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

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 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. 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 that is necessary for existing approaches to symbolic-state analysis of cryptographic protocols. Third, backwards search is not as useful for expressing liveness properties, used, for example, in fair exchange protocols. 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.
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 one given 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 $\Sigma = (S, \leq, \Sigma)$ with poset of sorts $(S, \leq)$. We also assume an $\Sigma$-sorted family $X = \{X_t\}_{t \in S}$ of disjoint variable sets with each $X_t$ countably infinite. $T_\Sigma(X)$ is the set of terms of sort $S$, and $T_\Sigma^{\leq}$ is the set of ground terms of sort $S$. We write $T_\Sigma(X)$ and $T_\Sigma^{\leq}$ for the corresponding order-sorted term algebras. For a term $t$, $\text{Var}(t)$ 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 $\epsilon$. Given $U \subseteq \Sigma \cup X$, $\text{Pos}_U(t)$ 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 $\text{Pos}(t)$, and the set of non-variable positions $\text{Pos}_X(t)$. The subterm of $t$ at position $p$ is $t|_p$ and $t|_p[u]$ is the term $t$ where $t|_p$ is replaced by $u$.

A substitution $\sigma \in \text{Subst}(\Sigma, X)$ is a sorted mapping from a finite subset of $X$ to $T_\Sigma(X)$. Substitutions are written as $\sigma = \{X_1 \mapsto t_1, \ldots, X_n \mapsto t_n\}$ where the domain of $\sigma$ is $\text{Dom}(\sigma) = \{X_1, \ldots, X_n\}$ and the set of variables introduced
by terms $t_1, \ldots, t_n$ is written $\text{Run}(\sigma)$. The identity substitution is $\text{id}$. Substitutions are homomorphically extended to $T_\Sigma(X)$. The application of a substitution $\sigma$ to a term $t$ is denoted by $t\sigma$. For simplicity, we assume that every substitution is idempotent, i.e., $\sigma$ satisfies $\text{Dom}(\sigma) \cap \text{Run}(\sigma) = \emptyset$. Substitution idempotency ensures $t = (t\sigma)\sigma$. The restriction of $\sigma$ to a set of variables $V$ is $\sigma|_V$. Composition of two substitutions $\sigma$ and $\alpha$ is denoted by $\sigma\alpha$.

A **Σ-equation** is an unoriented pair $t \leftrightarrow t'$, where $t, t' \in T_\Sigma(X)$, for some sort $s \in S$. Given $\Sigma$ and a set $E$ of $\Sigma$-equations, ordered-sorted equational logic induces a congruence relation $\equiv_E$ on terms $t, t' \in T_\Sigma(X)$ (see [18]). The $E$-equivalence class of a term $t$ is denoted by $\{t\}_E$ and $T_{\Sigma,E}(X)$ and $T_{\Sigma,E}^0$ denote the corresponding ordered-sorted term algebras modulo $E$. Throughout this paper we assume that $T_{\Sigma,X} \neq \emptyset$ for every sort $s$, because this affords a simpler deduction system. An **equational theory** $(\Sigma, E)$ is a pair with $\Sigma$ an ordered-sorted signature and $E$ a set of $\Sigma$-equations. The $E$-subsumption preorder $\equiv_E$ (or just $\equiv$ if $E$ is understood) holds between $t, t' \in T_\Sigma(X)$, denoted $t \equiv_E t'$ (meaning that $t$ is more general than $t'$ modulo $E$), if there is a substitution $\sigma$ such that $t\sigma = t'$; such a substitution $\sigma$ is said to be an $E$-match from $t'$ to $t$.

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

A **rewrite rule** is an ordered pair $l \rightarrow r$, where $l \not\in X$ and $l, r \in T_\Sigma(X)_1$ for some sort $s \in S$. An (unconditional) **order-sorted rewrite theory** is a triple $(\Sigma, E, R)$ with $\Sigma$ an ordered-signature, $E$ a set of $\Sigma$-equations, and $R$ a set of rewrite rules.

The rewriting relation on $T_\Sigma(X)$, written $t \rightarrow_R t'$ or $t \rightarrow_{p,R} t'$ holds between $t$ and $t'$ iff there exist $p \in \text{Pos}(t)$, $l \rightarrow r \in R$ and a substitution $\sigma$, such that $t\sigma = l\sigma$ and $t'\sigma = r\sigma$. The subterm $l\sigma$ is called a redex. The relation $\rightarrow_{R,E}$ on $T_\Sigma(X)$ is $t \rightarrow_{R,E} t'$ iff there exists a $t \rightarrow_{R,E} t'$ such that $t\sigma = l\sigma$ and $t'\sigma = r\sigma$. Note that $t\sigma = l\sigma$ and $t'\sigma = r\sigma$ are the same. The transitive ($\text{resp.}$ transitive and reflexive) closure of $\rightarrow_{R,E}$ is denoted $\rightarrow^*_E$ ($\text{resp.}$ $\rightarrow^*_E$).

The reducibility of the $\rightarrow_{R,E}$ 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 $\rightarrow_{R,E}$ on $T_\Sigma(X)$ is defined as: $t \rightarrow_{p,R,E} t'$ (or just $t \rightarrow_{R,E} t'$) iff there exist $p \in \text{Pos}(t)$, a rule $l \rightarrow r \in R$, and a substitution $\sigma$ such that $t\sigma = l\sigma$ and $t'\sigma = r\sigma$.

Let $l$ be a term and $W$ be a set of variables such that $\text{Var}(t) \subseteq W$, the $R$, $E$-narrowing relation on $T_\Sigma(X)$ is defined as $t \rightarrow_{p,R,E} t'$ ($t \rightarrow_{n,R,E}$ if $p$ is understood) if $R,E$ are only understood, and $\rightarrow_R$ if $R,E$ are understood. Let $\sigma$ be a substitution. If there is a non-variable $p$ such that $t\sigma = r\sigma$, then $l \rightarrow_{R,E} r$ properly renamed s.t. $(\var{t})_E \cup \var{r} \subseteq W = \emptyset$, and a unifier $\sigma \in \text{CSU}^W_E(\var{r}_E = 1)$ for $W \subseteq \text{Var}(t)$, such that $t' = (\var{t'})_E$. For convenience, in each narrowing step $t \rightarrow_{n} t'$ we only specify the part of $\sigma$ that binds variables of $t$. The transitive (resp. transitive and reflexive) closure of $\rightarrow_{n}$ is denoted by $\rightarrow^{+}_{n}$ (resp. $\rightarrow^{*}_{n}$). We may write $t \rightarrow^{+}_{n} t'$ if there are $u_1, \ldots, u_{k-1}$ and substitutions $\rho_1, \ldots, \rho_k$ such that $t \rightarrow_{p_1} u_1 \rightarrow_{p_{k-1}} u_{k-1} \rightarrow_{p_k} t'$, $k \geq 0$, 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 $T_{\Sigma, \mathcal{P}}$, where $\Sigma$ is the signature defining the sorts and function symbols (for the cryptographic functions and for all the state constructor symbols) and $\mathcal{P}$ is a set of equations specifying the algebraic properties of the cryptographic functions and the state constructors. Therefore, a state is an $\mathcal{P}$-equivalence class $[t] \in T_{\Sigma, \mathcal{P}}$, with $t$ a ground $\mathcal{P}$-term. However, we explore symbolic state patterns $\{(x_1, \ldots, x_n) \mid \text{Var}(X) \subseteq \mathcal{P}(X)\}$ on the free $(\Sigma, \mathcal{P})$-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., $t \in T_{\Sigma, \mathcal{P}}(X)\text{State}$) which has the form $\{S_1, \ldots, S_n \& \text{Var}(\{\mathcal{I}\})\}$ where $\&$ is an associative-commutative union operator with identity symbol $\perp$. Each element in the set is either a strand $S_i$ or the intruder knowledge $\{\mathcal{I}\}$ at that state.

The intruder knowledge $\{\mathcal{I}\}$ also belongs to the state and is represented as a set of facts using the comma as an associative-commutative union 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., $m \not\in \mathcal{I}$), where $m$ is a

As described in [12], & can also be treated as an associative-commutative-identity-impotent 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 identity-impotent, 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.

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 $\mathbf{[3]}$ specifies the sequence of messages sent and received by a principal executing the protocol and is represented as a sequence of messages $\mathcal{m}_1, \mathcal{m}_2, \ldots, \mathcal{m}_{n-1}, \mathcal{m}_n$ such that $\mathcal{m}_i$ (also written $\neg \mathcal{m}_i$) represents an input message, $\mathcal{m}^+_i$ (also written $\neg \mathcal{m}_i$) represents an output message, and each $\mathcal{m}_i$ is a term of sort $\text{Msg}$ (i.e., $\mathcal{m}_i \in T_{\Sigma_T, \text{E}_p}(X) \cup \text{Msg}$). For each positive message $\mathcal{m}^+_i$ in a strand, the variables occurring in message $\mathcal{m}_i$ must appear in all previous messages $\mathcal{m}_1, \ldots, \mathcal{m}_{i-1}$, except for variables denoting principal names (they are considered as initial knowledge available to all participants) and variables of sort $\text{Fresh}$. Variables of sort $\text{Fresh}$ are unique for each instance of a strand schema, 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 (for a strand for each operator an intruder is able to perform on terms). In Maude-NPA, strands evolve over time; the symbol $\mathbf{::}$ is used to divide past and future. That is, a strand $\mathcal{m}_1, \ldots, \mathcal{m}_{n+1}, \ldots, \mathcal{m}_n$ evolve over time; the symbol $\mathbf{::}$ is used to divide past and future. That is, a strand $\mathcal{m}_1, \ldots, \mathcal{m}_{n+1}, \ldots, \mathcal{m}_n$ is given by a strand $\mathcal{m}_1, \ldots, \mathcal{m}_{n+1}, \ldots, \mathcal{m}_n$ are the past messages, and messages $\mathcal{m}^+_1, \ldots, \mathcal{m}^+_n$ are the future messages ($m^+_i$ 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 $\mathcal{m}_1, \ldots, \mathcal{m}_n$ is a shorthand for $\mathbf{nil} \vdash \mathcal{m}_1, \ldots, \mathcal{m}_n, \mathbf{nil}$. 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 $\mathcal{m}^+ \notin \mathbf{\cdot}$.

Since $\text{Fresh}$ variables must be treated differently from other variables by Maude-NPA, we make them explicit by writing $\mathcal{m}_1^+, \ldots, \mathcal{m}_n^+, \mathbf{::} \mathcal{m}_1^+, \ldots, \mathcal{m}_n^+, \mathbf{::} \mathcal{m}_1^+, \ldots, \mathcal{m}_n^+, \mathbf{::}$. In this way, a state pattern $\mathcal{m}_1, \ldots, \mathcal{m}_n$ represents a set of ground terms.

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 a responder, Bob. We use the common notation $A \Rightarrow B : M$ 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 $M_1 \cdot M_2$. 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 $\cdot$, which is an associative-commutative symbol. Nonces are represented by $N_X$, denoting a nonce created by principal X. The expression $\text{sec}(A; X)$ denotes a secret term generated by principal A.

The protocol description is as follows.

1. $A \Rightarrow B : [A; B; g^X]$ Alice creates a new nonce $N_A$ and sends her name, Bob’s name, and $g^X$ to Bob.

2. $B \Rightarrow A : [B; A; g^X]$ Bob creates a new nonce $N_B$ and sends his name, Alice’s name, and $g^X$ to Alice.

3. $A \Rightarrow B : [\{\text{secret}\}; N_B; N_A]$ Alice computes $g^X$ and encrypts the secret data. The key $g^{N_B X A}$ is equal to $g^{N_B X A}$ using the algebraic property $X^YZ = X^Y = X^{YZ}$. Bob computes $g^{N_B X_A}$ 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.

In Section 3.2 we specify the rewrite rules governing the backtracking 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 $T_{\Sigma_T, \text{E}_p}$ is in general infinite, rather than exploring concrete protocol states $[I] \in T_{\Sigma_T, \text{E}_p}$, we explore symbolic state patterns $[f(x_1, \ldots, x_n)] \in T_{\Sigma_T, \text{E}_p}(X)$ on the free ($\Sigma_T, \text{E}_p$)-algebra over a set of variables X. In this way, a state pattern $[f(x_1, \ldots, x_n)]$ represents not a single concrete state but a possibly infinite set of such states, namely all the instances of the pattern $[f(x_1, \ldots, x_n)]$ 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.
Note that Maude-NPA does not display the initial knowledge of the intruder which is all the messages produced in the protocol execution. 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 a set of 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. The principal strands in the given protocol 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 $R_{BPP}$ of rewrite rules, so that the rewrite theory $(\Sigma_{B}, E_{B}, R_{BPP})$ characterizes the behavior of protocol $P$ modulo the equations $E_{B}$ for backwards execution. In this section we use $R_{BPP}$ to denote the rewrite rules associated to protocol $P$ for backwards execution, whereas in Section 1.2 we will use $R_{FP}$ to denote the new rewrite rules associated to protocol $P$ for forwards execution. The rules $R_{BPP}$ 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 forwards sense, are used in a backwards sense (by reversing the direction of the arrow) and describe the general state transitions:

$\text{SS} \& [L \mid M^+, L'] \& \{MeE, IK\}$

$\Rightarrow \text{SS} \& [L \mid M^+ \mid L'] \& \{MeE, IK\}$

(1)

$\text{SS} \& [L \mid M^+, L'] \& \{IK\}$

$\Rightarrow \text{SS} \& [L \mid M^+ \mid L'] \& \{IK\}$

(2)

$\text{SS} \& [L \mid M^+, L'] \& \{MeE, IK\}$

$\Rightarrow \text{SS} \& [L \mid M^+ \mid L'] \& \{MeE, IK\}$

(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 ($m\in E$ or $m\notin E$), 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

$\Rightarrow \text{SS} \& [L \mid M^+ \mid L'] \& \{MeE, IK\}$

(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 ($m\in E$ or $m\notin E$), 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

$\Rightarrow \text{SS} \& [L \mid M^+ \mid L'] \& \{MeE, IK\}$

(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 ($m\in E$ or $m\notin E$), 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

$\Rightarrow \text{SS} \& [L \mid M^+ \mid L'] \& \{MeE, IK\}$

(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 ($m\in E$ or $m\notin E$), and $SS$ denotes a set of strands.
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\#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 \# I into m \# I to identify the transition in which the fact MeI 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 the initial state. The execution of the protocol in the strand-based model requires much more than simply reversing the forwards semantics. The reader wards semantics requires much more than simply reversing the forwards semantics. The reader

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:

\{SS \& \{KeI, MeI, IK\}\}
\rightarrow \{SS \& \{-\{IK\}, \{-M\}, +\{e(K, M)\}\}\}
& \{KeI, MeI, e(K, M)\#I, IK\}\}

Second, the restriction mentioned in Footnote 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 \(r \rightarrow [+\{n(i, \{r\})\}]\) will be represented by a transition rule of the form:

\{SS \& \{IK\} \& \langle N \rangle\}
\rightarrow \{SS \& [+\{n(i, \{N\})\}] \& \{IK\} \& \langle N + 1 \rangle\}

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 (i), we write u^i_{exp}, 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 fresh variable, we write u^2_{exp} = exp(g, n(0, 1) * n(10, 11)), i.e., the substitution \{A \rightarrow 10, r \rightarrow 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 2, the initial state is just the empty state:

\(\emptyset \& \emptyset\) and \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:

\{[-\{a; b; Y\}, +\{b; a; \exp(g, n(b, \{r\})\}], -(\{\exp(Y, n(b, \{r\})), SR\}\}\}
& SS \& \{SR\#I, IK\}\}

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, eb, cb', r1, r2, r3, r4
are natural numbers but the actual value is irrelevant):

\[
\begin{align*}
&+ (ca; cb'; \exp (g, n(c, ra)),)
\quad - (ca; cb; \exp (g, n(c, r1))),
\quad + (e \exp (g, n(i, r2) \& n(ca, ra)), sec(ca, ra))),
\end{align*}
\]

The Dolev-Yao intruder strands are as follows:

\[
\begin{align*}
&+[\{ i(n(i, r1)) \}]
\quad - [(ca; cb; \exp (g, n(c, ra))), \{ + (ca; cb; \exp (g, n(c, ra))) \}]
\quad - [(ca; cb; \exp (g, n(c, ra))), \{ + (ca; cb; \exp (g, n(c, ra))) \}]
\end{align*}
\]

It is also possible to specify authentication attacks in
Mauve for the forwards semantics by using again Mauve’s
search command to search for an attack, but making it
conditional to the never patterns not having been encoun-
tered. 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 S1, . . . , Sn, we can search for an attack avoiding such patterns by giving to
Mauve the conditional search command:

\[
\begin{align*}
\text{search init} \rightarrow^* S \text{ such that }
p_{S_0}(S) = \text{false} \land \ldots \land p_{S_n}(S) = \text{false}
\end{align*}
\]

where each predicate pS holds for a concrete state s iff s
is an instance of the pattern S0. When a never pattern S0
shares variables v0, . . . , vk, with S, the predicate pS0 is
extended in the form pS0,S1, . . . , Sni,v0, . . . , vi; 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 pat-
terns Alice1 and Alice2 (not sharing variables with S) of
the Diffie-Hellman protocol in Example 3,

\[
S = \{ \psi \Rightarrow [- (a; b; c), + (b; a; e \exp (g, n(b, \psi))), - (e \exp (Y, n(b, \psi)), SR)] \mid n!f \}
\]

& SS & IK

Alice1 = r, r' \Rightarrow [- (a; b; c), e \exp (g, n(a, r))), + (e \exp (X, n(a, r)), sec(a, r'))]

Alice2 = r, r' \Rightarrow [+ (a; b; c), e \exp (g, n(a, r))), + (c; a; X),

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

\[
\begin{align*}
\text{search init} \rightarrow^* S \text{ such that }
p_{Alice1}(S) = \text{false} \land p_{Alice2}(S) = \text{false}
\end{align*}
\]

where the predicates pAlice1 and pAlice2 check whether any
strand of the concrete state S is an instance of the strands
Alice1 and Alice2, respectively.

**4.2 Forwards Operational Semantics**

In a forward reachability analysis, we define state changes
by means of a set \{ \text{rewrite rules} \} characterizing the behavior of protocol \mathcal{P}
modulo the equations \mathcal{E}_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 knowl-
gedge. 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 \text{m}+ to an existing strand and either adding or not adding to
the intruder’s knowledge, (ii) adding a negative term \text{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 \text{m}+ that either adds or not to the intruder’s
knowledge. For example, the intruder encryption capability
\text{[–K(–M), + (e(K, M))]} produces the rewrite rules:

\[
\begin{align*}
&\{ SS & \& \{ K \in t, IK \} & \& \langle \gamma \rangle \}
\rightarrow \{ SS & \& \{ –K(–M), + (e(K, M)) \} & \& \langle \gamma \rangle \}
\end{align*}
\]

where the right-hand side becomes a rewrite rule of the
form \{ \text{S & } \text{R} \} \rightarrow \{ \text{S & } \text{R} \}.

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

\[
\begin{align}
&\{ \forall [u_1^+, \ldots, u_{j-1}^+, u_j, u_{j+1}^+, \ldots, u_k] \in \mathcal{P} & \& j > 1 : \\
&\{ SS & \& IK & \& \{ u_1^+, \ldots, u_k^+, \langle \gamma \rangle \} \}
&\{ SS & \& \{ v_1^+, \ldots, v_k^+, \langle \gamma \rangle \}
&\{ SS & \& IK & \& \{ u_1^+, \ldots, u_k^+, \langle \gamma \rangle \} \}
&\{ SS & \& \{ v_1^+, \ldots, v_k^+, \langle \gamma \rangle \}
&\{ SS & \& IK & \& \{ u_1^+, \ldots, u_k^+, \langle \gamma \rangle \} \}
\end{align}
\]

\[
\begin{align}
&\{ \forall [u_1^+, \ldots, u_{j-1}^+, u_j, u_{j+1}^+, \ldots, u_k] \in \mathcal{P} & \& j > 1 : \\
&\{ SS & \& \{ u_1^+, \ldots, u_k^+, \langle \gamma \rangle \} \}
&\{ SS & \& IK & \& \{ u_1^+, \ldots, u_k^+, \langle \gamma \rangle \} \}
\end{align}
\]

\[
\begin{align}
&\{ \forall [u_1^+, \ldots, u_k^+] \in \mathcal{P} : \\
&\{ SS & \& IK & \& \langle \gamma \rangle \}
&\{ SS & \& \{ u_1^+, \ldots, u_k^+, \langle \gamma \rangle \} \}
\end{align}
\]

\[
\begin{align}
&\{ \forall [u_1^+, \ldots, u_k^+] \in \mathcal{P} : \\
&\{ SS & \& IK & \& \langle \gamma \rangle \}
&\{ SS & \& \{ u_1^+, \ldots, u_k^+, \langle \gamma \rangle \} \}
\end{align}
\]
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 but the intruder’s knowledge is not increased \(9\).

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

\[
\{ \forall [w_1^+, \ldots, w_j^+, \ldots, w_k^+] \in \mathcal{P} & \land j > 1 \ :
\{ SS \& (w_i \in \mathcal{I}, IK) \& [w_1^+, \ldots, w_j^+, \ldots, w_k^+] \& \langle N \rangle \\
\rightarrow \{ SS \& (w_i \in \mathcal{I}, IK) \& [w_1^+, \ldots, w_j^+, \ldots, w_k^+] \& \langle N \rangle \} \}
\] (9)

\[
\{ \forall [u_1^+, \ldots, u_j^+, \ldots, u_k^+] \in \mathcal{P} : \\
\{ SS \& (w_i \in \mathcal{I}, IK) \& \langle N \rangle \\
\rightarrow \{ SS \& (w_i \in \mathcal{I}, IK) \& \langle N \rangle \} \}
\] (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 \(\Sigma_F\) and equational theory \(\mathcal{E}_\mathcal{P}\). We define the forward rewrite theory characterizing \(\mathcal{P}\) to be \(\langle \Sigma_F, \mathcal{E}_\mathcal{P}, R_{\mathcal{F}} \rangle\) where \(R_{\mathcal{F}} = \{5\} \cup \{6\} \cup \{7\} \cup \{8\} \cup \{9\} \cup \{10\}\).

**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

\[
:: r, r' :: \{+\{A; B; \exp(g, n(A, r)) \}, -(B; A; X), +1(\exp(X, n(A, r)), \sec(A, r'))\}
\]

and the rewrite rules associated to it are as follows:

\[
\{ SS \& [1K] \& \langle N \rangle \rightarrow \{ SS \& [\{N; N + 1; \exp(g, n(N, N + 2))\}] \& \langle N + 3 \rangle \}
\]

\[
\{ SS \& \{A; B; \exp(g, n(A, R))\} \rightarrow \{ (B; A; X) \in \mathcal{I}, IK \} \& \langle N \rangle \}
\]

\[
\{ SS \& \{A; B; \exp(g, n(A, R))\}, -(B; A; X) \rightarrow \{ (B; A; X) \in \mathcal{I}, IK \} \& \langle N \rangle \}
\]

\[
\{ SS \& \{A; B; \exp(g, n(A, R))\}, -(B; A; X), [1K] \& \langle N \rangle \rightarrow \{ SS \& \{A; B; \exp(g, n(A, R))\}, -(B; A; X), +1(\exp(X, n(A, R)), \sec(A, N)) \} \& e(\exp(X, n(A, R)), \sec(A, N)) \in \mathcal{I}, IK \} \& \langle N + 1 \rangle \}
\]

When the intruder impersonates Bob, the first rule is:

\[
\{ SS \& [IK] \& \langle N \rangle \rightarrow \{ SS \& [\{N; i; \exp(g, n(N, N + 1))\}] \& \langle N + 2 \rangle \}
\]

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

\(\text{Note that the use of the global counter for new principal names in previous rules has to take into account when one of those principals is indeed the intruder; see Example 5\# for the case in which the intruder impersonates Bob.}\)

### 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:

\[
S = \{ r_1, \ldots, r_m : [u_{i1}^+, \ldots, u_{i-1}^+] \& [u_{i1}^+, \ldots, u_{in}^+] \}
\]

\[
\ldots
\]

\[
\{ r_k, \ldots, r_m : [u_{i1}^+, \ldots, u_{i-1}^+] \& [u_{i1}^+, \ldots, u_{in}^+] \} \& SS
\]

\[
\{ w_i \in \mathcal{I}, \ldots, w_n \in \mathcal{I}, w_{1\mathcal{E}_l}^\mathcal{I}, \ldots, w_{m\mathcal{E}_l}^\mathcal{I}, IK \}
\]

where for each \(1 \leq j \leq k\), there exists a strand \([m_{1j}^+, \ldots, m_{i-1j}^+], m_{i0}^+, \ldots, m_{in}^+ \] in \(\mathcal{P}\) and a substitution \(\rho_j : X \rightarrow \mathcal{T}_\mathcal{E}[X]\) such that \(m_1, \rho_j = \mathcal{E}_\mathcal{P}, u_{i1}, \ldots, m_n, \rho_j = \mathcal{E}_\mathcal{P}, u_{in}\). \(SS\) 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:

\[
s = \{ [u_{i1}^+, \ldots, u_{i-1}^+] \& \ldots \& [u_{i1}^+, \ldots, u_{in}^+] \}
\]

\[
\{ w_i \in \mathcal{I}, \ldots, w_n \in \mathcal{I} \} \& \langle N \rangle
\]

where for each \(1 \leq j \leq k\), there exists a strand \([m_{1j}^+, \ldots, m_{i-1j}^+], m_{i0}^+, \ldots, m_{in}^+ \] in \(\mathcal{P}\) and a substitution \(\rho_j : X \rightarrow \mathcal{T}_\mathcal{E}[X]\) such that \(m_1, \rho_j = \mathcal{E}_\mathcal{P}, u_{i1}, \ldots, m_n, \rho_j = \mathcal{E}_\mathcal{P}, u_{in}\).

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 \(\theta : \{ \text{Var}(S) \rightarrow \{ SS \} \} \rightarrow \mathcal{T}_\mathcal{E}, \) written \(S \rightarrow^* s\) if

- for each strand \(r_1, \ldots, r_m : [w_{j1}^+, \ldots, w_{j-1}^+] \& [w_{j1}^+, \ldots, w_{jn}^+]\) in \(S\), there exists a strand \([w_{j1}^+, \ldots, w_{jn}^+]\) in \(s\) such that \(\forall 1 \leq j \leq i = 1, v_j = \mathcal{E}_\mathcal{P}, u_{j1}.\)

- for each positive intruder fact \(w_i \in \mathcal{I}\) in \(S\), there exists a positive intruder fact \(w_i \in \mathcal{I}\) in \(s\) such that \(w_i = \mathcal{E}_\mathcal{P}, w_i.\)

- and for each negative intruder fact \(w_i \notin \mathcal{I}\) in \(S\), there is no positive intruder fact \(w_i \notin \mathcal{I}\) in \(s\) such that \(w_i = \mathcal{E}_\mathcal{P}, w_i.\)

Let us now prove that narrowing with the backwards rewrite theory \(\langle \Sigma_F, \mathcal{E}_F, R_{\mathcal{F}} \rangle\) is complete with respect to rewriting with the forwards rewrite theory \(\langle \Sigma_F, \mathcal{E}_F, R_{\mathcal{F}} \rangle\). 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 \( P \), two ground states \( s, s' \), a \( P \)-symbolic state \( S \), a substitution \( \theta \) s.t. \( s \rightarrow s' \) and \( S \rightarrow^\theta s' \), then there exist a \( P \)-symbolic state \( S' \) and a substitution \( \theta' \) s.t. \( S \rightarrow^\theta s \) and either \( S' \sim \theta S \) 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 \( P \), two ground states \( s, s_0 \), a symbolic \( P \)-state \( S \), a substitution \( \theta \), s.t. (i) \( s_0 \) is an initial state, (ii) \( s_0 \rightarrow^* s \), and (iii) \( S \rightarrow^\theta s \) then there exist a symbolic initial \( P \)-state \( S_0 \), two substitutions \( \mu \) and \( \theta' \), and \( k \leq n \), s.t. \( S_0 \sim \theta S \), and \( S_0 \rightarrow^\theta s_0 \).

In the following we prove that the backwards rewrite theory \((\Sigma_p, E_p, R^1_p)\) is sound with respect to the forward rewrite theory \((\Sigma_p, E_p, R^F_p)\). 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 \( S_0 \rightarrow^\theta s_0 \), there is a reachable concrete attack states such that \( s_0 \rightarrow^* S \). 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 \( P \), two symbolic \( P \)-states \( S, S' \), a ground state \( s \) and a substitution \( \theta \), s.t. \( S \sim \theta S' \) and \( S \rightarrow^\theta s \), then there exist a state \( s' \) and a substitution \( \theta' \) such that \( s \rightarrow s' \) and \( S' \rightarrow^\theta s' \).

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 \( P \), two symbolic \( P \)-states \( S_0, S' \), an initial ground state \( s_0 \) and a substitution \( \theta \), s.t. (i) \( s_0 \) is a symbolic initial state, and (ii) \( S_0 \rightarrow^\theta s_0 \), and (iii) \( S_0 \rightarrow^\theta s_0 \) then there exist a ground state \( s' \) and a substitution \( \theta' \), s.t. (i) \( s_0 \rightarrow s' \), and (ii) \( S' \rightarrow^\theta s' \).

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 [11], 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

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Length Forwards</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSPK-sec</td>
<td>9</td>
</tr>
<tr>
<td>NSPK-auth</td>
<td>9</td>
</tr>
<tr>
<td>NSL-XOR</td>
<td>13</td>
</tr>
<tr>
<td>Denning-Sacco</td>
<td>11</td>
</tr>
<tr>
<td>Diffie-Hellman</td>
<td>22</td>
</tr>
</tbody>
</table>

Table 2: States generated in each rewrite step

<table>
<thead>
<tr>
<th>Protocol</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSPK-sec</td>
<td>6</td>
<td>20</td>
<td>116</td>
<td>604</td>
<td>3026</td>
</tr>
<tr>
<td>NSPK-auth</td>
<td>6</td>
<td>20</td>
<td>116</td>
<td>604</td>
<td>3026</td>
</tr>
<tr>
<td>NSL-XOR</td>
<td>7</td>
<td>21</td>
<td>72</td>
<td>218</td>
<td>594</td>
</tr>
<tr>
<td>Denning-Sacco</td>
<td>7</td>
<td>28</td>
<td>132</td>
<td>596</td>
<td>2624</td>
</tr>
<tr>
<td>Diffie-Hellman</td>
<td>7</td>
<td>22</td>
<td>65</td>
<td>162</td>
<td>354</td>
</tr>
</tbody>
</table>

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. However, much work needs to be done, specially with respect to the efficiency of the analysis.
8. REFERENCES


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 element to an existing strand (rules (3), (5), and (7)), adding a positive fact to the intruder knowledge (rules (8), (9), and (10)), or repeating a positive intruder fact that is already in $S$ (rules (2) and (4)). This allows us to identify six cases for the grounding substitution $\theta'$ of $S$ into $s'$, depending upon whether the grounding substitution of $S$ under $\theta$ contains the relevant strands and positive intruder facts.

a) There is a strand $[u_1^+, \ldots, u_{i-1}^+, u_i^±, \ldots, u_n^±]$ in $P$, $n \geq 1$, $1 \leq i \leq n$, and a substitution $\rho$ such that $[u_1^±, \ldots, u_{i-1}^±, u_i^±] \rho$ is a strand in $S$, $[u_i^±, \ldots, u_n^±] \rho$ is a strand in $S'\theta'$, and $u_i \epsilon \mathcal{E}L$ appears in the intruder knowledge of $S'\theta'$. This is valid for rules in sets (3), (5), (7), and (9). If $i > 1$, then we also know that $[u_1^±, \ldots, u_{i-1}^±] \rho$ is a strand in $s$.

b) There is a strand $[u_1^±, \ldots, u_{i-1}^±, u_i^±, \ldots, u_n^±]$ in $P$, $n \geq 1$, $1 \leq i \leq n$, and a substitution $\rho$ such that $[u_1^±, \ldots, u_{i-1}^±, u_i^±] \rho$ is a strand in $S$, $[u_i^±, \ldots, u_n^±] \rho$ is a strand in $S'\theta'$, and $u_i \epsilon \mathcal{E}L$ does not appear in the intruder knowledge of $S'\theta'$. This is valid for rules in sets (3), (5), (7), and (10). If $i > 1$, then we also know that $[u_1^±, \ldots, u_{i-1}^±, u_i^±] \rho$ is a strand in $s$.

c) There is a strand $[u_1^±, \ldots, u_{i-1}^±, u_i^±, \ldots, u_n^±]$ in $P$, $n \geq 1$, $1 \leq i \leq n$, and a substitution $\rho$ such that $[u_1^±, \ldots, u_{i-1}^±, u_i^±] \rho$ is a strand in $S$, $u_i \epsilon \mathcal{E}L$ appears in the intruder knowledge of $S'\theta'$, but $[u_i^±, \ldots, u_n^±] \rho$ is not a strand in $S'\theta'$. This is valid for rules in sets (3), (7), (9), and (10). If $i > 1$, then we also know that $[u_1^±, \ldots, u_{i-1}^±] \rho$ is a strand in $s$.

d) There is a strand $[u_1^±, \ldots, u_{i-1}^±, u_i^±, \ldots, u_n^±]$ in $P$, $n \geq 1$, $1 \leq i \leq n$, and a substitution $\rho$ such that $[u_1^±, \ldots, u_{i-1}^±, u_i^±] \rho$ is a strand in $S$ but $u_i \epsilon \mathcal{E}L$ does not appear in the intruder knowledge of $S'\theta'$ and $[u_i^±, \ldots, u_n^±] \rho$ is not a strand in $S'\theta'$. This is valid for rules in sets (3), (5), (7), and (10). If $i > 1$, then we also know that $[u_1^±, \ldots, u_{i-1}^±] \rho$ is a strand in $s$.

e) There is a strand $[u_1^±, \ldots, u_{i-1}^±, u_i^±, \ldots, u_n^±]$ in $P$, $n \geq 1$, $1 \leq i \leq n$, and a substitution $\rho$ such that $[u_1^±, \ldots, u_{i-1}^±, u_i^±] \rho$ is a strand in $S$ and $[u_i^±, \ldots, u_n^±] \rho$ is a strand in $S'\theta'$. This is valid for rules in sets (2) and (6). If $i > 1$, then we also know that $[u_1^±, \ldots, u_{i-1}^±] \rho$ is a strand in $s$.

f) There is a strand $[u_1^±, \ldots, u_{i-1}^±, u_i^±, \ldots, u_n^±]$ in $P$, $n \geq 1$, $1 \leq i \leq n$, and a substitution $\rho$ such that $[u_1^±, \ldots, u_{i-1}^±, u_i^±] \rho$ is a strand in $S$ but $[u_i^±, \ldots, u_n^±] \rho$ is not a strand in $S'\theta'$. This is valid for rules in sets (3) and (5). If $i > 1$, then we also know that $[u_1^±, \ldots, u_{i-1}^±] \rho$ 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 (6), then there exist a substitution $\tau$, variables $SS'$ and $IK'$, and a strand $[u_1^±, \ldots, u_{i-1}^±, u_i^±, \ldots, u_n^±]$ in $P$ such that $s = \{SS'\tau \& \{IK'\tau\} \& \{(u_1^±, \ldots, u_{i-1}^±, u_i^±, \ldots, u_n^±)\} \}$. 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 $\mathfrak{S}$, then there exist a substitution $\tau$, variables $SS'$ and $IK'$, and a strand $[u_1^1, \ldots, u_n^1]$ in $\mathfrak{P}$ such that $s' = \{SS\tau \& \{IK'\tau\} \& \{(u_1^1)^\tau, \ldots, (u_n^1)^\tau\}\}$. This is similar to the case above of a rule in set $\mathfrak{S}$. Given states $s$ and $s'$ such that $s \rightarrow s'$ using a rule in set $\mathfrak{S}$, then there exist a substitution $\tau$, variables $SS'$ and $IK'$, and a strand $[u_1^2, \ldots, u_n^2]$ in $\mathfrak{P}$ such that $s = \{SS\tau \& \{IK'\tau\} \& \{(u_1^2)^\tau, \ldots, (u_n^2)^\tau\}\}$, and $s' = \{SS\tau \& \{IK'\tau\} \& \{(u_1^2)^\tau, \ldots, (u_n^2)^\tau\}\}. Since there exists a substitution $\theta$ such that $S' >_\theta s$, we consider the four applicable cases for substitution $\theta$:

- **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., $S' \sim_\tau S$. Since there is no extra variable in the rule, we have that the same substitution $\theta'$ 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., $S' \sim_\tau S$. But the variable $IK$ in state $S'$ gets instantiated $\sigma = \{IK \rightarrow u_1^1 \in I, IK^\theta\}$ in such a way that $u\theta = u_1^1 \tau$. 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 $\theta'$ 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 $\theta'$ 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 $\mathfrak{S}$, then there exist a substitution $\tau$, variables $SS'$ and $IK'$, and a strand $[u_1^3, \ldots, u_n^3]$ in $\mathfrak{P}$ such that $s' = \{SS\tau \& \{(u_1^3)^\tau, \ldots, (u_n^3)^\tau\}\}. Since there exists a substitution $\theta$ such that $S' >_\theta s$, we consider the four applicable cases for substitution $\theta$:

- **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., $S' \sim_\tau S$. Since there is no extra variable in the rule, we have that the same substitution $\theta'$ 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., $S' \sim_\tau S$. But the variable $IK$ in state $S'$ gets instantiated $\sigma = \{IK \rightarrow u_1^1 \in I, IK^\theta\}$ in such a way that $u\theta = u_1^1 \tau$. 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 $\theta'$ 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 $\theta'$ makes valid $S'$ as a symbolic state of $s$, i.e., $S = S'$ and $S' >_\theta S$.

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 $S \leadsto S'$ are present in $s$.

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

Note that substitution $\theta'$ is just a restriction of substitution $\theta$, 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. □