Protocol Analysis Modulo Combination of Theories: A Case Study in Maude-NPA

Ralf Sasse[1], Santiago Escobar[2], Catherine Meadows[3], and José Meseguer[1]

[1] University of Illinois at Urbana-Champaign, USA.
{rsasse,meseguer}@illinois.edu
[2] DSIC-ELP, Universidad Politécnica de Valencia, Spain. sescobar@dsic.upv.es
[3] Naval Research Laboratory, Washington DC, USA
catherine.meadows@nrl.navy.mil

Abstract. There is a growing interest in formal methods and tools to analyze cryptographic protocols modulo algebraic properties of their underlying cryptographic functions. It is well-known that an intruder who uses algebraic equivalences of such functions can mount attacks that would be impossible if the cryptographic functions did not satisfy such equivalences. In practice, however, protocols use a collection of well-known functions, whose algebraic properties can naturally be grouped together as a union of theories E_1 ... E_n. Reasoning symbolically modulo the algebraic properties E_1 ... E_n requires performing (E_1 ... E_n)-unification. However, even if a unification algorithm for each individual E_i is available, this requires combining the existing algorithms by methods that are highly non-deterministic and have high computational cost. In this work we present an alternative method to obtain unification algorithms for combined theories based on variant narrowing. Although variant narrowing is less efficient at the level of a single theory E_i, it does not use any costly combination method. Furthermore, it does not require that each E_i has a dedicated unification algorithm in a tool implementation. We illustrate the use of this method in the Maude-NPA tool by means of a well-known protocol requiring the combination of three distinct equational theories.

Keywords: Cryptographic protocol verification, equational unification, variants, exclusive or, narrowing

1 Introduction

In recent years there has been growing interest in the formal analysis of protocols in which the crypto-algorithms satisfy different algebraic properties [10,13,29,16]. Applications such as electronic voting, digital cash, anonymous

R. Sasse and J. Meseguer have been partially supported by NSF Grants CNS-0716638, CNS-0831064 and CNS-0904749. S. Escobar has been partially supported by the EU (FEDER) and the Spanish MEC/MICINN under grant TIN 2007-68093-C02-02. C. Meadows has been partially supported by NSF Grant CNS-0904749.

communication, and even key distribution, all can profit from the use of such cryptosystems. Thus, a number of tools and algorithms have been developed that can analyze protocols that make use of these specialized cryptosystems [29,28,6,2,14].

Less attention has been paid to combinations of algebraic properties. However, protocols often make use of more than one type of cryptosystem. For example, the Internet Key Exchange protocol [23] makes use of Diffie-Hellman exponentiation (for exchange of master keys), public and private key cryptography (for authentication of master keys), shared key cryptography (for exchange of session keys), and exclusive-or (used in the generation of master keys). All of these functions satisfy different equational theories. Thus it is important to understand the behavior of algebraic properties in concert as well as separately. This is especially the case for protocol analysis systems based on unification, where the problem of combining unification algorithms [3,35] for different theories is known to be highly non-deterministic and complex, even when efficient unification algorithms exist for the individual theories, and even when the theories are disjoint (that is, share no symbols in common).

The Maude-NPA protocol analysis tool, which relies on unification to perform backwards reachability analysis from insecure states, makes use of two different techniques to handle the combination problem. One is to use a general-purpose approach to unification called variant narrowing [20], which, although not as efficient as special purpose unification algorithms, can be applied to a broad class of theories that satisfy a condition known as the finite variant property [12]. A second technique applicable to special purpose algorithms, or theories that do not satisfy the finite variant property, uses a more general framework for combining unification algorithms.

One advantage of using variant narrowing is that there are well-known methods and tools for checking that a combination of theories has the finite variant property, including checking its local confluence and termination, and also its satisfaction of the finite variant property itself [17]. Furthermore, under appropriate assumptions some of these checks can be made modularly (see, e.g., [33] for a survey of modular confluence and termination proof methods). This makes variant narrowing easily applicable for unification combination and very suitable for experimentation with different theories. Later on, when the theory is better understood, it may be worth the effort to invest the time to apply the framework to integrate more efficient special purpose algorithms.

In this paper we describe a case study involving the use of variant narrowing to apply Maude-NPA to the analysis of a protocol that involves three theories: (i) an associative-commutative theory satisfied by symbols used in state construction, (ii) a cancellation theory for public key encryption and decryption, and (iii) the equational theory of the exclusive-or operator. This theory combination is illustrated in the analysis of a version of the Needham-Schroeder-Lowe protocol [28], denoted NSL$_{\oplus}$, in which one of the concatenation operators is replaced by an exclusive-or [8].

The rest of this paper is organized as follows. In Section 2 we give some necessary background. In Section 3 we give an overview of Maude-NPA. In Sections 4 and 5 we describe variant narrowing and how it is used in Maude-NPA. In Section 6 we describe our use of Maude-NPA on the NSL$_\oplus$ protocol. In Section 7 we discuss related work, and Section 8 concludes the paper.

2 Background on Term Rewriting

We follow the classical notation and terminology from [36] for term rewriting and from [30,31] for rewriting logic and order-sorted notions. We assume an order-sorted signature Σ with a finite poset of sorts $(S, \leq)$ (such that each connected component of $(S, \leq)$ has a top sort) and a finite number of function symbols. We assume an S-sorted family $X = \{X_s\}_{s \in S}$ of disjoint variable sets with each X_s countably infinite. $T_\Sigma(X)_s$ denotes the set of terms of sort s, and $T_{\Sigma,s}$ the set of ground terms of sort s. We write $T_\Sigma(X)$ and T_Σ for the corresponding term algebras. We write $Var(t)$ for the set of variables present in a term t. 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 result of replacing $t|_p$ by u in t. A substitution σ is a sort-preserving mapping from a finite subset of X to $T_\Sigma(X)$.

A Σ-equation is an unoriented pair $t = t'$, where $t \in T_\Sigma(X)_s$, $t' \in T_\Sigma(X)_{s'}$, and s and s' are sorts in the same connected component of the poset $(S, \leq)$. For a set E of Σ-equations, an E-unifier for a Σ-equation $t = t'$ is a substitution σ s.t. $\sigma(t) =_E \sigma(t')$. A complete set of E-unifiers of an equation $t = t'$ is written $CSU_E(t = t')$. We say that $CSU_E(t = t')$ is finitary if it contains a finite number of E-unifiers. A rewrite rule is an oriented pair $l \to r$, where $l \notin X$ and $l, r \in T_\Sigma(X)_s$ for some sort $s \in S$. An (unconditional) order-sorted rewrite theory is a triple $\mathcal{R} = (\Sigma, E, R)$ with Σ an order-sorted signature, E a set of Σ-equations, and R a set of rewrite rules. The rewriting relation $\to_{R,E}$ on $T_\Sigma(X)$ is $t \xrightarrow{p}_{R,E} t'$ (or $\to_{R,E}$) if $p \in Pos_\Sigma(t)$, $l \to r \in R$, $t|_p =_E \sigma(l)$, and $t' = t[\sigma(r)]_p$ for some σ. Assuming that E has a finitary and complete unification algorithm, the narrowing relation modulo on $T_\Sigma(X)$ is $t \leadsto^{p}_{\sigma,R,E} t'$ (or $\leadsto_{\sigma,R,E}$, $\leadsto_{R,E}$) if $p \in Pos_\Sigma(t)$, $l \to r \in R$, $\sigma \in CSU_E(t|_p = l)$, and $t' = \sigma(t[r]_p)$.

We say that the relation $\to_{R,E}$ is terminating if there is no infinite sequence $t_1 \to_{R,E} t_2 \to_{R,E} \cdots t_n \to_{R,E} t_{n+1} \cdots$. We say that the relation $\to_{R,E}$ is confluent if whenever $t \to^*_{R,E} t'$ and $t \to^*_{R,E} t''$, there exists a term t''' such that $t' \to^*_{R,E} t'''$ and $t'' \to^*_{R,E} t'''$. An order-sorted rewrite theory (Σ, E, R) is confluent (resp. terminating) if the relation $\to_{R,E}$ is confluent (resp. terminating). In a confluent, terminating, order-sorted rewrite theory, for each term $t \in T_\Sigma(X)$, there is a unique (up to E-equivalence) R,E-irreducible term t' obtained from t by rewriting to canonical form, which is denoted by $t \to^!_{R,E} t'$ or $t\downarrow_{R,E}$ (when t' is not relevant). The relation $\to_{R,E}$ is E-coherent [24] if $\forall t_1, t_2, t_3$ we have $t_1 \to_{R,E} t_2$ and $t_1 =_E t_3$ implies $\exists t_4, t_5$ such that $t_2 \to^*_{R,E} t_4$, $t_3 \to^+_{R,E} t_5$, and $t_4 =_E t_5$.

3 Protocol Specification and Analysis in Maude-NPA

Given a protocol P, we first explain how its states are modeled algebraically. The key idea is to model such states as elements of an initial algebra T_{Σ_P/E_P}, where Σ_P is the signature defining the sorts and function symbols for the cryptographic functions and for all the state constructor symbols and E_P is a set of equations specifying the algebraic properties of the cryptographic functions and the state constructors. Therefore, a state is an E_P-equivalence class $[t] \in T_{\Sigma_P/E_P}$ with t a ground Σ_P-term. However, since the number of states T_{Σ_P/E_P} is in general infinite, rather than exploring concrete protocol states $[t] \in T_{\Sigma_P/E_P}$ we explore symbolic state patterns $[t(x_1,\ldots,x_n)] \in T_{\Sigma_P/E_P}(X)$ on the free (Σ_P,E_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.

Let us introduce a motivating example that we will use to illustrate our approach based on exclusive–or. We use an exclusive–or version borrowed from [8] of the Needham-Schroeder-Lowe protocol [28] which we denote NSL$_\oplus$. In our analysis we use the protocol based on public key encryption, i.e., operators pk and sk satisfying the equations $pk(P,sk(P,M)) = M$ and $sk(P,pk(P,M)) = M$ and the messages are put together using concatenation and exclusive–or. Note that we use a representation of public-key encryption in which only principal P can compute $sk(P,X)$ and everyone can compute $pk(P,X)$. For exclusive–or we have the associativity and commutativity (AC) axioms for $\oplus$, plus the equations[4] $X \oplus 0 = X$, $X \oplus X = 0$, $X \oplus X \oplus Y = Y$.

1. $A \to B : pk(B, N_A; A)$
   A sends to B, encrypted under B's public key, a communication request containing a nonce N_A that has been generated by A, concatenated with its name.
2. $B \to A : pk(A, N_A; B \oplus N_B)$
   B answers with a message encrypted under A's public key, containing the nonce of A, concatenated with the exclusive–or combination of a new nonce created by B and its name.
3. $A \to B : pk(B, N_B)$
   A responds with B's nonce encrypted under B's public key.

A and B agree that they both know N_A and N_B and no one else does.

In the Maude-NPA [15,16], a state in the protocol execution is a term t of sort state, $t \in T_{\Sigma_P/E_P}(X)_{state}$. A state is a multiset built by an associative and commutative union operator $\_\&\_$. Each element in the multiset can be a strand or the intruder knowledge at that state (intruder knowledge is wrapped by $\{\_\}$). A strand [21] represents the sequence of messages sent and received by a principal executing the protocol and is indicated by a sequence of messages $[msg_1^-, msg_2^+, msg_3^-, \ldots, msg_{k-1}^-, msg_k^+]$ where each msg_i is a

[4] The third equation follows from the first two. It is needed for coherence modulo AC.

term of sort Msg (i.e., msg_i $T_{P}(X)_{Msg}$), msg^- represents an input message, and msg^+ represents an output message. In Maude-NPA, strands evolve over time and thus we use the symbol | to divide past and future in a strand, i.e., $[msg_1^{\pm}, \ldots, msg_{j-1}^{\pm} \mid msg_j^{\pm}, msg_{j+1}^{\pm}, \ldots, msg_k^{\pm}]$ where $msg_1^{\pm}, \ldots, msg_{j-1}^{\pm}$ are the past messages, and $msg_j^{\pm}, msg_{j+1}^{\pm}, \ldots, msg_k^{\pm}$ are the future messages ($msg_j^{\pm}$ is the immediate future message). The intruder knowledge is represented as a multiset of facts unioned together with an associative and commutativity union operator _,_. There are two kinds of intruder facts: positive knowledge facts (the intruder knows m, i.e., m I), and negative knowledge facts (the intruder does not yet know m but will know it in a future state, i.e., m /I), where m is a message expression. Facts of the form m /I make sense in a backwards analysis, since one state can have m I and a prior state can have m /I.

The strands associated to the three protocol steps above are given next. There are two strands, one for each principal in the protocol. Note that the first message passing A B : $pk(B, N_A; A)$ is represented by a message in Alice's strand sending $(pk(B, n(A, r); A))^+$, together with another message in Bob's strand that receives $(pk(B, N; A))^-$. When a principal cannot observe the contents of a concrete part of a received message (e.g., because a key is necessary to look inside), we use a generic variable for such part of the message in the strand (as with variable N of sort Nonce above, and similarly for X, Y below). We encourage the reader to compare the protocol in strand notation to the presentation of the protocol above. We also omit the initial and final nil in strands, which are needed in the tool but clutter the presentation.

- (Alice) :: r :: $[(pk(B, n(A, r); A))^+, (pk(A, n(A, r); B \quad Y))^-, (pk(B, Y))^+]$
- (Bob) :: r :: $[(pk(B, X; A))^-, (pk(A, X; B \quad n(B, r)))^+, (pk(B, n(B, r)))^-]$

Note that r, r are used for nonce generation (they are special variables handled as unique constants in order to obtain an infinite number of available constants).

There are also strands for initial knowledge and actions of the intruder, such as concatenation, deconcatenation, encryption, decryption, etc. For example, concatenation by the intruder is described by the strand $[(X)^-, (Y)^-, (X; Y)^+]$. We will show the full list of intruder capabilities in Section 6.

Our protocol analysis methodology is then based on the idea of 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. In order to perform such a reachability analysis we must describe how states change as a consequence of principals performing protocol steps and of intruder actions. This can be done by describing such state changes by means of a set R_P of rewrite rules, so that the rewrite theory $(\ _P, E_P, R_P)$ characterizes the behavior of protocol P modulo the equations E_P. The following rewrite rules describe the general state transitions, where each state transition implies moving rightwards the vertical bar of one strand:

$$\begin{array}{ll}
SS \ \& \ [L \mid M^-, L] \ \& \ \{M \quad I, IK\} & SS \ \& \ [L, M^- \mid L] \ \& \ \{IK\} \\
SS \ \& \ [L \mid M^+, L] \ \& \ \{IK\} & SS \ \& \ [L, M^+ \mid L] \ \& \ \{IK\} \\
SS \ \& \ [L \mid M^+, L] \ \& \ \{M \ /I, IK\} & SS \ \& \ [L, M^+ \mid L] \ \& \ \{M \quad I, IK\}
\end{array}$$

variables L, L denote lists of input and output messages (m^+, m^-) within a strand, IK denotes a set of intruder facts ($m \;\; I$, $m \,/ I$), and SS denotes a set of strands. An unbounded number of sessions is handled by another rewrite rule introducing an extra strand $[m_1^\pm, \ldots, m_{j-1}^\pm \mid m_j^+, msg_{j+1}^\pm, \ldots, m_k^\pm]$ for an intruder knowledge fact of the form $m_j \;\; I$. See [15] for further information.

The way to analyze backwards reachability is then relatively easy, namely to run the protocol "in reverse." This can be achieved by using the set of rules R_P^{-1}, where $v - \;\; u$ is in R_P^{-1} iff $u - \;\; v$ is in R_P. Reachability analysis can be performed symbolically, not on concrete states but on symbolic state patterns $[t(x_1, \ldots, x_n)]$ by means of narrowing modulo E_P (see Section 2 and [24,32]).

E_P-unification precisely models all the different ways in which an intruder could exploit the algebraic properties E_P of P to break the protocol; therefore, if an initial state can be shown unreachable by backwards reachability analysis modulo E_P from an attack state pattern, this ensures that, even if the intruder uses the algebraic properties E_P, the attack cannot be mounted. This means that efficient support for E_P-unification is a crucial feature of symbolic reachability analysis of protocols modulo their algebraic properties E_P.

4 A Unification Algorithm for XOR pk-sk AC

In general, combining unification algorithms for a theory $E = E_1 \;\; E_2 \;\; \ldots \;\; E_n$ is computationally quite expensive, and typically assumes that the symbols in E_i and E_j are pairwise disjoint for each $i = j$. This is due to the substantial amount of non-determinism involved in the inference systems supporting such combinations (see [3]). In our NSL example, $E = E_1 \;\; E_2 \;\; E_3$, where E_1 is the XOR theory, E_2 is the theory pk-sk given by the two public key encryption equations $pk(K, sk(K, M)) = M$ and $sk(K, pk(K, M)) = M$, and E_3 is the AC theory for each of the state constructors _,_ and _&_ explained in Section 3. To further complicate the matter, we need to combine not just untyped unification algorithms, but typed, and more precisely order-sorted ones.

Fortunately, the variant-narrowing-based approach that we use in this paper avoids all these difficulties by obtaining the (XOR pk-sk AC)-unification algorithm as an instance of the variant narrowing methodology supported by Maude-NPA. The point is that if an equational theory E has the finite variant property [12], then a finitary E-unification algorithm can be obtained by variant narrowing [20,19], as further explained in Section 5. In our case, the equations in the theory pk-sk are confluent and terminating and, furthermore, have the finite variant property. Likewise, the equations in the XOR theory presented in Section 3 are confluent, terminating and coherent modulo the AC axioms of and also have the finite variant property. Finally, the theory of AC for the state-building constructors _,_ and _&_ is of course finitary and can be viewed as a trivial case of a theory with the finite variant property (decomposed with no rules and only axioms). Note that all these three equational theories are disjoint, i.e., they do not share any symbols. The good news is that the following disjoint union theory XOR pk-sk AC with $_{NSL}$ being the entire (order-sorted) signature of our NSL$\oplus$ protocol example is also confluent, terminating and coherent modulo the AC axioms[5], and satisfies the finite variant property:

1. Rules:
   - $pk(K,sk(K,M)) = M$, $sk(K,pk(K,M)) = M$,
   - $X \oplus 0 = X$, $X \oplus X = 0$, $X \oplus X \oplus Y = Y$,
2. Axioms: AC for $\oplus$, AC for _,_ and AC for _&_

Therefore, Maude-NPA can analyze the NSL$\oplus$ protocol using variant narrowing. In the following we explain variant narrowing in more detail.

5 Variant Narrowing and Variant Unification

Suppose that an equational theory E is decomposed according to the following definition.

Definition 1 (Decomposition [19]). Let (Σ,E) be an order-sorted equational theory. We call $(\Sigma,Ax,\vec{E})$ a decomposition of (Σ,E) if $E = \vec{E} \uplus Ax$ and $(\Sigma,Ax,\vec{E})$ is an order-sorted rewrite theory satisfying the following properties.

1. Ax is regular, i.e., for each $t = t'$ in Ax, we have $Var(t) = Var(t')$, and sort-preserving, i.e., for each substitution θ, we have $t\theta \in T_\Sigma(X)_s$ iff $t'\theta \in T_\Sigma(X)_s$; furthermore all variables in $Var(t)$ have a top sort.
2. Ax has a finitary and complete unification algorithm.
3. For each $t \to t'$ in $\vec{E}$ we have $Var(t') \subseteq Var(t)$.
4. $\vec{E}$ is sort-decreasing, i.e., for each $t \to t'$ in $\vec{E}$, each $s \in S$, and each substitution θ, $t\theta \in T_\Sigma(X)_s$ implies $t'\theta \in T_\Sigma(X)_s$.
5. The rewrite rules $\vec{E}$ are confluent and terminating modulo Ax, i.e., the relation $\to_{\vec{E},Ax}$ is confluent and terminating.
6. The relation $\to_{\vec{E},Ax}$ is Ax-coherent.

Given a term t, an $\vec{E},Ax$-variant of t is a pair (t',θ) with t' an $\vec{E},Ax$-canonical form of the term $t\theta$. That is, the variants of a term intuitively give us all the irreducible patterns that instances of t can reduce to. Of course, some variants are more general than others, i.e., there is a natural preorder $(t_1,\theta_1) \sqsubseteq_{\vec{E},Ax} (t_2,\theta_2)$ defining when variant (t_2,θ_2) is more general than variant (t_1,θ_1). This is important, because even though the set of $\vec{E},Ax$-variants of a term t may be infinite,

[5] All these conditions are easily checkable. Indeed, coherence modulo the combined AC axioms is immediate, and we can use standard methods and tools to check the local confluence and termination of the combined theory; similarly, the method described in [17] can be used to check the finite variant property of the combined theory. Alternatively, one can use modular methods to check that a combined theory satisfies all these properties under certain assumptions: see [33] for a good survey of modularity results for confluence and termination. Likewise, the finite variant property can also be checked modularly under appropriate assumptions, but a discussion of this topic is beyond the scope of this paper.

the set of most general variants (i.e., maximal elements in the generalization preorder up to Ax-equivalence and variable renaming) may be finite.

The intimate connection of variants with E-unification is then as follows. Suppose that we add to our theory decomposition E Ax a binary equality predicate eq, a new constant tt[6] and for each top sort [s] and x of sort [s] an extra rule eq(x,x)   tt. Then, given any two terms t,t , if   is a E-unifier of t and t , then the E,Ax canonical forms of t and t must be Ax-equal and therefore the pair (tt,) must be a variant of the term eq(t,t). Furthermore, if the term eq(t,t) has a finite set of most general variants, then we are guaranteed that the set of most general E-unifiers of t and t is finite.

For any theory E Ax with E confluent, terminating, and coherent modulo Ax, the folding variant narrowing of [20] is a general and effective complete strategy. Complete both in the sense of computing a complete set of E Ax-unifiers, and of computing a minimal and complete set of variants for any input term t.

In the following, we characterize a notion of variant semantics for equational theories.

Definition 2 (Variant Semantics [20]). Let (,Ax,E) be a decomposition of an equational theory and t be a term. We define the set of variants of t as $[\![t]\!]_{E,Ax}$ = {(t ,) | Subst(,X),t $\overset{!}{}_{E,Ax}$ t , and t $=_{Ax}$ t }.

Example 1. Let us consider the equational theory XOR pk-sk, which, together with AC for _,_ and _&_ is used for our NSL protocol presented in Section 3. This equational theory is relevant because none of our previously defined unification procedures is directly applicable to it, e.g. unification algorithms for exclusive-or such as [22] do not directly apply if extra equations are added.

For (,Ax,E) a decomposition of XOR pk-sk, and for terms t = M   sk(K,pk(K,M)) and s = X   sk(K,pk(K,Y)), we have that $[\![t]\!]_{E,Ax}$ = {(0,id),...} and

$[\![s]\!]_{E,Ax}$ = {(X   Y,id),
(Z,{X   0,Y   Z}),(Z,{X   Z,Y   0}),
(Z,{X   Z   U,Y   U}),(Z,{X   U,Y   Z   U}),
(0,{X   U,Y   U}),(Z_1 Z_2,{X   U Z_1,Y   U Z_2}),
(0,{X   V   W,Y   V   W}),...}

We write (t_1, $_1$) $_{E,Ax}$ (t_2, $_2$) to denote that variant (t_2, $_2$) is more general than variant (t_1, $_1$).

Definition 3 (Variant Preordering [20]). Let (,Ax,E) be a decomposition of an equational theory and t be a term. Given two variants (t_1, $_1$),(t_2, $_2$) $[\![t]\!]_{E,Ax}$, we write (t_1, $_1$) $_{E,Ax}$ (t_2, $_2$), meaning (t_2, $_2$) is more general than

[6] We extend   to ^{b} by adding a new sort Truth, not related to any sort in , with constant tt, and for each top sort [s] of a connected component, an operator eq : [s] × [s]   Truth.

$(t_1, \ _1)$, iff there is a substitution such that $t_1 =_{Ax} t_2$ and $_{1\ E,Ax} =_{Ax}\ _2$. We write $(t_1, \ _1)\ _{E,Ax}\ (t_2, \ _2)$ if for every substitution such that $t_1 =_{Ax} t_2$ and $_{1\ E,Ax} =_{Ax}\ _2$, then is not a renaming.

Example 2. Continuing Example 1 we have $v_1 = (0, \{X \ \ U, Y \ \ U\})$ as a valid variant of s. Also, $v_2 = (0, \{X \ \ V \ W, Y \ \ V \ W\})$ is a valid variant of s but clearly $v_2 \ _{E,Ax} v_1$, and thus v_2 should not be included in the most general set of variants. On the other hand for $u_1 = (X \ Y, id)$ and $u_2 = (Z, \{X \ \ 0, Y \ \ Z\})$, we have that neither $u_1 \ _{E,Ax} u_2$ nor $u_2 \ _{E,Ax} u_1$ hold.

We are, indeed, interested in equivalence classes for variant semantics and provide a notion of equivalence of variants up to renaming, written $_{Ax}$.

Definition 4 (Ax-Equivalence [20]). Let $(\ , Ax, E)$ be a decomposition of an equational theory and t be a term. For $(t_1, \ _1), (t_2, \ _2) \ \ [\![t]\!]_{E,Ax}$, we write $(t_1, \ _1) \ _{Ax} (t_2, \ _2)$ if there is a variable renaming such that $t_1 \ =_{Ax} t_2$ and $_1 \ =_{Ax} \ _2$. For $S_1, S_2 \ \ [\![t]\!]_{E,Ax}$, we write $S_1 \ _{Ax} S_2$ if for each $(t_1, \ _1) \ \ S_1$, there exists $(t_2, \ _2) \ \ S_2$ s.t. $(t_1, \ _1) \ _{Ax} (t_2, \ _2)$, and for each $(t_2, \ _2) \ \ S_2$, there exists $(t_1, \ _1) \ \ S_1$ s.t. $(t_2, \ _2) \ _{Ax} (t_1, \ _1)$.

The preorder of Definition 3 allows us to provide a most general and complete set of variants that encompasses all the variants for a term t.

Definition 5 (Most General and Complete Variant Semantics [20]). Let $(\ , Ax, E)$ be a decomposition of an equational theory and t be a term. A most general and complete variant semantics of t, denoted $[\![t]\!]_{E,Ax}$, is a subset $[\![t]\!]_{E,Ax} \ \ [\![t]\!]_{E,Ax}$ such that: (i) $[\![t]\!]_{E,Ax} \ _{E,Ax} \ [\![t]\!]_{E,Ax}$, and (ii) for each $(t_1, \ _1) \ \ [\![t]\!]_{E,Ax}$, there is no $(t_2, \ _2) \ \ [\![t]\!]_{E,Ax}$ s.t. $(t_1, \ _1) \ _{Ax} \ (t_2, \ _2)$ and $(t_1, \ _1) \ _{E,Ax} (t_2, \ _2)$.

Example 3. Continuing Example 1 it is obvious that the following variants are most general w.r.t. $_{E,Ax}$: $[\![t]\!]_{E,Ax} = \{(0, id)\}$ and

$$\begin{aligned}[\![s]\!]_{E,Ax} = \{ & (X \ \ Y, id), \\ & (Z, \{X \ \ 0, Y \ \ Z\}), (Z, \{X \ \ Z, Y \ \ 0\}), \\ & (Z, \{X \ \ Z \ \ U, Y \ \ U\}), (Z, \{X \ \ U, Y \ \ Z \ \ U\}), \\ & (0, \{X \ \ U, Y \ \ U\}), (Z_1 \ \ Z_2, \{X \ \ U \ \ Z_1, Y \ \ U \ \ Z_2\})\}.\end{aligned}$$

Note that, by definition, all the substitutions in $[\![t]\!]_{E,Ax}$ are E,Ax-normalized. Moreover, $[\![t]\!]_{E,Ax}$ is unique up to $_{Ax}$ and provides a very succinct description of $[\![t]\!]_{E,Ax}$. Indeed, up to Ax-equality, $[\![t]\!]_{E,Ax}$ characterizes the set of maximal elements (therefore, most general variants) of the preorder $([\![t]\!]_{E,Ax}, \ _{E,Ax})$.

Again, let us make explicit the relation between variants and E-unification.

Proposition 1 (Minimal and Complete E-unification [20]). Let $(\ , Ax, E)$ be a decomposition of an equational theory $(\ , E)$. Let t, t be two terms. Then, $S = \{ \ \ | \ (tt, \ ) \ \ [\![eq(t, t)]\!]_{E,Ax}\}$ is a minimal and complete set of E-unifiers for $t = t$, where eq and tt are new symbols defined in Footnote 6 and $E = E \ \ \{eq(X, X) \ \ tt\}$.

The finite variant property defined by Comon-Lundh and Delaune [12], provides a useful sufficient condition for finitary E-unification. Essentially, it determines whether every term has a finite number of most general variants.

Definition 6 (Finite variant property [12]). Let $(,Ax,E)$ be a decomposition of an equational theory $(,E)$. Then $(,E)$, and thus $(,Ax,E)$, has the finite variant property iff for each term t, the set $[\![t]\!]_{E,Ax}$ is finite. We will call $(,Ax,E)$ a finite variant decomposition of $(,E)$ iff $(,Ax,E)$ has the finite variant property.

In [18] a technique is proposed to check whether an equational theory has the finite variant property. Using this technique it is easy to check that Example 1 has the finite variant property, as every right–hand side is a constant symbol or a variable. See [18, Example 2] for more details.

Finally, it is clear that when we have a finite variant decomposition, we also have a finitary unification algorithm.

Corollary 1 (Finitary E-unification [20]). Let $(,Ax,E)$ be a finite variant decomposition of an equational theory $(,E)$. Then, for any two given terms t,t, $S = \{\ \mid (tt,)\ \ [\![eq(t,t)]\!]_{E,Ax}\}$ is a finite, minimal, and complete set of E-unifiers for $t = t$, where E, eq, and tt are defined as in Proposition 1.

Note that the opposite does not hold: given two terms t,t that have a finite, minimal, and complete set of E-unifiers, the equational theory $(,E)$ may not have a finite variant decomposition $(,Ax,E)$. An example is the unification under homomorphism (or one-side distributivity), where there is a finite number of unifiers of two terms but the theory does not satisfy the finite variant property (see [12,18]); the key idea is that the term $eq(t,t)$ may have an infinite number of variants even though there is only a finite set of most general variants of the form $(tt,)$. We refer the reader to [20] for further information.

Currently, Maude-NPA restricts itself to a subset of theories satisfying the finite variant property:

1. The axioms Ax can declare some binary operators in to be commutative (with the comm attribute), or associative-commutative (with the assoc and comm attributes).
2. The set of rewrite rules E is strongly right irreducible, that is no instance of the right-hand side of a rule in E by a normalized substitution can be further simplified by the application the equations in E modulo Ax.

The reasons for restricting ourselves in this way is for efficiency and ease of implementation. Maude currently supports unification modulo commutative and associative-commutative theories, as well as syntactic unification, so this is what drives our choice of Ax. Furthermore, the restriction of E to strongly right irreducible theories means that the depth of the narrowing tree is bounded by the number of symbols in a term. Moreover, many of the finite variant theories that arise in cryptographic protocol analysis satisfy strong right irreducibility. These include encryption-decryption cancellation, exclusive-or, and modular exponentiation. The major exception is Abelian groups (other than those described by exclusive-or). We are currently working on implementing full variant narrowing in Maude-NPA to handle these and other cases not currently covered by strong right irreducibility.

6 Finding attacks modulo XOR pk-sk AC using Maude-NPA

We have analyzed the NSL protocol presented in Section 3 modulo its equational theory XOR pk-sk AC in Maude-NPA using variant narrowing.

We now explain in more detail all the operations available to the intruder. Its capabilities are all given in strand notation. Note that we are omitting the position marker | which is assumed to be at the beginning.

(s1) $[(X)^-, (Y)^-, (X;Y)^+]$ Concatenation
(s2) $[(X;Y)^-, (X)^+]$ Left-deconcatenation
(s3) $[(X;Y)^-, (Y)^+]$ Right-deconcatenation
(s4) $[(X)^-, (Y)^-, (X\quad Y)^+]$ Exclusive–or
(s6) $[(X)^-, (sk(i,X))^+]$ Encryption with i's private key
(s7) $[(X)^-, (pk(A,X))^+]$ Encryption with any public key
(s8) $[(0)^+]$ Generate the exclusive–or neutral element
(s9) $[(A)^+]$ Generate any principal's name.

The attack state pattern from which we start the backwards narrowing search in this example is given by one strand, representing Bob (b) wanting to communicate with Alice (a)

$$:: r :: [(pk(b,X;a))^-, (pk(a,X;b\quad n(b,r)))^+, (pk(b,n(b,r)))^- \,|\, nil]$$

together with requiring the intruder (i) to have learned Bob's nonce, i.e., n(b,r) I. What this represents is an attack in which Bob has properly executed the protocol and believes to be talking to Alice, while the intruder has obtained the nonce that Bob created and considers a secret shared between Alice and him.

See Figure 1 for a pictorial representation of the strand space and messages sent and received, depicting the attack found by Maude-NPA. This attack agrees with the one described in [8]. The figure has been created with the help of the Maude-NPA GUI [34], with the exclusive–or symbol textually represented as in the figure.

7 Related Work

There is a substantial amount of research on formal verification of cryptographic protocols. Much of it abstracts away from any equational theories obeyed by the cryptographic operators, but there is a growing amount of work addressing this

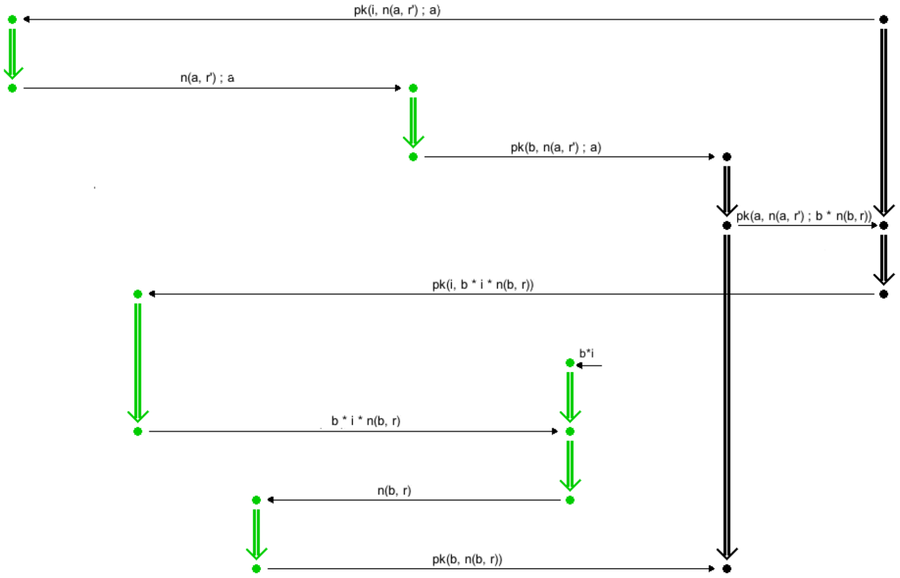

Fig. 1. Pictorial representation of the initial state, leading to an attack

problem. The earliest was the NRL Protocol Analyzer [29], which, like Maude-NPA, was based on unification and backwards search, implemented via narrowing over confluent equational theories. This was sufficient to handle, for example, the cancellation of encryption and decryption, although there were many theories of interest it did not address, such as exclusive-or and other Abelian group operators.

More recently, tools have begun to offer support for specification and, to some degree, analysis of protocols involving equational theories. These tools include, for example, ProVerif [6], OFMC [4], and CL-Atse [37]. Both OFMC and CL-Atse work in the bounded session model, while ProVerif uses abstraction and unbounded sessions. Both OFMC and CL-Atse support exclusive-or and Diffie-Hellman exponentiation. ProVerif can also be used to analyze these, but the equational theories it is known to work well with are more limited, e.g. not supporting associativity-commutativity or Diffie-Hellman exponentiation. However, Küsters and Truderung [25,26] have developed algorithms that can translate protocols using exclusive-or or Diffie-Hellman exponentiation to protocols that can be analyzed by ProVerif in a free algebra model; for exclusive-or they can handle protocols satisfying the -linearity property. According to a study by Lafourcade et al. [27], this produces analysis times that are only slightly slower than analyses by OFMC and CL-Atse, mainly because of the translation time.

There is also a growing amount of theoretical work on cryptographic protocol analysis using equational theories, e.g. [1,9,7,11,5]. This concentrates on the decidability of problems of interest to cryptographic protocol analysis, such as deducibility, which means that it is possible (e.g. for an intruder) to deduce a term from a set of terms, and static equivalence, which means that an intruder cannot tell the difference between two sets of terms. However, there is much less work on the combination of different theories, although Arnaud, Cortier, and Delaune [13] have considered the problem in terms of decidability of the problem for combination of disjoint theories, showing that if any two disjoint theories have decidable static equivalence problems, then so does their combination. More recently Chevalier and Rusinowitch analyze the security of cryptographic protocols via constraint systems and have also studied composition of theories. In [10], they give a general method for combining disjoint theories that is based on the Baader-Schulz combination algorithm for unification algorithms for different theories [3]. This can be thought of as a constraint-based analogue of the Maude-NPA combination framework, which is also based on the Baader-Schulz combination algorithm [3].

8 Conclusions and Future Work

To gain high assurance about cryptographic protocols using formal methods requires reasoning modulo the algebraic properties of the underlying cryptographic functions. In symbolic analyses this typically necessitates performing unification modulo such algebraic properties. However, since a protocol may use a variety of different functions — so that different protocols typically require reasoning modulo different theories— it is unrealistic to expect that a fixed set of unification algorithms will suffice for such analyses. That is, combination methods that obtain unification algorithm for a composition of theories out of a family of such algorithm for each of them, are unavoidable. Standard methods for obtaining a unification algorithm for a combined theory $E_1 \quad \ldots \quad E_n$ [3] are computationally costly due to the high degree of non-determinism in the combination method; furthermore, they require the existence of a unification algorithm for each individual theory E_i, which in practice may not be available in a tool's infrastructure. In this work we have proposed an alternative method based on variant narrowing to obtain a $(E_1 \quad \ldots \quad E_n)$-unification algorithm under simpler requirements. Specifically, dedicated implementations of unification algorithms for each of the theories E_i are not needed: in our example, only a dedicated AC-unification algorithm was used: no dedicated algorithms for XOR of pk-sk were needed. Furthermore, even though narrowing is less efficient than a dedicated algorithm for each individual theory E_i, the costly computational overhead of a standard combination method is avoided. The case study presented has shown that variant narrowing, as supported by the Maude-NPA, is indeed an effective method to deal with nontrivial combinations of equational theories; and for analyzing many protocols with even a modest infrastructure of built-in unification algorithms. The case study was chosen as a well-known protocol for illustration purposes, but many other examples could have been given.

We should emphasize that standard combination methods such as those described in [3], and the alternative variant narrowing method presented here are not "rival" methods. Instead they are highly complementary methods which, when used in tandem, allow a tool to analyze a much wider range of protocols than those analyzable by each method in isolation. Let us use our example theory XOR pk-sk AC to illustrate this important point. Variant narrowing decomposed this combined theory into: (i) three rewrite rules for XOR and two rewrite rules for pk-sk plus, (ii) three instances of AC: one for , another for _,_ and another for _&_. That is, variant narrowing with the rules in (i) was performed modulo the axioms in (ii). But the axioms in (ii) are themselves a combined theory (in fact, also combined with all the other function symbols in the protocol specification as free function symbols). The Maude infrastructure used by Maude-NPA has in fact used an order-sorted version of a standard combination method in the style of [3] to support unification with the combined axioms of (ii). Therefore, the advantage of using standard combination methods and variant narrowing in tandem is the following:

1. A given tool infrastructure can only have a finite number of predefined (finitary) unification algorithms for, say, theories $T_1, \ldots, T_k$; however, it should also be able to support any combination of such built-in theories by a standard combination method.
2. A given protocol may require performing unification modulo a combination of theories E_1 ... E_n, but some of the E_i may not belong to the library $T_1, \ldots, T_k$, so that the standard combination method cannot be used.
3. However, if E_1 ... E_n can be refactored as a theory decomposition (,B,R) that: (i) it has the finite variant property; and (ii) B is a combination of the theories $T_1, \ldots, T_k$ supported by the current library, then a finitary (E_1 ... E_n)-unification algorithm can be obtained by variant narrowing.

A very important direction for future work in formal tools supporting symbolic protocol analysis modulo equational properties consists in: (i) developing methods for expanding a tool's built-in unification infrastructure as described in (1) above to make it as efficient and extensible as possible; and (ii) improving and optimizing the methods for efficient variant narrowing modulo such infrastructure. Good candidates for new theories T_j to be added to the built-in infrastructure include commonly used theories, with high priority given to theories that lack the finite variant properties. For example, the theory of homomorphic encryption, which lacks the finite variant property, has been recently added to Maude-NPA for exactly this purpose.

References

1. M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. Theoretical Computer Science, 367(1-2):2–32, 2006.
2. A. Armando, D. A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron.

The avispa tool for the automated validation of internet security protocols and applications. In CAV, LNCS 3576:281–285. Springer, 2005.

3. F. Baader and K. U. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. In CADE, LNCS 607:50–65. Springer, 1992.
4. D. A. Basin, S. Mödersheim, and L. Viganò. An on-the-fly model-checker for security protocol analysis. In ESORICS 2003, LNCS 2808:253–270. Springer, 2003.
5. M. Baudet, V. Cortier, and S. Delaune. Yapa: A generic tool for computing intruder knowledge. In RTA 2009, LNCS 5595:148–163. Springer, 2009.
6. B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In CSFW, pages 82–96. IEEE Computer Society, 2001.
7. S. Bursuc and H. Comon-Lundh. Protocol security and algebraic properties: Decision results for a bounded number of sessions. In RTA 2009, LNCS 5595:133–147. Springer, 2009.
8. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with XOR. In LICS, pages 261–270. IEEE Computer Society, 2003.
9. Y. Chevalier and M. Rusinowitch. Hierarchical combination of intruder theories. Inf. Comput., 206(2-4):352–377, 2008.
10. Y. Chevalier and M. Rusinowitch. Symbolic protocol analysis in the union of disjoint intruder theories: Combining decision procedures. Theor. Comput. Sci., 411(10):1261–1282, 2010.
11. S. Ciobâca, S. Delaune, and S. Kremer. Computing knowledge in security protocols under convergent equational theories. In CADE-22, LNCS 5663:355–370. Springer, 2009.
12. H. Comon-Lundh and S. Delaune. The finite variant property: How to get rid of some algebraic properties. In RTA 2005, LNCS 3467:294–307. Springer, 2005.
13. V. Cortier, J. Delaitre, and S. Delaune. Safely composing security protocols. In FSTTCS, LNCS 4855:352–363. Springer, 2007.
14. C. J. F. Cremers. The Scyther tool: Verification, falsification, and analysis of security protocols. In CAV, pages 414–418, 2008.
15. S. Escobar, C. Meadows, and J. Meseguer. A rewriting-based inference system for the NRL protocol analyzer and its meta-logical properties. Theoretical Computer Science, 367(1-2):162–202, 2006.
16. S. Escobar, C. Meadows, and J. Meseguer. Maude-NPA: Cryptographic protocol analysis modulo equational properties. In FOSAD 2007/2008/2009 Tutorial Lectures, LNCS 5705:1–50. Springer, 2009.
17. S. Escobar, J. Meseguer, and R. Sasse. Effectively checking or disproving the finite variant property. Technical Report UIUCDCS-R-2008-2960, Department of Computer Science - University of Illinois at Urbana-Champaign, April 2008.
18. S. Escobar, J. Meseguer, and R. Sasse. Effectively checking the finite variant property. In RTA'08, LNCS 5117:79–93. Springer, 2008.
19. S. Escobar, J. Meseguer, and R. Sasse. Variant narrowing and equational unification. Electr. Notes Theor. Comput. Sci., 238(3):103–119, 2009.
20. S. Escobar, R. Sasse, and J. Meseguer. Folding variant narrowing and optimal variant termination. In WRLA 2010, LNCS 6381:52–68. Springer, 2010.
21. 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.
22. Q. Guo and P. Narendran. Unification and matching modulo nilpotence. In CADE-13, LNCS 1104:261–274. Springer-Verlag, 1996.
23. D. Harkins and D. Carrel. The Internet Key Exchange (IKE), November 1998. IETF RFC 2409.

24. J.-P. Jouannaud, C. Kirchner, and H. Kirchner. Incremental construction of unification algorithms in equational theories. In ICALP, LNCS 154:361–373. Springer, 1983.
25. R. Küsters and T. Truderung. Reducing protocol analysis with xor to the xor-free case in the Horn theory based approach. In ACM Conference on Computer and Communications Security, pages 129–138, 2008.
26. R. Küsters and T. Truderung. Using ProVerif to analyze protocols with Diffie-Hellman exponentiation. In CSF, pages 157–171. IEEE Computer Society, 2009.
27. P. Lafourcade, V. Terrade, and S. Vigier. Comparison of cryptographic verification tools dealing with algebraic properties. In Formal Aspects in Security and Trust, pages 173–185, 2009.
28. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In TACAS, pages 147–166, 1996.
29. C. Meadows. The NRL protocol analyzer: An overview. J. Log. Program., 26(2):113–131, 1996.
30. J. Meseguer. Conditional rewriting logic as a united model of concurrency. Theor. Comput. Sci., 96(1):73–155, 1992.
31. J. Meseguer. Membership algebra as a logical framework for equational specification. In F. Parisi-Presicce, editor, Proc. WADT'97, LNCS 1376:18–61. Springer, 1998.
32. J. Meseguer and P. Thati. Symbolic reachability analysis using narrowing and its application to verification of cryptographic protocols. Higher-Order and Symbolic Computation, 20(1–2):123–160, 2007.
33. E. Ohlebusch. Advanced Topics in Term Rewriting. Springer Verlag, 2002.
34. S. Santiago, C. L. Talcott, S. Escobar, C. Meadows, and J. Meseguer. A graphical user interface for Maude-NPA. Electr. Notes Theor. Comput. Sci., 258(1):3–20, 2009.
35. M. Schmidt-Schauß. Unification in a combination of arbitrary disjoint equational theories. J. Symb. Comput., 8(1/2):51–99, 1989.
36. TeReSe, editor. Term Rewriting Systems. Cambridge University Press, Cambridge, 2003.
37. M. Turuani. The cl-atse protocol analyser. In RTA 2006, LNCS 4098:277–286. Springer, 2006.