Deciding Strategy Properties of Contract-Signing Protocols [1]

Detlef Kähler
Institut für Informatik
Christian-Albrechts-Universität zu Kiel, 24098 Kiel, Germany
kaehler@ti.informatik.uni-kiel.de

and
Ralf Küsters
Universität Trier
FB IV - Informatik, 54286 Trier, Germany
kuesters@uni-trier.de

and
Thomas Wilke
Institut für Informatik
Christian-Albrechts-Universität zu Kiel, 24098 Kiel, Germany
wilke@ti.informatik.uni-kiel.de

Research on the automatic analysis of cryptographic protocols has so far concentrated on reachability properties, such as secrecy and authentication. In this paper, we prove that certain game-theoretic security properties, including balance for contract-signing protocols, can be decided in a Dolev-Yao style model with a bounded number of sessions. The decision algorithm that we develop is based on standard constraint-solving procedures, which, in the past, have successfully been employed in tools for reachability properties. Our result thus paves the way for extending these tools to deal with game-theoretic security properties.

Categories and Subject Descriptors: []: Computer Security

General Terms: Security Protocols, Verification

Additional Key Words and Phrases: Contract Signing, Decidability, Automatic Security Analysis

1. INTRODUCTION

Automatic analysis of cryptographic protocols has been studied intensively in the recent past (see, e.g., [Comon and Shmatikov 2002; Meadows 2003] for an overview). One of the central results of the area is that the security of cryptographic protocols when analyzed with respect to a finite number of sessions, without a bound on the message size, and in presence of the so-called Dolev-Yao intruder is decidable (see,

[1]This work is based on extended abstracts, [Kähler et al. 2005] and [Kähler and Küsters 2005]. It was partially supported by the *Deutsche Forschungsgemeinschaft (DFG)* under Grant KU 1434/4-2 and KU 1434/5-1 as well as the *Schweizerischer Nationalfonds* (SNF) under Grant 200021-116596.

e.g., [Rusinowitch and Turuani 2003; Amadio et al. 2002; Boreale 2001; Millen and Shmatikov 2001]). Based on this result, fully automatic, industrial-strength debugging tools (see, e.g., [Armando et al. 2005; Chevalier and Vigneron 2001; Millen and Shmatikov 2001]) have been developed and successfully applied to find flaws in published protocols (see, e.g., [Basin et al. 2003]). However, the mentioned decidability result and the tools developed are restricted to security properties such as authenticity and secrecy, which are reachability properties of the transition system associated with a protocol: Is a state in which the intruder possesses a certain secret, such as a session key, reachable? In contrast, crucial properties required of contract-signing and related protocols (see, e.g., [Garay et al. 1999; Asokan et al. 1998; Ben-Or et al. 1990; Zhou and Gollmann 1996]), for instance abuse-freeness [Garay et al. 1999] and balance [Chadha et al. 2001], are 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.

The main goal of this paper is to show that the aforementioned decidability result extends in a natural way to branching properties, such as balance, and similar properties of contract-signing protocols. In other words, the goal is to show that these branching properties are decidable with respect to a finite number of sessions, without a bound on the message size, and in presence of the so-called Dolev-Yao intruder. Many of the tools described above employ so-called constraint solving procedures (see, e.g., [Millen and Shmatikov 2001; Chevalier and Vigneron 2001; Basin et al. 2003]). A further goal is to demonstrate that constraint-solving procedures are still useful in the broader context of game-theoretic properties.

The protocol and intruder model that we suggest to use is similar to a model proposed in [Chadha et al. 2001]; it contains different features important for contract-signing protocols, which are absent in the models for authentication and key exchange protocols referred to above. First, as in [Chadha et al. 2001], we model write-protected channels which are *not* under the control of the intruder. For simplicity, in this paper we call these channels *secure channels*, although this notion is also used in cryptography with a different meaning. Second, for protocols in our model we explicitly define the induced transition systems. These transition systems have infinitely many states and are infinitely branching, but have paths of bounded length; they allow us to state crucial properties of contract-signing properties.

Our main result is that for the transition system induced by a cryptographic protocol properties expressing the existence of certain strategies for the intruder are decidable. We also show how to reduce balance to these properties, which then implies that balance is decidable. For deciding the strategy properties, we develop a constraint-based algorithm, which can be built on top of standard constraint-solving procedures (see, e.g., [Millen and Shmatikov 2001; Chevalier and Vigneron 2001; Basin et al. 2003] and references therein). As mentioned, such procedures have successfully been employed for deciding reachability properties and proved to be a good basis for practical implementations (see, e.g., [Chevalier and Vigneron 2001; Millen and Shmatikov 2001; Corin and Etalle 2002; Basin et al. 2003; Armando et al. 2005]). 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 a game-theoretic security property, first the algorithm guesses what we call a symbolic branching structure. This structure represents a potential attack on the protocol. In the second step of the algorithm, the symbolic branching structure is turned into a constraint system. Then a standard constraint-solving procedure 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 a 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 satisfies certain requirements. The first step as described above is analogous to the first step in a constraint-based algorithm for checking reachability properties of cryptographic protocols, where an interleaving of the actions of the individual principals is guessed.

There are some crucial differences of our constraint-based algorithm to algorithms for reachability properties: First, for reachability properties only interleavings, i.e., linear structures, instead of branching 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 5).

Further Related Work. In several papers, contract-signing and related protocols have been analyzed using finite-state model checkers (see, e.g., [Shmatikov and Mitchell 2002; Kremer and Raskin 2002; Chadha et al. 2004]). Due to the restriction to a finite state set, the Dolev-Yao intruder is, however, only approximated. A much more detailed model has been considered by Chadha et al. [Chadha et al. 2001], who analyzed the contract-signing protocol proposed by Garay, Jakobsson, and MacKenzie (the GJM protocol) [Garay et al. 1999]. However, the analysis was carried out by hand and without tool support. Our results show that the analysis for the balance property carried out in [Chadha et al. 2001] can be fully automated (given a specification of the protocol) without resorting to finite-state models as done in previous works. In [Chadha et al. 2005], Chadha et al. study impossibility results related to properties of contract-signing protocols.

Drielsma and Mödersheim [Drielsma and Mödersheim 2004] apply an automatic tool based on constraint solving originally intended for authentication and key exchange protocols in the analysis of the Asokan-Shoup-Waidner (ASW) protocol [Asokan et al. 1998]. Their analysis is, however, restricted to reachability properties as branching properties cannot be handled by their tool. Also, secure channels are not modeled explicitly in that paper.

As mentioned, the present paper is based on two extended abstracts, [Kähler et al. 2005] and [Kähler and Küsters 2005]. In [Kähler et al. 2005], the first decidability result for branching time properties, as described above, was presented. However, the decision procedure developed in [Kähler et al. 2005] only exploits the fact that the size of attacks can be bounded, and hence, attacks of bounded size can be guessed and checked. This is obviously unsuitable for practical implementations. In [Kähler and Küsters 2005], the above mentioned constraint-based algorithm has therefore been devised. In the present paper, this algorithm along with full proofs is presented.

We note that probabilistic contract-signing and related protocols, such as the probabilistic contract-signing protocol proposed in [Ben-Or et al. 1990], are out of the scope of the work presented here. Analysis of such protocols is the focus of the work in [Norman and Shmatikov 2006], in which a probabilistic finite model checker is employed.

Structure of the Paper. In Section 2, we introduce our protocol and intruder model, with an example provided in Section 3. In Section 4, the intruder strategies and the game-theoretic properties that we consider are presented. Section 5 provides the necessary background on constraint solving. In Section 6, we present our constraint-based decision algorithm along with an example and state our main result—soundness, completeness, and termination of the algorithm. In Section 7, the proofs of the main result are given. We conclude in Section 8.

2. THE PROTOCOL AND INTRUDER MODEL

As mentioned in the introduction, our model is similar to the one by Chadha et al. [Chadha et al. 2001]. When it comes to the technical exposition, our approach is, however, inspired by the term-rewriting approach of [Rusinowitch and Turuani 2003] rather than the multi-set rewriting approach of [Chadha et al. 2001]. We start with a rather precise, but informal description of our model.

In our model, a protocol is a finite set of principals and every principal is a finite tree, which represents all possible behaviors 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. Whether or not a principal gets an input and which input it gets is determined by the intruder and the secure channels. The intruder receives every message sent by a principal, can use all the messages he has received to construct new messages, and can provide input messages to any principal he wants.

The above is very similar to standard Dolev-Yao models for reachability properties (see, e.g., [Rusinowitch and Turuani 2003; Millen and Shmatikov 2001]). There are, however, two important ingredients that are not present in these models: secure channels and an explicit branching structure. In the remainder of this introduction to our model, we will explain these two ingredients in more detail.

Unlike in the standard Dolev-Yao models, in our model the input of a principal may not only come from the intruder but also from a so-called secure channel. While, as in [Chadha et al. 2001], a secure channel is not read-protected (the intruder can read the messages written onto this channel), the intruder does not control this channel. That is, he cannot delay, duplicate, or remove messages, or write messages onto this channel under a fake identity (unless he has corrupted a party).

As mentioned in the introduction, unlike authentication and key-exchange protocols, properties of contract-signing and related protocols cannot be stated as reachability properties, i.e., in terms of single runs of a protocol alone. One rather has to consider branching properties. We therefore describe the behavior of a protocol as an infinite-state transition system which comprises all runs of a protocol. To be able to express properties of contract-signing protocols we distinguish several types of transitions: there are intruder transitions (just as in [Rusinowitch and Turuani 2003; Millen and Shmatikov 2001]); there are ε-transitions, which can be used to model that a subprotocol is spawned without waiting for input from the intruder; and there are secure channel transitions, which model communication via secure channels. Since the intruder can construct an infinite number of messages, the transition system will have an infinite number of states, but it will have paths of a bounded length.

In the following subsections, we present a formal definition of our model.

2.1 Terms and Messages

Let $\mathcal{V}$ be a set of variables, $\mathcal{A}$ a finite set of atoms, $\mathcal{K}$ a finite set of public and private keys, $\mathcal{A}_I$ an infinite set of intruder atoms, and $\mathcal{N}$ a finite set of principal addresses. All of them are assumed to be disjoint. The set $\mathcal{K}$ is partitioned into a set $\mathcal{K}_{\mathsf{pub}}$ of public keys and a set $\mathcal{K}_{\mathsf{priv}}$ of private keys and 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 composed 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 grammar given in Figure 1, namely *plain terms* and *secure channel terms*. In this grammar, $\mathcal{V}$, $\mathcal{A}$, $\mathcal{A}_I$, $\mathcal{N}$, and $\mathcal{K}_{\mathsf{pub}}$ generate any of its elements.

Terms without variables are called *messages*. Similarly, plain terms and secure channel terms without variables are called *plain messages* and *secure channel messages*, respectively. The corresponding sets are denoted $\mathcal{T}$, $\mathcal{T}_{\mathsf{plain}}$, $\mathcal{T}_{\mathsf{sc}}$, $\mathcal{M}$, $\mathcal{M}_{\mathsf{plain}}$, and $\mathcal{M}_{\mathsf{sc}}$.

$$\begin{array}{rcl}\mathsf{plain\text{-}term} & ::= & \mathcal{V} \mid \mathcal{A} \mid \mathcal{A}_I \mid \langle \mathsf{plain\text{-}term}, \mathsf{plain\text{-}term}\rangle \mid \\ & & \{\mathsf{plain\text{-}term}\}^{\mathsf{s}}_{\mathsf{plain\text{-}term}} \mid \{\mathsf{plain\text{-}term}\}^{\mathsf{a}}_{\mathcal{K}_{\mathsf{pub}}} \mid \mathsf{hash}(\mathsf{plain\text{-}term}) \mid \\ & & \mathsf{sig}_{\mathcal{K}_{\mathsf{pub}}}(\mathsf{plain\text{-}term}) \\ \mathsf{sec\text{-}term} & ::= & \mathsf{sc}(\mathcal{N}, \mathcal{N}, \mathsf{plain\text{-}term}) \\ \mathsf{term} & ::= & \mathsf{plain\text{-}term} \mid \mathsf{sec\text{-}term} \mid \mathcal{N}\end{array}$$

Fig. 1. Syntax of Terms

As usual, $\langle t, t'\rangle$ is the pairing of t and t', the term $\{t\}^{\mathsf{s}}_{t'}$ stands for the symmetric encryption of t by t' (note that the key t' may be any plain term), $\{t\}^{\mathsf{a}}_{k}$ 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 a term t on the secure channel from n to n'. A principal may only generate such a term if he knows n and t (but does not need to know n'). This guarantees that a principal cannot impersonate other principals on secure channels. Hence, knowing n grants access to secure channels with sender address n.

Note that the above explanations are just to give some intuition on what these messages mean and how they can be used. In our formal model, there will be no restriction on the use of these messages in protocol descriptions except for the rules by which the intruder can derive messages.

By $\mathcal{V}(t)$ we denote the set of variables occurring in a term t. The set $\mathrm{Sub}(t)$ of subterms of a term t and the set $\mathrm{Sub}(E)$ of subterms of the terms in a set E of terms are defined in the obvious way.

A *substitution* assigns terms to variables, that is, it is a function $\mathcal{V} \to \mathcal{T}$. The *domain* of a substitution σ 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. A substitution σ is called *ground* if $\sigma(x)$ is a message for each $x \in \mathsf{dom}(\sigma)$. Given two substitutions σ and τ with disjoint domains, their union $\sigma \cup \tau$ is defined in the obvious way. Given a term t and a substitution σ, the term $t\sigma$ is obtained from t by simultaneously replacing each occurrence of a variable x in t by $\sigma(x)$. If σ and τ are two substitutions, then their *composition*, denoted $\sigma \circ \tau$, is the substitution defined by $(\sigma \circ \tau)(x) = (x\tau)\sigma$ for every $x \in \mathcal{V}$.

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

2.2 Intruder

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

(D1) $\mathcal{I} \subseteq d(\mathcal{I})$.

(D2) *Composition and decomposition*: If $t, t' \in d(\mathcal{I})$, then $\langle t, t'\rangle \in d(\mathcal{I})$. Conversely, if $\langle t, t' \rangle \in d(\mathcal{I})$, then $t \in d(\mathcal{I})$ and $t' \in d(\mathcal{I})$.

(D3) *Symmetric encryption and decryption*: If $t, t' \in d(\mathcal{I})$, then $\{t\}^{\mathsf{s}}_{t'} \in d(\mathcal{I})$. Conversely, if $\{t\}^{\mathsf{s}}_{t'} \in d(\mathcal{I})$ and $t' \in d(\mathcal{I})$, then $t \in d(\mathcal{I})$.

(D4) *Asymmetric encryption and decryption*: If $t \in d(\mathcal{I})$ and $k \in d(\mathcal{I}) \cap \mathcal{K}_{\mathsf{pub}}$, then $\{t\}^{\mathsf{a}}_{k} \in d(\mathcal{I})$. Conversely, if $\{t\}^{\mathsf{a}}_{k} \in d(\mathcal{I})$ and $k^{-1} \in d(\mathcal{I}) \cap \mathcal{K}_{\mathsf{priv}}$, then $t \in d(\mathcal{I})$.

(D5) *Hashing*: If $t \in d(\mathcal{I})$, then $\mathsf{hash}(t) \in d(\mathcal{I})$.

(D6) *Signing*: If $t \in d(\mathcal{I})$ and $k^{-1} \in d(\mathcal{I}) \cap \mathcal{K}_{\mathsf{priv}}$, then $\mathsf{sig}_k(t) \in d(\mathcal{I})$. (The signature contains the public key but can only be generated if the corresponding private key is known.)

(D7) *Writing onto secure channels:* If $t \in d(\mathcal{I})$, $n \in d(\mathcal{I}) \cap \mathcal{N}$, and $n' \in \mathcal{N}$, then $\mathsf{sc}(n, n', t) \in d(\mathcal{I})$.

(D8) *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 (D5) does not apply when t is of the form $\mathsf{sc}(n, n', t')$.

Note that the above definition is slightly more general than in other papers: Typically, one only considers sets $\mathcal{I}$ which consist exclusively of messages. Later, the more general definition, which allows arbitrary sets of terms for $\mathcal{I}$, will be useful.

In (D7), the precondition $n \in d(\mathcal{I}) \cap \mathcal{N}$ 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 $R \in \mathcal{T} \cup \{\varepsilon\}$ and $S \in \mathcal{T}$. We refer to R as the *left-hand side (LHS)* of $R \Rightarrow S$ and to S as the *right-hand side (RHS)* of $R \Rightarrow S$. A principal rule of the form $\varepsilon \Rightarrow S$ models that a message of the form S can be output by a principal without waiting for external input. For example, such a rule is used to model that a principal starts a protocol or a sub-protocol without waiting for external input.

A *rule tree* is a finite rooted tree where the edges are labeled by principal rules. Formally, a rule tree is a tuple $\Pi = (V, E, r, \ell)$ where (V, E) is a finite rooted tree as usual (in particular, $E \subseteq V \times V$), r is the root of (V, E), and ℓ maps every edge $(v, v') \in E$ to a principal rule, *the principal rule associated with* (v, v').

A *principal* is a rule tree Π as above which satisfies the following two conditions for every vertex $v \in V$ where we assume that $v_0, \ldots, v_n$ is the unique path from the root r to v and where $R_i \Rightarrow S_i$ is the principal rule associated with (v_i, v_{i+1}), for every $i < n$.

(1) *Well-formedness:* $\mathcal{V}(S_{n-1}) \subseteq \bigcup_{i<n} \mathcal{V}(R_i)$.

(2) *Feasibility:* $t \in d(\{R_0, \ldots, R_{n-1}\} \cup \mathcal{M})$ for every verification key t in R_{n-1}.

Well-formedness requires that every variable occurring on the RHS of a principal rule of an edge e also occurs on the LHS of the principal rule of some edge on the path from the root to e (inclusive). It guarantees that any output produced by a principal only depends on what he has received, which is a minimum requirement for a reasonable protocol specification. This condition is standard in Dolev-Yao models, see, for instance, [Rusinowitch and Turuani 2003; Millen and Shmatikov 2001].

Feasibility makes sure that a principal cannot match variables in keys, which would not be feasible in practice. For example, this condition is not satisfied for a principal whose rule tree only contains one edge and where this edge is labeled with the principal rule $\{y\}^{\mathsf{s}}_x \Rightarrow x$. On the other hand, it is satisfied for a principal whose rule tree is a path of length two and where the first edge is labeled by $x \Rightarrow \langle x, x\rangle$ and the second edge is labeled as before (by $\{y\}^{\mathsf{s}}_x \Rightarrow x$).

Let Π be a rule true. When v is a vertex of Π, we write $\Pi{\downarrow}v$ for the subtree of Π rooted at v. When σ is a substitution, we write $\Pi\sigma$ for the rule tree obtained from Π by simultaneously applying σ to all the principal rules (each side of them) of Π.

A *protocol* is a tuple $P = ((\Pi_1, \dots, \Pi_n), \mathcal{I})$ which consists of a tuple of principals and a finite set $\mathcal{I} \subseteq \mathcal{M}_{\mathsf{plain}} \cup \mathcal{N}$ of messages, the *initial intruder knowledge.* We require that each variable occurs in the rules of at most 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 (neither in any principal rule nor in $\mathcal{I}$).

2.4 The Transition Graph of a Protocol

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

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

(1) σ is a ground substitution,

(2) Π_i is a rule tree such that $\Pi_i\sigma$ is a principal for each i,

(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 transition system is in 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, $\Pi_i\sigma$ 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.

There are three kinds of transitions between states: intruder, secure channel, and ε-transitions. In what follows, for every $i \in \{1, \dots, n\}$, 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. A transition

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

with label τ exists if one of the three following conditions is satisfied:

(1) *Intruder transition,* $\tau = [i, m, I]$: There exists a vertex $v \in V_i$ such that $(r_i, v) \in E_i$ and $\ell_i(r_i, v) = R \Rightarrow S$, and there exists a ground substitution σ'' with $\mathsf{dom}(\sigma'') \subseteq \mathcal{V}(R\sigma)$ 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) either $S \in \mathcal{T}_{\mathsf{plain}}$ and $\mathcal{I}' = \mathcal{I} \cup \{S\sigma'\}$, or $S = \mathsf{sc}(\cdot,\cdot,t)$ for some term t and $\mathcal{I}' = \mathcal{I} \cup \{t\sigma'\}$, and

(f) either $S \in \mathcal{T}_{\mathsf{plain}}$ and $\mathcal{S}' = \mathcal{S}$, or $S \in \mathcal{T}_{\mathsf{sc}}$ and $\mathcal{S}' = \mathcal{S} \cup \{S\sigma'\}$.

This transition models that principal i receives the the message m from the intruder (i.e., from the public network). Note that the second clause in (e) accounts for the fact that our secure channels are not read-protected.

(2) *Secure channel transition,* $\tau = [i, m, \mathsf{sc}]$: There exists a vertex $v \in V_i$ with $(r_i, v) \in E_i$ and $\ell_i(r_i, v) = R \Rightarrow S$, and there exists a ground substitution σ'' with $\mathsf{dom}(\sigma'') \subseteq \mathcal{V}(R\sigma)$ such that

(a) $m \in \mathcal{S}$,

(b)–(e) as above, and

(f) either $S \in \mathcal{T}_{\mathsf{plain}}$ and $\mathcal{S}' = \mathcal{S} \setminus \{m\}$, or $S \in \mathcal{T}_{\mathsf{sc}}$ and $\mathcal{S}' = (\mathcal{S} \setminus \{m\}) \cup \{S\sigma'\}$.

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

(3) *ε-transition,* $\tau = [i]$: There exists a vertex $v \in V_i$ with $(r_i, v) \in E_i$ and $\ell_i(r_i, v) = \varepsilon \Rightarrow S$ such that $\sigma' = \sigma$ and (d)–(f) as given above for intruder transitions.

This transition models that principal i performs a step where neither a message is read from the intruder nor from a 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), respectively, the *principal rule associated with the transition*, and v the *principal vertex associated with the transition.*

Definition 2.1 *(transition graph).* Let $P = ((\Pi_1, \ldots, \Pi_n), \mathcal{I})$ be a protocol.

The *initial state of* P is $q_P = ((\Pi_1, \ldots, \Pi_n), \sigma, \mathcal{I}, \emptyset)$ where σ is the substitution with empty domain. The *set of states of* P, denoted S_P, is the set of states which are reachable from q_P by a sequence of transitions.

The *transition graph* $\mathcal{G}_P$ *of* 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 of P, and E_P is the set of all transitions among the states of P.

For convenience, we write $q \in \mathcal{G}_P$ when q is a state in $\mathcal{G}_P$ and we write $q \xrightarrow{\tau} q' \in \mathcal{G}_P$ when $q \xrightarrow{\tau} q'$ is a transition in $\mathcal{G}_P$.

Remark 2.2. 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. MODELING THE ORIGINATOR OF THE ASW PROTOCOL

To demonstrate that our framework can actually be used to analyze contract-signing protocols, we show how the originator of the Asokan-Shoup-Waidner (ASW) protocol [Asokan et al. 1998] can be modeled. In a similar fashion, other contract-signing protocols can be dealt with.

3.1 Overview of the Protocol

Our informal description of the ASW protocol follows [Shmatikov and Mitchell 2002] (see this work or [Asokan et al. 1998] for more details). For ease in notation, we will write $\mathsf{sig}[m, k]$ instead of $\langle m, \mathsf{sig}_k(m)\rangle$.

The ASW protocol enables two principals O (originator) and R (responder) to obtain each other's commitment on a previously agreed contractual text, say text, with the help of a trusted third party T, which, however, is only invoked in case of problems. In other words, the ASW protocol is an optimistic two-party contract-signing protocol.

There are two kinds of valid contracts: the standard contract,

$$\langle \mathsf{sig}[m_O, k_O], N_O, \mathsf{sig}[m_R, k_R], N_R\rangle \quad ,$$

and the replacement contract, $\mathsf{sig}[\langle \mathsf{sig}[m_O, k_O], \mathsf{sig}[m_R, k_R]\rangle, k_T]$, where

$$\begin{aligned} m_O &= \langle k_O, k_R, k_T, \mathsf{text}, \mathsf{hash}(N_O)\rangle \quad , \\ m_R &= \langle \mathsf{sig}[m_O, k_O], \mathsf{hash}(N_R)\rangle \quad , \end{aligned}$$

and k_O, k_R, and k_T are the public keys of O, R, and T, respectively, and N_O and N_R represent nonces generated by O and R, respectively. Note that a signed contractual text ($\mathsf{sig}[\mathsf{text}, k_O]$ or $\mathsf{sig}[\mathsf{text}, k_R]$) is not considered a valid contract.

The ASW protocol consists of four subprotocols: the exchange, abort, and two resolve protocols. However, we can describe every principal—O, R, and T—in terms of a single tree as introduced in Section 2.3.

The basic idea of the exchange protocol is that O first indicates his/her interest to sign the contract. To this end, O hashes a nonce N_O and signs it together with text and the keys of the principals involved. The resulting message is the message $\mathsf{sig}[m_O, k_O]$ from above. By sending it to R, the originator O commits to the contract. Then, similarly, R indicates his/her interest to sign the contract by hashing a nonce N_R and signing it together with what he/she received from O and the keys of the involved principals. This is the message $\mathsf{sig}[m_R, k_R]$ from above. By sending it to O, the responder R commits to the contract. Finally, first O and then R reveal N_O and N_R, respectively. This is why a standard contract is only valid if N_O and N_R are included.

If, after O has sent the first message, R does not respond, O may contact T to abort. Also, if after O has sent the second message, the nonce N_O, and R does not respond, then O may contact T to resolve. Similarly, R may contact T to resolve the protocol after having sent the message $\mathsf{sig}[m_R, k_R]$. However, R may not abort the protocol. In case the protocol is successfully resolved, the replacement contract $\mathsf{sig}[\langle \mathsf{sig}[m_O, k_O], \mathsf{sig}[m_R, k_R]\rangle, k_T]$ is issued. While this version of the contract only contains the message indicating O's and R's intention to sign the contract (and neither N_O nor N_R), the signature of T validates the contract. For T to respond to incoming resolve or abort requests appropriately it needs to store the requests in some kind of database. For example, if O has sent an abort request and afterwards R sends a resolve request T has to answer by an abort token to act consistently.

As we will see in the next subsection the model of the originator O of the ASW protocol as a rule tree is straightforward. To model the TTP T as a rule tree one has to encode the database which is needed to distinguish between different possible

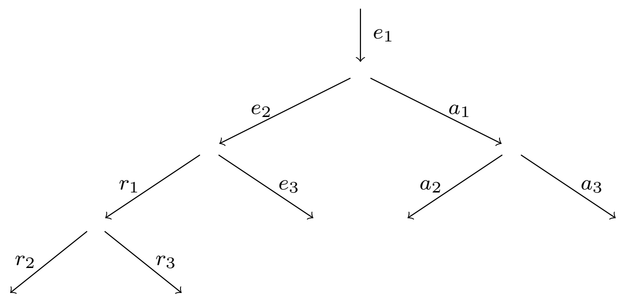


Fig. 2. A model of the Originator O in the ASW protocol

interleavings of resolve and abort requests. This can be done by unwinding these possible interleavings into a rule tree.

3.2 The Principal O

The principal O is defined by the tree Π_O depicted in Figure 2 where the edge labels for the principal rules defined below. Rules $e1$, $e2$, and $e3$ belong to the exchange protocol, rules $a1$, $a2$, and $a3$ belong to the abort protocol, and rules $r1$, $r2$, and $r3$ belong to the resolve protocol of O.

Exchange protocol. The actions performed in the exchange protocol have informally been discussed above.

Abort protocol. If, after the first step of the exchange protocol, O does not get an answer back from R, the principal O may start the abort protocol, i.e., send an abort request via a secure channel to T (rule $a1$). Then, T will either confirm the abort of the protocol by returning an abort token—in this case O will continue with rule $a3$—or send a replacement contract—in this case O will continue with rule $a2$. (The trusted third party T sends a replacement contract if R previously contacted T to resolve the protocol run.)

Resolve protocol. If after rule $e2$, i. e., after sending N_O, the principal O does not get an answer back from R, then O can start the resolve protocol by sending a resolve request to T via the secure channel (rule $r1$). After that, depending on the answer returned from T (which again will return an abort token or a replacement contract), one of the rules $r2$ or $r3$ is performed.

We now present the principal rules for O where the numbering corresponds to the one in Figure 2. In some of these rules we use extra constants which indicate certain events. We therefore call these constants *signal constants.* If, for example, principal O performs rule e_3, i.e., it gets the signature of the responder on the contract, it sends the signal constant $\mathsf{OHasValidContract}$ on the network, indicating that O now has a valid contract. Then, the property that at the end of a protocol execution the originator O does not have a valid contract is formalized by requiring that the intruder cannot derive the constant $\mathsf{OHasValidContract}$ at the end of a protocol execution. How security properties are modeled in our framework is described in detail in the next section.

$$\begin{tikzcd}
\Pi_1 : & & f_1 \arrow[d, "\mathsf{sc}(2,1,\langle x,b\rangle) \Rightarrow x"'] & \\
& & f_2 \arrow[d, "\{y\}^{\mathsf{s}}_k \Rightarrow y"'] & \\
& & f_3 \arrow[dl, "a \Rightarrow c_1"'] \arrow[d, "x \Rightarrow c_2"] \arrow[dr, "y \Rightarrow c_2"] & \\
& f_4 & f_5 & f_6
\end{tikzcd}
\qquad
\begin{tikzcd}
\Pi_2 : & & g_1 \arrow[dl, "\epsilon \Rightarrow \mathsf{sc}(2,1,\langle a,b\rangle)"'] \arrow[dr, "\epsilon \Rightarrow \mathsf{sc}(2,1,\langle b,b\rangle)"] & \\
& g_2 & & g_3
\end{tikzcd}$$

Fig. 3. Protocol $P_{\mathsf{ex}} = (\{\Pi_1, \Pi_2\}, \mathcal{I}_0)$ with $\mathcal{I}_0 = \{\{a\}^{\mathsf{s}}_k, \{b\}^{\mathsf{s}}_k\}$

(e_1) $\varepsilon \Rightarrow me_1$ where

$$me_1 = \mathsf{sig}[me_2, k_O] \quad \text{and} \quad me_2 = \langle k_O, k_R, k_T, \mathsf{text}, \mathsf{hash}(N_O)\rangle\,.$$

(e_2) $\mathsf{sig}[me_3, k_R] \Rightarrow N_O$ where $me_3 = \langle me_1, \mathsf{hash}(x)\rangle$.

(e_3) $x \Rightarrow \mathsf{OHasValidContract}$.

(a_1) $\varepsilon \Rightarrow \mathsf{sc}(O, T, ma_1)$ where $ma_1 = \mathsf{sig}[\langle \mathsf{aborted}, me_1\rangle, k_O]$.

(a_2) $\mathsf{sc}(T, O, ma_2) \Rightarrow \mathsf{OHasValidContract}$ where

$$ma_2 = \mathsf{sig}[\langle me_1, me_4\rangle, k_T] \quad \text{and} \quad me_4 = \mathsf{sig}[\langle me_1, z_1\rangle, k_R].$$

(a_3) $\mathsf{sc}(T, O, \mathsf{sig}[\langle \mathsf{aborted}, ma_1\rangle, k_T]) \Rightarrow \mathsf{OHasAbortToken}$.

(r_1) $\varepsilon \Rightarrow \mathsf{sc}(O, T, \langle me_1, \mathsf{sig}[me_3, k_R]\rangle)$.

(r_2) $\mathsf{sc}(T, O, \mathsf{sig}[\langle \mathsf{aborted}, ma_1\rangle, k_T]) \Rightarrow \mathsf{OHasAbortToken}$.

(r_3) $\mathsf{sc}(T, O, mr_1) \Rightarrow \mathsf{OHasValidContract}$ where

$$mr_1 = \mathsf{sig}[\langle me_1, mr_2\rangle, k_T] \quad \text{and} \quad mr_2 = \mathsf{sig}[\langle me_1, z_2\rangle, k_R].$$

4. STRATEGIES, STRATEGY PROPERTIES, AND STRATEGY PROBLEMS

In this section, we define intruder strategies on transition graphs of protocols, the type of goal these strategies try to achieve, and the formal decision problems we are interested in. We start with an informal explanation following an example and then turn to precise statements.

Throughout this and the following sections, we will use the protocol P_{ex} depicted in Figure 3 as our running example. This protocol consists of two principals Π_1 and Π_2 and the initial knowledge $\mathcal{I}_0 = \{\{a\}^{\mathsf{s}}_k, \{b\}^{\mathsf{s}}_k\}$ of the intruder. Informally speaking, Π_2 can, without waiting for input from the secure channel or the intruder, decide whether to write $\langle a, b\rangle$ or $\langle b, b\rangle$ into the secure channel from Π_2 to Π_1. While the intruder can read the message written into this channel, he cannot modify or delay this message. Also, he cannot insert his own message into this channel, for he does not have the principal address 2 in his intruder knowledge, and hence, cannot generate messages of the form $\mathsf{sc}(2, n, t)$ for any name n. Consequently, such messages must come from Π_2. Principal Π_1 first waits for a message of the form $\langle x, b\rangle$ in the secure channel from Π_2 to Π_1. In case Π_2 wrote, say, $\langle a, b\rangle$ into this channel, x is substituted by a, and this message is written into the network, and hence, given to the intruder. Next, Π_1 waits for input of the form $\{y\}^{\mathsf{s}}_{k}$. This is not a secure channel term, and thus, comes from the intruder. In case the intruder sends $\{b\}^{\mathsf{s}}_{k}$, say, then y is substituted by b. Finally, Π_1 waits for input of the form a (in the edges from f_3 to f_4 and f_3 to f_5) or b (in the edge from f_3 to f_6). Recall that x was substituted by a and y by b. If the intruder sends b, say, then Π_2 takes the edge from f_3 to f_6 and outputs c_2 into the network. If the intruder had sent a, Π_1 could have chosen between the first two edges. We note that this protocol is not meant to perform a useful task. It is rather designed to illustrate different aspects of our constraint-based algorithm that would be more cumbersome to demonstrate by realistic protocols, such as the ASW or GJM protocol.

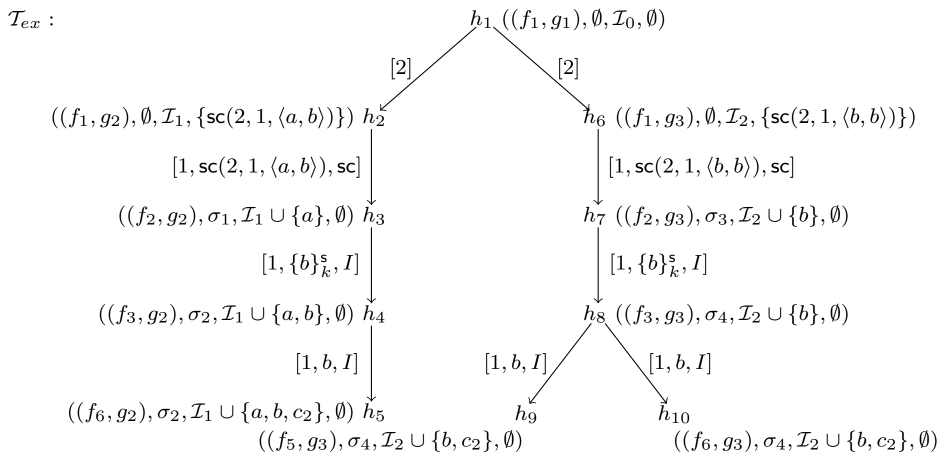
4.1 Intruder Strategies and Strategy Trees

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 (signal) constants and cannot derive others. For example, for balance the signal constant IntruderHasContract should be derivable by the intruder but HonestPartyHasContract should not be derivable by the intruder. The constant HonestPartyHasContract would be added to the specification of the honest principal, as illustrated in Section 3.2, where the constant OHasValidContract takes the role of HonestPartyHasContract. To check whether the intruder obtained a valid contract from the honest party, a new principal would be defined with only one principal rule. The principal rule would expect to receive a message that is of the form of a valid contract from the honest principal and then, if such a message is received, output the signal constant IntruderHasContract.

More concretely, let us consider the protocol P_{ex} depicted in Figure 3. We want to know if the intruder has a strategy to get to a state where he can derive atom c_2 but not atom c_1 (no matter what the principals Π_1, Π_2, and the secure channels do). Such a strategy of the intruder has to deal with both decisions principal Π_2 may make in the first step because the intruder cannot control which edge is taken by Π_2. It turns out that regardless of which message is sent by principal Π_2 in its first step, the following simple strategy allows the intruder to achieve his goal: The intruder can send $\{b\}^{\mathsf{s}}_{k}$ to principal Π_1 in the second step of Π_1 and in the last step of Π_1, the intruder sends b to principal Π_1. This guarantees that in the last step of Π_1, the left-most edge is never taken, and thus, c_1 is not returned, but at least one of the other two edges can be taken, which in any case yields c_2. Formally, such strategies are defined as trees. In our example, the strategy tree corresponding to the strategy informally explained above is depicted in Figure 4. Its construction is in accordance with the following definition of strategy trees.

Definition 4.1 *(strategy tree)*. Let P be a protocol with n principals and $q \in \mathcal{G}_P$. A *q-strategy tree* is a rooted tree $\mathcal{T}_q = (V, E, r, \ell_V, \ell_E)$ where every vertex $v \in V$ is labeled by a state, $\ell_V(v) \in \mathcal{G}_P$, and every edge $(v, v') \in E$ is labeled by a label of a transition, $\ell_E(v, v')$, such that the following conditions are satisfied:

(T1) $\ell_V(r) = q$.



$$\begin{tikzcd}
\mathcal{T}_{ex}: & h_1\ ((f_1,g_1),\emptyset,\mathcal{I}_0,\emptyset) \arrow[dl, "{[2]}"'] \arrow[dr, "{[2]}"] & & \\
((f_1,g_2),\emptyset,\mathcal{I}_1,\{\mathsf{sc}(2,1,\langle a,b\rangle)\})\ h_2 \arrow[d, "{[1,\mathsf{sc}(2,1,\langle a,b\rangle),\mathsf{sc}]}"'] & & h_6\ ((f_1,g_3),\emptyset,\mathcal{I}_2,\{\mathsf{sc}(2,1,\langle b,b\rangle)\}) \arrow[d, "{[1,\mathsf{sc}(2,1,\langle b,b\rangle),\mathsf{sc}]}"] & \\
((f_2,g_2),\sigma_1,\mathcal{I}_1\cup\{a\},\emptyset)\ h_3 \arrow[d, "{[1,\{b\}^{\mathsf{s}}_k,I]}"'] & & h_7\ ((f_2,g_3),\sigma_3,\mathcal{I}_2\cup\{b\},\emptyset) \arrow[d, "{[1,\{b\}^{\mathsf{s}}_k,I]}"] & \\
((f_3,g_2),\sigma_2,\mathcal{I}_1\cup\{a,b\},\emptyset)\ h_4 \arrow[d, "{[1,b,I]}"'] & & h_8\ ((f_3,g_3),\sigma_4,\mathcal{I}_2\cup\{b\},\emptyset) \arrow[dl, "{[1,b,I]}"'] \arrow[dr, "{[1,b,I]}"] & \\
((f_6,g_2),\sigma_2,\mathcal{I}_1\cup\{a,b,c_2\},\emptyset)\ h_5 & h_9\ ((f_5,g_3),\sigma_4,\mathcal{I}_2\cup\{b,c_2\},\emptyset) & & h_{10}\ ((f_6,g_3),\sigma_4,\mathcal{I}_2\cup\{b,c_2\},\emptyset)
\end{tikzcd}$$

Fig. 4. Strategy tree $\mathcal{T}_{ex}$ for P_{ex} with $\mathcal{I}_1 = \mathcal{I}_0 \cup \{\langle a,b\rangle\}$, $\mathcal{I}_2 = \mathcal{I}_0 \cup \{\langle b,b\rangle\}$, $\sigma_1 = \{x \mapsto a\}$, $\sigma_2 = \sigma_1 \cup \{y \mapsto b\}$, $\sigma_3 = \{x \mapsto b\}$, and $\sigma_4 = \sigma_3 \cup \{y \mapsto b\}$. Also, for brevity of notation, in the first component of the states we write, for instance, f_1 instead of $\Pi_1{\downarrow}f_1$. The strategy property we consider is $((C_{ex}, C'_{ex})) = ((\{c_2\}, \{c_1\}))$.

(T2) $\ell_V(v) \xrightarrow{\ell_E(v,v')} \ell_V(v') \in \mathcal{G}_P$ for all $(v,v') \in E$.

(T3) Whenever $\ell_V(v) = q'$ and $q' \xrightarrow{[j]} q'' \in \mathcal{G}_P$ for some $v \in V$, $q', q'' \in \mathcal{G}_P$, and $j \in \{1,\ldots,n\}$, then there exists $v'' \in V$ such that $(v,v'') \in E$, $\ell_V(v'') = q''$, and $\ell_E(v,v'') = [j]$.

(T4) Whenever $\ell_V(v) = q'$ and $q' \xrightarrow{[j,m,\mathsf{sc}]} q'' \in \mathcal{G}_P$ for some $v \in V$, $q', q'' \in \mathcal{G}_P$, $j \in \{1,\ldots,n\}$, and $m \in \mathcal{M}$, then there exists $v'' \in V$ such that $(v,v'') \in E$, $\ell_V(v'') = q''$, and $\ell_E(v,v'') = [j,m,\mathsf{sc}]$.

(T5) Whenever $(v,v') \in E$, $\ell_E(v,v') = [j,m,I]$, and there exists $q' \neq \ell_V(v')$ with $\ell_V(v) \xrightarrow{[j,m,I]} q' \in \mathcal{G}_P$ for some $v \in V$, $q' \in \mathcal{G}_P$, $j \in \{1,\ldots,n\}$, and $m \in \mathcal{M}$, then there exists v'' with $(v,v'') \in E$, $\ell_E(v,v'') = [j,m,I]$ and $\ell_V(v'') = q'$.

(T2) guarantees that all edges in $\mathcal{T}_q$ correspond to transitions in $\mathcal{G}_P$. (T3) says that every ε-transition of the transition graph of P 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, since these rules do not depend on input from the intruder. (T4) is similar: The intruder should not be able to block secure channels. (T5) 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 the LHS of more than one rule).

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 2.2 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 in every leaf of $\mathcal{T}_q$ all elements from C can be derived by the intruder and no element from C' can. Formally, for every leaf v of $\mathcal{T}_q$ with $\ell_V(v) = ((\Pi_1,\ldots,\Pi_n),\sigma,\mathcal{I},\mathcal{S})$ it is required that $C \subseteq d(\mathcal{I})$ and $C' \cap d(\mathcal{I}) = \emptyset$ hold.

Definition 4.2 *(strategy property).* A *strategy property* is a tuple

$$\mathcal{C} = ((C_1, C_1'), \ldots, (C_l, C_l')) \tag{1}$$

where $C_i, C_i' \subseteq \mathcal{A} \cup \mathcal{K} \cup \mathcal{N}$ for all $i \in \{1, \ldots, l\}$. A state $q \in \mathcal{G}_P$ *satisfies* $\mathcal{C}$ if there exist q-strategy trees $\mathcal{T}_1, \ldots, \mathcal{T}_l$ such that every $\mathcal{T}_i$ satisfies (C_i, C_i').

4.2 Strategy Problems

We can now define the decision problem we are interested in:

Definition 4.3 *(*STRATEGY*).* The decision problem STRATEGY asks, given a protocol P and a strategy property $\mathcal{C}$, whether there exists a state $q \in \mathcal{G}_P$ that satisfies $\mathcal{C}$. When this is the case we write $(P, \mathcal{C}) \in$ 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.

In general, positional strategies can be very restrictive. Here, however, we work in a framework where the "games" are of finite depth (bounded number of steps) and the winning conditions are reachability conditions. In such situations, positional strategies are no restriction, as we will show in what follows for our framework.

A q-strategy tree $\mathcal{T}_q$ is *positional* if different subtrees rooted at vertices with the same vertex label are isomorphic. (In particular, one could define positional strategies as subgraphs of $\mathcal{G}_P$.) We define P-STRATEGY analogously to STRATEGY, but with positional intruder strategies instead of arbitrary intruder strategies, and use similar notation.

The above assertion can now be stated formally:

PROPOSITION 4.4. *For every instance* $(P, \mathcal{C})$ *of* STRATEGY *(and* P-STRATEGY*),*

$$(P, \mathcal{C}) \in \text{STRATEGY} \qquad \textit{iff} \qquad (P, \mathcal{C}) \in \text{P-STRATEGY} .$$

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, starting with the subtrees of maximum depth and iteratively 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. □

Note that in [Kähler et al. 2005] positional intruder strategies are considered. The main motivation for using general rather than positional intruder strategies in this paper is that they are much better suited for the constraint solving approach (see Section 6).

5. CONSTRAINT SOLVING

In this section, we briefly recall the notion of constraint system and state the well-known fact that procedures for solving constraint systems exist (see, e.g., [Millen and Shmatikov 2001] for more details). In Section 6, we will then use such a procedure as a black box in our constraint-based algorithm.

A *constraint* is of the form $t\colon T$ 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 do not need to consider them here.) A sequence

$$\mathbf{C} = t_1\colon T_1, \dots, t_n\colon T_n$$

of constraints is called a *constraint system.*

Given a constraint system $\mathbf{C}$ as above, one would like to determine all ground substitutions σ that satisfy $t_i\sigma \in d(T_i\sigma)$ for every $i \in \{1, \dots, n\}$ and where $\mathsf{dom}(\sigma)$ contains only variables occurring in $\mathbf{C}$. These substitutions are called *solutions* of $\mathbf{C}$ and one writes $\sigma \vdash \mathbf{C}$ for such a substitution. Since there is, in general, an infinite number of such solutions, all solutions are typically described by a finite number of so-called simple constraint systems, which have obvious solutions, as will be explained in what follows.

For our purposes, a *simple constraint system* is of the form

$$\mathbf{C}' = x_1\colon T'_1, \dots, x_m\colon T'_m$$

where the x_j's are pairwise distinct variables. Such a system has obvious solutions, namely, every substitution σ where every variable is assigned a different intruder atom from $\mathcal{A}_I$. Such a solution will be called *solution associated with* $\mathbf{C}'$.

Given a constraint system $\mathbf{C}$, a constraint solver produces a finite set U of pairs $(\mathbf{C}', \tau)$ where $\mathbf{C}'$ is simple and τ is a substitution such that

—no variable from $\mathbf{C}'$ belongs to $\mathsf{dom}(\tau)$, and

—for every $x \in \mathsf{dom}(\tau)$, every variable occurring in $\tau(x)$ also occurs in $\mathbf{C}'$.

The important property of U is that for every $(\mathbf{C}', \tau) \in U$ and for every solution σ' of $\mathbf{C}'$, the substitution $\sigma' \circ \tau$ is a solution of $\mathbf{C}$, in particular, this is true for every solution associated with $\mathbf{C}'$ as described above. This is why U is called a *sound set* for $\mathbf{C}$. It is said to be a *sound and complete set* for $\mathbf{C}$ if, in addition, every solution of $\mathbf{C}$ can be obtained in the described way.

If σ' is a solution associated with $\mathbf{C}'$ the solution $\sigma' \circ \tau$ will be called *complete solution associated with* $(\mathbf{C}', \tau)$.

If a given constraint system $\mathbf{C}$ satisfies certain conditions, which will be described below, sound and complete sets for $\mathbf{C}$ can be computed.

For a given constraint system $\mathbf{C}$ as above and a variable x in $\mathbf{C}$, define $\mathsf{occ}(x) \in \{1, \dots, n\}$ to be the minimal index such that $x \in \mathcal{V}(t_{\mathsf{occ}(x)})$, i.e., $\mathsf{occ}(x)$ is the index of the first constraint in the sequence where x occurs on the left-hand side of this constraint. If such an index does not exist, $\mathsf{occ}(x)$ is undefined.

We call a constraint system $\mathbf{C}$ as above *valid* if it satisfies the following properties for every $i \in \{1, \dots, n\}$:

(1) *Origination:* $\mathcal{V}(T_i) \subseteq \mathcal{V}(\{t_1, \dots, t_{i-1}\})$, which means that $\mathsf{occ}(x)$ is defined and $\mathsf{occ}(x) < i$ for every $x \in \mathcal{V}(T_i)$.

(2) *Monotonicity:*
   (a) $T_1 \subseteq T_i$.
   (b) For every $x \in \mathcal{V}(T_i)$ we have that $T_{\mathsf{occ}(x)} \subseteq T_i$.

Intuitively, origination corresponds to well-formedness of principals and monotonicity captures that the intruder does not forget information. Our definition of monotonicity above differs from the definition in [Millen and Shmatikov 2001].[2] Our definition is slightly stronger. However, it is simpler and suffices for our applications. From a technical point of view, we could also have used the more complex definition in [Millen and Shmatikov 2001].

The following fact is well-known (see, e.g., [Chevalier and Vigneron 2001; Millen and Shmatikov 2001; Basin et al. 2003] and references therein):

FACT 5.1. *There exists a procedure which given a valid constraint system $\boldsymbol{C}$ outputs a sound and complete set for $\boldsymbol{C}$.*

In our model we deal with an infinite set $\mathcal{A}_I$ of intruder atoms. Constraint solving procedures do not handle these kind of sets of atoms. It is easy to see that the set output by a constraint solving procedure for a given constraint system $\mathbf{C}$ also forms a sound and complete set for $\mathbf{C}$ when one considers the set $\mathcal{A}_I$ of intruder atoms.

While different constraint solving procedures (and implementations thereof) may compute different sound and complete sets, our constraint-based algorithm to be introduced in Section 6 works with any of them. It is only important that the set computed is sound and complete. So we fix one of these procedures for the remainder of the paper and refer to it as *constraint solver.* As already mentioned in the introduction we only need one element of a sound and complete set at a time. We therefore view the constraint solver as a non-deterministic algorithm which at the end of each run on a given input returns an element from a (fixed) sound and complete set. We require that for every element in the sound and complete set, there exists a run of the constraint solver that returns it. 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. This is, for example, the case for private contract signatures, which are used in some contract signing protocols [Garay et al. 1999]; a good starting point for such an extension in this case could be the model of private contract signatures presented in [Kähler et al. 2005].

6. THE CONSTRAINT-BASED ALGORITHM

We now present our constraint-based algorithm, called SolveStrategy, for deciding the problem STRATEGY. We first describe its main steps, with details given in subsequent sections. (In particular, the notions set in italics will be explained

[2]Using our notation, the definition in [Millen and Shmatikov 2001] reads as follows: (a) $T_1 \subseteq T_i$. (b) For every $x \in \mathcal{V}(T_i)$ there exists $T_{\bar{x}} \subseteq T_i$ such that (i) $x \notin \mathcal{V}(T_{\bar{x}})$, (ii) for all variables y with $occ(y) > occ(x)$ it holds that $y \notin \mathcal{V}(T_{\bar{x}})$, and (iii) for all solutions σ of $\mathbf{C}$ it holds that $d(T_{occ(x)}) \subseteq d(T_{\bar{x}})$. Note that our definition implies this definition: Simply choose $T_{\bar{x}}$ to be $T_{occ(x)}$.

later.) The input to SolveStrategy is a protocol P and a strategy property $\mathcal{C} = ((C_1, C'_1), \ldots, (C_l, C'_l))$.

SolveStrategy$(P, \mathcal{C})$: {yes, no}

(A1) Guess a *symbolic branching structure* $\mathbf{B}$ for P and $\mathcal{C}$.
That is, guess a *symbolic path* π^s from the initial state of P to a *symbolic state* q^s and, for every $i \in \{1, \ldots, l\}$, a *symbolic q^s-strategy tree* $\mathcal{T}^s_{i,q^s}$ (see Section 6.1 for details).

(A2) From the symbolic branching structure $\mathbf{B}$ and the strategy property $\mathcal{C}$, derive the *induced valid constraint system* $\mathbf{C}_\mathbf{B}$ (see Section 6.2 for the definition).

(A3) Run the constraint solver on $\mathbf{C}_\mathbf{B}$. If it returns no, then halt. Otherwise, let $(\mathbf{C}', \tau)$ be the output returned by the solver and ν a complete solution associated with it.

(A4) Perform certain tests on $\mathbf{B}$ and ν to 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 the q-strategy trees $\mathcal{T}_{i,q}$ satisfying (C_i, C'_i) for every $i \in \{1, \ldots, l\}$ (see Section 6.3 for details of these tests).

(A5) If any of the tests failed, output no, and otherwise output yes.

In the following three sections, (A1), (A2), and (A4) will be further explained. Our main result is the following theorem, with the proof presented in Section 7:

Theorem 6.1. *Algorithm SolveStrategy is a (non-deterministic) decision procedure for* Strategy.

Corollary 6.2. Strategy *is decidable.*

We conclude this subsection with some remarks.

Remark 6.3. In (A3), it is not sufficient to consider just one simple constraint system returned by the constraint solver.

Remark 6.4. (A4) is indispensable.

The two previous remarks will be illustrated in Section 6.3.

Remark 6.5. If yes is output, then $\mathbf{B}$ with ν applied is a solution of $(P, \mathcal{C})$.

The previous remark will become clear from the proof of Theorem 6.1.

Remark 6.6. SolveStrategy is phrased as a non-deterministic algorithm only for simplicity of presentation; one can easily turn it into a deterministic one, simply by successively checking all symbolic branching structures (of which there are only finitely many) and all simple constraint systems returned by a deterministic constraint solver.

Remark 6.7. SolveStrategy works with *any* constraint solving procedure.

Remark 6.8. The proof of Theorem 6.1 (see Section 7) is quite different from the proof of Corollary 6.2 given in [Kähler et al. 2005]. Here, we make use of the fact that constraint solvers 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.

6.1 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 several other notions. These notions will be illustrated by the example in Figure 3.

One technical problem we have to overcome is that a symbolic strategy tree will represent a set of runs of protocols arranged in trees. Clearly, when two principal rules sharing variables are applied in independent branches of such a tree, the shared variables that have not been instantiated during the common history of the two branches need not be instantiated with the same terms. This makes it necessary to apply a renaming scheme for the variables involved: When x is a variable from the original protocol and u is a vertex of a symbolic tree (to be defined), we will use x_u to denote a new variable. We speak of *non-indexed* and *indexed* variables.

A *symbolic state*

$$q^s = ((\Pi_1, \ldots, \Pi_n), \mathcal{I}, \mathcal{S}, \mathcal{X}) \tag{2}$$

consists of an n-tuple $(\Pi_1, \ldots, \Pi_n)$ of rule trees, a finite set $\mathcal{I} \subseteq \mathcal{T}$ of terms (which may contain variables), a finite multi-set $\mathcal{S} \subseteq \mathcal{T}_{\mathsf{sc}}$ of secure channel terms (which may contain variables), and a finite set $\mathcal{X}$ of variables, which are called *used variables* and are used for renaming variables correctly, as explained in the previous paragraph. Obviously, a symbolic state is defined just as a concrete state except that the substitution is omitted, the intruder knowledge $\mathcal{I}$ and the secure channel $\mathcal{S}$ may contain terms instead of messages, and a set of reference variables has been added.

A *symbolic tree* is of the form $(V, E, r, \ell_V, \ell_E)$ with finite vertex and edge set V and E, respectively, root $r \in V$, vertex labeling function ℓ_V, which maps vertices to symbolic states, and edge labeling function ℓ_E, which maps edges to labels of symbolic transitions. Let $(u, u') \in E$ be an edge and assume

$$\begin{aligned}
\ell_V(u) &= ((\Pi_1, \ldots, \Pi_n), \mathcal{I}, \mathcal{S}, \mathcal{X}) \ , \quad && \Pi_i = (V_i, E_i, r_i, \ell_i) \text{ for } i \in \{1, \ldots, n\} \ , \\
\ell_V(u') &= ((\Pi'_1, \ldots, \Pi'_n), \mathcal{I}', \mathcal{S}', \mathcal{X}') \ , \quad && \Pi'_i = (V'_i, E'_i, r'_i, \ell'_i) \text{ for } i \in \{1, \ldots, n\} \ , \\
\ell_E((u, u')) &= \tau \ .
\end{aligned}$$

Then one the following conditions must be satisfied.

(1) *Symbolic intruder transition,* $\tau = [i, v, I]$: $v \in V_i$ with $(r_i, v) \in E_i$ and $\ell_i(r_i, v) = R \Rightarrow S$ for some $i \in \{1, \ldots, n\}$, 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 new variable $x_{u'}$ (see above);
   (b) $\mathcal{I}' = \mathcal{I} \cup \{t'\}$ where either $t = S \notin \mathcal{T}_{\mathsf{sc}}$ or $S = \mathsf{sc}(n, n', t)$ for some $n, n' \in \mathcal{N}$, and t' is obtained from t by replacing every variable $x \in \mathcal{V}(R) \setminus \mathcal{X}$ by the new indexed variable $x_{u'}$;
   (c) either $\mathcal{S}' = \mathcal{S}$ and $S \notin \mathcal{T}_{\mathsf{sc}}$ or $\mathcal{S}' = \mathcal{S} \cup \{S'\}$, where S' is obtained from S by replacing every variable $x \in \mathcal{V}(R) \setminus \mathcal{X}$ by $x_{u'}$.

(2) *Symbolic secure channel transition,* $\tau = [i, v, R', \mathsf{sc}]$: $v \in V_i$ 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 either $\mathcal{S}' = \mathcal{S} \setminus \{R'\}$ if $S \notin \mathcal{T}_{\mathsf{sc}}$ or $\mathcal{S}' = (\mathcal{S} \setminus \{R'\}) \cup \{S'\}$ otherwise.

$$\begin{tikzcd}
\mathcal{T}_{ex}: & h_1\ ((f_1,g_1),\emptyset,\mathcal{I}_0,\emptyset,\emptyset) \arrow[dl, swap, "{[2,g_2]}"] \arrow[dr, "{[2,g_3]}"] & & \\
((f_1,g_2),\emptyset,\mathcal{I}_1,\{\mathsf{sc}(2,1,\langle a,b\rangle)\},\emptyset)\ h_2 \arrow[d, swap, "{[1,f_2,\mathsf{sc}(2,1,\langle a,b\rangle),\mathsf{sc}]}"] & & h_6\ ((f_1,g_3),\emptyset,\mathcal{I}_2,\{\mathsf{sc}(2,1,\langle b,b\rangle)\},\emptyset) \arrow[d, "{[1,f_2,\mathsf{sc}(2,1,\langle b,b\rangle),\mathsf{sc}]}"] & \\
((f_2,g_2),\sigma_1,\mathcal{I}_1\cup\{x_{h_3}\},\emptyset,\{x_{h_3}\})\ h_3 \arrow[d, swap, "{[1,f_3,I]}"] & & h_7\ ((f_2,g_3),\sigma_3,\mathcal{I}_2\cup\{x_{h_7}\},\emptyset,\{x_{h_7}\}) \arrow[d, "{[1,f_3,I]}"] & \\
((f_3,g_2),\sigma_2,\mathcal{I}_1\cup\{x_{h_3},y_{h_4}\},\emptyset,\mathcal{X}_1)\ h_4 \arrow[d, swap, "{[1,f_6,I]}"] & & h_8\ ((f_3,g_3),\sigma_4,\mathcal{I}_2\cup\{x_{h_7},y_{h_8}\},\emptyset,\mathcal{X}_2) \arrow[dl, swap, "{[1,f_5,I]}"] \arrow[dr, "{[1,f_6,I]}"] & \\
((f_6,g_2),\sigma_2,\mathcal{I}_1\cup\{x_{h_3},y_{h_4},c_2\},\emptyset,\mathcal{X}_1)\ h_5 & h_9\ ((f_5,g_3),\sigma_4,\mathcal{I}_2\cup\{x_{h_7},y_{h_8},c_2\},\emptyset,\mathcal{X}_2) & & h_{10}\ ((f_6,g_3),\sigma_4,\mathcal{I}_2\cup\{x_{h_7},y_{h_8},c_2\},\emptyset,\mathcal{X}_2)
\end{tikzcd}$$

Fig. 5. Symbolic strategy tree $\mathcal{T}_{ex}$ for the protocol P_{ex} with $\mathcal{I}_1 = \mathcal{I}_0 \cup \{\langle a,b\rangle\}$, $\mathcal{I}_2 = \mathcal{I}_0 \cup \{\langle b,b\rangle\}$, $\mathcal{X}_1 = \{x_{h_3}, y_{h_4}\}$, and $\mathcal{X}_2 = \{x_{h_7}, y_{h_8}\}$. Note that, for sake of presentation, the first component of the symbolic states associated to vertices of $\mathcal{T}_{ex}$ only contain the roots of rule trees rather than the whole rule trees. The strategy property we consider is $((C_{ex}, C'_{ex})) = ((\{c_2\}, \{c_1\}))$.

(3) *Symbolic ε-transition,* $\tau = [i, v]$: $v \in V_i$ with $(r_i, v) \in E_i$ and $\ell_i(r_i, v) = \varepsilon \Rightarrow S$ with (a)–(c) from (1).

In addition, $\mathcal{X}' = \mathcal{X} \cup \{x_{u'} \mid x \in \mathcal{V}(R) \setminus \mathcal{X}\}$.

Just as with (concrete) transitions, in a situation like above, we speak of a *symbolic transition*, 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.* We call $\ell_V(u) \xrightarrow{\ell_E(u,u')} \ell_V(u')$ the *symbolic transition corresponding to (or associated with)* (u, u').

Let q^s be a symbolic state. A *symbolic q^s-tree* is a symbolic tree where the root is labeled q^s. Let $\mathcal{X}_{\mathcal{T}^s_{q^s}} = \bigcup_{v\in V} \mathcal{X}_v$ be the *set of used variables in* $\mathcal{T}^s_{q^s}$ where $\mathcal{X}_v$ is the set of used variables in state $\ell_V(v)$.

Figure 5 depicts a symbolic q^s_0-tree $\mathcal{T}_{ex}$ for P_{ex} (Figure 3) 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. Note that copies of the variables x and y have been introduced, namely $x_{h_3}, x_{h_7}, y_{h_4}, y_{h_8}$, following the aforementioned renaming scheme.

We can now define symbolic paths and symbolic strategy trees as special cases of symbolic trees.

Let $P = ((\Pi_1, \dots, \Pi_n), \mathcal{I})$ be a protocol as usual. A *symbolic path* π^s of P is a symbolic q^s-tree where $q^s = ((\Pi_1, \dots, \Pi_n), \mathcal{I}, \emptyset, \emptyset)$ is the *symbolic initial state of P* and every vertex has at most one successor.

If q^s is a symbolic state, then a *symbolic q^s-strategy tree* is a symbolic q^s-tree $\mathcal{T}^s_{q^s} = (V, E, r, \ell_V, \ell_E)$ which satisfies the following conditions:

(S1) For every $v \in V$ and every symbolic ε-transition from $\ell_V(v)$ with label $[i, f]$, there exists v' with $(v, v') \in E$ such that $\ell_E(v, v') = [i, f]$.

(S2) Whenever $(v, v'), (v, v'') \in E$ with $v' \neq v''$, $\ell_E(v, v') = [j', f']$, and $\ell_E(v, v'') =$

$[j'', f'']$, then $(j', f') \neq (j'', f'')$.

(S3) Whenever $(v, v'), (v, v'') \in E$ with $v' \neq v''$, $\ell_E(v, v') = [j', f', R', \mathsf{sc}]$, and $\ell_E(v, v') = [j'', f'', R'', \mathsf{sc}]$, then $(j', f', R') \neq (j'', f'', R'')$.

(S4) Whenever $(v, v'), (v, v'') \in E$ with $v' \neq v''$, $\ell_E(v, v') = [j', f', I]$, and $\ell_E(v, v'') = [j'', f'', I]$, then $j' = j''$ and $f' \neq f''$.

(S1) corresponds to (T3) and says that all symbolic ε-transitions that can be taken are present in a symbolic strategy tree. (S2) and (S3) make sure symbolic strategy trees cannot be too large: (S2) prevents superfluous symbolic ε-transitions to be present, while (S3) prevents superfluous secure channel transitions to be present. (S4) makes sure the intruder may only send one message to one principal when following his strategy.

There are certain properties required of strategy trees, for instance (T4) and (T5), which are not reflected in the definition of symbolic strategy trees. These properties are taken care of by the test in (A4) of SolveStrategy, which will be explained in Section 6.3.

The symbolic tree $\mathcal{T}_{ex}$ depicted in Figure 5 is in fact a symbolic q_0^s-strategy tree where $q_0^s = (\{\Pi_1, \Pi_2\}, \mathcal{I}_0, \emptyset, \emptyset)$ is the symbolic initial state of P_{ex}.

Given a protocol P, we call $\mathbf{B} = (\pi^s, \mathcal{T}_1^s, \ldots, \mathcal{T}_l^s)$ a *symbolic branching structure of* P if the following conditions are satisfied.

(1) π^s is a symbolic path of P.

(2) Let v be the leaf of π^s and $q^s = \ell_V(v)$. For each $i \in \{1, \ldots, l\}$, $\mathcal{T}_i^s$ is a symbolic q^s-strategy tree with root v.

(3) For $i \in \{1, \ldots, l\}$, let V_i be the set of vertices of $\mathcal{T}_i^s$. For $i, j \in \{1, \ldots, l\}$ with $i \neq j$, $V_i \cap V_j = \{v\}$.
Observe that, by our naming convention, this implies $\mathcal{X}_{\mathcal{T}_i^s} \cap \mathcal{X}_{\mathcal{T}_j^s} = \mathcal{X}_{\pi^s}$.

Obviously, there is a non-deterministic exponential time algorithm which given P can guess all possible symbolic branching structures (up to renaming of variables).

Consider our running example and the strategy property $((C_{ex}, C'_{ex})) = ((\{c_2\}, \{c_1\}))$. Then $\mathcal{T}_{ex}$ (Figure 5) can be viewed as a symbolic branching structure $\mathbf{B}_{ex}$ of P_{ex} when the path π^s is considered empty (and $l = 1$).

6.2 Constructing and Solving the Induced Constraint System

In this subsection, we explain how the constraint system $\mathbf{C_B}$ is derived from the symbolic branching structure $\mathbf{B} = (\pi^s, \mathcal{T}_1^s, \ldots, \mathcal{T}_l^s)$ computed in (A1) of SolveStrategy and the given strategy property $\mathcal{C}$, that is, we give the details for (A2).

The constraint system $\mathbf{C_B}$ can be shown to be valid, and hence, by Fact 5.1, a constraint solver can be used to solve it. In other words, (A3) is well-defined.

6.2.1 *Illustrating the Construction by an Example.* Before we get to technical details, we give some informal explanations and illustrate the construction with our running example, see Figure 3.

We first explain informally 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

$$\begin{array}{rrcl}
0. & \{a\}^{\mathsf{s}}_{k} & : & \mathcal{I}_0 \\
1. & \{\langle x_{h_3}, b\rangle\}^{\mathsf{s}}_{k_1} & : & \mathcal{I}_1, \{\langle a, b\rangle\}^{\mathsf{s}}_{k_1} \\
2. & \{\langle x_{h_7}, b\rangle\}^{\mathsf{s}}_{k_2} & : & \mathcal{I}_2, \{\langle b, b\rangle\}^{\mathsf{s}}_{k_2} \\
3. & \{y_{h_4}\}^{\mathsf{s}}_{k} & : & \mathcal{I}_1, \{\langle a, b\rangle\}^{\mathsf{s}}_{k_1}, x_{h_3} \\
4. & \{y_{h_8}\}^{\mathsf{s}}_{k} & : & \mathcal{I}_2, \{\langle b, b\rangle\}^{\mathsf{s}}_{k_2}, x_{h_7} \\
5. & y_{h_4} & : & \mathcal{I}_1, \{\langle a, b\rangle\}^{\mathsf{s}}_{k_1}, x_{h_3}, y_{h_4} \\
6. & x_{h_7} & : & \mathcal{I}_2, \{\langle b, b\rangle\}^{\mathsf{s}}_{k_2}, x_{h_7}, y_{h_8} \\
7. & y_{h_8} & : & \mathcal{I}_2, \{\langle b, b\rangle\}^{\mathsf{s}}_{k_2}, x_{h_7}, y_{h_8} \\
8. & c_2 & : & \mathcal{I}_1, \{\langle a, b\rangle\}^{\mathsf{s}}_{k_1}, x_{h_3}, y_{h_4}, c_2 \\
9. & c_2 & : & \mathcal{I}_2, \{\langle b, b\rangle\}^{\mathsf{s}}_{k_2}, x_{h_7}, y_{h_8}, c_2 \\
10. & c_2 & : & \mathcal{I}_2, \{\langle b, b\rangle\}^{\mathsf{s}}_{k_2}, x_{h_7}, y_{h_8}, c_2
\end{array}$$

Fig. 6. The valid constraint system $\mathbf{C}_{ex}$ derived from the symbolic strategy tree $\mathbf{B}_{ex}$ (Figure 5). Constants $k_1, k_2 \in \mathcal{A}$ are new atoms and we write $t_1, \ldots, t_n$ instead of $\{t_1, \ldots, t_n\}$.

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 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 $\{y\}^{\mathsf{s}}_{x} \Rightarrow x$ 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 $\mathbf{C}_{ex}$ (Figure 6) 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 5). The constraint system $\mathbf{C}_{ex}$ is obtained from $\mathbf{B}_{ex}$ as follows: First we add a dummy constraint in order to ensure the validity of the constructed constraint system. Then, 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 $\{\langle a, b\rangle\}^{\mathsf{s}}_{k_1}$ and $\{\langle b, b\rangle\}^{\mathsf{s}}_{k_2}$ 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 satisfy the monotonicity property of constraint systems. As explained above, since we use *new* keys and due to the feasibility condition on principals, this does not cause problems. The constraints 8.–10. 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.

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 5.1).

6.2.2 *Formal Definition.* We now provide a formal construction of the constraint system $\mathbf{C}_\mathbf{B}$ from the symbolic branching structure $\mathbf{B} = (\pi^s, \mathcal{T}_1^s, \ldots, \mathcal{T}_l^s)$ and the given strategy property $\mathcal{C}$.

The construction consists of three steps: In the first two steps, $\mathbf{B}$ is transformed, and in the last step the resulting branching structure is turned into the constraint system $\mathbf{C}_\mathbf{B}$. 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 communication with the secure channel as explained above.

Encoding the derivation of secure channel terms. An intruder does not have secure channel terms in its initial knowledge and he does not receive secure channel terms during the run of a protocol. Hence, to construct a term $\mathsf{sc}(n, n', m)$ he has to derive n and m (not necessarily n'). Since 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 new addresses. Thus, the only addresses the intruder knows are those in his initial knowledge. Therefore, the intruder can construct $\mathsf{sc}(n, n', m)$ only if n belongs to his initial knowledge and he can derive m. This motivates the following transformation:

(A2.1) For every edge e of $\mathbf{B}$ with which a symbolic intruder transition $q \xrightarrow{\tau} q'$ is associated and where the LHS of the associated the principal rule is of the form $\mathsf{sc}(n, n', R)$, do the following:
If n belongs to the initial intruder knowledge, then replace the LHS of the associated principal rule simply by R, else stop and output no (for all of SolveStrategy).

We refer to the symbolic branching structure obtained by the transformation just described as $\hat{\mathbf{B}}$. Strictly speaking, the resulting structure is not a symbolic branching structure anymore as it does not have the required form. By abuse of terminology, we still call it symbolic branching structure.

Encoding secure channel communication. As already explained above, we eliminate the secure channel component in symbolic states and instead let the intruder handle all communication.

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.

(A2.2) For every vertex v of $\hat{\mathbf{B}}$ and every successor v' of v in $\hat{\mathbf{B}}$ do the following (we traverse 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 $\{R\}^{\mathsf{s}}_{k_{v'}}$ into the intruder knowledge at v' and all successors of v' in $\hat{\mathbf{B}}$. Also, do the following for every edge (u, u') which is reachable from v' over edges not labelled $[\cdot, \cdot, \mathsf{sc}(n, n', R), \mathsf{sc}]$: Replace the label of (u, u') by $\{R\}^{\mathsf{s}}_{k_{v'}}$ and the LHS of the principal rule associated with (u, u') by $\{R'\}^{\mathsf{s}}_{k_{v'}}$ where R' is such that $\mathsf{sc}(n, n', R')$ is the LHS of the principal associated with (u, u').

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

We call $\{R\}^{\mathsf{s}}_{k_{v'}}$ 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 a key was generated).

Deriving the constraint system. From $\overline{\mathbf{B}}$ we now derive the constraint system $\mathbf{C}_{\mathbf{B}}$:

(A2.3) Let $\mathbf{C}_{\mathbf{B}}$ be the constraint system consisting a constraint $m : \mathcal{I}$ for some $m \in \mathcal{I}$.

(A2.3.a) 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). For every vertex v of $\overline{\mathbf{B}}$ and every successor v' of v do the following:
If the transition corresponding to the edge (v, v') is a symbolic secure channel or intruder transition, $\mathcal{I}$ is the intruder knowledge in the state associated with v, and R is the LHS of the principal rule associated with the transition, then add $R\colon \mathcal{I}$ to $\mathbf{C}_{\mathbf{B}}$ at the end.
We call constraints of this kind *intruder constraints of* $\boldsymbol{C_B}$.

(A2.3.b) For every $i \in \{1, \ldots l\}$, leaf v of $\mathcal{T}_i^s$, and $a \in C_i$ do the following:
If $\mathcal{I}$ is the intruder's knowledge in the state associated with v, then add $a\colon \mathcal{I}$ to $\mathbf{C}_{\mathbf{B}}$ at the end.
We call constraints of this this kind *strategy property constraints of* $\boldsymbol{C_B}$.

This completes the description of (A2) of SolveStrategy.

To be able to apply Fact 5.1, we need the following lemma. The proof of this lemma is straightforward and therefore omitted: To show monotonicity one has to make use of our naming convention for variables in different branches of $\overline{\mathbf{B}}$ and the fact that secure channel encoding messages are not removed anymore from the intruder's knowledge.

Lemma 6.9. *$\boldsymbol{C_B}$ is a valid constraint system.*

As a consequence of this lemma and Fact 5.1, we can employ the constraint solver to solve $\mathbf{C}_{\mathbf{B}}$.

6.3 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 (A1) of SolveStrategy and let $(\mathbf{C}', \tau)$ be the output returned by the constraint solver when applied to $\mathbf{C}_{\mathbf{B}}$ in (A2). Let ν be the complete solution associated with $(\mathbf{C}', \tau)$ in (A3), see Section 5. 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 $\mathbf{B}$.

In (A4), we want to check whether 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 $(P, \mathcal{C})$. 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 some state in $\mathcal{G}_P$, say q, and ii) $\nu(\mathcal{T}_i^s)$ corresponds to a q-strategy tree for (C_i, C'_i) for every $i \in \{1, \ldots, l\}$.

Since ν is a complete solution of $\mathbf{C_B}$, 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, (T1)–(T3), do not need to be checked. Moreover, we know that $\nu(\mathcal{T}_i^s)$ satisfies $(C_i, \emptyset)$. Hence, SolveStrategy only needs to make sure that $\nu(\mathcal{T}_i^s)$ fulfills $(\emptyset, C'_i)$ and that (T4) and (T5) are satisfied for every tree $\nu(\mathcal{T}_i^s)$.

This leads to the following tests:

(A4) For every $i \in \{1, \ldots, l\}$ do:

- (A4.1) *Check the strategy property:* For every leaf v of $\mathcal{T}_i^s$ check that $C'_i \cap d(\mathcal{I}\nu) = \emptyset$ where $\mathcal{I}$ is the intruder knowledge in the symbolic state associated with v.
- (A4.2) For every vertex v in $\mathcal{T}_i^s$ perform the following tests. Let $(\{\Pi'_h\}_h, \mathcal{I}, \mathcal{S}, \mathcal{X})$ be the symbolic state associated with v in $\mathcal{T}_i^s$ and for every $h \in \{1, \ldots, n\}$ let $\Pi'_h = (V'_h, E'_h, r'_h, \ell'_h)$.
  - (A4.2.a) *Check (T4):* For all $m \in \mathcal{S}\nu$, $h \in \{1, \ldots, n\}$, $n'_h \in V'_h$, and R such that
    - A. $(r'_h, n'_h) \in E'_h$,
    - B. R is the LHS of $\ell'_h(r'_h, n'_h)$, and
    - C. $R\nu$ matches m,

    check that there exists a successor v' of v in $\mathcal{T}_i^s$ such that (v, v') is labelled $[h, n'_h, R', \mathsf{sc}]$ for some R' with $R'\nu = m$.
  - (A4.2.b) *Check (T5):* For every successor v' of v in $\mathcal{T}_i^s$, $h \in \{1, \ldots, n\}$, $n'_h, n''_h \in V'_h$ with $n'_h \neq n''_h$, R, and R' such that
    - A. $[h, n'_h, I]$ is the label of (v, v'),
    - B. R is the LHS of $\ell'_h(r'_h, n'_h)$,
    - C. $(r'_h, n''_h) \in E'_h$,
    - D. R' is the LHS of $\ell'_h(r'_h, n''_h)$, and
    - E. $R'\nu$ matches with $\hat{R}\nu$ where $\hat{R}$ is obtained from R by replacing each non-indexed variable x in R by $x_{v'}$,

    check that there exists a successor v'' of v in $\mathcal{T}_i^s$ such that (v, v'') is labelled $[h, n''_h, I]$.

It is clear that the tests in (A4.2) can easily be performed given ν and $\mathcal{T}_i^s$. In [Chevalier et al. 2003], it was shown that the 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, (A4.1) 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 $\mathbf{B}$. In this case, SolveStrategy outputs yes.

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, (A4.2.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 pairs $(\mathbf{C}', \tau)$ in the sound and complete solution set for $\mathbf{C}_{\mathbf{B}}$.

7. PROOF OF THE MAIN THEOREM

In this section, we prove Theorem 6.1. Obviously, SolveStrategy always terminates; we only need to prove soundness and completeness. This is done in Section 7.2 and 7.3, respectively. We start with some additional notation that will be used in the proof.

For all of this section, we fix a protocol P and a strategy property $\mathcal{C}$ as usual.

7.1 Further Terminology

Concrete Branching Structures and Naming Conventions. Let $q \in \mathcal{G}_P$. A *q-tree* is a q-strategy tree where, however, only (T1) and (T2) need to be satisfied (see Definition 4.1). 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 has at most one successor. A *concrete branching structure* for P is defined analogously to *symbolic branching structure* only that now the symbolic path is a concrete path π and the symbolic strategy trees $\mathcal{T}_i$ are concrete strategy trees for every $i \in \{1, \ldots, l\}$. Such a branching structure satisfies $\mathcal{C}$ if $\mathcal{T}_i$ satisfies (C_i, C'_i) for every $i \in \{1, \ldots, l\}$.

Remark 7.1. $(P, \mathcal{C}) \in$ STRATEGY iff there exists a concrete branching structure for P which satisfies $\mathcal{C}$.

Concrete branching structures for P will be denoted $\mathbf{B} = (\pi, \mathcal{T}_1, \ldots, \mathcal{T}_l)$ 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 $i \in \{1, \ldots, l\}$.

Let $\mathbf{B}$ be a concrete branching structure for P. Note that $\mathbf{B}$ can be viewed as a q_0-tree $(V, E, r, \ell_V, \ell_E)$ where $V = V^\pi \cup V^1 \cup \cdots \cup V^l$, $E = E^\pi \cup E^1 \cup \cdots \cup E^l$, $r = r^\pi$, 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 express 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 viewed as trees 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'). When i is the principal associated with this transition, we call i the principal associated with (v, v'), respectively. Similarly, we speak of the principal rule and the principal vertex associated with (v, v').

Our notation and naming conventions for symbolic branching structures will be analogous; we will, however, always add an "s" as a superscript. That is, a symbolic branching structure will be denoted by $\mathbf{B}^s = (\pi^s, \mathcal{T}_1^s, \ldots, \mathcal{T}_l^s)$, we will write $\pi^s = (V^{s,\pi}, E^{s,\pi}, r^{s,\pi}, \ell_V^{s,\pi}, \ell_E^{s,\pi})$ and $\Pi_j^{s,v} = (V_j^{s,v}, E_j^{s,v}, r_j^{s,v}, \ell_j^{s,v})$.

Recall that for a symbolic branching structure $\mathbf{B}^s$ we denote the branching structure constructed from $\mathbf{B}^s$ in (A2) of SolveStrategy by $\overline{\mathbf{B}}^s$. We refer to the components of $\overline{\mathbf{B}}^s$ by appropriate decorations of the components of $\mathbf{B}^s$, i.e., $\overline{\ell}_E^{s,\pi}$, $\overline{V}_j^{s,v}$ and so on.

Substitutions of Symbolic States and Trees. Let $q^s = ((\Pi_1, \ldots, \Pi_n), \mathcal{I}, \mathcal{S}, \mathcal{X})$ be a symbolic state reachable from the initial symbolic state of P. Recall from Section 6.1 that the variables in $\mathcal{X}$ are indexed with vertices, that is, they are of the form x_v. Consider the mapping $\delta\colon x_v \mapsto x$ which drops the respective index. Since q^s is reachable from the initial symbolic state of P, the function δ is one-to-one.

Let ν be a substitution of the variables in $\mathcal{X}$. We denote by $\nu(q^s)$ the (concrete) state obtained from q^s by substituting the variables in q^s according to ν. More precisely, let $\mathcal{X}'$ be the image of δ. Then $\delta\colon \mathcal{X} \to \mathcal{X}'$ is a bijection. We define $\nu(q^s)$ to be 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 a substitution with domain $\mathcal{X}'$ and $\nu'|_{\mathcal{X}'}(x) = \nu(\delta^{-1}(x))$ for every $x \in \mathcal{X}'$.

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 instantiation obtained from $\mathcal{T}$ by substituting the variables in $\mathcal{T}$ according to ν. More precisely, we define $\nu(\mathcal{T})$ in such a way that it has the form of a (concrete) $\nu(q^s)$-tree. For this, we need to adjust the format of the labeling of edges and of course replace the symbolic states by concrete ones. However, we note that $\nu(\mathcal{T})$ is not necessarily a $\nu(q^s)$-tree in the sense defined above since $\nu(q^s)$ may not be a state of $\mathcal{G}_P$ and the transitions associated to the edges of this tree may not be transitions of $\mathcal{G}_P$. Formally, $\nu(\mathcal{T})$ is defined to be the 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 $(v, v') \in E$ 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. Note that $\nu(\mathcal{T})$ is in fact a concrete $\nu(q^s)$-tree.

Solutions of Symbolic Branching Structures. For a symbolic branching structure $\mathbf{B}^s = (\pi^s, \mathcal{T}_1^s, \ldots, \mathcal{T}_l^s)$ and a substitution ν, let $\nu(\mathbf{B}^s) = (\nu(\pi^s), \nu(\mathcal{T}_1^s), \ldots, \nu(\mathcal{T}_l^s))$.

We say that ν *solves* $\boldsymbol{B}^s$ *(with respect to 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 $R_i\sigma \in d(\mathcal{I} \cup \{S_j\sigma \mid j < i\})$ for every i. We call σ a *solution of* π' *w.r.t.* $\mathcal{I}$ if σ is a solution of $\rho_{\pi'}$ w.r.t. $\mathcal{I}$.

7.2 Soundness of SolveStrategy

In this section, we show soundness of SolveStrategy. The main problem is that when translating the symbolic branching structure that was guessed in (A1) of SolveStrategy into a constraint system we model the communication via secure channels by introducing what we call secure channel encoding terms. These encoding terms are written into the knowledge of the intruder and the idea is that the intruder delivers these terms instead of the secure channel terms that would have been delivered by the secure channel in the model. The problem is that the intruder might use these messages also in other contexts. We need to show that one can eliminate these secure channel encoding messages from a solution of the constructed constraint system.

We start with some lemmas on substitution, especially in the context of encryption.

7.2.1 *Preliminaries.* For terms t, t', t'', we denote by $t[t' \mapsto t'']$ the term obtained from t by simultaneously replacing every occurrence of t' by t''. This definition is extended to sets of terms and substitutions in the obvious way. We write $E[t' \mapsto t'']$ and $\sigma[t' \mapsto t'']$, respectively.

The following lemma can easily be shown by induction on the length of derivations using the same techniques as for example in [Rusinowitch and Turuani 2003].

LEMMA 7.2. *Let E be a set of messages and m_1, m_2, m_3, m_4 messages with $m_1 = \{m_2\}^{\mathsf{s}}_{m_3}$. If $m_1 \in E$ and $m_3 \notin d(E)$, then*

$$d(E)[m_1 \mapsto m_4] \subseteq d(E[m_1 \mapsto m_4]) \ . \tag{3}$$

Terms can be viewed as finite vertex-labeled ordered trees where the vertices are labeled with function symbols and the number of successors of a vertex is exactly the arity of the function symbol the vertex is labelled with. By convention, if a vertex is labeled by the encryption symbol $\{\cdot\}^{\mathsf{s}}_{\cdot}$, then the right successor corresponds to the key. Similarly, for the $\{\cdot\}^{\mathsf{a}}_{\cdot}$ and $\mathsf{sig}_{\cdot}(\cdot)$ symbols. 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* $\{t'\}^{\mathsf{s}}_t$ *in the term* t'' if for all $v \in t''$ with $t''{\downarrow}v = t$ we have that $t''{\downarrow}v' = \{t'\}^{\mathsf{s}}_t$ where v' is the predecessor of v in t''. For a set of terms E we say that t *only occurs in the context of* $\{t'\}^{\mathsf{s}}_t$ *in* E if t only occurs in the context of $\{t'\}^{\mathsf{s}}_t$ in all terms $t'' \in E$. Let σ be a substitution. We say that t *only occurs in the context of* $\{t'\}^{\mathsf{s}}_t$ *in* σ if t only occurs in the context of $\{t'\}^{\mathsf{s}}_t$ in $\sigma(x)$ for all $x \in \mathsf{dom}(\sigma)$.

LEMMA 7.3. *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 $\{m_1\}^{\mathsf{s}}_k$ in σ. Then*

$$(R\sigma)[\{m_1\}^{\mathsf{s}}_k \mapsto m_2] = R(\sigma[\{m_1\}^{\mathsf{s}}_k \mapsto m_2]) \ . \tag{4}$$

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

The following lemma can be proved by a simple induction.

LEMMA 7.4. *Let E be a set of messages and k, m be messages. If k only occurs in the context of $\{m\}^{\mathsf{s}}_k$ in E, then k only occurs in the context of $\{m\}^{\mathsf{s}}_k$ in $d(E)$.*

A vertex v of a term is a *key node* if it is a descendant of the right successor (inclusive) of a vertex labeled with $\{\cdot\}^{\mathsf{s}}_{\cdot}$, $\{\cdot\}^{\mathsf{a}}_{\cdot}$, or $\mathsf{sig}_{\cdot}(\cdot)$. A term t *occurs only in a key position* in term t' if every vertex v of t' such that the term corresponding to v is t is a key node. 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 $\{\langle a,b\rangle\}^{\mathsf{s}}_{\langle c,x\rangle}$ but not in the term $\{\langle x,b\rangle\}^{\mathsf{s}}_{c}$.

From the above lemmas, we obtain:

Lemma 7.5. *Let $k_1,\ldots,k_n \in \mathcal{A}$ be pairwise distinct atoms, let $t_1,\ldots,t_n$ be terms, and let $\mathcal{I}_1$ and $\mathcal{I}_2$ be sets of terms such that $\{k_1,\ldots,k_n\} \cap Sub(\mathcal{I}_1) = \emptyset$, $\mathcal{I}_2 = \{\{t_1\}^{\mathsf{s}}_{k_1},\ldots,\{t_n\}^{\mathsf{s}}_{k_n}\}$, and $\{k_1,\ldots,k_n\} \cap Sub(t_i) = \emptyset$ for $i \in \{1,\ldots,n\}$. 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 $R\sigma \in d(\mathcal{I}_1\sigma)$.

(2) For every i, $\{R\}^{\mathsf{s}}_{k_i}\sigma \in d(\mathcal{I}_1\sigma \cup \mathcal{I}_2\sigma)$ implies $R\sigma = t_i\sigma$.

Proof. To show (1), we first observe that $k_i \notin d(\mathcal{I}_1\sigma \cup \mathcal{I}_2\sigma)$. Now, iteratively applying Lemma 7.2 for every i with $m_1 = \{t_i\}^{\mathsf{s}}_{k_i}\sigma$ and some intruder atom m_4 allows us to eliminate the messages in $\mathcal{I}_2\sigma$. (2) easily follows with Lemma 7.4. □

7.2.2 *Proof of Soundness.* Assume SolveStrategy outputs yes. We have to show $(P,\mathcal{C}) \in$ Strategy.

Let $\mathbf{B}^s = (\pi^s, \mathcal{T}^s_1,\ldots,\mathcal{T}^s_l)$ be the symbolic branching structure guessed in (A1) of SolveStrategy. Let $\mathbf{C} = \mathbf{C}_{\mathbf{B}^s}$ be the corresponding constraint system constructed in (A2) and let $\mathbf{C}'$ be the simple constraint system returned by the constraint solver in (A3) such that the complete solution ν associated with $\mathbf{C}'$ passes the tests in (A4) of SolveStrategy.

We would like to show that ν solves $\mathbf{B}^s$ with respect to $\mathcal{C}$. Recall that this means that $\nu(\mathbf{B}^s)$ is a concrete branching structure which satisfies $\mathcal{C}$. The main problem is that ν might contain secure channel keys (see Section 6.2.2). With these keys, ν cannot solve $\mathbf{B}^s$ since they do not occur in $\mathbf{B}^s$. We will therefore iteratively eliminate these keys from ν. This will be done in such a way that before and after every step the current substitution satisfies certain conditions with respect to $\overline{\mathbf{B}}^s$ and $\mathbf{C}$ (recall the definition of $\overline{\mathbf{B}}^s$ and $\mathbf{C}$ from Section 6.2.2). At the end, we will have a ground substitution μ for which we then show that it solves $\mathbf{B}^s$ with respect to $\mathcal{C}$.

For a given substitution σ, the conditions are:

(P1) For every root path π of $\overline{\mathbf{B}}^s$ the substitution σ is a solution for π.

(P2) The substitution σ solves the strategy property constraints of $\mathbf{C}$.

(P3) The substitution σ passes the tests in (A4).

We first prove:

Lemma 7.6. *Let σ be a substitution satisfying (P1)–(P3) and let $k \in K_{\mathsf{sc}}$ be a secure channel key and $\{t\}^{\mathsf{s}}_{k}$ be the corresponding encoding term.*

Then k only occurs in the context of $\{t\sigma\}^{\mathsf{s}}_{k}$ 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 $\{t\sigma\}^{\mathsf{s}}_k$ 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 with $z \notin \mathcal{X}_{v'}$ and every $z' \in \mathcal{X}_{v'}$ it holds that k only occurs in the context of $\{t\sigma\}^{\mathsf{s}}_k$ 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 $\rho = \rho^{\overline{\mathbf{B}}}_{\pi'} = \rho_1, \ldots, \rho_{n-1}$ be the pr-sequence associated with π' where $\rho_i = R_i \Rightarrow S_i$. Since σ satisfies condition (P1), 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 $\{t\sigma\}^{\mathsf{s}}_k$ 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 $\{t\sigma\}^{\mathsf{s}}_k$ for every $z' \in \mathcal{V}(\overline{\mathcal{I}}^{s,v_{n-1}})$. Hence, k only occurs in the context of $\{t\sigma\}^{\mathsf{s}}_k$ in $\overline{\mathcal{I}}^{s,v_{n-1}}\sigma$. Lemma 7.4 thus yields:

(*) k only occurs in the context of $\{t\sigma\}^{\mathsf{s}}_k$ 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 $\{t\sigma\}^{\mathsf{s}}_k$ in $\sigma(z)$. From this it follows that there exists a subterm of R_n of the form $\{t\}^{\mathsf{s}}_z$ 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 7.6, we obtain:

Lemma 7.7. *Let σ be a substitution satisfying (P1)–(P3) and let $k \in K_{\mathsf{sc}}$ be a secure channel key and $\{t\}^{\mathsf{s}}_k$ be the corresponding encoding term.*

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'. In addition, let $\mathcal{I}_{v_i} = \overline{\mathcal{I}}^{s,v_i}$ be as in the proof of Lemma 7.6.

Then k only occurs in the context of $\{t\sigma\}^{\mathsf{s}}_k$ 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 $\{t\sigma\}^{\mathsf{s}}_k$ in $\mathcal{I}_{v_i}$ and R_i for every i. From this together with Lemma 7.6, the claim follows immediately. □

We can now state:

Proposition 7.8. *Let σ be a substitution satisfying (P1)–(P3) and let $k \in K_{\mathsf{sc}}$ be a secure channel key and $\{t\}^{\mathsf{s}}_k$ be the corresponding encoding term.*

Let $a \in \mathcal{A}_I$ be a new intruder atom and define $\sigma' = \sigma[m \mapsto a]$ where $m = \{t\sigma\}^{\mathsf{s}}_k$. Then σ' satisfies (P1)–(P3).

Proof. First note that due to Lemma 7.6 atom k does not occur in σ' anymore. We have to show that σ' satisfies (P1)–(P3).

(P1) Let $\pi' = v_1, \ldots, v_n$, ρ, ρ_i, R_i, S_i, and $\mathcal{I}_{v_i}$ be given as in Lemma 7.7. By assumption the substitution σ satisfies (P1), 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 proceed with a case distinction.

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 $\{t\}^{\mathsf{s}}_k \in \mathcal{I}_{v_{i-1}}$, and hence, $m \in \mathcal{I}_{v_{i-1}}\sigma$. By Lemma 7.7, we know:

(*) k only occurs in the context of $\{t\sigma\}^{\mathsf{s}}_k$ 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 $\{t'\}^{\mathsf{s}}_k$ for some term t'. It follows that $R_i\sigma = m$. Obviously, we have that $m = \{t\}^{\mathsf{s}}_k\sigma = \{t\}^{\mathsf{s}}_k\sigma'$. Thus, $m \in \mathcal{I}_{v_{i-1}}\sigma'$, and therefore, $R_i\sigma' \in d(\mathcal{I}_{v_{i-1}}\sigma')$.

If $k \notin \mathrm{Sub}(R_i)$, we can apply Lemma 7.3 and obtain that $(R_i\sigma)[m \mapsto a] = R_i\sigma'$. Moreover, using Lemma 7.4 and (*), we know that $k \notin d(\mathcal{I}_{v_{i-1}}\sigma)$. Now, we can apply Lemma 7.2 and obtain that $R_i\sigma' = (R_i\sigma)[m \mapsto a] \in d((\mathcal{I}_{v_{i-1}}\sigma)[m \mapsto a])$. For $S \in \mathcal{I}_{v_{i-1}}$ such that $k \notin S$, Lemma 7.3 implies $S\sigma' = (S\sigma)[m \mapsto a]$. By construction of $\overline{\mathbf{B}}^s$, if $k \in \mathrm{Sub}(S)$, then $S = \{t\}^{\mathsf{s}}_k$, and hence, $S\sigma = m$. Since $a \in \mathcal{A}_I \subseteq d(\mathcal{I}_{v_{i-1}}\sigma')$, we obtain that $d((\mathcal{I}_{v_{i-1}}\sigma)[m \mapsto a]) \subseteq d(\mathcal{I}_{v_{i-1}}\sigma')$. Consequently, $R_i\sigma' \in d(\mathcal{I}_{v_{i-1}}\sigma')$.

(P2) 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 $\mathcal{T}^s_i$, 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 $c \in d(\overline{\mathcal{I}}^{s,v_n}\sigma')$.

(P3) We have to show that σ' also passes (A4.1), (A4.2.b), and (A4.2.b), and we know that σ passes them. First, we show that σ' passes (A4.1). We know:

(**) $C'_i \cap d(\overline{\mathcal{I}}^{s,v}\sigma) = \emptyset$ for every leaf v of $\mathcal{T}^s_i$ 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 a new intruder atom, from $(\overline{\mathcal{I}}^{v,s}\sigma')[a \mapsto \{t\}^{\mathsf{s}}_k\sigma]$ we can still derive c. Together with the fact that $(\overline{\mathcal{I}}^{s,v}\sigma')[a \mapsto \{t\}^{\mathsf{s}}_k\sigma] = \overline{\mathcal{I}}^{s,v}\sigma$, this is a contradiction to (**).

To see that σ' passes (A4.2.a) and (A4.2.b), it suffices to observe that if condition C. and conditions C.–E. in (A4.2.a) and (A4.2.b), respectively, are satisfied for σ', then also for σ. Here we again use that a is a new intruder atom and that we can replace a again by $\{t\}^{\mathsf{s}}_k\sigma$. $\square$

We can now phrase what we were aiming for:

Proposition 7.9. *Let μ be the substitution obtained from ν by repeatedly applying the transformation described in Proposition 7.8 (from σ to σ') until no secure channel encoding terms are left.*
Then $\mu(\boldsymbol{B}^s)$ is a concrete branching structure for P which satisfies $\mathcal{C}$.

Note that since ν was returned by SolveStrategy, and hence, solves $\mathbf{C}$ and passes the tests in (A4), ν satisfies (P1)–(P3) and Proposition 7.8 can be applied repeatedly.

Note that by Remark 7.1 this proposition implies soundness of SolveStrategy.

Proof. 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:

(i) $\mu(\ell^s_V)(v_1)$ is the initial state of P,

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

(iii) $\mu(\ell^s_V)(v_i) \xrightarrow{\mu(\ell^s_E)(v_i,v_{i+1})} \mu(\ell^s_V)(v_{i+1}) \in \mathcal{G}_P$ for $i \in \{1,\ldots,h-1\}$.

For $h = 1$, we only need to show that $\mu(\ell^s_V)(v_1)$ 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 $\rho^s = \rho^s_1,\ldots,\rho^s_h$ with $\rho^s_i = R^s_i \Rightarrow S^s_i$ be the pr-sequence associated with π' in $\mathbf{B}^s$ and let $\overline{\rho}^s = \overline{\rho}^s_1,\ldots,\overline{\rho}^s_h$ with $\overline{\rho}^s_i = \overline{R}^s_i \Rightarrow \overline{S}^s_i$ be the pr-sequence associated with π' in $\overline{\mathbf{B}}^s$. Moreover, let $\mu(\mathcal{I}^{s,v_i})$ be the knowledge of the intruder at v_i in $\mu(\mathbf{B}^s)$. Note that $\mu(\mathcal{I}^{s,v_i}) \subseteq \mathcal{I}_0 \cup \{S^s_1\mu,\ldots,S^s_{i-1}\mu\}$ where $\mathcal{I}_0$ is the initial intruder knowledge in P. This inclusion could be strict since some of the S^s_j may be secure channel terms, and hence, are not written into the intruder's knowledge. Let $\mu(\overline{\mathcal{I}}^{s,v_i}) = \mathcal{I}_0 \cup \{\overline{S}^s_1\mu,\ldots,\overline{S}^s_{i-1}\mu\}$ 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.

We next show that the statements (i), (ii), and (iii) from above hold for π'. Obviously, statement (i) still holds. By the induction hypothesis, we know that $\mu(\ell^s_V)(v_i) \in \mathcal{G}_P$ for $i \in \{1,\ldots,h\}$ and $\mu(\ell^s_V)(v_i) \xrightarrow{\mu(\ell^s_E)(v_i,v_{i+1})} \mu(\ell^s_V)(v_{i+1}) \in \mathcal{G}_P$ for every $i \in \{1,\ldots,h-1\}$. We have to show that $\mu(\ell^s_V)(v_{h+1}) \in \mathcal{G}_P$ and $\mu(\ell^s_V)(v_h) \xrightarrow{\mu(\ell^s_E)(v_h,v_{h+1})} \mu(\ell^s_V)(v_{h+1}) \in \mathcal{G}_P$. 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 we have $R^s_h\mu \in d(\mu(\mathcal{I}^{s,v_h}))$. First assume that R^s_h is not a secure channel term. It follows that $R^s_h = \overline{R}^s_h$. Since μ satisfies condition (P1) from above, we know that $\overline{R}^s_h\mu \in d(\mu(\overline{\mathcal{I}}^{s,v_h}))$. 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^s_h, and the terms in $\mathcal{I}^{s,v_h}$ do not contain secure channel keys, we immediately obtain $R^s_h\mu \in d(\mu(\mathcal{I}^{s,v_h}))$ by Lemma 7.5. The argument in case R^s_h is a secure channel term is similar.

For symbolic secure channel transitions, assume that $\ell^s_E(v_h,v_{h+1})$ 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 $\mu(R^s_{v_h}) = \mu(\mathsf{sc}(n,n',R'))$ where $\mathcal{S}^{s,v_h}$ denotes the secure channel in the symbolic state $\ell^s_V(v_h)$. 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 $R^s_{v_h}$ is of the form $\mathsf{sc}(n,n',R)$, then, by construction of $\overline{\mathbf{B}}^s$, $\overline{R}^s_h$ is of the form $\{R\}^{\mathsf{s}}_k$ for some secure channel key k such that $\{R'\}^{\mathsf{s}}_k \in \overline{\mathcal{I}}^{s,v_h}$ and k does not occur in any other term in $\overline{\mathcal{I}}^{s,v_h}$. We know that $\overline{R}^s_h\mu \in d(\overline{\mathcal{I}}^{s,v_h}\mu)$. Since k does not occur in μ nor in R and R', with Lemma 7.5 we obtain that $R\mu = R'\mu$, and thus, $\mu(R^s_h) = \mu(\mathsf{sc}(n,n',R'))$. Thus, we have shown that the edges in $\mu(\mathbf{B}^s)$ correspond to transitions in $\mathcal{G}_P$.

The last step is to show that $\mu(\mathcal{T}^s_j)$ is a strategy tree for (C_j, C'_j). Properties (T1) and (T2) of Definition 4.1 are satisfied as we have just shown. (T3) follows directly from (S3). (T4) and (T5) follow directly from the fact that μ passes (A4.2.a) and (A4.2.b). $\square$

7.3 Completeness of SolveStrategy

In this section, we show completeness of SolveStrategy. To this end, let $(P,\mathcal{C})$ be an instance of Strategy and assume $(P,\mathcal{C}) \in$ Strategy. We have to show that there is a run of SolveStrategy in which yes is returned. From Remark 7.1, we conclude that there exists a branching structure $\mathbf{B}$ for P that satisfies $\mathcal{C}$.

Our proof proceeds in three steps:

(C1) 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 step (A1).

(C2) We consider the constraint system $\mathbf{C}$ that is constructed in (A2), provided $\mathbf{B}^s$ was guessed in (A1), 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 $\mathbf{C}'$.

(C3) We show that the substitution $\tau_{\mathbf{C}'}$ associated with $\mathbf{C}'$ passes the tests performed in the third step of SolveStrategy.
This means SolveStrategy outputs yes

The technical problem we have to overcome is that the above steps can only be carried out if $\mathbf{B}$ is what we call a minimal branching structure. So we also need to show that this is the case without loss of generality.

Roughly speaking, minimal branching structures satisfy two properties: First, they should not contain superfluous transitions. Second, the secure channel transitions are in some sense complete. We now define these structures formally.

7.3.1 *Preliminaries.* 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, (C1).

The symbolic branching structure $\mathbf{B}^s$ and the substitution τ 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), 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 what we call 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 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.

We now turn to the formal definition of the notions explained above. 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 such that $\mathcal{V}(S_i) \subseteq \mathsf{dom}(\sigma)$ for some $i \in \{1, \ldots, k\}$, and a message m, we say that the sc-sequence s' is an *(S, m, σ)-successor* of s if

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

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

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

(F1) $\mathcal{S}^v = \sigma^v(\{S_1, \ldots, S_k\})$ with $\ell_{\mathsf{sc}}(v) = (S_1, \ldots, S_k)$. (Note that $\{S_1, \ldots, S_k\}$ should be treated as a multiset.)

(F2) 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

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

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

(F3) If v' is a successor of v in $\mathbf{B}$ and $\ell_E(v, v')$ is of the form $[i, m, \mathsf{sc}]$ (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 (F3), 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 will be defined in such a way that they satisfy two properties: i) there are no superfluous transitions and ii) the secure channel transitions are in some sense complete. To motivate the latter condition, we need to sketch (C3): We show that $\tau_{\mathbf{C}'}$ passes the tests performed in (A4), in particular, (A4.2.a), which 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 4.1, 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 transitions already occur in $\mathbf{B}$. This is captured by the notion of sc-completeness defined next.

Recall the definition of V^j from the beginning of Section 7. 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

—the label of the transition associated with (v, v') is $[i, m, \mathsf{sc}]$ 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 $[i, m, \mathsf{sc}]$, and the principal vertex associated with (v, v'') is f.

We can now define minimal concrete branching structures.

Definition 7.10. A branching structure $\mathbf{B}$ for P is called *minimal* if it satisfies the following conditions:

(M1) $\mathbf{B}$ is sc-complete.

(M2) For all $i \in \{1, \ldots, l\}$ 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:
—The principal associated with (v, v') differs from the principal associated with (v, v''), or
—the principal vertex associated with (v, v') differs from the principal vertex associated with (v, v'').

(M3) For all $i \in \{1, \ldots, l\}$ and $(v, v'), (v, v'') \in E^i$ where $v' \neq v''$ where the transitions associated with (v, v') and (v, v'') are secure channel transitions:
—The principal associated with (v, v') differs from the principal associated with (v, v''), or
—the principal vertex associated with (v, v') differs from the principal vertex associated with (v, v''), or
—the sc-term removed in (v, v') differs from the sc-term removed in (v, v'').

(M4) For all $i \in \{1, \ldots, l\}$ and $(v, v'), (v, v'') \in E^i$: If $\ell_V(v, v') = [j, m, I]$ for some j and m, then $\ell_V(v, v'') = [j, m, I]$.

(M2) – (M4) say that there are no superfluous transitions in the strategy trees of a minimal branching structure. These conditions correspond to the conditions (S2) – (S4) for symbolic strategy trees. (M1) is the completeness condition explained above.

The following lemma is easy to show, simply by cutting down a given branching structure appropriately.

LEMMA 7.11. *If there exists a branching structure for P which satisfies $\mathcal{C}$, then there exists a minimal branching structure with this property.*

7.3.2 *Proof of Completeness.* We can now carry out (C1), (C2), and (C3), assuming that the given branching structure $\mathbf{B}$ is minimal.

Step (C1). 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 be the principal vertex associated with this transition. Let $\mathcal{X}' = \mathcal{V}(R) \setminus \mathsf{dom}(\sigma^v)$ where $\ell^v_j(r^v_j, f) = R \Rightarrow S$ is the principal rule associated with (v, v') in $\mathbf{B}$. (Note that $\mathcal{X}'$ is the set of new variables in R). 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 $x_{v'}$.

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

In the following, we denote by $\hat{R}$ and $\hat{S}$ the terms R and S where the variables occurring in R and S, respectively, are renamed by there corresponding indexed version from $\mathcal{X}_{v'}$.

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 set
  (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 $\hat{S} = \mathsf{sc}(\cdot, \cdot, R')$; and
  (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 $(\hat{S}, m, \sigma^v)$-successor 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 h, it is easy to see that for each root path $v_1, \ldots, v_h$ in $\mathbf{B}^s$ we have that $\tau(\ell_V^s(v_i)) = \ell_V(v_i)$ and $\tau(\ell_E^s(v_i, v_{i+1})) = \ell_E(v_i, v_{i+1})$ for every i. From this, it immediately follows that $\tau(\mathbf{B}^s) = \mathbf{B}$.

Step (C2). We have to show that τ is a solution of the constraint system $\mathbf{C}$ constructed from $\overline{\mathbf{B}}^s$. There are two types of constraints in $\mathbf{C}$: First, constraints that were introduced for an edge $(v, v') \in E$, the intruder constraints, and second, the strategy constraints.

Let $R\colon T$ 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) $\ell_E(v, v') = [j, m, \mathsf{sc}]$.

Case (a). We know that $\tau(R) = \sigma^{v'}(R) = m \in d(\mathcal{I}^v)$ because R is the left-hand side of the principal 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 $m \in d(\tau(T))$.

Case (b). We know that m is of the form $m = \mathsf{sc}(n, n', m')$ 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 $n \in \mathcal{I}^v$ 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, $m' \in d(\tau(T))$.

Case (c). The term R is a term of the form $\{R'\}^{\mathsf{s}}_{k_{w'}}$ 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 $\mathsf{sc}(n, n', R')$ and the right-hand side of the principal rule associated with (w, w') in $\mathbf{B}^s$ has the form $\mathsf{sc}(n, n', R'')$. We know that $\tau(\mathsf{sc}(n, n', R')) = m = \tau(\mathsf{sc}(n, n', R''))$. Hence, $\tau(\{R'\}^{\mathsf{s}}_{k_{w'}}) = \tau(\{R''\}^{\mathsf{s}}_{k_{w'}})$. Since $\{R''\}^{\mathsf{s}}_{k_{v''}} \in T$, it follows that $\tau(\{R'\}^{\mathsf{s}}_{k_{w'}}) \in d(\tau(T))$.

Since a strategy constraint is of the form $a : \overline{\mathcal{I}}^{s,v}$ 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.

Step (C3). We will first show that τ passes the tests that in (A4) of SolveStrategy and from this it will follow that $\tau_{\mathbf{C}'}$ passes these tests.

Test (A4.1). 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').

Test (A4.2). For every i and vertex $v \in V^i$, we have to show that the following conditions are satisfied.

Test (A4.2.a). For $m \in \mathcal{S}^{s,v}\tau$, h, f, R such that

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

B. R is the LHS of $\ell_h^{s,v}(r_h^{s,v}, f)$, and

C. $R\tau$ matches with m

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

Test (A.4.2.b). If there exists an edge $(v, v') \in E$ and $f' \neq f$ such that

A. the transition corresponding to the edge (v, v') is labeled with $[h, f, I]$,

B. R is the LHS of $\ell_h^{s,v}(r^{s,v}, f)$,

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

D. R' is the LHS of $\ell_h^{s,v}(r_h^{s,v}, f')$, and

E. $R'\tau$ matches with $\hat{R}\tau$ where $\hat{R}$ is obtained from R by replacing every non-indexed variable x in R 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 (A4.2.a), let $m \in \mathcal{S}^{s,v}\tau$, h, and f satisfy A.–C. 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 is sufficient.

To show (A4.2.b), let $(v, v') \in E$ and h, f, f', R, and R' satisfy A.– C. 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 tests in (A4) of SolveStrategy. Now we show that $\tau_{\mathbf{C}'}$ passes these tests as well.

Test (A4.1). Let v be a leaf of some $\mathcal{T}_i^s$ and suppose that there is an atom $a \in C_i'$ such that $a \in d(\mathcal{I}^{s,v}\tau_{\mathbf{C}'})$. Then it is easy to see that $a \in d(\mathcal{I}^{s,v}\tau)$, which is a contradiction to (C1).

Test (A4.2.a). Let $m \in \mathcal{S}^{s,v}\tau_{\mathbf{C}'}$, h, f such that

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

B. R is the LHS of $\ell_h^{s,v}(r_h^{s,v}, f)$, and

C. $R\tau_{\mathbf{C}'}$ 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.

Test (A4.2.b). Let $(v, v') \in E, h, f, R, f'$ and R' satisfy A.–C. 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 (A4.2.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.

8. CONCLUSION

In this paper we have shown that certain game-theoretic security properties, such as balance, of contract-signing and related protocols are decidable when there is no bound on the message size for a Dolev-Yao intruder and when there are only a finite number of sessions. This extends known results on the decidability of reachability problems for cryptographic protocols in a natural way. Also, our decision algorithm uses 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

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

Armando, A., Basin, D., Boichut, Y., Chevalier, Y., Compagna, L., Cuéllar, J., Drielsma, P., Héam, P.-C., Kouchnarenko, O., Mantovani, J., Mödersheim, S., von Oheimb, D., Rusinowitch, M., Santiago, J., Turuani, M., Viganò, L., and Vigneron, L. 2005. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In *Computer Aided Verification, 17th International Conference (CAV 2005)*, K. Etessami and S. Rajamani, Eds. Lecture Notes in Computer Science, vol. 3576. Springer-Verlag, 281–285.

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

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

Ben-Or, M., Goldreich, O., Micali, S., and Rivest, R. 1990. A fair protocol for signing contracts. *IEEE Transactions on Information Theory 36,* 1, 40–46.

Boreale, M. 2001. Symbolic trace analysis of cryptographic protocols. In *Automata, Languages and Programming, 28th International Colloquium (ICALP 2001)*, F. Orejas, P. Spirakis, and J. van Leeuwen, Eds. Lecture Notes in Computer Science, vol. 2076. Springer-Verlag, 667–681.

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

Chadha, R., Kremer, S., and Scedrov, A. 2004. Formal analysis of multi-party contract signing. In *17th IEEE Computer Security Foundations Workshop (CSFW-17)*, R. Focardi, Ed. IEEE Computer Society Press, 266–279.

Chadha, R., Mitchell, J., Scedrov, A., and Shmatikov, V. 2005. Contract signing, optimism and advantage. *Journal of Logic and Algebraic Programming (Special issue on Modeling and Verification of Cryptographic Protocols) 64,* 2, 189–218.

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

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

Corin, R. and Etalle, S. 2002. An Improved Constraint-Based System for the Verification of Security Protocols. In *Proceedings of the 9th International Symposium on Static Analysis (SAS 2002)*, M.V. Hermenegildo and G. Puebla, Eds. Number 2477 in Lecture Notes in Computer Science. Springer-Verlag, 326–341.

Comon, H. and Shmatikov, V. 2002. Is it possible to decide whether a cryptographic protocol is secure or not? *Journal of Telecommunications and Information Technology 4*.

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

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

Kähler, D. and Küsters, R. 2005. Constraint Solving for Contract-Signing Protocols. In *Proceedings of the 16th International Conference on Concurrency Theory (CONCUR 2005)*, M. Abadi and L. de Alfaro, Eds. Lecture Notes in Computer Science, vol. 3653. Springer, 233–247.

Kähler, D., Küsters, R., and Wilke, Th. 2005. Deciding Properties of Contract-Signing Protocols. In *Proceedings of the 22nd Symposium on Theoretical Aspects of Computer Science (STACS 2005)*, V. Diekert and B. Durand, Eds. Number 3404 in Lecture Notes in Computer Science. Springer-Verlag, 158–169.

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

Norman, G. and Shmatikov, V. 2006. Analysis of Probabilistic Contract Signing. *Journal of Computer Security 14,* 6, 561–589.

Meadows, C. 2003. Formal Methods for Cryptographic Protocol Analysis: Emerging Issues and Trends. *IEEE Journal on Selected Areas in Communication 21,* 1 (January), 44–54.

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

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

Shmatikov, V. and Mitchell, J. 2002. 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.

Zhou, J. and Gollmann, D. 1996. A fair non-repudiation protocol. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*. IEEE Computer Society Press, 55–61.

Received April 2007; accepted January 2009