A Rewriting-based Forwards Semantics for Maude-NPA

Santiago Escobar
DSIC-ELP, Universitat Politècnica de València, Spain
sescobar@dsic.upv.es

Catherine Meadows
Naval Research Laboratory, Washington DC, USA
meadows@itd.nrl.navy.mil

José Meseguer
University of Illinois at Urbana-Champaign, USA
meseguer@illinois.edu

Sonia Santiago
DSIC-ELP, Universitat Politècnica de València, Spain
ssantiago@dsic.upv.es

ABSTRACT

The Maude-NRL Protocol Analyzer (Maude-NPA) is a tool for reasoning about the security of cryptographic protocols in which the cryptosystems satisfy different equational properties. It tries to find secrecy or authentication attacks by searching backwards from an insecure attack state pattern that may contain logical variables, in such a way that logical variables become properly instantiated in order to find an initial state. The execution mechanism for this logical reachability is narrowing modulo an equational theory. Although Maude-NPA also possesses a forwards semantics naturally derivable from the backwards semantics, it is not suitable for state space exploration or protocol simulation.

In this paper we define an executable forwards semantics for Maude-NPA, instead of its usual backwards one, and restrict it to the case of *concrete states*, that is, to terms without logical variables. This case corresponds to standard rewriting modulo an equational theory. We prove soundness and completeness of the backwards narrowing-based semantics with respect to the rewriting-based forwards semantics. We show its effectiveness as an analysis method that complements the backwards analysis with new prototyping, simulation, and explicit-state model checking features by providing some experimental results.

Categories and Subject Descriptors

C.2.2 [**Computer-communication Networks**]: Network Protocols; D.2.4 [**Software Engineering**]: Software/Program Verification; D.3.2 [**Programming Languages**]: Language Classifications; D.4.6 [**Operating Systems**]: Security and Protection; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs

HotSoS '14 April 08 - 09 2014, Raleigh, NC, USA

http://dx.doi.org/10.1145/2600176.2600186.

General Terms

Protocol Verification, Term Rewriting, Model Checking

Keywords

cryptographic protocol analysis, logical narrowing-based reachability analysis, standard rewriting-based model checking, reasoning modulo an equational theory

1. INTRODUCTION

Over the years a number of different techniques have been applied to security analysis of cryptographic protocols via state space exploration. Many of the earlier approaches [15, 19, 6] made use of explicit-state model-checking using forward search. More recently the emphasis has been on symbolic-state model-checking, in which states are represented by terms containing variables [3, 12, 2, 9, 16]. Here state transitions are computed using unification or constraint based techniques, and search is generally performed backwards from a symbolic specification of an insecure state. This approach has many advantages. In particular, the combination of symbolic states and goal-directed backwards search can result in a smaller search space.

However, there are still a number of cases when an explicit-state model checking may be preferable. First, when one wants to check via simulation whether one has specified a protocol that perform its functions correctly in the absence of an attacker, symbolic-state model checking may be overkill. Second, there are some cases in which cryptographic functions may obey equational theories that do not integrate well with the unification techniques that have been developed for symbolic-state based cryptographic protocol analysis. One example is encryption that is homomorphic over a free or Abelian group operator. Although unification in these theories is decidable, they lack the *finite variant property* [8] that is necessary for existing approaches to symbolic-state analysis of cryptographic protocols [11]. Third, backwards search is not as useful for expressing liveness properties, used, for example, in fair exchange protocols [4]. Finally, even if one is using backwards search, it may still be more practical to develop definitions of certain properties using a forwards semantics that gives a more natural representation of the way an intruder increases its knowledge. This is helpful for reasoning about security properties involving subtle properties of the intruder knowledge, such as Abadi and Fournet's observational equivalence [1].

For the above reasons, one may prefer not to limit oneself to one approach, but to switch back and forth between these two. By integrating the two approaches we can obtain the best of both worlds, using each technique where it works best. But for such an integration to be correct and useful two requirements should be met: (1) the forwards and backwards tools should share the *same semantic model* and language; and (2) the operational semantics used in the forwards and backwards analyses should *agree with each other*.

In this paper we show how we achieve this integration via a novel rewriting-based forwards semantics appropriate for the Maude-NPA protocol analysis tool [12]. Maude-NPA is a symbolic cryptographic protocol analysis tool designed to reason about the security of protocols in which the cryptographic algorithms can be described using different equational theories. It has an operational semantics based on rewrite rules and is implemented in the Maude rewriting language [7] via backwards narrowing with respect to the rewrite rules. Thus it is a natural choice for this integration, since a forwards semantics can also be defined by rewrite rules, but executed by rewriting instead of narrowing.

Although Maude-NPA already has an intuitive forwards semantics obtained by reversing the rewrite rules defining the backwards semantics, it is not suitable for model checking. First, the rewrite rules in this semantics can introduce *extra variables*. This is unproblematic for narrowing-based symbolic analysis, but unacceptable for rewriting-based forwards execution. Second, in this semantics a state contains explicit information about events occurring in the future (since they were observed "earlier" in the backwards search). This information must be removed in the forwards semantics, while still ensuring that reachability is not affected. Third, in this semantics fresh values (nonces, session keys, etc.) are represented by special variables. These must be replaced by constants in the forwards semantics, again without affecting reachability.

The approach we developed is: (1) use the *strand* [13] model of protocols as the shared semantic model and specification language for the forwards analysis (Maude-NPA already used it for backwards analysis); and (2) have two different operational semantics, the given one for backwards symbolic execution, and the new one for forwards concrete execution. In this way, a protocol is specified once, and can then be: (i) executed using the forwards semantics for prototyping, simulation, and explicit-state model checking; and (ii) formally analyzed using the backwards semantics. To ensure the *correctness* of this integration we prove the backwards semantics sound and complete with respect to the forwards semantics. In this way, any security properties proved or counter-examples found using one technique are guaranteed to hold for the results of applying the other.

We realized several benefits from developing the forwards semantics. One is that the forwards semantics can be *executed* and *model checked* in Maude. We have taken advantage of this feature to implement a prototype explicit-state cryptographic protocol model-checker in Maude that, like Maude-NPA, can be used to reason in the presence of different equational theories. Although we have not yet implemented any state space reduction techniques for it —so there is no current way of achieving termination— it has nevertheless been able to automatically find attacks on some simple protocols that use various equational theories. We summarize the experimental results for this prototype in Section 5.

The definition of this novel rewriting-based protocol analysis is relevant, since the new forwards semantics is directly implementable in rule-based programming languages such as Maude without any need for constraint solving or unification procedures as it is done in most current approaches (see [5]), including Maude-NPA, allowing us to explore applications such as the simulation of prototypes and reasoning about theories without the finite variant property.

The rest of the paper is organized as follows. In Section 2 we give some basic preliminaries on rewriting and narrowing necessary to understand the paper. In Section 3 we give a description of Maude-NPA, including its strand model and backwards semantics. In Section 4 we define the forwards semantics for Maude-NPA and prove the soundness and completeness of the backwards semantics with respect to it. In Section 5 we present experimental results obtained with our forwards rewriting tool. In Section 6 we conclude the paper and discuss some of the issues that comparing the backwards with the forwards semantics helped us resolve.

The Science

This work contributes to the science of security by providing additional theoretical foundations to the Maude-NRL Protocol Analyzer (Maude-NPA), a tool for the formal analysis of cryptographic protocols. It achieves this by providing a new forwards semantics that is proved sound and complete with respect to the original Maude-NPA executable semantics, which searches backwards from a state description specified by the user. Moreover, the forwards semantics is directly implementable in a rule-based language such as Maude, without any need for constraint solving as is done in most current approaches. This novel analysis method still needs performance improvements, but allows for easy prototyping and simulation, as is illustrated by our preliminary experimental results. Thus it not only extends the theoretical foundations of Maude-NPA, but provides tools that can be used to enhance its usefulness as a system for cryptographic protocol analysis.

2. PRELIMINARIES

We follow the classical notation and terminology from [22] for term rewriting, and from [17] for rewriting logic and order-sorted notions. We assume an order-sorted signature $\mathbf{\Sigma} = (\mathsf{S}, \leqslant, \Sigma)$ with poset of sorts $(\mathsf{S}, \leqslant)$. We also assume an S-sorted family $\mathcal{X} = \{\mathcal{X}_\mathsf{s}\}_{\mathsf{s}\in\mathsf{S}}$ of disjoint variable sets with each $\mathcal{X}_\mathsf{s}$ countably infinite. $\mathcal{T}_\Sigma(\mathcal{X})_\mathsf{s}$ is the set of terms of sort s, and $\mathcal{T}_{\Sigma,\mathsf{s}}$ is the set of ground terms of sort s. We write $\mathcal{T}_\Sigma(\mathcal{X})$ and $\mathcal{T}_\Sigma$ for the corresponding order-sorted term algebras. For a term t, $Var(t)$ denotes the set of variables in t.

Positions are represented by sequences of natural numbers denoting an access path in the term when viewed as a tree. The top or root position is denoted by the empty sequence $\wedge$. Given $U \subseteq \Sigma \cup \mathcal{X}$, $Pos_U(t)$ denotes the set of positions of a term t that are rooted by symbols or variables in U. The set of positions of a term t is written $Pos(t)$, and the set of non-variable positions $Pos_\Sigma(t)$. The subterm of t at position p is $t|_p$ and $t[u]_p$ is the term t where $t|_p$ is replaced by u.

A *substitution* $\sigma \in \mathcal{S}ubst(\Sigma, \mathcal{X})$ is a sorted mapping from a finite subset of $\mathcal{X}$ to $\mathcal{T}_\Sigma(\mathcal{X})$. Substitutions are written as $\sigma = \{X_1 \mapsto t_1, \ldots, X_n \mapsto t_n\}$ where the domain of σ is $Dom(\sigma) = \{X_1, \ldots, X_n\}$ and the set of variables introduced

by terms $t_1, \ldots, t_n$ is written $Ran(\sigma)$. The identity substitution is *id*. Substitutions are homomorphically extended to $\mathcal{T}_\Sigma(\mathcal{X})$. The application of a substitution σ to a term t is denoted by $t\sigma$. For simplicity, we assume that every substitution is idempotent, i.e., σ satisfies $Dom(\sigma) \cap Ran(\sigma) = \emptyset$. Substitution idempotency ensures $t\sigma = (t\sigma)\sigma$. The restriction of σ to a set of variables V is $\sigma|_V$. Composition of two substitutions σ and σ' is denoted by $\sigma\sigma'$.

A Σ-*equation* is an unoriented pair $t = t'$, where $t, t' \in \mathcal{T}_\Sigma(\mathcal{X})_\mathsf{s}$ for some sort $\mathsf{s} \in \mathsf{S}$. Given Σ and a set E of Σ-equations, order-sorted equational logic induces a congruence relation $=_E$ on terms $t, t' \in \mathcal{T}_\Sigma(\mathcal{X})$ (see [18]). The E-equivalence class of a term t is denoted by $[t]_E$ and $\mathcal{T}_{\Sigma/E}(\mathcal{X})$ and $\mathcal{T}_{\Sigma/E}$ denote the corresponding order-sorted term algebras modulo E. Throughout this paper we assume that $\mathcal{T}_{\Sigma,\mathsf{s}} \neq \emptyset$ for every sort s, because this affords a simpler deduction system. An *equational theory* (Σ, E) is a pair with Σ an order-sorted signature and E a set of Σ-equations. The *E-subsumption* preorder $\sqsupseteq_E$ (or just $\sqsupseteq$ if E is understood) holds between $t, t' \in \mathcal{T}_\Sigma(\mathcal{X})$, denoted $t \sqsupseteq_E t'$ (meaning that t is *more general* than t' modulo E), if there is a substitution σ such that $t\sigma =_E t'$; such a substitution σ is said to be an *E-match* from t' to t.

An *E-unifier* for a Σ-equation $t = t'$ is a substitution σ such that $t\sigma =_E t'\sigma$. For $Var(t) \cup Var(t') \subseteq W$, a set of substitutions $CSU_E^W(t = t')$ is said to be a *complete* set of unifiers for the equality $t = t'$ modulo E away from W iff: (i) each $\sigma \in CSU_E^W(t = t')$ is an E-unifier of $t = t'$; (ii) for any E-unifier ρ of $t = t'$ there is a $\sigma \in CSU_E^W(t = t')$ such that $\sigma|_W \sqsupseteq_E \rho|_W$; (iii) for all $\sigma \in CSU_E^W(t = t')$, $Dom(\sigma) \subseteq (Var(t) \cup Var(t'))$ and $Ran(\sigma) \cap W = \emptyset$. If the set of variables W is irrelevant or is understood from the context, we write $CSU_E(t = t')$ instead of $CSU_E^W(t = t')$. An E-unification algorithm is *complete* if for any equation $t = t'$ it generates a complete set of E-unifiers. A unification algorithm is said to be *finitary* and complete if it always terminates after generating a finite and complete set of solutions.

A *rewrite rule* is an oriented pair $l \to r$, where[1] $l \notin \mathcal{X}$ and $l, r \in \mathcal{T}_\Sigma(\mathcal{X})_\mathsf{s}$ for some sort $\mathsf{s} \in \mathsf{S}$. An *(unconditional) order-sorted rewrite theory* is a triple (Σ, E, R) with Σ an order-sorted signature, E a set of Σ-equations, and R a set of rewrite rules.

The rewriting relation on $\mathcal{T}_\Sigma(\mathcal{X})$, written $t \to_R t'$ or $t \to_{p,R} t'$ holds between t and t' iff there exist $p \in Pos_\Sigma(t)$, $l \to r \in R$ and a substitution σ, such that $t|_p = l\sigma$, and $t' = t[r\sigma]_p$. The subterm $t|_p$ is called a *redex*. The relation $\to_{R/E}$ on $\mathcal{T}_\Sigma(\mathcal{X})$ is $=_E;\to_R;=_E$, i.e., $t \to_{R/E} t'$ iff there exists u, u' s.t. $t =_E u \to_R u' =_E t'$. Note that $\to_{R/E}$ on $\mathcal{T}_\Sigma(\mathcal{X})$ induces a relation $\to_{R/E}$ on the free (Σ, E)-algebra $\mathcal{T}_{\Sigma/E}(\mathcal{X})$ by $[t]_E \to_{R/E} [t']_E$ iff $t \to_{R/E} t'$. The transitive (resp. transitive and reflexive) closure of $\to_{R/E}$ is denoted $\to^+_{R/E}$ (resp. $\to^*_{R/E}$).

The reducibility of the $\to_{R/E}$ relation is undecidable in general since E-equivalence can be undecidable. Therefore, R/E-rewriting is usually implemented [14] by R,E-rewriting. A relation $\to_{R,E}$ on $\mathcal{T}_\Sigma(\mathcal{X})$ is defined as: $t \to_{p,R,E} t'$ (or just $t \to_{R,E} t'$) iff there exist $p \in Pos_\Sigma(t)$, a rule $l \to r$ in R, and a substitution σ such that $t|_p =_E l\sigma$ and $t' = t[r\sigma]_p$.

Let t be a term and W be a set of variables such that $Var(t) \subseteq W$, the R, E-*narrowing* relation on $\mathcal{T}_\Sigma(\mathcal{X})$ is defined as $t \leadsto_{p,\sigma,R,E} t'$ ($\leadsto_{\sigma,R,E}$ if p is understood, $\leadsto_\sigma$ if R, E are also understood, and $\leadsto$ if σ is also understood) if there is a non-variable position $p \in Pos_\Sigma(t)$, a rule $l \to r \in R$ properly renamed s.t. $(Var(l) \cup Var(r)) \cap W = \emptyset$, and a unifier $\sigma \in CSU_E^{W'}(t|_p = l)$ for $W' = W \cup Var(l)$, such that $t' = (t[r]_p)\sigma$. For convenience, in each narrowing step $t \leadsto_\sigma t'$ we only specify the part of σ that binds variables of t. The transitive (resp. transitive and reflexive) closure of $\leadsto$ is denoted by $\leadsto^+$ (resp. $\leadsto^*$). We may write $t \leadsto^k_\sigma t'$ if there are $u_1, \ldots, u_{k-1}$ and substitutions $\rho_1, \ldots, \rho_k$ such that $t \leadsto_{\rho_1} u_1 \cdots u_{k-1} \leadsto_{\rho_k} t'$, $k \geqslant 0$, and $\sigma = \rho_1 \cdots \rho_k$.

3. MAUDE-NPA

In this section we give a high-level summary of Maude-NPA. For further information, please see [12].

Given a protocol $\mathcal{P}$, states are modeled as elements of an initial algebra $T_{\Sigma_\mathcal{P}/E_\mathcal{P}}$, where $\Sigma_\mathcal{P}$ is the signature defining the sorts and function symbols (for the cryptographic functions and for all the state constructor symbols) and $E_\mathcal{P}$ is a set of equations specifying the *algebraic properties* of the cryptographic functions and the state constructors. Therefore, a state is an $E_\mathcal{P}$-equivalence class $[t] \in T_{\Sigma_\mathcal{P}/E_\mathcal{P}}$ with t a ground $\Sigma_\mathcal{P}$-term. However, we explore *symbolic state patterns* $[t(x_1, \ldots, x_n)] \in T_{\Sigma_\mathcal{P}/E_\mathcal{P}}(X)$ on the free $(\Sigma_\mathcal{P}, E_\mathcal{P})$-algebra over a set of sorted variables X. There are three relevant sorts in Maude-NPA, State, Msg, and Fresh, which are described below. Also, due to the symbolic representation, we use uppercase names for variables (we omit the sort of a variable when it is easy to deduce from the context) and lowercase names for terms (with or without variables). Indeed, we will make explicit when a term does not contain variables.

In Maude-NPA [12], a *state pattern* in a protocol execution is a term t of sort State (i.e., $t \in T_{\Sigma_\mathcal{P}/E_\mathcal{P}}(X)_{\mathsf{State}}$) which has the form $\{S_1 \,\&\, \cdots \,\&\, S_n \,\&\, \{IK\}\}$ where & is an associative-commutative union[2] operator with identity symbol $\emptyset$. Each element in the set is either a *strand* S_i or the *intruder knowledge* $\{IK\}$ at that state.

The *intruder knowledge* $\{IK\}$ also belongs to the state and is represented as a set of facts using the comma as an associative-commutative union [3] operator with identity operator *empty*. There are two kinds of intruder facts: positive knowledge facts (the intruder knows m, i.e., $m \in \mathcal{I}$), and negative knowledge facts (the intruder *does not yet know* m but *will know it in a future state*, i.e., $m \notin \mathcal{I}$), where m is a

[1] Note that we do not impose here the standard condition $Var(r) \subseteq Var(l)$, since extra variables will be introduced in the righthand side of a rule when Maude-NPA introduces extra strands. However, $Var(r) \subseteq Var(l)$ will be required in the forwards semantics, as we will make explicit in the paper.

[2] As described in [12], & can also be treated as an associative-commutative-idempotent union operator with an identity symbol because the combination of fresh variables and the learn-only-once rule allows for that. However, in the forward semantics, & *cannot be* idempotent, since there will be situations where two occurrences of the same strand will lead to completely different strands later on but their current partial representation in the state makes them equal. So keeping only one occurrence of the partial strand is wrong.

[3] Again, the comma for the intruder's knowledge is described in [12] as an associative-commutative union operator with an identity symbol but it can be understood as being idempotent, though only for the positive intruder facts. In the forward semantics, since there are only positive intruder facts, this is not a problem.

message expression.

A *strand* [13] specifies the sequence of messages sent and received by a principal executing the protocol and is represented as a sequence of messages $[msg_1^-, msg_2^+, msg_3^-, \ldots, msg_{k-1}^-, msg_k^+]$ such that msg_i^- (also written $-msg_i$) represents an input message, msg_i^+ (also written $+msg_i$) represents an output message, and each msg_i is a term of sort Msg (i.e., $msg_i \in T_{\Sigma_\mathcal{P}/E_\mathcal{P}}(X)_{\mathsf{Msg}}$). For each positive message msg_i^+ in a strand, the variables occurring in message msg_i must appear[4] in previous messages $msg_1, \ldots, msg_{i-1}$, except for variables denoting principal names (they are considered as initial knowledge available to all participants) and variables of sort Fresh. Variables of sort Fresh are unique for each instance of a strand schemata, i.e., if we compare two strands for Alice or a strand for Alice and a strand for Bob, they will have different, unique, fresh variables associated with them.

Strands are used to represent both the actions of honest principals (with a strand specified for each protocol role) and the actions of an intruder (with a strand for each operator an intruder is able to perform on terms). In Maude-NPA, strands evolve over time; the symbol | is used to divide past and future. That is, given a strand $[\, m_1{}^{\pm}, \ldots, m_i{}^{\pm} \mid m_{i+1}{}^{\pm}, \ldots, m_k{}^{\pm} \,]$, messages $m_1^{\pm}, \ldots, m_i^{\pm}$ are the *past messages*, and messages $m_{i+1}^{\pm}, \ldots, m_k^{\pm}$ are the *future messages* ($m_{i+1}^{\pm}$ is the immediate future message). We often remove the nils for clarity, except when there is nothing else between the vertical bar and the beginning or end of a strand. A strand $[msg_1^{\pm}, \ldots, msg_k^{\pm}]$ is a shorthand for $[nil \mid msg_1^{\pm}, \ldots, msg_k^{\pm}, nil]$. An initial state is a state where the bar is at the beginning for all strands in the state, and the intruder knowledge is empty. A final state is a state where the bar is at the end for all strands in the state and there is no intruder fact of the form $m \notin \mathcal{I}$.

Since Fresh variables must be treated differently from other variables by Maude-NPA, we make them explicit by writing $:: r_1, \ldots, r_k :: [m_1^{\pm}, \ldots, m_n^{\pm}]$, where each r_i first appears in an output message $m_{j_i}^+$ and can later be used in any input and output message of $m_{j_i+1}^{\pm}, \ldots, m_n^{\pm}$. If there are no Fresh variables, we write $:: nil :: [m_1^{\pm}, \ldots, m_n^{\pm}]$.

Let us remark that the restriction mentioned in Footnote 4 that the variables of each positive message in a strand must appear in previous input messages, except for variables denoting principal names and variables of sort Fresh, together with the explicit identification of which are the variables of sort Fresh created by each strand are essential in the rewriting-based forwards semantics below for obtaining rewrite rules without extra variables, i.e., rewrite rules $l \rightarrow r$ where $Var(r) \subseteq Var(l)$, which allows for effectively executable rewriting computations.

EXAMPLE 1. *Let us consider the well-known Diffie-Hellman protocol, used without authentication. This protocol uses exponentiation to share a secret between two parties, Alice and Bob. The protocol involves an initiator, Alice, and a responder, Bob. We use the common notation $A \hookrightarrow B : M$ to stand for "A sends message M to B". Encryption of message M using a key K is denoted by $\{M\}_K$. Decryption is done when the principal knows the appropriate key. Concatenation of two messages M_1 and M_2 is denoted by $M_1; M_2$. Raising message M to the power of exponent X is denoted by $(M)^X$. There is a public term denoted by g, which will be the base of our exponentiations. We represent the product of exponents by using the symbol $*$, which is an associative-commutative symbol. Nonces are represented by N_X, denoting a nonce created by principal X. The expression* sec(A,r) *denotes a secret term generated by principal A. The protocol description is as follows.*

1. $A \hookrightarrow B : A; B; g^{N_A}$
   Alice creates a new nonce N_A and sends her name, Bob's name, and g^{N_A} to Bob.
2. $B \hookrightarrow A : B; A; g^{N_B}$
   Bob creates a new nonce N_B and sends his name, Alice's name, and g^{N_B} to Alice.
3. $A \hookrightarrow B : \{secret\}_{g^{N_B N_A}}$
   Alice computes $g^{N_B N_A}$ and encrypts the secret data. The key $g^{N_B N_A}$ is equal to $g^{N_B * N_A}$ using the algebraic property $X^{YZ} = X^{ZY} = X^{Y*Z}$. Bob computes $g^{N_A N_B}$ and obtains the secret data.

This protocol is described using strands as follows. Here encryption $\{M\}_K$ is denoted by $e(K, M)$ and exponentiation X^Y is denoted by $exp(X, Y)$. Nonces are denoted by terms of the form $n(A, r)$, where r is a fresh variable that ensures uniqueness and A is a variable used to identify which principal generated the nonce.

$$:: r, r' :: [(A; B; exp(g, n(A, r)))^+, (B; A; X)^-, e(exp(X, n(A, r)), sec(A, r'))^+] \;\&$$
$$:: r'' :: [(A; B; Y)^-, (B; A; exp(g, n(B, r')))^+, e(exp(Y, n(B, r')), Sr)]$$

Intruder strands are also included for each function. For example, encryption by the intruder is described by the strand $[-(K), -(M), +(e(K, M))]$, and decryption by the intruder is described by the strand $[-(K), -(M), +(d(K, M))]$ together with the equational property $d(K, e(K, M)) = M$. The intruder can also generate nonces via the strand $:: r :: [+(n(i, r))]$.

In Section 3.1 we show how backwards analysis works in Maude-NPA and in Section 3.2 we specify the rewrite rules governing the backwards semantics. In Section 4.1 we define the new forwards semantics for a protocol and in Section 4.2 we specify the rewrite rules governing the forwards semantics.

3.1 Backwards Reachability Analysis

Since the number of states $T_{\Sigma_\mathcal{P}/E_\mathcal{P}}$ is in general infinite, rather than exploring concrete protocol states $[t] \in T_{\Sigma_\mathcal{P}/E_\mathcal{P}}$ we explore *symbolic state patterns* $[t(x_1, \ldots, x_n)] \in T_{\Sigma_\mathcal{P}/E_\mathcal{P}}(X)$ on the free $(\Sigma_\mathcal{P}, E_\mathcal{P})$-algebra over a set of variables X. In this way, a state pattern $[t(x_1, \ldots, x_n)]$ represents not a single concrete state but a possibly infinite set of such states, namely all the instances of the pattern $[t(x_1, \ldots, x_n)]$ where the variables $x_1, \ldots, x_n$ have been instantiated by concrete ground terms.

The protocol analysis methodology of Maude-NPA is then based on the idea of *symbolic backward reachability analysis*, where we begin with one or more state patterns corresponding to *attack states*, and want to prove or disprove that they are *unreachable* from the set of initial protocol states.

[4]This restriction is common in protocol analysis using constraint systems and corresponds to *deterministic constraint systems*, see [5].

EXAMPLE 2. *Given the protocol of Example 1, the final state pattern associated to Bob receiving some secret data from a communication with Alice and the intruder learning the secret is as follows (where Y, SR, SS, and IK are variables and we use lowercase a and b to represent the actual names of Alice and Bob instead of variable names A and B):*

$$:: r'' :: [-(a; b; Y), +(b; a; exp(g, n(b, r''))), \\ -(e(exp(Y, n(b, r'')), SR)) \mid nil])$$

The strands of the initial state found by the tool correspond to a very general man-in-the-middle attack, with two sessions and variables B', NS and NS'. The principal strands are as follows, where Alice (principal name a) is talking to some principal name B' and Bob (principal name b) believes is talking to Alice:

$$:: r_1, r_2 :: [+(a; B'; exp(g, n(a, r_2))), \\ -(B'; a; exp(g, NS)), \\ +(e(exp(g, NS * n(a, r_2)), sec(a, r_1)))]) \ \& \\ :: r_3 :: [-(a; b; exp(g, NS')), \\ +(b; a; exp(g, n(b, r_3))), \\ -(e(exp(g, NS' * n(b, r_3)), sec(a, r_1)))]$$

The Dolev-Yao intruder strands are as follows, where variables NS and NS' correspond to sets of nonces generated by the intruder but the tool does not actually try to find instances of those variables:

$$[-(a; B'; exp(g, n(a, r_2))), +(B'; exp(g, n(a, r_2)))] \ \& \\ [-(B'; exp(g, n(a, r_2))), +(exp(g, n(a, r_2)))] \ \& \\ [-(exp(g, n(a, r_2))), -(NS), +(exp(g, NS * n(a, r_2)))] \ \& \\ [-(exp(g, NS * n(a, r_2))), -(e(exp(g, NS * n(a, r_2)), sec(a, r_1))), \\ +(sec(a, r_1))] \ \& \\ [-(exp(g, n(b, r_3))), -(NS'), +(exp(g, NS' * n(b, r_3)))] \ \& \\ [-(exp(g, NS' * n(b, r_3))), -(sec(a, r_1)), \\ +(e(exp(g, NS' * n(b, r_3)), sec(a, r_1)))] \ \& \\ [-(b; a; exp(g, n(b, r_3))), +(a; exp(g, n(b, r_3)))] \ \& \\ [-(a; exp(g, n(b, r_3))), +(exp(g, n(b, r_3)))]$$

Note that Maude-NPA does not display the initial knowledge of the intruder, since it corresponds to all the input and output messages appearing in the initial strands above.

Maude-NPA also allows verification of authentication properties by using never patterns*, i.e., the reachability analysis succeeds when none of the states in the reachability sequence is an instance of the never pattern. Never patterns can share variables with the attack pattern in order to have more specific patterns and the vertical bar is not included in strands of never patterns, since all the combinations of the vertical bar are taken into account. For instance, we can specify the following authentication attack pattern for Diffie-Hellman by including Bob's strand and adding never patterns for Alice's strand (note that we have to specify two never patterns because states may contain always partial strands, this also happens in the forwards semantics):*

$$\{:: r'' :: [-(a; b; Y), +(b; a; exp(g, n(b, r''))), \\ -(e(exp(Y, n(b, r'')), SR)) \mid nil] \ \& \ SS \ \& \ \{IK\}\} \ \wedge \\ never(:: r, r' :: [+(a; b; exp(g, n(a, r)))]) \ \wedge \\ never(:: r, r' :: [+(a; b; exp(g, n(a, r))), -(b; a; X), \\ +(e(exp(X, n(a, r)), sec(a, r')))])$$

The initial state above is also a solution of this attack pattern with never patterns, since Alice was talking to a different participant in a different session. Because (as we shall see in Section 3.2) terms available to the intruder are not always explicitly represented in the intruder knowledge, we assume that never patterns as implemented in Maude-NPA consist only of strands, and do not describe intruder knowledge terms. This is generally the case for authentication patterns. However, if we do wish to specify a never pattern in which the intruder knows a particular message, this can be represented as a set of never patterns, each one containing one of the possible strands containing that message as a positive term. Note that explicitly specifying the message as part of the intruder knowledge in the never pattern would not *rule out all states in which the message is produced, since in both the forwards and backwards semantics the intruder knowledge is only guaranteed to contain the messages the intruder uses to get to the (main) final state; it is not guaranteed to contain all the messages produced in the protocol execution.*

3.2 Backwards Operational Semantics

In the backwards reachability analysis performed by Maude-NPA sketched in Section 3.1, state changes are described by means of a set $R_{B\mathcal{P}}$ of *rewrite rules*, so that the rewrite theory $(\Sigma_{\mathcal{P}}, E_{\mathcal{P}}, R_{B\mathcal{P}})$ characterizes the behavior of protocol $\mathcal{P}$ modulo the equations $E_{\mathcal{P}}$ for backwards execution. In this section we use $R_{B\mathcal{P}}$ to denote the rewrite rules associated to protocol $\mathcal{P}$ for backwards execution, whereas in Section 4.2 we will use $R_{F\mathcal{P}}$ to denote the new rewrite rules associated to protocol $\mathcal{P}$ for forwards execution. The rules $R_{B\mathcal{P}}$ are defined in two blocks below: (i) *generic rules* (1), (2), and (3), and (ii) *protocol-specific* rules (4) generated for each principal and intruder strand in the given protocol.

The following rewrite rules, though written in a forward sense, are used in a backwards sense (by reversing the direction of the arrow) and describe[5] the general state transitions:

$$\{SS \ \& \ [L \mid M^-, L'] \ \& \ \{M{\in}\mathcal{I}, IK\}\} \\ \rightarrow \{SS \ \& \ [L, M^- \mid L'] \ \& \ \{M{\in}\mathcal{I}, IK\}\} \quad (1)$$

$$\{SS \ \& \ [L \mid M^+, L'] \ \& \ \{IK\}\} \\ \rightarrow \{SS \ \& \ [L, M^+ \mid L'] \ \& \ \{IK\}\} \quad (2)$$

$$\{SS \ \& \ [L \mid M^+, L'] \ \& \ \{M{\notin}\mathcal{I}, IK\}\} \\ \rightarrow \{SS \ \& \ [L, M^+ \mid L'] \ \& \ \{M{\in}\mathcal{I}, IK\}\} \quad (3)$$

Variables L and L' denote lists of input and output messages of the form m^+ or m^- within a strand, IK denotes a set of intruder facts ($m{\in}\mathcal{I}$ or $m{\notin}\mathcal{I}$), and SS denotes a set of strands.

Rule (1) used in a forwards sense means that the intruder knows the message that a strand is waiting to receive, but when executed backwards by narrowing on a symbolic state with variables SS' and IK' it may either unify the input message with some term already in the intruder knowledge or unify SS' or IK' with parts of the rule, thus adding new information to the symbolic state. Rule (2) used in a forwards sense, means that the intruder did not learn a message generated by a strand, but when executed backwards by narrowing on a symbolic state with variables SS' and IK', it

[5]We do not include the fresh variables in rules (1), (2), and (3) for simplicity, but a expression $:: r_1, \ldots, r_k ::$ should always appear before each strand.

either moves the bar to the left or unifies variable SS' with parts of the rule. Rule (3) used in a forwards sense means that the intruder learns a message M generated by a strand that it did not know before (expressed by $M \notin \mathcal{I}$) , but when executed backwards by narrowing on a symbolic state with variables SS' and IK', it either detects the instant where the intruder is learning a message and, thus, transforms a fact $m \in \mathcal{I}$ into $m \notin \mathcal{I}$ to identify the transition in which the fact $M \in \mathcal{I}$ was learned, or unifies variables SS' or IK' with parts of the rule, thus adding new information to the symbolic state.

For an unbounded number of sessions, we have extra rewrite rules (one for each positive message in a protocol or intruder strand) that dynamically introduce additional strands into a state:

$$\begin{array}{l} \forall\ [l_1, u^+, l_2] \in \mathcal{P} : \\ \{\{SS \& [l_1 | u^+, l_2\,] \,\&\, \{u \notin \mathcal{I}, IK\}\} \rightarrow \{SS \& \{u \in \mathcal{I}, IK\}\}\} \end{array} \quad (4)$$

Note that these rules are essential in a backwards sense, since they will dynamically introduce new strands guided by existing terms in the intruder knowledge. For example, the intruder encryption capability $[-(K), -(M), +(e(K, M)]$ produces the following extra rewrite rule adding a new strand (when the rules are executed backwards) if a message of the form $e(K, M)$ appears in the intruder knowledge:

$$\begin{array}{l} \{SS \,\&\, [-(K), -(M) \mid +(e(K, M)] \,\&\, \{(e(K, M) \notin \mathcal{I}, IK\}\} \\ \rightarrow \{SS \,\&\, \{e(K, M) \in \mathcal{I}, IK\}\} \end{array}$$

The way to analyze *backwards* reachability is then relatively easy, namely, to symbolically run the protocol "in reverse" by narrowing modulo the equations $E_{\mathcal{P}}$. This can be achieved by using the set of rules $R_{B\mathcal{P}}^{-1}$ (where $v \longrightarrow u$ is in $R_{B\mathcal{P}}^{-1}$ iff $u \longrightarrow v$ is in $R_{B\mathcal{P}}$), and performing backwards *narrowing* (see Section 2) steps of the form $S \rightsquigarrow_{R_{B\mathcal{P}}^{-1}, E_{\mathcal{P}}} S'$ to search for an initial state pattern.

Definition 1. *Let $\mathcal{P}$ be a protocol with signature $\Sigma_{\mathcal{P}}$ and equational theory $E_{\mathcal{P}}$. We define the* backwards rewrite theory characterizing *$\mathcal{P}$ to be $(\Sigma_{\mathcal{P}}, E_{\mathcal{P}}, R_{B\mathcal{P}}^{-1})$ where $R_{B\mathcal{P}}^{-1}$ is the result of reversing the rewrite rules $\{(1), (2), (3)\} \cup (4)$.*

4. A REWRITING-BASED FORWARDS SEMANTICS FOR MAUDE-NPA

In this section we define a rewriting-based forwards semantics for Maude-NPA and prove that the backwards symbolic semantics of Maude-NPA is sound and complete w.r.t. the forwards semantics.

As explained in the Introduction, designing a suitable forwards semantics requires much more than simply reversing the transition rules of the backwards semantics. The reader may notice that rules of type (1), (2), and (3) allow the definition of an intuitive forwards semantics associated to Maude-NPA (see [12]) but it works only for validation of a given execution sequence, since an initial state must contain all the strands, and not for searching for an attack, where the initial state must be empty and strands would have to be added during forwards search.

Our solution is based on two ideas. First, to match input terms in a strand always with the intruder's knowledge, so that the previous backwards semantics rule is defined in our forward semantics differently:

$$\begin{array}{l} \{SS \,\&\, \{K \in \mathcal{I}, M \in \mathcal{I}, IK\}\} \\ \rightarrow \{SS \,\&\, [-(K), -(M), +(e(K, M))] \\ \quad \&\, \{K \in \mathcal{I}, M \in \mathcal{I}, e(K, M) \in \mathcal{I}, IK\}\} \end{array}$$

Second, the restriction mentioned in Footnote 4 ensures that principal names as well as new fresh variables are the only extra variables in rules of this form. Thus, they can be treated as numeric constants by using a global counter $\langle N \rangle$ that will be appropriately incremented. For instance, the Dolev-Yao strand for new nonces $:: r :: [+(n(i, r))]$ will be represented by a transition rule of the form:

$$\begin{array}{l} \{SS \;\&\; \{IK\} \;\&\; \langle N \rangle\} \\ \rightarrow \{SS \;\&\; [+(n(i, N))] \;\&\; \{IK\} \;\&\; \langle N + 1 \rangle\} \end{array}$$

where the global counter N is incremented by one. The formal definition of how forwards transition rules are generated from the strand specification now requires some notation to indicate how the global counter is increased. Given a message u and a counter $\langle i \rangle$, we write $u\uparrow_i^n$, to denote that those principal names and fresh variables appearing in term u that are identified as new have been numbered starting with i and ending in $n - 1$, with n the next available value of the counter. For example, given the term $u = exp(g, n(0, 1) * n(A, r))$ where 0 corresponds to a principal name already replaced and 1 to a fresh variable already replaced, but A is a new principal name and r is a new fresh variable, we write $u\uparrow_{10}^{12} = exp(g, n(0, 1) * n(10, 11))$, i.e., the substitution $\{A \mapsto 10, r \mapsto 11\}$ has been applied. In the forwards semantics, we remove the list of fresh variables at the beginning of each strand and the vertical bar, since they are no longer necessary.

In Section 4.1 we give an overview of how a forwards protocol analysis can be performed in the strand-based model. Then in Section 4.2 we define a forwards *rewriting-based* semantics for Maude-NPA and, finally, in Section 4.3 we prove that the backwards symbolic semantics of Section 3.2 is sound and complete w.r.t. the forwards semantics.

4.1 Forward Reachability Analysis

In a forward execution of a protocol we begin with an empty initial state containing no information in the intruder knowledge and, since we consider the unbounded session case, no strand in the initial state. The execution of the protocol implies searching for a final state which is an instance of the pattern denoting the desired class of attack states.

Example 3. *Given the protocol of Example 1, the initial state is just the empty state:*

$$\{\varnothing \;\&\; \{empty\} \;\&\; \langle 0 \rangle\}$$

The final state pattern where the intruder has learned the secret is as follows, where Y, SR, SS, IK, r, and r' are variables and a and b represent the actual names of Alice and Bob:

$$\begin{array}{l} \{[-(a; b; Y), +(b; a; exp(g, n(b, r''))), -(e(exp(Y, n(b, r'')), SR))] \\ \;\&\; SS \;\&\; \{SR \in \mathcal{I}, IK\}\} \end{array}$$

The forwards analysis is easily performed in the Maude system by using Maude's `search` *command, which receives*

the initial term and the final pattern as input and generates the search state space. Similar to the backwards analysis, the solution to the forwards reachability analysis is as follows. The principal strands are (where ca, cb, cb′, r_1, r_2, r_3, r_4 are natural numbers but the actual value is irrelevant):

$$[+(ca; cb'; exp(g, n(ca, r_2))),\\ -(cb'; ca; exp(g, n(i, r_4))),\\ +(e(exp(g, n(i, r_4) * n(ca, r_2)), sec(ca, r_1)))]) \;\&$$
$$[-(ca; cb; exp(g, n(i, r_4))),\\ +(cb; ca; exp(g, n(cb, r_3))),\\ -(e(exp(g, n(i, r_4) * n(cb, r_3)), sec(ca, r_1)))] \;\&$$

The Dolev-Yao intruder strands are as follows:

$$[+(n(i, r_4))] \;\&$$
$$[-(ca; cb'; exp(g, n(ca, r_2))), +(cb'; exp(g, n(ca, r_2)))] \;\&$$
$$[-(cb'; exp(g, n(ca, r_2))), +(exp(g, n(ca, r_2)))] \;\&$$
$$[-(exp(g, n(ca, r_2))), -(n(i, r_4)),\\ +(exp(g, n(i, r_4) * n(ca, r_2)))] \;\&$$
$$[-(exp(g, n(i, r_4) * n(ca, r_2))),\\ -(e(exp(g, n(i, r_4) * n(ca, r_2)), sec(ca, r_1))),\\ +(sec(ca, r_1))] \;\&$$
$$[-(exp(g, n(cb, r_3))), -(n(i, r_4)),\\ +(exp(g, n(i, r_4) * n(cb, r_3)))] \;\&$$
$$[-(exp(g, n(i, r_4) * n(cb, r_3))), -(sec(ca, r_1)),\\ +(e(exp(g, n(i, r_4) * n(cb, r_3)), sec(ca, r_1)))] \;\&$$
$$[-(cb; ca; exp(g, n(cb, r_3))), +(ca; exp(g, n(cb, r_3)))] \;\&$$
$$[-(ca; exp(g, n(cb, r_3))), +(exp(g, n(cb, r_3)))]$$

It is also possible to specify authentication attacks in Maude for the forwards semantics by using again Maude's `search` command to search for an attack, but making it *conditional* to the never patterns not having been encountered. Note that the forwards semantics is *monotonic* in the sense that for s, s' concrete states such that $s \rightarrow^* s'$, then s' "stores" s as a "substate." This means that if a never pattern is avoided by s' it is also avoided by s. Therefore, given an attack pattern S and never patterns $S_1, \ldots, S_n$, we can search for an attack avoiding such patterns by giving to Maude the conditional search command:

$$\begin{array}{l} \mathit{search\ init} \rightarrow^* S \ \mathit{such\ that} \\ p_{S_1}(S) = \mathit{false} \wedge \ldots \wedge p_{S_n}(S) \;=\; \mathit{false}\ . \end{array}$$

where each predicate p_{S_i} holds for a concrete state s iff s is an instance of the pattern S_i. When a never pattern S_i shares variables $v_0, \ldots, v_{k_i}$ with S, the predicate p_{S_i} is extended in the form $p_{S_i}(S, v_0, \ldots, v_{k_i})$; if a never pattern has variables not appearing in S, these will be created and used within the predicate. This method has allowed us to analyze by forward model checking all the examples in Section 5.

EXAMPLE 4. *Given the following attack S, and never patterns $Alice_1$ and $Alice_2$ (not sharing variables with S) of the Diffie-Hellman protocol in Example 2:*

$$\begin{array}{rl} S \;=\; \{ :: r'' :: [-(a; b; Y), +(b; a; exp(g, n(b, r'')))), \\ \qquad -(e(exp(Y, n(b, r'')), SR)) \mid nil] \\ \qquad \& \; SS \; \& \; \{IK\}\} \\ Alice_1 = :: r, r' :: [+(a; b; exp(g, n(a, r)))] \\ Alice_2 = :: r, r' :: [+(a; b; exp(g, n(a, r))), -(b; a; X), \\ \qquad +(e(exp(X, n(a, r)), sec(a, r')))] \end{array}$$

The Maude conditional search command to search for the attack S avoding the never patterns for Alice's strand, is as follows:

$$\begin{array}{l} \mathit{search\ init} \rightarrow^* S \ \mathit{such\ that} \\ p_{Alice_1}(S) = \mathit{false} \wedge p_{Alice_2}(S) \;=\; \mathit{false}\ . \end{array}$$

where the predicates p_{Alice_1} and p_{Alice_2} check whether any strand of the concrete state S is an instance of the strands $Alice_1$ and $Alice_2$, respectively.

4.2 Forwards Operational Semantics

In a forward reachability analysis, we define state changes by means of a set $R_{F\mathcal{P}}$ of *rewrite rules*, so that the rewrite theory $(\Sigma_{\mathcal{P}}, E_{\mathcal{P}}, R_{F\mathcal{P}})$ characterizes the behavior of protocol $\mathcal{P}$ modulo the equations $E_{\mathcal{P}}$. Here we do not have generic transition rules, as in the backwards semantics, and all the rules are generated from principal and intruder strands. The intuitive idea is that a state consists of a multiset of partially executed strands and a set of terms in the intruder's knowledge. Unlike the backwards semantics, only the part of the strand that has already executed is present in the state, and each such partial strand instantiates a prefix of a strand in $\mathcal{P}$. One progresses by either: (i) adding a positive term m^+ to an existing strand and either adding or not adding m to the intruder's knowledge, (ii) adding a negative term m^- to an existing strand only if it is already present in the intruder's knowledge, or (iii) starting a new strand, and if it starts with a m^+ that either adds or not to the intruder's knowledge. For example, the intruder encryption capability $[-(K), -(M), +(e(K, M)]$ produces the rewrite rules:

$$\begin{array}{l} \{SS \,\&\, \{K{\in}\mathcal{I}, IK\} \,\&\, \langle N\rangle\} \\ \rightarrow \{SS \,\&\, [-(K)] \,\&\, \{K{\in}\mathcal{I}, IK\} \,\&\, \langle N\rangle\} \\ \{SS \,\&\, [-(K)] \,\&\, \{M{\in}\mathcal{I}, IK\} \,\&\, \langle N\rangle\} \\ \rightarrow \{SS \,\&\, [-(K), -(M)] \,\&\, \{M{\in}\mathcal{I}, IK\} \,\&\, \langle N\rangle\} \\ \{SS \,\&\, [-(K), -(M)] \,\&\, \{IK\} \,\&\, \langle N\rangle\} \\ \rightarrow \{SS \,\&\, [-(K), -(M), +(e(K, M)] \,\&\, \\ \quad \{e(K, M){\in}\mathcal{I}, IK\} \,\&\, \langle N\rangle\} \end{array}$$

The sets of rewrite rules for output messages are generated as follows, note that some rewrite rules are conditional:

$$\left\{ \begin{array}{l} \forall\ [u_1^{\pm}, \ldots, u_{j-1}^{\pm}, u_j^{+}, u_{j+1}^{\pm}, \ldots, u_n^{\pm}] \in \mathcal{P} \wedge j > 1 : \\ \{SS \,\&\, \{IK\} \,\&\, [u_1^{\pm}, \ldots, u_{j-1}^{\pm}] \,\&\, \langle N\rangle\} \\ \rightarrow \\ \{SS \,\&\, \{u_j{\uparrow}_N^M{\in}\mathcal{I}, IK\} \,\&\, [u_1^{\pm}, \ldots, u_{j-1}^{\pm}, (u_j{\uparrow}_N^M)^+] \,\&\, \langle M\rangle\} \\ \qquad \mathsf{IF}\ (u_j{\uparrow}_N^M{\in}\mathcal{I}) \notin IK \end{array} \right\} \quad (5)$$

$$\left\{ \begin{array}{l} \forall\ [u_1^{\pm}, \ldots, u_{j-1}^{\pm}, u_j^{+}, u_{j+1}^{\pm}, \ldots, u_n^{\pm}] \in \mathcal{P} \wedge j > 1 : \\ \{SS \,\&\, \{IK\} \,\&\, [u_1^{\pm}, \ldots, u_{j-1}^{\pm}] \,\&\, \langle N\rangle\} \\ \rightarrow \{SS \,\&\, \{IK\} \,\&\, [u_1^{\pm}, \ldots, u_{j-1}^{\pm}, (u_j{\uparrow}_N^M)^+] \,\&\, \langle M\rangle\} \end{array} \right\} \quad (6)$$

$$\left\{ \begin{array}{l} \forall\ [u_1^{+}, \ldots, u_n^{\pm}] \in \mathcal{P} : \\ \{SS \,\&\, \{IK\} \,\&\, \langle N\rangle\} \\ \rightarrow \{SS \,\&\, [(u_1{\uparrow}_N^M)^+] \,\&\, \{u_1{\uparrow}_N^M{\in}\mathcal{I}, IK\} \,\&\, \langle M\rangle\} \\ \qquad \mathsf{IF}\ (u_1{\in}\mathcal{I}{\uparrow}_N^M) \notin IK \end{array} \right\} \quad (7)$$

$$\left\{ \begin{array}{l} \forall\ [u_1^{+}, \ldots, u_n^{\pm}] \in \mathcal{P} : \\ \{SS \,\&\, \{IK\} \,\&\, \langle N\rangle\} \rightarrow \{SS \,\&\, [(u_1{\uparrow}_N^M)^+] \,\&\, \{IK\} \,\&\, \langle M\rangle\} \end{array} \right\} \quad (8)$$

Each transition rule of type (5) accepts output messages and the intruder's knowledge is positively increased, while each transition rule of type (6) simply accepts output messages without modifying the intruder's knowledge. Each transition rule in (7) and (8) introduces a new strand beginning with an output message. Similarly, rules of type (7) introduce a new strand and the intruder's knowledge is positively increased, whereas rules of type (8) introduce a new strand but the intruder's knowledge is not increased [6] .

The following set of rewrite rules describes the general state transition for a negative message, generating specific rewrite rules according to the protocol strands:

$$\left\{\begin{array}{l}\forall\ [u_1^{\pm},\ldots,u_{j-1}^{\pm},u_j^{-},u_{j+1}^{\pm},\ldots,u_n^{\pm}]\in\mathcal{P}\wedge j>1:\\ \{SS\ \&\{u_j{\in}\mathcal{I},IK\}\,\&\,[u_1^{\pm},\ldots,u_{j-1}^{\pm}]\,\&\,\langle N\rangle\}\\ \rightarrow\{SS\,\&\,\{u_j{\in}\mathcal{I},IK\}\,\&\,[u_1^{\pm},\ldots,u_{j-1}^{\pm},u_j^{-}]\,\&\,\langle N\rangle\}\end{array}\right\}\quad(9)$$

$$\left\{\begin{array}{l}\forall[u_1^{-},u_2^{\pm},\ldots,u_n^{\pm}]\in\mathcal{P}:\\ \{SS\,\&\,\{u_1{\in}\mathcal{I},IK\}\,\&\,\langle N\rangle\}\\ \rightarrow\{SS\ \&\ [u_1^{-}]\,\&\,\{u_1{\in}\mathcal{I},IK\}\,\&\,\langle N\rangle\}\end{array}\right\}\quad(10)$$

Each transition rule in (9) and (10) accepts input messages if the intruder's knowledge matches them. Note that in (10) a new strand is introduced.

DEFINITION 2. *Let $\mathcal{P}$ be a protocol with signature $\Sigma_{\mathcal{P}}$ and equational theory $E_{\mathcal{P}}$. We define the* forward rewrite theory characterizing $\mathcal{P}$ *to be* $(\Sigma_{\mathcal{P}},E_{\mathcal{P}},R_{F\mathcal{P}})$ *where* $R_{F\mathcal{P}}=(5)\cup(6)\cup(7)\cup(8)\cup(9)\cup(10)$.

EXAMPLE 5. *The rewrite rules associated to Alice's strand in the forwards semantics of our running example are as follows, where the increment of the global counter can be clearly identified. Alice's strand is defined as*

$$::: r,r' ::: [+(A;B;exp(g,n(A,r))),-(B;A;X),\\ +(e(exp(X,n(A,r)),sec(A,r')))]$$

and the rewrite rules associated to it are as follows:

$$\begin{array}{l}\{SS\ \&\ \{IK\}\ \&\ \langle N\rangle\}\\ \rightarrow\{SS\ \&\ [+(N;N+1;exp(g,n(N,N+2)))]\ \&\\ \quad\{(N;N+1;exp(g,n(N,N+2)))\in\mathcal{I},IK\}\ \&\ \langle N+3\rangle\}\end{array}$$

$$\begin{array}{l}\{SS\ \&\ [+(A;B;exp(g,n(A,R)))]\ \&\\ \ \{(B;A;X)\in\mathcal{I},IK\}\ \&\ \langle N\rangle\}\\ \rightarrow\{SS\ \&\ [+(A;B;exp(g,n(A,R))),-(B;A;X)]\ \&\\ \quad\{(B;A;X)\in\mathcal{I},IK\}\ \&\ \langle N\rangle\}\end{array}$$

$$\begin{array}{l}\{SS\ \&\ [+(A;B;exp(g,n(A,R))),-(B;A;X)]\ \&\\ \ \{IK\}\ \&\ \langle N\rangle\}\\ \rightarrow\{SS\ \&\ [+(A;B;exp(g,n(A,R))),-(B;A;X),\\ \qquad+(e(exp(X,n(A,R)),sec(A,N)))]\ \&\\ \quad\{e(exp(X,n(A,R)),sec(A,N)))\in\mathcal{I},IK\}\ \&\ \langle N+1\rangle\}\end{array}$$

When the intuder impersonates Bob, the first rule is:

$$\begin{array}{l}\{SS\ \&\ \{IK\}\ \&\ \langle N\rangle\}\\ \rightarrow\{SS\ \&\ [+(N;i;exp(g,n(N,N+1)))]\ \&\\ \quad\{(N;i;exp(g,n(N,N+1)))\in\mathcal{I},IK\}\ \&\ \langle N+2\rangle\}\end{array}$$

where i is a constant denoting the intruder's name. Note that it is not necessary to duplicate the other two rules.

[6]Note that the use of the global counter for new principal names in previous rules has to take into account when one of those principals is indeed the intruder; see Example 5 for the case in which the intruder impersonates Bob.

4.3 Soundness and Completeness of the Forwards Semantics

In the previous section we defined the rewriting-based forwards semantics for Maude-NPA. Now we need to prove that the backwards operational semantics of Maude-NPA is sound and complete w.r.t. to this semantics. We first introduce some definitions and concepts that will be used in these proofs. First, we define what a symbolic state is, i.e., a state with variables.

DEFINITION 3 (SYMBOLIC $\mathcal{P}$-STATE). *Given a protocol $\mathcal{P}$, a* symbolic $\mathcal{P}$*-state S is a term of the form:*

$$\begin{array}{l}S=\{::r_{1_1},\ldots,r_{m_1}::[u_{1_1}^{\pm},\ldots u_{i_1-1}^{\pm}\mid u_{i_1}^{\pm},\ldots,u_{n_1}^{\pm}]\ \&\\ \qquad\vdots\\ \qquad::r_{1_k},\ldots,r_{m_k}::[u_{1_k}^{\pm},\ldots,u_{i_k-1}^{\pm}\mid u_{i_k}^{\pm},\ldots,u_{n_k}^{\pm}]\ \&\ SS\\ \qquad\{w_1{\in}\mathcal{I},\ldots,w_m{\in}\mathcal{I},\ w_1'{\notin}\mathcal{I},\ldots,w_{m'}'{\notin}\mathcal{I},IK\}\}\end{array}$$

where for each $1\leqslant j\leqslant k$, there exists a strand $[m_{1_j}^{\pm},\ldots m_{i_j-1}^{\pm},m_{i_j}^{\pm},\ldots,m_{n_j}^{\pm}]\in\mathcal{P}$ and a substitution $\rho_j:\mathcal{X}\rightarrow\mathcal{T}_\Sigma(\mathcal{X})$ such that $m_{1_j}\rho_j=_{E_{\mathcal{P}}}u_{1_j},\ \ldots,\ m_{n_j}\rho_j=_{E_{\mathcal{P}}}u_{n_j}$, SS is a variable denoting a (possibly empty) set of strands, and IK is a variable denoting a (possibly empty) set of intruder's knowledge facts.

Second, we define what a ground state is, i.e., a state without variables.

DEFINITION 4 (GROUND $\mathcal{P}$-STATE). *Given a protocol $\mathcal{P}$, a* ground $\mathcal{P}$*-state s is a term without variables of the form:*

$$\begin{array}{l}s=\{[u_{1_1}^{\pm},\ldots u_{i_1-1}^{\pm}]\ \&\cdots\&\ [u_{1_k}^{\pm},\ldots,u_{i_k-1}^{\pm}]\ \&\\ \qquad\{w_1{\in}\mathcal{I},\ldots,w_m{\in}\mathcal{I}\}\ \&\ \langle J\rangle\}\end{array}$$

where for each $1\leqslant j\leqslant k$, there exists a strand $[m_{1_j}^{\pm},\ldots m_{i_j-1}^{\pm},m_{i_j}^{\pm},\ldots,m_{n_j}^{\pm}]\in\mathcal{P}$ and a substitution $\rho_j:\mathcal{X}\rightarrow\mathcal{T}_\Sigma$ such that $m_{1_j}\rho_j=_{E_{\mathcal{P}}}u_{1_j},\ \ldots,\ m_{i_j}\rho_j=_{E_{\mathcal{P}}}u_{i_j}$.

Third, we define a suitable instantiation relation between symbolic and ground states.

DEFINITION 5 (LIFTING RELATION). *Given a symbolic $\mathcal{P}$-state S and a ground state s we say that s* lifts *to S, or that S* instantiates *to s with a* grounding *substitution $\theta:(Var(S)-\{SS,IK\})\rightarrow\mathcal{T}_\Sigma$, writen $S>^{\theta}s$ iff*

- *for each strand $::r_1,\ldots,r_m::[u_1^{\pm},\ldots u_{i-1}^{\pm}\mid u_i^{\pm},\ldots,u_n^{\pm}]$ in S, there exists a strand $[v_1^{\pm},\ldots v_{i-1}^{\pm}]$ in s such that $\forall 1\leqslant j\leqslant i-1$, $v_j=_{E_{\mathcal{P}}}u_j\theta$.*
- *for each positive intruder fact $w{\in}\mathcal{I}$ in S, there exists a positive intruder fact $w'{\in}\mathcal{I}$ in s such that $w'=_{E_{\mathcal{P}}}w\theta$, and*
- *for each negative intruder fact $w{\notin}\mathcal{I}$ in S, there is no positive intruder fact $w'{\in}\mathcal{I}$ in s such that $w'=_{E_{\mathcal{P}}}w\theta$.*

Let us now prove that narrowing with the backwards rewrite theory $(\Sigma_{\mathcal{P}},E_{\mathcal{P}},R_{B\mathcal{P}}^{-1})$ is complete with respect to rewriting with the forwards rewrite theory $(\Sigma_{\mathcal{P}},E_{\mathcal{P}},R_{F\mathcal{P}})$. First, the lemma below shows how the lifting of a ground term to a symbolic state induces a lifting of a forward rewriting step in the forwards semantics to a backwards narrowing step in the backwards semantics. This will be used to prove

Theorem 1, which allows us to lift a rewriting sequence in the forwards semantics to a narrowing sequence in the backwards semantics.

LEMMA 1 (LIFTING LEMMA). *Given a protocol $\mathcal{P}$, two states s and s', a $\mathcal{P}$-symbolic state S' and a substitution θ' s.t. $s \rightarrow s'$ and $S' >^{\theta'} s'$, then there exist a $\mathcal{P}$-symbolic state S and a substitution θ s.t $S >^{\theta} s$ and either $S' \leftsquigarrow S$ or $S = S'$.*

The following theorem states that the symbolic reachability analysis is *complete* with respect to the forwards rewriting-based semantics, i.e., any concrete attack state s, matching an attack pattern S and reachable by the forwards semantics from a concrete initial state s_0 can be *found* by backwards symbolic reachability analysis from the attack pattern S. Its proof is a straightforward corollary of Lemma 1.

THEOREM 1 (COMPLETENESS). *Given a protocol $\mathcal{P}$, two ground states s, s_0, a symbolic $\mathcal{P}$-state S, a substitution θ s.t. (i) s_0 is an initial state, (ii) $s_0 \rightarrow^n s$, and (iii) $S >^{\theta} s$ then there exist a symbolic initial $\mathcal{P}$-state S_0, two substitutions μ and θ', and $k \leqslant n$, s.t. $S_0 \overset{k}{\leftsquigarrow}_{\mu} S$, and $S_0 >^{\theta'} s_0$.*

In the following we prove that the backwards rewrite theory $(\Sigma_{\mathcal{P}}, E_{\mathcal{P}}, R_{B\mathcal{P}}^{-1})$ is *sound* with respect to the forward rewrite theory $(\Sigma_{\mathcal{P}}, E_{\mathcal{P}}, R_{F\mathcal{P}})$. That is, we need to show that if we find a symbolic initial state S_0 from a symbolic attack pattern S then, for any concrete initial state s_0 such that $S_0 >^{\theta} s_0$, there is a reachable concrete attack state such that $s_0 \rightarrow_s^*$ with $S >^{\theta'} s$. We first provide a lemma that says that for any backwards narrowing step there exist a corresponding sequence of forwards rewriting steps.

LEMMA 2. *Given a protocol $\mathcal{P}$, two symbolic $\mathcal{P}$-states S, S', a ground state s and a substitution θ, if $S \overset{\mu}{\leftsquigarrow} S'$ and $S >^{\theta} s$, then there exist a state s' and a substitution θ' such that $s \rightarrow s'$, and $S' >^{\theta'} s'$.*

The following theorem is a straightforward corollary of Lemma 2. It proves that the symbolic reachability analysis is *sound* with respect to the forwards rewriting-based semantics,

THEOREM 2 (SOUNDNESS). *Given a protocol $\mathcal{P}$, two symbolic $\mathcal{P}$-states S_0, S', an initial ground state s_0 and a substitution θ s.t. (i) S_0 is a symbolic initial state, and (ii) $S_0 \overset{*}{\leftsquigarrow} S'$, and (iii) $S_0 >^{\theta} s_0$ then there exist a ground state s' and a substitution θ', s.t. (i) $s_0 \rightarrow^* s'$, and (ii) $S' >^{\theta'} s'$.*

The proofs of Lemmas 1 and 2 are provided in Appendix A.

5. EXPERIMENTS

We have performed several experiments to evaluate the feasibility of the rewriting-based forwards semantics defined in Section 4. We have used four protocols to perform five experiments: (i) the standard Needham-Schroeder protocol (NSPK) [20], (ii) a version of the Needham-Schroeder-Lowe protocol in which one of the concatenation operators is replaced by an exclusive-or, presented in [21] (NSL-XOR), (iii) the Denning-Sacco Symmetric Key protocol [10], and (iv) a protocol with Diffie-Hellman exponentiation. More specifically, we have verified both secrecy and authentication properties for NSPK, secrecy properties for XOR-NSL and Diffie-Hellman, and authentication properties for Denning-Sacco. Since the forwards rewrite-based semantics defined in this paper does not include optimizations to reduce the search space and, therefore, is not currently possible to obtain a finite search space, we have analyzed insecure protocols, i.e., protocols with known secrecy and/or authentication attacks. The specifications of these protocols and more detailed information can be found at `http://www.dsic.upv.es/~ssantiago/forwards-semantics.html`.

Table 1: Rewrite steps until finding the attack

Protocol	Length Forwards
NSPK-sec	9
NSPK-auth	9
NSL-XOR	13
Denning-Sacco	11
Diffie-Hellman	22

Table 2: States generated in each rewrite step

Protocol	1	2	3	4	5
NSPK-sec	6	20	116	604	3026
NSPK-auth	6	20	116	604	3026
NSL-XOR	7	21	72	218	594
Denning-Sacco	7	28	132	596	2624
Diffie-Hellman	7	22	65	162	354

Table 1 provides an experimental validation of the implementation of the forwards semantics defined in this paper in Maude w.r.t. the symbolic backwards operational semantics of Maude-NPA, since for each protocol the forwards search found the same authentication or secrecy attacks that are found by Maude-NPA.

Table 2 gathers for each experiment the number of states generated during the first five steps of the forwards search. The reader can check that the number of generated states is the same for both experiments of the NSPK protocol, since the search space is the same and only the attack being searched is different. We used experimental heuristics to decrease the size of the state space. However systematic study of state space reduction techniques in the forwards semantics is left for future work.

Summarizing, the results of our early experimental evaluation suggest that, even though its implementation is still at an early stage, the forwards semantics presented in this paper is feasible and encouraging. However, much work needs to be done, specially with respect to the efficiency of the analysis.

6. CONCLUSIONS

In this paper we have presented a forwards rewriting-based semantics for Maude-NPA (or protocol analysis tools based on strands). We have proved that the backwards narrowing-based semantics of Maude-NPA is sound and complete w.r.t. the forwards semantics. This work serves two purposes. First, it gives the definition of a novel rewriting-based protocol analysis, where the new forwards semantics is directly implementable in rule-based programming languages such as Maude without any need for constraint solving or unification procedures as it is done in most current

approaches. Second, it can be used as a logical foundation for the Maude-NPA tool, from which the analyses are proved sound, thus linking the results of the Maude-NPA backwards execution with the results of the rewriting-based forward execution. Here the Maude-NPA already had an intuitive forwards semantics, but it was not optimized for model checking, and so could not be used in this way. This work also reduces the gap between the Maude-NPA and the realm of standard model checking, shedding some light on how its internal semantics and the logical reachability analysis correspond to an intuitive forward execution of a protocol with the intruder model. This opens up several research directions: the integration of Maude-NPA state reduction techniques into the forwards semantics, clarification of the relation of equational theories in the forward semantics, and investigation of how standard model-checking techniques can improve the protocol analysis in the forwards semantics. Also, there is a vast literature in term rewriting and tree automata on forward and backwards reachability analysis and their pros and cons, which is outside the scope of this paper, but that should be very useful for improving the forwards analysis.

7. ACKNOWLEDGMENTS

Santiago Escobar and Sonia Santiago have been partially supported by the Spanish MICINN, under grant TIN2010-21062-C02-02, and by Generalitat Valenciana under grant PROMETEO2011/052. José Meseguer has been partially supported by NSF Grants CNS 09-04749, CCF 09-05584 and CNS 13-19109.

8. REFERENCES

[1] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *POPL*, pages 104–115, 2001.

[2] D. Basin, S. Mödersheim, and L. Viganò. An on-the-fly model-checker for security protocol analysis. In *In Proceedings of Esorics'03, LNCS 2808*, pages 253–270. Springer-Verlag, 2003.

[3] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *CSFW*, pages 82–96. IEEE Computer Society, 2001.

[4] R. Chadha, S. Kremer, and A. Scedrov. Formal analysis of multiparty contract signing. *J. Autom. Reasoning*, 36(1-2):39–83, 2006.

[5] Y. Chevalier and M. Rusinowitch. Hierarchical combination of intruder theories. In F. Pfenning, editor, *RTA*, volume 4098 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2006.

[6] E. M. Clarke, S. Jha, and W. R. Marrero. Verifying security protocols with brutus. *ACM Trans. Softw. Eng. Methodol.*, 9(4):443–487, 2000.

[7] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. L. Talcott, editors. *All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*, volume 4350 of *Lecture Notes in Computer Science*. Springer, 2007.

[8] H. Comon-Lundh and S. Delaune. The finite variant property: How to get rid of some algebraic properties. In *RTA*, pages 294–307, 2005.

[9] C. J. F. Cremers. The Scyther tool: Verification, falsification, and analysis of security protocols. In *CAV*, pages 414–418, 2008.

[10] D. E. Denning and G. M. Sacco. Timestamps in key distribution protocols. *Commun. ACM*, 24(8):533–536, 1981.

[11] S. Erbatur, S. Escobar, D. Kapur, Z. Liu, C. Lynch, C. Meadows, J. Meseguer, P. Narendran, S. Santiago, and R. Sasse. Effective symbolic protocol analysis via equational irreducibility conditions. In *ESORICS*, pages 73–90, 2012.

[12] S. Escobar, C. Meadows, and J. Meseguer. Maude-NPA: Cryptographic protocol analysis modulo equational properties. In *Foundations of Security Analysis and Design V, FOSAD 2007/2008/2009 Tutorial Lectures*, LNCS vol. 5705, pages 1–50. Springer, 2009.

[13] F. J. T. Fabrega, J. Herzog, and J. Guttman. Strand Spaces: What Makes a Security Protocol Correct? *Journal of Computer Security*, 7:191–230, 1999.

[14] J.-P. Jouannaud and H. Kirchner. Completion of a set of rules modulo a set of equations. *SIAM J. Comput.*, 15(4):1155–1194, 1986.

[15] G. Lowe. Breaking and fixing the Needham-Schroeder public key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS '96)*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer-Verlag, 1996.

[16] S. Meier, B. Schmidt, C. Cremers, and D. A. Basin. The tamarin prover for the symbolic analysis of security protocols. In *CAV*, pages 696–701, 2013.

[17] J. Meseguer. Conditional rewriting logic as a united model of concurrency. *Theor. Comput. Sci.*, 96(1):73–155, 1992.

[18] J. Meseguer. Membership algebra as a logical framework for equational specification. In F. Parisi-Presicce, editor, *WADT*, volume 1376 of *Lecture Notes in Computer Science*, pages 18–61. Springer, 1997.

[19] J. C. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using mur-phi. In *IEEE Symposium on Security and Privacy*, pages 141–151, 1997.

[20] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.

[21] R. Sasse, S. Escobar, C. Meadows, and J. Meseguer. Protocol analysis modulo combination of theories: A case study in maude-npa. In J. Cuéllar, J. Lopez, G. Barthe, and A. Pretschner, editors, *STM*, volume 6710 of *Lecture Notes in Computer Science*, pages 163–178. Springer, 2010.

[22] TeReSe, editor. *Term Rewriting Systems*. Cambridge University Press, Cambridge, 2003.

APPENDIX

A. PROOF OF THEOREMS IN SECTION 4

Proof of Lemma 1

First of all, all the forward rewriting rules act on the ground state s' by either adding one more element to an existing strand (rules (5), (6) , and (9)), adding a positive fact to the intruder knowledge (rules (5) and (7)), adding a new strand (rules (7), (8), and (9)), or repeating a positive intruder fact that is already in s' (rules (9) and (10)). This allows us to identify six cases for the grounding substitution θ' of S' into s', depending upon whether the grounding substitution of S under θ contains the relevant strands and positive intruder facts.

a) There is a strand $[u_1^{\pm}, \ldots, u_{i-1}^{\pm}, u_i^{\pm}, \ldots, u_n^{\pm}]$ in $\mathcal{P}$, $n \geqslant 1$, $1 \leqslant i \leqslant n$, and a substitution ρ such that $[u_1^{\pm}, \ldots, u_{i-1}^{\pm}, u_i^{\pm}]\rho$ is a strand in s', $[u_1^{\pm}, \ldots, u_{i-1}^{\pm} \mid u_i^{\pm}, \ldots, u_n^{\pm}]\rho$ is a strand in $S'\theta'$, and $u_i\rho \in \mathcal{I}$ appears in the intruder knowledge of $S'\theta'$. This is valid for rules in sets (5), (7), (9), and (10). If $i > 1$, then we also know that $[u_1^{\pm}, \ldots, u_{i-1}^{\pm}]\rho$ is a strand in s.

b) There is a strand $[u_1^{\pm}, \ldots, u_{i-1}^{\pm}, u_i^{\pm}, \ldots, u_n^{\pm}]$ in $\mathcal{P}$, $n \geqslant 1$, $1 \leqslant i \leqslant n$, and a substitution ρ such that $[u_1^{\pm}, \ldots, u_{i-1}^{\pm}, u_i^{\pm}]\rho$ is a strand in s', $[u_1^{\pm}, \ldots, u_{i-1}^{\pm} \mid u_i^{\pm}, \ldots, u_n^{\pm}]\rho$ is a strand in $S'\theta'$, but $u_i\rho \in \mathcal{I}$ does not appear in the intruder knowledge of $S'\theta'$. This is valid for rules in sets (5), (7), (9), and (10). If $i > 1$, then we also know that $[u_1^{\pm}, \ldots, u_{i-1}^{\pm}]\rho$ is a strand in s.

c) There is a strand $[u_1^{\pm}, \ldots, u_{i-1}^{\pm}, u_i^{\pm}, \ldots, u_n^{\pm}]$ in $\mathcal{P}$, $n \geqslant 1$, $1 \leqslant i \leqslant n$, and a substitution ρ such that $[u_1^{\pm}, \ldots, u_{i-1}^{\pm}, u_i^{\pm}]\rho$ is a strand in s', $u_i\rho \in \mathcal{I}$ appears in the intruder knowledge of $S'\theta'$, but $[u_1^{\pm}, \ldots, u_{i-1}^{\pm} \mid u_i^{\pm}, \ldots, u_n^{\pm}]\rho$ is not a strand in $S'\theta'$. This is valid for rules in sets (5), (7), (9), and (10). If $i > 1$, then we also know that $[u_1^{\pm}, \ldots, u_{i-1}^{\pm}]\rho$ is a strand in s.

d) There is a strand $[u_1^{\pm}, \ldots, u_{i-1}^{\pm}, u_i^{\pm}, \ldots, u_n^{\pm}]$ in $\mathcal{P}$, $n \geqslant 1$, $1 \leqslant i \leqslant n$, and a substitution ρ such that $[u_1^{\pm}, \ldots, u_{i-1}^{\pm}, u_i^{\pm}]\rho$ is a strand in s' but $u_i\rho \in \mathcal{I}$ does not appear in the intruder knowledge of $S'\theta'$ and $[u_1^{\pm}, \ldots, u_{i-1}^{\pm} \mid u_i^{\pm}, \ldots, u_n^{\pm}]\rho$ is not a strand in $S'\theta'$. This is valid for rules in sets (5), (7), (9), and (10). If $i > 1$, then we also know that $[u_1^{\pm}, \ldots, u_{i-1}^{\pm}]\rho$ is a strand in s.

e) There is a strand $[u_1^{\pm}, \ldots, u_{i-1}^{\pm}, u_i^{\pm}, \ldots, u_n^{\pm}]$ in $\mathcal{P}$, $n \geqslant 1$, $1 \leqslant i \leqslant n$, and a substitution ρ such that $[u_1^{\pm}, \ldots, u_{i-1}^{\pm}, u_i^{\pm}]\rho$ is a strand in s' and $[u_1^{\pm}, \ldots, u_{i-1}^{\pm} \mid u_i^{\pm}, \ldots, u_n^{\pm}]\rho$ is a strand in $S'\theta'$. This is valid for rules in sets (6) and (8). If $i > 1$, then we also know that $[u_1^{\pm}, \ldots, u_{i-1}^{\pm}]\rho$ is a strand in s.

f) There is a strand $[u_1^{\pm}, \ldots, u_{i-1}^{\pm}, u_i^{\pm}, \ldots, u_n^{\pm}]$ in $\mathcal{P}$, $n \geqslant 1$, $1 \leqslant i \leqslant n$, and a substitution ρ such that $[u_1^{\pm}, \ldots, u_{i-1}^{\pm}, u_i^{\pm}]\rho$ is a strand in s' but $[u_1^{\pm}, \ldots, u_{i-1}^{\pm} \mid u_i^{\pm}, \ldots, u_n^{\pm}]\rho$ is not a strand in $S'\theta'$. This is valid for rules in sets (6) and (8). If $i > 1$, then we also know that $[u_1^{\pm}, \ldots, u_{i-1}^{\pm}]\rho$ is a strand in s.

Now, we consider each forward rewrite rule application in the step $s \rightarrow s'$.

- Given states s and s' such that $s \rightarrow s'$ using a rule in set (5), then there exist a substitution τ, variables SS' and IK', and a strand $[u_1^{\pm}, \ldots, u_{j-1}^{\pm}, u_j^{+}, u_{j+1}^{\pm}, \ldots, u_n^{\pm}]$ in $\mathcal{P}$ such that $s = \{SS'\tau \,\&\, \{IK'\tau\} \,\&\, [(u_1\tau)^{\pm}, \ldots, (u_{j-1}\tau)^{\pm}]\}$, and $s' = \{SS'\tau \,\&\, \{(u_j\tau) \in \mathcal{I}, IK'\tau\} \,\&\, [(u_1\tau)^{\pm}, \ldots, (u_{j-1}\tau)^{\pm}, (u_j\tau)^{+}]\}$ and $(u_j\tau) \in \mathcal{I}$ appears in $IK'\tau$. Since there exists a substitution θ' s.t $S' >^{\theta'} s'$, we consider the four applicable cases for substitution θ':
  - *Case a)* Both the strand and the intruder fact appear in $S'\theta'$ and thus we can perform a backwards narrowing step from S' with rule (3) to obtain a state S, i.e., $S' \rightsquigarrow S$. Since there is no extra variable in the rule, we have that the same substitution θ' is valid for S and $S >^{\theta'} s$.
  - *Case b)* The strand appears in $S'\theta'$ but not the intruder fact. We also perform a backwards narrowing step from S' with rule (3) to obtain a state S, i.e., $S' \rightsquigarrow_{\sigma} S$. But the variable IK in state S' gets instantiated $\sigma = \{IK \mapsto w \in \mathcal{I}, IK''\}$ in such a way that $w\theta' =_{E_{\mathcal{P}}} u_j\tau$. Since there is no extra variable in the rule, again $S >^{\theta'} s$.
  - *Case c)* The intruder fact appears in $S'\theta'$ but not the strand. Here we perform a backwards narrowing step from S' with a rule in set (4) to obtain a state S, i.e., $S' \rightsquigarrow S$. This rules introduces a new strand into the symbolic state S, i.e., there is a substitution γ such that $[(u_1\gamma)^{\pm}, \ldots, (u_{j-1}\gamma)^{\pm} \mid (u_j\gamma)^{+}, (u_{j+1}\gamma)^{\pm}, \ldots, (u_n\gamma)^{\pm}]$ is a strand in S. Note that this new strand contains variables but there is a substitution θ such that $S >^{\theta} s$, since $[(u_1\gamma\theta)^{\pm}, \ldots, (u_{j-1}\gamma\theta)^{\pm}]$ corresponds to $[(u_1\tau)^{\pm}, \ldots, (u_{j-1}\tau)^{\pm}]$.
  - *Case d)* The strand and the intruder fact do not appear in $S'\theta'$. This case is very simple, since θ' makes valid S' as a symbolic state of s, i.e., $S = S'$ and $S' >^{\theta'} s$.
- Given states s and s' such that $s \rightarrow s'$ using a rule in set (6), then there exist a substitution τ, variables SS' and IK', and a strand $[u_1^{\pm}, \ldots, u_{j-1}^{\pm}, u_j^{+}, u_{j+1}^{\pm}, \ldots, u_n^{\pm}]$ in $\mathcal{P}$ such that $s = \{SS'\tau \,\&\, \{IK'\tau\} \,\&\, [(u_1\tau)^{\pm}, \ldots, (u_{j-1}\tau)^{\pm}]\}$, and $s' = \{SS'\tau \,\&\, \{IK'\tau\} \,\&\, [(u_1\tau)^{\pm}, \ldots, (u_{j-1}\tau)^{\pm}, (u_j\tau)^{+}]\}$. Since there exists a substitution θ' s.t $S' >^{\theta'} s'$, we consider the two applicable cases for substitution θ':
  - *Case e)* The strand appears in $S'\theta'$ and thus we can perform a backwards narrowing step from S' with rule (2) to obtain a state S, i.e., $S' \rightsquigarrow S$. Since there is no extra variable in the rule, we have that the same substitution θ' is valid for S and $S >^{\theta'} s$.
  - *Case f)* The strand does not appear in $S'\theta'$. This case is very simple, since θ' makes valid S' as a symbolic state of s, i.e., $S = S'$ and $S' >^{\theta'} s$.
- Given states s and s' such that $s \rightarrow s'$ using a rule in set (7), then there exist a substitution τ, variables SS' and IK', and a strand $[u_1^{\pm}, \ldots, u_n^{\pm}]$ in $\mathcal{P}$ such that $s' = \{SS'\tau \,\&\, \{(u_1\tau) \in \mathcal{I}, IK'\tau\} \,\&\, [(u_1\tau)^{\pm}]\}$ and $(u_j\tau) \in \mathcal{I}$ does not appear in $IK'\tau$. This is similar to the case above of a rule in set (5).

- Given states s and s' such that $s \rightarrow s'$ using a rule in set (8), then there exist a substitution τ, variables SS' and IK', and a strand $[u_1^{\pm}, \ldots, u_n^{\pm}]$ in $\mathcal{P}$ such that $s' = \{SS'\tau \,\&\, \{IK'\tau\} \,\&\, [(u_1\tau)^{\pm}]\}$. This is similar to the case above of a rule in set (6).
- Given states s and s' such that $s \rightarrow s'$ using a rule in set (9), then there exist a substitution τ, variables SS' and IK', and a strand $[u_1^{\pm}, \ldots, u_{j-1}^{\pm}, u_j^{+}, u_{j+1}^{\pm}, \ldots, u_n^{\pm}]$ in $\mathcal{P}$ such that $s = \{SS'\tau \,\&\, \{(u_j\tau) \in \mathcal{I}, IK'\tau\} \,\&\, [(u_1\tau)^{\pm}, \ldots, (u_{j-1}\tau)^{\pm}]\}$, and $s' = \{SS'\tau \,\&\, \{(u_j\tau) \in \mathcal{I}, IK'\tau\} \,\&\, [(u_1\tau)^{\pm}, \ldots, (u_{j-1}\tau)^{\pm}, (u_j\tau)^{-}]\}$. Since there exists a substitution θ' s.t $S' >^{\theta'} s'$, we consider the four applicable cases for substitution θ':
  - *Case a)* Both the strand and the intruder fact appear in $S'\theta'$ and thus we can perform a backwards narrowing step from S' with rule (1) to obtain a state S, i.e., $S' \rightsquigarrow S$. Since there is no extra variable in the rule, we have that the same substitution θ' is valid for S and $S >^{\theta'} s$.
  - *Case b)* The strand appears in $S'\theta'$ but not the intruder fact. We also perform a backwards narrowing step from S' with rule (1) to obtain a state S, i.e., $S' \rightsquigarrow_{\sigma} S$. But the variable IK in state S' gets instantiated $\sigma = \{IK \mapsto w \in \mathcal{I}, IK''\}$ in such a way that $w\theta' =_{E_{\mathcal{P}}} u_j\tau$. Since there is no extra variable in the rule, again $S >^{\theta'} s$.
  - *Case c)* The intruder fact appears in $S'\theta'$ but not the strand. This case is very simple, since θ' makes valid S' as a symbolic state of s, i.e., $S = S'$ and $S' >^{\theta'} s$.
  - *Case d)* The strand and the intruder fact do not in $S'\theta'$. This case is very simple, since θ' makes valid S' as a symbolic state of s, i.e., $S = S'$ and $S' >^{\theta'} s$.
- Given states s and s' such that $s \rightarrow s'$ using a rule in set (10), then there exist a substitution τ, variables SS' and IK', and a strand $[u_1^{\pm}, \ldots, u_n^{\pm}]$ in $\mathcal{P}$ such that $s' = \{SS'\tau \,\&\, \{(u_1\tau) \in \mathcal{I}, IK'\tau\} \,\&\, [(u_1\tau)^{-}]\}$. This is similar to the case above of a rule in set (9).

This concludes the proof. □

Proof of Lemma 2

Since rewriting is simply a special case of narrowing, the proof of this lemma is simpler than that of Lemma 1. We take into account that $S >^{\theta} s$ implies the strand and intruder facts used in the narrowing step $S \overset{\mu}{\leftsquigarrow} S'$ are present in s.

- If we use rule (1) in $S \overset{\mu}{\leftsquigarrow} S'$, then there are associated rules in the sets (9) and (10).
- If we use rule (2) in $S \overset{\mu}{\leftsquigarrow} S'$, then there are associated rules in the sets (6) and (8).
- If we use rule (3) in $S \overset{\mu}{\leftsquigarrow} S'$, then there are associated rules in the sets (5) and (7).
- And if we use a rule in set (4) in $S \overset{\mu}{\leftsquigarrow} S'$, then there are associated rules in the sets (5) and (7).

Note that substitution θ' is just a restriction of substitution θ, since each backwards narrowing step instantiates some variable or add new terms (possibly with new variables) but never removes any term or variable already present. This concludes the proof. □