# A unification-based decision procedure for cryptographic protocol analysis

Deepak Bhardwaj and Deepak D'Souza

Department of Computer Science and Automation
Indian Institute of Science, Bangalore, India.
{ramasya, deepakd}@csa.iisc.ernet.in

**Abstract.** We present a sound and complete decision procedure for the bounded process cryptographic protocol insecurity problem, based on the notion of normal proofs [2] and classical unification. We also show a result about the existence of attacks with "high" normal cuts. Our proof of correctness provides an alternate proof and new insights into the fundamental result of Rusinowitch and Turuani [9] for the same setting.

## 1 Introduction

In a typical cryptographic protocol, principals exchange messages according to the rules of the protocol, with the aim of setting up shared secret keys or exchanging secret information. A security flaw, or an "attack" on the protocol, is a particular instantiation of the protocol in which a malicious agent (or an "intruder") can obtain information meant to be kept secret between other principals in the protocol. The work in this paper is about a decision procedure for detecting the existence of such attacks on a given protocol.

The problem of finding the existence of an attack is modelled as the *protocol insecurity problem*. A protocol is presented as a finite set of "roles", each of which is a finite sequence of 'send' or 'receive' messages. These messages could contain variables which are place holders for message parts that might have been received by the instance of the role in the past. In a particular scenario or instantiation of the protocol, the roles are instantiated into multiple "processes" depending on how the principals take part in them. Each principal could potentially take part in multiple instances of roles. One of the principles is an "intruder" who can instrument a run of the protocol by intercepting and substituting messages sent from one agent to another. The messages the intruder can send must be legally derivable from his knowledge of messages exchanged in the protocol till that point. For example, he can obtain the contents of an encrypted message only if he has the corresponding decryption key. We assume in the present context that the cryptographic operations appear as black boxes to the intruder – he cannot make use of properties of a particular cryptographic operation. The protocol insecurity problem is then to tell whether there exists a particular instantiation of the given protocol, which an intruder can instrument to obtain a secret message.

While the problem can be seen to be undecidable in general [7, 4], several restrictions of the problem have been shown to be decidable, most notably for the case when we have a *bound* on the number of instances of roles or processes to be considered $[5, 1, 9, 6]$. The approach used in both $[9, 6]$ is to model the problem as a constraint solving problem. For a particular instance of the protocol, each run of the protocol is modelled as a sequence of constraints of the form $m_1 : T_1, \ldots, m_n : T_n$, corresponding to the $n$ messages $m_1, \ldots, m_n$ the intruder has to send, with each $T_i$ being his knowledge-set up to that point. The message to be kept secret usually appears (in the clear) as the final $m_i$ in constraint sequence. An attack by the intruder now corresponds to a "solution" to the constraint sequence in the form of a substitution $\sigma$ for variables in the $m_i$'s and $T_i$'s such that for each $i$, the substituted messages $m_i\sigma$ are derivable from the substituted set of messages $T_i\sigma$. The bounded process protocol insecurity problem can now be phrased in terms of checking whether any of a finite number of constraint sequences has a solution. We note that while the ground derivability problem (i.e. whether a concrete message is derivable from a set of concrete messages) is decidable, the decidability of the constraint solving problem is not obvious since there is no *a priori* bound on the depth of messages a solution may use.

The work of Rusinowitch and Turuani [9] is fundamental in this context. They provide a decision procedure for checking whether a constraint sequence has a solution. They show the basic result that if there is a solution, there is one that unifies each variable with a term with a concrete prefix that appears somewhere in the constraint sequence. This gives a bound on the size of substitutions one need look for, leading to an effective decision procedure for the problem.

In this paper we provide an alternate decision procedure for the constraint solving problem which leads us in a natural way to the basic result of Rusinowitch-Turuani. Our procedure is motivated by the existence of normal derivations [2] and the classical unification algorithm of Robinson [8]. The normality result says that if a concrete message $m$ is derivable from a set of concrete terms $T$, then there is a derivation of $m$ obtained by compositions of a set of subterms of $m$ (which we call a "cut" in $m$) each of which are subterms in $T$ *and* are derivable from $T$. The hope is now that (a) if there is a solution to the constraint sequence, there is a solution $\sigma$ whose normal cut lies in the unsubstituted parts of the $m_i\sigma$'s and is a unifier of the terms in the cuts of the $m_i$'s paired with subterms in the unsubstituted parts of terms in the constraint sequence; and finally that (b) the most general unifier (*mgu*) of these pairs (which is essentially unique and can be computed by Robinson's unification algorithm) is also a solution to the constraint sequence. Our correctness proof shows that this is indeed the case, and we obtain a decision procedure which essentially guesses cuts in the $m_i$'s, pairs them up with other subterms in the protocol, computes their pairwise simultaneous *mgu*, and uses the ground derivability procedure to check whether or not it is a solution to the constraint sequence.

We show in Section 6 that in fact the Rusinowitch-Turuani result can be used to give a straightforward proof of correctness of our decision procedure. Nonetheless, the completeness proof we give is independently of interest for several reasons. Firstly it leads us in a natural way to the result of Rusinowitch and Turuani, supplying motivation and a slightly different proof of their result. Our approach gives some useful insight into their result by way of counter-examples which demonstrate why some seemingly plausible restrictions to their technique would be incomplete. For example, the counter-examples point out why restricting the matches of variables to just the 'send' messages in the protocol would not be complete.

Secondly, we prove a result which shows the existence of solutions (whenever a solution does exist) that have "high" normal cuts, in the sense that they lie wholly in the unsubstituted parts of the left-hand side terms. This result does not follow from the technique of Rusinowitch and Turuani, and is interesting in its own right.

## 2 The problem setting

In this section we describe the problem setting and how the protocol insecurity problem can be phrased as a constraint satisfaction problem. We use some terms informally here, and define the required notions formally in the next section.

The messages in a protocol are represented by terms built using pairing $\langle \_, \_ \rangle$, public key encryption $\{\_\}_{\_}$ and symmetric key encryption $[\_]_{\_}$. We assume a finite set of atomic messages comprising principal names, nonces, and keys. Public and private keys of a principal $p$ are denoted by $pb(p)$ and $pv(p)$ respectively, while the name of the intruder is denoted by $\epsilon$.

We follow the protocol model used in [6]. A protocol is specified by a finite set of parameterized roles which are finite sequences of messages where each message has a '+' or '−' sign prefixed to it. A '$+m$' in a role denotes that $m$ is sent by the principal corresponding to this role. Similarly '$-m$' denotes the receipt of $m$. Message terms may contain variables which are denoted by capital letters. For example the initiator and responder roles of the Needham-Schroeder public-key handshake protocol can be specified as

$$\begin{aligned} \mathit{Init}(A, B, N) &= +\{\langle A, N \rangle\}_{pb(B)} \ -\{\langle N, M \rangle\}_{pb(A)} \ +\{M\}_{pb(B)}, \\ \mathit{Resp}(A, B, M) &= -\{\langle A, N \rangle\}_{pb(B)} \ +\{\langle N, M \rangle\}_{pb(A)} \ -\{M\}_{pb(B)}. \end{aligned}$$

Variables in the roles are of two kinds: parameters, and place holders for parts of a received message. In the above example of the initiator role, $A$, $B$ and $N$ are parameters. For example if $a$ is the initiator in some instance of the protocol, who wants to talk to $b$ using nonce $n$, then the parameters $A$, $B$ and $N$ will be instantiated to $a$, $b$ and $n$ respectively. However $M$ is a place holder for a term that the initiator will receive from another principal, and hence is the other kind of variable. The following role instance depicts this. We will use lowercase letters for the parameters and uppercase letters for the place holder variables.

$$\mathit{Init}(a, b, n) = +\{\langle a, n \rangle\}_{pb(b)} \ -\{\langle n, M \rangle\}_{pb(a)} \ +\{M\}_{pb(b)}.$$

We note that in a concrete instance of a role, variables (place holders) always appear in a '−' term for the first time. Also these variables can be viewed as being under the control of the attacker, as the recipient has to accept whatever is sent to her in place of that variable. In this notion of a role, a principal has no guarantee that the messages sent by her reach the intended receiver or that the messages received by her are from the purported sender, which models the insecure nature of the communication channel. In other words all '+' messages are sent to attacker and all '−' messages are received from attacker.

The capabilities of the intruder are as per the Dolev-Yao model [3]. In the Dolev-Yao model the intruder can eavesdrop, memorize, divert, compose, decompose, encrypt and decrypt messages if he has the decryption keys. For example, if the intruder has $t_1$ and $t_2$, he can generate messages $\langle t_1, t_2\rangle, [t_1]_{t_2}$, and if he has $\{t\}_{pb(\epsilon)}$ and $pv(\epsilon)$ he can get $t$, and so on.

We use interleavings of the role instances to model the different orderings of the messages sent to or received from the attacker. An interleaving is a shuffling of the messages in the role instance with the restriction that the messages pertaining to an instance must appear in the same order as the role. We call an interleaving *executable* if the attacker can generate all the '−' messages in it using his initial knowledge and the messages he has seen till that point in time.

We will consider the problem of finding attacks when there are only a finite number of protocol instances running. Since the number of the protocol instances bounds the number of the principals that can be involved in the protocol instances, its sufficient to consider the finite set of role instances in which principal names are instantiated from a finite set of names.

We focus on the verification of secrecy properties. A secrecy property states that some designated message should not be made public. For a finite set of role instances, leakage of a secret $s$ can be checked by looking for an executable interleaving of the role instances with an added single message role $(-s)$, denoting that an arbitrary principal receives the secret (from the attacker) in plain.

Every '$-m$' term in an interleaving generates a *constraint*, for the attacker, of the form $m : T$ where $T$ is a set of terms, corresponding to the attacker's initial knowledge along with the set of terms corresponding to the '+' terms appearing before '$-m$'. Thus an interleaving is executable if there exists a substitution $\sigma$ for the variables such that each $m_i\sigma$ is derivable from $T_i\sigma$.

In the Needham-Schroeder protocol for example, let us take the concrete roles $\mathit{Init}(a, \epsilon, n)$ and $\mathit{Resp}(a, b, m)$, meaning that $a$ is playing the initiator role in one instance of the protocol with $\epsilon$, and in another instance of the protocol $b$ is playing the responder role with $a$ with whom he wants to share the secret $m$. We add the role $(-m)$ to our role set and consider the following interleaving.

$$+\{\langle a, n\rangle\}_{pb(\epsilon)} - \{\langle a, N\rangle\}_{pb(b)} + \{\langle N, m\rangle\}_{pb(a)} - \{\langle n, M\rangle\}_{pb(a)} + \{M\}_{pb(\epsilon)} - m.$$

The initial knowledge of the intruder is $T_0 = \{a, b, \epsilon, pb(a), pb(b), pb(\epsilon)\}$, and the corresponding constraint sequence for this interleaving is

$$\begin{aligned}
\{\langle a, N\rangle\}_{pb(b)} &: T_1 = T_0 \cup \{\{\langle a, n\rangle\}_{pb(\epsilon)}\}, \\
\{\langle n, M\rangle\}_{pb(a)} &: T_2 = T_1 \cup \{\{\langle N, m\rangle\}_{pb(a)}\}, \\
m &: T_3 = T_2 \cup \{\{M\}_{pb(\epsilon)}\}.
\end{aligned}$$

The substitution $\sigma : N \mapsto n$, $M \mapsto m$ is a solution to this constraint sequence and corresponds to an attack in which the intruder is able to obtain the secret $m$ which $b$ believes to be shared between $a$ and him.

## 3 Preliminary notions

We assume a finite set of constants *Const* and a set of variables *Var*. A *term* is defined by the grammar: $t ::= c \mid x \mid \langle t,t \rangle \mid \{t\}_t \mid [t]_t$, where $c \in \mathit{Const}$ and $x \in \mathit{Var}$. A term with no variables in it is called a *ground* term. We denote the set of terms by $\mathcal{T}$. For a set of terms $T$, we denote the set of subterms of $T$ (defined in the standard way) by $St(T)$.

A term can be represented naturally as a tree where every leaf is a variable or constant and the internal nodes are operators like pairing and encryption. For example see Fig 1(a). Every node $u$ in the tree represents a term corresponding to the subtree rooted at $u$, and we denote this term by $\widetilde{u}$.

A *cut* of a term $t$ is a set of nodes $c$ in tree of $t$, such that any path from the root to a leaf in the tree of $t$ has exactly one node from the set. We extend the earlier notation to denote the set of terms corresponding to the cut $c$ by $\widetilde{c}$. Fig. 1(b) shows two cuts $c_1$ and $c_2$ of the term $[[[a]_{k_1}]_{k_2}]_{k_3}$. Here $\widetilde{c_1} = \{[[a]_{k_1}]_{k_2}, k_3\}$ while $\widetilde{c_2} = \{[a]_{k_1}, k_2, k_3\}$.

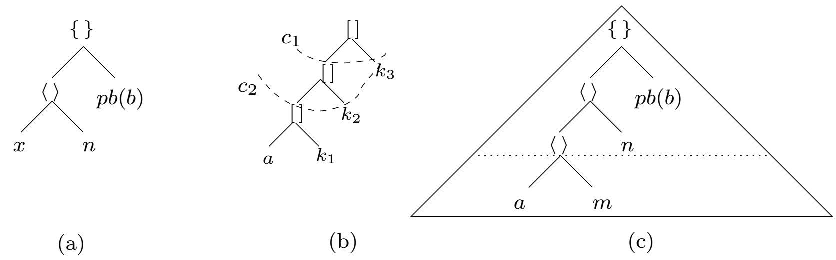


**Fig. 1.** (a) Term $\{\langle x,n \rangle\}_{pb(b)}$ as a tree, and (b) two cuts of $[[[a]_{k_1}]_{k_2}]_{k_3}$.

The intruder's capabilities are modeled by the inference rules given below.

| | | | |
|---|---|---|---|
| *Public key Encryption (PE)* | $\dfrac{u\ ,\ v}{\{u\}_v}$ | *Symmetric Encryption (SE)* | $\dfrac{u\ ,\ v}{[u]_v}$ |
| *Pairing (P)* | $\dfrac{u\ ,\ v}{\langle u, v\rangle}$ | *Unpairing (U)* | $\dfrac{\langle u, v\rangle}{u\ ,\ v}$ |
| *Public key decryption (PD)* | $\dfrac{\{u\}_{pb(p)}\ ,\ pv(p)}{u}$ | *Symmetric decryption (SD)* | $\dfrac{[u]_v\ ,\ v}{u}$ |
| *Public key decryption (PD)* | $\dfrac{\{u\}_{pv(p)}\ ,\ pb(p)}{u}$ | | |

Rules $PE$, $SE$, and $P$ are called *composition* rules while the rules $U$, $PD$ and $SD$ are called *decomposition* rules.

A *derivation* of a term $m$ from a set of terms $T$ (which we will assume to be wrt the inference system above) is a tree labeled with terms such that the root is labeled with $m$, leaves with terms in $T$, and every internal node labeled with a term $t$ has $n$ children $t_1, \ldots, t_n$ such that $\dfrac{t_1, \ldots, t_n}{t}$ is an instance of one of the rules $P$, $SE$, $PE$, $U$, $SD$, $PD$. We say $m$ is derivable from $T$, written $T \vdash m$, if there is a derivation of $m$ from $T$. The *size* of a derivation is the number of nodes in the derivation tree. A derivation of $m$ from $T$ is *minimal* if there is no derivation of $m$ from $T$ of smaller size.

We begin with some results that are well known in the literature.

**Proposition 1.** *If $T \vdash m$ then there exists a derivation of $m$ from $T$ in which all the terms are from $St(T \cup \{m\})$.*

*Proof.* It is sufficient to show that in a minimal derivation of $m$ from $T$,

- If the last rule is a decomposition rule then all the nodes in the derivation tree have labels $t$ with $t \in St(T)$
- If the last rule is a composition rule then all the nodes in the derivation tree have labels $t$ with $t \in St(T \cup \{m\})$.

The proof proceeds by induction on the size of a minimal derivation tree. □

Let $m$ be a term and $T$ a finite set of terms. A *normal* cut in $m$ (wrt $T$) is a cut $c$ in $m$ satisfying: (i) each term in $\widetilde{c}$ occurs in $St(T)$ and (ii) each term in $\widetilde{c}$ is derivable from $T$. It is easy to see that if a normal cut exists, there is a unique *uppermost* cut satisfying (i) and (ii), and hence there exists a unique uppermost normal cut.

**Proposition 2.** *$T \vdash m$ iff there exists a normal cut of $m$ wrt $T$.*

*Proof.* If there is a normal cut $c$ of $m$ (wrt $T$) then $m$ is derivable from $T$ since we can apply composition rules corresponding to the nodes above $c$, to the terms in $c$ which are all derivable from $T$. Conversely, consider a minimal derivation of $m$ from $T$. Starting from the root of the derivation tree, we start moving down as long as we find nodes which are results of an application of a compositional rule. We stop along a branch when we encounter either a leaf, or an internal node which is a result of a decomposition rule. Let $c$ be this set of nodes. Each node in $c$ is a subterm of $m$, and $c$ corresponds to a cut in $m$. Further, each $t$ in $\widetilde{c}$ is either a leaf node in the derivation tree or is a subterm obtained by a decomposition rule in the minimal derivation of $m$ from $T$. Thus in either case $t$ is a subterm of a term in $T$ (cf. Prop.1). Further, each $t$ is derivable from $T$. Hence $c$ is a normal cut in $m$ wrt $T$. □

**Proposition 3.** *Given a finite set of ground terms $T$ and a ground term $m$, the problem of checking whether $m$ is derivable from $T$ is decidable.*

*Proof.* The following procedure finds out whether $m$ is derivable from $T$.

1. Consider the uppermost cut $c$ of $m$ such that all the terms in $\widetilde{c}$ are subterms in $T$. Guess a subset $\mathcal{S}$ of $St(T)$ containing $\widetilde{c}$ and an ordering $s_1 \prec \ldots \prec s_n$ on $\mathcal{S}$.
2. For every $i = 1, \ldots, n$ check that $s_i$ is one-step derivable from $T \cup \{s_1, \ldots, s_{i-1}\}$.

The correctness of the procedure follows from Prop. 1 and Prop. 2. □

A *substitution* is a mapping $\sigma : \mathit{Var} \to \mathcal{T}$ from the set of variables to the set of all terms $\mathcal{T}$. The effect of a substitution $\sigma$ on a term $t$, written $t\sigma$, is defined inductively as follows: $a\sigma = a$, $x\sigma = \sigma(x)$, $\langle s, t\rangle\sigma = \langle s\sigma, t\sigma\rangle$, $[s]_t\sigma = [s\sigma]_{t\sigma}$, $\{s\}_t\sigma = \{s\sigma\}_{t\sigma}$. Thus the effect of $\sigma$ on $t$ is to simultaneously replace every occurrence of a variable $x$ with the term $\sigma(x)$. We use the notation $[t/x]$ to denote a substitution which sends $x$ to $t$ and is the identity map on all variables different from $x$. The composition of two substitutions $\sigma$ and $\tau$ is denoted by $\sigma \cdot \tau$ and defined to be substitution which sends each variable $x$ to $(x\sigma)\tau$. A *ground* substitution maps every variable to a ground term. Substitutions are extended to act on sets of terms in natural way.

A *unifier* of two terms $t$ and $t'$ is a substitution $\sigma$ such that $t\sigma = t'\sigma$. A unifier $\sigma$ of $t$ and $t'$ is called a *most general unifier* (*mgu*) of $t$ and $t'$ if every unifier $\sigma'$ of the two terms can be obtained by composing $\sigma$ with some other substitution $\tau$: i.e. $\sigma' = \sigma \cdot \tau$. The unification algorithm of Robinson [8] (given in the appendix) computes an *mgu* of two given terms.

Given a term $t$ and a substitution $\sigma$ there is a natural correspondence between nodes in the trees of $t$ and $t\sigma$. The upper portion in the tree of $t\sigma$ which corresponds to the tree of $t$ is called the *unsubstituted* part of $t\sigma$ while rest of the nodes come in the *substituted* part of $t\sigma$. For a variable $x$ which occurs in $t$, we assume that the node corresponding to $\sigma(x)$ in $t\sigma$ occurs in the unsubstituted part of $t\sigma$. Fig. 1(c) shows the two parts separated by a dotted line, for the substitution $x \mapsto \langle a, m\rangle$ on the term in (a).

A constraint sequence is a sequence $\mathcal{C}$ of pairs of the form $m : T$, where $m$ is a term and $T$ is a finite set of terms. $\mathcal{C}$ will be written as $\{m_i : T_i \mid 1 \leq i \leq n\}$. A constraint sequence is assumed to satisfy two properties: The first time a variable occurs in a constraint in the sequence, it appears only in the left side of the constraint (this corresponds to the fact that in a protocol role a variable first appears in a '$-$' term); and for each $i$, $T_i \subseteq T_{i+1}$ (representing the fact that the intruder remembers messages he has seen in the past).

Given a constraint sequence $\mathcal{C} = \{m_i : T_i \mid 1 \leq i \leq n\}$, the problem of deciding whether there exists a substitution $\sigma$ such that $T_i\sigma \vdash m_i\sigma$ for each $i$ is called the *constraint solving problem* and such a substitution $\sigma$ is called a solution to $\mathcal{C}$.

## 4 Preliminary attempts at constraint solving

Our aim is to find a decision procedure for the constraint solving problem based on the notion of normal cuts. For the rest of the paper, we fix the constraint sequence $\mathcal{C} = \{m_i : T_i \mid 1 \leq i \leq n\}$.

The idea is to somehow identify a finite set of substitutions which contains a solution to $\mathcal{C}$ iff there exists a solution to $\mathcal{C}$, and to check this set exhaustively for solutions by applying them to $\mathcal{C}$ and checking derivability using the ground derivability procedure. If there exists a solution $\sigma$ to $\mathcal{C}$, then for every constraint $m_i\sigma : T_i\sigma$, $m_i\sigma$ will have a normal cut $c_i$ wrt $T_i\sigma$ (in future we avoid mentioning "wrt $T_i\sigma$" when it is clear from the context). The hope is that for each $i$, the uppermost normal cut $c_i$ lies in the unsubstituted part of $m_i\sigma$. In that case, let the corresponding cut in $m_i$ be $c'_i$. We can further hope that the terms in $\widetilde{c_i}$, which appear in $T_i\sigma$, have a prefix in the unsubstituted portion of $T_i\sigma$. If this turns out to be true then $\sigma$ would be a unifier of terms in $\widetilde{c'_i}$ with terms in $St(T_i)$. Finally, the hope is that the most general unifier ($mgu$) for these pairs would also be a solution. We can then bound the set of substitutions we need to examine for finding the existence of an attack.

*Procedure 1:* Guess a cut $c_i$ in every $m_i$. For each term $t \in \widetilde{c_i}$ guess a term $t'$ from $St(T_i)$. Do this for all $c_i$'s. Compute a simultaneous most general unifier of all these pairs $(t, t')$, apply it to $\mathcal{C}$, and use the ground derivability procedure to check whether it is a solution or not.

Observe that soundness of this procedure follows from soundness of the ground derivability procedure. The number of cuts in a given $m_i$ is finite. In a guessed cut there are only finitely many choices for every term in the term set of this cut from $St(T_i)$ as $St(T_i)$ is finite. Finally there are only finitely many $mgu$'s for any given set of pair of terms. Hence the number of substitutions computed here is finite. To argue completeness of this procedure we must show that whenever $\mathcal{C}$ has a solution it has one which is a most general unifier of terms in the cuts of $m_i$'s, paired with the terms in $St(T_i)$'s. However, this is not true. Here is a counter example:

*Example 1.*

$$x : \{a, k_1\}$$
$$[x]_{k_2} : \{a, k_1, [[a]_{k_1}]_{k_2}\}$$

For first constraint the two possible pairings are $(x, a)$ and $(x, k_1)$. But neither $\{a/x\}$ nor $\{k_1/x\}$ is a solution to the constraint sequence. The only solution to this constraint sequence is $\sigma = \{[a]_k/x\}$. If we apply $\sigma$ on the first constraint it becomes $[a]_k : \{a, k\}$ and the normal cut lies in substituted part of $m_i\sigma$ (here $x\sigma$). That is why this procedure did not work. But we observe that the term with which $x$ unifies here appears in right side of the second constraint (in $[[a]_{k_1}]_{k_2}$). So if we use $St(\bigcup_{i=1}^{i=n} T_i)$ instead of $St(T_i)$ when we are guessing terms to be paired with the terms in $\widetilde{c_i}$'s, there may be still some hope. So we modify the procedure as follows.

*Procedure 2:* Guess a cut $c_i$ in each $m_i$. For each term in $\widetilde{c_i}$ guess a term from $St(\bigcup_{i=1}^{i=n} T_i) = St(T_n)$. Do this for all $c_i$'s. Compute the simultaneous most general unifier of all these pairs and apply it to $\mathcal{C}$ and use ground derivability procedure to see whether it is a solution.

Again soundness follows from soundness of ground derivability procedure.

To consider completeness, let us first define the notion of a $T_n$-normal cut. Let $\sigma$ be a substitution. A $T_n$-*normal* cut in $m_i\sigma$ wrt $T_i\sigma$, is a cut $c$ in $m_i\sigma$ satisfying: (i) $\widetilde{c} \subseteq St(T_n)$, and (ii) each term in $\widetilde{c}$ is derivable from $T_i$. It is easy to see that if a cut satisfying (i) and (ii) above exists, then there is a unique uppermost cut satisfying (i) and (ii), and hence there exists a unique uppermost $T_n$-normal cut. If $\sigma$ is a solution to $\mathcal{C}$, then both the normal and $T_n$-normal cuts can be seen to exist in $m_i\sigma$, and the uppermost $T_n$-normal cut coincides with or lies above the uppermost normal cut in $m_i\sigma$.

A natural way to show the completeness of Procedure 2 is following three step approach.

*Step 1:* If there is a solution to $\mathcal{C}$, then there is a solution $\sigma_1$ whose uppermost $T_n$-normal cut lies in the unsubstituted parts of the $m_i\sigma_1$'s. Fig. 2(a) shows cut $c_1$ of $m_i\sigma_1$ lies in unsubstituted part while $c_2$ lies in substituted part.

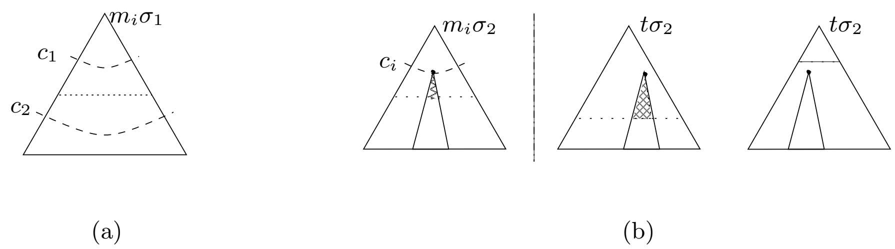


**Fig. 2.** Cuts in $m_i\sigma_1$ and possible pairing for a node in a cut in $m_i\sigma_2$

*Step 2:* If there is a solution $\sigma_1$ whose uppermost $T_n$-normal cut ($c_i$) lies in the unsubstituted parts of the $m_i\sigma_1$'s, then there is a solution $\sigma_2$ whose uppermost $T_n$-normal cut lies in the unsubstituted parts of the $m_i\sigma_2$'s *and* which unifies every term $t$ in $\widetilde{c_i}$ with some term $t'$in $St(T_n)$. In Fig. 2(b), $\sigma_2$ unifies the two subterms shown by shaded regions.

*Step 3:* If there is a solution $\sigma_2$ whose uppermost $T_n$-normal cut lies in the unsubstituted parts of the $m_i\sigma_2$'s *and* which unifies every term in $\widetilde{c_i}$ with some term in $St(T_n)$, then there is a solution $\sigma_3$ whose uppermost normal cut lies in the unsubstituted parts of the $m_i\sigma_3$'s and is a most general unifier of the terms in $\widetilde{c_i}$ paired with subterms in $St(T_n)$.

The first step for Procedure 2 goes through:

**Lemma 1.** *If $\mathcal{C}$ has a solution $\sigma$, then it has a solution $\sigma'$ with $T_n$-normal cuts that lie wholly in the unsubstituted parts of each $m_i\sigma'$.*

*Proof.* Let the uppermost $T_n$-normal cut in each $m_i\sigma$ be $c_i$ respectively. Suppose $c_k$ does not lie wholly in the unsubstituted part of $m_k\sigma$. Consider a node $u$ in $c_k$ which lies in the substituted part of $m_k\sigma$. Let the term rooted at $u$ be $t$ (i.e. $t = \widetilde{u}$). Since $u$ lies in the unsubstituted part of $m_k\sigma$, there must exist a variable $x$ in $m_k$ such that $t$ is a strict subterm of $x\sigma$ (Fig. 3(a)).

Now consider the substitution $\sigma'$ which is the same as $\sigma$ except that $\sigma'(x)$ equals $t$ instead of $\sigma(x)$. We argue that $\sigma'$ is also a solution to $\mathcal{C}$ and that $\sigma'$ has a $T_n$-normal cut which is similar to the $c_i$'s of $\sigma$, but is moved up into the unsubstituted part in at least one point (namely in $m_k$). In fact, it is sufficient to prove the latter part, since by Prop. 2 it would imply that each $m_i\sigma'$ is derivable from $T_i\sigma'$, and hence that $\sigma'$ is a solution to $\mathcal{C}$.

We first note the following facts about $\sigma$.

1. The variable $x$ does not occur anywhere in the $T_i$'s.

This is because $x$ lies above the cut $c_k$ in $m_k\sigma$, and hence $x\sigma$ is derivable from $T_k\sigma$. Thus if $x\sigma$ occured in $T_n\sigma$, we would have a $T_n$-normal cut passing through $x$ (a node higher than the root of $t$), contradicting the assumption that $c_k$ was the uppermost $T_n$-normal cut in $m_k\sigma$.

2. If $x$ occurs in some $m_j$, then the cut $c_j$ must pass through the descendant node of $x$ which corresponds to the root of the term $t$ (Fig. 3(b)).

By a similar reasoning to the one above, the cut $c_j$ cannot pass through a node above the root of $t$ since this would contradict the fact that $c_k$ was the uppermost $T_n$-normal cut and it passed through the root of $t$ in $m_k\sigma$. Similarly, it cannot pass below the root of $t$ since then $c_j$ would not be the uppermost $T_n$-normal cut in $m_j\sigma$.

Now consider the cuts $c'_i$ defined as follows. Considering that the nodes in each $m_i\sigma'$ also occur in $m_i\sigma$, we take the liberty of identifying the nodes in $m_i\sigma'$ with the corresponding nodes in $m_i\sigma$. If $x$ does not occur in $m_i$, define $c'_i = c_i$; if $x$ occurs in $m_i$, define $c'_i$ to be $c_i - X_i \cup \{u_i\}$, where $u_i$ is the node corresponding to the root of $t$ and $X_i$ is the set of nodes in $x\sigma$ which are part of $c_i$. The cut $c'_i$ is depicted in Fig. 3 (c).

We now argue that each $c'_i$ constitutes a $T_n$-normal cut in $m_i\sigma'$. We first observe that by item 1 above, $T_i\sigma = T_i\sigma'$ for each $i$, and hence the right-hand sides remain unchanged. Next, by construction, each $\widetilde{c'_i}$ is a subset of $\widetilde{c_i}$, and

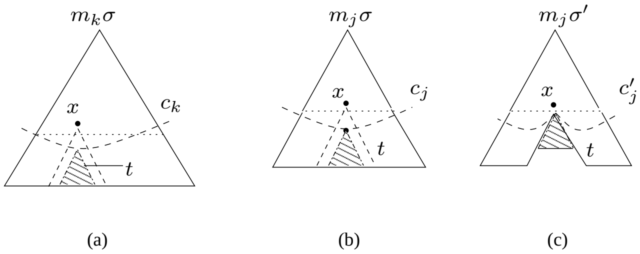


**Fig. 3.** 1st step

hence $\widetilde{c'_i} \subseteq St(T_i\sigma')$. Since the right hand sides remain unchanged, and each $t'$ in $\widetilde{c_i}$ was derivable from $T_i\sigma$, we have that each $t'$ in $\widetilde{c'_i}$ is derivable from $T_i\sigma'$.

Note that for the $m_i$'s in which $x$ occurs (in particular $m_k$), the cut $c'_i$ has at least one less node in the unsubstituted part than $c_i$. This argument can be repeated to obtain a solution with a $T_n$-normal cut which lies wholly in the unsubstituted parts of the $m_i$'s. □

However, the second step fails for this procedure. Here is a counter example.

*Example 2.*

$$\begin{aligned} x &: \{a, k_1\} \\ y &: \{a, k_1, k_2\} \\ [[x]_{k_2}]_{k_3} &: \{a, k_1, k_2, [y]_{k_3}\} \\ [[[a]_{k_1}]_{k_2}]_{k_3} &: \{a, k_1, k_2, [y]_{k_3}\} \end{aligned}$$

The only solution to this constraint sequence is $\sigma = \{[a]_{k_1}/x,\ [[a]_{k_1}]_{k_2}/y\}$. Although the uppermost $T_n$-normal cut of all the constraints lie in unsubstituted part, there is no term in $St(T_4)$ which $\sigma$ can unify with $x$. Therefore this step fails. The only terms which $\sigma$ unifies with $x$, appear in the terms on left only (eg. $[[[a]_{k_1}]_{k_2}]_{k_3}$). Hence this procedure is also not complete.

## 5 A sound and complete decision procedure

In Example 2 we observe that the only solution unified $x$ with a subterm of a receive message and not with a subterm of any send message. This motivates the following modified decision procedure which turns out to be sound and complete.

We first introduce some notation. Let $\sigma$ be a substitution, and let $u$ be a node in the tree of $\mathcal{C}\sigma$. We say $u$ has a *match* in $\mathcal{C}\sigma$ if there exists a node $v$ different from $u$ in $\mathcal{C}\sigma$ such that $\widetilde{u} = \widetilde{v}$. We also define the notion of a $\mathcal{C}$-normal cut. A $\mathcal{C}$-*normal* cut in $m_i\sigma$, is a cut $c$ in $m_i\sigma$ satisfying: (i) each node $u$ in $c$ has a match in $\mathcal{C}\sigma$; and (ii) each term in $\widetilde{c}$ is derivable from $T_i\sigma$. Clearly, each $T_n$-normal cut is also a $\mathcal{C}$-normal cut.

**Algorithm 1**: Decision procedure for constraint solving

**Input**: A constraint sequence $\mathcal{C}$.
**Output**: A solution to $\mathcal{C}$, if there exist one.

Guess a set $P$ of pairs of terms as follows:
    Guess a cut $c_i$ in each $m_i$;
    Pair each node $u$ in $c_i$ with a node $u'$ in $\mathcal{C}$ different from $u$, and add $(\widetilde{u}, \widetilde{u'})$ to $P$.
Let $G$ be the finite set of such guesses above.
**for** *each* $P$ *in* $G$ **do**
    find the pairwise simultaneous *mgu* $\sigma$ of $P$;
    Check if $\sigma$ is a solution to $C$ using ground derivability procedure;
    If yes, output $\sigma$ as a solution.
**end**
If no solution is found above, output "No solution exists".

We now prove the following three lemmas which will argue completeness of the procedure above.

**Lemma 2.** *If $\mathcal{C}$ has a solution, it has a solution $\sigma'$ with $\mathcal{C}$-normal cuts that lie wholly in the unsubstituted parts of the $m_i\sigma'$.*

*Proof.* Follows from Lemma 1 since a $T_n$-normal cut is also a $\mathcal{C}$-normal cut. □

**Lemma 3.** *If $\mathcal{C}$ has a solution, it has a solution $\sigma'$ with cuts $c_i$ that lie wholly in the unsubstituted parts of $m_i\sigma'$, with the property that each $u$ in $c_i$ has a match which lies in the unsubstituted part of $\mathcal{C}\sigma$. Thus, viewing each $c_i$ as a cut in $m_i$, and the match of each $u$ as a node $v$ in $\mathcal{C}$, $\sigma'$ is a unifier of each $(\widetilde{u}, \widetilde{v})$ pair.*

*Proof.* By lemma 2 we can assume that there exists a solution $\sigma$ to $\mathcal{C}$, with $\mathcal{C}$-normal cuts in $m_i\sigma$ that lie wholly in the unsubstituted parts of $m_i\sigma$. It is not difficult to see that for any cut which lies below these cuts, every node in the cut has a match in $\mathcal{C}\sigma$. Let us concentrate in particular on the "lowest" cut in the unsubstituted part of the $m_i\sigma'$, namely the cut corresponding to the cut in $m_i$ which passes through the leaf nodes. Let us call these cuts $c_i$.

If the $c_i$'s above are such that every node in them has a match which lies in the unsubstituted part of $\mathcal{C}\sigma$, then we are though. Else, there exists a cut $c_i$ and a node $u$ in $c_i$, such that all matches of $u$ lie in the substituted parts of $\mathcal{C}\sigma$. In such a case, we will produce a modified substitution $\sigma'$ which (a) is still a solution to $\mathcal{C}$, and (b) is such that the number of unmatched nodes in the lowest cut of $m_i\sigma'$ is at least one less than for $\sigma$.

We first note that the node $u$ (for which no desired match exists) cannot be a constant (i.e. $\widetilde{u}$ is not a constant). Otherwise it would mean that all the other occurrences (and there is at least one since $u$ has a match) of the constant are as part of $x\sigma$ for some variables $x$. Now consider the first time such a variable (say $y$) occurs in the constraint sequence, say in the constraint $m_k : T_k$. By the property of our constraint sequences, $y$ must appear only in $m_k$ and not in $T_k$.

Hence the constant appears in $m_k\sigma$ but not in $T_k\sigma$. But this means that $m_k\sigma$ is not derivable from $T_k\sigma$, which contradicts the fact that $\sigma$ was a solution to $\mathcal{C}$. Thus $u$ corresponds to a variable leaf node in $m_i$, say $x$, and $\widetilde{u}$ equals $\sigma(x)$.

Let us define the substitution $\sigma'$ as follows: For each variable $y$, $\sigma'(y) = \delta(y\sigma)$, where $\delta$ of a term $t$ replaces all occurrences of the term $x\sigma$ in $t$ by the constant $\epsilon$. We first show that $\sigma'$ is also a solution to $C$. We note that $x\sigma$ appears in any constraint $m_i\sigma : T_i\sigma$ only as a result of $\sigma$ applied to $x$ or some other variable $y$ for which $x\sigma$ is a subterm of $y\sigma$. Hence for every term $t'$ in the constraint set $t'\sigma' = \delta(t'\sigma)$.

Now suppose $x\sigma$ appears for the first time in the constraint $m_k\sigma : T_k\sigma$. For $i < k$, $m_i\sigma = m_i\sigma'$ and $T_i\sigma = T_i\sigma'$. Hence $T_i\sigma' \vdash m_i\sigma'$ for $i < k$.

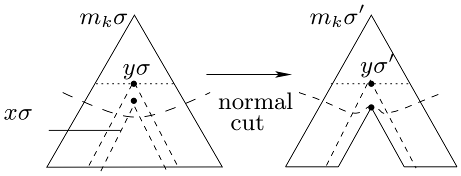


**Fig. 4.** Normal cut of $m_k\sigma$ passes strictly below the root of $t$.

Now consider $m_k\sigma : T_k\sigma$. Since a variable appears for the first time only in the left side of some constraint, $x\sigma$ doesn't appear in $T_k\sigma$ and therefore $T_k\sigma = T_k\sigma'$. As $T_k\sigma \vdash m_k\sigma$, there exists a normal cut of $m_k\sigma$ (by proposition 2). Now this cut must pass strictly below the root of $x\sigma$, or otherwise $x\sigma$ would be a subterm in $T_k\sigma$. Since all the terms corresponding to nodes above the normal cut are also derivable, we have that $x\sigma$ is derivable from $T_k\sigma$. Further, since the normal cut passes strictly below the root of $x\sigma$, $x\sigma$ must be of the form $op(t_1, t_2)$, and both $t_1$ and $t_2$ are derivable from $T_k\sigma$.

The term $m_k\sigma'$ is same as $m_k\sigma$ except that all the occurrences of $x\sigma$ are replaced by $\epsilon$. Consider the cut in $m_k\sigma'$ as shown in the figure. All the terms in this cut, possibly except $\epsilon$, are present in the normal cut of $m_k\sigma$ also. Since $\epsilon$ is always present in all $T_i$'s, all the terms in this cut are derivable from $T_k\sigma'$. Hence $m_k\sigma'$ is derivable from $T_k\sigma'$.

Now consider $m_i : T_i$ where $i > k$. As $\sigma$ is a solution to $\mathcal{C}$, we have $T_i\sigma \vdash m_i\sigma$. Since $x\sigma$ is derivable from $T_k\sigma$ and $i > k$, $T_i\sigma' \vdash x\sigma$. Lets look at a minimal derivation of $m_i\sigma$ from $T_i\sigma$. We will show that for every node $t$ in this derivation, $T_i\sigma' \vdash \delta(t)$. In particular $T_i\sigma' \vdash \delta(m_i\sigma) = m_i\sigma'$. We do this using induction on the height of the node $t$ in the derivation tree. For the base case, the leaves of the derivation tree are the terms in $T_i\sigma$. For every $t \in T_i$ we have $t\sigma' = \delta(t\sigma) \in T_i\sigma'$. Hence $T_i\sigma' \vdash \delta(t)$. For internal nodes $t$, we consider the inference rule used to derive this node.

- Case 1: Its a compositional rule. Let it be $SE$. If $t = x\sigma$, then $\delta(t) = \epsilon$ and $T_i\sigma' \vdash \delta(t)$. Otherwise $t = [t_1]_{t_2}$ and $t$ has two children $t_1$ and $t_2$. As

$t \neq x\sigma$, $\delta(t) = [\delta(t_1)]_{\delta(t_2)}$. By induction hypothesis $\delta(t_1), \delta(t_2)$ are derivable from $T_i\sigma'$. By applying $SE$ on $\delta(t_1), \delta(t_2)$ we derive $\delta(t)$. For other rules the proof goes through similarly.

– Case 2: Its a decomposition rule. Let it be $SD$. i.e. $t$ has two children $[t]_s$ and $s$. By induction hypothesis $\delta([t]_s), \delta(s)$ are derivable from $T_i\sigma'$. If $[t]_s \neq x\sigma$, $\delta([t]_s) = [\delta(t)]_{\delta(s)}$. By applying $SD$ on $[\delta(t)]_{\delta(s)},\ \delta(s)$, we derive $\delta(t)$. If $[t]_s = x\sigma$ then $\delta(t) = t$ and we need to show $T_i\sigma' \vdash t$. Recall that $x\sigma$ was of the form $op(t_1, t_2)$ and both $t_1$ and $t_2$ were derivable from $T_k\sigma'$. Hence $t$ equals $t_1$ and is derivable from $T_i\sigma'$.

This completes the proof that $\sigma'$ is a solution to $\mathcal{C}$.

Now consider the lowest cut in $m_i\sigma'$ (which passes through the nodes corresponding to leaves of $m_i$). It is easy to check that (1) all the nodes in the cut corresponding to the nodes in the cut in $m_i\sigma$ that had matches in the unsubstituted part of $\mathcal{C}\sigma$, continue to have such matches in $\mathcal{C}\sigma'$; and (2) we have at least one more node (corresponding to the variable $x$) which now has a match ($\epsilon$) which lies in the unsubstituted part of $\mathcal{C}\sigma'$.

Hence repeatedly following the above argument we can find a substitution $\sigma'$ satisfying the requirements of lemma 3. □

**Lemma 4.** *If $\mathcal{C}$ has a solution, then it has a solution $\sigma$ such that there exist cuts $c_i$ in each $m_i$, and a pairing $P$ of each node $u$ in a cut with a node $v$ in $\mathcal{C}$ different from $u$, such that $\sigma$ is a simultaneous pairwise mgu of the pairs of terms in the set $\widetilde{P} = \{(\widetilde{u}, \widetilde{v}) \mid (u, v) \in P\}$.*

*Proof.* From lemma 3 we have a solution $\sigma$ to $\mathcal{C}$ with cuts $c_i$ corresponding to the leaf nodes in each $m_i$, and a pairing $P$ of each node $u$ in a cut with a node $v$ in $\mathcal{C}$ different from $u$, such that $\sigma$ is a simultaneous pairwise unifier of the pairs of terms in $\widetilde{P}$. Now consider a partition of the set of terms in $\widetilde{P}$ where each block comprises terms that must be unified after considering the transitive closure of the pairs in $\widetilde{P}$. For instance, if $(x, t)$ and $(x, y)$ are in $\widetilde{P}$ then $x$, $y$, and $t$ all belong to the same block. Note that a substitution is a unifier of $\widetilde{P}$ iff it is a unifier of the terms in the blocks of this partition.

Now there are two cases. If every block of the partition contains a non-variable term, then it is easy to see that any unifier (in particular the mgu also) of these blocks must be a *concrete* substitution. Thus in this case, $\sigma$ is also the mgu of the terms in $\widetilde{P}$.

The other case is that there is a block which contains only variables. In this case, we can consider the substitution $\sigma'$ which is the same as $\sigma$ except that it sends each variable in such a block to $\epsilon$. By a similar argument to lemma 3 we can see that $\sigma'$ is also a solution, and further that it falls into the previous case above, and is hence an mgu of the corresponding pairs of terms $\widetilde{P'}$.

## 6 Comparison with Rusinowitch-Turuani

Rusinowitch and Turuani show that if there is a solution, there is one that substitutes each variable with a term with a concrete prefix which appears somewhere in the constraint set. They use this result to give a bound on the size of substitutions they need to look for. Their decision procedure generates all the substitutions of size less than this bound and then checks whether there is any solution among these substitutions. Our completeness proof gives an alternate proof of their result as follows. We show in the proof of Lemma 4 that if there is a solution then there is one for which the partition is such that every block contains a non-variable term, which is the required match for the variables in the block. The proof itself uses some ideas from the proof of Rusinowitch and Turuani, though the motivation and initial approach are different.

We also observe that the Examples 1 and 2 are also counter examples for the corresponding restrictions of the Rusinowitch-Turuani approach. For instance, Example 2 shows that the Rusinowitch-Turuani approach will become incomplete if we consider only $T_i$'s (send messages) for $\sigma$ matches of a variable, as in Example 2 the $\sigma$ match of $x$ appears only in a received message ($[[[a]_{k_1}]_{k_2}]_{k_3}$).

Conversely the Rusinowitch-Turuani result can also be used to argue the completeness of our approach. By their result $\mathcal{C}$ has a solution $\sigma$ such that for every variable $x$ there is a subterm $t_x$ in $\mathcal{C}$ such that $x\sigma = t_x\sigma$. i.e. $\sigma$ is a unifier for the pairs $(x, t_x)$. Similar to the arugment in lemma 4, it follows that the *mgu* for this set of pairs of terms is concrete, and hence coincides with $\sigma$. Hence we can guess the cut passing through leaves of the $m_i$'s, and pair every variable leaf $x$ with the corresponding $t_x$, and every constant leaf $a$ with the constant $a$, and then the most general unifier of this pairing will be $\sigma$.

## References


1. R. Amadio and D.Lugiez. On the reachability problem in cryptographic protocols. In *CONCUR 2000, LNCS 1877*, pages 380–394. Springer Verlag, 2000.
2. H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proc.18th IEEE Symposium on Logic in Computer Science*, Washington, DC, USA, 2003. IEEE Computer Society.
3. D. Dolev and A. C. Yao. On the security of public key protocols. Technical report, Stanford, CA, USA, 1981.
4. N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. *Wk. Formal Methods and Security Protocols, FLOC*, 1999.
5. A. Huima. Efficient infinite-state analysis of security protocols. In *Workshop on Formal Methods and Security Protocols, FLOC*, 1999.
6. Jonathan K. Millen and Vitaly Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *ACM Conference on Computer and Communications Security*, pages 166–175, 2001.
7. J. Tygar N. Heintze. A model for secure protocols and their compositions. *IEEE Trans. Software Engineering*, 22(1):16–30, 1996.
8. J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, 1965.
9. M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. *14th IEEE Computer Security Foundations Workshop*, pages 174–190, 2001.