Finitary Deduction Systems

Yannick Chevalier

Finitary Deduction Systems

Yannick Chevalier

May 6, 2011

Abstract

Cryptographic protocols are the cornerstone of security in distributed systems. The formal analysis of their properties is accordingly one of the focus points of the security community, and is usually split among two groups. In the first group, one focuses on trace-based security properties such as confidentiality and authentication, and provides decision procedures for the existence of attacks for an on-line attackers. In the second group, one focuses on equivalence properties such as privacy and guessing attacks, and provides decision procedures for the existence of attacks for an offline attacker. In all cases the attacker is modeled by a deduction system in which his possible actions are expressed.

We present in this paper a notion of finitary deduction systems that aims at relating both approaches. We prove that for such deduction systems, deciding equivalence properties for on-line attackers can be reduced to deciding reachability properties in the same setting.

1 Introduction

Context. Security protocols, *i.e.* protocols in which the messages are cryptographically secured, are a cornerstone of security in distributed applications. The need for optimizing resource utilization and their distributed nature make their design error prone, and formal methods have been applied successfully to detect errors in the past [29, 6]. But they are limited in expressiveness since in most cases authors either were focused on the resolution of reachability problems, or considered models in which the attacker could not interfere with the on-going communications among the honest agents. In contrast we consider in this paper the general case of equivalence properties *w.r.t.* an on-line attacker.

Formal models of cryptographic protocols usually present the reader with a dichotomy between the honest agents—translated into a constraint system [5, 30, 31] or a frame [3]—, and the attacker—modeled by a deduction system expressing its possible actions. In contrast we have introduced in [15] a notion of *symbolic derivation* that unifies the honest and dishonest agent models: the actions of all agents are represented by a sequence of deductions, nonce creation, and communication actions. The notion of equivalence considered in this paper is the one of symbolic derivations representing honest agents.

Intuition. First, a trivial remark: since one can construct deduction systems for which reachability is decidable but static equivalence is not, it is clear that generally speaking being able to decide reachability does not imply being able to decide symbolic equivalence. However, in most cases, one can model reachability as the satisfiability of a constraint system, and describe the decision procedure using constraint transformation rules. A *solved form* is defined as a constraint system in which the attacker just has to instantiate variables by any term he can construct. In practice, the proof of completeness of the procedure consists in assuming the existence of a sequence of deduction steps that satisfies the constraint system, and in proving that as long as one such sequence exists, either the constraint system is in solved form or there exists a transformation rule applicable on the constraint system. Then, an argument is given to prove that there is no infinite sequence of transformations. Using König's lemma, the finiteness (also to be proved) of the number of possible successors of each constraint system implies termination of the procedure.

Our motivation was that such procedures actually do much more than simply deciding reachability, as they end with a set of constraint systems in solved form that, as long as the completeness proof is along the lines given above, cover all possible attacks. Formalizing this argument is however not trivial, since

- not all instances of the variables occurring in a constraint system in solved form correspond to attacks; and
- when testing the equivalence of two protocols, we have to take into account the equality tests the attacker can perform to analyze the responses of the honest agents.

We have bypassed the first difficulty by imposing that the attacker instantiates the first-order variables in a constraint system in solved form with constants, and proved that replacing these constants by any possible construction yields another attacks. This replacement is formalized by on ordering on the attacks, the attacks corresponding to solved forms being the minimal ones. Finitary deduction systems are those for which the set of minimal attacks is always finite. The second difficulty is solved by first proving that it suffices to consider an attacker that performs at most one test, and then proving that this test can be guessed *before* the computation of solved forms. Finally and implementation-wise, we consider *effective* finitary deduction system, for which we assume that this finite set is computable.

Applications. The symbolic equivalence notion we consider in this paper has three straightforward applications, related respectively to on-line guessing attacks, to proving cryptographic properties in a symbolic setting, and to privacy. We have proved, in collaboration with M. Rusinowitch [19] that every protocol narration (for any deduction system) can be compiled into an active frame, which is a simplified form of symbolic derivations with a total ordering on states and no intermediate computations between communications.

Guessing attacks. Introduced by Schneier [34] under the name of *dictionary attacks*, they consist in guessing a secret piece of data, and then being able to check whether the guess is correct. They can be offline, in which case the attacker observes interactions between honest participants and has to decide whether the guessed piece of data has been employed, or on-line, in which case the intruder can interact with the honest participants.

Guessing attacks have been formalized thanks to the concept of indistinguishability (see *e.g.* [2]). We can say now that a protocol is vulnerable to undetectable on-line guessing attacks whenever *(i)* the honest agents cannot distinguish between a session with the right piece of data and one involving a wrong guess, whereas *(ii)* the intruder can distinguish the two executions. We model the first point by stating that the tests performed by the honest agents succeed in both cases, and the second point by saying that the two executions are not equivalent.

Cryptographic properties. A line of works initiated by [4] showed that computational proofs of indistinguishability ensuring the security of a protocol can be derived, under some natural hypothesis on cryptographic primitives, from symbolic equivalence proofs. This has opened the path to the automation of computational proofs. It was shown by [20] that *in presence of an active attacker* observational equivalence of the symbolic processes can be transferred to the computational level.

Privacy. Symbolic equivalence is a crucial notion for specifying security properties such as anonymity or secrecy of a ballot in vote protocols [22]. More generally, the analysis of privacy, *e.g.* client's identity in an anonymization protocol such as IDEMIX [32, 13], in communication protocols is inherently an equivalence problem. One has to prove that a protocol preserves the *strong secrecy* of an attribute, *i.e.* that an observer cannot distinguish the execution of a protocol transmitting this attribute's value, be it a vote or her identity, from one in which a random piece of data is exchanged.

Related works. We believe that Mathieu Baudet's modeling of attacks by instantiation of *second-order* variables [8] is the real breakthrough that enabled the formal analysis of the equivalence problem in the on-line attacker setting. Indeed, it was the first-time that the actions of the attacker were represented explicitly in solutions, instead of just keeping track (with a substitution on the first-order variables of the constraint system) of their interaction with the honest participants.

In collaboration with M. Rusinowitch [19] we have given another proof of Baudet's result in the setting of symbolic derivations. We believe that this setting is more complex but introduces a langage fit to prove decidability and complexity results. Also it possesses a symmetry between honest participants and the attacker that permits to greatly simplify otherwise redundant proofs. We consider in this paper a setting in which the actions of the honest agents are represented by one *Honest symbolic derivation* (HSD) and those of a *unique* intruder by one *Attacker Symbolic Derivation* (ASD). Symbolic derivations can

be seen as standing between symbolic traces [8] and the simple cryptographic processes of [21]: the sequence of messages is not totally ordered as it is the case in [8], but there is no branching but for termination on error nor any recursive process.

Few decidability results are available. In the article [26] Hüttel proves decidability for a fragment of the spi-calculus without recursion for framed bisimilarity. Since, the only original decidability result on the equivalence of symbolic traces[1] we are aware of is for the class of subterm deduction systems and was given by M. Baudet [8, 9]. We have recently given another proof of this result [18], on which this paper elaborates. Implementation-wise, an efficient procedure is presented in [14] in which one considers only the Dolev-Yao deduction system. In spite of the relevance of this problem, we are not aware of any extension of Baudet's decidability results to other classes of deduction systems.

In [35] the authors consider, as Hüttel [26], the same problem in the simpler case of the standard Dolev-Yao syntactic deduction system (with no equational theory). They employ the notion of solved form as introduced in [5], and more specifically that solved forms cover all possible attacks. The existence of such a finite set of solved forms corresponds exactly to our notion of finitary deduction system.

However, we note that their setting enforces a strict separation between the values of the first order variables and the observer process. This has in our opinion two negative side-effects. First, it is well-known that not all instances of the first-order substitutions constructed are instances of attacks. Second, given that the authors of [35] only keep track of the constraints that remain to be solved, the attacks themselves are not represented explicitly in the solution. Hence it is not possible to reason on all first-order instances of a solved form (since they are not all attacks) nor on the observer processes (since only their interaction with the processes under scrutiny is recorded). This is the reason why we believe that the symbolic derivation setting adopted in this paper, while more cumbersome at first, is better suited to reason on sets of solutions, and therefore on process equivalence.

Many works have been dedicated to proving correctness properties of cryptographic protocols using equivalences on process calculi. In particular *framed bisimilarity* has been introduced by Abadi and Gordon [3] for this purpose, for the spi-calculus. Another approach that circumvents the context quantification problem is presented in [12] where labeled transition systems are constrained by the knowledge the environment has of names and keys. This approach allows for more direct proofs of equivalence.

In [21] the authors show how to apply the result by Baudet on S-equivalence to derive a decision procedure for symbolic equivalence for subterm convergent theories for simple processes. Since [21] relies on the proof of Baudet's result, that is long and difficult [9], we believe that providing a simple criterion will be useful to derive other decidability results in process algebras.

[1] a restriction of symbolic equivalence in which the actions of all the honest agents are totally ordered.

To the best of our knowledge, the only tool (besides [14]) capable of verifying equivalence-based secrecy is the resolution-based algorithm of ProVerif [10] that has been extended for handling equivalences of processes that differ only in the choice of some terms in the context of the applied π-calculus [11]. This allows to add some equational theories for modeling properties of the underlying cryptographic primitives.

Example finitary deduction systems. We remark that the standard Dolev-Yao deduction system [24] is finitary, since for every attack one can guess a subsequence of deduction steps which is itself an attack [16]. In this regard, this work extends [35] to other deduction systems such as subterm deduction systems (the proof that from every attack one can guess a sequence of deductions bounded by the size of the input protocol is given *e.g.* in [28]). We leave to future work the extension to contracting saturated deduction systems, also defined in [28].

Organization of this paper. We reuse in this paper the notions and notations for terms, equational theories, deduction systems, and symbolic derivations introduced in earlier papers (sections 2–3). We give in Section 4 a few properties of symbolic derivations, and define finitary deduction systems accordingly. We present in Section 5 a sketch of the proof the symbolic equivalence is decidable for finitary deduction systems, and conclude in Section 6.

2 Formal setting

2.1 Term algebra

We consider a countable set of free constants C, a countable set of variables $\mathcal{X}$, and a signature $\mathcal{F}$ (*i.e.* a set of function symbols with arities). We denote by $\mathcal{T}(\mathcal{F})$ (resp. $\mathcal{T}(\mathcal{F},\mathcal{X})$) the set of terms over $\mathcal{F}\cup C$ (resp. $\mathcal{F}\cup C\cup\mathcal{X}$). The former is called the set of ground terms over $\mathcal{F}$, while the latter is simply called the set of terms over $\mathcal{F}$. Variables are denoted by x, y, terms are denoted by $s,t,u,v,\ldots$, and decorations thereof, respectively.

A *constant* is either a *free* constant in C or a function symbol of arity 0. Given a term t we denote by $\mathrm{Var}(t)$ the set of variables occurring in t and by $\mathrm{Const}(t)$ the set of constants occurring in t. We denote by $\mathrm{atoms}(t)$ the set $\mathrm{Var}(t)\cup\mathrm{Const}(t)$. We denote by $\mathcal{A}$ the set of all constants and variables. A substitution σ is an idempotent mapping from $\mathcal{X}$ to $\mathcal{T}(\mathcal{F},\mathcal{X})$ such that $\mathrm{Supp}(\sigma)=\{x|\sigma(x)\neq x\}$, the *support* of σ, is a finite set. The application of a substitution σ to a term t is denoted $t\sigma$ and is equal to the term t where all variables x have been replaced by the term $x\sigma$. A substitution σ is *ground* w.r.t. $\mathcal{F}$ if the image of $\mathrm{Supp}(\sigma)$ is included in $\mathcal{T}(\mathcal{F})$.

The set of the *subterms* of a term t, denoted $\mathrm{Sub}(t)$, is defined inductively as follows. If t is a constant or a variable then $\mathrm{Sub}(t)=\{t\}$. Otherwise, t must be of the form $f(t_1,\ldots,t_n)$, and we define $\mathrm{Sub}(t)=\{t\}\cup\bigcup_{i=1}^{n}\mathrm{Sub}(t_i)$.

The *positions* in a term t are defined recursively as usual (*i.e.* as sequences of integers), ϵ being the empty sequence. We denote by $t_{|p}$ the subterm of t at position p. We denote by $t[p \leftarrow s]$ the term obtained by replacing in t the syntactic subterm $t_{|p}$ by s.

2.2 Equational theories and Unification

We consider in this paper an equational theory $\mathcal{E}$ that defines a congruence on the terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$. We assume it is consistent, *i.e.* that it has a model with more than one element. *Ordered rewriting* [23] then permits us to employ the unfailing completion procedure of [25] to produce a (possibly infinite) set of equations for which ordered rewriting is convergent *on ground terms*, its *o-completion*. In turn, this convergence permits us to constructively choose one element in the congruence class of each ground term t, called its *normal form*, and denoted $(t)\downarrow$. We use in this paper the fact that since ordered rewriting is a relation on ground terms, if a term t is ground then the term $(t)\downarrow$ is also a ground term.

This construction relies on the assumption that the ground terms are totally ordered by a simplification ordering, and that the minimum for this ordering is a free constant $\mathrm{c_{min}}$.

2.2.1 Unification and equational theory type

Our result on deduction systems may seem vacuous as the definitions—based on an ordering on the "attacks" on a protocol—are not constructive. They however follow a classical line of definitions in the context of unification and equational theories. We present in this subsection these classical notions (and refer the reader *e.g.* to [27] for a more complete overview) in order to hilight the similitudes between our definitions and the classical ones for unification.

Definition 1 *($\mathcal{E}$-unifiers) Let $\mathcal{E}$ be an equational theory. We say that two terms t and s are* $\mathcal{E}$-equal*, and denote $s =_{\mathcal{E}} t$, if $\mathcal{E} \models_{=} t = s$. We say that a substitution σ is a* $\mathcal{E}$-unifier *of s and t if $\mathcal{E} \models_{=} t\sigma = s\sigma$.*

We say that two terms that have a $\mathcal{E}$-unifier are *$\mathcal{E}$-unifiable*.

We denote $\Sigma_{\mathcal{E}}(t, t')$ the set of all unifiers of t and t'. This set is not empty if, and only if, t and t' are unifiable. We extend the notion of unifier to conjunctions of equations as follows.

Definition 2 *(Unification systems) Let $\mathcal{E}$ be an equational theory. An* $\mathcal{E}$-Unification system *S is a finite set of equations denoted by $\{u_i \stackrel{?}{=} v_i\}_{i\in\{1,\ldots,n\}}$ with terms $u_i, v_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. It is satisfied by a substitution σ, and we note $\sigma \models_{\mathcal{E}} S$, if for all $i \in \{1, \ldots, n\}$ $u_i\sigma =_{\mathcal{E}} v_i\sigma$.*

One defines an *instantiation ordering* on unifiers by setting $\sigma \leq_i \tau$ whenever there exists a substitution θ such that $\sigma\theta =_{\mathcal{E}} \tau$. Equational theories are classified [33] *w.r.t.* the possible cardinalities of *complete sets of unifiers*.

Definition 3 *(Complete set of unifiers) Let $\mathcal{E}$ be an equational theory and t, t' be two terms. We say that a subset S of $\Sigma_{\mathcal{E}}(t, t')$ is a* complete set of unifiers *of t and t' if, for every substitution $\sigma \in \Sigma_{\mathcal{E}}(t, t')$ there exists a substitution $\tau \in S$ and a substitution θ such that $\tau\theta =_{\mathcal{E}} \sigma$.*

Or, using the instantiation ordering terminology, a complete set of unifiers is a set of minimal unifiers for the instantiation ordering such that every unifier is an instance of a unifier in this set. Finally, we define a set of *most general unifiers* to be a minimal set, for standard set inclusion, among the complete sets of unifiers. The rationale for this definition is that modulo an equational theory, two substitutions may be non-trivial instances one of the other. In this case one of the two is redundant and can be removed, hence the following definition.

Definition 4 *(Most general $\mathcal{E}$-unifiers) Let $\mathcal{E}$ be an equational theory. We call a set of most general $\mathcal{E}$-unifiers of t and t', and denote $\mathrm{mgu}_{\mathcal{E}}(t, t')$, a minimal (for set inclusion) complete set of unifiers of two terms t and t'.*

In the rest of this paper, and as long as there is no ambiguity, we simply refer to such sets as sets of most general unifiers, or sets of mgu. Also, the notion of mgu is extended as usual to unification systems. One proves the next lemma by constructing explicitly an injection from each complete set of unifiers to the other.

Lemma 1 *Let $\mathcal{E}$ be an equational theory, t, t' be two terms, and S, S' be two sets of most general unifiers of t and t'. Then S and S' have the same cardinality.*

The finiteness or even the existence of a minimal complete set of unifiers of two terms unifiable modulo $\mathcal{E}$ is not guaranteed. We say that an equational theory is *finitary* whenever, for every two unifiable terms t, t', $\mathrm{mgu}_{\mathcal{E}}(t, t')$ is a finite set.

One important property of unification systems that we shall use in the rest of this paper is the following replacement property.

Lemma 2 *For any equational theory $\mathcal{E}$, if a $\mathcal{E}$-unification system $\mathcal{S}$ is satisfied by a substitution σ, and c is any free constant in C away from $\mathcal{S}$, then for any term t, $\sigma\delta_{c,t}$ is also a solution of $\mathcal{S}$.*

Variables and constants. Using Lemma 2 we can clarify the difference and similitudes between variables and free constants. First, a formal point: since free constants do not occur in the equations of the equational theory they are not among the constants obtained by skolemization. Second, we agree that in the resolution procedure [1], variables have a special role whereas by Herbrand's theorem we know that it suffices to consider models of a set of clauses with at most one free constant. In spite of this we almost use variables and free constants (as in Lemma 2) interchangeably.

The rationale is that ordered completion yields a rewriting relation which is convergent on *ground* terms, and thus cannot be employed to normalize terms

that contain variables. Lemma 2 is thus fundamental since it implies that some of the free constants that may appear in an unifier can be replaced, the main difference with variables being that if, for a simplification ordering $<$, we have $t < t'$, then for every substitution σ we also have $t\sigma < t'\sigma$, whereas it is not the case that for every replacement $\delta_{c,s}$ we also have $t\delta_{c,s} < t'\delta_{c,s}$.

2.3 Deduction systems

Our protocol analysis is based on the assumption that all the agents operate on messages *via* a message manipulation library. We consider a signature $\mathcal{F}$ containing the function symbols employed to denote the messages, with a special subset of symbols $\mathcal{F}_p$ denoting the functions of the library which can be employed by all participants.

Definition 5 *(Deduction systems) A* deduction system *is defined by a triple* $(\mathcal{E}, \mathcal{F}, \mathcal{F}_p)$ *where* $\mathcal{E}$ *is an equational presentation on a signature* $\mathcal{F}$ *and* $\mathcal{F}_p$ *a subset of* public *constructors in* $\mathcal{F}$*.*

Example 1 *For instance the following deduction system models public key cryptography:*

$$(\{\mathrm{dec_p}(\mathrm{enc_p}(x,y),y^{-1}) = x\},\\ \{\mathrm{dec_p}(\_,\_),\mathrm{enc_p}(\_,\_),\_^{-1}\},\\ \{\mathrm{dec_p}(\_,\_),\mathrm{enc_p}(\_,\_)\})$$

The equational theory is reduced here to a single equation that expresses that one can decrypt a cipher text when the inverse key is available.

3 Symbolic derivations

We present in this section our model for agents.

3.1 Definitions

Symbolic derivations. Given a deduction system $(\mathcal{F}, \mathcal{P}, \mathcal{E})$, a role applies public symbols in $\mathcal{P}$ to construct a response from its initial knowledge and from messages received so far. Additionally, it may test equalities between messages to check the well-formedness of a message. Hence the activity of a role can be expressed by a fixed symbolic derivation:

Definition 6 *(Symbolic Derivations) A symbolic derivation for a deduction system* $(\mathcal{F}, \mathcal{P}, \mathcal{E})$ *is a tuple* $(\mathcal{V}, \mathcal{S}, \mathcal{K}, \mathrm{In}, \mathrm{Out})$ *where* $\mathcal{V}$ *is a mapping from a finite ordered set* $(\mathrm{Ind}, <)$ *to a set of variables* $\mathrm{Var}(\mathcal{V})$*,* $\mathcal{K}$ *is a set of ground terms (the initial knowledge)* In *is a subset of* Ind*,* Out *is a multiset of elements of* Ind *and* $\mathcal{S}$ *is a unification system.*

The set Ind *represents internal states of the symbolic derivation. We impose that any* $i \in \mathrm{Ind}$ *is exactly one of the following kind:*

Deduction state: *There exists a public symbol* $f \in \mathcal{P}$ *of arity* n *such that* $\mathcal{V}(i) \stackrel{?}{=} f(\mathcal{V}(\alpha_1), \ldots, \mathcal{V}(\alpha_n)) \in \mathcal{S}$ *with* $\alpha_j < i$ *for* $j \in \{1, \ldots, n\}$.

Re-use state: *if there exists* $j < i$ *with* $\mathcal{V}(j) = \mathcal{V}(i)$*;*

Memory state: *if there exists* t *in* $\mathcal{K}$ *and an equation* $\mathcal{V}(i) \stackrel{?}{=} t$ *in* $\mathcal{S}$*;*

Reception state: *if* $i \in$ In*;*

Additionally, a state i *is also an* ***emission state*** *if* $i \in$ Out.

The unification system $\mathcal{S}$ *contains no equation but those described above and equations* $\mathcal{V}(i) \stackrel{?}{=} \mathcal{V}(j)$*, and the mapping* $\mathcal{V}$ *must be injective on non-re-use states.*

A symbolic derivation is closed *if it has no reception state. A substitution* σ satisfies *a closed symbolic derivation if* $\sigma \models_{\mathcal{E}} \mathcal{S}$*.*

We believe that using symbolic derivations instead of more standard constraint systems permits one to simplify the proofs by having a more homogeneous framework. There is however one drawback to their usage. While most of the time it is convenient to have an identification between the order of deduction of messages and their send/receive order, building in this identification too strictly would prevent us from expressing simple problems. Re-use states are employed to reorder the deduced messages to fit an order of sending messages which can be different. For example consider an intruder that knows (after reception) two messages a and b received in that order, and that he has to send first b, then a. Since the states in a symbolic derivation have to be ordered, we have to use at least one re-use state (for a) to be able to consider a sending of a *after* the sending of b. We note that re-use states that are not employed in a connection can be safely eliminated without changing the deductions, the definition of the knowledge nor the tests in the unification system.

With respect to earlier definitions, we have chosen to consider injective variable-state mapping functions. The rationale for this choice is essentially aesthetic, as using this more strict definition implies that every equality test performed by the attacker is an equality $\mathcal{V}(i) \stackrel{?}{=} \mathcal{V}(j)$ in the unification system. Not having this restriction would require the introduction of *a*) an equivalence class on ASDs to model the fact that two ASDs can be solutions to exactly the same HSDs, and *b*) the subset of ASDs that have an injective variable-state mapping function,and *c*) the construction, by adding equality tests, for every ASD of an equivalent ASD in this subset.

Example 2 *Let us consider the cryptographic protocol for deduction system* $\mathcal{DY}$ *where* $\mathcal{F}_{\mathcal{D}}$ *and* $\mathcal{P}_{\mathcal{D}}$ *have been extended by a free public symbol* f*:*

$A \to B$: $\mathrm{enc_p}(N_a, \mathrm{pk}(B))$
$B \to A$: $\mathrm{enc_p}(f(N_a), \mathrm{pk}(A))$
where
A ***knows*** $A, B, \mathrm{pk}(B), \mathrm{pk}(A), \mathrm{sk}(A)$
B ***knows*** $A, B, \mathrm{pk}(A), \mathrm{pk}(B), \mathrm{sk}(B)$

Let us define a symbolic derivation for role B:

$$\begin{array}{rcl}
\textsc{Ind}_B & = & \{1,\ldots,9\} \\
\mathcal{V}_B & = & i \in \textsc{Ind} \mapsto x_i \\
\mathcal{K}_B & = & \{A, B, \mathrm{pk}(A), \mathrm{pk}(B), \mathrm{sk}(B)\} \\
\textsc{In}_B & = & \{6\} \\
\textsc{Out}_B & = & \{9\} \\
\mathcal{S}_B & = & \{x_1 \stackrel{?}{=} A, x_2 \stackrel{?}{=} B, x_3 \stackrel{?}{=} \mathrm{pk}(A), x_4 \stackrel{?}{=} \mathrm{pk}(B), x_5 \stackrel{?}{=} \mathrm{sk}(B) \\
 & & x_7 \stackrel{?}{=} \mathrm{dec_p}(x_6, x_5), x_8 \stackrel{?}{=} f(x_7), x_9 \stackrel{?}{=} \mathrm{enc_p}(x_8, x_3)\}
\end{array}$$

The set of deduction states in B is $\{7,8,9\}$, there are no re-use state, the set of memory states is $\{1,\ldots,5\}$ and the only reception state is 6. Assuming that the role B tests whether the received message is a cipher, one may add a tenth deduction state with $x_{10} \stackrel{?}{=} \mathrm{enc_p}(x_7, x_4)$ and an equation $x_6 \stackrel{?}{=} x_{10}$.

Similarly, a symbolic derivation for role A would be:

$$\begin{array}{rcl}
\textsc{Ind}_A & = & \{1,\ldots,10\} \\
\mathcal{V} & = & i \in \textsc{Ind} \mapsto y_i \\
\mathcal{K} & = & \{A, B, \mathrm{pk}(A), \mathrm{pk}(B), \mathrm{sk}(A), Na\} \\
\textsc{In} & = & \{9\} \\
\textsc{Out} & = & \{7\} \\
\mathcal{S} & = & \{y_1 \stackrel{?}{=} A, y_2 \stackrel{?}{=} B, y_3 \stackrel{?}{=} \mathrm{pk}(A), y_4 \stackrel{?}{=} \mathrm{pk}(B), y_5 \stackrel{?}{=} \mathrm{sk}(A), y_6 \stackrel{?}{=} Na \\
 & & y_7 \stackrel{?}{=} \mathrm{enc_p}(y_5, y_3), y_8 \stackrel{?}{=} f(y_6), y_{10} \stackrel{?}{=} \mathrm{dec_p}(y_9, y_5), y_{10} \stackrel{?}{=} y_8\}
\end{array}$$

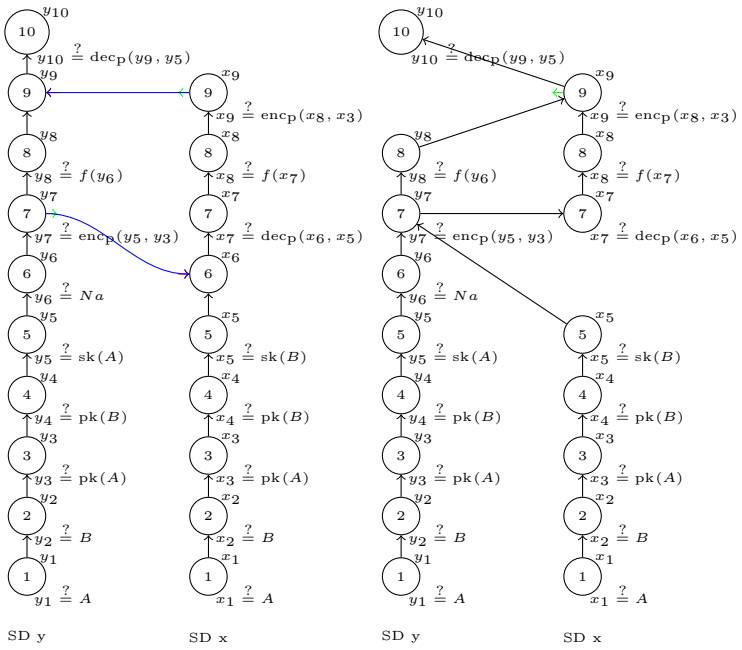
The set of deduction states in A is $\{6,7,9\}$, there are no re-use state, the set of memory states is $\{0,\ldots,5\}$ and the only reception state is 8. We have added an equality test $y_9 \stackrel{?}{=} y_7$ to model that A checks whether the message received actually contains the encryption of $f(Na)$. Generally speaking, if ground reachability and ground symbolic equivalence for the deduction system are decidable (see Section 3.3) then an as prudent as possible *set of deductions and equality tests for the narration can be computed (see [17]).*

In addition we assume that two symbolic derivations do not share any variable, and that equality between symbolic derivations is defined modulo a renaming of variables. The proof of the following lemma is a direct consequence of the definition.

Lemma 3 *(Properties of symbolic derivations) Let $\mathcal{C} = (\mathcal{V}, \mathcal{S}, \mathcal{K}, \textsc{In}, \textsc{Out})$ be a symbolic derivation. We have:*

(i)

1. *For every variable $\mathcal{V}(i)$ there is at most one equation in $\mathcal{S}$ of the form $\mathcal{V}(i) \stackrel{?}{=} f(t_1,\ldots,t_n)$;*
2. *If $\mathcal{V}(i)$ is a variable such that the above equation is in $\mathcal{S}$, then either a) i is a deduction state and $i = \min(j \mid \mathcal{V}(i) = \mathcal{V}(j))$, or b) i is a re-use state.*



(a) With a connection, before computing the resulting symbolic derivation

(b) After computing the result-symbolic derivation resulting from the connection

Figure 1: Honest symbolic derivations of Example 2 with a connection corresponding to the intended communications and the test equations not shown

We rely on the normal form defined by the o-completion of the equational theory $\mathcal{E}$ to prove that every closed symbolic derivation defines in a unique way the terms deduced.

Lemma 4 *Let $\mathcal{I}$ be a deduction system, and consider a closed and satisfiable $\mathcal{I}$-symbolic derivation $\mathcal{C} = (\mathcal{V}, \mathcal{S}, \mathcal{K}, \text{In}, \text{Out})$. Then there exists a unique ground substitution σ in normal form that satisfies $\mathcal{S}$.*

By Lemma 4, if a derivation is closed, then for every $i \in \text{Ind}$ the variable $\mathcal{V}(i)$ is instantiated by a ground term. Figuratively we say that a term t is *known at step i* in a closed symbolic derivation if there exists $j \leq i$ such that $\mathcal{V}(j)$ is instantiated by t.

Ground symbolic derivations. An important case when considering protocol refutation is the one in which the attacker cannot alter the messages exchanged among the honest participants. This case can either be employed to model a weaker attacker or, when trying to refute a cryptographic protocol, by

guessing first which messages are sent by the attacker, and then by checking whether these guesses correspond to messages the attacker can actually send.

Definition 7 *(Ground symbolic derivation) We say that a symbolic derivation* $\mathcal{C}_h = (\mathcal{V}_h, \mathcal{S}_h, \mathcal{K}_h, \text{IN}_h, \text{OUT}_h)$ *is a* ground symbolic derivation *whenever* $\mathcal{S}_h$ *is satisfiable and there exists a ground substitution* σ *such that, for every unifier* τ *of* $\mathcal{S}_h$ *and every* $i \in \text{IND}_h$ *we have* $\Updownarrow\rangle\backslash_h(i)\sigma = \Updownarrow\rangle\backslash_h(i)\tau$.

In other words the input and output messages of a ground symbolic derivation are fixed ground terms. We note that since $\mathcal{C}_h$ is not closed, and in spite of having $\mathcal{S}_h$ satisfiable, it is not necessarily true that $\mathcal{C}_h^\star \neq \emptyset$. Also a simple analysis of the case study of the proof of Lemma 4 shows that it suffices to assume that σ is defined only on indexes $i \in \text{IN}_h$.

Connection. We express the communication between two agents represented each by a symbolic derivation by *connecting* these symbolic derivations. This operation consists in identifying some input variables of one derivation with some output variables of the other and contrariwise. This connection should be compatible with the variable orderings inherited from each symbolic derivation, as detailed in the following definition:

Definition 8 *Let* $\mathcal{C}_1$, $\mathcal{C}_2$ *be two symbolic derivations with for* $i \in \{1, 2\}$ $\mathcal{C}_i = (\mathcal{V}_i, \mathcal{S}_i, \mathcal{K}_i, \text{IN}_i, \text{OUT}_i)$, *with disjoint sets of variables and index sets* $(\text{IND}_1, <_1)$ *and* $(\text{IND}_2, <_2)$ *respectively. Let* I_1, I_2, *be subsets of* IN_1, IN_2, *and* O_1, O_2 *be sub-multisets of* OUT_1, OUT_2 *respectively.*

Assume that there is a monotone bijection ϕ *from* $I_1 \cup I_2$ *to* $O_1 \cup O_2$ *such that* $\phi(I_1) = O_2$ *and* $\phi(I_2) = O_1$. *A* connection *of* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *over the* connection *function* ϕ, *denoted* $\mathcal{C}_1 \circ_\phi \mathcal{C}_2$ *is a symbolic derivation*

$$\mathcal{C} = (\mathcal{V}, \phi(\mathcal{S}_1 \cup \mathcal{S}_2), \mathcal{K}_1 \cup \mathcal{K}_2, (\text{IN}_1 \cup \text{IN}_2) \setminus (I_1 \cup I_2), (\text{OUT}_1 \cup \text{OUT}_2) \setminus (O_1 \cup O_2))$$

where:

- $(\text{IND}, <)$ *is defined by:*
  - $\text{IND} = (\text{IND}_1 \setminus I_1) \cup (\text{IND}_2 \setminus I_2)$;
  - $<$ *is the transitive closure of the relation:* $<_1 \cup <_2$;
- ϕ *is extended to a renaming of variables in* $\text{Var}(\mathcal{V}_1) \cup \text{Var}(\mathcal{V}_2)$ *such that* $\phi(\mathcal{V}_1(i)) = \mathcal{V}_2(j)$ *(resp.* $\phi(\mathcal{V}_2(i)) = \mathcal{V}_1(j)$*) if* $i \in I_1$ *(resp.* I_2*) and* $\phi(i) = j$

When the exact connection function in a connection does not matter, is uniquely defined, or is described otherwise, we will omit the subscript and denote it $\mathcal{C}_1 \circ \mathcal{C}_2$.

A connection is *satisfiable* if the resulting symbolic derivation is satisfiable. It can easily computed, when it exists, by considering increasing sequences of states in each symbolic derivation and mapping input states of one SD with output states of the other.

Example 3 *Let $\mathcal{C}_h$ be the symbolic derivation in Example 2:*

$$\begin{array}{rcl}
\textsc{Ind}_h & = & \{0,\ldots,8\} \\
\mathcal{V}_h & = & i \in \textsc{Ind} \mapsto x_i \\
\mathcal{K}_h & = & \{A, B, \mathrm{pk}(A), \mathrm{pk}(B), \mathrm{sk}(B)\} \\
\textsc{In}_h & = & \{5\} \\
\textsc{Out}_h & = & \{0,\ldots,8,8\} \\
\mathcal{S}_h & = & \{x_0 \stackrel{?}{=} A, x_1 \stackrel{?}{=} B, x_2 \stackrel{?}{=} \mathrm{pk}(A), x_3 \stackrel{?}{=} \mathrm{pk}(B), x_4 \stackrel{?}{=} \mathrm{sk}(B) \\
 & & x_6 \stackrel{?}{=} \mathrm{dec_p}(x_5, x_4), x_7 \stackrel{?}{=} f(x_6), x_8 \stackrel{?}{=} \mathrm{enc_p}(x_7, x_2)\}
\end{array}$$

We model the initial knowledge of the intruder with another symbolic derivation $\mathcal{C}_K$:

$$\begin{array}{rcl}
\textsc{Ind}_K & = & \{0^k,\ldots,3^k\} \\
\mathcal{V}_K & = & i^k \in \textsc{Ind}_k \mapsto y_i \\
\mathcal{K}_K & = & \{A, B, \mathrm{pk}(A), \mathrm{pk}(B)\} \\
\textsc{In}_K & = & \emptyset \\
\textsc{Out}_K & = & \textsc{Ind}_K \\
\mathcal{S}_K & = & \{y_0 \stackrel{?}{=} A, y_1 \stackrel{?}{=} B, y_2 \stackrel{?}{=} \mathrm{pk}(A), y_3 \stackrel{?}{=} \mathrm{pk}(B)\}
\end{array}$$

and we let $\mathcal{C}'$ be the following derivation:

$$\begin{array}{rcl}
\textsc{Ind}' & = & \{0',\ldots,8\} \\
\mathcal{V}' & = & i' \in \textsc{Ind}' \mapsto z_i \\
\mathcal{K} & = & \{n\} \subset C_{new} \\
\textsc{In}' & = & \{0',\ldots,3',8'\} \\
\textsc{Out}' & = & \{5'\} \cup \textsc{Ind}' \\
\mathcal{S}' & = & \{z_4 \stackrel{?}{=} n, z_5 \stackrel{?}{=} \mathrm{enc_p}(z_4, z_3), \\
 & & z_6 \stackrel{?}{=} f(z_4), z_7 \stackrel{?}{=} \mathrm{enc_p}(z_6, z_2), z_8 \stackrel{?}{=} z_7\}
\end{array}$$

Let ϕ be the application from $0^k,\ldots,3^k,5',8$ to $0',\ldots,3',5,8'$ respectively and ψ be a function of empty domain. Then we have $(\mathcal{C}_h \circ_\psi \mathcal{C}_K) \circ_\phi \mathcal{C}'$:

$$\begin{array}{rcl}
\textsc{Ind} & = & \{0,\ldots,4,0^k,\ldots,3^k,5',6',7',6,7,8\} \\
\mathcal{V} & = & \mathcal{V}_{h|\textsc{Ind}} \cup \mathcal{V}_{K|\textsc{Ind}} \cup \mathcal{V}'_{|\textsc{Ind}} \\
\mathcal{K} & = & \{A, B, \mathrm{pk}(A), \mathrm{pk}(B), \mathrm{sk}(B), n\} \\
\textsc{In} & = & \emptyset \\
\textsc{Out} & = & \textsc{Ind} \cap \textsc{Ind}' \\
\mathcal{S} & = & \{x_0 \stackrel{?}{=} A, x_1 \stackrel{?}{=} B, x_2 \stackrel{?}{=} \mathrm{pk}(A), x_3 \stackrel{?}{=} \mathrm{pk}(B), x_4 \stackrel{?}{=} \mathrm{sk}(B) \\
 & & x_6 \stackrel{?}{=} \mathrm{dec_p}(x_5, x_4), x_7 \stackrel{?}{=} f(x_6), x_8 \stackrel{?}{=} \mathrm{enc_p}(x_7, x_2) \\
 & & y_0 \stackrel{?}{=} A, y_1 \stackrel{?}{=} B, y_2 \stackrel{?}{=} \mathrm{pk}(A), y_3 \stackrel{?}{=} \mathrm{pk}(B) \\
 & & z_5 \stackrel{?}{=} n, z_6 \stackrel{?}{=} \mathrm{enc_p}(z_5, z_3), \\
 & & z_7 \stackrel{?}{=} f(z_5), z_8 \stackrel{?}{=} \mathrm{enc_p}(z_7, z_2), z_9 \stackrel{?}{=} z_8\}
\end{array}$$

with the ordering:

$$\begin{array}{l}
0 < 1 < 2 < 3 < 4 < 5' < 6 < 7 < 8 \\
0^k < \ldots < 3^k < 4' < \ldots < 7' < 8
\end{array}$$

The connection of two symbolic derivations $\mathcal{C}_1$ and $\mathcal{C}_2$ identifies variables in the input of one with variables in the output of the other. Variables that have been identified are removed from the input/output set of the resulting symbolic derivation $\mathcal{C}$. The set of equality constraints of $\mathcal{C}$ is the union of the equality constraints in $\mathcal{C}_1$ and $\mathcal{C}_2$, plus equalities stemming from the identification of input and output. We have chosen to have a multiset of output variables to enable the modeler to specify whether a communication between two participants is hidden—when the output state occurs only once in the initial output multiset—or visible—in which case there is more than one occurrence of the output state in the initial output multiset—to an external observer.

One easily checks that a connection of two symbolic derivations is also a symbolic derivation. Also, the associativity of function composition applied on the connections implies the associativity of the connection of symbolic derivations. Since connection functions are bijective, we will also identify $\mathcal{C} \circ \mathcal{C}'$ and $\mathcal{C}' \circ \mathcal{C}$. Thus when we compose several symbolic derivations, we will freely re-arrange or remove parentheses.

Traces. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two $\mathcal{I}$-symbolic derivations and φ be a connection such that $\mathcal{C} = \mathcal{C}_1 \circ_\varphi \mathcal{C}_2 = (\mathcal{V}, \mathcal{S}, \mathcal{K}, \text{In}, \text{Out})$ is closed and satisfiable. Lemma 4 implies that there exists a unique ground substitution τ in normal form such that any unifier σ of $\mathcal{S}_1 \cup \mathcal{S}_2$ is equal to τ on the image of $\mathcal{V}$. We denote $\text{Tr}_{\mathcal{C}_1 \circ_\varphi \mathcal{C}_2}(\mathcal{C}')$ the restriction of this substitution τ to the variables in the sequence of $\mathcal{C}'$, for $\mathcal{C}' \in \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_1 \circ_\varphi \mathcal{C}_2\}$, and call it the *trace* of the connection on $\mathcal{C}'$. In the rest of this paper we will always assume that trace substitutions are in normal form.

3.2 Solutions of symbolic derivations

3.2.1 Honest and attacker symbolic derivations

Generally speaking, a *solution* of a symbolic derivation $\mathcal{C}$ is any couple $(\mathcal{C}', \varphi)$ such that $\mathcal{C} \circ_\varphi \mathcal{C}'$ is closed and satisfiable. We specialize this definition for the case of protocol analysis in order to ensure that every term possessed by the attacker, including her initial knowledge, has been either leaked by the protocol or is a nonce she has created. This consideration lead us to consider two types of symbolic derivations, one that is employed to model honest agents, and one to model an attacker.

Honest derivations. We do not impose constraints on the symbolic derivations representing honest principals, but for the avoidance of constants in an infinite set $C_{\text{new}} \subseteq C$. These constants are employed to model new values created by an attacker. We assume that nonces created by the honest agents are created at the beginning of their execution and are constants away from C_{new}.

Definition 9 *(Honest symbolic derivations) A symbolic derivation $\mathcal{C}$ is an* honest symbolic derivation *or HSD, if the constants occurring in $\mathcal{C}$ are away from C_{new}.*

Example 4 *The symbolic derivation for role B in Example 2 is honest.*

Attacker derivations. We consider an attacker modeled by a symbolic derivation in which only the following actions are possible:

- create a fresh, random value;
- receive from and send a message to one of the honest participant;
- deduce a new message from the set of already known messages;
- every state is in $\textsc{Out}$ given that the intruder should be able to observe his own knowledge;
- given that we consider an actual execution, the set of states is totally ordered.

The definition of *attacker* symbolic derivations models these constraints:

Definition 10 *(Attacker symbolic derivations) Let* $\mathcal{C} = (\mathcal{V}, \mathcal{S}, \mathcal{K}, \textsc{In}, \textsc{Out})$ *be a symbolic derivation. It is an* attacker symbolic derivation*, or ASD, if a)* $\textsc{Ind}$ *is a total order, and b)* $\textsc{Out}$ *contains at least one occurrence of each index in* $\textsc{Ind}$*, and c)* $\mathcal{K}$ *is a subset of* C_{new}*.*

The fact that the initial knowledge of the attacker is empty but for the nonces is not a restriction when analyzing protocols, as one can see from Ex. 3.

Example 5 *The following derivation* $\mathcal{C}'$ *is an ASD for the same deduction system as Example 2:*

$$\begin{aligned}
\textsc{Ind}' &= \{0', \ldots, 8\} \\
\mathcal{V}' &= i' \in \textsc{Ind}' \mapsto z_i \\
\mathcal{K} &= \{n\} \subset C_{new} \\
\textsc{In}' &= \{0', \ldots, 3', 8'\} \\
\textsc{Out}' &= \{5'\} \cup \textsc{Ind}' \\
\mathcal{S}' &= \{z_4 \stackrel{?}{=} n, z_5 \stackrel{?}{=} \mathrm{enc_p}(z_4, z_3), \\
&\quad\ z_6 \stackrel{?}{=} f(z_4), z_7 \stackrel{?}{=} \mathrm{enc_p}(z_6, z_2), z_8 \stackrel{?}{=} z_7\}
\end{aligned}$$

Informally the ASD expresses that the attacker receives some key k*, creates a nonce* n*, sends the encrypted nonce to a role* B *as in Example 2. Then the attacker tries to check that applying* f *to* n *gives a term equal to the decryption of B's response.*

Solutions of a symbolic derivation. Given a symbolic derivation $\mathcal{C}_h$ we denote $\mathcal{C}_h^\star$ the set of couples $(\mathcal{C}, \varphi)$ where $\mathcal{C}$ is an ASD and φ is a connection function between $\mathcal{C}$ and $\mathcal{C}_h$ such that $\mathcal{C}_h \circ \mathcal{C}$ is closed and satisfiable. In that case we say that $\mathcal{C}$ is a solution of $\mathcal{C}_h$.

Example 6 *In Example 3 the ASD* $\mathcal{C}'$ *is a solution of* $\mathcal{C}_h \circ \mathcal{C}_K$ *since* $(\mathcal{C}_h \circ_\psi \mathcal{C}_K) \circ_\phi \mathcal{C}'$ *is closed and* $\mathcal{S}$ *is satisfiable (by simply propagating the equalities* $x_0 = A, x_1 = B, \ldots$*).*

3.3 Decision problems

Satisfiability. The problem of the existence of a secrecy attack on a bounded protocol execution—shown to be NP-complete in [31] for the standard Dolev-Yao deduction system—is equivalent to the satisfiability problem below.

$\mathcal{I}$-Satisfiability

Input: a HSD $\mathcal{C}$
Output: SAT iff $\mathcal{C}^\star \neq \emptyset$

A variant of $\mathcal{I}$-satisfiability is its restriction to set of inputs $\mathcal{C}$ which are ground symbolic derivations, and that we call $\mathcal{I}$-ground satisfiability.

Ground $\mathcal{I}$-Satisfiability

Input: a ground HSD $\mathcal{C}$
Output: SAT iff $\mathcal{C}^\star \neq \emptyset$

Equivalence. Let us now define the equivalence of HSDs *w.r.t.* an active intruder.

Definition 11 *Two HSDs $\mathcal{C}_h$ and $\mathcal{C}'_h$ are* symbolically equivalent *iff $\mathcal{C}_h^\star = {\mathcal{C}'_h}^\star$.*

$\mathcal{I}$-Symbolic Equivalence

Input: Two honest $\mathcal{I}$-symbolic derivations $\mathcal{C}_h$ and $\mathcal{C}'_h$
Output: SAT iff ${\mathcal{C}_h}^\star = {\mathcal{C}'_h}^\star$.

Again it is possible to define a ground version of the $\mathcal{I}$-symbolic equivalence problem when the input consists in two ground symbolic derivations. One can easily encode static equivalence problems into ground $\mathcal{I}$-Symbolic Equivalence problems by publishing every constant not hidden in the frame.

Ground $\mathcal{I}$-Symbolic Equivalence

Input: Two honest $\mathcal{I}$-ground symbolic derivations $\mathcal{C}_h$ and $\mathcal{C}'_h$
Output: SAT iff ${\mathcal{C}_h}^\star = {\mathcal{C}'_h}^\star$.

Remark. Another possible definition of the set of solutions would be a set of ASDs, without mention of the connection function. The equivalence relation would have been distinct since in that case an ASD can be in two sets of solutions but without the same connection function. However, this would have had no impact on our decidability result. Our choice in this paper corresponds to diff-equivalence between biprocesses [11]: the diff operator defines a bijection between the in- and output states of two processes derivations, and the equality of the sets of solutions is understood modulo this one-to-one function.

4 Finitary Deduction Systems

An equational theory $\mathcal{E}$ is *finitary* whenever every $\mathcal{E}$-unification system has a finite set of more general unifiers. We define an analog for deduction systems *w.r.t.* symbolic derivations rather than equational theories *w.r.t.* unification systems. In the rest of this paper, we consider *effective* finitary deduction systems, *i.e.* deduction systems for which it is possible to compute a finite set of "most general attacks".

4.1 Stutter-free ASDs

We say that an ASD $\mathcal{C}_\mathcal{I}$ is *well-formed w.r.t.* a HSD $\mathcal{C}_h$ and a connection φ if, in the connection $\mathcal{C}_h \circ_\varphi \mathcal{C}_\mathcal{I}$, a deduction subsequently applied on a deduced term t, or a re-use of the term t is always applied by referring to the state in which t was first deduced.

Definition 12 *(Well-formed ASD) Let $\mathcal{C}_h$ be a HSD and consider an ASD $\mathcal{C}_\mathcal{I} = (\mathcal{V}_\mathcal{I}, \mathcal{S}_\mathcal{I}, \mathcal{K}_\mathcal{I}, \text{IN}_\mathcal{I}, \text{OUT}_\mathcal{I})$ such that $(\mathcal{C}_\mathcal{I}, \varphi) \in \mathcal{C}_h^\star$, and $\sigma = \text{Tr}_{\mathcal{C}_\mathcal{I} \circ_\varphi \mathcal{C}_h}(\mathcal{C}_\mathcal{I})$. We say that $\mathcal{C}_\mathcal{I}$ is* $(\mathcal{C}_h, \varphi)$-well-formed *if for every deduction states i, for every state $j \in \text{IND}_\mathcal{I}$ with $i < j$ we have $\mathcal{V}_\mathcal{I}(i)\sigma = \mathcal{V}_\mathcal{I}(j)\sigma$ implies that*

- *either $\mathcal{V}_\mathcal{I}(i) = \mathcal{V}_\mathcal{I}(j)$, i.e. j is a re-use state;*
- *or there is no equation $x \stackrel{?}{=} f(\ldots, \mathcal{V}_\mathcal{I}(j), \ldots)$ in $\mathcal{S}_\mathcal{I}$ and j is not an emission state.*

This restriction is mostly syntactic, and can be assumed *w.l.o.g.* for our purpose, as shown by the Lemma 8.

Our aim is the reduction of equivalence problems to reachability problems for finitary deduction systems. In the latter problems, one only considers which terms are deducible by the attacker. Hence the following definitions that will be employed to split an ASD into a *deduction only* part solving a reachability problem and a *testing* part modeling the possible tests.

Definition 13 *(Deduction-only ASD) An ASD $\mathcal{C}_\mathcal{I} = (\mathcal{V}_\mathcal{I}, \mathcal{S}_\mathcal{I}, \mathcal{K}_\mathcal{I}, \text{IN}_\mathcal{I}, \text{OUT}_\mathcal{I})$ is* deduction-only *if $\mathcal{S}_\mathcal{I}$ contains no equation $\mathcal{V}_\mathcal{I}(i) \stackrel{?}{=} \mathcal{V}_\mathcal{I}(j)$.*

Definition 14 *(Testing ASD) An ASD $\mathcal{C}_\mathcal{I} = (\mathcal{V}_\mathcal{I}, \mathcal{S}_\mathcal{I}, \mathcal{K}_\mathcal{I}, \text{IN}_\mathcal{I}, \text{OUT}_\mathcal{I})$ is* testing *if $\mathcal{K}_\mathcal{I} = \emptyset$.*

Definition 15 *(Stutter-free ASDs) A well-formed deduction-only ASD is said to be* stutter-free.

Given a HSD $\mathcal{C}_h$ we denote ${\mathcal{C}_h}^{\text{sf}}$ the set of stutter-free solutions of $\mathcal{C}_h$. These ASDs have the special property that a connection cannot be unsatisfiable because of a rejection by the attacker. Formally speaking, we have the following proposition.

Proposition 1 *Let $\mathcal{C}_\mathcal{I} = (\mathcal{V}_\mathcal{I}, \mathcal{S}_\mathcal{I}, \mathcal{K}_\mathcal{I}, \text{IN}_\mathcal{I}, \text{OUT}_\mathcal{I}) \in \mathcal{C}_h^\star$ be a deduction-only ASD. Then for any ground substitution σ of domain $\text{IN}_\mathcal{I}$ the unification system $\mathcal{S}_\mathcal{I}\sigma$ is satisfiable in the empty theory.*

4.2 Sets of solutions

Outline. We prove in this section that ASDs are such that, when replacing a constant in C_{new} by the result of a sequence of compositions (this operation is called *opening*) we obtain another ASD which can be connected to all the HSDs

the original ASD could be connected to (Lemma 5). This notion of replacement acts as the instantiation of a unifier modulo an equational theory. Accordingly we define from it a well-founded ordering on ASDs mimicking the role of the instantiation ordering on unifiers. Finally, we prove that given a set of ASDs S, the inclusion $S \subseteq \mathcal{C}_h^\star$ can be check by testing only the minimal ASDs in S (Lemma 6).

Opening of symbolic derivations. If $\mathcal{C} = (\mathcal{V}, \mathcal{S}, \mathcal{K}, \text{IN}, \text{OUT})$ and $C \subseteq C_{\text{new}} \cap \mathcal{K}$ is a set such such that $C \cap \text{Sub}(\mathcal{K} \setminus C) = \emptyset$, we *open* $\mathcal{C}$ *on* C, and denote the operation $\text{open}_C(\mathcal{C})$, when for each $c \in C$:

- If $i \in \text{IND}$ is the first knowledge state with $\mathcal{V}(i) \stackrel{?}{=} c \in \mathcal{S}$, we remove this equation from $\mathcal{S}$ and add i to the input states;
- we replace all occurrences of c in $\mathcal{C}$ by $\mathcal{V}(i)$.

We note that the set $\mathcal{K}'$ obtained from $\mathcal{K}$ after the replacement is still a set of ground terms since $C \cap \text{Sub}(\mathcal{K} \setminus C) = \emptyset$, and thus the result of the operation is still a symbolic derivation. Also, $\mathcal{C}$ is an ASD, then so is $\text{open}_C(\mathcal{C})$.

Lemma 5 *Let $\mathcal{C}_\mathcal{I} \in \mathcal{C}_h^\star$ with $\mathcal{C}_\mathcal{I} = (\mathcal{V}_\mathcal{I}, \mathcal{S}_\mathcal{I}, \mathcal{K}_\mathcal{I}, \text{IN}_\mathcal{I}, \text{OUT}_\mathcal{I})$, let $C \subseteq \mathcal{K}_\mathcal{I}$ and let $\mathcal{C}_c \in {\mathcal{C}'_h}^{\text{sf}}$ for some HSD $\mathcal{C}'_h$. If a connection $\mathcal{C}_c \circ \mathcal{C}_h \circ \text{open}_C(\mathcal{C}_\mathcal{I})$ is closed then it is satisfiable.*

Ordering on symbolic derivations. Consider two symbolic derivations:

$$\begin{cases} \mathcal{C}_\mathcal{I} &= (\mathcal{V}_\mathcal{I}, \mathcal{S}_\mathcal{I}, \mathcal{K}_\mathcal{I}, \text{IN}_\mathcal{I}, \text{OUT}_\mathcal{I}) \\ \mathcal{C}'_\mathcal{I} &= (\mathcal{V}'_\mathcal{I}, \mathcal{S}'_\mathcal{I}, \mathcal{K}'_\mathcal{I}, \text{IN}'_\mathcal{I}, \text{OUT}'_\mathcal{I}) \end{cases}$$

We say that $\mathcal{C}_\mathcal{I} \leq \mathcal{C}'_\mathcal{I}$ if:

- there exists $C \subseteq \mathcal{K}_\mathcal{I}$, a stutter-free symbolic derivation $\mathcal{C}_C$ and a connection φ such that $\mathcal{C}_C \circ_\varphi \text{open}_C(\mathcal{C}_\mathcal{I}) = \mathcal{C}'_\mathcal{I}$ modulo a renaming of variables;
- or there exists a set of memory states $I \subseteq \text{IND}'_\mathcal{I}$ such that $C_\mathcal{I}$ is equal to $\mathcal{C}''_\mathcal{I} = (\mathcal{V}''_\mathcal{I}, \mathcal{S}''_\mathcal{I}, \mathcal{K}''_\mathcal{I}, \text{IN}''_\mathcal{I}, \text{OUT}''_\mathcal{I})$ where:
  - $\mathcal{V}''_\mathcal{I}$ is the restriction of $\mathcal{V}'_\mathcal{I}$ to the domain $\text{IND}'_\mathcal{I} \setminus I$
  - and $\mathcal{S}''_\mathcal{I} = \mathcal{S}'_\mathcal{I} \setminus \{\mathcal{V}'_\mathcal{I}(i) \stackrel{?}{=} c_i\}_{i \in I}$.

We say that $\mathcal{C}_\mathcal{I}, \mathcal{C}'_\mathcal{I}$ are *equivalent modulo a renaming of nonces*, and denote $\mathcal{C}_\mathcal{I} \equiv \mathcal{C}'_\mathcal{I}$, whenever there exists $C \subseteq \mathcal{K}_\mathcal{I}$, a stutter-free symbolic derivation $\mathcal{C}_C$ *with only memory states*, and a connection φ such that $\mathcal{C}_C \circ_\varphi \text{open}_C(\mathcal{C}_\mathcal{I}) = \mathcal{C}'_h$. Given a set S of ASDs we denote $\min_<(S)$ the set of ASDs in S that are minimal in S modulo renaming of nonces.

Since $\mathcal{C} \leq \mathcal{C}'$ implies that either: *a*) $\mathcal{C}$ has strictly less deduction states than $\mathcal{C}'$, and less states, *b*) $\mathcal{C}$ has strictly less states than $\mathcal{C}$', *c*) or $\mathcal{C}$ and $\mathcal{C}'$ are equivalent modulo a renaming of nonces, it is clear that $<$ is a well-founded ordering relation modulo this renaming.

Lemma 6 *Let S be a set of ASDs and $\mathcal{C}_h$ be a HSD. If $\min_<(S) \subseteq {\mathcal{C}_h}^\star$ then $S \subseteq {\mathcal{C}_h}^\star$.*

Complete sets of solutions. The ordering $<$ plays the same role *w.r.t.* the solutions of a HSD as the instantiation ordering on substitutions *w.r.t.* the solutions of an unification system. In particular the traditional notion of most general unifier is translated into a notion of minimal solution.

Definition 16 *(Complete set of solutions) A set Σ of ASDs is a* complete set of solutions *of an HSD $\mathcal{C}_h$ whenever:*

- $\Sigma \subseteq \mathcal{C}_h^\star$;
- *for every ASD $\mathcal{C}_\mathcal{I} \in \mathcal{C}_h^{\mathrm{sf}}$ there exists an ASD $\mathcal{C}_m \in \Sigma$ and a stutter free ASD $\mathcal{C}_c$ such that $\mathcal{C}_m \leq \mathcal{C}_\mathcal{I} \circ \mathcal{C}_c$.*

We have departed from our line of translating terms from the unification framework to the symbolic derivation framework by introducing a symbolic derivation $\mathcal{C}_c$. It permits us to consider cases in which the computation of a complete set of unifiers introduces unnecessary deduction steps in individual ASDs. A common example of such addition is the normalization of messages $\langle t, t' \rangle$, *i.e.* the automatic deduction of the two messages t and t' even when they are not useful for the attacker.

4.3 Finitary deduction systems

We have already noted that a NP decision procedure for the satisfiability of HSDs for the Dolev-Yao deduction system is known since [31]. While this procedure is based on the guessing of an attack of minimal size, other procedures have been proposed [5, 30] that instead cover all possible stutter-free derivations [16], *i.e.* compute a complete set of solutions. We define deduction systems for which such a procedure exists to be *finitary*.

Definition 17 *(Finitary Deduction Systems) Let $\mathcal{I}$ be a deduction system. If there exists a procedure that computes for every $\mathcal{I}$-HSD $\mathcal{C}_h$ a finite complete set of solutions we say that $\mathcal{I}$ is a* finitary deduction system.

5 Decidability of Symbolic Equivalence

This section is devoted to the proof of the main theorem of this paper.

Theorem 1 *Symbolic equivalence is decidable for finitary deduction systems.*

We first prove that every ASD can be written as the connection between a stutter-free ASD and a testing ASD in which no new term is deduced (Lemma 7). This implies the reduction of the inclusion problem to the one of checking whether, for any stutter-free ASD in $\mathcal{C}_h^\star$, the connections of this ASD with

$\mathcal{C}_h$ and $\mathcal{C}'_h$ result in *closed* symbolic derivations $\mathcal{C}_1$ and $\mathcal{C}_2$ such that $\mathcal{C}_1^\star \subseteq \mathcal{C}_2^\star$ (Lemma 9). Given a stutter-free ASD in $\mathcal{C}_h^\star$ this latter test is simple since it suffices to consider the connection with ASD that have at most one deduction (Prop. 2).

We relate these types of ASD with well-formed ASDs with the following lemma.

Lemma 7 *Let $\mathcal{C}_\mathcal{I}$ be a $(\mathcal{C}_h, \varphi)$-well-formed ASD. Then there exists a connection ψ, a well-formed deduction-only ASD $\mathcal{C}_d$, and a testing ASD $\mathcal{C}_t$ such that:*

- $\mathcal{C}_\mathcal{I} = \mathcal{C}_d \circ_\psi \mathcal{C}_t$,
- *for all HSD $\mathcal{C}'$ and connection ψ, the connection $\mathcal{C}' \circ_\psi \mathcal{C}_\mathcal{I}$ is closed if, and only if, $\mathcal{C}' \circ_\psi \mathcal{C}_d$ is closed.*

Lemma 8 *Let $\mathcal{C}_h, \mathcal{C}'_h$ be two HSDs such that $\mathcal{C}_h^\star \setminus \mathcal{C}_h'^\star \neq \emptyset$. Then $\mathcal{C}_h^\star \setminus \mathcal{C}_h'^\star$ contains a $(\mathcal{C}_h, \varphi)$-well-formed ASD.*

As a consequence, we obtain the following lemma that permits to split the symbolic equivalence problem into two simpler problems.

Lemma 9 *Let $\mathcal{C}_h$ and $\mathcal{C}'_h$ be two HSDs. We have $\mathcal{C}_h^\star \subseteq \mathcal{C}_h'^\star$ if, and only if:*

- $\mathcal{C}_h^{\mathrm{sf}} \subseteq \mathcal{C}_h'^\star$;
- *and for each ASD $\mathcal{C}_\mathcal{I} \in \mathcal{C}_h^{\mathrm{sf}}$ and for all testing ASD $\mathcal{C}_t \in (\mathcal{C}_\mathcal{I} \circ \mathcal{C}_h)^\star$ we have $\mathcal{C}_t \in (\mathcal{C}_\mathcal{I} \circ \mathcal{C}'_h)^\star$.*

Then we prove that if in the previous lemma the testing part is known, the stutter-free part is also a stutter-free solution of the connection between the testing part and the HSD.

Lemma 10 *Assume $\mathcal{C}_\mathcal{I} \in \mathcal{C}_h^{\mathrm{sf}}$ and $\mathcal{C}_t \in (\mathcal{C}_\mathcal{I} \circ \mathcal{C}_h)^\star$. Then $\mathcal{C}_\mathcal{I} \in (\mathcal{C}_t \circ \mathcal{C}_h)^{\mathrm{sf}}$.*

The next step is to bound the size of the testing ASD $\mathcal{C}_t$ obtained in Lemma 9. To this end, given an ASD $\mathcal{C}_\mathcal{I} \in \mathcal{C}_h^{\mathrm{sf}}$ we define:

$$\chi(\mathcal{C}_\mathcal{I}) = \{\mathcal{C}_t \text{ testing ASD} \,|\, \mathcal{C}_t \circ \mathcal{C}_\mathcal{I} \in \mathcal{C}_h^\star \setminus \mathcal{C}_h'^\star\}$$

i.e. the set of testing ASDs that distinguish $\mathcal{C}_h$ from $\mathcal{C}'_h$. By Lemma 9, $\mathcal{C}_h^\star \not\subseteq \mathcal{C}_h'^\star$ if, and only if, there exists an ASD $\mathcal{C}_\mathcal{I}$ such that $\chi(\mathcal{C}_\mathcal{I}) \neq \emptyset$. By ordering the equations in the unification system of an ASD $\mathcal{C}_t \in \chi(\mathcal{C}_\mathcal{I})$ and keeping a minimal one, we prove that an ASD of bounded length can be constructed from $\mathcal{C}_t$.

Proposition 2 *$\mathcal{C}_h^\star \not\subseteq \mathcal{C}_h'^\star$ if, and only if, there exists $\mathcal{C}_\mathcal{I} \in \mathcal{C}_h^{\mathrm{sf}}$ such that $\chi(\mathcal{C}_\mathcal{I})$ contains an ASD $\mathcal{C}_t$ with at most one deduction and one equality test.*

Now we simply gather the results from Lemma 10 and Proposition 2.

Proposition 3 *Given two HSDs $\mathcal{C}_h$ and $\mathcal{C}'_h$ we have $\mathcal{C}_h^\star \subseteq \mathcal{C}_h'^\star$ if, and only if, there exists a symbolic testing derivation $\mathcal{C}_t$ with at most one deduction state and one equality and a connection φ such that $(\mathcal{C}_h \circ_\varphi \mathcal{C}_t)^{\mathrm{sf}} \subseteq (\mathcal{C}'_h \circ_\varphi \mathcal{C}_t)^\star$.*

The proof of the following theorem depends on the fact that for finitary deduction systems, the set $\min_<((\mathcal{C}_t \circ \mathcal{C}_h)^{\mathrm{sf}})$ is by definition finite. The test of Proposition 3 thus becomes effective by Lemma 6 when a finite witness set is available.

Theorem 2 *(Inclusion of $\mathcal{C}_h^\star$ into $\mathcal{C}_h'^\star$) Let $\mathcal{D}$ be a finitary deduction system. The inclusion $\mathcal{C}_h^\star \subseteq \mathcal{C}_h'^\star$ is decidable for any two honest $\mathcal{D}$-symbolic derivations $\mathcal{C}_h, \mathcal{C}'_h$.*

As a trivial consequence we obtain the announced theorem.

Theorem 1, p. 19. *Symbolic equivalence is decidable for finitary deduction systems.*

6 Conclusion

We have introduced in this paper the notion of *finitary deduction systems*, and proved that symbolic equivalence is decidable for such attacker models. We believe that definition also captures the essence of *lazy intruder* techniques that are employed in many tools. Accordingly, we believe that a practical consequence of this paper will be the inclusion in existing reachability analysis tools of a symbolic equivalence checking algorithm.

In terms of comparison of expected runtimes for tools currently deciding reachability, a back-of-the-enveloppe computation for tools employing lazy constraint solving techniques such as OFMC [7] and CL-AtSe [36] would be twice (given that two protocols have to be analyzed and assuming tool is not parallelized) the runtime for safe (since these tools usually stop at the first attack found, and thus typically have a much shorter running time in these cases) protocols of a similar size. We refer the interested reader to [36] for more details, but given that CL-AtSe now implements a concurrent search algorithm and has been deployed on Amazon's EC2, we believe that less than 10s for reasonable industrial protocols is achievable nowadays.

References

[1] Robinson J. A. A machine-oriented logic based on the resolution principle. *J. Assoc. Comput. Mach.*, 12:23–41, 1965.

[2] Martín Abadi, Mathieu Baudet, and Bogdan Warinschi. Guessing attacks and the computational soundness of static equivalence. In Luca Aceto and Anna Ingólfsdóttir, editors, *FoSSaCS*, volume 3921 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 2006.

[3] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *ACM Conference on Computer and Communications Security*, pages 36–47, 1997.

[4] Martin Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *J. Cryptol.*, 20(3):395–395, 2007.

[5] Roberto M. Amadio and Denis Lugiez. On the reachability problem in cryptographic protocols. In Catuscia Palamidessi, editor, *CONCUR*, volume 1877 of *Lecture Notes in Computer Science*, pages 380–394. Springer, 2000.

[6] Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuéllar, and M. Llanos Tobarra. Formal analysis of saml 2.0 web browser single sign-on: breaking the saml-based single sign-on for Google Apps. In Vitaly Shmatikov, editor, *FMSE*, pages 1–10. ACM, 2008.

[7] David A. Basin, Sebastian Mödersheim, and Luca Viganò. Ofmc: A symbolic model checker for security protocols. *Int. J. Inf. Sec.*, 4(3):181–208, 2005.

[8] Mathieu Baudet. Deciding security of protocols against off-line guessing attacks. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *ACM Conference on Computer and Communications Security*, pages 16–25. ACM, 2005.

[9] Mathieu Baudet. *Sécurité des protocoles cryptographiques : aspects logiques et calculatoires*. Thèse de doctorat, Laboratoire Spécification et Vérification, ENS Cachan, France, January 2007.

[10] Bruno Blanchet. Automatic proof of strong secrecy for security protocols. In *IEEE Symposium on Security and Privacy*, pages 86–. IEEE Computer Society, 2004.

[11] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. In *LICS*, pages 331–340. IEEE Computer Society, 2005.

[12] Michele Boreale, Rocco De Nicola, and Rosario Pugliese. Proof techniques for cryptographic processes. In *LICS*, pages 157–166, 1999.

[13] Jan Camenisch, Sebastian Mödersheim, and Dieter Sommer. A formal model of identity mixer. In *FMICS'10*, LNCS 6371. Springer, 2010.

[14] Vincent Cheval, Hubert Comon-Lundh, and Stéphanie Delaune. Automating security analysis: symbolic equivalence of constraint systems. In Jürgen Giesl and Reiner Haehnle, editors, *Proceedings of the 5th International Joint Conference on Automated Reasoning (IJCAR'10)*, volume 6173 of *Lecture Notes in Artificial Intelligence*, pages 412–426, Edinburgh, Scotland, UK, July 2010. Springer-Verlag.

[15] Yannick Chevalier, Denis Lugiez, and Michaël Rusinowitch. Towards an automatic analysis of web service security. In Boris Konev and Frank Wolter, editors, *Frontiers of Combining Systems, 6th International Symposium, FroCoS 2007, Liverpool, UK, September 10-12, 2007, Proceedings*, volume 4720 of *Lecture Notes in Computer Science*, pages 133–147. Springer, 2007.

[16] Yannick Chevalier, Denis Lugiez, and Michaël Rusinowitch. Verifying cryptographic protocols with subterms constraints. In Nachum Dershowitz and Andrei Voronkov, editors, *LPAR*, volume 4790 of *Lecture Notes in Computer Science*, pages 181–195. Springer, 2007.

[17] Yannick Chevalier and Michaël Rusinowitch. Compiling and Securing Cryptographic Protocols. *Information Processing Letters*, (submitted), 2008.

[18] Yannick Chevalier and Michaël Rusinowitch. Decidability of equivalence of symbolic derivations. Submitted to the *Journal of Automated Reasoning*, 2009.

[19] Yannick Chevalier and Michaël Rusinowitch. Compiling and securing cryptographic protocols. *Inf. Process. Lett.*, 110(3):116–122, 2010.

[20] Hubert Comon-Lundh and Véronique Cortier. Computational soundness of observational equivalence. In *ACM Conference on Computer and Communications Security*, pages 109–118, 2008.

[21] Véronique Cortier and Stéphanie Delaune. A method for proving observational equivalence. In *Proceedings of the 22nd IEEE Computer Security Foundations Symposium (CSF'09)*, pages 266–276. IEEE Computer Society Press, 2009.

[22] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.

[23] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 243–320. Elsevier and MIT Press, 1990.

[24] D. Dolev and A. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 2(29), 1983.

[25] Jieh Hsiang and Michaël Rusinowitch. A new method for establishing refutational completeness in theorem proving. In Jörg H. Siekmann, editor, *CADE*, volume 230 of *Lecture Notes in Computer Science*, pages 141–152. Springer, 1986.

[26] Hans Hüttel. Deciding framed bisimilarity. Presented at the INFINITY'02 workshop, June 2002.

[27] Claude Kirchner, editor. *Unification*. Academic Press, 1986.

[28] Mounira Kourjieh. *Logical Analysis and Verification of Cryptographic Protocols.* Thèse de doctorat, Université Paul Sabatier, Toulouse, France, décembre 2009.

[29] Gavin Lowe. Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR. *Software - Concepts and Tools*, 17(3):93–102, 1996.

[30] Jonathan K. Millen and Vitaly Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *ACM Conference on Computer and Communications Security*, pages 166–175, 2001.

[31] Michaël Rusinowitch and Mathieu Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *CSFW*, pages 174–. IEEE Computer Society, 2001.

[32] S. Mödersheim and D. Sommer. A formal model of identity mixer. Technical report RZ 3749, IBM Research Zurich 2009.

[33] Manfred Schmidt-Schauß. Unification under associativity and idempotence is of type nullary. *J. Autom. Reasoning*, 2(3):277–281, 1986.

[34] Bruce Schneier. *Applied cryptography.* Addison-Wesley, 1996.

[35] Alwen Tiu and Jeremy E. Dawson. Automating open bisimulation checking for the spi calculus. In *CSF*, pages 307–321. IEEE Computer Society, 2010.

[36] Mathieu Turuani. The cl-atse protocol analyser. In Frank Pfenning, editor, *RTA*, volume 4098 of *Lecture Notes in Computer Science*, pages 277–286. Springer, 2006.