404, pp. 349–354. Springer-Verlag (2002). URL of the AVISS and AVISPA projects: www.avispa-project.org
7. Armando, A., Compagna, L.: Automatic SAT-Compilation of Protocol Insecurity Problems via Reduction to Planning. In: Proceedings of FORTE 2002, LNCS 2529, pp. 210–225. Springer-Verlag (2002)
8. Armando, A., Compagna, L.: Abstraction-driven SAT-based Analysis of Security Protocols. In: E. Giunchiglia, A. Tacchella (eds.) Theory and Applications of Satisfiability Testing, LNCS 2919, pp. 257–271. Springer-Verlag (2004). Selected Revised Papers. Presented to SAT 2003, S. Margherita Ligure, Italy. Available at www.avispa-project.org
9. Armando, A., Compagna, L.: SATMC: a SAT-based Model Checker for security protocols. In: Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA 2004), LNAI 3229. Springer-Verlag, Lisbon, Portugal (2004)
10. Armando, A., Compagna, L., Ganty, P.: SAT-based Model-Checking of Security Protocols using Planning Graph Analysis. In: Proceedings of FME'2003, LNCS 2805. Springer-Verlag (2003)
11. Armando, A., Compagna, L., Lierler, Y.: Automatic compilation of protocol insecurity problems into logic programming. In: Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA 2004), LNAI 3229. Springer-Verlag, Lisbon, Portugal (2004)
12. Basin, D., Mödersheim, S., Viganò, L.: An On-The-Fly Model-Checker for Security Protocol Analysis. In: E. Sneekenes, D. Gollmann (eds.) Proceedings of ESORICS'03, LNCS 2808, pp. 253–270. Springer-Verlag (2003). Available at <http://www.avispa-project.org>
13. Basin, D., Mödersheim, S., Viganò, L.: Constraint Differentiation: A New Reduction Technique for Constraint-Based Analysis of Security Protocols. In: V. Atluri, P. Liu (eds.) Proceedings of CCS'03, pp. 335–344. ACM Press (2003). Available at <http://www.avispa-project.org>
14. Basin, D., Mödersheim, S., Viganò, L.: OFMC: A Symbolic Model-Checker for Security Protocols. International Journal of Information Security (2004)
15. Biere, A., Cimatti, A., Clarke, E., Zhu, Y.: Symbolic Model Checking without BDDs. In: W.R. Cleaveland (ed.) Proceedings of TACAS'99, LNCS, vol. 1579, pp. 193–207. Springer-Verlag (1999)
16. Blanchet, Podelski: Verification of cryptographic protocols: Tagging enforces termination. TCS: Theoretical Computer Science **333** (2005)
17. Blanchet, B.: An efficient cryptographic protocol verifier based on prolog rules. In: Proceedings of CSFW'01, pp. 82–96. IEEE Computer Society Press (2001)
18. Blanchet, B.: Automatic verification of cryptographic protocols: A logic programming approach (invited talk). In: Proceedings of PPDP'03, pp. 1–3. ACM Press (2003)
19. Blanchet, B.: Automatic Proof of Strong Secrecy for Security Protocols. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 86–100. IEEE Computer Society Press (2004)
20. Blum, A., Furst, M.: Fast planning through planning graph analysis. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 95), pp. 1636–1642 (1995). URL citeseer.nj.nec.com/blum95fast.html
21. Bodei, C., Buchholtz, M., Degano, P., Nielson, F., Nielson, H.R.: Control flow analysis can find new flaws too. In: Proceedings of Workshop on Issues in the Theory of Security (WITS 04) (2004). URL <http://www2.imm.dtu.dk/pubdb/p.php?3058>
22. Bodei, C., Buchholtz, M., Degano, P., Nielson, F., Riis Nielson, H.: Automatic validation of protocol narration. In: Proceedings of CSFW'03, pp. 126–140. IEEE Computer Society Press (2003)
23. Bugliesi, M., Focardi, R., Maffei, M.: Compositional analysis of authentication protocols. In: D.A. Schmidt (ed.) ESOP, *Lecture Notes in Computer Science*, vol. 2986, pp. 140–154. Springer (2004). URL <http://springerlink.metapress.com/openurl.asp?genre=article&issn=0302-9743&volume=2986&page=140>
24. Cervesato, I., Durgin, N.A., Lincoln, P., Mitchell, J.C., Scedrov, A.: A meta-notation for protocol analysis. In: CSFW, pp. 55–69 (1999). URL citeseer.nj.nec.com/cervesato99metanotation.html
25. Chevalier, Y., Compagna, L., Cuellar, J., Hanks Drieslma, P., Mantovani, J., Mödersheim, S., Vigneron, L.: A High Level Protocol Specification Language for Industrial Security-Sensitive Protocols. In: In Proceedings of SAPS'2004 (2004)
26. Chevalier, Y., Vigneron, L.: Automated Unbounded Verification of Security Protocols. In: E. Brinksma, K.G. Larsen (eds.) Proceedings of CAV'02, LNCS 2404, pp. 324–337. Springer-Verlag (2002)
27. Clark, J., Jacob, J.: A Survey of Authentication Protocol Literature: Version 1.0, 17. Nov. 1997 (1997). URL: www.cs.york.ac.uk/~jac/papers/drareview.ps.gz
28. Clarke, E.M., Jha, S., Marrero, W.R.: Partial order reductions for security protocol verification. In: TACAS, pp. 503–518 (2000)
29. Compagna, L.: SAT-based Model-Checking of Security Protocols. Ph.D. thesis, Università degli Studi di Genova and the University of Edinburgh (joint programme) (2005)
30. Corin, R., Etalle, S.: An Improved Constraint-Based System for the Verification of Security Protocols. In: Proceedings of SAS 2002, LNCS 2477, pp. 326–341. Springer-Verlag (2002)
31. Dolev, D., Yao, A.: On the Security of Public-Key Protocols. IEEE Transactions on Information Theory **2**(29) (1983)
32. Donovan, B., Norris, P., Lowe, G.: Analyzing a Library of Security Protocols using Casper and FDR. In: Proceedings of the Workshop on Formal Methods and Security Protocols (1999)
33. Durante, Focardi, Gorrieri: CVS at work: A report on new failures upon some cryptographic protocols. In: MM-MACNS: International Workshop on Methods, Models and Architectures for Network Security, LNCS (2001)
34. Durgin, N., Lincoln, P., Mitchell, J., Scedrov, A.: Multiset rewriting and the complexity of bounded security protocols (2002)
35. Ernst, M.D., Millstein, T.D., Weld, D.S.: Automatic SAT-compilation of Planning Problems. In: Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97), pp. 1169–1177. Morgan Kaufmann (1997)
36. Even, S., Goldreich, O.: On the security of multi-party ping pong protocols. In: Proceedings of 24th IEEE Symposium on Foundations of Computer Science. IEEE Computer Society (1983)
37. Fábrega, F.J.T., Herzog, J.C., Guttman, J.D.: Strand spaces: Why a security protocol is correct? In: Proceedings of the 1998 IEEE Symposium on Security and Privacy, pp. 160–171. IEEE Computer Society Press, New York (1998)
38. Focardi, R., Gorrieri, R., Martinelli, F.: Secrecy in security protocols as non interference. *Electr. Notes Theor. Comput. Sci* **32** (2000). URL <http://www.elsevier.com/gej-ng/31/29/23/57/23/show/Products/notes/index.htm#007>

39. Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., Wilkins, D.: The PDDL Planning Domain Definition Language (1998). The AIPS-98 Planning Competition Comittee
40. Gordon, Jeffrey: Typing correspondence assertions for communication protocols. *TCS: Theoretical Computer Science* **300** (2003)
41. Gordon, A.D., Jeffrey, A.: Authenticity by typing for security protocols. *Journal of Computer Security* **11**(4), 451–520 (2003)
42. Hansen, S.M., Skriver, J., Nielson, H.R.: Using static analysis to validate the SAML single sign-on protocol. In: WITS (2005). URL <http://www2.imm.dtu.dk/pubdb/p.php?3657>
43. Heather, J., Lowe, G., Schneider, S.: How to prevent type flaw attacks on security protocols. In: Proceedings of The 13th Computer Security Foundations Workshop (CSFW'00). IEEE Computer Society Press (2000)
44. ISO/IEC: ISO/IEC 9798-3: Information technology - Security techniques - Entity authentication - Part 3: Mechanisms using digital signature techniques (1997)
45. Kautz, H., McAllester, H., Selman, B.: Encoding Plans in Propositional Logic. In: L.C. Aiello, J. Doyle, S. Shapiro (eds.) *KR'96: Principles of Knowledge Representation and Reasoning*, pp. 374–384. Morgan Kaufmann (1996)
46. Kautz, H.A., Selman, B.: Planning as satisfiability. In: *ECAI*, pp. 359–363 (1992)
47. Lifschitz, V.: Answer set programming and plan generation. *Artificial Intelligence* **138**(1-2), 39–54 (2002). URL [http://dx.doi.org/10.1016/S0004-3702\(02\)00186-8](http://dx.doi.org/10.1016/S0004-3702(02)00186-8)
48. Lin, F., Zhao, Y.: ASSAT: computing answer sets of a logic program by SAT solvers. *Artif. Intell.* **157**(1-2), 115–137 (2004)
49. Lowe, G.: Breaking and Fixing the Needham-Shroeder Public-Key Protocol Using FDR. In: T. Margaria, B. Steffen (eds.) *Proceedings of TACAS'96, LNCS 1055*, pp. 147–166. Springer-Verlag (1996)
50. Lowe, G.: A hierarchy of authentication specifications. In: *Proceedings of the 10th IEEE Computer Security Foundations Workshop (CSFW'97)*, pp. 31–43. IEEE Computer Society Press (1997)
51. Lowe, G.: Towards a completeness result for model checking of security protocols. In: *Proceedings of CSFW'98*. IEEE Computer Society Press (1998). URL citeseer.nj.nec.com/article/lowe98towards.html
52. Marrero, W.R., Clarke, E.M., Jha, S.: Model checking for security protocols. In: *DIMACS Workshop on Design and Formal Verification of Security Protocols* (1997)
53. Martinelli, F.: Analysis of security protocols as open systems. *Theoretical Computer Science* **290**(1), 1057–1106 (2003)
54. Millen, J.K., Shmatikov, V.: Constraint solving for bounded-process cryptographic protocol analysis. In: *Proceedings of the ACM Conference on Computer and Communications Security CCS'01*, pp. 166–175 (2001)
55. Mitchell, J.C., Mitchell, M., Stern, U.: Automated Analysis of Cryptographic Protocols Using Murphi. In: *Proceedings of IEEE Symposium on Security and Privacy*, pp. 141–153 (1997)
56. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: Engineering an Efficient SAT Solver. In: *Proceedings of the 38th Design Automation Conference (DAC'01)* (2001). URL <http://www.ee.princeton.edu/~chaff/DAC2001v56.pdf>
57. Needham, R.M., Schroeder, M.D.: Using Encryption for Authentication in Large Networks of Computers. *Tech. Rep. CSL-78-4*, Xerox Palo Alto Research Center, Palo Alto, CA, USA (1978). Reprinted June 1982
58. Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. *Ann. Math. Artif. Intell.* **25**(3-4), 241–273 (1999)
59. Paulson, L.C.: The Inductive Approach to Verifying Cryptographic Protocols. *Journal of Computer Security* **6**(1), 85–128 (1998)
60. Rintanen, J., Heljanko, K., Niemelä, I.: Planning as satisfiability: parallel plans and algorithms for plan search (2005). Submitted for journal publication. Earlier version: Technical report 216, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, 2005. Available at <http://www.informatik.uni-freiburg.de/~rintanen>
61. Rusinowitch, M., Turuani, M.: Protocol Insecurity with Finite Number of Sessions is NP-complete. In: *Proceedings of the 14th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press (2001)
62. Song, D., Berezin, S., Perrig, A.: Athena: a novel approach to efficient automatic security protocol analysis. *Journal of Computer Security* **9**, 47–74 (2001)
63. Sperschnieder, V., Antoniou, G.: Logic: A Foundation for Computer Science. Addison-Wesley: Reading, MA (1991)
64. Turuani, M.: Sécurité des Protocoles Cryptographiques: Décidabilité et Complexité. Ph.D. thesis, Université Henri Poincaré, Nancy (2003)



Alessandro Armando is associate professor at DIST where he got his PhD in 1994. His appointments include a research position at the University of Edinburgh and one at INRIA-Lorraine. He has contributed to the development and combination of automated reasoning techniques and their application to automatic program analysis and synthesis, and to the automatic analysis of security protocols. He is the author of more than 70 research works in international journals and conferences. He has coordinated a number of national and international research projects, including the AVISPA Project.



Luca Compagna holds a doctoral degree from the University of Genoa and the University of Edinburgh in the context of a bilateral agreement for the realisation of a PhD in cooperation (2005). He contributed to the development of a number of research projects on automatic security protocol analysis, including the AVISPA Project, and he has published various scientific publications in the area of model checking of security protocols and automated reasoning. Since March 2006 he is a senior scientist at SAP Labs France for the Security and Trust Research Program.

A Appendix: Proofs of the Main Results

A.1 Proofs of the results of Section 3.3

Lemma 1 *Let χ be a run of $\Gamma = \langle \mathcal{I}, \mathcal{R} \rangle$ and χ' be a sequence of rule instances of Γ . If χ' is rule equivalent to χ then also χ' is a run of Γ and $\text{app}_{\chi'}(\mathcal{I}) \sim \text{app}_{\chi}(\mathcal{I})$.*

Proof The proof is by induction on the length of χ . The key steps of the proof amount to showing for any rule instance $\langle \rho, \sigma \rangle$ with $L \xrightarrow{\rho} \exists \mathbf{y}. R$ that (i) if $\langle \rho, \sigma \rangle$ is applicable to S then it is also applicable to all the states that are equivalent to S (this follows from the observation that $L\sigma$ does not contain any fresh constant); (ii) $\text{app}_{\langle \rho, \sigma \rangle}(S) \sim \text{app}_{\langle \rho, \sigma' \rangle}(S)$ for all states S and ground substitutions σ' .

Theorem 1 *Let χ be a solution to $\Xi = \langle \Gamma, \mathcal{B} \rangle$ and χ' be a sequence of rule instances of Γ . If χ' is rule equivalent to χ then also χ' is a solution to Ξ .*

Proof The proof is based on Lemma 1 and the observation that the set of bad states is closed under renaming of fresh constants.

A.2 Proofs of the results of Section 4.2

Lemma 2 *Let ρ_1, \dots, ρ_n be composable rules, let $\rho_{k_1}, \dots, \rho_{k_n}$ be a linearisation of $\rho_1 \parallel \dots \parallel \rho_n$, and let $S_0 \supseteq \bigcup_{i=1}^n \text{pre}(\rho_i)$, then $\text{app}_{\rho_{k_1}; \dots; \rho_{k_n}}(S_0) = \text{app}_{\rho_1 \parallel \dots \parallel \rho_n}(S_0)$.*

Proof The proof is by induction on n . Since the base case is trivial, here we focus on the step case that is proved as shown in Figure 8.

Lemma 4 *Let ρ_1, \dots, ρ_n be composable rules. If ρ_i does not interfere with ρ_j for $i, j = 1, \dots, n$, then all permutations of ρ_1, \dots, ρ_n are linearisations of $\rho_1 \parallel \dots \parallel \rho_n$.*

Proof Let $\rho_{k_1}, \dots, \rho_{k_n}$ be a generic permutation of ρ_1, \dots, ρ_n . We must show that $\rho_{k_{i+1}}$ is applicable in S_i for $i = 0, \dots, n-1$, where $S_0 \supseteq \bigcup_{i=1}^n \text{pre}(\rho_i)$ and $S_{i+1} = \text{app}_{\rho_{k_{i+1}}}(S_i)$ for $i = 0, \dots, n-1$. To this end we prove a more general fact, namely that ρ_{k_j} is applicable in S_i for all $j = i+1, \dots, n$ and all $i = 0, \dots, n-1$, from which the above fact can be readily inferred by considering the cases in which $j = i+1$. The proof is by induction on i . The base case (i.e. $i = 0$) boils down to showing that ρ_{k_j} is applicable in S_0 for all $j = 1, \dots, n$. This trivially holds since $S_0 \supseteq \bigcup_{i=1}^n \text{pre}(\rho_i) = \bigcup_{j=1}^n \text{pre}(\rho_{k_j})$. For the step case we must show that ρ_{k_j} is applicable in S_{i+1} for all $j = i+2, \dots, n$. Let j be such that $i+1 < j \leq n$. By induction hypothesis it readily follows that ρ_{k_j} is applicable in S_i or, in other words, that $\text{pre}(\rho_{k_j}) \subseteq S_i$. Since, by hypothesis, ρ_{k_j} and $\rho_{k_{i+1}}$ do not interfere, we know that $\text{pre}(\rho_{k_j}) \cap \text{del}(\rho_{k_{i+1}}) = \emptyset$. From this we can conclude that $\text{pre}(\rho_{k_j}) \subseteq S_i \setminus \text{del}(\rho_{k_{i+1}})$ and therefore also that $\text{pre}(\rho_{k_j}) \subseteq (S_i \setminus \text{del}(\rho_{k_{i+1}})) \cup \text{add}(\rho_{k_{i+1}}) = S_{i+1}$. Thus ρ_{k_j} is applicable in S_i and this concludes the proof.

A.3 Proofs of the results of Section 4.3

Theorem 3 (Soundness) *If M is a model of $\llbracket \Gamma \rrbracket_k$ for $k \geq 0$, $S_i = \{f \in \mathcal{F} : f^i \in M\}$ for $i = 0, \dots, k$, and $\rho_i = \|\{\rho \in \mathcal{L} : \rho^i \in M\}\}$, for $i = 0, \dots, k-1$ then $S_0 \rho_0 S_1 \dots S_{k-1} \rho_{k-1} S_k$ is an initialised path of $\hat{\Gamma}$.*

Proof The proof is by induction on k . The base case is easy as $\llbracket \Gamma \rrbracket_0$ is logically equivalent to $I(\mathbf{f}^0)$ and by requirement R1 we know that $\{f \in \mathcal{F} : f^0 \in M\} = S_0 \in \mathcal{I}$. Therefore S_0 is a initialised path of $\hat{\Gamma}$. Let us now consider the step case. Let M be a model of $\llbracket \Gamma \rrbracket_k$ for $k > 0$. Since $\llbracket \Gamma \rrbracket_k = \llbracket \Gamma \rrbracket_{k-1} \wedge T_{k-1}(\mathbf{f}^{k-1}, \boldsymbol{\rho}^{k-1}, \mathbf{f}^k)$, then M is also a model of $\llbracket \Gamma \rrbracket_{k-1}$ and of $T_{k-1}(\mathbf{f}^{k-1}, \boldsymbol{\rho}^{k-1}, \mathbf{f}^k)$. By induction hypothesis we know that $S_0 \rho_0 S_1 \dots S_{k-2} \rho_{k-2} S_{k-1}$ is an initialised path of $\hat{\Gamma}$, where $S_i = \{f \in \mathcal{F} : f^i \in M\}$ for $i = 0, \dots, k-1$ and ρ_i is the parallel composition of the rules in $\{\rho \in \mathcal{L} : \rho^i \in M\}$ for $i = 0, \dots, k-2$. By part (a) of requirement R2 we know that the parallel composition ρ_{k-1} of the rules in $\{\rho \in \mathcal{L} : \rho^{k-1} \in M\}$ is linearisable and that $\text{app}_{\rho_{k-1}}(S_{k-1}) = \{f \in \mathcal{F} : f^k \in M\} = S_k$. We can therefore conclude that $S_0 \rho_0 S_1 \dots S_{k-1} \rho_{k-1} S_k$ is an initialised path of $\hat{\Gamma}$.

Theorem 4 (Completeness) *If $\pi_k = S_0 \rho_0 S_1 \dots S_{k-1} \rho_{k-1} S_k$ is an initialised path of $\hat{\Gamma}$, then there exists a model M of $\llbracket \Gamma \rrbracket_k$ such that $\pi(M) = \pi_k$.*

Proof The proof is by induction on k . The base case is easy as $\pi_0 = S_0 \in \mathcal{I}$ and from R1 we know that $M = \{f^0 : f \in S_0\}$ is a model of $I(\mathbf{f}^0)$. But $\llbracket \Gamma \rrbracket_0$ is logically equivalent to $I(\mathbf{f}^0)$ and therefore M is a model of $\llbracket \Gamma \rrbracket_k$ for $k = 0$. Moreover $\pi(M) = \{f : f^0 \in M\} = S_0 = \pi_0$ is an initialised path of $\hat{\Gamma}$. Let us now consider the step case. Let $\pi_k = S_0 \rho_0 S_1 \dots S_{k-1} \rho_{k-1} S_k$ be an initialised path of $\hat{\Gamma}$ where $\rho_i \in \mathcal{L}$ is the parallel composition of the linearisable rules in $A_i \subseteq \mathcal{L}$ for $i = 0, \dots, k-1$. Clearly also $\pi_{k-1} = S_0 \rho_0 S_1 \dots S_{k-2} \rho_{k-2} S_{k-1}$ is an initialised path of $\hat{\Gamma}$. By induction hypothesis there exists a model M' of $\llbracket \Gamma \rrbracket_{k-1}$ such that $\pi(M') = S_0 \rho_0 S_1 \dots S_{k-2} \rho_{k-2} S_{k-1}$ is an initialised path of $\hat{\Gamma}$. Since S_{k-1} is reachable in $k-1$ steps in $\hat{\Gamma}$, $\text{pre}(\rho_{k-1}) \subseteq S_{k-1}$, and $\text{app}_{\rho_{k-1}}(S_{k-1}) = S_k$, by part (b) of requirement R2 we know that $M'' = \{f^{k-1} : f \in S_{k-1}\} \cup \{\rho^{k-1} : \rho \in A_{k-1}\} \cup \{f^k : f \in S_k\}$ is a model of $T_{k-1}(\mathbf{f}^{k-1}, \boldsymbol{\rho}^{k-1}, \mathbf{f}^k)$. From this fact we can conclude that $M = M' \cup M''$ is a model of $\llbracket \Gamma \rrbracket_k = \llbracket \Gamma \rrbracket_{k-1} \wedge T(\mathbf{f}^{k-1}, \boldsymbol{\rho}^{k-1}, \mathbf{f}^k)$ and is such that $\pi(M) = S_0 \rho_0 S_1 \dots S_{k-2} \rho_{k-2} S_{k-1} \rho_{k-1} S_k$ is an initialised path of $\hat{\Gamma}$.

A.4 Proofs of the results of Section 4.3.1

We now show that $I(\mathbf{f}^0)$ enjoys requirement R1 (Lemma 5) and $T_i(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$ enjoys part (a) and part (b) of requirement R2 (Lemma 6 and Lemma 7 respectively). From this it readily follows by Theorem 3 that the formulae (22)-(28) encode faithfully the behaviours of $\hat{\Gamma}$ of length k and hence also the soundness and completeness of the linear encoding (Theorem 5). We recall that we write $\bar{T}(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$ in place of $T_i(\mathbf{f}^i, \boldsymbol{\rho}^i, \mathbf{f}^{i+1})$. Moreover, since Ξ is ground we assume that $\mathcal{B} = \langle \mathcal{G}, \emptyset \rangle$ (with \mathcal{G} ground) and define $\llbracket \mathcal{B} \rrbracket_k = \mathcal{G}^k$ where \mathcal{G}^k is obtained from \mathcal{G} by replacing every fact p with p^k .

Lemma 5 (Soundness and Completeness of $I(\mathbf{f}^0)$) *$I(\mathbf{f}^0)$ is such that if M is a model of $I(\mathbf{f}^0)$ then $\{f \in \mathcal{F} : f^0 \in M\} \in \mathcal{I}$ and, conversely, if $S \in \mathcal{I}$ then $\{f^0 : f \in S\}$ is a model of $I(\mathbf{f}^0)$.*

Proof This proof readily follows from the definition of $I(\mathbf{f}^0)$ given in (22).

$$\begin{aligned}
& \text{app}_{\rho_{k_1}, \dots, \rho_{k_n}}(S) \\
= & \text{app}_{\rho_{k_n}}(\text{app}_{\rho_{k_1}, \dots, \rho_{k_{n-1}}}(S)) && \text{[by definition]} \\
= & \text{app}_{\rho_{k_n}}(\text{app}_{\rho_{k_1} \parallel \dots \parallel \rho_{k_{n-1}}}(S)) && \text{[by induction hypothesis]} \\
= & (((S \setminus \bigcup_{i=1}^{n-1} \text{del}(\rho_{k_i})) \cup \bigcup_{i=1}^{n-1} \text{add}(\rho_{k_i})) \setminus \text{del}(\rho_{k_n})) \cup \text{add}(\rho_{k_n}) && \text{[by definition]} \\
= & (((S \setminus \bigcup_{i=1}^{n-1} \text{del}(\rho_{k_i})) \setminus \text{del}(\rho_{k_n})) \cup \bigcup_{i=1}^{n-1} \text{add}(\rho_{k_i})) \cup \text{add}(\rho_{k_n}) \\
& \hspace{10em} [\rho_1, \dots, \rho_n \text{ are composable and thus } \text{del}(\rho_{k_n}) \cap \bigcup_{i=1}^{n-1} \text{add}(\rho_{k_i}) = \emptyset] \\
= & (S \setminus \bigcup_{i=1}^n \text{del}(\rho_{k_i})) \cup \bigcup_{i=1}^n \text{add}(\rho_{k_i}) && \text{[by simplification]} \\
= & \text{app}_{\rho_1 \parallel \dots \parallel \rho_n}(S) && \text{[by definition]}
\end{aligned}$$

Fig. 8 Proof of the step case of Lemma 2

Lemma 6 (Soundness of $\overline{T}(\mathbf{f}^i, \rho^i, \mathbf{f}^{i+1})$) *If M is a model of $\overline{T}(\mathbf{f}^i, \rho^i, \mathbf{f}^{i+1})$ then the rules in $\Lambda = \{\rho \in \mathcal{L} : \rho^i \in M\}$ are composable and their parallel composition $\parallel \Lambda$ is linearisable and such that $\text{app}_{\parallel \Lambda}(S) = S'$ for $S = \{f \in \mathcal{F} : f^i \in M\}$ and $S' = \{f \in \mathcal{F} : f^{i+1} \in M\}$ for $i = 0, \dots, k-1$.*

Proof If M is a model of $\overline{T}(\mathbf{f}^i, \rho^i, \mathbf{f}^{i+1})$, then M is a model of (23)-(28). The truth of (24) and (25) in M implies that the effects of the rules in $\Lambda = \{\rho \in \mathcal{L} : \rho^i \in M\}$ are such that $\{f^{i+1} : f \in \text{add}(\rho)\} \subset M$, $\{f^{i+1} : f \in \text{del}(\rho)\} \cap M = \emptyset$, and therefore they do not contradict each other, i.e. $\bigcup_{\rho \in \Lambda} \text{add}(\rho) \cap \bigcup_{\rho \in \Lambda} \text{del}(\rho) = \emptyset$. Hence the rules in Λ are composable and because of (28) they are pairwise non-interfering and thus linearisable. Moreover for any $\rho \in \Lambda$, the formulae (23) state that $\{f^i : f \in \text{pre}(\rho)\} \subset M$. As a consequence, the union of the preconditions of the rules in Λ are contained in $S = \{f \in \mathcal{F} : f^i \in M\}$ and therefore $\parallel \Lambda$ is applicable in S .

To conclude the proof we must thus show that $S' = \{f \in \mathcal{F} : f^{i+1} \in M\}$ is equal to $(S \setminus \bigcup_{\rho \in \Lambda} \text{del}(\rho)) \cup \bigcup_{\rho \in \Lambda} \text{add}(\rho)$. This boils down to proving that $p \in (S \setminus \bigcup_{\rho \in \Lambda} \text{del}(\rho)) \cup \bigcup_{\rho \in \Lambda} \text{add}(\rho)$ if, and only if, $p^{i+1} \in M$. The proof is articulated in the following two cases:

Case 1 (if): Let $p \in (S \setminus \bigcup_{\rho \in \Lambda} \text{del}(\rho)) \cup \bigcup_{\rho \in \Lambda} \text{add}(\rho)$, i.e. either $p \in S \setminus \bigcup_{\rho \in \Lambda} \text{del}(\rho)$ or $p \in \bigcup_{\rho \in \Lambda} \text{add}(\rho)$, we must prove that $p^{i+1} \in M$.

Case 1.1: Let $p \in S \setminus \bigcup_{\rho \in \Lambda} \text{del}(\rho)$, i.e. $p \in S$ and $p \notin \text{del}(\rho)$ for all $\rho \in \Lambda$ (and thus $\rho^i \in M$). Therefore every rule $\tilde{\rho}$ that deletes p is not in Λ , i.e. $\tilde{\rho}^i \notin M$. As a consequence the conclusion of the first explanatory frame axiom (26) for fact p is false in M . Moreover, since $p \in S$ from the definition of S it follows that $p^i \in M$. From this we conclude that the truth value of (26) in M requires $p^{i+1} \in M$.

Case 1.2: Let $p \in \bigcup_{\rho \in \Lambda} \text{add}(\rho)$, i.e. there exists a rule $\rho \in \Lambda$ (and thus $\rho^i \in M$) such that $p \in \text{add}(\rho)$. Because of this, the truth of $(\rho^i \supset p^{i+1})$, i.e. one of the formulae (24), in M implies that $p^{i+1} \in M$.

Case 2 (only if): Let $p^{i+1} \in M$, we must prove that $p \in (S \setminus \bigcup_{\rho \in \Lambda} \text{del}(\rho)) \cup \bigcup_{\rho \in \Lambda} \text{add}(\rho)$ i.e. either (i) $p \in S$ and $p \notin \bigcup_{\rho \in \Lambda} \text{del}(\rho)$, or (ii) $p \in \bigcup_{\rho \in \Lambda} \text{add}(\rho)$. Since $p^{i+1} \in M$, then for all $\tilde{\rho}$ such that $p \in \text{del}(\tilde{\rho})$ the truth of $(\tilde{\rho}^i \supset \neg p^{i+1})$, i.e. one of the formulae (25), in M implies that $\tilde{\rho}^i \notin M$. Because of this and from the definition of Λ , we derive that $\tilde{\rho} \notin \Lambda$ and therefore that $p \notin \bigcup_{\rho \in \Lambda} \text{del}(\rho)$. To conclude the proof we must show that either $p \in S$ or $p \in \bigcup_{\rho \in \Lambda} \text{add}(\rho)$. This is equivalent to proving that either

$p^i \in M$ or there exists a rule ρ such that $\rho^i \in M$ and $p \in \text{add}(\rho)$. To demonstrate this it suffices to observe that, for M to satisfy $(\neg p^i \supset \bigvee \{\rho^i \mid \rho \in \mathcal{L}, p \in \text{add}(\rho)\})$ —i.e. the formula constructed through schema (27) and simplified by using the assumption $p^{i+1} \in M$ —it must hold that either $p^i \in M$ or there exists a rule ρ such that $\rho^i \in M$ and $p \in \text{add}(\rho)$.

Lemma 7 (Completeness of $\overline{T}(\mathbf{f}^i, \rho^i, \mathbf{f}^{i+1})$) *If $S \subseteq \mathcal{F}$ is reachable in i steps in $\hat{\Gamma}$, the rules in $\Lambda = \{\rho_1, \dots, \rho_n\} \subseteq \mathcal{L}$ are linearisable and their parallel composition $\parallel \Lambda$ is applicable in S , and $S' = \text{app}_{\parallel \Lambda}(S)$, then $M = \{f^i : f \in S\} \cup \{f^{i+1} : f \in S'\} \cup \{\rho_1^i, \dots, \rho_n^i\}$ is a model of $\overline{T}(\mathbf{f}^i, \rho^i, \mathbf{f}^{i+1})$ for $i = 0, \dots, k-1$.*

Proof Let $\tilde{\rho}$ and f be a rule and a fact respectively. By definition of M and Λ , it is immediate to see that (i) $\tilde{\rho} \in \Lambda$ if and only if $\tilde{\rho}^i \in M$, (ii) $f \in S$ if and only if $f^i \in M$, and (iii) $f \in S'$ if and only if $f^{i+1} \in M$.

To show that M is model of $\overline{T}(\mathbf{f}^i, \rho^i, \mathbf{f}^{i+1})$, we must prove that M satisfies (23)-(28).

The formulae (23), (24), and (25) are true in M if and only if for every rule $\tilde{\rho} \in \mathcal{L}$ if $\tilde{\rho}^i \in M$ (i.e. $\tilde{\rho} \in \Lambda$), then (a) any fact $f \in \text{pre}(\tilde{\rho})$ is such that $f^i \in M$ (i.e. $f \in S$), (b) any fact $f \in \text{add}(\tilde{\rho})$ is such that $f^{i+1} \in M$ (i.e. $f \in S'$), and (c) any fact $f \in \text{del}(\tilde{\rho})$ is such that $f^{i+1} \notin M$ (i.e. $f \notin S'$). This is trivially satisfied in M by any rule $\tilde{\rho}$ that is not in Λ . Let us consider the case in which $\tilde{\rho} \in \Lambda$. By hypothesis the parallel composition $\parallel \Lambda$ is applicable in S (i.e. $\text{pre}(\parallel \Lambda) = \bigcup_{\rho \in \Lambda} \text{pre}(\rho) \subseteq S$), and the application of $\parallel \Lambda$ leads to $S' = (S \setminus \bigcup_{\rho \in \Lambda} \text{del}(\rho)) \cup \bigcup_{\rho \in \Lambda} \text{add}(\rho)$. Since $\tilde{\rho} \in \Lambda$, we have that $\text{pre}(\tilde{\rho}) \subseteq \text{pre}(\parallel \Lambda) \subseteq S$ and thus (a) is proved. Similarly $\text{add}(\tilde{\rho}) \subseteq S'$ and therefore also (b) is demonstrated. Finally, to prove (c) it suffices to show that $\text{del}(\tilde{\rho}) \cap S' = \emptyset$, i.e. both $\text{del}(\tilde{\rho}) \cap (S \setminus \bigcup_{\rho \in \Lambda} \text{del}(\rho)) = \emptyset$ and $\text{del}(\tilde{\rho}) \cap \bigcup_{\rho \in \Lambda} \text{add}(\rho) = \emptyset$. The former simply derives from the fact that $\tilde{\rho} \in \Lambda$ and thus $\text{del}(\tilde{\rho}) \subseteq \bigcup_{\rho \in \Lambda} \text{del}(\rho)$. By definition of parallel composition $\bigcup_{\rho \in \Lambda} \text{del}(\rho) \cap \bigcup_{\rho \in \Lambda} \text{add}(\rho) = \emptyset$ and since $\text{del}(\tilde{\rho}) \subseteq \bigcup_{\rho \in \Lambda} \text{del}(\rho)$ it follows immediately also the latter.

Formula (26) is true in M if and only if for every fact $f \in \mathcal{F}$ if $f^i \in M$ (i.e. $f \in S$) and $f^{i+1} \notin M$ (i.e. $f \notin S'$), then at least one of the rules, say $\tilde{\rho}$, that deletes f must be such that $\tilde{\rho}^i \in M$ (i.e. $\tilde{\rho} \in \Lambda$). It follows immediately that $M \models (26)$ holds for every fact f such that either $f \notin S$ or $f \in S'$. Let us consider the case $f \in S$ and $f \notin S'$. By hypothesis $S' = (S \setminus \bigcup_{\rho \in \Lambda} \text{del}(\rho)) \cup \bigcup_{\rho \in \Lambda} \text{add}(\rho)$ and then, by applying the steps of Figure 9, we conclude that for every fluent f such

that $f \in S$ and $f \notin S'$, we have that $f \in \bigcup_{\rho \in \Lambda} \text{del}(\rho)$, i.e. there exists at least one rule $\tilde{\rho} \in \Lambda$ such that $f \in \text{del}(\tilde{\rho})$, and thus (26) is true in M .

The proof of $M \models (27)$ proceeds in a dual way.

To conclude the overall proof we must demonstrate that $M \models (28)$. This means to show that for all pair of rules $\rho_1, \rho_2 \in \mathcal{L}$ such that $\rho_1 \oplus \rho_2$ either $\rho_1^i \notin M$ (i.e. $\rho_1 \notin \Lambda$) or $\rho_2^i \notin M$ (i.e. $\rho_2 \notin \Lambda$). This is a simple consequence of the fact that the rules in Λ are linearisable, i.e. they are pairwise non-interfering.

Theorem 5 (Soundness and Completeness of the Linear Encoding) *If M is a model of $\llbracket \Xi \rrbracket_k$ for $k \geq 0$, then $S_0 \rho_0 S_1 \cdots S_{k-1} \rho_{k-1} S_k$ is a (partial-order) solution of $\hat{\Xi}$, where $S_i = \{f \in \mathcal{F} : f^i \in M\}$ for $i = 0, \dots, k$ and ρ_i is the parallel composition of the rules in $\{\rho \in \mathcal{L} : \rho^i \in M\}$ for $i = 0, \dots, k-1$. Conversely, if $\pi_k = S_0 \rho_0 S_1 \cdots S_{k-1} \rho_{k-1} S_k$ is a (partial-order) solution of $\hat{\Xi}$, then there exists a model M of $\llbracket \Xi \rrbracket_k$ such that $\pi(M) = \pi_k$.*

Proof Soundness readily follows from Lemma 5, Lemma 6 and Theorem 3. Similarly, completeness follows from Lemma 5, Lemma 7, and Theorem 4.

A.5 Proofs of the results of Section 4.3.2

Lemma 8 (Soundness and Completeness of $I(\mathbf{f}^0)$) *$I(\mathbf{f}^0)$ is such that if M is a model of $I(\mathbf{f}^0)$ then $\{f \in \mathcal{F} : f^0 \in M\} \in \mathcal{I}$ and, conversely, if $S \in \mathcal{I}$ then $\{f^0 : f \in S\}$ is a model of $I(\mathbf{f}^0)$.*

Proof This proof readily follows from the definition of $I(\mathbf{f}^0)$ given in (37).

Lemma 9 (Soundness of $T_i(\mathbf{f}^i, \rho^i, \mathbf{f}^{i+1})$) *If M is a model of $T_i(\mathbf{f}^i, \rho^i, \mathbf{f}^{i+1})$ then the rules in $\Lambda = \{\rho \in \mathcal{L} : \rho^i \in M\}$ are composable and their parallel composition $\parallel \Lambda$ is linearisable and such that $\text{app}_{\parallel \Lambda}(S) = S'$ for $S = \{f \in \mathcal{F} : f^i \in M\}$ and $S' = \{f \in \mathcal{F} : f^{i+1} \in M\}$ for $i = 0, \dots, k-1$.*

Proof It must be noted that the formula $T_i(\mathbf{f}^i, \rho^i, \mathbf{f}^{i+1})$ is defined over a subset of the atoms in $T^L(\mathbf{f}^i, \rho^i, \mathbf{f}^{i+1})$ and namely over those propositions occurring in $\{p^i : p \in N_f^i\} \cup \{p^i : p \in N_\rho^i\} \cup \{p^{i+1} : p \in N_f^{i+1}\}$. As a consequence Λ , S , and S' , that by hypothesis are defined in terms of the model M , are such that $\Lambda \subseteq N_\rho^i$, $S \subseteq N_f^i$, and $S' \subseteq N_f^{i+1}$. Therefore for any rule $\rho \in \Lambda$ we have that $\text{pre}(\rho)$, $\text{add}(\rho)$, and $\text{del}(\rho)$ can be used interchangeably with $\{f : \rho \xrightarrow[\text{pre}]{} f\}$, $\{f : \rho \xrightarrow[\text{add}]{} f\}$, and $\{f : \rho \xrightarrow[\text{del}]{} f\}$ respectively. Given this the proof goes along the lines of Lemma 6 in which (23)-(25), (26), and (28) are replaced by (38)-(40), (41), and (44) respectively. The only significant difference concerns “case 2” of Lemma 6 that we describe entirely here.

Case 2 (only if): Let $p^{i+1} \in M$, we must prove that $p \in (S \setminus \bigcup_{\rho \in \Lambda} \text{del}(\rho)) \cup \bigcup_{\rho \in \Lambda} \text{add}(\rho)$ i.e. either both $p \in S$ and $p \notin \bigcup_{\rho \in \Lambda} \text{del}(\rho)$, or $p \in \bigcup_{\rho \in \Lambda} \text{add}(\rho)$. Since $p^{i+1} \in M$, then for all $\tilde{\rho}$ such that $p \in \text{del}(\tilde{\rho})$ the truth of $(\tilde{\rho}^i \supset \neg p^{i+1})$, i.e. the formula constructed through axiom (40), in M requires $\tilde{\rho}^i \notin M$. Because of this and from the definition of Λ , we derive that $\tilde{\rho} \notin \Lambda$ and therefore that $p \notin \bigcup_{\rho \in \Lambda} \text{del}(\rho)$. To conclude the proof we must show

that either $p \in S$ or $p \in \bigcup_{\rho \in \Lambda} \text{add}(\rho)$, that is equivalent to prove that either $p^i \in M$ or there exists a rule ρ such that $\rho^i \in M$ and $p \in \text{add}(\rho)$. To prove this we must notice that $p^{i+1} \in M$ if and only if $p \in S' = \{f \in \mathcal{F} : f^{i+1} \in M\} \subseteq N_f^{i+1}$. Moreover by definition of planning graph we know that $N_f^i \subseteq N_f^{i+1}$ and therefore $N_f^{i+1} = N_f^i \cup N_f^{i+1} \setminus N_f^i$. We thus consider the following two cases.

Case 2.1: Let $p \in N_f^i$, the truth of the formula constructed through axiom (42) and simplified by using the assumption $p^{i+1} \in M$ implies that either $p^i \in M$ or there exists a rule ρ such that $\rho^i \in M$ and $p \in \text{add}(\rho)$.

Case 2.2: Let $p \in N_f^{i+1} \setminus N_f^i$. This implies trivially that $p \notin S$ (since by definition $S \subseteq N_f^i$) and thus we must prove that there exists a rule ρ such that $\rho^i \in M$ and $p \in \text{add}(\rho)$. This is simply derived from the truth value of the formula constructed through axiom (43) and simplified by using the assumption $p^{i+1} \in M$.

Lemma 10 (Completeness of $T_i(\mathbf{f}^i, \rho^i, \mathbf{f}^{i+1})$) *If $S \subseteq \mathcal{F}$ is reachable in i steps in $\hat{\Gamma}$, the rules in $\Lambda = \{\rho_1, \dots, \rho_n\} \subseteq \mathcal{L}$ are linearisable and their parallel composition $\parallel \Lambda$ is applicable in S , and $S' = \text{app}_{\parallel \Lambda}(S)$, then $M = \{f^i : f \in S\} \cup \{f^{i+1} : f \in S'\} \cup \{\rho_1^i, \dots, \rho_n^i\}$ is a model of $T_i(\mathbf{f}^i, \rho^i, \mathbf{f}^{i+1})$ for $i = 0, \dots, k-1$.*

Proof The proof goes along the lines of Lemma 7. The differences are minor but many and therefore we prefer to present the overall proof here.

Let $\tilde{\rho}$ and f be a rule and a fact respectively. First of all notice that by definition of planning graph the reachability of S in i steps implies that $S \subseteq N_f^i$. Similarly the reachability of S' in one step from S by applying the rules in Λ implies that $S' \subseteq N_f^{i+1}$, $\Lambda \subseteq N_\rho^i$, and that for each $\rho \in N_\rho^i$ $\text{pre}(\rho)$, $\text{add}(\rho)$, and $\text{del}(\rho)$ can be used interchangeably with $\{f : \rho \xrightarrow[\text{pre}]{} f\}$, $\{f : \rho \xrightarrow[\text{add}]{} f\}$, and $\{f : \rho \xrightarrow[\text{del}]{} f\}$ respectively. Because of this and by definition of M and Λ , it is immediate to see that (i) $\tilde{\rho} \in \Lambda$ if and only if $\tilde{\rho}^i \in M$, (ii) $f \in S$ if and only if $f^i \in M$, and (iii) $f \in S'$ if and only if $f^{i+1} \in M$.

To show that M is model of $\overline{T}(\mathbf{f}^i, \rho^i, \mathbf{f}^{i+1})$, we must prove that M satisfies (38)-(44).

The formulae (38), (39), and (40) are true in M if and only if for every rule $\tilde{\rho} \in N_\rho^i$ if $\tilde{\rho}^i \in M$ (i.e. $\tilde{\rho} \in \Lambda$), then (a) any fact $f \in \text{pre}(\tilde{\rho})$ is such that $f^i \in M$ (i.e. $f \in S$), (b) any fact $f \in \text{add}(\tilde{\rho})$ is such that $f^{i+1} \in M$ (i.e. $f \in S'$), and (c) any fact $f \in \text{del}(\tilde{\rho})$ is such that $f^{i+1} \notin M$ (i.e. $f \notin S'$). This is trivially satisfied in M by any rule $\tilde{\rho}$ that is in N_ρ^i but not in Λ . Let us now consider the case in which $\tilde{\rho} \in \Lambda$. By hypothesis the parallel composition $\parallel \Lambda$ is applicable in S (i.e. $\text{pre}(\parallel \Lambda) = \bigcup_{\rho \in \Lambda} \text{pre}(\rho) \subseteq S$), and the application of $\parallel \Lambda$ leads to $S' = (S \setminus \bigcup_{\rho \in \Lambda} \text{del}(\rho)) \cup \bigcup_{\rho \in \Lambda} \text{add}(\rho)$. Since $\tilde{\rho} \in \Lambda$, we have that $\text{pre}(\tilde{\rho}) \subseteq \text{pre}(\parallel \Lambda) \subseteq S$ and thus (a) is proved. Similarly $\text{add}(\tilde{\rho}) \subseteq S'$ and therefore also (b) is demonstrated. Finally, to prove (c) it suffices to show that $\text{del}(\tilde{\rho}) \cap S' = \emptyset$, i.e. both $\text{del}(\tilde{\rho}) \cap (S \setminus \bigcup_{\rho \in \Lambda} \text{del}(\rho)) = \emptyset$ and $\text{del}(\tilde{\rho}) \cap \bigcup_{\rho \in \Lambda} \text{add}(\rho) = \emptyset$. The former simply derives from the fact that $\tilde{\rho} \in \Lambda$ and thus $\text{del}(\tilde{\rho}) \subseteq \bigcup_{\rho \in \Lambda} \text{del}(\rho)$. By definition of parallel composition $\bigcup_{\rho \in \Lambda} \text{del}(\rho) \cap \bigcup_{\rho \in \Lambda} \text{add}(\rho) = \emptyset$ and since $\text{del}(\tilde{\rho}) \subseteq \bigcup_{\rho \in \Lambda} \text{del}(\rho)$ it follows immediately also the latter.

$$\begin{aligned}
& f \in S \wedge f \notin S' \\
= & f \in S \wedge f \notin ((S \setminus \bigcup_{\rho \in \Lambda} \text{del}(\rho)) \cup \bigcup_{\rho \in \Lambda} \text{add}(\rho)) && \text{[by definition]} \\
= & f \in S \wedge f \in \bigcup_{\rho \in \Lambda} \text{del}(\rho) \wedge f \notin \bigcup_{\rho \in \Lambda} \text{add}(\rho) && \text{[by simplification]}
\end{aligned}$$

Fig. 9 Proof related to Lemma 7

The formula (41) is true in M if and only if for every fact $f \in N_f^i$ if $f^i \in M$ (i.e. $f \in S$) and $f^{i+1} \notin M$ (i.e. $f \notin S'$), then at least one of the rules, say $\tilde{\rho}$, that deletes f must be such that $\tilde{\rho}^i \in M$ (i.e. $\tilde{\rho} \in \Lambda$). It follows immediately that $M \models (41)$ holds for every fact $f \in N_f^i$ such that either $f \notin S$ or $f \in S'$. Let us consider the case $f \in N_f^i$ such that both $f \in S$ and $f \notin S'$. By hypothesis $S' = (S \setminus \bigcup_{\rho \in \Lambda} \text{del}(\rho)) \cup \bigcup_{\rho \in \Lambda} \text{add}(\rho)$ and then, by applying the steps of Figure 10, we conclude that for every fluent $f \in N_f^i$ such that $f \in S$ and $f \notin S'$, we have that $f \in \bigcup_{\rho \in \Lambda} \text{del}(\rho)$, i.e. there exists at least one rule $\tilde{\rho} \in \Lambda$ such that $f \in \text{del}(\tilde{\rho})$, and thus (41) is true in M .

To prove $M \models (27)$ it suffices to proceed as above but dually.

The formula (43) is true in M if and only if for every fact $f \in N_f^{i+1} \setminus N_f^i$ if $f^{i+1} \in M$ (i.e. $f \in S'$), then at least one of the rules, say $\tilde{\rho}$, that adds f must be such that $\tilde{\rho}^i \in M$ (i.e. $\tilde{\rho} \in \Lambda$). By definition of S' and since $S \subseteq N_f^i$ it follows that for all fact $f \in N_f^{i+1} \setminus N_f^i$, $f \in S'$ if and only if $f \in \bigcup_{\rho \in \Lambda} \text{add}(\rho)$. Hence $M \models (43)$ holds for all $f \in N_f^{i+1} \setminus N_f^i$ such that $f \in S'$. In the other case, i.e. $f \in N_f^{i+1} \setminus N_f^i$ such that $f \notin S'$, again $M \models (43)$ trivially holds since the antecedent of the implication is false.

To conclude the overall proof we are left with proving that $M \models (44)$. Mutexes are divided into two disjoint classes, static mutexes over rules (denoted with $\hat{\oplus}$ and constructed by considering only part (a) of PG4) and dynamic mutexes. As pointed out in [29] dynamic mutexes are implied by static mutexes in the overall encoding and therefore if we prove that for all pair of rules $\rho_1, \rho_2 \in N_\rho^i$ such that $\rho_1 \hat{\oplus}^i \rho_2$ $M \models \neg(\rho_1^i \wedge \rho_2^i)$ holds, then also $M \models (44)$ is proved. This amount showing that for all pair of rules $\rho_1, \rho_2 \in N_\rho^i$ such that $\rho_1 \hat{\oplus}^i \rho_2$ either $\rho_1^i \notin M$ (i.e. $\rho_1 \notin \Lambda$) or $\rho_2^i \notin M$ (i.e. $\rho_2 \notin \Lambda$). This is a simple consequence of the facts that static mutexes are constructed by considering only part (a) of PG4 and that the rules in Λ are applicable and linearisable, i.e. their effects do not contradict each other and they are pairwise non-interfering.

Theorem 6 (Soundness and Completeness of the Graphplan-based Encoding) *If M is a model of $[\Xi]_k$ for $k \geq 0$, then $S_0 \rho_0 S_1 \cdots S_{k-1} \rho_{k-1} S_k$ is a (partial-order) solution of $\hat{\Xi}$, where $S_i = \{f \in \mathcal{F} : f^i \in M\}$ for $i = 0, \dots, k$ and ρ_i is the parallel composition of the rules in $\{\rho \in \mathcal{L} : \rho^i \in M\}$ for $i = 0, \dots, k-1$. Conversely, if $\pi_k = S_0 \rho_0 S_1 \cdots S_{k-1} \rho_{k-1} S_k$ is a (partial-order) solution of $\hat{\Xi}$, then there exists a model M of $[\Xi]_k$ such that $\pi(M) = \pi_k$.*

Proof Soundness readily follows from Lemma 8, Lemma 9 and Theorem 3. Similarly, completeness follows from Lemma 8, Lemma 10, and Theorem 4.

A.6 Proofs of the results of Section 5

Theorem 7 *Let $\Xi = \langle \Gamma, \mathcal{B} \rangle$ be a BNAS problem obtained by specialising the composition rules of the intruder as described in Section 3.5. When $(\Gamma)_i$ has levelled-off, if $[\Xi]_i$ is unsatisfiable, then there is no attack on Ξ .*

Proof We recall that the planning graph built at step i represents an over-approximation, in terms of reachable states and applicable rules, of the forward search tree of depth i . Notice that in the general case the unsatisfiability of $[\Xi]_i$ does not allow to conclude that there is no attack on Ξ . In fact, a solution can require multiple executions of the same rule at different steps as well as the sequential execution of rules that are in mutual exclusion. Thus, it could be that i steps are not enough to execute the attack. However, if Ξ is a BNAS problem in which the intruder has no rules for composing messages and $(\Gamma)_i$ has levelled-off, then the unsatisfiability of $[\Xi]_i$ guarantees that there is no attack on Ξ . To prove this it suffices to show that if $(\Gamma)_i$ has levelled-off then there cannot be multiple executions of the same rule nor sequential execution of rules that are in mutex.

- Let us first consider the multiple executions of the same rule. For the rules of the honest agents this is prevented by the ordering on the **state** facts. For the decomposition rules it is immediate to see that once one of them has been applied it is useless to repeat that rule since it would just cause stuttering.
- Let us now consider the sequential execution of rules that are in mutual exclusion. For rules of the honest agents to be conflicting they must have the same **state** fact in the left-hand-side. (Notice that the rewrite rules of the intruder cannot possibly generate conflicts.) But then there cannot be a path in which the two conflicting rules can be both executed. In fact once one of the two rules has been applied the **state** fact is deleted and it will never occur again in the state.

$$\begin{aligned} & f \in S \wedge f \notin S' \\ = & f \in S \wedge f \notin ((S \setminus \bigcup_{\rho \in A} \text{del}(\rho)) \cup \bigcup_{\rho \in A} \text{add}(\rho)) && \text{[by definition]} \\ = & f \in S \wedge f \in \bigcup_{\rho \in A} \text{del}(\rho) \wedge f \notin \bigcup_{\rho \in A} \text{add}(\rho) && \text{[by simplification]} \end{aligned}$$

Fig. 10 Proof related to Lemma 10