INSTITUT FÜR INFORMATIK

UND PRAKTISCHE MATHEMATIK

A Constraint-Based Algorithm for Contract-Signing Protocols

Detlef Kähler, Ralf Küsters

Bericht Nr. 0503
April 2005

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

KIEL

Institut für Informatik und Praktische Mathematik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

A Constraint-Based Algorithm for Contract-Signing Protocols

Detlef Kähler, Ralf Küsters

Bericht Nr. 0503
April 2005

e-mail: {kaehler,kuesters}@ti.informatik.uni-kiel.de

A Constraint-Based Algorithm for Contract-Signing Protocols

Detlef Kähler and Ralf Küsters

Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität zu Kiel, Germany
{kaehler,kuesters}@ti.informatik.uni-kiel.de

Abstract

Research on the automatic analysis of cryptographic protocols has so far mainly concentrated on reachability properties, such as secrecy and authentication. Only recently it was shown that certain game-theoretic security properties, such as balance for contract-signing protocols, are decidable in a Dolev-Yao style model with a bounded number of sessions but unbounded message size. However, this result does not provide a practical algorithm as it merely bounds the size of attacks. In this paper, we prove that game-theoretic security properties can be decided based on standard constraint solving procedures. This paves the way for extending existing implementations and tools for reachability properties to deal with game-theoretic security properties.

1 Introduction

One of the central results in the area of automatic analysis of cryptographic protocols is that the security of cryptographic protocols is decidable when analyzed w.r.t. a finite number of sessions, without a bound on the message size, and in presence of the so-called Dolev-Yao intruder (see, e.g., [13, 1]). Based on this result, many fully automatic tools (see, e.g., [2, 7, 12]) have been developed and successfully applied to find flaws in published protocols, where most of these tools employ so-called *constraint solving procedures* (see, e.g., [12, 7, 4]). However, the mentioned decidability result and tools are restricted to security properties such as authentication and secrecy which are reachability properties of the transition system associated with a given protocol. In contrast, crucial properties required of contract-signing and related protocols (see, e.g., [9, 3]), for instance abuse-freeness [9] and balance [5], are game-theoretic properties of the structure of the transition system associated with a protocol. Balance, for instance, requires

that in no stage of a protocol run, the intruder or a dishonest party has both a strategy to abort the run and a strategy to successfully complete the run and thus obtain a valid contract.

Only recently [10], the central decidability result mentioned above was extended to such game-theoretic security properties, including, for instance, balance. However, the decision algorithm presented there is merely based on the fact that the size of attacks can be bounded, and hence, all potential attacks up to a certain size have to be enumerated and checked. Clearly, this is completely impractical.

The main contribution of the present work is a *constraint-based* decision algorithm for the game-theoretic security properties of the kind considered in [10]. The main feature of our algorithm is that it can be built on top of *standard constraint solving procedures* (see, e.g., [12, 7, 4] and references therein). As mentioned, such procedures have successfully been employed for reachability properties in the past and proved to be a good basis for practical implementations. Hence, our algorithm paves the way for extending existing implementations and tools for reachability properties to deal with game-theoretic security properties.

In a nutshell, our constraint-based algorithm works as follows: Given a protocol along with the considered game-theoretic security property, first the algorithm guesses what we call a symbolic branching structure. This structure represents a potential attack on the protocol and corresponds to the interleavings, which are, however, linear structures, guessed for reachability properties. In the second step of the algorithm, the symbolic branching structure is turned into a so-called constraint system. This step requires some care due to the branching issue and write-protected channels considered in our model (also called secure channels here), i.e., channels that are not under the control of the intruder. Then, a standard constraint solving procedure (see above) is used to compute a finite sound and complete set of so-called simple constraint systems. A simple constraint system in such a set represents a (possibly infinite) set of solutions of the original constraint system and the sound and complete set of these simple constraint systems represents the set of *all* solutions of the original constraint system. Finally, it is checked whether (at least) one of the computed simple constraint systems in the sound and complete set passes certain additional tests.

There are some crucial differences of our constraint-based algorithm to algorithms for reachability properties: First, as mentioned, instead of symbolic branching structures for reachability properties only interleavings, i.e., linear structures, need to be guessed. Turning these interleavings into constraint systems is immediate due to the absence of the branching issue and the absence of secure channels. Second, and more importantly, for reachability properties it suffices if the constraint solving procedure only returns one simple constraint system, rather than a sound and complete set. Third, the final step of our constraint-based algorithm—performing additional tests on the simple constraint system—is not required for reachability properties.

We emphasize that even though for reachability properties it suffices if the constraint

solving procedure returns only one simple constraint system, standard constraint solving procedures are typically capable of computing sound and complete sets of simple constraint systems. Any such procedure can be used by our constraint-based algorithm as a black-box for solving constraint systems. This makes it possible to extend existing implementations and tools for reachability properties to deal with game-theoretic properties since the core of the algorithms—solving constraint systems—remains the same, provided that the considered cryptographic primitives can be dealt with by the constraint solving procedure (see Section 4).

The protocol and intruder model that we use is basically the one proposed in [10], which in turn is the "bounded session" version of a model proposed in [5]. We slightly modify the model of [10]—without changing its expressivity and accuracy—in order to simplify our constraint-based algorithm (see Section 2 and 3). For instance, while in [10] intruder strategies are positional, it turns out that the constraint-based algorithm is considerably simpler for intruder strategies which may depend on the history of the protocol run. However, it is not hard to show that both notions of strategies are equivalent in our setting, and hence, we can w.l.o.g. choose the notion of strategy that fits best for our purpose.

Further related work. Contract-signing and related protocols have been analyzed both manually [5], based on a relatively detailed model (as mentioned, our model is a "bounded session" version of this model), and using finite-state model checking (see, e.g., [14, 11]), based on a coarser finite-state model. Drielsma and Mödersheim [8] were the first to apply an automatic tool based on constraint solving to the contract-signing protocol by Asokan, Shoup, and Waidner [3]. Their analysis is, however, restricted to reachability properties since game-theoretic properties cannot be handled by their tool. The results shown in the present work pave the way for extending such tools in order to be able to analyze game-theoretic properties.

Structure of this paper. In Section 2 we recall the protocol and intruder model and in Section 3 the intruder strategies and the game-theoretic properties first introduced in [10], and point out the mentioned differences. Section 4 provides the necessary background on constraint solving. In Section 5, we present our constraint-based decision algorithm along with an example and state our main result—soundness, completeness, and termination of the algorithm, with the proof given in Section 6. We conclude in Section 7.

2 The Protocol and Intruder Model

Our protocol and intruder model that we use basically coincides with the model first introduced in [10], which in turn is the "bounded session" version of the model proposed

in [5]. We only slightly modify the model in [10] in that we impose a restriction on principals which is necessary for principals to perform feasible computations. We refer the reader to [10] for more details on the model and examples.

In our model, a protocol is a finite set of principals and every principal is a finite tree, which represents all possible behaviours of the principal. Each edge of such a tree is labeled by a rewrite rule, which describes the receive-send action that is performed when the principal takes this edge in a run of the protocol.

When a principal carries out a protocol, it traverses its tree, starting at the root. In every node, the principal takes its current input, chooses one of the edges leaving the node, matches the current input with the left-hand side of the rule the edge is labeled with, sends out the message which is determined by the right-hand side of the rule, and moves to the node the chosen edge leads to. While in the standard Dolev-Yao model (see, e.g., [13]) inputs to principals are always provided by the intruder, in our model it can also come from the secure channel, which the intruder does not control, i.e., the intruder cannot delay, duplicate, remove messages, or write messages onto this channel under a fake identity (unless he has corrupted a party). However, just as in [5], the intruder can read the messages written onto the secure channel. We note that our results also hold in case of read-protected secure channels. Another difference to standard Dolev-Yao models is that, in order to be able to formulate game-theoretic properties, we explicitly describe the behavior of a protocol as an infinite-state transition graph which comprises all runs of a protocol.

2.1 Terms and Messages

We have a finite set $\mathcal{V}$ of variables, a finite set $\mathcal{A}$ of atoms, a finite set $\mathcal{K}$ of public and private keys, an infinite set $\mathcal{A}_I$ of intruder atoms, and a finite set $\mathcal{N}$ of principal addresses. All of them are assumed to be disjoint.

The set $\mathcal{K}$ is partitioned into a set $\mathcal{K}_{pub}$ of public keys and a set $\mathcal{K}_{priv}$ of private keys. There is a bijective mapping $\cdot^{-1}\colon \mathcal{K} \to \mathcal{K}$ which assigns to every public key the corresponding private key and to every private key its corresponding public key.

Typically, the set $\mathcal{A}$ contains names of principals, atomic symmetric keys, and nonces (i.e., random numbers generated by principals). We note that we will allow non-atomic symmetric keys as well. The atoms in $\mathcal{A}_I$ are the nonces, symmetric keys, etc. the intruder may generate. The elements of $\mathcal{N}$ are used as addresses of principals in secure channels.

We define two kinds of terms by the following grammar, namely *plain terms* and

secure channel terms:

$$\begin{array}{rcl} \textit{plain-terms} & ::= & \mathcal{V} \mid \mathcal{A} \mid \mathcal{A}_I \mid \langle \textit{plain-terms}, \textit{plain-terms} \rangle \mid \\ & & \{\textit{plain-terms}\}^s_{\textit{plain-terms}} \mid \{\textit{plain-terms}\}^a_{\mathcal{K}} \mid \mathsf{hash}(\textit{plain-terms}) \mid \\ & & \mathsf{sig}_{\mathcal{K}}(\textit{plain-terms}) \\ \textit{sec-terms} & ::= & \mathsf{sc}(\mathcal{N}, \mathcal{N}, \textit{plain-terms}) \\ \textit{terms} & ::= & \textit{plain-terms} \mid \textit{sec-terms} \mid \mathcal{N} \end{array}$$

Plain terms, secure channel terms, and terms without variables (i.e., ground terms) are called *plain messages*, *secure channel messages*, and *messages*, respectively. As usual, $\langle t, t' \rangle$ is the pairing of t and t', the term $\{t\}^s_{t'}$ stands for the symmetric encryption of t by t' (note that the key t' may be any plain term), $\{t\}^a_k$ is the asymmetric encryption of t by k, the term $\mathsf{hash}(t)$ stands for the hash of t, and $\mathsf{sig}_k(t)$ is the signature on t which can be verified with the public key k.

A secure channel term of the form $\mathsf{sc}(n, n', t)$ stands for feeding the secure channel from n to n' with t. A principal may only generate such a term if he knows n and t (but not necessarily n'). This guarantees that a principal cannot impersonate other principals on the secure channel. Knowing n grants access to secure channels with sender address n. Let $\mathcal{V}(t)$ denote the set of variables occurring in the term t.

The set $\mathrm{Sub}(t)$ of subterms of t and the set $\mathrm{Sub}(E)$ of subterms of a set E of terms is defined in the obvious way.

A *(ground) substitution* assigns (ground) terms to variables. The *domain* of a substitution is denoted by $\mathsf{dom}(\sigma)$ and defined by $\mathsf{dom}(\sigma) = \{x \in \mathcal{V} \mid \sigma(x) \neq x\}$. Substitutions are required to have finite domains. Given two substitutions σ and σ' with disjoint domains, their union $\sigma \cup \sigma'$ is defined in the obvious way. Given a term t, the term $t\sigma$ is obtained from t by simultaneously substituting each variable x occurring in t by $\sigma(x)$. For substitutions σ and τ we define $(\tau \circ \sigma)(x) = \tau(\sigma(x))$ for every x.

The set of subterms of a term is defined as expected. We say that t is used as a *verification key* in t' if t' has a subterm of the form $\{s\}^s_t$, $\{s\}^a_{t^{-1}}$, or $\mathsf{sig}_t(s)$ for some term s. Intuitively, a verification key is needed to decrypt messages or verify the signature. For example, if a principal A receives the message $m = \{\langle A, x \rangle\}^s_a$ and wants to check whether the plain text matches with $t' = \langle A, x \rangle$, then the principal first needs to decyrpt m with the verification key a.

2.2 Intruder

Given a set $\mathcal{I}$ of general messages, the (infinite) set $d(\mathcal{I})$ of general messages the intruder can derive from $\mathcal{I}$ is the smallest set satisfying the following conditions:

1. $\mathcal{I} \subseteq d(\mathcal{I})$.

2. *Composition and decomposition*: If $m, m' \in d(\mathcal{I})$, then $\langle m, m' \rangle \in d(\mathcal{I})$. Conversely, if $\langle m, m' \rangle \in d(\mathcal{I})$, then $m \in d(\mathcal{I})$ and $m' \in d(\mathcal{I})$.

3. *Symmetric encryption and decryption*: If $m, m' \in d(\mathcal{I})$, then $\{m\}^s_{m'} \in d(\mathcal{I})$. Conversely, if $\{m\}^s_{m'} \in d(\mathcal{I})$ and $m' \in d(\mathcal{I})$, then $m \in d(\mathcal{I})$.

4. *Asymmetric encryption and decryption*: If $m \in d(\mathcal{I})$ and $k \in d(\mathcal{I}) \cap \mathcal{K}$, then $\{m\}^a_k \in d(\mathcal{I})$. Conversely, if $\{m\}^a_k \in d(\mathcal{I})$ and $k^{-1} \in d(\mathcal{I})$, then $m \in d(\mathcal{I})$.

5. *Hashing*: If $m \in d(\mathcal{I})$, then $\mathsf{hash}(m) \in d(\mathcal{I})$.

6. *Signing*: If $m \in d(\mathcal{I}), k^{-1} \in d(\mathcal{I}) \cap \mathcal{K}$, then $\mathsf{sig}_k(m)$. (The signature contains the public key but can only be generated if the corresponding private key is known.)

7. *Writing onto the secure channel:* If $m \in d(\mathcal{I})$, $n \in d(\mathcal{I}) \cap \mathcal{N}$, and $n' \in \mathcal{N}$, then $\mathsf{sc}(n, n', m) \in d(\mathcal{I})$.

8. *Generating fresh constants*: $\mathcal{A}_I \subseteq d(\mathcal{I})$.

Each of the above rules only applies when the resulting expression is a term according to the grammar stated above. For instance, a hash of a secure channel term is not a term, so rule 5 does not apply when m is of the form $\mathsf{sc}(n, n', m')$.

Intuitively, $n \in d(\mathcal{I}) \cap \mathcal{N}$ means that the intruder has corrupted the principal with address n and therefore can impersonate this principal when writing onto the secure channel.

2.3 Principals and Protocols

Principal rules are of the form $R \Rightarrow S$ where R is a term or ε and S is a term.

A *rule tree* $\Pi = (V, E, r, \ell)$ is a finite tree rooted at $r \in V$ where ℓ maps every edge $(v, v') \in E$ of Π to a principal rule $\ell(v, v')$.

A *principal* is a tuple consisting of a rule tree $\Pi = (V, E, r, \ell)$ and a set $\mathcal{I}_\Pi$ of plain messages, the *initial knowledge of principal* Π such that the following two conditions are satisfied for every $v_0 = r, v_1, \ldots, v_n \in V$ with $(v_i, v_{i+1}) \in E$ and $\ell(v_i, v_{i+1}) = R_i \Rightarrow S_i$:

1. $\mathcal{V}(S_{n-1}) \subseteq \bigcup_{j<n} \mathcal{V}(R_j)$.

2. $t \in d(\{R_0, \ldots, R_{n-1}\} \cup \mathcal{I}_\Pi)$ for every verification key t in R_{n-1} where the variables in $\{R_0, \ldots, R_{n-1}\}$ are considered as atoms.

The first condition says that every variable occurring on the right-hand side of a principal rule $\ell(v, v')$ also occurs on the left-hand side of $\ell(v, v')$ or on the left-hand side of a principal rule on the path from r to v. This condition is standard in Dolev-Yao models

[13, 12] and guarantees that the output produced by a principal depends on the input and cannot be arbitrary.

The second condition says that the decryption and signature verification operations performed by Π when receiving a message are feasible and it is a necessary (not sufficient) condition for the principal to only perform feasible computations. We therefore call this condition the *feasibility condition for principals*. For example, this condition is not satisfied for a principal whose rule tree only contains one edge and this edge is labeled with the principal rule $\{y\}^s_x \Rightarrow x$ since x cannot be derived from the left-hand side of the rule. In fact, this rule allows the principal to extract the encryption key of every ciphertext, which clearly is infeasible. For instance, given $\{a\}^s_b$ as input the principal would output b.

For $v \in V$, we write $\Pi{\downarrow}v$ to denote the subtree of Π rooted at v. For a substitution σ, we write $\Pi\sigma$ for the principal obtained from Π by substituting all variables x occurring in the principal rules of Π by $\sigma(x)$.

A *protocol* $P = ((\Pi_1, \ldots, \Pi_n), \mathcal{I})$ consists of a finite set of principals and a finite set $\mathcal{I}$ of plain messages and principal addresses, the *initial intruder knowledge*. We require that each variable occurs in the rules of only one principal, i.e., different principals must have disjoint sets of variables. We assume that intruder atoms, i.e., elements of $\mathcal{A}_I$, do not occur in P.

2.4 Transition Graphs Induced by a Protocol

A transition graph $\mathcal{G}_P$ induced by a protocol P comprises all runs of a protocol. To define this graph, we first introduce states and transitions between these states.

A *state* is of the form $((\Pi_1, \ldots, \Pi_n), \sigma, \mathcal{I}, \mathcal{S})$ where

1. σ is a substitution,
2. for each i, Π_i is a rule tree such that $\Pi_i\sigma$ is a principal,
3. $\mathcal{I}$ is a finite set of messages, the *intruder knowledge*, and
4. $\mathcal{S}$ is a finite multi-set of secure channel messages, the *secure channel*.

The idea is that when the transistion system gets to such a state, then the substitution σ has been performed, the accumulated intruder knowledge is what can be derived from $\mathcal{I}$, the secure channels hold the messages in $\mathcal{S}$, and for each i, Π_i is the "remaining protocol" to be carried out by principal i. This also explains why $\mathcal{S}$ is a multi-set: messages sent several times should be delivered several times.

Given a protocol $P = ((\Pi_1, \ldots, \Pi_n), \mathcal{I})$ the *initial state of* P is set to be $((\Pi_1, \ldots, \Pi_n), \sigma, \mathcal{I}, \emptyset)$ where σ is the substitution with empty domain.

We have three kinds of transitions: intruder, secure channel, and ε-transitions. In what follows, let $\Pi_i = (V_i, E_i, r_i, \ell_i)$ and $\Pi'_i = (V'_i, E'_i, r'_i, \ell'_i)$ denote rule trees. We define under which circumstances there is a transition

$$((\Pi_1, \ldots, \Pi_n), \sigma, \mathcal{I}, \mathcal{S}) \xrightarrow{\tau} ((\Pi'_1, \ldots, \Pi'_n), \sigma', \mathcal{I}', \mathcal{S}') \tag{1}$$

with τ an appropriate label.

1. *Intruder transitions:* The transition (1) with label i, m, I exists if there exists $v \in V_i$ with $(r_i, v) \in E_i$ and $\ell_i(r_i, v) = R \Rightarrow S$, and a substitution σ'' of the variables in $R\sigma$ such that

   (a) $m \in d(\mathcal{I})$,

   (b) $\sigma' = \sigma \cup \sigma''$,

   (c) $R\sigma' = m$,

   (d) $\Pi'_j = \Pi_j$ for every $j \neq i$, $\Pi'_i = \Pi_i{\downarrow}v$,

   (e) $\mathcal{I}' = \mathcal{I} \cup \{S\sigma'\}$ if $S \neq \mathsf{sc}(\cdot, \cdot, \cdot)$, and $\mathcal{I}' = \mathcal{I} \cup \{t\sigma'\}$ if $S = \mathsf{sc}(\cdot, \cdot, t)$ for some t.

   (f) $\mathcal{S}' = \mathcal{S}$ if $S \neq \mathsf{sc}(\cdot, \cdot, \cdot)$, and $\mathcal{S}' = \mathcal{S} \cup \{S\sigma'\}$ otherwise.

   This transition models that principal i reads the message m from the intruder (i.e., the public network).

2. *Secure channel transitions:* The transition (1) with label i, m, sc exists if there exists $v \in V_i$ with $(r_i, v) \in E_i$ and $\ell_i(r_i, v) = R \Rightarrow S$, and a substitution σ'' of the variables in $R\sigma$ such that $m \in \mathcal{S}$, (b)–(e) from 1., and $\mathcal{S}' = \mathcal{S} \setminus \{m\}$ if $S \neq \mathsf{sc}(\cdot, \cdot, \cdot)$, and $\mathcal{S}' = (\mathcal{S} \setminus \{m\}) \cup \{S\sigma'\}$ otherwise.

   This transition models that principal i reads message m from the secure channel.

3. *ε-transitions:* The transition (1) with label i exists if there exists $v \in V_i$ with $(r_i, v) \in E_i$ and $\ell_i(r_i, v) = \varepsilon \Rightarrow S$ such that $\sigma' = \sigma$ and (d), (e), (f) from above.

   This transition models that i performs a step where neither a message is read from the intruder nor from the secure channel.

We call i the *principal associated with the transition*, $R \Rightarrow S$ (for the intruder and secure channel transitions) and $\varepsilon \Rightarrow S$ (for the ε-transitions) the *principal rule associated with the transition*, and v the *principal vertex associated with the transition.*

Definition 1 *Given a protocol P, the* transition graph $\mathcal{G}_P$ induced by P *is the tuple (S_P, E_P, q_P) where q_P is the initial state of P, S_P is the set of states reachable from q_P by a sequence of transitions, and E_P is the set of all transitions among states in S_P.*

We write $q \in \mathcal{G}_P$ if q is a state in $\mathcal{G}_P$ and $q \xrightarrow{\tau} q' \in \mathcal{G}_P$ if $q \xrightarrow{\tau} q'$ is a transition in $\mathcal{G}_P$.

Remark 2 *The transition graph $\mathcal{G}_P$ of P is a DAG since by performing a transition the size of the first component of a state decreases. While the graph may be infinite branching, the maximal length of a path in this graph is bounded by the total number of edges in the principals Π_i of P.*

3 Intruder Strategies and Strategy Properties

In this section, we define intruder strategies on transition graphs and the goal the intruder tries to achieve following his strategy.

As mentioned in the introduction, while in [10] positional intruder strategies have been considered where the strategy of the intruder at a global state of the protocol run may only depend on the current state, here we allow the intruder to take the whole path leading to this state (i.e., the history) into account. While this potentially provides the intruder with more power, these notions are in fact equivalent (see Proposition 4). The main motivation for the new notion of strategy is that it is much better suited for the constraint solving approach (Section 5).

To define intruder strategies, we introduce the notion of a strategy tree, which captures that the intruder has a way of acting such that regardless of how the other principals act he achieves a certain goal, where goal in our context means that a state will be reached where the intruder can derive certain constants and cannot derive others, e.g., for balance, the intruder tries to obtain IntruderHasContract but tries to prevent HonestPartyHasContract from occurring (see [10] for more details).

Definition 3 *For $q \in \mathcal{G}_P$ a* q-strategy tree *$\mathcal{T}_q = (V, E, r, \ell_V, \ell_E)$ is an unordered tree where every vertex $v \in V$ is mapped to a state $\ell_V(v) \in \mathcal{G}_P$ and every edge $(v, v') \in E$ is mapped to a label of a transition such that the following conditions are satisfied for all $v, v' \in V$, principals j, messages m, and states q', q'':*

1. $\ell_V(r) = q$.
2. $\ell_V(v) \xrightarrow{\ell_E(v,v')} \ell_V(v') \in \mathcal{G}_P$ *for all* $(v, v') \in E$.
3. *If* $\ell_V(v) = q'$ *and* $q' \xrightarrow{j} q'' \in \mathcal{G}_P$, *then there exists* $v'' \in V$ *such that* $(v, v'') \in E$, $\ell_V(v'') = q''$, *and* $\ell_E(v, v'') = j$.
4. *If* $\ell_V(v) = q'$ *and* $q' \xrightarrow{j,m,\mathsf{sc}} q'' \in \mathcal{G}_P$, *then there exists* $v'' \in V$ *such that* $(v, v'') \in E$, $\ell_V(v'') = q''$, *and* $\ell_E(v, v'') = j, m, \mathsf{sc}$.
5. *If* $(v, v') \in E$, $\ell_E(v, v') = j, m, I$, *and there exists* $q'' \neq \ell_V(v')$ *with* $\ell_V(v) \xrightarrow{j,m,I} q'' \in \mathcal{G}_P$, *then there exists* v'' *with* $(v, v'') \in E$, $\ell_E(v, v'') = j, m, I$ *and* $\ell_V(v'') = q''$.

The second condition in the above definition guarantees that all edges in $\mathcal{T}_q$ correspond to transitions in $\mathcal{G}_P$. The third condition says that every ε-transition of the original transition system must be present in the strategy graph; this is because the intruder should not be able to prevent a principal from performing an ε-rule. The fourth condition is similar: the intruder should not be able to block secure channels. The fifth condition says that although the intruder can choose to send a particular message to a particular principal, he cannot decide which transition this principal uses (if the message matches two rules).

Note that if $\ell_V(v) = q'$ in a q-strategy tree $\mathcal{T}_q$, then there exists a path from q to q' in $\mathcal{G}_P$. By Remark 2 this implies that $\mathcal{T}_q$ has finite depth.

Let $C, C' \subseteq \mathcal{A} \cup \mathcal{K} \cup \mathcal{N}$. We say that a q-strategy tree $\mathcal{T}_q$ satisfies (C, C') if for all leaves v of $\mathcal{T}_q$ all elements from C_i can be derived by the intruder and all elements from C' cannot, i.e., $C \subseteq d(\mathcal{I})$ and $C' \cap d(\mathcal{I}) = \emptyset$ where $\mathcal{I}$ denotes the intruder knowledge in state $\ell_V(v)$.

A *strategy property* is a tuple $((C_1, C_1'), \ldots, (C_s, C_s'))$ where $C_i, C_i' \subseteq \mathcal{A} \cup \mathcal{K} \cup \mathcal{N}$. A state $q \in \mathcal{G}_P$ satisfies $((C_1, C_1'), \ldots, (C_s, C_s'))$ if there exist q-strategy trees $\mathcal{T}_1, \ldots, \mathcal{T}_s$ such that every $\mathcal{T}_i$ satisfies (C_i, C_i').

The decision problem G-STRATEGY *(with general intruder strategy)* asks, given a protocol P and a strategy property $((C_1, C_1'), \ldots, (C_s, C_s'))$, whether there exists a state $q \in \mathcal{G}_P$ that satisfies the property. In this case we write $(P, (C_1, C_1'), \ldots, (C_s, C_s')) \in$ G-STRATEGY.

Note that in a q-strategy tree $\mathcal{T}_q$ there may exist vertices $v' \neq v$ with $\ell_V(v') = \ell_V(v)$ such that the subtrees $\mathcal{T}_q{\downarrow}v$ and $\mathcal{T}_q{\downarrow}v'$ of $\mathcal{T}_q$ rooted at v and v', respectively, are not isomorphic. In other words, the intruder's strategy may depend on the path that leads to a state (i.e., the history) rather than on the state alone, as is the case for positional strategies.

As mentioned, the strategies defined in [10] are positional and can therefore be defined as subgraphs of $\mathcal{G}_P$. This is equivalent to postulating that in q-strategy trees $\mathcal{T}_q$ different subtrees rooted at vertices with the same vertex label are isomorphic. Following common terminology, we call such a q-strategy tree *positional*. Now, we define P-STRATEGY analogously to G-STRATEGY with positional intruder strategies.

The following proposition shows that as far as the strategy properties are concerned there is no difference between positional and non-positional strategies.

Proposition 4 $(P, (C_1, C_1'), \ldots, (C_l, C_l')) \in$ G-STRATEGY *iff* $(P, (C_1, C_1'), \ldots, (C_l, C_l')) \in$ P-STRATEGY

PROOF. It suffices to show that for every $q \in \mathcal{G}_P$ and $(C, C') \subseteq \mathcal{A} \cup \mathcal{K} \cup \mathcal{N}$ there exists a q-strategy tree which satisfies (C, C') iff there exists a positional q-strategy tree which satisfies (C, C'). The direction from right to left is trivial. For the other direction, let $\mathcal{T}_q$ be a q-strategy tree which satisfies (C, C'). If there exist $v' \neq v$ in $\mathcal{T}_q$ with

$\ell_V(v) = \ell_V(v') = q'$, then it is not hard to see that the tree obtained by replacing $\mathcal{T}_q{\downarrow}v'$ with $\mathcal{T}_q{\downarrow}v$ (and consistently renaming the vertices) is still a q-strategy tree satisfying (C, C'). Now, iteratively starting with the subtrees of maximum depth and replacing all subtrees rooted at vertices labeled with the same state by one of the subtrees among them, we obtain the desired positional q-strategy tree. □

In [10] it was shown that p-Strategy is decidable. As an immediate corollary of Proposition 4 we obtain:

Corollary 5 g-Strategy *is decidable.*

4 Constraint Solving

In this section, we introduce constraint systems and state the well-known fact that procedures for solving these systems exist (see, e.g., [12] for more details). In Section 5, we will then use such a procedure as a black-box for our constraint-based algorithm.

A *constraint* is of the form $t : T$ where t is a plain term and T is a finite non-empty set of plain terms. Since we will take care of secure channel terms outside of constraint solving, we can disallow secure channel terms in constraints.

A *constraint system* $\mathbf{C}$ is a tuple consisting of a sequence $s = t_1 : T_1, \ldots, t_n : T_n$ of constraints and a substitution τ such that the domain of τ is disjoint from the set of variables occurring in s and $\tau(x)$, for all x, only contains variables also occurring in s. We call $\mathbf{C}$ *simple* if t_i is a variable for all i.

A ground substitution σ where the domain of σ is the set of variables in $t_1 : T_1, \ldots, t_n : T_n$ is a *solution* of $\mathbf{C}$ ($\sigma \vdash \mathbf{C}$) if $t_i\sigma \in d(T_i\sigma)$ for every i. We call $\sigma \circ \tau$ (the composition of σ and τ read from right to left) a *complete* solution of $\mathbf{C}$ ($\sigma \circ \tau \vdash_c \mathbf{C}$) with τ as above.

For a variable x in $\mathbf{C}$, define $occ(x) \in \{1, \ldots, n\}$ to be the minimal index such that $x \in \mathcal{V}(t_{occ(x)})$, i.e., $occ(x)$ is the index of the first constraint in the sequence where x occurs on the left-hand side of this constraint.

We call a constraint system $\mathbf{C}$ *valid* if it satisfies the origination and monotonicity property as defined in [12], i.e.:

1. $\mathcal{V}(T_i) \subseteq \mathcal{V}(\{t_1, \ldots, t_{i-1}\})$.
2. $T_1 \subseteq T_i$ for every $i = 1, \ldots, n$.
3. For every i and $x \in \mathcal{V}(T_i)$, there exists $T \subseteq T_i$ such that $\mathcal{V}(T) \cap (\mathcal{V}(\{t_{occ(x)}, \ldots, t_n\}) \setminus \mathcal{V}(\{t_1, \ldots, t_{occ(x)-1}\}) = \emptyset$ and $d(T_{occ(x)}\sigma) \subseteq d(T\sigma)$ for every $\sigma \vdash \mathbf{C}$.

These are standard restrictions on constraint systems imposed by constraint solving procedures. The first condition, the origination property, corresponds to the first condition

imposed on principals. The second and third condition form the monotonicity property. The third condition captures the fact that the intruder does not forget information.

A simple constraint system $\mathbf{C}$ obviously has a solution. One such solution, which we denote by σ_C, replaces all variables in $\mathbf{C}$ by new intruder atoms $a \in \mathcal{A}_I$ where different variables are replaced by different atoms. We call σ_C the *solution associated with* $\boldsymbol{C}$ and $\sigma_C \circ \tau$ the *complete solution associated with* $\boldsymbol{C}$.

Given a constraint system $\mathbf{C}$, a finite set $\{\mathbf{C}_1, \ldots, \mathbf{C}_n\}$ of simple constraint systems is called a *sound and complete solution set for* $\boldsymbol{C}$ if $\{\nu \mid \nu \vdash_c \mathbf{C}\} = \{\nu \mid \exists i \text{ s.t. } \nu \vdash_c \mathbf{C}_i\}$. Note that $\mathbf{C}$ does not have a solution iff $n = 0$.

The following fact is well-known (see, e.g., [7, 12, 4] and references therein):

Fact 1 *There exists a procedure which given a valid constraint system* $\boldsymbol{C}$ *outputs a sound and complete solution set for* $\boldsymbol{C}$.

While different constraint solving procedures (and implementations thereof) may compute different sound and complete solution sets, our constraint-based algorithm introduced in Section 5 works with any of these procedures. It is only important that the set computed is sound and complete. As already mentioned in the introduction, to decide reachability properties it suffices if the procedure only returns one simple constraint system in the sound and complete set. However, the constraint solving procedures proposed in the literature are typically capable of returning a sound and complete solution set.

In what follows, we fix one such procedure and call it the *constraint solver*. More precisely, w.l.o.g., we consider the constraint solver to be a non-deterministic algorithm which non-deterministically chooses a simple constraint system from the sound and complete solution set and returns this system as output. We require that for every simple constraint system in the sound and complete solution set, there is a run of the constraint solver that returns this system. If the sound and complete set is empty, the constraint solver always returns no.

We note that while standard constraint solving procedures can deal with the cryptographic primitives considered here, these procedures might need to be extended when adding further cryptographic primitives. For example, this is the case for private contract signatures, which are used in some contract signing protocols [9] and were taken into account in [10]. However, constraint solving procedures can easily be extended to deal with these signatures. We have not considered them here for brevity of presentation and since the main focus of the present work is not to extend constraint solving procedures but to show how these procedures can be employed to deal with game-theoretic security properties.

5 The Constraint-Based Algorithm

We now present our constraint-based algorithm, called SolveStrategy, for deciding problem G-STRATEGY. As mentioned, it uses a standard constraint solver (Fact 1) as a subprocedure.

In what follows, we present the main steps performed by SolveStrategy, with more details given in subsequent sections. The input to SolveStrategy is a protocol P and a strategy property $((C_1, C_1'), \ldots, (C_l, C_l'))$.

1. Guess a symbolic branching structure $\mathbf{B}$, i.e., guess a symbolic path π^s from the initial state of P to a symbolic state q^s and a symbolic q^s-strategy tree $\mathcal{T}^s_{i,q^s}$ for every (C_i, C_i') starting from this state (see Section 5.1 for more details).

2. Derive from the symbolic branching structure $\mathbf{B} = \pi^s, \mathcal{T}^s_{1,q^s}, \ldots, \mathcal{T}^s_{l,q^s}$ and the strategy property $((C_1, C_1'), \ldots, (C_l, C_l'))$ the induced and valid (!) constraint system $\mathbf{C} = \mathbf{C_B}$ (see Section 5.2 for the definition). Then, run the constraint solver on $\mathbf{C}$. If it returns no, then halt. Otherwise, let $\mathbf{C}'$ be the simple constraint system returned by the solver. (Recall that $\mathbf{C}'$ belongs to the sound and complete solution set and is chosen non-deterministically by the solver.)

3. Let ν be the complete solution associated with $\mathbf{C}'$. Check whether ν when applied to $\mathbf{B}$ yields a valid path in $\mathcal{G}_P$ from the initial state of P to a state q and q-strategy trees $\mathcal{T}_{i,q}$ satisfying (C_i, C_i') for every i. If so, output yes and $\mathbf{B}$ with ν applied, and otherwise return no (see Section 5.3 for more details).

   Note that the above state, path, and strategy trees are not symbolic anymore and that in case yes is returned, $\mathbf{B}$ with ν applied yields a concrete solution of the problem instance $(P, (C_1, C_1'), \ldots, (C_l, C_l'))$.

We emphasize that, for simplicity of presentation, SolveStrategy is formulated as a nondeterministic algorithm. Hence, the overall decision of SolveStrategy is yes if there exists at least one computation path where yes is returned. Otherwise, the overall decision is no (i.e., $(P, (C_1, C_1'), \ldots, (C_l, C_l')) \notin$ G-STRATEGY).

In the following three sections, the three steps of SolveStrategy are further explained. Our main result is the following theorem, with the proof presented in Section 6:

Theorem 6 *SolveStrategy is a decision procedure for* G-STRATEGY.

While we know that G-STRATEGY is decidable from Corollary 5, the main point of Theorem 6 is that SolveStrategy uses standard constraint solving procedures as a black-box, and as such, is a good basis for extending existing practical constraint-based algorithms for reachability properties to deal with game-theoretic security properties.

The proof of Theorem 6 (see Section 6) is quite different from the cut-and-paste argument in [10] where, similar to [13], it was shown that an attack can be turned into

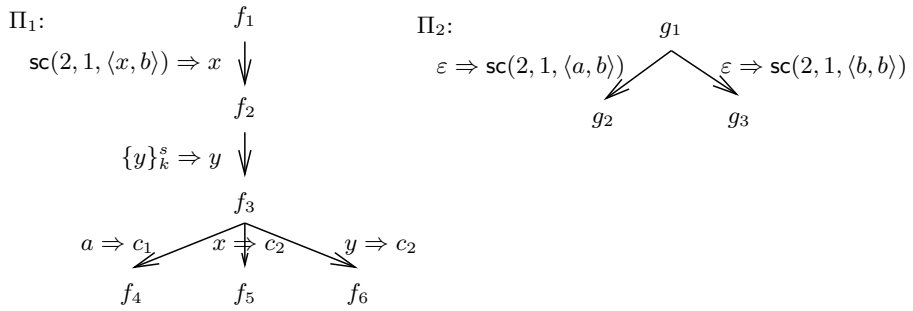

Figure 1: Protocol $P_{ex} = (\{\Pi_1, \Pi_2\}, \mathcal{I}_0)$ with $\mathcal{I}_0 = \{\{a\}_k^s, \{b\}_k^s\}$, initial knowledge $\{1, a, b, k, c_1, c_2\}$ of Π_1 and initial knowledge $\{2, a, b\}$ of Π_2.

a "small" attack. Here we rather make use of the fact that procedures for computing sound and complete solution sets exist, which on the one hand makes our proof (and also our algorithm) more modular and easier to extend, but on the other hand requires an appropriate encoding of the problem into constraint systems.

We note that if we used positional strategies as in [10], SolveStrategy would have to be extended to guess the symbolic states of symbolic branching structures that coincide after the substitution is applied. To avoid this, we employ the strategies with history as explained above.

As mentioned in the introduction, unlike constraint-based algorithms for reachability properties, we cannot dispense with step 3. of SolveStrategy and it might not suffice to compute just one simple constraint system. This is illustrated in Section 5.3.

5.1 Guessing the Symbolic Branching Structure

To describe the first step of SolveStrategy in more detail, we first define symbolic branching structures, which consist of symbolic paths and symbolic strategy trees. To define symbolic paths and strategy trees, we need to introduce symbolic states, transitions, and trees. These notions will be illustrated by the example in Figure 1

A *symbolic state* q^s is a tuple $(\{\Pi_i\}_i, \mathcal{I}, \mathcal{S})$ consisting of a finite family $\{\Pi_i\}_i$ of rule trees, a finite set $\mathcal{I}$ of terms (which may contain variables), and a finite multi-set $\mathcal{S}$ of secure channel terms (which may contain variables). Note that a symbolic state does not contain a substitution.

A *symbolic transition* is a transition between symbolic states and is of the form

$$((\Pi_1, \ldots, \Pi_n), \mathcal{I}, \mathcal{S}) \xrightarrow{\tau} ((\Pi'_1, \ldots, \Pi'_n), \mathcal{I}', \mathcal{S}') \tag{2}$$

with τ an appropriate label. Just as for (concrete) transitions, symbolic transitions have to satisfy certain conditions and we distinguish between symbolic intruder, secure channel, and ε-transitions. The conditions are formulated with respect to a set $\mathcal{X}$ of

variables and an identifier κ, where $\mathcal{X}$ will later be defined to be the set of "used" variables at vertex κ (and its ancestors) in a symbolic strategy tree. When performing a symbolic transition, new variables, i.e., those not in $\mathcal{X}$, are replaced by new copies, i.e., copies that do not appear anywhere else in a symbolic strategy tree. These new copies are obtained by indexing the variables with κ. The new copies are needed since a symbolic strategy tree comprises different runs of a protocol, and hence, the same principal rule may be performed in different runs with different substitutions of the variables. By introducing new copies, we guarantee that the different applications of the same principal rule do not interfere (as far as the new variables are concerned). In what follows, let $\Pi_i = (V_i, E_i, r_i, \ell_i)$ and $\Pi'_i = (V'_i, E'_i, r'_i, \ell'_i)$ denote rule trees.

1. *Symbolic intruder transitions:* Transition (2) with label $\tau = i, v, I$ exists if $v \in V_i$ with $(r_i, v) \in E_i$ and $\ell_i(r_i, v) = R \Rightarrow S$ for some R and S such that
   (a) for every $j \neq i$ we have $\Pi'_j = \Pi_j$ and Π'_i is obtained from $\Pi_i{\downarrow}v$ by replacing every occurrence of a variable $x \in \mathcal{V}(R) \setminus \mathcal{X}$ by the copy x_κ,
   (b) $\mathcal{I}' = \mathcal{I} \cup \{t'\}$ where $t = S$ if $S \neq \mathsf{sc}(\cdot,\cdot,\cdot)$, and $S = \mathsf{sc}(\cdot,\cdot,t)$ otherwise, where t' is obtained from t by replacing variables $x \in \mathcal{V}(R) \setminus \mathcal{X}$ by x_κ.
   (c) $\mathcal{S}' = \mathcal{S}$ if $S \neq \mathsf{sc}(\cdot,\cdot,\cdot)$, and $\mathcal{S}' = \mathcal{S} \cup \{S'\}$ otherwise, where S' is obtained from S by replacing variables $x \in \mathcal{V}(R) \setminus \mathcal{X}$ by x_κ.
2. *Symbolic secure channel transitions:* Transition (2) with label $\tau = i, v, R', \mathsf{sc}$ exists if $v \in V_i$ with $(r_i, v) \in E_i$ and $\ell_i(r_i, v) = R \Rightarrow S$ such that $R' \in \mathcal{S}$, (a) and (b) from 1., and $\mathcal{S}' = \mathcal{S} \setminus \{R'\}$ if $S \neq \mathsf{sc}(\cdot,\cdot,\cdot)$, and $\mathcal{S}' = (\mathcal{S} \setminus \{R'\}) \cup \{S'\}$ otherwise.
3. *Symbolic ε-transitions:* Transition (2) with label $\tau = i, v$ exists if $v \in V_i$ with $(r_i, v) \in E_i$ and $\ell_i(r_i, v) = \varepsilon \Rightarrow S$ with (a)–(c) from above.

Analogously to (concrete) transitions, we call i the *principal associated with the transition*, $R \Rightarrow S$ (for the intruder and secure channel transitions) and $\varepsilon \Rightarrow S$ (for the ε-transitions) the *principal rule associated with the transition*, and v the *principal vertex associated with the transitions*.

After applying one of the above symbolic transitions, the set of used variables is updated to be the union of $\mathcal{X}$ and the variables occurring on the left-hand side of the principal rule associated with the transition, i.e., the set is $\mathcal{X} \cup (\mathcal{V}(R) \setminus \mathcal{X})_\kappa$ for symbolic intruder and secure channel transitions, and $\mathcal{X}$ for symbolic ε-transitions where $(\mathcal{V}(R) \setminus \mathcal{X})_\kappa$ is the set of variables in $\mathcal{V}(R) \setminus \mathcal{X}$ indexed by the identifier κ of the transition.

We can now define symbolic paths and symbolic strategy trees as special cases of symbolic trees. Let $P = ((\Pi_1, \ldots, \Pi_n), \mathcal{I}_0)$.

For a symbolic state q^s associated with a set $\mathcal{X}$ of variables (used so far), a *symbolic q^s-tree* is an unordered vertex- and edge-labeled tree of the form $\mathcal{T}^s_{q^s} = (V, E, r, \ell_V, \ell_E)$ with vertex and edge set V and E, respectively, root $r \in V$, vertex labeling function ℓ_V

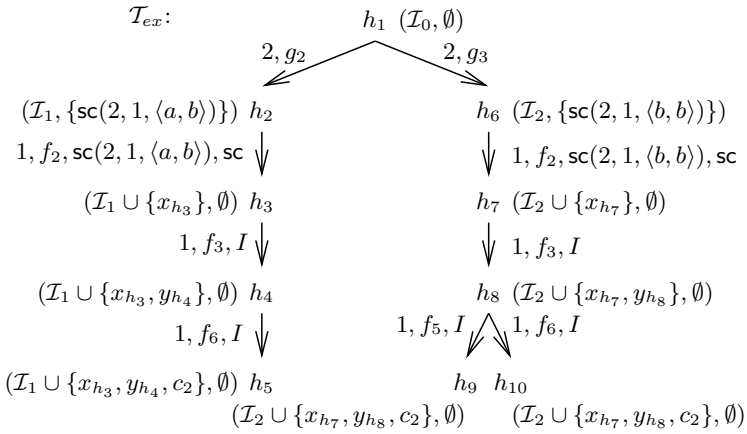

Figure 2: Symbolic strategy tree $\mathcal{T}_{ex}$ for the protocol P_{ex} with the first component of the symbolic states omitted, $\mathcal{I}_1 = \mathcal{I}_0 \cup \{\langle a, b\rangle\}$, and $\mathcal{I}_2 = \mathcal{I}_0 \cup \{\langle b, b\rangle\}$. The strategy property we consider is $((C_{ex}, C'_{ex})) = ((\{c_2\}, \{c_1\}))$.

which maps vertices to symbolic states, and edge labeling function ℓ_E which maps edges to labels of symbolic transitions such that

1. $\ell_V(r) = q^s$,
2. $\ell_V(v) \overset{\ell_E(v,v')}{\longrightarrow} \ell_V(v')$ is a symbolic transition w.r.t. the set $\mathcal{X}_v$ of used variables in v and identifier v' for all $(v, v') \in E$ where for the root r we set $\mathcal{X}_r = \mathcal{X}$ and for all other vertices in $\mathcal{T}^s_{q^s}$ the set $\mathcal{X}_v$ is determined by the symbolic transition applied in the edge leading to v.

Let $\mathcal{X}_{\mathcal{T}^s_{q^s}} = \bigcup_{v \in V} \mathcal{X}_v$ be the *set of used variables in* $\mathcal{T}^s_{q^s}$. The symbolic transition $\ell_V(v) \overset{\ell_E(v,v')}{\longrightarrow} \ell_V(v')$ is called the *symbolic transition corresponding to (or associated with)* (v, v').

Figure 2 depicts a symbolic q^s_0-tree $\mathcal{T}_{ex}$ for P_{ex} (Figure 1) where $q^s_0 = (\{\Pi_1, \Pi_2\}, \mathcal{I}_0, \emptyset)$ is the symbolic initial state of P_{ex} and the set of used variables associated with q^s_0 is empty. The first component of the symbolic states in this tree have been omitted for the sake of presentation. Note that following our naming convention, copies of the variables x and y have been introduced, namely $x_{h_3}, x_{h_7}, y_{h_4}, y_{h_8}$.

A *symbolic path* π^s of P is a symbolic q^s-tree where $q^s = ((\Pi_1, \ldots, \Pi_n), \mathcal{I}_0, \emptyset)$ is the *symbolic initial state of* P, the set of used variables associated with q^s is empty, and every vertex has at most one successor.

For a symbolic state q^s associated with a set $\mathcal{X}$ of used variables a *symbolic* q^s*-strategy tree* is a finite symbolic q^s-tree $\mathcal{T}^s_{q^s} = (V, E, r, \ell_V, \ell_E)$ which in addition to the conditions on symbolic trees also satisfies the following conditions:

3. For every v and every symbolic ε-transition from $\ell_V(v)$ with label $\tau = i, n$, there exists v' with $(v, v') \in E$ such that $\ell_E(v, v') = i, n$.

4. For all $(v, v'), (v, v'') \in E$ where $v' \neq v''$, $\ell_E(v, v') = j', n'$, and $\ell_E(v, v'') = j'', n''$ it follows that $j' \neq j''$ or $n' \neq n''$.

5. For all $(v, v'), (v, v'') \in E$ where $v' \neq v''$, $\ell_E(v, v') = j', n', R', \mathsf{sc}$, and $\ell_E(v, v') = j'', n'', R'', \mathsf{sc}$ it follows that $j' \neq j''$, $n' \neq n''$, or $R' \neq R''$.

6. For all $(v, v'), (v, v'') \in E$ where $v' \neq v''$, $\ell_E(v, v') = j', n', I$ and $\ell_E(v, v'') = j'', n'', I$ it follows that $j' = j''$ and $n' \neq n''$.

Condition 3. says that all symbolic ε-transitions that can be taken are present in the symbolic strategy tree. Condition 4. prevents superfluous symbolic ε-transitions, i.e., different edges with the same label of a symbolic ε-transition. Condition 5. is analogous for symbolic secure channel transitions. Condition 6. implies that for all direct successors $v_1, \ldots, v_h$ of v in $\mathcal{T}^s_{q^s}$ with $\ell_E(v, v_j) = i_j, f_j, I$ we have that $(i :=) i_1 = \cdots = i_h$. That is, when following his strategy, the intruder may only send a message to one principal Π_i. However, given this message, Π_i might be able to apply different principal rules. In case $h > 1$, we interpret this as guessing that all principal rules $R_1 \Rightarrow S_1, \ldots, R_h \Rightarrow S_h$ in the labels of $(r_i, f_1), \ldots, (r_i, f_h)$ of Π_i where r_i is the root of Π_i can be applied given the message delivered by the intruder and no other rules. In the constraint system to be constructed (2. step of SolveStrategy) we will therefore guarantee that the $R_1, \ldots, R_h$ are unified and in the third step of SolveStrategy we check that no other rules can be applied.

The symbolic tree $\mathcal{T}_{ex}$ depicted in Figure 2 is in fact a symbolic q^s_0-strategy tree where $q^s_0 = (\{\Pi_1, \Pi_2\}, \mathcal{I}_0, \emptyset)$ is the symbolic initial state of P_{ex}.

Given a protocol P, we call $\mathbf{B} = \pi^s, \mathcal{T}^s_1, \ldots, \mathcal{T}^s_l$ a *symbolic branching structure of* P if

1. π^s is a symbolic path of P where the label of the leave is q^s for some symbolic state q^s.

2. $\mathcal{T}^s_i$ is a symbolic q^s-strategy of P tree where the root of $\mathcal{T}^s_i$ coincides with the leave of π^s and q^s is associated with the set $\mathcal{X}_{\pi^s}$ of used variables of π^s for every i, and

3. the sets of vertices of the $\mathcal{T}^s_i$ are disjoint, except that they share a common root.

Note that a symbolic branching structure is a symbolic q^s_0-tree where the root is the root of π^s and q^0_s is the label of this root, and hence, q^s_0 is the symbolic initial state of P. By our naming convention, we have that $\mathcal{X}_{\mathcal{T}^s_i} \cap \mathcal{X}_{\mathcal{T}^s_j} = \mathcal{X}_{\pi^s}$ for every $i \neq j$.

Obviously, there is a non-deterministic exponential time algorithm which given P can guess all possible symbolic branching structures.

For the strategy property $((C_{ex}, C'_{ex})) = ((\{c_2\}, \{c_1\}))$, we can consider $\mathcal{T}_{ex}$ (Figure 2) also as a symbolic branching structure $\mathbf{B}_{ex}$ of P_{ex} where the path π^s is empty and $l = 1$.

5.2 Constructing and Solving the Induced Constraint System

We now show how the constraint system $\mathbf{C} = \mathbf{C_B}$ is derived from the symbolic branching structure $\mathbf{B} = \pi^s, \mathcal{T}_1^s, \ldots, \mathcal{T}_l^s$ from the first step of SolveStrategy and the given strategy property $((C_1, C'_1), \ldots, (C_l, C'_l))$. This constraint system can be shown to be valid, and hence, by Fact 1, a constraint solver can be used to solve it.

In the next section, we first illustrate how $\mathbf{C}$ is derived from $\mathbf{B}$ and the strategy property by the example in Figure 1, with a formal definition provided in Section 5.2.2.

5.2.1 Illustrating the Construction by an Example

Before turning to the example, we informally explain how to encode in a constraint system communication involving the secure channel. The basic idea is that we write messages intended for the secure channel into the intruder's knowledge and let the intruder deliver these messages. The problem is that while every message in the secure channel can only be read once, the intruder could try to deliver the same message several times. To prevent this, every such message when written into the intruder's knowledge is encrypted with a *new* key not known to the intruder and this key is also (and only) used in the principal rule which according to the symbolic branching structure is supposed to read the message. This guarantees that the intruder cannot abusively deliver the same message several times to unintended recipients or make use of these encrypted messages in other contexts. Here we use the feasibility condition on principals introduced in Section 2, namely that verification keys can be derived by a principal. As explained before, without this condition, a principal rule of the form $\{y\}_x^s \Rightarrow x$ would be allowed even if the principal does not know (i.e., cannot derive) x. Such a rule would equip a principal with the unrealistic ability to derive any secret key from a ciphertext. Hence, the intruder, using this principal as an oracle, could achieve this as well and could potentially obtain the new keys used to encrypt messages intended for the secure channel.

We now turn to our example and explain how the (valid) constraint system, which we call $\mathbf{C}_{ex}$ derived from $\mathbf{B}_{ex}$ and $((C_{ex}, C'_{ex}))$ looks like and how it is derived from $\mathbf{B}_{ex}$, where $\mathbf{B}_{ex}$, as explained above, is simply the symbolic strategy tree $\mathcal{T}_{ex}$ (Figure 2): $\mathbf{C}_{ex}$ is the following sequence of constraints with an empty substitution where $k_1, k_2, k_3 \in \mathcal{A}$

are new atoms and we write $t_1, \ldots, t_n$ instead of $\{t_1, \ldots, t_n\}$.

$$
\begin{array}{rrcl}
1. & \{\langle x_{h_3}, b\rangle\}^s_{k_1} & : & \mathcal{I}_1, \{\langle a, b\rangle\}^s_{k_1} \\
2. & \{\langle x_{h_7}, b\rangle\}^s_{k_2} & : & \mathcal{I}_2, \{\langle b, b\rangle\}^s_{k_2} \\
3. & \{y_{h_4}\}^s_k & : & \mathcal{I}_1, \{\langle a, b\rangle\}^s_{k_1}, x_{h_3} \\
4. & \{y_{h_8}\}^s_k & : & \mathcal{I}_2, \{\langle b, b\rangle\}^s_{k_2}, x_{h_7} \\
5. & y_{h_4} & : & \mathcal{I}_1, \{\langle a, b\rangle\}^s_{k_1}, x_{h_3}, y_{h_4} \\
6. & x_{h_7} & : & \mathcal{I}_2, \{\langle b, b\rangle\}^s_{k_2}, x_{h_7}, y_{h_8} \\
7. & y_{h_8} & : & \mathcal{I}_2, \{\langle b, b\rangle\}^s_{k_2}, x_{h_7}, y_{h_8} \\
8. & \{x_{h_7}\}^s_{k_3} & : & \mathcal{I}_2, \{\langle b, b\rangle\}^s_{k_2}, x_{h_7}, y_{h_8}, \{y_{h_8}\}^s_{k_3} \\
9. & c_2 & : & \mathcal{I}_1, \{\langle a, b\rangle\}^s_{k_1}, x_{h_3}, y_{h_4}, c_2 \\
10. & c_2 & : & \mathcal{I}_2, \{\langle b, b\rangle\}^s_{k_2}, x_{h_7}, y_{h_8}, c_2 \\
11. & c_2 & : & \mathcal{I}_2, \{\langle b, b\rangle\}^s_{k_2}, x_{h_7}, y_{h_8}, c_2
\end{array}
$$

This constraint system is obtained from $\mathbf{B}_{ex}$ as follows: We traverse the vertices of $\mathbf{B}_{ex}$ in a top-down breadth first manner. Every edge induces a constraint except those edges which correspond to symbolic ε-transitions. This is how the constraints 1.–7. come about where 1., 3., and 5. are derived from the left branch of $\mathbf{B}_{ex}$ and 2., 4., 6., and 7. from the right branch. Note that in 1. and 2. we encode the communication with the secure channel by encrypting the terms with new keys k_1 and k_2. The terms $\{\langle a, b\rangle\}^s_{k_1}$ and $\{\langle b, b\rangle\}^s_{k_2}$ are not removed anymore from the right-hand side of the constraints, i.e., from the intruder knowledge in order for $\mathbf{C}_{ex}$ to be valid, in particular, satisfy the monotonicity property of constraint systems. As explained above, since we use *new* keys and due to feasibility condition on principals, this does not cause problems. The constraints 9.–11. are used to ensure that c_2 can be derived at every leaf of $\mathcal{T}_{ex}$, a requirement that comes from our example security property $((C_{ex}, C'_{ex}))$ where $C_{ex} = \{c_2\}$. In vertex h_8 of $\mathcal{T}_{ex}$ two symbolic intruder transitions leave the vertex, which, as explained above, means that the associated principal rules should both be able to read the message delivered by the intruder. Constraint 8. ensures that the left-hand sides of these principal rules are unified, where again we use the "trick" of encrypting messages with new keys. For correctness and completeness of our algorithm, constraint 8. is not necessary, but it helps to reduce the search space (see also Section 5.2.2).

Let $\mathbf{C}_1$ and $\mathbf{C}_2$ be constraint systems with empty sequences of constraints and the substitution $\nu_1 = \{x_{h_3} \mapsto a, x_{h_7} \mapsto b, y_{h_4} \mapsto a, y_{h_8} \mapsto b\}$ and $\nu_2 = \{x_{h_3} \mapsto a, x_{h_7} \mapsto b, y_{h_4} \mapsto b, y_{h_8} \mapsto b\}$, respectively. It is easy to see that $\{\mathbf{C}_1, \mathbf{C}_2\}$ is a sound and complete solution set for $\mathbf{C}_{ex}$. Since $\mathbf{C}_{ex}$ is valid, such a set can be computed by the constraint solver (Fact 1).

5.2.2 Formal Definition

We now provide a more formal construction of the constraint system $\mathbf{C} = \mathbf{C_B}$ from the symbolic branching structure $\mathbf{B} = \pi^s, \mathcal{T}_1^s, \ldots, \mathcal{T}_l^s$ and the given strategy property

$((C_1, C_1'), \ldots, (C_l, C_l'))$.

For the sake of presentation, the translation from $\mathbf{B}$ into the constraint system $\mathbf{C}$ is performed in several steps. In the first two steps, $\mathbf{B}$ is transformed, and in the last step the branching structure obtained this way is turned into the constraint system $\mathbf{C}$. In the first step, $\mathbf{B}$ is transformed to get rid of secure channel terms generated by the intruder. In the second, step we encode the communication with the secure channel as explained in Section 5.2.1.

Encoding the derivation of secure channel terms. An intruder does not have secure channel terms in the set of messages given to him. Hence, to construct a term $\mathsf{sc}(n, n', m)$ he has to derive n and m (not necessarily n'). Since the addresses such as n may only occur in the first or second component of a secure channel term and the intruder cannot read these components, the intruder cannot derive a new address. Hence, the only addresses the intruder knows are those in his initial intruder knowledge. Thus, to check whether the intruder can construct $\mathsf{sc}(n, n', m)$, we only need to check whether n belongs to his initial intruder knowledge and whether he can derive m. Therefore, we replace in every symbolic intruder transition $q \xrightarrow{\tau} q'$ associated with an edge in $\mathbf{B}$ the left-hand side of the associated principal rule by R if it is of the form $\mathsf{sc}(n, n', R)$ and n belongs to the intruder knowledge of q. If there is a transition and an associated principal rule with a left-hand side of the form $\mathsf{sc}(n, n', R)$ but where n does not belong to the intruder knowledge of q, then SolveStrategy stops. We refer to the symbolic branching structure obtained by the transformation just described as $\hat{\mathbf{B}}$.

Encoding secure channel communication. As already explained in Section 5.2.1, we want make the secure channel component in symbolic states superfluous and instead, as in the standard Dolev-Yao model, let the intruder handle all communication. We therefore write messages intended for the secure channel into the intruder's knowledge and let the intruder deliver these messages. The problem is that while every message in the secure channel can only be read once, the intruder could try to deliver the same message several times. Therefore, when written into the intruder's knowledge, the messages are encrypted with a new key not known to the intruder and this key is also (and only) used in the principal rules which according to the symbolic branching structure are supposed to read the message. This guarantees that the intruder cannot abusively deliver the same message several times to unintended recipiences or make use of these encrypted messages in other contexts. Following this idea, we now present the transformation of $\hat{\mathbf{B}}$: In what follows, with "new key" we mean a constant in $\mathcal{A}$ that does not occur anywhere else and which, in particular, the intruder does not (and will not get to) know.

For every vertex v of $\hat{\mathbf{B}}$ and every successor v' of v in $\hat{\mathbf{B}}$ do the following (we travers the vertices of $\hat{\mathbf{B}}$ in a top-down breadth first manner starting with the root of $\hat{\mathbf{B}}$): If in

the transition associated with the edge (v, v') of $\hat{\mathbf{B}}$ a message of the form $\mathsf{sc}(n, n', R)$ is written into the secure channel, then generate a new key $k_{v'}$ and write $\{R\}^s_{k_{v'}}$ into the intruder knowledge at v' and all successors of v' in $\hat{\mathbf{B}}$. Also, starting from v', find all vertices v'' in $\hat{\mathbf{B}}$ of minimal distance from v' (where the distance is measured in terms of the length of paths) such that v'' has an outgoing secure channel transition where $\mathsf{sc}(n, n', R)$ is read, i.e., the label of the transition is of the form $\cdot, \cdot, \mathsf{sc}(n, n', R), \mathsf{sc}$. For all such outgoing transitions of v'' (there may be more than one), replace $\mathsf{sc}(n, n', R)$ in the label by $\{R\}^s_{k_{v'}}$ and replace the left-hand side of the principal rule associated with the transition by $\{R'\}^s_{k_{v'}} \to \cdot$ where the principal rule is of the form $\mathsf{sc}(n, n', R') \to \cdot$.

We refer to the symbolic branching structure resulting from the transformation just described by $\overline{\mathbf{B}}$.

We call $\{R\}^s_{k_{v'}}$ the *(secure channel) encoding term associated with* $k_{v'}$. If a ground substitution is applied to this term, we call it the *(secure channel) encoding message associated with* $k_{v'}$. We call the keys introduced in the step described above *secure channel keys* (sc-keys) and denote the set of secure channel keys by K_{sc}. We refer to the key generated for the edge (v, v') by $k_{v'}$ (if any).

Deriving the constraint system. From $\overline{\mathbf{B}}$ we now derive the constraint system $\mathbf{C}$. The substitution τ of $\mathbf{C}$ is defined to be empty. The sequence of constraints is constructed by iteratively adding one constraint at the end of the sequence as follows:

1. We traverse the vertices of $\overline{\mathbf{B}}$ in a top-down breadth first manner starting with the root of $\overline{\mathbf{B}}$ (and hence, the root of π^s) and for every vertex v of $\overline{\mathbf{B}}$ we do the following: For every successor v' of v where the corresponding transition of the edge (v, v') is a symbolic secure channel or intruder transition add the constraint $R : \mathcal{I}$ given that the intruder knowledge in v is $\mathcal{I}$ and that the left-hand side of the principal rule associated with the transition is R. (We call the constraints of this kind the *intruder constraints of* $\boldsymbol{C}$.) In addition, if v has more than one outgoing symbolic intruder transition, then let $v_1, \ldots, v_n$ denote all the corresponding successors of v, let $R_1, \ldots, R_n$ denote the left-hand sides of the principal rules associated with the respective transitions, and let k denote a new key. Add the constraint

$$\{\langle R_1, \langle R_2, \langle \ldots R_{n-2}, R_{n-1}\rangle \cdots\rangle\rangle\}^s_k : \mathcal{I}, \{\langle R_2, \langle R_3, \langle \ldots R_{n-1}, R_n\rangle \cdots\rangle\rangle\}^s_k \qquad (3)$$

   where $\mathcal{I}$ is defined as above. (This last constaint forces the unification of the R_i. We therefore call constraints of this kind the *unification constraints* of $\mathbf{C}$.)

2. For every i and leave v of $\mathcal{T}^s_i$ add the constraint $a : \mathcal{I}$ for every $a \in C_i$ where $\mathcal{I}$ is the intruder's knowledge in the state associated with v. (We call these constraints the *strategy property constraints of* $\boldsymbol{C}$.)

We note that for correctness and completeness of our algorithm the constraint (3) is not necessary. But in an optimized version of our algorithm it helps to reduce the search space.

This completes the construction of $\mathbf{C}$. To be able to apply Fact 1, we need the following lemma, which can easily be verified:

Lemma 7 *$\boldsymbol{C}$ is a valid constraint system.*

To prove this lemma, we use that secure channel encoding messages are not removed anymore from the intruder's knowledge. Also, in the constraint (3) the terms R_i (and hence, there variables) occur in the constraints $R_i : \mathcal{I}$ previously added to the constraint sequence.

As a consequence of this lemma and Fact 1, we can employ the constraint solver to solve $\mathbf{C}$.

5.3 Checking the Induced Substitutions

Let $\mathbf{B} = \pi^s, \mathcal{T}_1^s, \ldots, \mathcal{T}_l^s$ be the symbolic branching structure obtained in the first step of SolveStrategy and let $\mathbf{C}'$ be the simple constraint system returned by the constraint solver when applied to $\mathbf{C} = \mathbf{C_B}$ in the second step of SolveStrategy. Let ν be the complete solution associated with $\mathbf{C}'$ (see Section 4). We emphasize that for our algorithm to work, it is important that ν replaces the variables in $\mathbf{C}'$ by *new* intruder atoms from $\mathcal{A}_I$ not occurring in $\mathbf{B}$.

Basically, we want to check that when applying ν to $\mathbf{B}$, which yields $\nu(\mathbf{B}) = \nu(\pi^s), \nu(\mathcal{T}_1^s), \ldots, \nu(\mathcal{T}_l^s)$, we obtain a solution of the problem instance $(P, (C_1, C_1'), \ldots, (C_l, C_l'))$. Hence, we need to check whether i) $\nu(\pi^s)$ corresponds to a path in $\mathcal{G}_P$ from the initial state of $\mathcal{G}_P$ to a state $q \in \mathcal{G}_P$ and ii) $\nu(\mathcal{T}_i^s)$ corresponds to a q-strategy tree for (C_i, C_i') for every i. However, since ν is a complete solution of $\mathbf{C}$, some of these conditions are satisfied by construction. In particular, $\nu(\pi^s)$ is guaranteed to be a path in $\mathcal{G}_P$ starting from the initial state. Also, the conditions 1.–3. of strategy trees (Definition 3) do not need to be checked and we know that $\nu(\mathcal{T}_i^s)$ satisfies $(C_i, \emptyset)$. Hence, SolveStrategy only needs to make sure that 4. and 5. of Definition 3 are satisfied for every $\nu(\mathcal{T}_i^s)$ and that $\nu(\mathcal{T}_i^s)$ fulfills $(\emptyset, C_i')$.

More concretely, SolveStrategy checks the following conditions for every i where 1. corresponds to checking wether $\nu(\mathcal{T}_i^s)$ satisfies $(C_i, \emptyset)$, 2.,(a) corresponds to checking 4. of Definition 3, and 2.,(b) corresponds to checking 5. of Definition 3:

1. For every leave v of $\mathcal{T}_i^s$ it holds that $C_i' \cap d(\mathcal{I}\nu) = \emptyset$ where $\mathcal{I}$ is the intruder knowledge in the state associated with v.

2. For every vertex v in $\mathcal{T}_i^s$ the following conditions are satisfied: Let $(\{\Pi_h'\}_h, \mathcal{I}, \mathcal{S})$ be the symbolic state associated with v in $\mathcal{T}_i^s$ with $\Pi_h' = (V_h', E_h', r_h', \ell_h')$.

(a) For every $m \in \mathcal{S}\nu$, h, n'_h such that
   i. $(r'_h, n'_h) \in E'_h$,
   ii. $\ell'_h(r'_h, n'_h) = R \rightarrow \cdot$ for some R, and
   iii. $R\nu$ matches with m

   there exists a successor v' of v in $\mathcal{T}_i^s$ such that the label of the edge of (v, v') in $\mathcal{T}_i^s$ is h, n'_h, R', sc for some R' with $R'\nu = m$.

(b) If there exists a successor of v' in $\mathcal{T}_i^s$ such that
   i. the transition corresponding to the edge (v, v') is labeled with h, n'_h, I,
   ii. $\ell'_h(r'_h, n'_h) = R \rightarrow \cdot$ for some R, and
   iii. there exists $n''_h \neq n'_h$ with $(r'_h, n''_h) \in E'_h$, $\ell'_h(r'_h, n''_h) = R' \rightarrow \cdot$ for some R', and $R'\nu$ matches with $\hat{R}\nu$ where $\hat{R}$ is obtained from R by replacing all variables of the form x in R, i.e., those variables not indexed by vertices yet, by $x_{v'}$,

   then there exists v'' such that (v, v'') is an edge in $\mathcal{T}_i^s$ and the label of (v, v'') is h, n''_h, I.

It is clear that the tests in 2. can easily be performed given ν and $\mathcal{T}_i^s$. In [6] it was shown that that derivation problem, i.e., the problem of deciding $m \in d(\mathcal{I})$ given a ground term m and a finite set of ground terms $\mathcal{I}$, can be decided in polynomial time. Hence, 1. can also be checked efficiently in the size of the security property and $\mathcal{I}\nu$.

If the above checks are successful, we say that ν is *valid* for $\mathbf{B}$. Our algorithm SolveStrategy outputs yes and $\nu(\mathbf{B})$ if ν is valid for $\mathbf{B}$.

In our example, the induced substitution for $\mathbf{C}_i$ is ν_i as $\mathbf{C}_i$ does not contain any variables. It can easily be verified that with $\mathbf{C}' = \mathbf{C}_2$ and the induced substitution ν_2, the above checks are all successful. However, 2.,(b) fails for $\mathbf{C}' = \mathbf{C}_1$ and ν_1 because in h_4 the rule $a \Rightarrow c_1$ could also be applied but it is not present in $\mathbf{B}_{ex}$. In fact, $\mathbf{B}_{ex}\nu_1$ would not yield a solution of the instance $(P_{ex}, ((C_{ex}, C'_{ex})))$. This example illustrates that in SolveStrategy one cannot dispense with the last step, namely checking the substitutions, and that one has to try the different constraint systems in the sound and complete solution set for $\mathbf{C}$.

6 Proof of the Main Theorem

We now present the proof of Theorem 6. Obviously, SolveStrategy always terminates. We need to prove soundness and completeness of SolveStrategy. This is done in Section 6.1 and 6.2, respectively.

First we will introduce some additional notation that will be used in the proof.

Let P be a protocol, $\mathcal{C} = ((C_1, C'_1), \ldots, (C_l, C'_l))$ be a strategy property, and $q \in \mathcal{G}_P$. A *q-tree* is a q-strategy tree where, however, only condition 1. and 2. need to be satisfied

(see Definition 3). A *(concrete) path* π for P is a q_0-tree where q_0 is the initial state of P and every vertex in π only has at most one direct successor. A *(concrete) branching structure* for P is defined analogously to *symbolic* branching structures only that now the symbolic path is a concrete path π and the symbolic strategy trees $\mathcal{T}_i$ are concrete trees. Such a branching structure satisfies $\mathcal{C}$ if $\mathcal{T}_i$ satisfies (C_i, C_i') for every i. Obviously, we have:

Remark 8 $(P, \mathcal{C}) \in$ G-Strategy *iff there exists a concrete branching structure for* P *which satisfies* $\mathcal{C}$.

With $\mathbf{B} = \pi, \mathcal{T}_1, \ldots, \mathcal{T}_l$ we will always denote a *concrete* branching structure for protocol P where $\pi = (V^\pi, E^\pi, r^\pi, \ell_V^\pi, \ell_E^\pi)$ and $\mathcal{T}_i = (V^i, E^i, r^i, \ell_V^i, \ell_E^i)$ for every i. Recall that $\mathbf{B}$ is a q_0-tree $(V, E, r, \ell_V, \ell_E)$. In particular, $V = V^\pi \cup V^1 \cup \cdots \cup V^l$, $E = E^\pi \cup E^1 \cup \cdots \cup E^l$, $r = r^\pi$, and ℓ_V and ℓ_E coincide with the labeling functions of the components of $\mathbf{B}$ on the respective domains. For a symbolic or concrete tree $\mathcal{T}$, we write $v \in \mathcal{T}$ to say that v is a vertex in $\mathcal{T}$. Analogously, we write $(v, v') \in \mathcal{T}$ if (v, v') is an edge in $\mathcal{T}$. Note that since branching structures are trees so we may also write $v \in \mathbf{B}$ if $v \in V$ and $(v, v') \in \mathbf{B}$ if $(v, v') \in E$.

For $v \in \mathbf{B}$ let $\ell_V(v) = ((\Pi_1^v, \ldots, \Pi_n^v), \sigma^v, \mathcal{I}^v, \mathcal{S}^v)$ and $\Pi_j^v = (V_j^v, E_j^v, r_j^v, \ell_j^v)$.

As usual, for an edge (v, v') in $\mathbf{B}$, we talk about the transition (in $\mathcal{G}_P$) associated with (v, v'). If i, $R \Rightarrow S$, f is the principal, principal rule, and principal vertex associated with this transition, respectively, then we call i, $R \Rightarrow S$, and f, the principal, principal rule, and principal vertex, respectively, associated with (v, v').

Analogously, $\mathbf{B}^s = \pi^s, \mathcal{T}_1^s, \ldots, \mathcal{T}_l^s$ will always denote a *symbolic* branching structure. The naming convention is the same as above where, however, we always add an s as a superscript. For instance, we write $\pi^s = (V^{s,\pi}, E^{s,\pi}, r^{s,\pi}, \ell_V^{s,\pi}, \ell_E^{s,\pi})$ and $\Pi_j^{s,v} = (V_j^{s,v}, E_j^{s,v}, r_j^{s,v}, \ell_j^{s,v})$. The transitions, prinicpals, principal rules, and principal vertices associated with edges in $\mathbf{B}^s$ are defined as above.

Recall that for a symbolic branching structure $\mathbf{B}^s$ we denote the branching structure constructed from $\mathbf{B}^s$ in step two of SolveStrategy by $\overline{\mathbf{B}}^s$. We refer to the components of $\overline{\mathbf{B}}^s$ by overlined versions of the components of $\mathbf{B}^s$, i.e., $\overline{\ell}_E^{s,\pi}$, $\overline{V}_j^{s,v}$ and so on.

Let $q^s = ((\Pi_1, \ldots, \Pi_n), \mathcal{I}, \mathcal{S})$ be a symbolic state reachable from the initial symbolic state of P and associated with the set $\mathcal{X}$ of used variables. (Recall from Section 5.1 that the variables in $\mathcal{X}$ are indexed with vertices.) Let ν be a substitution of the variables in $\mathcal{X}$. We denote by $\nu(q^s)$ the tuple $\nu(q^s) = ((\Pi_1', \ldots, \Pi_n'), \nu|_\mathcal{X}, \mathcal{I}\nu, \mathcal{S}\nu)$ where Π_i' is obtained from Π_i by dropping the indices of variables in $\mathcal{X}$ and $\nu|_\mathcal{X}$ is ν with the domain restricted to $\mathcal{X}$.

For a symbolic q^s-tree $\mathcal{T} = (V, E, r, \ell_V, \ell_E)$ and a substitution ν of the variables used in $\mathcal{T}$ we denote by $\nu(\mathcal{T})$ the tuple $\nu(\mathcal{T}) = (V, E, r, \nu(\ell_V), \nu(\ell_E))$ with $\nu(\ell_V)(v) =$

$\nu(\ell_V(v))$ and

$$\nu(\ell_E)(v,v') = \begin{cases} j & \text{if } \ell_E(v,v') = j, f \\ j, \nu(R), I & \text{if } \ell_E(v,v') = j, f, I \text{ and } \ell_j(r_j, f) = R \Rightarrow S \\ j, \nu(R), \mathsf{sc} & \text{if } \ell_E(v,v') = j, f, R, \mathsf{sc} \end{cases}$$

for every $v, v' \in V$ and $(v, v') \in E$ where r_j denotes the root of the jth principal Π_j in the symbolic state $\ell_V(v)$ and ℓ_j denotes the labeling function of Π_j.

Now, for a branching structure $\mathbf{B}^s = \pi^s, \mathcal{T}_1^s, \ldots, \mathcal{T}_l^s$ and a substitution ν, we define $\nu(\mathbf{B}^s)$ to be $\nu(\pi^s), \nu(\mathcal{T}_1^s), \ldots, \nu(\mathcal{T}_l^s)$.

We say that ν *solves* $\boldsymbol{B}^s$ *(w.r.t. the strategy property* $\mathcal{C}$*)* if $\nu(\mathbf{B}^s)$ is a concrete branching structure (which satisfies $\mathcal{C}$).

A *root path* π' in $\mathbf{B}$ is a sequence $\pi' = v_1, \ldots, v_n$ of vertices $v_i \in \mathbf{B}$ such that $(v_i, v_{i+1}) \in \mathbf{B}$ and v_1 is the root of $\mathbf{B}$ (and hence, of π). If $\rho_i = R_i \Rightarrow S_i$ denotes the principal rule associated with the edge $(v_i, v_{i+1}) \in \mathbf{B}$, then $\rho = \rho_{\pi'} = \rho_1, \ldots, \rho_{n-1}$ is called the *pr-sequence associated with* π'. Given a set $\mathcal{I}$ of messages we call a ground substitution σ of the variables occurring in ρ, a *solution of* ρ *w.r.t.* $\mathcal{I}$ if $R_i\sigma \in d(\mathcal{I}^{s,v_i}\sigma\})$ for every i. We call σ a *solution of* π' if σ is a solution of $\rho_{\pi'}$ w.r.t. $\mathcal{I}_0$.

6.1 Soundness of SolveStrategy

In this section, we show soundness of SolveStrategy. First, we need to introduce some more notation and prove basic properties.

6.1.1 Notation and Basic Properties

For terms t, t', t'', we denote by $t_{|t' \to t''}$ the term obtained from t by replacing every occurrence of t' by t''. For a set E of terms and a substitution σ, $E_{|t' \to t''}$ and $\sigma_{|t' \to t''}$ are defined in the obvious way.

Terms can be viewed as finite vertex-labeled ordered trees where the vertices are labeled with function symbols and the number of direct successors of a vertex is exactly the arity of the function symbol the vertex is labelled with. For a term t, we write $v \in t$ to say that v is a vertex of t and $t{\downarrow}v$ to denote the subterm of t rooted at vertex v of t. We say that a term t *only occurs in the context of* $\{t'\}_t^s$ *in the term* t'' if for all $v \in t''$ with $t''{\downarrow}v = t$ we have that $t''{\downarrow}v' = \{t'\}_t^s$ where v' is the direct predecessor of v in t''. For a set of terms E we say that t *only occurs in the context of* $\{t'\}_t^s$ *in* E if t only occurs in the context of $\{t'\}_t^s$ in all terms $t'' \in E$. Let σ be a substitution. We say that t *only occurs in the context of* $\{t'\}_t^s$ *in* σ if t only occurs in the context of $\{t'\}_t^s$ in all $\sigma(x)$ for $x \in \mathsf{dom}(\sigma)$.

A term t *only occurs in key positions* in a term t' if for every vertex v of t' with $t'{\downarrow}v = t$, there exists an ancestor v' of v in t' labeled with $\{\cdot\}_\circ^s$, $\{\cdot\}_\circ^a$, or $\mathsf{sig}_\circ(\cdot)$ such that v is the direct succesor of v' that corresponds to the $\circ$ subterm or is a descendant of the

direct succesor of v' that corresponds to the $\circ$ subterm. Otherwise, we say that t *occurs in a non-key position* in t'. For example, x only occurs in a key position in the term $\{\langle a, b\rangle\}^s_{\langle c,x\rangle}$ but not in the term $\{\langle x, b\rangle\}^s_c$.

The following lemma can easily be shown by induction on the length of derivations using the same techniques as for example in [13, 10].

Lemma 9 *Let E be a set of messages, and m_1, m_2, m_3, m_4 be messages with $m_1 = \{m_2\}^s_{m_3}$. If $m_1 \in E$ and $m_3 \notin d(E)$, then we have $d(E)_{|m_1 \to m_4} \subseteq d(E_{|m_1 \to m_4})$.*

We will also need the following lemma:

Lemma 10 *Let R be a term, m_1, m_2 be messages, $k \in \mathcal{A}$ be an atom, and σ be a ground substitution such that $k \notin Sub(R)$ and k only occurs in the context of $\{m_1\}^s_k$ in σ. Then, $(R\sigma)_{|\{m_1\}^s_k \to m_2} = R(\sigma_{|\{m_1\}^s_k \to m_2})$.*

Proof. First observe that there does not exist a subterm t of R such that t is not a variable and $t\sigma = \{m_1\}^s_k$. Otherwise, since $k \notin \mathrm{Sub}(R)$, t would be of the form $\{t'\}^s_x$ for some t' and variable x, and hence, $\sigma(x) = k$ in contradiction to the assumption that k only occurs in the context of $\{m_1\}^s_k$ in σ. From this, the claim of the lemma follows easily. □

The proof of the following lemma is obvious.

Lemma 11 *Let E be a set of messages and k, m messages. If k only occurs in the context of $\{m\}^s_k$ in E, then k only occurs in the context of $\{m\}^s_k$ in $d(E)$.*

From the above lemmas, we obtain:

Lemma 12 *Let $k_1, \ldots, k_n \in \mathcal{A}$ be pairwise distinct atoms, let $\mathcal{I}_1$ and $\mathcal{I}_2$ be sets of terms such that $\{k_1, \ldots, k_n\} \cap Sub(\mathcal{I}_1) = \emptyset$ and $\mathcal{I}_2 = \{\{t_1\}^s_{k_1}, \ldots, \{t_n\}^s_{k_n}\}$ for some terms t_i such that $\{k_1, \ldots, k_n\} \cap Sub(t_i) = \emptyset$. Let σ be a ground substitution such that $\{k_1, \ldots, k_n\} \cap Sub(\sigma(x)) = \emptyset$ for every variable x. Then, the following is true for every term R with $\{k_1, \ldots, k_n\} \cap Sub(R) = \emptyset$:*

1. *$R\sigma \in d(\mathcal{I}_1\sigma \cup \mathcal{I}_2\sigma)$ implies that $R\sigma \in d(\mathcal{I}_1\sigma)$.*
2. *$\{R\}^s_{k_i}\sigma \in d(\mathcal{I}_1\sigma \cup \mathcal{I}_2\sigma)$ implies that $R\sigma = t_i\sigma$ for every i.*

Proof. To show statement 1., we first observe that $k_i \notin d(\mathcal{I}_1\sigma \cup \mathcal{I}_2\sigma)$. Now, iteratively applying Lemma 9 for every i with $m_1 = \{t_i\}^s_{k_i}\sigma$ and some intruder atom m_4 allows us to eliminate the messages in $\mathcal{I}_2\sigma$. Statement 2. easily follows with Lemma 11. □

6.1.2 Proof of Soundness

Let $P = ((\Pi_1, \ldots, \Pi_n), \mathcal{I})$ be a protocol and $\mathcal{C} = ((C_1, C_1'), \ldots, (C_l, C_l'))$ be a strategy property. We have to show that if SolveStrategy outputs yes, then $(P, \mathcal{C}) \in$ G-Strategy.

Assume that SolveStrategy outputs yes and let $\mathbf{B}^s = \pi^s, \mathcal{T}_1^s, \ldots, \mathcal{T}_l^s$ be the symbolic branching structure guessed in the first step of SolveStrategy. Let $\mathbf{C} = \mathbf{C}_{\mathbf{B}^s}$ be the corresponding constraint system constructed in the second step of the algorithm and let $\mathbf{C}'$ be the simple constraint system returned by the constraint solver such that the complete solution ν associated with $\mathbf{C}'$ passes the tests in the third step of SolveStrategy.

Basically, we want to show that ν solves $\mathbf{B}^s$ w.r.t. $\mathcal{C}$. Recall that this means that $\nu(\mathbf{B}^s)$ is a conrete branching structure which satisfies $\mathcal{C}$. The main problem is that ν might contain secure channel keys (see Section 5.2.2). With these keys, ν cannot solve $\mathbf{B}^s$ since they do not occur in $\mathbf{B}^s$. In what follows, we will therefore iteratively eliminate these keys in ν such that in every step ν satisfies certain conditions w.r.t. $\overline{\mathbf{B}}^s$ and $\mathbf{C}$ (recall the definition of $\overline{\mathbf{B}}^s$ and $\mathbf{C}$ from Section 5.2.2). This iterative process will yield a ground substitution μ for which we then show that it solves $\mathbf{B}^s$ w.r.t. $\mathcal{C}$.

Given a substitution σ, the mentioned conditions w.r.t. $\overline{\mathbf{B}}^s$ and $\mathbf{C}$ are the following:

a) For every root path π of $\overline{\mathbf{B}}^s$ the substitution σ is a solution for π.

b) σ solves the strategy property constraints of $\mathbf{C}$.

c) σ solves the unification constraints of $\mathbf{C}$.

d) σ passes the checks of step 3. of SolveStrategy.

In what follows, let σ be a substitution which satisfies the condition a) to d), and let $k \in K_{\mathsf{sc}}$ be a secure channel key and $\{t\}_k^s$ be the corresponding encoding term (see Section 5.2.2). We want to eliminate k from σ (if it occurs) and show that the resulting substitution still satisfies the above conditions. For this purpose, we first prove:

Lemma 13 *The secure channel key k only occurs in the context of $\{t\sigma\}_k^s$ in σ.*

Proof. Assume that there exists a variable z in the domain of σ such that k does not only occur in the context of $\{t\sigma\}_k^s$ in $\sigma(z)$. Note that by construction, z is of the form x_v where v is a vertex in $\mathbf{B}^s$ and x is a variable in P. Let z be minimal with this property. That is, there exists $v \in \overline{\mathbf{B}}^s$ such that $z \in \mathcal{X}_v$ (the set of used variables at vertex v) and for every proper ancestor $v' \in \overline{\mathbf{B}}^s$ of v, $z \notin \mathcal{X}_{v'}$ and every $z' \in \mathcal{X}_{v'}$ k only occurs in the context of $\{t\sigma\}_k^s$ in $\sigma(z')$. Let z and v be as above and let $\pi' = v_1, \ldots, v_n$ with $v_n = v$ be a root path in $\overline{\mathbf{B}}^s$ and $\rho = \rho_{\pi'}^{\overline{\mathbf{B}}} = \rho_1, \ldots, \rho_{n-1}$ the pr-sequence associated with π' where $\rho_i = R_i \Rightarrow S_i$. Since σ satisfies condition a), we know that $R_n\sigma \in d(\overline{\mathcal{I}}^{s,v_{n-1}}\sigma)$ (recall the definition of $\overline{\mathcal{I}}^{s,v_{n-1}}$ from above). Clearly, k only occurs in the context of $\{t\sigma\}_k^s$ in $\overline{\mathcal{I}}^{s,v_{n-1}}$. Also, $\mathcal{V}(\overline{\mathcal{I}}^{s,v_{n-1}}) \subseteq \mathcal{X}_{v_{n-1}}$. And hence, k only occurs in the context of $\{t\sigma\}_k^s$ for

every $z' \in \mathcal{V}(\overline{\mathcal{I}}^{s,v_{n-1}})$. From this we can conclude (*): k only occurs in the context of $\{t\sigma\}^s_k$ in $\overline{\mathcal{I}}^{s,v_{n-1}}\sigma$. By Lemma 11, we conclude that k only occurs in the context of $\{t\sigma\}^s_k$ in $R_n\sigma$. However, we also have that $z \in \mathcal{V}(R_n)$ and that k does not only occur in the context of $\{t\sigma\}^s_k$ in $\sigma(z)$. From this it follows that there exists a subterm of R_n of the form $\{t\}^s_z$ for some term t and that $\sigma(z) = k$. By the feasibility condition on principals we know that $z \in d(\mathcal{I} \cup \{R_1, \ldots, R_n\})$ where $\mathcal{I}$ is the union of the initial knowledge of the principals in P and we consider variables as constants. Because of the minimality, we know that z only occurs in R_n and it follows that z must occur in a non-key position in R_n. Hence, k also occurs in a non-key position in $R_n\sigma$ in contradiction to (*). □

Using Lemma 13, we obtain:

Lemma 14 *Let $\pi' = v_1, \ldots, v_n$ be a root path of $\overline{\boldsymbol{B}}^s$ and $\rho = \rho_1, \ldots, \rho_{n-1}$ with $\rho_i = R_i \Rightarrow S_i$ the pr-sequence associated with π'. Also, let $\mathcal{I}_{v_i} = \overline{\mathcal{I}}^{s,v_i}$ be defined as above. Then, k only occurs in the context of $\{t\sigma\}^s_k$ in $\mathcal{I}_{v_i}\sigma$ and $R_i\sigma$ for every i.*

Proof. By the construction of $\overline{\mathbf{B}}^s$, we know that k only occurs in the context of $\{t\sigma\}^s_k$ in $\mathcal{I}_{v_i}$ and R_i for every i. From this together with Lemma 13, the claim follows immediately. □

Let $a \in \mathcal{A}_I$ be a new intruder atom and define $\sigma' = \sigma_{|m \to a}$ where $m = \{t\sigma\}^s_k$. Note that, due to Lemma 13, k does not occur in σ' anymore.

Claim I. σ' satisfies the condition a) to d).

Proof of Claim I. a) Let $\pi' = v_1, \ldots, v_n$, ρ, ρ_i, R_i, S_i, and $\mathcal{I}_{v_i}$ be given as in Lemma 14. By assumption the substitution σ satisfies a), and hence, σ solves π'. That is, $R_i\sigma \in d(\mathcal{I}_{v_{i-1}}\sigma)$ for every i. We want to show that $R_i\sigma' \in d(\mathcal{I}_{v_{i-1}}\sigma')$ for every i. We consider different cases:

If k does not occur in $R_i\sigma$ and $\mathcal{I}_{v_{i-1}}\sigma$, then $R_i\sigma = R_i\sigma'$ and $\mathcal{I}_{v_{i-1}}\sigma = \mathcal{I}_{v_{i-1}}\sigma'$, and hence, $R_i\sigma' \in d(\mathcal{I}_{v_{i-1}}\sigma')$.

Otherwise, by construction of $\overline{\mathbf{B}}^s$, we have that $\{t\}^s_k \in \mathcal{I}_{v_{i-1}}$, and hence, $m \in \mathcal{I}_{v_{i-1}}\sigma$. By Lemma 14, we know that (**) k only occurs in the context of $\{t\sigma\}^s_k$ in $\mathcal{I}_{v_i}\sigma$ and $R_i\sigma$. Now, if $k \in \mathrm{Sub}(R_i)$, then by construction of $\overline{\mathbf{B}}^s$, we know that R_i is of the form $\{t'\}^s_k$ for some term t'. It follows that $R_i\sigma = m$. Obviously, we have that $m = \{t\}^s_k\sigma = \{t\}^s_k\sigma'$. Thus, $m \in \mathcal{I}_{v_{i-1}}\sigma'$, and therefore, $R_i\sigma' \in d(\mathcal{I}_{v_{i-1}}\sigma')$.

In case, $k \notin \mathrm{Sub}(R_i)$, we can apply Lemma 10 and obtain that $(R_i\sigma)_{|m \to a} = R_i\sigma'$. Moreover, using Lemma 11 and (**), we know that $k \notin d(\mathcal{I}_{v_{i-1}}\sigma)$. Now, we can apply Lemma 9 and obtain that $R_i\sigma' = (R_i\sigma)_{|m \to a} \in d((\mathcal{I}_{v_{i-1}}\sigma)_{|m \to a})$. For $S \in \mathcal{I}_{v_{i-1}}$ such that $k \notin S$, Lemma 10 implies $S\sigma' = (S\sigma)_{|m \to a}$. By construction of $\overline{\mathbf{B}}^s$, if $k \in \mathrm{Sub}(S)$, then $S = \{t\}^s_k$, and hence, $S\sigma = m$. Since $a \in \mathcal{A}_I \subseteq d((\mathcal{I}_{v_{i-1}}\sigma)_{|m \to a})$, we obtain that $d((\mathcal{I}_{v_{i-1}}\sigma)_{|m \to a}) \subseteq d(\mathcal{I}_{v_{i-1}}\sigma')$. Consequently, $R_i\sigma' \in d(\mathcal{I}_{v_{i-1}}\sigma')$.

b) We need to show that for every $c \in C_i$ and root path $\pi' = v_1, \ldots, v_n$ where v_n is a leaf in $\mathcal{T}_i^s$, we have that $c \in d(\overline{\mathcal{I}}^{s,v_n}\sigma')$. We know that σ solves $\mathbf{C}$, and hence, $c \in d(\overline{\mathcal{I}}^{s,v_n}\sigma)$. From this, using the same arguments as above, we obtain that $c \in d(\overline{\mathcal{I}}^{s,v_n}\sigma')$.

c) Let

$$\{\langle R_1, \langle R_2, \langle \ldots R_{n-2}, R_{n-1}\rangle \cdots\rangle\rangle\}^s_{k'} : \mathcal{I}, \{\langle R_2, \langle R_3, \langle \ldots R_{n-1}, R_n\rangle \cdots\rangle\rangle\}^s_{k'}$$

be a unification constraint. We know that σ solves this constraint. Hence,

$$\{\langle R_1, \langle R_2, \langle \ldots R_{n-2}, R_{n-1}\rangle \cdots\rangle\rangle\}^s_{k'}\sigma \in d(\mathcal{I}\sigma \cup \{\{\langle R_2, \langle R_3, \langle \ldots R_{n-1}, R_n\rangle \cdots\rangle\rangle\}^s_{k'}\sigma\}).$$

Since by construction of $\mathbf{C}$, k' is only used in this contraint, it is easy to see that $k' \notin d(\mathcal{I}\sigma)$. Therefore,

$$\{\langle R_1, \langle R_2, \langle \ldots R_{n-2}, R_{n-1}\rangle \cdots\rangle\rangle\}^s_{k'}\sigma = \{\langle R_2, \langle R_3, \langle \ldots R_{n-1}, R_n\rangle \cdots\rangle\rangle\}^s_{k'}\sigma.$$

Moreover, by construction of $\mathbf{C}$ we have that $k \notin \mathrm{Sub}(R_i)$ for every i, and hence, using Lemma 10 it follows that

$$\{\langle R_1, \langle R_2, \langle \ldots R_{n-2}, R_{n-1}\rangle \cdots\rangle\rangle\}^s_{k'}\sigma' = \{\langle R_2, \langle R_3, \langle \ldots R_{n-1}, R_n\rangle \cdots\rangle\rangle\}^s_{k'}\sigma'.$$

Thus, σ' solves the unification constraint.

d) We know that σ passes the tests 1., 2.,(a), and 2.,(b) in the third step of SolveStrategy. We know that (*) $C_i' \cap d(\overline{\mathcal{I}}^{s,v}\sigma) = \emptyset$ for every leaf v of $\mathcal{T}_i^s$ and every i. If there exists $c \in C_i' \cap d(\overline{\mathcal{I}}^{s,v}\sigma')$, then it is easy to see that, since a is new intruder atom, from $(\overline{\mathcal{I}}^{v,s}\sigma')_{|a\to\{t\}^s_k\sigma}$ we can still derive c. Together with the fact that $(\overline{\mathcal{I}}^{s,v}\sigma')_{|a\to\{t\}^s_k\sigma} = \overline{\mathcal{I}}^{s,v}\sigma$, this is a contradiction to (*).

To see that for σ', the tests 2.,(a) and 2.(b) still pass, it suffices to observe that if the matching conditions in 2.,(a),iii and 2.,(b),iii are satisfied for σ', then also for σ. Here we again use that a is a new constant and that we can replace a again by $\{t\}^s_k\sigma$. This concludes the proof of Claim I.

By assumption, ν was returned by SolveStrategy, and hence, solves $\mathbf{C}$ and passes the test in the third step of SolveStrategy, we immediately obtain that ν satisfies the conditions a) to d). Claim I allows us to iteratively eliminate all secure channel keys in ν. We denote the resulting substitution by μ.

Claim II. $\mu(\mathbf{B}^s)$ is a concrete branching structure for P which satisfies $\mathcal{C}$.

By Remark 8, Claim II implies soundness of SolveStrategy.

Proof of Claim II. We first show that every edge in $\mu(\mathbf{B}^s)$ corresponds to a transition in $\mathcal{G}_P$ by induction on the length of root paths $\pi' = v_1, \ldots, v_h$ in $\mathbf{B}^s$. That is, we show by induction on h that

(i) $\mu(\ell_V^s)(v_1)$ is the initial state of P,

(ii) $\mu(\ell_V^s)(v_i) \in \mathcal{G}_P$ for $i \in \{1, \ldots, h\}$, and

(iii) $\mu(\ell_V^s)(v_i) \overset{\mu(\ell_E^s)(v_i, v_{i+1})}{\longrightarrow} \mu(\ell_V^s)(v_{i+1}) \in \mathcal{G}_P$ for $i \in \{1, \ldots, h-1\}$.

For $h = 1$, we only need to show that $\mu(\ell_V^s)(v_1)$ is the initial state of P, which is obvious. For the induction step assume that we are given a root path $\pi' = v_1, \ldots, v_{h+1}$ in $\mathbf{B}^s$ (and thus, in $\overline{\mathbf{B}}^s$). Let $\rho^s = \rho_1^s, \ldots, \rho_h^s$ with $\rho_i^s = R_i^s \Rightarrow S_i^s$ be the pr-sequence associated with π' in $\mathbf{B}^s$ and let $\overline{\rho}^s = \overline{\rho}_1^s, \ldots, \overline{\rho}_h^s$ with $\overline{\rho}_i^s = \overline{R}_i^s \Rightarrow \overline{S}_i^s$ be the pr-sequence associated with π' in $\overline{\mathbf{B}}^s$. Moreover, let $\mu(\overline{\mathcal{I}}^{s,v_i})$ be the intruder's knowledge at v_i in $\mu(\mathbf{B}^s)$. Note that $\mu(\mathcal{I}^{s,v_i}) \subseteq \mathcal{I}_0 \cup \{S_1^s\mu, \ldots, S_{i-1}^s\mu\}$ where $\mathcal{I}_0$ is the initial intruder knowledge in P. This inclusion could be strict since some of the S_j^s may be secure channel terms, and hence, are not written into the intruder's knowledge. Let $\mu(\overline{\mathcal{I}}^{s,v_i}) = \mathcal{I}_0 \cup \{\overline{S}_1^s\mu, \ldots, \overline{S}_{i-1}^s\mu\}$ be the intruder's knowledge at v_i w.r.t. $\overline{\mathbf{B}}^s$. Recall that for $\overline{\mathbf{B}}^s$ all messages are written into the intruder's knowledge.

Obviously, point (i) still holds. By the induction hypothesis, we know that $\mu(\ell_V^s)(v_i) \in \mathcal{G}_P$ for $i \in \{1, \ldots, h\}$ and $\mu(\ell_V^s)(v_i) \overset{\mu(\ell_E^s)(v_i, v_{i+1})}{\longrightarrow} \mu(\ell_V^s)(v_{i+1}) \in \mathcal{G}_P$ for every $i \in \{1, \ldots, h-1\}$. We have to show that $\mu(\ell_V^s)(v_{h+1}) \in \mathcal{G}_P$ and $\mu(\ell_V^s)(v_h) \overset{\mu(\ell_E^s)(v_h, v_{h+1})}{\longrightarrow} \mu(\ell_V^s)(v_{h+1}) \in \mathcal{G}_P$. We distinguish between the different types of the symbolic transitions. For symbolic ε-transitions this is obvious.

For symbolic intruder transitions, the main point to show is that $R_h^s\mu \in d(\mu(\mathcal{I}^{s,v_h}))$. First assume that R_h^s is not a secure channel term. It follows that $R_h^s = \overline{R}_h^s$. Since μ satisfies condition a), we know that $\overline{R}_h^s\mu \in d(\mu(\overline{\mathcal{I}}^{s,v_h}))$. We know that $\mu(\mathcal{I}^{s,v_h}) \subseteq \mu(\overline{\mathcal{I}}^{s,v_h})$ and that the only difference between $\mu(\mathcal{I}^{s,v_h})$ and $\mu(\overline{\mathcal{I}}^{s,v_h})$ is that the latter set may contain secure channel encoding messages. Since μ, R_h^s, and the terms in $\mathcal{I}^{s,v_h}$ do not contain secure channel keys, we immediately obtain $R_h^s\mu \in d(\mu(\mathcal{I}^{s,v_h}))$ by Lemma 12. The argument in case R_h^s is a secure channel term is similar.

For symbolic secure channel transition, assume that $\ell_E^s(v_h, v_{h+1})$ is of the form $j, f, \mathsf{sc}(n, n', R'), \mathsf{sc}$. We have to show that $\mu(\mathsf{sc}(n, n', R')) \in \mu(\mathcal{S}^{s,v_h})$ and $\mu(R_{v_h}^s) = \mu(\mathsf{sc}(n, n', R'))$ where $\mathcal{S}^{s,v_h}$ denotes the secure channel in the symbolic state $\ell_V^s(v_h)$. By definition of symbolic secure channel transitions, we know that $\mathsf{sc}(n, n', R') \in \mathcal{S}^{s,v_h}$, and thus, $\mu(\mathsf{sc}(n, n', R')) \in \mu(\mathcal{S}^{s,v_h})$. If $R_{v_h}^s$ is of the form $\mathsf{sc}(n, n', R)$, then, by construction of $\overline{\mathbf{B}}^s$, $\overline{R}_h^s$ is of the form $\{R\}_k^s$ for some secure channel key k such that $\{R'\}_k^s \in \overline{\mathcal{I}}^{s,v_h}$ and k does not occur in any other term in $\overline{\mathcal{I}}^{s,v_h}$. We know that $\overline{R}_h^s\mu \in d(\overline{\mathcal{I}}^{s,v_h}\mu)$. Since k does not occur in μ nor in R and R', with Lemma 12 we obtain that $R\mu = R'\mu$, and thus, $\mu(R_h^s) = \mu(\mathsf{sc}(n, n', R'))$. We now have shown that the edges in $\mu(\mathbf{B}^s)$ correspond to transitions in $\mathcal{G}_P$.

The last step is to show that $\mu(\mathcal{T}_j^s)$ is a strategy tree for (C_j, C_j'). Properties 1. and 2. of Definition 3 are satisfied as we have just shown. Property 3. of Definition 3 follows

directly from Property 3. of the definition of symbolic strategy trees. Properties 4. and 5. of Definition 3 follow directly from the fact that μ passes the checks 2.,(a) and 2.,(b) in the last step of SolveStrategy.

6.2 Completeness of SolveStrategy

In this section, we show completeness of SolveStrategy. We have to show that if $(P, \mathcal{C}) \in$ G-STRATEGY, then there is a run of SolveStrategy in which yes is returned. By Remark 8, if $(P, \mathcal{C}) \in$ G-STRATEGY, there exists a branching structure $\mathbf{B}$ for P that satisfies $\mathcal{C}$. Given $\mathbf{B}$, our proof of completeness proceeds in three steps. First, we turn $\mathbf{B}$ into a symbolic branching structure $\mathbf{B}^s$ together with a substitution τ such that $\tau(\mathbf{B}^s) = \mathbf{B}$. Note that SolveStrategy can guess $\mathbf{B}^s$ in its first step. Second, we consider the constraint system $\mathbf{C}$ that is constructed in the second step of SolveStrategy if $\mathbf{B}^s$ was guessed in the first step of the algorithm and show that τ is a solution of $\mathbf{C}$. It follows that τ is also a solution of a simple constraint system, say $\mathbf{C}'$, in the sound and complete solution set of $\mathbf{C}$. Thus, there is a run of the constraint solver which outputs $\mathbf{C}'$. We finally show that the substitution $\tau_{\mathbf{C}'}$ associated with $\mathbf{C}'$ passes the tests performed in the third step of SolveStrategy, and hence, SolveStrategy outputs yes.

For our proof to work, we have to assume that $\mathbf{B}$ is what we call a minimal branching structure. We show that this assumption is w.l.o.g. Roughly speaking, minimal branching structures satisfy two properties: First, they should not contain superfluous transistions. Second, the secure channel transitions are in some sense complete. We now define these structures formally.

6.2.1 Notation and Basic Properties

To define minimal branching structures, we first need to introduce sc-functions and sc-completeness. To motivate these notions, we sketch the first step of our completeness proof, namely, how $\mathbf{B}$ is turned into a symbolic branching structure $\mathbf{B}^s$.

Given $\mathbf{B}$, the symbolic branching structure $\mathbf{B}^s$ and the substitution τ mentioned above are constructed in an inductive manner starting from the root of $\mathbf{B}$. Given a vertex v such that the symbolic state associated with v is already defined and the substitution τ is already defined for the variables in $\mathcal{X}_v$ (the set of variables used in vertex v),[1] we will define for each vertex $v' \in \mathbf{B}$ with $(v, v') \in \mathbf{B}$ the symbolic transition label of the edge (v, v') and the symbolic state associated with v' in $\mathbf{B}^s$ together with the substitution of the variables used in v'. In order to define the symbolic secure channel components of the state associated with vertex $v \in \mathbf{B}^s$ we will define so-called valid sc-functions. Informally speaking, a valid sc-function for a concrete branching structure $\mathbf{B}$ associates with each vertex $v \in \mathbf{B}$ a sequence of secure channel terms which corresponds

[1] Recall the definition of used variables from Section 5.1.

to a possible symbolic secure channel state in $v \in \mathbf{B}^s$. More precisely, a valid sc-function ℓ_{sc} associates with the root r of $\mathbf{B}$ the empty sequence, this corresponds to the empty secure channel in the symbolic state associated with r in $\mathbf{B}^s$. Whenever a secure channel message m is written into the secure channel by a transition from vertex v to v' in $\mathbf{B}$, then the sequence of secure channel terms associated with v' is that of v extended by the right-hand side of the principal rule associated with the edge (v, v'). Whenever a secure channel message m' is read from the secure channel, a particular secure channel term which matches with m' is removed from the sequence $\ell_{\mathsf{sc}}(v)$ of secure channel terms associated with v.

Formally, we call a sequence $s = (S_1, \ldots, S_k)$ of sc-terms, an sc-sequence. For a term S, a substitution σ of variables occurring in S_i for some i, and a message m, we say that the sc-sequence s' is a *(S, m, σ)-successor* of s if

$$s' = \begin{cases} (S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_k) & \text{if } S_i\sigma = m \text{ and } S_j \neq S_i \text{ for all } j < i \text{ and} \\ & S \text{ is not a sc-term,} \\ (S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_k, S) & \text{if } S_i\sigma = m \text{ and } S_j \neq S_i \text{ for all } j < i \text{ and} \\ & S \text{ is a sc-term.} \end{cases}$$

We call the term S_i the *term removed from* s.

A function ℓ_{sc} which maps every vertex $v \in \mathbf{B}$ to a sc-sequence is called an *sc-function*. Such a function is *valid* if for every $v \in \mathbf{B}$ the following conditions are satisfied:

1. $\mathcal{S}^v = \sigma^v(\{S_1, \ldots, S_k\})$ with $\ell_{\mathsf{sc}}(v) = (S_1, \ldots, S_k)$.

2. If v' is a successor of v in $\mathbf{B}$ and $\ell_E(v, v')$ is of the form i or i, m, I (i.e., the transition associated with (v, v') is an ε- or intruder transition), then

$$\ell_{\mathsf{sc}}(v') = \begin{cases} \ell_{\mathsf{sc}}(v) & \text{if } S \text{ is not an sc-term} \\ (S_1, \ldots, S_k, S) & \text{otherwise} \end{cases}$$

   where S is the right-hand side of the principal rule associated with (v, v').

3. If v' is a successor of v in $\mathbf{B}$ and $\ell_E(v, v')$ is of the form i, m, sc (i.e., the transition associated with (v, v') is a secure channel transition), then $\ell_{\mathsf{sc}}(v')$ is a (S, m, σ^v)-successor of $\ell_{\mathsf{sc}}(v)$ where S is the right-hand side of the principal rule associated with (v, v').

In 3., if S' is the term removed from $\ell_{\mathsf{sc}}(v)$, we call S' the *sc-term removed in* (v, v'). For ε- and intruder transitions, there is no removed sc-term.

As mentioned above, minimal branching structures satisfy two properties: i) there are not superfluous transitions and ii) the secure channel transitions are in some sense complete. To motivate the latter condition, we need to sketch the last step of our completeness proof.

In the last step of the proof of completeness of SolveStrategy, we show that $\tau_{\mathbf{C}'}$ passes the checks performed in the last step of SolveStrategy. Condition 2.,(a) of these checks says that all secure channel messages in the secure channel which could be read by applying a principal rule are in fact read by applying this rule. For this condition to be satisfied by $\tau_{\mathbf{C}'}$, we have to impose some specific structure on the branching structure $\mathbf{B}$.

Suppose, for example, that the symbolic secure channel $\mathcal{S}^{s,v}$ associated with $v \in \mathbf{B}^s$ contains terms $\mathsf{sc}(n, n', x)$ and $\mathsf{sc}(n, n', y)$ and $\tau(x) = \tau(y)$ where $\mathbf{B}^s$ and τ are the symbolic branching structure and substitution constructed from $\mathbf{B}$, respectively. Furthermore, assume that there is a principal rule of the form $\mathsf{sc}(n, n', z) \Rightarrow S$ applicable at v in $\mathbf{B}$, then, by definition of strategy trees (Definition 3, 4.) there has to be a secure channel transition with $\mathsf{sc}(n, n', z) \Rightarrow S$ being the associated principal rule of this transition where the message $\tau(\mathsf{sc}(n, n', x)) = \tau(\mathsf{sc}(n, n', y))$ is read from the secure channel. In the corresponding symbolic secure channel transition one of the two terms $\mathsf{sc}(n, n', x)$ and $\mathsf{sc}(n, n', y)$ is removed from the symbolic secure channel. The substitution $\tau_{\mathbf{C}'}$ does not have to fulfill the condition $\tau_{\mathbf{C}'}(x) = \tau_{\mathbf{C}'}(y)$ anymore. So the messages $\tau_{\mathbf{C}'}(\mathsf{sc}(n, n', x))$ and $\tau_{\mathbf{C}'}(\mathsf{sc}(n, n', y))$ are not necessarily the same. Still, both could be read when applying $\mathsf{sc}(n, n', z) \Rightarrow S$, and hence, in the concrete branching structure induced by $\mathbf{B}^s$ and $\tau_{\mathbf{C}'}$ there must be secure channel transitions for both messages. Therefore, we want to make sure that these transition already occur in $\mathbf{B}$. This is captured by the notion of sc-completeness.

We call $\mathbf{B}$ *sc-complete* if there is a valid sc-function ℓ_{sc} for $\mathbf{B}$ and for all j, $v \in V_j$, and successors v' of v in $\mathbf{B}$ such that[2]

- the label of the transition associated with (v, v') is i, m, sc for some i and $m \in \mathcal{S}^v$,
- the principal vertex associated with (v, v') is f for some vertex f in Π_i^v, and
- the right-hand side of the principal rule associated with (v, v') is S for some term S

the following conditions are satisfied: For every (S, m, σ^v)-successor s of $\ell_{\mathsf{sc}}(v)$, there is a successor v'' of v such that $\ell_{\mathsf{sc}}(v'') = s$, the label of the transition associated with (v, v'') is i, m, sc, and the principal vertex associated with (v, v'') is f.

We can now define minimal concrete branching structures. The conditions 2.-4. say that there are no superfluous transitions in the strategy trees of a minimal branching structure. These conditions correspond to the conditions 4. to 6. for symbolic strategy trees. The first condition is the completeness condition explained above.

Definition 15 *A branching structure* $\boldsymbol{B}$ *for* P *is called* minimal *if it satisfies the following conditions:*

[2]Recall the definition of V_j from the beginning of Section 6.

1. ***B*** *is* sc*-complete.*

2. *For all i and $(v, v'), (v, v'') \in E_i$ where $v' \neq v''$ and the transitions associated with (v, v') and (v, v'') are ε-transitions, it holds that the principals associated with (v, v') and (v, v''), respectively, differ or the principal vertices associated with (v, v') and (v, v''), respectively, differ.*

3. *For all i and $(v, v'), (v, v'') \in E_i$ where $v' \neq v''$ and the transitions associated with (v, v') and (v, v'') are secure channel transitions, it holds that the principals associated with (v, v') and (v, v''), respectively, differ, the principal vertices associated with (v, v') and (v, v''), respectively, differ, or the* sc*-terms removed in (v, v') and (v, v''), respectively, differ.*

4. *For all i and $(v, v'), (v, v'') \in E_i$ it holds that if $\ell_V(v, v') = j, m, I$ for some j and m, then $\ell_V(v, v'') = j, m, I$.*

The following lemma is easy to show.

Lemma 16 *If there exists a branching structure for P which satisfies $\mathcal{C}$, then there exists a minimal branching structure with this property.*

6.2.2 Proof of Completeness

Having defined minimal branching structures, we can now proceed with the proof of completeness. As mentioned, by Remark 8, we have to show that if there exists a branching structure $\mathbf{B}$ for P which satisfies $\mathcal{C}$, then there exists a successful run of SolveStrategy, i.e., SolveStrategy outputs yes.

By Lemma 16, we may assume that there exists a minimal branching structure $\mathbf{B}$ for P which satisfies $\mathcal{C}$. Let ℓ_{sc} be the corresponding valid sc-function for $\mathbf{B}$. Throughout the rest of this section, we use the notation as introduced in Section 6.2.1. We recall the main steps of the proof of completeness, which were briefly sketched above:

1. Construct a symbolic branching structure $\mathbf{B}^s$ and a substitution τ such that $\tau(\mathbf{B}^s) = \mathbf{B}$.

2. Show that τ is a solution of the constraint system $\mathbf{C}$ constructed from $\mathbf{B}^s$ in the second step of SolveStrategy.

3. Since τ is a solution of $\mathbf{C}$, we know that there exists a simple constraint system $\mathbf{C}'$ in the sound and complete solution set of $\mathbf{C}$ such that τ solves $\mathbf{C}'$. Let $\tau_{\mathbf{C}'}$ be the solution associated with $\mathbf{C}'$. We show that $\tau_{\mathbf{C}'}$ passes the checks performed in the third step of SolveStrategy.

The symbolic branching structure $\mathbf{B}^s$ in 1. can be guessed in the first step of algorithm SolveStrategy. Moreover, there is a computation path of the constraint solver such that $\mathbf{C}'$ is returned. Given 3., $\tau_{\mathbf{C}'}$ passes the checks in the third step of SolveStrategy. Hence, SolveStrategy returns yes, which shows that SolveStrategy is complete. It remains to show 1. to 3.:

1. Constructing $\mathbf{B}^s$. Our goal is to construct a symbolic branching structure $\mathbf{B}^s$ for P and a substitution τ such that $\tau(\mathbf{B}^s) = \mathbf{B}$. In particular, we define $\mathbf{B}^s$ in such a way that the set of vertices and edges of $\mathbf{B}^s$ coincides with the set of vertices and edges in $\mathbf{B}$, respectively. More precisely, define

- $V^{s,\pi} = V^\pi$ and $V^{s,i} = V^i$ for every i, and
- $E^{s,\pi} = E^\pi$ and $E^{s,i} = E^i$ for every i, and
- $r^{s,\pi} = r^\pi$ and $r^{s,i} = r^i$ for every i,

It remains to define the labeling functions ℓ^s_V and ℓ^s_E of $\mathbf{B}^s$ and the sets of used variables associated with each vertex $v \in V$. For the root r of π^s we set $\ell^s_V(r) = q^s_0$ where q^s_0 is the symbolic initial state of the protocol P. We define the set $\mathcal{X}_r$ of used variables to be empty.

Assume that for $v \in V$ the set of used variables $\mathcal{X}_v$ and the symbolic state $\ell^s_V(v)$ associated with v are already defined. Let $v' \in V$ with $(v, v') \in E$. We need to define $\ell^s_E(v, v')$ and $\ell^s(v')$.

We define the first component of $\ell_E(v, v')$ to be j. We know that $\ell_V(v) \xrightarrow{\ell_E(v,v')} \ell_V(v') \in \mathcal{G}_P$. Let f the principal vertex associated with this transition. Let $\mathcal{X}' = \mathcal{V}(R) \setminus \mathcal{X}_v$ where $\ell^v_j(r^v_j, f) = R \Rightarrow S$ is the principal rule associated with (v, v') in $\mathbf{B}$. Define $\Pi^{s,v'}_j = \widehat{\Pi}^{s,v}_j \downarrow f$ where $\widehat{\Pi}^{s,v}_j$ is $\Pi^{s,v}_j$ with variables $x \in \mathcal{X}'$ renamed by $x_{v'}$. In the following, we denote by $\hat{R}$ and $\hat{S}$ the terms R and S where the variables $x \in \mathcal{X}'$ are renamed by $x_{v'}$.

The set $\mathcal{X}_{v'}$ of used variables in v' is set to be $\mathcal{X}_{v'} = \mathcal{X}_v \cup \{x_{v'} \mid x \in \mathcal{X}'\}$.

To define the intruder knowledge $\mathcal{I}^{s,v'}$, the secure channel component $\mathcal{S}^{s,v'}$ of $\ell^s_V(v')$ and the edge label $\ell^s_E(v, v')$, we distinguish between the different types of transition labels $\ell_E(v, v')$.

- $\ell_E(v, v') = j$ (ε-transition): We define
  (a) $\ell^s_E(v, v') = j, f$.
  (b) $\mathcal{I}^{s,v'} = \mathcal{I}^{s,v} \cup \{\hat{S}\}$ if $\hat{S}$ is not a secure channel term and $\mathcal{I}^{s,v'} = \mathcal{I}^{s,v} \cup \{R'\}$ if $\hat{S}$ is a secure channel term of the form $\hat{S} = \mathsf{sc}(\cdot, \cdot, R')$.
  (c) $\mathcal{S}^{s,v'} = \mathcal{S}^{s,v}$ if $\hat{S}$ is not a secure channel term and $\mathcal{S}^{s,v'} = \mathcal{S}^{s,v} \cup \{\hat{S}\}$ if $\hat{S}$ is a secure channel term.

- $\ell_E(v,v') = j, m, I$ (intruder transition): We define $\ell^s_E(v,v') = j, f, I$ and the rest as in (b) and (c) above.

- $\ell_E(v,v') = j, m, \mathsf{sc}$ (secure channel transition): We define $\ell^s_E(v,v') = j, f, S', \mathsf{sc}$ where $\ell_{\mathsf{sc}}(v')$ is a $(\hat{S}, m, \sigma^v)$-succesor of $\ell_{\mathsf{sc}}(v)$ and S' is the term removed from $\ell_{\mathsf{sc}}(v)$. The intruder knowledge $\mathcal{I}^{s,v'}$ is updated as in (b) above. The secure channel component is updated to $\mathcal{S}^{s,v'} = \mathcal{S}^{s,v} \setminus \{S'\}$ if $\hat{S}$ is not a secure channel term and $\mathcal{S}^{s,v'} = \mathcal{S}^{s,v} \setminus \{S'\} \cup \{\hat{S}\}$ if $\hat{S}$ is a secure channel term.

Using the fact that $\mathbf{B}$ is a minimal (concrete) branching structure, it is easy to verify that $\mathbf{B}^s$ is a symbolic branching structure.

We now define the substitution τ. The domain of τ is the set $\mathcal{X} = \bigcup_{v \in V} \mathcal{X}_v$, i.e., the set of used variables in $\mathbf{B}^s$. For a variable $x_v \in \mathcal{X}$ we set $\tau(x_v) = \sigma^v(x)$. By induction on n, it is easy to see that for each root path $v_1, \ldots, v_n$ in $\mathbf{B}^s$ we have that $\tau(\ell^s_V(v_i)) = \ell_V(v_i)$ and $\tau(\ell^s_E(v_i, v_{i+1})) = \ell_E(v_i, v_{i+1})$ for every i. From this, it immediately follows that $\tau(\mathbf{B}^s) = \mathbf{B}$.

2. τ solves $\mathbf{C}$. We have to show that τ is a solution of the constraint system $\mathbf{C}$ constructed from $\overline{\mathbf{B}}^s$. There are three types of constraints in $\mathbf{C}$: First, constraints that were introduced for an edge $(v,v') \in E$, the intruder constraints; second, the unification constraints; and third, the strategy constraints.

Let $R : T$ be an intruder constraint of $\mathbf{C}$ that was introduced for the edge $(v,v') \in E$. We have to show that $\tau(R) \in d(\tau(T))$. There are three different cases that we have to distinguish: (a) $\ell_E(v,v') = j, m, I$ and m is not a secure channel message, (b) $\ell_E(v,v') = j, m, I$ and m is a secure channel message, and (c) $\ell_E(v,v') = j, m, \mathsf{sc}$.

(a) We know that $\tau(R) = \sigma^{v'}(R) = m \in d(\mathcal{I}^v)$ because R is the left-hand side of the principel rule associated with the edge (v,v') in $\mathbf{B}$. By construction of $\overline{\mathbf{B}}^s$, we have that $\mathcal{I}^v \subseteq \tau(\overline{\mathcal{I}}^{s,v}) = \tau(T)$ where $\overline{\mathcal{I}}^{s,v}$ is the intruder's knowledge at vertex v in $\overline{\mathbf{B}}^s$. Thus, it follows that $m \in d(\tau(T))$.

(b) We know that m is of the form $m = \mathsf{sc}(\mathbf{n}, \mathbf{n}', m')$ with $m' = \tau(R)$. We have that $m \in d(\mathcal{I}^v)$. Since, by construction, $\mathcal{I}^v$ does not contain secure channel terms, it follows that $\mathbf{n} \in \mathcal{I}^v$ and $m' \in d(\mathcal{I}^v)$. As in (a) we know that $\mathcal{I}^v \subseteq \tau(\overline{\mathcal{I}}^{s,v}) = \tau(T)$. Thus, $m' \in d(\tau(T))$.

(c) The term R is a term of the form $\{R'\}^s_{k_{w'}}$ where $k_{w'}$ is the sc-key introduced in the transition associated with an edge $(w,w') \in E$ where w is a predecessor of v. So the left-hand side of the principal rule associated with (v,v') in $\mathbf{B}^s$ has the form $\mathsf{sc}(\mathbf{n}, \mathbf{n}', R')$ and the right-hand side of the principal rule associated with (w,w') in $\mathbf{B}^s$ has the form $\mathsf{sc}(\mathbf{n}, \mathbf{n}', R'')$. We know that $\tau(\mathsf{sc}(\mathbf{n}, \mathbf{n}', R')) = m = \tau(\mathsf{sc}(\mathbf{n}, \mathbf{n}', R''))$. Hence, $\tau(\{R'\}^s_{k_{w'}}) = \tau(\{R''\}^s_{k_{w'}})$. Since $\{R''\}^s_{k_{v''}} \in T$, it follows that $\tau(\{R'\}^s_{k_{w'}}) \in d(\tau(T))$.

Now, let

$$\{\langle R_1, \langle R_2, \langle \ldots R_{n-2}, R_{n-1}\rangle \cdots\rangle\rangle\}^s_k : \mathcal{I}, \{\langle R_2, \langle R_3, \langle \ldots R_{n-1}, R_n\rangle \cdots\rangle\rangle\}^s_k$$

be a unification constraint that was introduced when traversing $v \in V$ in step 2. of SolveStrategy. Because $\mathbf{B}$ is minimal, by Definition 15, 4. we know that if v has more than one outgoing intruder transition, then the messages delivered are the same. From this it follows that τ fulfills the unification constraint above.

Since a strategy constraint is of the form $a : \mathcal{I}^{s,v}$ for a leaf $v \in \mathcal{T}_i$ and $a \in C_i$ for some i and $\tau(\mathcal{T}_i^s) = \mathcal{T}_i$ fulfills (C_i, C_i') we know that τ fulfills this constraint.

3. $\tau_{\mathbf{C}'}$ passes the tests. We will first show that τ passes the checks that are performed in step 3. of SolveStrategy and from this it will follow that $\tau_{\mathbf{C}'}$ passes these checks.

I) Let v be a leaf of some $\mathcal{T}_i$. We know that $C_i' \cap d(\mathcal{I}^v) = \emptyset$ and that $\tau(\ell_V^s(v)) = \ell_V(v)$. Consequently, τ passes the first check because $\mathcal{T}_i$ satisfies (C_i, C_i').

II) For every i and vertex $v \in V^i$, we have to show that the following conditions are satisfied:

(a) For every (ground term) $m \in \mathcal{S}^{s,v}\tau$, h, f such that
   i. $(r_h^{s,v}, f) \in E_h^{s,v}$,
   ii. $\ell_h^{s,v}(r_h^{s,v}, f) = R \to \cdot$ for some R,
   iii. $R\tau$ matches with m

   there is an edge $(v, v') \in E$ such that the label $\tau(\ell_E^s(v, v'))$ is h, m, sc and $\ell_E^s(v, v')$ has the form $h, f, \cdot, \mathsf{sc}$.

(b) If there exists an edge $(v, v') \in E$ such that
   i. the transition corresponding to the edge (v, v') is labeled with h, f, I,
   ii. $\ell_h^{s,v}(r^{s,v}, f) = R \to \cdot$ for some R, and
   iii. there exists f' such that $f' \neq f$, $(r_h^{s,v}, f') \in E_h^{s,v}$, $\ell_h^{s,v}(r_h^{s,v}, f') = R' \to \cdot$ for some R', and $R'\tau$ matches with $\hat{R}\tau$ where $\hat{R}$ is obtained from R by replacing all variables of the form x in R, i.e., those variables not indexed by vertices yet, by $x_{v'}$.

   then there exists v'' such that $(v, v'') \in E$ and the label of (v, v'') in $\mathbf{B}^s$ is h, f', I.

Let $v \in V^i$ for some i. To show (a), let $m \in \mathcal{S}^{s,v}\tau$, h, and f satisfy i, ii, and iii as above. We know that $\tau(\mathcal{T}_i^s) = \mathcal{T}_i$ for all i. Since $\mathcal{T}_i$ is a strategy tree, there is an edge $(v, v') \in E$ such that the label is $\tau(\ell_E^s(v, v')) = \ell_E(v, v') = h, m, \mathsf{sc}$. This completes test (a).

To show (b), let $(v, v') \in E$ and h, f, f', R, and R' satisfy i, ii, and iii as above. We know that $\tau(\mathcal{T}_i^s) = \mathcal{T}_i$ for all i. Since $\mathcal{T}_i$ is a strategy tree, there is a vertex v'' such that the label of (v, v'') is h, m, I and the principal vertex associated with (v, v') in $\mathbf{B}$ is f. By construction of $\mathbf{B}^s$, we have that the label of (v, v'') is h, f', I as desired.

So τ passes the checks performed in step 3. of SolveStrategy. Now we show that $\tau_{\mathbf{C}'}$ passes these checks as well.

A) Let v be a leaf of some $\mathcal{T}_i^s$ and suppose that there is an atom $a \in C_i'$ such that $a \in d(I_v^s \tau_{\mathbf{C}'})$. Then it is easy to see that $a \in d(I_v^s \tau)$, which is a contradiction to I).

B) We now show that $\tau_{\mathbf{C}'}$ passes the checks (a) and (b) of the third step of algorithm SolveStrategy. We start with (a). For this, let $m \in \mathcal{S}^{s,v}\tau_{\mathbf{C}'}$, h, f such that

   i. $(r_h^{s,v}, f) \in E_h^{s,v}$,

   ii. $\ell_h^{s,v}(r_h^{s,v}, f) = R \to S$ for some R, and

   iii. $R\tau_{\mathbf{C}'}$ matches with m.

   Let $R' \in \mathcal{S}^{s,v}$ such that $m = R'\tau_{\mathbf{C}'}$. Since $R\tau_{\mathbf{C}'}$ matches with $R'\tau_{\mathbf{C}'}$, it is easy to see that $R'\tau$ matches with $R\tau$ (since τ is obtained from $\tau_{\mathbf{C}'}$ by replacing intruder atoms by messages). By definition of ℓ_{sc}, there is a successor v' of v such that $r_h^{v'} = f$ and $\ell_{\mathsf{sc}}(v')$ is a valid (S, m, σ^v)-successor of $\ell_{\mathsf{sc}}(v)$ with the term R' removed. So by construction of $\mathbf{B}^s$, the label $\ell_E^s(v, v')$ is $\ell_E^s(v, v') = h, f, R', \mathsf{sc}$, and thus, $\tau_{\mathbf{C}'}(\ell_E^s(v, v')) = h, m, \mathsf{sc}$ as desired.

   Now we show that $\tau_{\mathbf{C}'}$ passes the test (b). For this let $(v, v') \in E, h, f, R, f'$ and R' satisfy i, ii, and iii as above. We have to show that there is a successor v'' of v in $\mathbf{B}$ such that the label of (v, v'') in $\mathbf{B}^s$ is h, f', I. Since we know that $R'\tau_{\mathbf{C}'}$ matches with $\hat{R}\tau_{\mathbf{C}'}$, as above we obtain that $R'\tau$ matches with $\hat{R}\tau$. Since τ passes the test (b), there exists a vertex v'' such that $(v, v'') \in E$ and the label of (v, v'') in $\mathbf{B}^s$ is h, f', I.

This completes the proof of completeness of SolveStrategy.

7 Conclusion

We have shown that game-theoretic security properties, such as balance, of contract-signing and related protocols can be decided using standard constraint solving procedures as a black-box. This opens the way for extending existing constraint-based implementations and tools, which have successfully been employed for reachability properties, to deal with game-theoretic security properties.

References

[1] R.M. Amadio, D. Lugiez, and V. Vanackere. On the symbolic reduction of processes with cryptographic functions. *Theoretical Computer Science*, 290(1):695–740, 2002.

[2] A. Armando, D. Basin, M. Bouallagui, Y. Chevalier, L. Compagna, S. Mödersheim, M. Rusinowitch, M. Turuani, L. Viganò, and L. Vigneron. The AVISS Security Protocol Analysis Tool. In E. Brinksma and K. G. Larsen, editors, *Proceedings of the 14th International Conference on Computer Aided Verification (CAV 2002)*, volume 2404 of *Lecture Notes in Computer Science*, pages 349–353. Springer, 2002.

[3] N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 86–99, 1998.

[4] D. Basin, S. Mödersheim, and L. Viganò. An On-The-Fly Model-Checker for Security Protocol Analysis. In E. Snekkenes and D. Gollmann, editors, *Proceedings of the 8th European Symposium on Research in Computer Security (ESORICS 2003)*, volume 2808 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2003.

[5] R. Chadha, M.I. Kanovich, and A.Scedrov. Inductive methods and contract-signing protocols. In P. Samarati, editor, *8-th ACM Conference on Computer and Communications Security (CCS 2001)*, pages 176–185. ACM Press, 2001.

[6] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. In *Proceedings of the Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS 2003)*, pages 261–270. IEEE, Computer Society Press, 2003.

[7] Y. Chevalier and L. Vigneron. A Tool for Lazy Verification of Security Protocols. In *Proceedings of the 16th IEEE Conference on Automated Software Engineering (ASE 2001)*, pages 373–376. IEEE CS Press, 2001.

[8] P. H. Drielsma and S. Mödersheim. The ASW Protocol Revisited: A Unified View. In *Workshop on Automated Reasoning for Security Protocol Analysis (ARSPA)*, 2004.

[9] J.A. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free optimistic contract signing. In *Advances in Cryptology – CRYPTO'99, 19th Annual International Cryptology Conference*, volume 1666 of *Lecture Notes in Computer Science*, pages 449–466. Springer-Verlag, 1999.

[10] D. Kähler, R. Küsters, and Th. Wilke. Deciding Properties of Contract-Signing Protocols. In Volker Diekert and Bruno Durand, editors, *Proceedings of the 22nd*

Symposium on Theoretical Aspects of Computer Science (STACS 2005), number 3404 in Lecture Notes in Computer Science, pages 158–169. Springer-Verlag, 2005.

[11] S. Kremer and J.-F. Raskin. Game analysis of abuse-free contract signing. In *Computer Security Foundations Workshop 2002 (CSFW 2002)*, pages 206–220. IEEE Computer Society, 2002.

[12] J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 166–175. ACM Press, 2001.

[13] M. Rusinowitch and M. Turuani. Protocol insecurity with a finite number of sessions, composed keys is NP-complete. *Theoretical Computer Science*, 299(1–3):451–475, 2003.

[14] V. Shmatikov and J.C. Mitchell. Finite-state analysis of two contract signing protocols. *Theoretical Computer Science (TCS), special issue on Theoretical Foundations of Security Analysis and Design*, 283(2):419–450, 2002.