Symbolic Protocol Analysis with Products and Diffie-Hellman Exponentiation

Jonathan Millen and Vitaly Shmatikov
Computer Science Laboratory
SRI International
{millen,shmat}@csl.sri.com

Abstract

*We demonstrate that for any well-defined cryptographic protocol, the symbolic trace reachability problem in the presence of an Abelian group operator (*e.g.*, multiplication) can be reduced to solvability of a particular system of quadratic Diophantine equations. This result enables formal analysis of protocols that employ primitives such as Diffie-Hellman exponentiation, products, and* `xor`*, with a bounded number of role instances, but without imposing any bounds on the size of terms created by the attacker. In the case of* `xor`*, the resulting system of Diophantine equations is decidable. In the case of a general Abelian group, decidability remains an open question, but our reduction demonstrates that standard mathematical techniques for solving systems of Diophantine equations are sufficient for the discovery of protocol insecurities.*

1. Introduction

Conventional formal analysis of cryptographic protocols relies on the so called "Dolev-Yao" attacker model. The underlying cryptographic primitives are treated as black boxes, and it is assumed, for analysis purposes, that the attacker does not have access to the algebraic properties of the corresponding mathematical functions. This assumption is simply not true for primitives such as `xor` (exclusive or), multiplication, and Diffie-Hellman exponentiation, which are widely used in protocol constructions. The attacker can and will exploit associativity, commutativity, cancellation, and other properties of cryptographic operations. For example, Bull's recursive authentication protocol was formally proved correct in a model that treated `xor` as an abstract encryption, and then found to be vulnerable once self-cancellation properties of `xor` are taken into account [17, 21].

We use *symbolic trace reachability* as the standard representation of the protocol analysis problem for a trace-based security property (including secrecy and authentication properties), a finite number of sessions and an unbounded attacker (*i.e.*, there are no upper limits on the size of terms the attacker may create) [1, 10, 3, 16]. The problem is symbolic because messages in the protocol specification may contain variables, representing a data field whose value is not known in advance to the recipient. The attacker can choose arbitrary values for the variables, provided he does so consistently throughout the trace. Consider a trace representing an attack (*e.g.*, in the case of a secrecy property, an interleaving of several protocol executions at the end of which the attacker outputs the value that is supposed to remain secret). Then asking

"Is the protocol insecure?"

is equivalent to asking:

"Given the symbolic attack trace, is there a consistent instantiation of variables such that every computation performed by the attacker in this trace is feasible?"

Some previous research on symbolic trace reachability assumed a free term algebra, which does not represent algebraic properties of the cryptographic functions (decryption is handled implicitly). Even in this setting, the problem has been shown to be NP-complete [20]. The free term algebra assumption, however, is inadequate for analyzing protocols based on Diffie-Hellman exponentiation such as [23], those involving `xor` (*e.g.*, block ciphers), or any protocol using multiplication (*e.g.*, ElGamal).

Our main contribution is to extend the constraint solving approach, first proposed in [16], to handle the algebraic properties of Abelian group operators. For any well-defined cryptographic protocol, we show that symbolic trace reachability is equivalent to solvability in integers of a special system of quadratic equations. Quadratic Diophantine equations have been a subject of extensive investigation in mathematics, and partial solution techniques can thus be used to complete the analysis. If the group operator is interpreted as `xor`, the system degenerates into one for which solutions can be found by exhaustive enumeration, thus establishing that protocol analysis can be fully automated for `xor`. Decidability of finite-session protocol analysis without equational properties, first proved in [20, 6, 16], also follows as a degenerate case.

Overview. In Section 2, we introduce our formal model and describe how to reduce the protocol analysis problem to a sequence of symbolic constraints. In Section 3, we posit the variable stability condition which is a necessary property of any well-defined protocol. In Section 4, we summarize the theory of ground term derivability in the presence of an Abelian group operator, due to Comon-Lundh and Shmatikov, which is being published concurrently [7] (see related work below).

The main technical result of the paper appears in Section 5. If the constraint sequence has a solution, we prove that it has a *conservative* solution. Intuitively, the conservative solution only uses the structure that is already present in the original sequence. We show that the substitution for any variable is a product of terms (and their inverses) drawn from a finite set: the non-variable, non-product subterms of the original symbolic sequence. The resulting set of product derivability problems is naturally reduced to a system of quadratic Diophantine equations, as shown in Section 6. One of the steps along the way is Abelian group unification, which is known to be decidable [2].

In Section 7, we extend our approach to protocols with Diffie-Hellman exponentiation such as GDH [23], under the restriction that multiplication may appear only in exponents (the problem is undecidable otherwise). We replace exponentials by a combination of products and uninterpreted functions, which reduces the symbolic analysis problem for such protocols to the solvability of a symbolic constraint sequence with an Abelian group operator. Conclusions are in Section 8.

Related work. Pereira and Quisquater [19] developed a technique for analyzing group Diffie-Hellman (GDH) protocols that takes into account algebraic properties of Diffie-Hellman exponents. Their approach is specific to GDH-based protocols, and the attacker model is restricted correspondingly (*e.g.*, the attacker is not even equipped with the ability to perform standard symmetric encryption). They do not attempt to address the general problem of deciding whether a term is derivable in an attacker algebra with the equational theory of multiplication, or whether a particular symbolic attack trace has a feasible instantiation. Since they only consider the problem in the ground case, the resulting system of equations is linear, whereas the system we obtain in the general case with variables is quadratic (see Section 6). An application of our approach to one of the Pereira-Quisquater examples is summarized in Section 7.

Recent research by Narendran *et al.* focuses on decidability of unification modulo the equational theory of multiplication and exponentiation [15, 12, 13]. While equational unification is an important subproblem in symbolic protocol analysis, unification alone is insufficient to decide whether a particular symbolic attack trace is feasible.

Decidability of symbolic protocol analysis in the presence of `xor` has been proved in [4, 7]. Chevalier *et al.* [4] showed that the problem is NP-complete in a restricted protocol model which is very similar to the one proposed in this paper. They do not consider Abelian group operators at all, and their techniques cannot be used directly to model products or Diffie-Hellman exponentiation. Independently, Comon-Lundh and Shmatikov [7] demonstrated decidability of symbolic protocol analysis with `xor` in the unrestricted model.

This paper lifts the results of [7] by considering the symbolic analysis problem in the presence of an arbitrary Abelian group operator, resulting in a substantially more complicated theory than in the `xor` case. By contrast, [7] only considers Abelian group operators in the ground case, and obtains symbolic decidability results for `xor` only.

2. Model

We use the strand space model of [24]. A strand is a sequence of nodes. Associated with each node is a message term with a sign, $+$ or $-$, indicating that the message is sent or received, respectively. Although messages in strand nodes are ground terms, protocol roles are specified as *strand schemas*, in which message terms may contain variables. A variable can represent either a data field whose value is chosen externally, or a nonce generated by that role.

A protocol specification is a set of roles, and instances of those roles form a strand space. A strand bundle is essentially a Lamport diagram [14] in which the processes are strands, and message connections are obtained by matching receive nodes with send nodes, yielding a causal partial ordering of nodes. (Lamport called this a space-time diagram, but others renamed it in the context of distributed systems.)

It is shown in [24] and elsewhere how security questions can be reduced to questions about the existence of a bundle including strands from the protocol roles, plus additional strands representing attacker computations and security property tests. In our constraint solving approach begun in [16], bundle existence is determined by starting with a *semibundle* (terminology from [22]), in which message terms may contain variables and the sources of received messages have not yet been determined. In a semibundle

$\langle t_1, t_2 \rangle$	Pairing of terms t_1 and t_2.
$\{t_1\}_{t_2}$	Term t_1 encrypted with term t_2 using a symmetric algorithm.
$t_1 \cdot \ldots \cdot t_n$	Product of terms.
t^{-1}	Multiplicative inverse of term t.
$f(t)$	Any free function.

Figure 1. Message term constructors

to be analyzed, the number of instances of each role has been determined and variables representing nonces (or session keys) have been instantiated to symbolic constants in the generating roles but not in the roles that receive these nonces.

Our semibundles have no attacker strands; instead, we write *derivation constraints* asserting that each received message is derivable, using attacker term-generation rules, from messages that were previously sent (in some node ordering consistent with the semibundle) and are thus available to the attacker. The existence of a bundle extending the semibundle is reduced to the solvability of a constraint sequence. Another way to look at a semibundle is to consider it as a *symbolic trace* representing a successful attack on the protocol. If the attacker can find a consistent instantiation for variables in this trace that makes all sent messages derivable, then the protocol is vulnerable to an attack.

An efficient method for solving a set of derivation constraints by applying rules for successive transformations of the constraint set is given in [16], but in this paper we finesse the constraint solution step by reducing it to a choice among a finite selection of substitutions. The constraint set is transformed in other ways, however.

2.1. Term algebra

For the purposes of this paper we use a simplified term algebra that includes pairing, symmetric encryption and multiplication (we describe an extension with exponentials in Section 7). For multiplication there is a unit $\mathbf{1}$ and a multiplicative inverse. We allow constants of an atomic ground message type. As before in constraint solving work, we do not distinguish between keys and other kinds of messages. Any number of free function symbols with any number of arguments are also permitted. A function f is included as a typical representative of this class. The notation for these operations is shown in Figure 1.

Multiplication forms an Abelian group, and its relations are given in Figure 2. The rules of Figure 2 are convergent modulo associativity and commutativity of $\cdot$. Therefore, every term t has a unique normal form $t\downarrow$ up to associativity and commutativity. We assume that terms are kept in normal form.

Rules for products:	Associative, commutative, and		
	$t \cdot \mathbf{1}$	$\rightarrow$	t
	$t \cdot t^{-1}$	$\rightarrow$	$\mathbf{1}$
Rules for inverses:	$(t^{-1})^{-1}$	$\rightarrow$	t
	$(t_1 \cdot t_2)^{-1}$	$\rightarrow$	${t_2}^{-1} \cdot {t_1}^{-1}$

Figure 2. Normalization rules for products and inverses

Unpairing (UL, UR)

$$\frac{T \vdash \langle u, v \rangle}{T \vdash u} \qquad \frac{T \vdash \langle u, v \rangle}{T \vdash v}$$

Decryption (D)

$$\frac{T \vdash \{u\}_v \quad T \vdash v}{T \vdash u}$$

Pairing (P)

$$\frac{T \vdash u \quad T \vdash v}{T \vdash \langle u, v \rangle}$$

Encryption (E)

$$\frac{T \vdash u \quad T \vdash v}{T \vdash \{u\}_v}$$

Function app (F)

$$\frac{T \vdash u}{T \vdash f(u)}$$

Inversion (I)

$$\frac{T \vdash u}{T \vdash u^{-1}}$$

Multiplication (M)

$$\frac{T \vdash u_1 \ \ldots \ T \vdash u_n}{T \vdash u_1 \cdot \ldots \cdot u_n}$$

Figure 3. Attacker's capabilities

2.2. Attacker model

We use the standard attacker model augmented with rules concerning products and inverses (an extension to exponentials can be found in Section 7). The attacker's ability to derive terms is characterized as a term set closure under the inference rules of Figure 3. The rules of Figure 3 without (I) and (M) are sometimes referred to as the (generalized) *Dolev-Yao* model.

2.3. Symbolic constraints

Suppose we are given a semibundle representing a security analysis problem, *i.e.*, the problem of finding whether there exists a feasible instantiation of some symbolic trace representing an attack. The issues involved in finding the right trace, such as defining a test role, choosing the number of instances of each legitimate role, and choosing constants representing nonces in the strands that originate them, are the same as in the previous research on symbolic trace reachability [16, 3, 10] and are not covered here.

We generate all possible total orderings of the nodes that are consistent with the semibundle. Each ordering yields an ordered set of derivation constraints. If $u_1, ..., u_k$ is the sequence of messages occurring in *receive* nodes only, and T_i is the set of messages *sent* in nodes prior to the node in which u_i is received, then the constraints are just the sequence

$$\mathbf{C} = \{u_i : T_i\}$$

Each individual constraint $u_i : T_i$ can be interpreted as "at step i, the attacker knows messages in T_i and needs to generate message u_i." Both u_i and messages in T_i may contain variables. We assume that T_1 contains terms that are initially known to the attacker, such as $\mathbf{1}$ and constants specific to the protocol. The properties of protocol-generated sequences are discussed in Section 3.

Say that σ is a *solution* of $u : T$ (written $\sigma \Vdash u : T$) if $T\sigma \vdash u\sigma$ is derivable using the inference rules of Figure 3. Given a constraint sequence $\mathbf{C} = \{u_1 : T_1, \ldots, u_n : T_n\}$, σ is a solution of the constraint sequence ($\sigma \Vdash \mathbf{C}$) if $\forall i\ T_i\sigma \vdash u_i\sigma$ is derivable using the rules of Figure 3.

The symbolic analysis problem is to determine, given a symbolic constraint sequence $\mathbf{C}$, whether there exists a ground substitution σ such that $\forall\ u : T \in \mathbf{C}\sigma, T\sigma \vdash u\sigma$ is derivable. We emphasize that, unlike in finite-state analysis, there are no *a priori* bounds on the size of the substitution.

2.4. Subterms and product closures

We introduce a few simple definitions. If T is a finite set of terms, let $\mathrm{St}(T)$ be the least set of terms such that:

- If $t \in T$ then $t \in \mathrm{St}(T)$
- If $\langle u, v\rangle \in \mathrm{St}(T)$ then $u, v \in \mathrm{St}(T)$
- If $\{u\}_v \in \mathrm{St}(T)$ then $u, v \in \mathrm{St}(T)$
- If $f(u) \in \mathrm{St}(T)$ then $u \in \mathrm{St}(T)$
- If $u_1 \cdot \ldots \cdot u_n \in \mathrm{St}(T)$ and $u_1, \ldots, u_n$ are not headed with $\cdot$, then $u_1, \ldots, u_n \in \mathrm{St}(T)$

Define

$$\begin{aligned}
\mathrm{PC}(T) &\overset{def}{=} \{(t_1 \cdot \ldots \cdot t_n)\downarrow\ \mid \\
&\qquad \forall i\ t_i\downarrow \text{ or } {t_i}^{-1}\downarrow \in T\} \\
\mathrm{St}(\mathbf{C}) &\overset{def}{=} \textstyle\bigcup_{u_i : T_i \in \mathbf{C}} \mathrm{St}(T_i \cup u_i) \\
\mathrm{PSt}(T) &\overset{def}{=} \mathrm{PC}(\mathrm{St}(T)) \\
\mathrm{Var}(\mathbf{C}) &\overset{def}{=} \mathrm{Var}(\mathrm{St}(\mathbf{C})) \\
\mathrm{Var}(T) &\overset{def}{=} \{x \in \mathrm{St}(T) \mid x \text{ is a variable}\} \\
\text{St–v}(\mathbf{C}) &\overset{def}{=} \mathrm{St}(\mathbf{C}) \setminus \mathrm{Var}(\mathbf{C}) \\
\text{PS–v}(\mathbf{C}) &\overset{def}{=} \mathrm{PC}(\text{St–v}(\mathbf{C}))
\end{aligned}$$

Intuitively, St–v($\mathbf{C}$) is the set of all non-variable subterms of $\mathbf{C}$, and PS–v($\mathbf{C}$) is the closure of this set under $\cdot$ and inverse.

3. Well-Defined Constraint Sequences

A protocol is *well-defined* if it does not require a participant to perform an impossible or ill-defined computation. A sequence is well-defined if it corresponds to a well-defined protocol. In this section, we posit the variable stability property which is a necessary (but not sufficient) condition that must be satisfied by every well-defined constraint sequence.

For example, we wish to rule out the following constraint sequence:

$$x \cdot y : T, \quad a : T, x$$

where, for notational convenience, we write T, x instead of $T \cup \{x\}$. This sequence corresponds to a protocol in which some honest participant receives $x \cdot y$, where both x and y are variables whose values are unknown to him, and is then expected to return x. The honest participant is thus asked to perform an ill-defined nondeterministic computation—split a product of two unknown terms.

On the other hand, the protocol corresponding to this constraint sequence:

$$x \cdot y : T, \quad \langle y, a\rangle : T, \quad a : T, x$$

is not ill-defined. Even though the honest participant receives a product of unknown variables $x \cdot y$, he later receives the term $\langle y, a\rangle$ which enables him to learn y and, multiplying $x \cdot y$ with y^{-1}, obtain x.

3.1. Monotonicity

Let $\mathbf{C} = \{u_1 : T_1, \ldots, \quad u_n : T_n\}$ be any constraint sequence generated from a protocol.

Definition 3.1 (Monotonicity) $\mathbf{C}$ *satisfies the* monotonicity property *if,* $\forall i, j \in \{1, ..., n\}$*, if* $j < i$*, then* $T_j \subseteq T_i$.

This property means that the attacker does not "forget" terms. As the protocol progresses, the set of the terms available to the attacker does not decrease. The constraint generation procedure described in Section 2.3 produces monotonic constraint sequences.

Lemma 3.2 (Stability of monotonicity) *Let* $\mathbf{C} = \{u_1 : T_1, \ldots, u_n : T_n\}$ *be a constraint sequence generated from a protocol. If* $j < i$*, then* $\forall\theta\ T_j\theta \subseteq T_i\theta$.

3.2. Variable stability

The variable stability property as defined in this section is a necessary condition: if it is not satisfied, the protocol requires an ill-defined computation. Therefore, the property is satisfied by any protocol that one may want to analyze in practice. Even though our analysis technique can only be applied to sequences that satisfy the variable stability property, this does not affect our ability to analyze any existing or proposed protocol.

3.2.1 Variable entanglement

Intuitively, variables x and y are entangled in term t if the recipient cannot use t to learn x unless he already knows y, and vice versa. For example, x and y are entangled in $x \cdot y$ and in $\{\langle y, a\rangle \cdot \{x\}_k\}_{k'}$. They are *not* entangled in $\langle\langle y, b\rangle, \langle y, a\rangle \cdot \{x\}_k\rangle$. If an honest participant receives a term with entangled variables and is then expected to send another term which uses only one of these variables, the protocol effectively requires an impossible computation. We wish to exclude such protocols from our consideration.

Definition 3.3 (Variable entanglement) *Say that variables x and y are* entangled *in term t if (i) for every occurrence of x in t, there is a superterm t' (*i.e., *$x \in \mathrm{St}(t')$) such that $u_1 \cdot \ldots t' \ldots \cdot u_n \in \mathrm{St}(t)$ and, for some $u_i \neq t'$, $y \in \mathrm{St}(u_i)$, and (ii) the symmetric condition holds for every occurrence of y.*

Proposition 3.4 *Let t be some term such that variables $x, z \in \mathrm{St}(t)$. If there exists a substitution $\sigma = [x \to t']$ such that $x, z \notin \mathrm{St}(t\sigma)$, then x and z are entangled in t.*

Intuitively, Proposition 3.4 means that if a term t contains a variable z, substitution of some *other* variable x can cause z to disappear only if x and z are entangled in t.

Proposition 3.5 (Disentanglement is not well-defined) *Consider a protocol that requires a participant to receive some term t in which variables x and y are entangled, and let $T_{xy} \subseteq \mathrm{St}(t)$ be the set of subterms in which x and y are entangled. Suppose the protocol then requires a participant, before any other term containing x (resp. y) is received, to send a term t' such that $x \in \mathrm{St}(t')$ (resp. $y \in \mathrm{St}(t')$). If there is an occurence of x (y) in t' such that $x(y) \notin \mathrm{St}(\hat{t})$ for some $\hat{t} \in T_{xy} \cap \mathrm{St}(t')$, then the protocol requires an ill-defined computation: namely, it requires a participant to split a product of two unknown values.*

For example, suppose the protocol requires a participant to receive $\langle x \cdot y, a\rangle$, in which case $T_{xy} = \{x \cdot y, \langle x \cdot y, a\rangle\}$. If the protocol then requires a participant to send $\langle x, b\rangle$, then it effectively requires him to extract x from $x \cdot y$ ("disentangle" $x \cdot y$), since x occurs outside of any subterms from T_{xy}. Splitting a product of two unknown values is an ill-defined computation.

We emphasize that the absence of entanglement is a necessary, but not sufficient condition for a protocol to be well-defined. If it is possible to learn x from t, then x is *not* entangled with an unknown y, but the reverse is not true. For example, x and y are not entangled in $t = \langle \mathit{hash}(y), \langle y, a\rangle \cdot \{x\}_k\rangle$, but x cannot be learned from t even if y is known.

3.2.2 Origination

Definition 3.6 (Origination) *Given a constraint sequence $\mathbf{C} = \{u_i : T_i\}$, define $k_x(\mathbf{C})$ for any variable $x \in \mathrm{Var}(\mathbf{C})$ to be the index of the constraint in which x appears for the first time,* i.e., *$x \in \mathrm{St}(u_{k_x} : T_{k_x})$, but $\forall i < k_x\ x \notin \mathrm{St}(u_i : T_i)$. Where $\mathbf{C}$ is clear from the context, we will refer to $k_x(\mathbf{C})$ simply as k_x.*

Variable x satisfies the variable origination property *with respect to a linear ordering $\prec$ on $\mathrm{Var}(\mathbf{C})$ if*

- $x \in \mathrm{St}(u_{k_x})$*, and*
- $\forall y \in \mathrm{Var}(T_{k_x})\ y \prec x$

$\mathbf{C}$ *satisfies the* origination property *if*

- $\exists$ *linear ordering $\prec$ on $\mathrm{Var}(\mathbf{C})$ such that $\forall x \in \mathrm{Var}(\bigcup_i T_i)$ satisfies the variable origination property with respect to $\prec$, and*
- *if $k_x < k_y$, then $x \prec y$.*

Any $\mathbf{C}$ generated from a well-defined protocol satisfies the origination property. Recall that a variable represents a term whose value can be chosen by the attacker. The origination property requires that there be no variable loops and that each variable appear for the first time in some attacker-generated term. If the protocol requires some participant to send a term involving x, *i.e.*, $x \in \mathrm{Var}(\bigcup T_i)$, he is not expected to do so *before* he receives x for the first time. Intuitively, $\prec$ represents the order in which variables appear in the protocol.

3.2.3 Necessity of variable stability

Definition 3.7 (Variable stability) *A constraint sequence $\mathbf{C}$ satisfies the* variable stability *property if, for any (partial) substitution θ, $\mathbf{C}\theta$ satisfies the origination property.*

Lemma 3.8 (Variable stability is necessary) *If $\mathbf{C}$ does not satisfy the variable stability property, then the corresponding protocol is not well-defined: it requires a participant to split a product of two unknown values.*

A proof sketch can be found in Appendix A.

Lemma 3.8 implies that the sequence generated from any well-defined protocol *must* satisfy the variable stability property. Therefore, for the remainder of this paper, we will focus on constraint sequences that satisfy Definition 3.7.

4. Ground Derivability

In this section, we outline the theory of ground term derivability in the attacker model with an Abelian group operator. This theory is based on the results of Comon-Lundh and Shmatikov that are being published concurrently [7]. In [7], however, Abelian groups are considered only in the ground case. By contrast, the rest of this paper is devoted to solving the problem in the *symbolic* case.

The normalization property stated in lemma 4.4 may appear superficially similar to that previously established for the free attacker algebra (*e.g.*,[18, 5]). Note, however, that normalization results obtained in [18, 5] are simply *false* for attacker models with non-atomic encryption keys and equational theories with cancellation, requiring development of a new proof normalization theory. Details can be found in [7].

Definition 4.1 (Ground proof) *A* proof *of $T \vdash u$ is a tree labeled with sequents $T \vdash v$ and such that:*

- *Every leaf is labeled with $T \vdash v$ such that $v \in T$*
- *Every node labeled with $T \vdash v$ has n parents $s_1, \dots, s_n$ such that $\dfrac{s_1 \;\cdots\; s_n}{T \vdash v}$ is an instance of one of the inference rules of Figure 3*
- *The root is labeled with $T \vdash u$*

The *size* of a proof is the number of its nodes.

Informally, there exists a proof of $T \vdash u$ if and only if the attacker can construct term u from the term set T using its capabilities as defined by the rules of Figure 3.

Lemma 4.2 *If there is a minimal size proof $\mathcal{P}$ of one of the following forms:*

$$\dfrac{\dfrac{\vdots}{T \vdash \langle u, v\rangle}}{T \vdash u} \qquad \dfrac{\dfrac{\vdots}{T \vdash \langle v, u\rangle}}{T \vdash u} \qquad \dfrac{\dfrac{\vdots}{T \vdash \{u\}_v} \quad \dfrac{\vdots}{T \vdash v}}{T \vdash u}$$

Then $\langle u, v\rangle \in \mathrm{St}(T)$ (resp. $\langle v, u\rangle \in \mathrm{St}(T)$, resp. $\{u\}_v \in \mathrm{St}(T)$).

Definition 4.3 (Normal ground proof) *A ground proof $\mathcal{P}$ of $T \vdash u$ is* normal *if*

- *either $u \in \mathrm{St}(T)$ and every node of $\mathcal{P}$ is labeled $T \vdash v$ with $v \in \mathrm{St}(T)$,*
- *or $\mathcal{P} = C[\mathcal{P}_1, \dots, \mathcal{P}_n]$ where every proof $\mathcal{P}_i$ is a normal proof of some $T \vdash v_i$ with $v_i \in \mathrm{St}(T)$ and context C is built using the inference rules (P),(E),(F),(M),(I) only.*

Lemma 4.4 (Existence of normal ground proof) *If there is a ground proof of $T \vdash u$, then there is a normal ground proof of $T \vdash u$.*

Proposition 4.5 *If there exists a proof of $T \vdash t$ such that $\langle u, v\rangle \in \mathrm{St}(t)$ (resp. $\langle v, u\rangle \in \mathrm{St}(t)$, resp. $\{u\}_v \in \mathrm{St}(t)$), then*

- *either $\langle u, v\rangle \in \mathrm{St}(T)$ (resp. $\langle v, u\rangle \in \mathrm{St}(T)$, resp. $\{u\}_v \in \mathrm{St}(T)$),*
- *or there exist normal proofs of $T \vdash u$ and $T \vdash v$ in which no nodes are labeled with $T \vdash \langle u, v\rangle$ (resp. $T \vdash \langle v, u\rangle$, resp. $T \vdash \{u\}_v$).*

5. Conservative Solutions

We will use the ground derivability results of Section 4 to reason about solutions of symbolic constraint sequences. Our main insight is that, assuming the symbolic constraint sequence $\mathbf{C}$ is generated from a well-defined protocol and has a solution, there exists a *conservative* solution that uses only the structure already present in $\mathbf{C}$. Even though variables may need to be instantiated, in the conservative solution all instantiations are products of subterms (and their inverses) that are already present in the original sequence $\mathbf{C}$.

To illustrate by example, consider the following constraint sequence, where x and y are variables:

$$\begin{aligned}&x : \mathbf{1}, a,\\ &\langle y, b\rangle : \mathbf{1}, a, \langle x, b\rangle,\\ &\{a\}_k : \mathbf{1}, a, \langle x, b\rangle, \{x \cdot y\}_k\end{aligned}$$

One solution of this sequence is the substitution $\sigma = [x \rightarrow a \cdot \langle a,a\rangle, y \rightarrow \langle a,a\rangle^{-1}]$. This solution, however, is not conservative, since the $\langle a,a\rangle$ term is not among the subterms of the original (uninstantiated) constraint sequence. We demonstrate that if the constraint sequence is solvable, then it also has a conservative solution—in this case, $\sigma^* = [x \rightarrow a, y \rightarrow \mathbf{1}]$. We can then reduce the protocol analysis problem to the search for conservative solutions of the corresponding constraint sequence.

Proposition 5.1
$\forall \sigma, T\ \mathrm{St}(T\sigma) \subseteq \mathrm{St}(T)\sigma \cup \bigcup_{x \in \mathrm{Var}(T)} \mathrm{St}(x\sigma)$.

Lemma 5.2 (Instantiation doesn't introduce structure) *Let $\mathbf{C}$ be a constraint sequence generated from a protocol, and let T_i be some term set such that $u_i : T_i \in \mathbf{C}$. If, for some term u, there is a minimal size normal proof $\mathcal{P}$ of one of the following forms:*

$$\dfrac{\dfrac{\vdots}{T_i\sigma \vdash \langle u,v\rangle}}{T_i\sigma \vdash u} \qquad \dfrac{\dfrac{\vdots}{T_i\sigma \vdash \langle v,u\rangle}}{T_i\sigma \vdash u}$$

$$\dfrac{\dfrac{\vdots}{T_i\sigma \vdash \{u\}_v} \qquad \dfrac{\vdots}{T_i\sigma \vdash v}}{T_i\sigma \vdash u}$$

Then $\langle u,v\rangle \in \mathrm{St}(T_i)\sigma$ (resp. $\langle v,u\rangle \in \mathrm{St}(T_i)\sigma$, resp. $\{u\}_v \in \mathrm{St}(T_i)\sigma$).

Proof: We prove the lemma for $\langle u,v\rangle$ (the proofs for $\langle v,u\rangle$ and $\{u\}_v$ are similar).

By Lemma 4.2 and Proposition 5.1, $\langle u,v\rangle \in \mathrm{St}(T_i\sigma) \subseteq \mathrm{St}(T_i)\sigma \cup \bigcup_{x\in \mathrm{Var}(T_i)} \mathrm{St}(x\sigma)$. If $\mathrm{Var}(T_i)$ is empty, the lemma follows trivially. Suppose $\mathrm{Var}(T_i)$ is not empty. Since $\mathbf{C}$ is generated from a well-defined protocol, $\mathbf{C}$ satisfies the variable stability property by Lemma 3.8. Therefore, there exists a linear ordering $\prec$ on $\mathrm{Var}(\mathbf{C})$ such that every $x \in \mathrm{Var}(T_i)$ satisfies the origination assumption with respect to $\prec$. Arrange variables $x_1, \ldots, x_n \in \mathrm{Var}(T_i)$ so that $x_1 \prec \ldots \prec x_n$.

We prove the lemma by induction over the size of $\mathrm{Var}(T_i)$. More precisely, we will show, $\forall x_k \in \mathrm{Var}(T_i)$, that if $\langle u,v\rangle \in \mathrm{St}(x_k\sigma)$, then $\langle u,v\rangle \in \mathrm{St}(T_i)\sigma \cup \mathrm{St}(x_1\sigma) \cup \ldots \cup \mathrm{St}(x_{k-1}\sigma)$.

By the origination property (Definition 3.6), $\exists\ j < i$ such that $x_k \in \mathrm{St}(u_j)$ and $\forall y \in \mathrm{Var}(T_j)\ y \prec x_k$. Observe that $\mathrm{St}(T_j\sigma) = \mathrm{St}(T_j)\sigma \cup \bigcup_{y\in\mathrm{Var}(T_j)} \mathrm{St}(y\sigma)$. By the monotonicity property (Definition 3.1), $T_j \subseteq T_i$ (in fact, since $x_k \in \mathrm{Var}(T_i)$ and $x_k \notin \mathrm{Var}(T_j)$, $T_j \subset T_i$). Therefore, $\mathrm{St}(T_j)\sigma \subseteq \mathrm{St}(T_i)\sigma$ and $\mathrm{Var}(T_j) \subset \mathrm{Var}(T_i)$. Since $\forall y \in \mathrm{Var}(T_j)\ y \prec x_k$, it must be that $y = x_l \in \mathrm{Var}(T_i)$ where $l < k$. Therefore, $\forall\ y \in \mathrm{Var}(T_j)\ \mathrm{St}(y\sigma) \subseteq \mathrm{St}(x_1\sigma) \cup \ldots \cup \mathrm{St}(x_{k-1}\sigma)$. We conclude that $\mathrm{St}(T_j\sigma) \subseteq \mathrm{St}(T_i)\sigma \cup \mathrm{St}(x_1\sigma) \cup \ldots \cup \mathrm{St}(x_{k-1}\sigma)$. To complete the induction step, it is now sufficient to show that $\langle u,v\rangle \in \mathrm{St}(T_j)\sigma$.

Since $x_k \in \mathrm{St}(u_j)$ (by origination) and $\langle u,v\rangle \in \mathrm{St}(x_k\sigma)$ (by assumption), $\langle u,v\rangle \in \mathrm{St}(u_j\sigma)$. We apply Proposition 4.5 to the proof of $T_j\sigma \vdash u_j\sigma$. If $\langle u,v\rangle \in \mathrm{St}(T_j\sigma)$, we are done.

Now suppose $\langle u,v\rangle \notin \mathrm{St}(T_j\sigma)$. By Proposition 4.5, there must exist a proof of $T_j\sigma \vdash u$ such that no node is labeled with $T \vdash \langle u,v\rangle$. Since $T_j\sigma \subset T_i\sigma$, this proof can also serve as the proof of $T_i\sigma \vdash u$. But this contradicts the assumption that the minimal size normal proof of $T_i\sigma \vdash u$ relies on $T_i\sigma \vdash \langle u,v\rangle$. We conclude that $\langle u,v\rangle \in \mathrm{St}(T_j\sigma)$. $\square$

Proposition 5.3 *If $t = \langle u,v\rangle \in$ PS–v$(\mathbf{C})\sigma$ (resp. $\{u\}_v \in$ PS–v$(\mathbf{C})\sigma$, $f(u) \in$ PS–v$(\mathbf{C})\sigma$), then $\exists t' = \langle u',v'\rangle \in \mathrm{St}(\mathbf{C})$ (resp. $\{u'\}_{v'} \in \mathrm{St}(\mathbf{C}), f(u') \in \mathrm{St}(\mathbf{C})$) such that $t'\sigma = t$.*

Proposition 5.4 *Suppose $\mathcal{P}$ is a proof as defined by 4.1, and let $\dfrac{T\vdash v_1 \quad [T \vdash v_2]}{T \vdash v_3}$ be an inference in $\mathcal{P}$ which is an instance of some rule other than (M) or (I) (the $T \vdash v_2$ premise may be absent). For $i \in \{1,2,3\}$, if $v_i \in$ PS–v$(\mathbf{C})\sigma$ for some σ, then $v_i \in \mathrm{St}(\mathbf{C})\sigma$.*

Definition 5.5 (Conservative substitution) *Substitution σ is* conservative *if*

$$\forall x \in \mathrm{Var}(\mathbf{C})\ \mathrm{St}(x\sigma) \subseteq \text{PS–v}(\mathbf{C})\sigma$$

Intuitively, a conservative substitution does not introduce any structure that was not already present in the original symbolic sequence $\mathbf{C}$. We will prove that if there exists some solution $\sigma \Vdash \mathbf{C}$, then there exists a conservative solution $\sigma^* \Vdash \mathbf{C}$.

Define transformation $\nu_{\mathbf{C}}$ on normalized ground terms as follows:

$$
\begin{array}{lcl}
\nu_{\mathbf{C}}(c) & = & c \text{ if } c \in \text{PS–v}(\mathbf{C})\sigma, \\
 & & \mathbf{1} \text{ otherwise} \\
\nu_{\mathbf{C}}(\langle u, v\rangle) & = & \langle \nu_{\mathbf{C}}(u), \nu_{\mathbf{C}}(v)\rangle \\
 & & \quad \text{if } \langle u, v\rangle \in \text{PS–v}(\mathbf{C})\sigma, \\
 & & \mathbf{1} \text{ otherwise} \\
\nu_{\mathbf{C}}(\{u\}_v) & = & \{\nu_{\mathbf{C}}(u)\}_{\nu_{\mathbf{C}}(v)} \\
 & & \quad \text{if } \{u\}_v \in \text{PS–v}(\mathbf{C})\sigma, \\
 & & \mathbf{1} \text{ otherwise} \\
\nu_{\mathbf{C}}(u_1 \cdot \ldots \cdot u_k) & = & \nu_{\mathbf{C}}(u_1) \cdot \ldots \cdot \nu_{\mathbf{C}}(u_k) \downarrow \\
 & & (k > 1, \text{no } u_i \text{ is headed with } \cdot) \\
\nu_{\mathbf{C}}(u^{-1}) & = & \nu_{\mathbf{C}}(u)^{-1} \downarrow
\end{array}
$$

Then define substitution σ^* as $x\sigma^* = \nu_{\mathbf{C}}(x\sigma)$. Essentially, $x\sigma^*$ is the same as $x\sigma$, except that all subterms of $x\sigma$ that are not in PS–v$(\mathbf{C})\sigma$ have been eliminated by the transformation $\nu_{\mathbf{C}}$.

Proposition 5.6 *If $v \in$ Var$(\mathbf{C})$ or $v \in$ PS–v$(\mathbf{C})$, then $v\sigma^* = \nu_{\mathbf{C}}(v\sigma)$.*

Proposition 5.7 *σ^* is a conservative substitution.*

Proposition 5.8 *Given $u : T \in \mathbf{C}$ and some ground term t, if there exists a proof of $T\sigma \vdash t$, then there exists a proof of $T\sigma^* \vdash \nu_{\mathbf{C}}(t)$.*

Proof: Let $\mathcal{P}$ be the normal proof of $T\sigma \vdash t$ using the inference rules of Figure 3. Such a proof exists by Lemma 4.4. We prove the proposition by induction over the structure of $\mathcal{P}$.

For the induction basis, suppose that $\mathcal{P}$ consists of a single leaf node $T\sigma \vdash t$ such that $t \in T\sigma$. This implies that $\exists v \in T$ s.t. $t = v\sigma$. Either $v \in$ Var$(\mathbf{C})$, or $v \in T\setminus$Var$(\mathbf{C}) \subseteq$ PS–v$(\mathbf{C})$. In either case, by Lemma 5.6, $v\sigma^* = \nu_{\mathbf{C}}(t)$. Therefore, $\nu_{\mathbf{C}}(t) \in T\sigma^*$, and the proof of $T\sigma^* \vdash \nu_{\mathbf{C}}(t)$ consists of a single node $T\sigma^* \vdash \nu_{\mathbf{C}}(t)$.

Consider one inference of $\mathcal{P}$ of the form $\dfrac{T\sigma \vdash t_1 \quad [\ldots\ T\sigma \vdash t_k]}{T\sigma \vdash t}$ (the $T\sigma \vdash t_i$ premises for $i > 1$ may be absent) and assume as the induction hypothesis that $\forall i$ there exist proofs $\mathcal{P}_i$ of $T\sigma^* \vdash \nu_{\mathbf{C}}(t_i)$. To complete the induction, it is sufficient to show that for any inference rule, there exists a proof of $T\sigma^* \vdash \nu_{\mathbf{C}}(t)$.

If the rule is (D), then $t_1 = \{t\}_{t_k}, t_2 = t_k$ for some term t_k. By Lemma 5.2, $\{t\}_{t_k} \in$ St$(T)\sigma$, *i.e.*, $\exists\ \{v_1\}_{v_2} \in T$ such that $v_1\sigma = t, v_2\sigma = t_k$. Since $\{v_1\}_{v_2} \notin$ Var(T), we obtain that $\{v_1\}_{v_2} \in$ St–v$(T) \subseteq$ PS–v$(\mathbf{C})$. Therefore, $\{t\}_{t_k} \in$ PS–v$(\mathbf{C})\sigma$, thus $\nu_{\mathbf{C}}(\{t\}_{t_k}) = \{\nu_{\mathbf{C}}(t)\}_{\nu_{\mathbf{C}}(t_k)}$. By the induction hypothesis, $\mathcal{P}_1$ is the proof of $T\sigma^* \vdash \nu_{\mathbf{C}}(\{t\}_{t_k})$ and $\mathcal{P}_2$ is the proof of $T\sigma^* \vdash \nu_{\mathbf{C}}(t_k)$. The proof of $T\sigma^* \vdash \nu_{\mathbf{C}}(t)$ is then constructed as follows:

$$\dfrac{\begin{array}{c}\mathcal{P}_1 \\ T\sigma^* \vdash \{\nu_{\mathbf{C}}(t)\}_{\nu_{\mathbf{C}}(t_k)}\end{array} \qquad \begin{array}{c}\mathcal{P}_2 \\ T\sigma^* \vdash \nu_{\mathbf{C}}(t_k)\end{array}}{T\sigma^* \vdash \nu_{\mathbf{C}}(t)}$$

A similar argument applies for rules (UL),(UR).

If the rule is (P), then $t = \langle t_1, t_2\rangle$. According to the definition of $\nu_{\mathbf{C}}$, there are two possibilities. If $\nu_{\mathbf{C}}(\langle t_1, t_2\rangle) = \mathbf{1}$, then $\nu_{\mathbf{C}}(\langle t_1, t_2\rangle) \in T\sigma^*$, and the proof of $T\sigma^* \vdash \nu_{\mathbf{C}}(\langle t_1, t_2\rangle)$ consists of one node $T\sigma^* \vdash \mathbf{1}$. If $\nu_{\mathbf{C}}(\langle t_1, t_2\rangle) = \langle \nu_{\mathbf{C}}(t_1), \nu_{\mathbf{C}}(t_2)\rangle$, the proof of $T\sigma^* \vdash \nu_{\mathbf{C}}(\langle t_1, t_2\rangle)$ is constructed as follows:

$$\dfrac{\begin{array}{c}\mathcal{P}_1 \\ T\sigma^* \vdash \nu_{\mathbf{C}}(t_1)\end{array} \qquad \begin{array}{c}\mathcal{P}_2 \\ T\sigma^* \vdash \nu_{\mathbf{C}}(t_2)\end{array}}{T\sigma^* \vdash \nu_{\mathbf{C}}(\langle t_1, t_2\rangle)}$$

A similar argument applies for rules (E),(F),(M),(I). □

Theorem 1 (Existence of conservative solution) *If there exists a solution $\sigma \Vdash \mathbf{C}$, then there exists a conservative solution $\sigma^* \Vdash \mathbf{C}$.*

Proof: Let $\sigma \Vdash \mathbf{C}$ be a solution of $\mathbf{C}$. Define substitution σ^* as $\forall x \in$ Var$(\mathbf{C})$ $x\sigma^* = \nu_{\mathbf{C}}(x\sigma)$. By Proposition 5.7, σ^* is a conservative substitution. To show that $\sigma^* \Vdash \mathbf{C}$, consider any constraint $u : T \in \mathbf{C}$. Since $\sigma \Vdash \mathbf{C}$, there exists a proof of $T\sigma \vdash u\sigma$. Either $u \in$ Var$(\mathbf{C})$, or $u \in$ St$(\mathbf{C}) \setminus$ Var$(\mathbf{C}) \subseteq$ PS–v$(\mathbf{C})$. In either case, by Proposition 5.6, $u\sigma^* = \nu_{\mathbf{C}}(u\sigma)$. By Proposition 5.8, there exists a proof of $T\sigma^* \vdash \nu_{\mathbf{C}}(u\sigma) = u\sigma^*$. Therefore, $\sigma^* \Vdash \mathbf{C}$. □

Proposition 5.9 *If σ is a conservative substitution, then* St$(\mathbf{C})\sigma \subseteq$ PS–v$(\mathbf{C})\sigma$.

Proof: First, observe that St$(\mathbf{C})\sigma \subseteq$ St–v$(\mathbf{C})\sigma \cup \bigcup_{x\in \text{Var}(\mathbf{C})}$ St$(x\sigma)$. By definition of St–v$(\mathbf{C})$, St–v$(\mathbf{C})\sigma \subseteq$ PS–v$(\mathbf{C})\sigma$. By Definition 5.5, $\forall x \in$ Var$(\mathbf{C})$ St$(x\sigma) \subseteq$ PS–v$(\mathbf{C})\sigma$. Therefore, St$(\mathbf{C})\sigma \subseteq$ PS–v$(\mathbf{C})\sigma$. □

Definition 5.10 (Conservative proof) *Given $u : T \in \mathbf{C}$ and a substitution σ, a proof $\mathcal{P}$ of $T\sigma \vdash u\sigma$ is* conservative *if, for every node of $\mathcal{P}$ labeled $T\sigma \vdash v$,*

- *either $v \in$ St$(\mathbf{C})\sigma$, or*
- *node $T\sigma \vdash v$ is obtained by (M) or (I) inference rule* and *is only used as a premise of an (M) or (I) rule.*

Lemma 5.11 (Existence of conservative proof) *If $\sigma^* \Vdash \mathbf{C} = \{u_i : T_i\}$ is a conservative solution, then $\forall i$ there exists a conservative proof of $T_i\sigma^* \vdash u_i\sigma^*$.*

Proof: Let σ^* be the conservative solution of $\mathbf{C}$. Consider any constraint $u_i : T_i$. Since $\sigma^* \Vdash \mathbf{C}$, by Lemma 4.4 there exists a normal proof $\mathcal{P}_i$ of $T_i\sigma^* \vdash u_i\sigma^*$. Let $\hat{T}_i = T_i \cup \{u_i\}$, and let $V_i = \mathrm{Var}(\hat{T}_i) \subseteq \mathrm{Var}(\mathbf{C})$. From Definition 4.3 of normal proofs, it follows that every node of $\mathcal{P}_i$ is labeled $T_i\sigma^* \vdash v$ where $v \in \mathrm{St}(\hat{T}_i\sigma^*)$. Since $\hat{T}_i \subseteq \mathrm{St}(\mathbf{C})$, by Proposition 5.9 $v \in$ PS–v$(\mathbf{C})\sigma^*$.

Any inference in $\mathcal{P}_i$ other than (M) or (I) must have the form $\dfrac{T \vdash v_1 \quad [T \vdash v_2]}{T \vdash v_3}$. Since $v_{1,2,3} \in$ PS–v$(\mathbf{C})\sigma^*$, by Proposition 5.4 $v_{1,2,3} \in \mathrm{St}(\mathbf{C})\sigma^*$. □

6. Constraint Solving Procedure

In this section, we describe a finite procedure that reduces any constraint sequence $\mathbf{C}$ generated from a protocol to a system of quadratic Diophantine equations which is solvable if and only if $\mathbf{C}$ has a solution. The symbolic trace reachability problem is thus reduced to the solvability of a particular system of equations.

If $\mathbf{C}$ has a solution, then, by Theorem 1, there exists a conservative solution. For the remainder of this section, we pick a particular conservative solution $\sigma \Vdash \mathbf{C}$ (if it exists). We then transform $\mathbf{C}$ successively to $\mathbf{C}_1$, in which subterms that are merged by σ are unified; then to $\mathbf{C}_2$, in which constraints for derivable non-product subterms have been added; then to $\mathbf{C}_3$, in which all but product constraints have been removed; and finally a system of Diophantine equations is generated from $\mathbf{C}_3$.

Running example. We will use the following symbolic trace as an (artificial) running example to illustrate our constraint solving procedure. An event $A \longrightarrow t$ is a $+t$ node in an A-role strand, *etc.*

1. A	$\longrightarrow$	$a \cdot b$		4. B	$\longleftarrow$	$\{Y\}_b$
2. B	$\longleftarrow$	$a \cdot X \cdot Y$		5. B	$\longrightarrow$	$b \cdot X$
3. A	$\longrightarrow$	$\{a\}_b$		6. A	$\longleftarrow$	a^6

Recall that the goal of symbolic protocol analysis is to determine whether this trace is *feasible*, *i.e.*, whether there exists an instantiation of variables X and Y such that every term sent from the network and received by an honest participant (*i.e.*, every term of the form $P \longleftarrow$) is derivable using the rules of Figure 3. This is equivalent to deciding whether the corresponding symbolic constraint sequence $\mathbf{C}$ has a solution:

$$\begin{array}{rcl} a \cdot X \cdot Y & : & a \cdot b \\ \{Y\}_b & : & a \cdot b, \{a\}_b \\ a^6 & : & a \cdot b, \{a\}_b, b \cdot X \end{array}$$

6.1. Determine subterm equalities

In the first step, we guess (*i.e.*, find by exhaustive enumeration of finitely many possibilities) the equivalence relation on $\mathrm{St}(\mathbf{C})$ induced by substitution σ. More precisely, $\forall s_i, s_j \in \mathrm{St}(\mathbf{C})$, we guess whether $s_i\sigma = s_j\sigma$ or not. Since $\mathrm{St}(\mathbf{C})$ is finite, there are only a finite number of possible equivalence relations to consider. Each equivalence relation represents a set of unification problems in an Abelian group, which are decidable [2]. There are finitely many most general unifiers consistent with any given equivalence relation.

By exhaustive enumeration of all possible equivalence relations and all possible most general unifiers for each, we discover a partial substitution θ consistent with σ, *i.e.*, $s_i\theta = s_j\theta$ if and only if $s_i\sigma = s_j\sigma$. Let $\mathbf{C}_1 = \mathbf{C}\theta$.

Proposition 6.1 $\forall s, s' \in \mathrm{St}(\mathbf{C}_1)$ *if $s \neq s'$, then $s\sigma \neq s'\sigma$.*

Lemma 6.2 $\sigma \Vdash \mathbf{C}$ *if and only if* $\sigma \Vdash \mathbf{C}_1$.

Proof: Follows directly from the choice of θ such that $\sigma = \theta \circ \sigma'$. □

Running example. In our running example, we guess that the only subterm equality is $\{Y\}_b = \{a\}_b$, giving us partial substitution $[Y \rightarrow a]$ and producing the following $\mathbf{C}_1$:

$$\begin{array}{l} a^2 \cdot X : a \cdot b \\ \{a\}_b : a \cdot b, \{a\}_b \\ a^6 : a \cdot b, \{a\}_b, b \cdot X \end{array}$$

6.2. Determine order of subterm derivation

In the second step, we determine which subterms of $\mathbf{C}_1\sigma$ can be computed by the attacker and the order in which they are computed.

1. Guess $S_\vdash = \{s \in \mathrm{St}(\mathbf{C}_1) \mid \exists u_i : T_i \in \mathbf{C}_1$ s.t. there exists a proof of $T_i\sigma \vdash s\sigma\}$, *i.e.*, $S_\vdash$ is the set of subterms that are derivable from *some* term set available to the attacker. This terminates, since there are only finitely many subsets of $\mathrm{St}(\mathbf{C}_1)$ to consider.

2. $\forall s \in S_\vdash$ guess $j_s \in \{1, ..., n\}$ such that there exists a proof of $T_{j_s}\sigma \vdash s\sigma$, but there is no proof of $T_{j_s-1}\sigma \vdash s\sigma$. In other words, j_s is the index of the first constraint in $\mathbf{C}_1$ from whose term set s can be constructed. This

terminates, since $S_\vdash$ is finite, and, for each member of $S_\vdash$, there are only finitely many constraints in $\mathbf{C}_1$ to consider.

3. By definition of the normal proof, for any solution $\sigma \Vdash \mathbf{C}_1$, any term set T such that $u : T \in \mathbf{C}_1$, any $s, s' \in S_\vdash$, if the minimal size normal proof of $T\sigma \vdash s\sigma$ contains a node labeled $T\sigma \vdash s'\sigma$, then the minimal size normal proof of $T\sigma \vdash s'\sigma$ does not contain a node labeled $T\sigma \vdash s\sigma$. Therefore, σ is consistent with at least one linear ordering $\prec$ on $S_\vdash$ that satisfies the following property:

   - If $s \prec s'$, then the normal proof of $T\sigma \vdash s\sigma$ does not contain any node labeled with $T\sigma \vdash s'\sigma$.

   Of all possible orderings on $S_\vdash$ that satisfy this property, we pick one that also satisfies

   - If $j_s < j_{s'}$, then $s \prec s'$.

   This is possible since $j_s < j_{s'}$ means that there does not exist a proof of $T_{j_s}\sigma \vdash s'\sigma$. Therefore, the proof of $T_{j_s}\sigma \vdash s\sigma$ cannot have a node labeled $T_{j_s}\sigma \vdash s'\sigma$.

   Intuitively, $\prec$ is the order in which members of $S_\vdash$ are derived. Since there are only finitely many possible linear orderings on $S_\vdash$ to consider, we find $\prec$ by exhaustive enumeration.

4. We arrange $s_1, \dots, s_k \in S_\vdash$ according to the ordering $\prec$, and insert each s_i in the constraint sequence immediately before the $u_{s_i} : T_{s_i}$ constraint. More precisely, we replace $\mathbf{C}_1$ with

   $$\begin{array}{l} u_1 : T_1, \quad \dots \quad u_{j_{s_1}-1} : T_{j_{s_1}-1}, \\ s_1 : T_{j_{s_1}}, \\ u_{j_{s_1}} : T_{j_{s_1}}, s_1, \quad \dots \quad u_n : T_n, s_1 \end{array}$$

   Call the resulting sequence $\mathbf{C}_1^{(1)}$, and repeat this step for $s_2, \dots, s_k \in S_\vdash$. Given $\mathbf{C}_1^{(i-1)}$, $\mathbf{C}_1^{(i)}$ is constructed by inserting $s_i : T$ immediately before $u_{j_{s_i}} : T \in \mathbf{C}_1^{(i-1)}$, and adding s_i to the term sets of all subsequent constraints.

   Let $\mathbf{C}_2 = \mathbf{C}_1^{(k)}$.

Proposition 6.3 $\forall s, s' \in \mathrm{St}(\mathbf{C}_2)$ *if* $s \neq s'$, *then* $s\sigma \neq s'\sigma$.

Proposition 6.4 $\forall u : T \in \mathbf{C}_2\ \forall s \in S_\vdash$ *such that* $s' \prec u$, $s \in T$.

Lemma 6.5 $\sigma \Vdash \mathbf{C}_1$ *if and only if* $\sigma \Vdash \mathbf{C}_2$.

Proof: Follows directly from construction of $\mathbf{C}_2$ since only constraints solved by σ are added to $\mathbf{C}_1$ to obtain $\mathbf{C}_2$. □

Running example. In our running example, we guess that no subterms (other than those already appearing as target terms) are derivable, and, therefore, $\mathbf{C}_2 = \mathbf{C}_1$.

6.3. Eliminate all inferences other than (M) or (I)

In this step, we eliminate all constraints other than those involving application of rules (M) and (I) only.

Lemma 6.6 *Consider any* $u : T \in \mathbf{C}_2$ *and the last inference of the proof of* $T\sigma \vdash u\sigma$.

- *If* $u\sigma \in T\sigma$, *then* $u \in T$.
- *If* $u\sigma$ *is obtained by (UL), then* $\langle u, t'\rangle \in T$ *for some term* t'.
- *If* $u\sigma$ *is obtained by (UR), then* $\langle t', u\rangle \in T$ *for some term* t'.
- *If* $u\sigma$ *is obtained by (D), then* $\{u\}_{t'} \in T$ *for some term* t'.
- *If* $u\sigma$ *is obtained by (P), then* $u = \langle u_1, u_2\rangle$ *and* $u_{1,2} \in T$ *for some terms* $u_{1,2}$.
- *If* $u\sigma$ *is obtained by (E), then* $u = \{u_1\}_{u_2}$ *and* $u_{1,2} \in T$ *for some terms* $u_{1,2}$.
- *If* $u\sigma$ *is obtained by (F), then* $u = f(t')$ *and* $t' \in T$ *for some term* t'.

Proof: Consider any $u : T \in \mathbf{C}_2$. Since $\sigma \Vdash \mathbf{C}_2$ is a conservative solution, by Lemma 5.11, there exists a conservative proof of $T\sigma \vdash u\sigma$.

If $u\sigma \in T\sigma$, then $\exists t \in T \subseteq \mathrm{St}(\mathbf{C}_2)$ s.t. $t\sigma = u\sigma$. From Proposition 6.3, it follows that $u = t \in T$. We conclude that, whenever $u\sigma \in T\sigma$, $u : T \in \mathbf{C}_2$ is such that $u \in T$.

If $u\sigma \notin T\sigma$, consider the *last* inference of the conservative proof of $T\sigma \vdash u\sigma$. It must have the form $\dfrac{T\sigma \vdash v_1 \quad [\dots T\sigma \vdash v_k]}{T\sigma \vdash u\sigma}$ (the $T\sigma \vdash v_i$ premises for $i > 1$ may be absent). If this inference is an instance of any rule other than (M) or (I), then $i \leq 2$ and, by Definition 5.10, $v_{1,2} \in \mathrm{St}(\mathbf{C}_2)\sigma$, *i.e.*, $\exists v'_{1,2} \in \mathrm{St}(\mathbf{C}_2)$ such that $v'_1\sigma = v_1$ and $v'_2\sigma = v_2$.

Since there exist proofs of $T\sigma \vdash v'_{1,2}\sigma$, it must be that $v'_{1,2} \in S_\vdash$. Since the proof of $T\sigma \vdash u\sigma$ contains nodes labeled $T\sigma \vdash v'_{1,2}\sigma$, by definition of ordering $\prec$ it must be that $v'_{1,2} \prec u$. By Proposition 6.4, $v'_{1,2} \in T$. We conclude that the proof of $T\sigma \vdash u\sigma$ consists of one inference:

$$\frac{T\sigma \vdash v'_1\sigma \in T\sigma \quad [T\sigma \vdash v'_2\sigma \in T\sigma]}{T\sigma \vdash u\sigma}$$

Consider all possible cases for this inference rule other than (M) or (I).

If the rule is (UL), then $v_1'\sigma = \langle u\sigma, t\rangle$ for some term t. By Proposition 5.9, $v_1'\sigma \in$ PS–v$(\mathbf{C})\sigma$. By Proposition 5.3, this means that $\exists\ \langle u', t'\rangle \in \mathrm{St}(\mathbf{C}_2)$ such that $\langle u', t'\rangle\sigma = v_1'\sigma$. By Proposition 6.3, this implies that $v_1' = \langle u', t'\rangle$. Since $u', u \in \mathrm{St}(\mathbf{C}_2)$, Proposition 6.3 also implies that $u' = u$. Therefore, $v_1' = \langle u, t'\rangle \in T$. We conclude that, whenever $u\sigma$ is obtained by (UL) rule, $u : T \in \mathbf{C}_2$ is such that $\langle u, t'\rangle \in T$. The proofs for (UR) and (D) is similar.

If the rule is (E), then $u\sigma = \{v_1\sigma\}_{v_2\sigma}$. By Proposition 5.9, $u\sigma \in$ PS–v$(\mathbf{C})\sigma$. By Proposition 5.3, $\exists u' = \{v_1'\}_{v_2'} \in \mathrm{St}(\mathbf{C}_2)$ and $u\sigma = u'\sigma$. Since $u \in \mathrm{St}(\mathbf{C}_2)$ by Definition 5.10, Proposition 6.3 implies $u' = u$. We conclude that, whenever $u\sigma$ is obtained by (E) rule, $u = \{u_1\}_{u_2}$ and $\{u_1\}_{u_2} : T \in \mathbf{C}_2$ where $u_{1,2} \in T$. The proofs for (P) and (F) are similar. □

Lemma 6.6 implies that we can discover all $u : T \in \mathbf{C}_2$ such that $u\sigma \in T\sigma$ or $u\sigma$ is obtained by (UL),(UR),(D),(P),(E) or (F) rule by syntactic inspection of $\mathbf{C}_2$.

Let $\mathbf{C}_3$ be the constraint sequence obtained from $\mathbf{C}_2$ by eliminating all such $u : T$.

Lemma 6.7 *$\sigma \Vdash \mathbf{C}_3$ if and only if $\sigma \Vdash \mathbf{C}_2$.*

Proposition 6.8 *$\forall u : T \in \mathbf{C}_3$, proof of $T\sigma \vdash u\sigma$ uses (M) and (I) rules only.*

Running example. In our example, we guess the first and third constraints were obtained by application of rules (M) and (I) only. We eliminate the second constraint, obtaining the following $\mathbf{C}_3$:

$$\begin{array}{rcl} a^2 \cdot X & : & a \cdot b \\ a^6 & : & a \cdot b, \{a\}_b, b \cdot X \end{array}$$

6.4. Solve system of quadratic equations in integers

In the last step of the constraint solving procedure, we convert the constraint sequence $\mathbf{C}_3$ into a system of quadratic Diophantine equations which is solvable if and only if $\exists\ \sigma \Vdash \mathbf{C}_3$, and, by lemmas 6.2, 6.5 and 6.7, $\sigma \Vdash \mathbf{C}$.

Recall that $\mathrm{Var}(\mathbf{C}_3) = \{x_1, \dots, x_L\}$ is the set of all variables that appear in $\mathbf{C}_3$. Let St–vp$(\mathbf{C}_3)$ = St–v$(\mathbf{C}_3) \setminus \{u \in$ St–v$(\mathbf{C}_3) \mid u = u_1 \cdot \ldots \cdot u_n, n > 1\}$ be the (finite) set of all non-variable, non-product subterms of $\mathbf{C}_3$, and suppose St–vp$(\mathbf{C}_3) = \{t_1, \dots, t_M\}$.

Since we are only considering conservative solutions, by Definition 5.5, $\forall j \in \{1, ..., L\}\ x_j\sigma \in$ PS–v$(\mathbf{C}_3)\sigma$. Since PS–v$(\mathbf{C}_3)$ is closed under $\cdot$, PS–v$(\mathbf{C}_3)$ = PC(St–v$(\mathbf{C}_3)$) = PC(St–vp$(\mathbf{C}_3)$). For brevity, let S = St–vp$(\mathbf{C}_3)$. Any possible value for a variable x_j is a product (with possible inverses) of instances of elements of S.

To derive the Diophantine equations, we write products and inverses using integer exponents. For example, $a \cdot a \cdot b^{-1} = a^2 b^{-1}$. Thus, for some integer values $y[j,k]$,

$$x_j\sigma = \Pi_{t_k \in S}(t_k\sigma)^{y[j,k]} \quad 1 \le j \le L, 1 \le k \le M$$

If the substitution is well-defined, then $x\sigma$ cannot depend on the instances of terms that include x or variables that depend on x. Therefore, any solution is consistent with some linear ordering $\prec$ on $\mathrm{Var}(\mathbf{C})$ so that if $x \prec x'$, then $x\sigma = \Pi(t_k\sigma)^{y[j,k]}$ such that $\forall t_k\ x' \notin \mathrm{St}(t_k)$. We guess this ordering by exhaustive enumeration, and set $y[j,k] = 0$ if $\exists x' \in \mathrm{Var}(t_k)\ x_j \preceq x'$. This ensures that any substitution we compute is well-defined.

For any term t, define $\mathbf{F}(t)$ to be the set of all top-level factors of t, *i.e.*, $\mathbf{F}(t) = \{t_1^{s_1}, \dots, t_n^{s_n}\}$ if $t = t_1^{s_1} \cdot \ldots \cdot t_n^{s_n}$ and none of t_i are headed with $\cdot$. For example, $\mathbf{F}(a^{-2} \cdot b^3) = \{a^{-2}, b^3\}$.

Now consider any constraint $u_i : T_i \in \mathbf{C}_3$, and let $U_i = \mathbf{F}(u_i)$. By Proposition 6.8, the proof of $T_i\sigma \vdash u_i\sigma$ contains applications of (M) and (I) rules only. Therefore, for some integer values $z[i,s]$

$$\Pi_{u \in U_i} u\sigma = \Pi_{t \in T_i}(t\sigma)^{z[i,s]} \quad 1 \le s \le |T_i|$$

Replacing each $x_j\sigma \in U_i\sigma \cup T_i\sigma$ with $\Pi_{t_k \in S}(t_k\sigma)^{y[j,k]}$, we obtain

$$\begin{array}{l} \Pi_{x_j^{q_j} \in U_i}(\Pi_{t_k \in S}(t_k\sigma)^{y[j,k]})^{q_j} \cdot \Pi_{t_k^{q_k} \in U_i}(t_k\sigma)^{q_k} = \\ \Pi_{x_j^{r_j} \in \mathbf{F}(t_s \in T_i)}(\Pi_{t_k \in S}(t_k\sigma)^{y[j,k]})^{r_j \cdot z[i,s]} \ \cdot \\ \Pi_{t_{s'}^{r_k} \in T_i \setminus \mathrm{Var}(T_i)}(t\sigma)^{r_k \cdot z[i,s']} \\ \text{where } 1 \le j \le L, 1 \le k \le M, 1 \le s, s' \le |T_i| \end{array}$$

Since S = St–vp$(\mathbf{C}_3) \subseteq \mathrm{St}(\mathbf{C}_3) \subseteq \mathrm{St}(\mathbf{C}_2)$, by Proposition 6.3 $\forall t, t' \in S$ if $t \neq t'$, then $t\sigma \neq t'\sigma$. Therefore, the following equations must hold for some integer values $y[j,k], z[i,s], z[i,s']$:

$$\begin{array}{l} \forall\ u_i : T_i \in \mathbf{C}_3, \quad \forall\ x_j \in \mathrm{Var}(\mathbf{C}_3), \\ \forall\ t_k \in \mathrm{St}(\mathbf{C}_3) \text{ without variables and products} \\ \underbrace{q_j \cdot y[j,k]}_{\text{if } x_j^{q_j} \in U_i} \quad + \quad \underbrace{q_k}_{\text{if } t_k^{q_k} \in U_i} \quad = \\ \Sigma_s \underbrace{r_j \cdot y[j,k] \cdot z[i,s]}_{\text{if } x_j^{r_j} \in \mathbf{F}(t_s),\ t_s \in T_i} \quad + \\ \Sigma_{s'} \underbrace{r_k \cdot z[i,s']}_{\text{if } t_k^{r_k} \in \mathbf{F}(t_{s'}),\ t_{s'} \in T_i} \end{array} \qquad (1)$$

where $y[j,k], z[i,s], z[i,s']$ are variables;
q_j, q_k, r_j, r_k are integer constants;
$y[j,k] = 0$ if $\exists y \in \mathrm{Var}(t_k)$ s.t. $x_j \preceq y$

The total number of equations of this form is equal to $N \times L \times M$ where N is the number of constraints in $\mathbf{C}_3$, L is the number of variables, and M is the number of subterms other than variables and products.

Running example. Recall that, in our running example, we are solving the following $\mathbf{C}_3$:

$$\begin{array}{rcl} a^2 \cdot X & : & a \cdot b \\ a^6 & : & a \cdot b, \{a\}_b, b \cdot X \end{array}$$

Since St–vp($\mathbf{C}_3$) $=$ $\{a, b, \{a\}_b\}$, we replace X with $a^{y_1} \cdot b^{y_2} \cdot (\{a\}_b)^{y_3}$ and obtain the following equations, which must be solvable for some integer values $y_1, y_2, y_3, z_1, z_{21}, z_{22}, z_{23}$:

$$\begin{array}{rcl} a^2 \cdot a^{y_1} \cdot b^{y_2} \cdot (\{a\}_b)^{y_3} & = & (a \cdot b)^{z_1} \\ a^6 & = & (a \cdot b)^{z_{21}} \cdot (\{a\}_b)^{z_{22}} \cdot \\ & & (b \cdot (a^{y_1} \cdot b^{y_2} \cdot (\{a\}_b)^{y_3}))^{z_{23}} \end{array}$$

which is equivalent to the following system of quadratic Diophantine equations:

$$\begin{array}{rclrcl} 2 + y_1 & = & z_1 & 6 & = & y_1 \cdot z_{23} + z_{21} \\ y_2 & = & z_1 & 0 & = & y_2 \cdot z_{23} + z_{21} + z_{23} \\ y_3 & = & 0 & 0 & = & y_3 \cdot z_{23} + z_{22} \end{array}$$

Solving it directly, we obtain:

$$\begin{array}{rclrclrcl} y_2 - y_1 & = & 2 & y_3 & = & 0 & z_1 & = & y_2 \\ z_{21} & = & 6 + 2y_1 & z_{22} & = & 0 & z_{23} & = & -2 \end{array}$$

There are an infinite number of solutions, all of the form $X = a^{z_1 - 2} \cdot b^{z_1}$. For example, if $X = b^2$, the attacker constructs a^6 as $(a \cdot b)^6 \cdot (b \cdot b^2)^{-2}$.

Theorem 2 (Soundness and completeness) *Symbolic constraint sequence* $\mathbf{C}$ *has a solution if and only if the system of equations (1) has a solution in integers.*

Decidability of simultaneous quadratic Diophantine equations is not known in general (systems of linear Diophantine equations are decidable [8]). There are various partial results, listed in standard surveys. For example, the two-variable equation can be solved completely [11]. The following quote is taken from [25, p. 1164]: "There is a fairly complete theory of homogeneous quadratic Diophantine equations with three variables, and on the basis of results from the early and mid-1900s a finite procedure should in principle be able to handle quadratic Diophantine equations with any number of variables. (The same is not true of simultaneous quadratic Diophantine equations)." As our running example of Section 6 shows, in particular cases it may be reasonably easy to find solutions despite the lack of a general algorithm. We conjecture that the system of equations (1) is decidable.

Corollary 6.9 (Decidability with `xor`) *If* $\cdot$ *is interpreted as* `xor` *(*i.e., $\forall t \; t^{-1} = t$*), then symbolic trace reachability is decidable.*

Proof: If $\forall t \; t^{-1} = t$, then PS–v(T) is a finite set for any T, and the value of $x\sigma$ can be found by exhaustive enumeration. Also, the $y[j,k], z[i,s], z[i,s']$ variables of system (1) can only assume 0 or 1 value. Therefore, solution can be found by exhaustive enumeration. □

Corollary 6.10 (Decidability with free term algebra) *In the absence of any operators with algebraic properties, symbolic trace reachability is decidable.*

Proof: In the absence of rules (M) and (I), the first three steps of the reduction procedure of Section 6 give a sound and complete decision procedure for solvability of $\mathbf{C}$, resulting in an empty $\mathbf{C}_3$ if and only if $\mathbf{C}$ has a solution. □

7. Extension to Group Diffie-Hellman

In this section, we extend the constraint solving approach developed in Section 6 to protocols such as group Diffie-Hellman [23]. Our extension, however, applies only in a restricted setting. We assume that the Abelian group (multiplication) operator appears *only* in the exponents. In particular, exponentials are not multiplied with each other, *i.e.*, terms such as $g_1^x \cdot g_2^y$ do not appear in the protocol specification, nor is the attacker permitted to multiply exponentials. This restriction is necessary to preserve decidability, since it has been shown that unification (and, therefore, the symbolic analysis problem) is undecidable in the presence of equational theories for *both* Abelian groups and exponentiation [12, 13]. This does not affect our ability to analyze protocols such as [23] since they satisfy the restriction (a similar restriction is adopted by [19]).

We extend the message term constructors of Figure 1 with terms $t_1^{t_2}$ representing exponentials. We also extend the rules of Figure 2 with the rules for exponentials, as shown in Figure 4. The rules of Figure 4 were shown in [15] to lead to unique normal forms up to associativity and commutativity of the $\cdot$ operator.

For the class of protocols we are interested in, it is typically assumed that all exponentiation is ultimately from a constant base, that is, the normal form of every exponential ground term is g^t where g is a public constant. Also, in the protocols of interest, multiplication, exponentiation and other arithmetic is typically performed in a large finite field. We assume, as protocol designers do, that the discrete logarithm problem is hard, and exponentiations are thus not invertible in practice.

Rules of Figure 2, and

$$t^{\mathbf{1}} \rightarrow t \qquad (t_1^{t_2})^{t_3} \rightarrow t^{t_2 t_3}$$

Figure 4. Normalization rules for exponentials

Consider a derivation constraint of the form

$$g^u : t_1, \dots, t_m \tag{2}$$

Under the Computational Diffie-Hellman Assumption (it is not feasible to compute g^{xy} from g^x and g^y), the *only* way the attacker can compute the needed exponential g^u is to take one (and no more than one!) of the exponentials at his disposal, *i.e.*, some $t_i = g^v$ (at least one such term is available, since $g = g^1$ is the publicly known base) and raise it to a power $\overline{v}u$ such that $(g^v)^{\overline{v}u} = g^u$, provided that such a $\overline{v}$ exists, and that $\overline{v}u$ is derivable from $t_1, \dots, t_m$.

Hence, every time we encounter a constraint of form (2), we nondeterministically choose one of the exponential terms g^v from the term set available to the attacker, and we replace the constraint (2) with

$$\overline{v}u : t_1, \dots, t_m \quad \text{where some } t_i = g^v. \tag{3}$$

These constraints are solved by the procedure in Section 6, treating $\overline{v}$ as v^{-1}. However, there is one proviso. When arithmetic is performed mod p, existence of $\overline{v}$ requires that $v\overline{v} \equiv 1 \mod (p-1)$, since $g^{p-1} \equiv 1 \mod p$. Such a $\overline{v}$ does not exist for divisors of $p-1$ (other than $\mathbf{1}$). This means that our procedure in Section 6, in this case, may produce some formal solutions that cannot actually be instantiated. (Thanks to Lida Wang for noting that we should mention this.)

Our constraint solving procedure only applies to finite symbolic traces. Therefore, it can be used to analyze only a finite number of instances of group Diffie-Hellman-based protocols. This restriction is inherent in the symbolic analysis approach. In fact, protocol analysis with unbounded instances is known to be undecidable even in the free term algebra case [9, 1], and thus certainly undecidable once the equational properties of multiplication are taken into account. While symbolic constraint solving cannot be used to prove that a protocol is correct in the general case, it can used for fully automated discovery of attacks on specific instances without imposing any bounds on the size of terms created by the attacker, or the attacker's ability to exploit algebraic properties of multiplication and `xor`.

7.1. Pereira-Quisquater Example

In [19], Pereira and Quisquater find an attack against key authentication in an authenticated group Diffie-Hellman (A-GDH.2) protocol. The way that attack would be identified using the approach in this paper is sketched here.

The attack involves two key distribution sessions, the first among four parties, one of whom is dishonest, and the second among three parties after the dishonest party has been removed from the group. The dishonest party causes one of the honest parties to accept a non-authentic, compromised key in the second session.

In a decidable finite-session context, the analyst chooses how many strands of each role to consider. It is sufficient in this example, involving a four-party group of M_1, M_2, M_3 and M_4, to choose two instances of role M_2 and one of M_4. M_1 is ignored, M_2 is compromised, and M_3 is the dishonest party, whose role is taken over by the attacker.

The role of the intermediate party M_2 is this:

$$- < g, g^x > + < g^r, g^x, g^{xr_2} > -g^{yk_2}$$

where r_2 is a secret nonce and k_2 is M_2's secret key, shared with M_4. The role of the last party M_4 is this:

$$- < g^{z_1}, g^{z_2}, g^{z_3}, g^{z^4} > + < ..., g^{z_2 r_4 k_2}, g^{z_3 r_4 k_3}, ... >$$

In the semibundle to be analyzed, nonces are instantiated in the strands where they are generated, so we will have as constants r_2 and r_2', r_4, and the keys k_2 and k_3. x, x', y', and z_i remain variables. Primed variables are associated with the second session. The second "$-$" node in the first instance of M_2 is not used, but it is included, using y', in the second instance. (It is not necessary to predict this, but it simplies the presentation.)

The analysis considers all the finite number of possible event sequences consistent with the semibundle, in particular the one identified by [19], which yields the constraint sequence:

$$\begin{array}{rcl}
g^{\mathbf{1}} & : & g^{\mathbf{1}}, k_3 \\
g^{x} & : & '' \\
g^{z_1} & : & '', g^{r_2}, g^{x}, g^{xr_2} \\
g^{z_2} & : & '' \\
g^{z_3} & : & '' \\
g^{x'} & : & '' \\
g^{y'k_2} & : & '', g^{r_2'}, g^{x'}, g^{x'r_2'} \\
g^{y'r_2'} & : & ''
\end{array}$$

The ditto mark $''$ means to repeat the right-hand terms from the constraint just above. The last constraint says that the key $g^{y'r_2'}$, which is computed and accepted as the secret group key by M_2, is derivable; this is the vulnerability goal.

All of the constraints have an exponential on the left, so we do not have to go through any of the unification or subterm constraint introduction steps that might arise in general. For each of the exponentials on the left we find, by finite search or by oracle, the exponential on the right from which it is computed. For example, the third constraint is solved by computing g^{z_1} from g^{r_2}. With the correct choices, the exponential removal step leads to product constraints

$$\begin{array}{rcl}
x & : & k_3 \\
\overline{r_2} z_1 & : & '' \\
\overline{x r_2} z_2 & : & '' \\
\overline{x r_2} z_3 & : & '' \\
\overline{z_3 r_4}\,\overline{k_3} x' & : & '' \\
\overline{r_2}' z_1' & : & '' \\
\overline{x}' z_2' & : & '' \\
\overline{x}' \overline{r_2}' z_3' & : & '' \\
\overline{z_2 r_4} y' & : & '' \\
\overline{x}' y' & : & ''
\end{array}$$

Here we have omitted exponential terms on the right because they will not occur in exponents, and $\mathbf{1}$ is implicit as a zeroth power.

Now we solve for the variables by expressing them as products of unknown powers of the factor terms, all of which are constants. After substitution, the products on the left of each constraint are set equal to a product of unknown powers of the terms on the right. There are no variables on the right, so the Diophantine equations will be linear.

For example, if $x' = k_3^{m_1} r_2^{m_2} r_4^{m_3}$ and $z_2' = k_3^{n_1} r_2^{n_2} r_4^{n_3}$, the seventh constraint (the second one with x') leads to the Diophantine equations $n_1 - m_1 = j, n_2 - m_2 = 0, n_3 - m_3 = 0$, where j is the unknown power of k_3 on the right. The full set of Diophantine equations has many solutions, including $x = \mathbf{1}, x' = z_2' = y' = r_2 r_4, z_1 = z_2 = z_3 = r_2, z_1' = r_2', z_3' = r_2 r_4 r_2'$, slightly simpler than the solution in [19].

8. Conclusions

We have presented a constraint solving technique that reduces the problem of symbolic protocol analysis in the presence of an Abelian group operator to a system of quadratic Diophantine equations. The significance of this result is that it enables formal analysis of a wide class of protocols that cannot be analyzed in a free-algebra model.

The most important question for further investigation is whether the resulting system of equations is decidable in the general case of an Abelian group operator (as opposed to `xor`), and, if so, what is the complexity of solving it? We conjecture that it is, in fact, decidable. By comparison, for both the free algebra [20] and `xor` [4], the symbolic analysis problem has been proved NP-complete.

Results presented in this paper are but the first step towards reducing the gap between formal methods and mathematical proofs typically employed in cryptographic analysis of security protocols. Even though we take into account some mathematical properties of the underlying cryptographic primitives, we are still analyzing an abstract model, and thus possibly missing attacks due to our idealized treatment of cryptography. It would be interesting to know whether the results of this paper, especially the existence of conservative solutions, can be extended to algebraic theories other than Abelian groups, or to richer equational theories that more accurately represent properties of the relevant cryptographic functions. At the same time, recent undecidability results for equational unification [12, 13] suggest that the symbolic constraint solving problem is undecidable in the presence of rich equational theories. Therefore, it is very likely that symbolic analysis can be fully automated only for abstract protocol models, or for protocols that employ cryptographic primitives without visible mathematical properties.

References

[1] R. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. In *Proc. CONCUR '00*, volume 1877 of *LNCS*, pages 380–394, 2000.

[2] F. Baader and W. Snyder. Unification theory. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 1, chapter 8, pages 445–532. Elsevier Science, 2001.

[3] M. Boreale. Symbolic trace analysis of cryptographic protocols. In *Proc. ICALP '01*, pages 667–681, 2001.

[4] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with XOR. In *Proc. LICS '03 (to appear)*, 2003.

[5] E. Clarke, S. Jha, and W. Marrero. Verifying security protocols with Brutus. *ACM Transactions in Software Engineering Methodology (TOSEM)*, 9(4):443–487, 2000.

[6] H. Comon, V. Cortier, and J. Mitchell. Tree automata with one memory, set constraints, and ping-pong protocols. In *Proc. ICALP '01*, pages 682–693, 2001.

[7] H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proc. LICS '03 (to appear)*, 2003.

[8] E. Contejean and H. Devie. An efficient algorithm for solving systems of Diophantine equations. *Information and Computation*, 113(1):143–172, 1994.

[9] N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Proc. FLOC Workshop on Formal Methods in Security Protocols*, 1999.

[10] M. Fiore and M. Abadi. Computing symbolic models for verifying cryptographic protocols. In *Proc. 14th IEEE Computer Security Foundations Workshop*, pages 160–173, 2001.

[11] K. Ito, editor. *Encyclopedic Dictionary of Mathematics*. Springer-Verlag, 2nd edition, 1994.

[12] D. Kapur, P. Narendran, and L. Wang. A unification algorithm for analysis of protocols with blinded signatures. Technical Report TR 02-5, Department of Computer Science, SUNY Albany, 2002.

[13] D. Kapur, P. Narendran, and L. Wang. Analyzing protocols that use modular exponentiation: Semantic unification techniques. In *Proc. RTA '03 (to appear)*, 2003.

[14] L. Lamport. Time, clocks and the ordering of events in a distributed system. *Comms. ACM*, 21(7):558–565, 1978.

[15] C. Meadows and P. Narendran. A unification algorithm for the group Diffie-Hellman protocol. In *Proc. Workshop of Issues in Theory of Security (WITS)*, 2002.

[16] J. Millen and V. Shmatikov. Constraint solving for bounded process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security (CCS '01)*, pages 166–175, 2001.

[17] L. Paulson. Mechanized proofs for a recursive authentication protocol. In *Proc. 10th IEEE Computer Security Foundations Workshop*, pages 84–95, 1997.

[18] L. Paulson. The inductive approach to verifying cryptographic protocols. *J. Computer Security*, 6(1/2):85–128, 1998.

[19] O. Pereira and J.-J. Quisquater. A security analysis of the Cliques protocols suites. In *Proc. 14th IEEE Computer Security Foundations Workshop*, pages 73–81, 2001.

[20] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proc. 14th IEEE Computer Security Foundations Workshop*, pages 174–190, 2001.

[21] P. Ryan and S. Schneider. An attack on a recursive authentication protocol: A cautionary tale. *Information Processing Letters*, 65(1):7–10, 1998.

[22] D. Song. Athena: a new efficient automatic checker for security protocol analysis. In *Proc. 12th IEEE Computer Security Foundations Workshop*, pages 192–202, 1999.

[23] M. Steiner, G. Tsudik, and M. Waidner. Diffie-Hellman key distribution extended to group communication. In *Proc. 3rd ACM Conference on Computer and Communications Security (CCS '96)*, pages 31–37, 1996.

[24] F. Thayer, J. Herzog, and J. Guttman. Strand spaces: proving security protocols correct? *J. Computer Security*, 7(1), 1999.

[25] S. Wolfram. *A New Kind of Science*. Wolfram Media, 2002.

A. Proof of Lemma 3.8

Lemma 3.8 *If* $\mathbf{C}$ *does not satisfy the variable stability property, then the corresponding protocol is not well-defined: it requires an honest participant to split a product of two unknown values.*

Proof: (SKETCH) Suppose $\exists\theta$ such that $\mathbf{C}\theta$ does not satisfy the origination property. Let θ be the smallest such substitution, *i.e.*, $\theta = \theta' \circ \sigma$ where $\mathbf{C}\theta'$ satisfies the origination property and $\sigma = [x \to t]$ is a one-variable substitution. We will show that $\mathbf{C}\theta' = \mathbf{C}'$ (and, by induction over θ', $\mathbf{C}$) requires an ill-defined nondeterministic computation.

By our choice of $\mathbf{C}'$ and σ, $\exists z \in \mathrm{Var}(x\sigma)$ that does not satisfy the origination property in $\mathbf{C}'\sigma$. Since $\mathrm{Var}(\mathbf{C}') = \mathrm{Var}(\mathbf{C}'\sigma) \cup \{x\}, z \in \mathrm{Var}(\mathbf{C}')$. Let $k_x = k_x(\mathbf{C}'), k_z = k_z(\mathbf{C}')$ be the indices of the constraints in which x and z, respectively, appear in $\mathbf{C}'$ for the first time, and let $k'_z = k_z(\mathbf{C}'\sigma)$ to the index of the constraint in which z first appears in $\mathbf{C}'\sigma$.

By assumption, both x and z satisfy the origination property in $\mathbf{C}'$ with respect to some ordering $\prec$. There are three possibilities: $k_z < k_x, k_x < k_z$, or $k_x = k_z$.

Consider $k_z < k_x$. By choice of σ, it does not affect any constraints not involving x. Therefore, $\forall i < k_x\ (u_i : T_i)\sigma = u_i : T_i$. Since $k_z < k_x$, we obtain that $k'_z = k_z$ and $u_{k_z}\sigma : T_{k_z}\sigma = u_{k_z} : T_{k_z}$. By Definition 3.6 applied to z in $\mathbf{C}'$, $z \in \mathrm{St}(u_{k_z}) = \mathrm{St}(u_{k'_z})$ and $\forall y \in \mathrm{Var}(T_{k_z}) = \mathrm{Var}(T_{k'_z})\ y \prec z$. This contradicts our assumption that z does not satisfy the origination property in $\mathbf{C}'\sigma$.

If $k_x < k_z$, then, by choice of $\prec$, $x \prec z$. By Definition 3.6 applied to x in $\mathbf{C}'$, $x \in \mathrm{St}(u_{k_x})$ and $\forall y \in \mathrm{Var}(T_{k_x})\ y \prec x$. Since $x \prec z$, $\forall y\ y \prec z$. Moreover, by definition of k_x and k_z, $\forall i < k_x\ x, z \notin \mathrm{Var}(u_i : T_i)$. Since $T_{k_x}\sigma = T_{k_x}$, we obtain that $z \in \mathrm{St}(x\sigma) \subseteq \mathrm{St}(u_{k_x}\sigma)$ and $\forall y \in \mathrm{Var}(T_{k_x}\sigma)\ y \prec z$. Therefore, $k'_z = k_x$ and z satisfies the origination property in $\mathbf{C}'\sigma$, which contradicts our assumption.

Finally, consider $k_z = k_x$. Since σ does not affect any constraints preceding k_z, if $z \in \mathrm{St}(u_{k_z}\sigma)$, then $k'_z = k_z = k_x$ and the origination property is satisfied, which contradicts our assumption. Therefore, it must be that $z \notin \mathrm{St}(u_{k_z}\sigma)$. Since $z \in \mathrm{St}(u_{k_z})$, by Proposition 3.4 x and z must be entangled in u_{k_z}. Let $u_{xz} \in \mathrm{St}(u_{k_z})$ be the minimal subterm of u_{k_z} in which x and z are entangled, and observe that $x, z \notin u_{xz}\sigma$.

Now consider $u_{k'_z} : T_{k'_z}$, *i.e.*, the constraint in which z appears for the first time in $\mathbf{C}'\sigma$. Observe that $k_z < k'_z$ (σ does not affect constraints preceding k_z). Since z does not satisfy the origination property, there are two possibilities: $z \notin \mathrm{St}(u_{k'_z})$, or $\exists y \in \mathrm{Var}(T_{k'_z})\ z \prec y$.

If $z \notin \mathrm{St}(u_{k'_z})$, it must be that $z \in \mathrm{St}(T_{k'_z})$. Let $\hat{u}_{k'_z} : \hat{T}_{k'_z} \in \mathbf{C}'$ be the constraint from which $u_{k'_z} : T_{k'_z}$ was obtained by substitution σ, *i.e.*, $\hat{u}_{k'_z}\sigma = u_{k'_z}$, $\hat{T}_{k'_z}\sigma = T_{k'_z}$. Either x, or $z \in \mathrm{St}(\hat{T}_{k'_z})$. Moreover, even if $u_{xz} \in \mathrm{St}(\hat{T}_{k'_z})$, either x, or z appears in a subterm other than u_{xz} since $x, z \notin u_{xz}\sigma$. Therefore, by Lemma 3.5, the protocol requires an ill-defined computation. More precisely, it requires some honest participant to receive a term in which x and z are entangled and extract either x, or z from it. Such protocols are not well-defined, and we shall exclude them from our consideration.

The last case is $z \notin \mathrm{St}(T_{k'_z})$ and $\exists y_1, \ldots, y_m \in \mathrm{Var}(T_{k'_z})$ such that $z \prec y_1 \ldots \prec y_m$. If $\exists\ y_i$ such that $y_i \in \mathrm{Var}(T_{k'_z})$ but $y_i \notin \mathrm{Var}(\hat{T}_{k'_z})$, then it must be that $y_i \in \mathrm{Var}(x\sigma)$. By induction on the structure of $x\sigma$, we prove that in this case $z \in \mathrm{Var}(T_{k'_z})$, which contradicts our assumption that $z \notin \mathrm{St}(T_{k'_z})$. Therefore, it must be that $y_1, \ldots, y_m \in \mathrm{Var}(\hat{T}_{k'_z})$

Since $y_i \in \mathrm{Var}(\mathbf{C}')$, they satisfy the origination property in $\mathbf{C}'$ with respect to some $\prec$. Let $\prec' = \prec$ with z inserted immediately after y_m, *i.e.*, if $\ldots \prec x_i \prec z \prec x_{i+1} \ldots \prec y_m \prec \ldots$, then $\ldots \prec'\ x_i \prec'\ x_{i+1} \ldots \prec'\ y_m \prec'\ z \prec' \ldots$. By Definition 3.6, z satisfies the origination property in $\mathbf{C}'\sigma$ with respect to $\prec'$, and $\forall v \in \mathrm{Var}(\mathbf{C}'\sigma)$ such that $v \neq z$ satisfies the origination property with respect to $\prec'$ by construction. This contradicts our assumption that $\mathbf{C}'\sigma$ does not satisfy the origination property. □