Information and Computation 238 (2014) 106–127

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Deducibility constraints and blind signatures

Sergiu Bursuc [a], Hubert Comon-Lundh [b], Stéphanie Delaune [b,*]

[a] Queen's University of Belfast, UK
[b] LSV, CNRS & ENS de Cachan & INRIA Saclay, France

ARTICLE INFO

Article history:
Received 18 February 2011
Available online 16 July 2014

Keywords:
Formal methods
Verification
Security protocol
Constraint system

ABSTRACT

Deducibility constraints represent in a symbolic way the infinite set of possible executions of a finite protocol. Solving a deducibility constraint amounts to finding all possible ways of filling the gaps in a proof. For finite *local* inference systems, there is an algorithm that reduces any deducibility constraint to a finite set of solved forms. This allows one to decide any trace security property of cryptographic protocols.

We investigate here the case of infinite local inference systems, through the case study of blind signatures. We show that, in this case again, any deducibility constraint can be reduced to finitely many solved forms (hence we can decide trace security properties). We sketch also another example to which the same method can be applied.

1. Introduction

This paper is concerned with the formal verification of security protocols. The formal models of security protocols are (infinite) transition systems, that are infinitely branching, because of the unbounded number of possible fake messages that can be sent by an attacker. In numerous cases, however, only finitely many such messages are relevant for mounting an attack. This is essentially what is proved in [19,20]: if we assume a fixed number of protocol sessions and the classical public key encryption and pairing primitives, then there is an attack if, and only if, there is an attack in which the messages sent by the attacker are taken out of a fixed set of messages. We refer to this result as *the small attack property*.

More practical algorithms, that do not need to enumerate all possible relevant attacker's behavior, rely on *deducibility constraints*, introduced in [17]. The idea is to represent symbolically all messages that can be forged by an attacker at a given stage of the protocol execution. An atomic deducibility constraint is an expression $T \vdash^{?} u$ where T is a finite set of terms and u is a term, both of which may contain variables. The deduction relation is interpreted according to the attacker capabilities and the variable instances correspond to the attacker choices of message forging. Deciding the satisfiability of such constraints then allows one to decide whether an attacker, after interacting with the protocol, may get a message that was supposed to remain secret.

These works have two limitations: they are restricted to some basic cryptographic primitives and they only consider the property of being able to get a supposedly secret message. They also consider some authentication properties, through an appropriate encoding. Concerning the first limitation, there are numerous extensions to other cryptographic primitives, for instance exclusive-or [11,6], modular exponentiation [7,21,18], any monoidal theory [13] and blind signatures [3]. Concerning the second limitation, the idea is to transform the deducibility constraints that have been mentioned above into finitely many solved forms, that represent in a convenient way all possible traces in presence of an active attacker. Then, whether a security property ϕ holds, can be checked by deciding the satisfiability of $\neg\phi$, together with each solved constraint. This

* Corresponding author.

http://dx.doi.org/10.1016/j.ic.2014.07.006
0890-5401/

is typically what is proposed in [10] (and also more recently in [22]), where it is shown how to decide trace properties, using the solved deducibility constraints, in case of public key and symmetric key encryption and signatures. This raises the problem of systematically designing deducibility constraint solving techniques, that would be applicable for both several primitives and any trace property.

In [4], we show that a locality property of the deduction system is sufficient for designing a deducibility constraint solving algorithm. Locality is a syntactic subformula property of normal proofs [16]: if there is a proof of t, then there is a proof of t whose every intermediate step is either a subterm of t or subterm of some hypothesis. Locality yields a tractable Entscheidungsproblem [16]. As shown in [2], this is also equivalent to a saturation property of the set of inference rules w.r.t. the subterm ordering. Therefore, if a set of inference rules modeling the attacker's capabilities on a given set of primitives can be saturated, yielding a finite set of inference rules, then, according to [4], we can simplify the deducibility constraints into finitely many solved forms and decide any trace security property. It turns out that some relevant proof systems cannot be finitely saturated w.r.t. the subterm ordering. This is the case of blind signatures, as modeled in [15]: saturating the inference rules does not terminate.

Yet, we show in this paper that we can extend the deducibility constraint solving procedures to some infinite local inference systems. Typically, such inference systems are obtained by saturating finite non-local inference systems. We consider the case study of blind signatures and briefly mention another example (homomorphic encryption) to which a very similar procedure works.

The basic idea for solving deducibility constraints is straightforward: given $T \stackrel{?}{\vdash} u$, guess the last inference rule that yields a proof of $u\sigma$ and decompose the constraint accordingly. This hardly terminates in general. In case of a local inference system, we roughly require that the new terms appearing in the constraint are either subterms of u or subterms of T, which, together with a simple strategy, guarantees termination. If the inference system is infinite, there are a priori infinitely many possible last rules that may yield $u\sigma$, which again raises a termination issue. In case of a relatively regular set of inference rules (which is the case for blind signatures and for homomorphic encryption), we may fold infinitely many such last steps in a single one, using an additional abstraction. This is what we show: we consider the case of blind signatures and add a predicate symbol, that allows us to consider all possible last deduction steps at once. We design then a constraint solving procedure, that includes this new predicate symbol, and show that it is terminating and yields solved forms. Trace security properties can be decided using these solved forms. As a witness, we mention the decidability of the first order formulas with equalities together with the deducibility constraints. Finally, we give another example of application of the same method. This second example witnesses the scope of the method, though we do not have a general class of primitives to which it could be applied.

Application to formal verification of security protocols In general, security protocols are specified in a process algebra like applied pi-calculus [1]. When one is interested in analysing a bounded number of sessions of the protocol, the set of all possible interleavings of actions is finite and can be determined from the specification. Each interleaving determines an infinite set of traces corresponding to all possible executions of the protocol in that context. As shown for instance in [12], this set of traces can be symbolically represented by a deducibility constraint system $\mathcal{C}$: every solution of $\mathcal{C}$ corresponds to a trace. Then, verifying a trace-based property of the given security protocol amounts to deciding whether for all solution of $\mathcal{C}$ the property is satisfied. That is why we consider deducibility constraint systems as our formal model of security protocols.

Furthermore, we restrict our attention to a particular intruder theory, modeling blind signatures. There are at least two important applications of blind signatures. In electronic payments, they allow individuals to provide proof of payment, without third parties being able to trace the payee, time or amount of payment [5]. In electronic voting protocols, they allow administrators to check the eligibility of voters and sign their ballots, without being able to determine how voters have voted [14]. A realistic model of protocols will in general have to consider more cryptographic primitives, including for example the classical Dolev–Yao theory of encryption and pairing. While we could include the Dolev–Yao theory in our model, we prefer not to do it because: 1) this does not raise any technical challenge for our procedure and 2) we think the procedure of [10] could be combined with ours in the more general framework of combining decision procedures for disjoint intruder theories [8].

2. Preliminaries

2.1. Term algebra

Messages are represented by terms, constructed on an infinite set of *names* $\mathcal{N} = \{a, n, k, \ldots\}$, an infinite set of *variables* $\mathcal{X} = \{x, y, \ldots\}$ and a set $\mathcal{F}$ of function symbols. In this paper, $\mathcal{F} = \{\mathsf{blind}, \mathsf{sign}, \mathsf{vk}\}$ together with arities $ar(\mathsf{blind}) = ar(\mathsf{sign}) = 2$ and $ar(\mathsf{vk}) = 1$. The term $\mathsf{sign}(m, sk)$ represents the message m signed by the private key sk. The function blind is supposed to hide a message, thus the term $\mathsf{blind}(m, r)$ represents the blinding of m with the random r. This allows one to request a signature without revealing the content of the message.

We write $vars(t)$ for the set of variables occurring in t and $st(t)$ is the set of subterms of t. The *size* of a term t, denoted $|t|$, is the number of symbols occurring in it. *Substitutions* are written $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ with $dom(\sigma) = \{x_1, \ldots, x_n\}$. The application of a substitution σ to a term t is written $t\sigma$. We denote by $\#T$ the *cardinal* of the set T.