Information and Computation ••• (••••) •••–•••

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

A procedure for deciding symbolic equivalence between sets of constraint systems ☆

Vincent Cheval [a], Hubert Comon-Lundh [b], Stéphanie Delaune [c,*]

[a] LORIA, Inria Nancy & CNRS & Université de Lorraine, France
[b] LSV, ENS Cachan & Université Paris Saclay, France
[c] CNRS & IRISA, France

ARTICLE INFO

Article history:
Received 11 November 2014
Received in revised form 15 September 2016
Available online xxxx

Keywords:
Formal methods
Verification
Security protocols
Privacy-type properties
Symbolic model

ABSTRACT

We consider security properties of cryptographic protocols that can be modelled using trace equivalence, a crucial notion when specifying privacy-type properties, like anonymity, vote-privacy, and unlinkability. Infinite sets of possible traces are symbolically represented using deducibility constraints. We describe an algorithm that decides trace equivalence for protocols that use standard primitives and that can be represented using such constraints. More precisely, we consider symbolic equivalence between *sets* of constraint systems, and we also consider disequations. Considering sets and disequations is actually crucial to decide trace equivalence for processes that may involve else branches and/or private channels (for a bounded number of sessions). Our algorithm for deciding symbolic equivalence between sets of constraint systems is implemented and performs well in practice. Unfortunately, it does not scale up well for deciding trace equivalence between processes. This is however the first implemented algorithm deciding trace equivalence on such a large class of processes.

1. Introduction

The present work is motivated by the decision of security properties of cryptographic protocols. Such protocols are proliferating, because of the expansion of digital communications and the increasing concern on security issues. Finding attacks/proving the security of such protocols is challenging and has a strong societal impact.

In our work, we assume perfect cryptographic primitives: we consider a formal, symbolic, model of execution. Such an assumption may prevent from finding some attacks; the relevance of symbolic models is studied in other research papers (see *e.g.,* [1,2]), but it is beyond the scope of the present work.

In this context, the protocols are described in some process algebra, using function symbols to represent the cryptographic primitives and symbolic terms to represent messages. We use the applied pi-calculus [3] in this paper. Many attacks on several protocols have been found during the last 20 years. For example, a flaw has been discovered (see [4]) in the Single-Sign-On protocol used *e.g.,* by Google Apps. These attacks on formal models of protocols can of course be reproduced

☆ The research leading to these results has received funding from the ANR project Sequoia ANR-14-CE28-0030-01.
* Corresponding author.
E-mail address: stephanie.delaune@irisa.fr (S. Delaune).

http://dx.doi.org/10.1016/j.ic.2017.05.004
0890-5401/

on the concrete versions of the protocols. Several techniques and tools have been designed for the formal verification of cryptographic protocols. For instance CSP/FDR [5], PROVERIF [6], SCYTHER [7], AVISPA [8] and others.

Most results and tools only consider security properties that can be expressed as the (un)reachability of some bad state. For instance, the (weak) secrecy of s is the non-reachability of a state, in which s is known by the attacker. Authentication is also expressed as the impossibility to reach a state, in which two honest parties hold different values for a variable on which they are supposed to agree. In our work, we are interested in more general properties, typically strong secrecy, anonymity, or more generally any privacy-type property that cannot be expressed as the (non) reachability of a given state, but rather requires the *indistinguishability* of two processes. For instance, the strong secrecy of s is specified as the indistinguishability of $P(s)$ from $P(s')$, where s' is a new name. It expresses that the attacker cannot learn any piece of the secret s. Formally, these properties, as well as many other interesting security properties, can be expressed using *trace equivalence*: roughly, two processes P and Q are trace equivalent if any sequence of attacker's actions yields indistinguishable outputs of P and Q.

Some related work. The automated verification of equivalence properties for security protocols was first considered in [9] (within the spi-calculus). PROVERIF also checks some equivalence properties (so-called diff-equivalence) [10], which is a stronger equivalence, often too strong, as we will see below with a simple example. More recently, the approach behind the TAMARIN verification tool [11] has been extended to check equivalence-based properties [12]. Actually, the equivalence notion is quite similar to the notion of diff-equivalence used in ProVerif, and therefore suffers from the same drawbacks. A few other procedures have been published:

- In [13,14] a decision procedure for the trace equivalence of bounded deterministic processes is proposed. Their procedure relies on an other procedure for deciding the equivalence of constraint systems such as the one developed by [15] or [16]. In particular, the processes are restricted to be determinate and do not contain (non-trivial) conditional branching. Furthermore, the procedure seems to be not well-suited for an implementation. Regarding primitives, these works allow any primitives that are defined using a subterm convergent rewriting system.
- [17] gives a decision procedure for open-bisimulation for bounded processes in the spi-calculus. This procedure has been implemented. The scope is however limited: open-bisimulation is a stronger equivalence notion, and the procedure assumes a fixed set of primitives (in particular no asymmetric encryption) and no conditional branching.
- [18] designs a procedure based on Horn clauses for the class of optimally reducing theories, which encompasses subterm convergent theories. The procedure is sound and complete but its termination is not guaranteed. It applies to determinate processes without replication nor else branches. Moreover, when processes are not determinate, the procedure can be used for both under- and over-approximations of trace equivalence.

Our contribution. Our aim was to design a procedure, which is general enough and efficient enough, so as to automatically verify the security of some simple protocols, such as the private authentication protocol (see Example 1) or the e-passport protocol analysed *e.g.,* in [19]. Both protocols are beyond the scope of any above mentioned results. An extension of PROVERIF has been developed allowing one to analyse the private authentication protocol [20]. However, PROVERIF is still unable for instance to deal with the e-passport protocol.

Example 1. We consider the protocol given in [21] designed for transmitting a secret, while not disclosing the identity of the sender. In this protocol, a is willing to engage in a communication with b. However, a does not want to disclose her identity (nor the identity of b) to the outside world. Consider for instance the following protocol:

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\langle n_a, \mathsf{pub}(ska)\rangle, \mathsf{pub}(skb)) \\ B \to A & : & \mathsf{aenc}(\langle n_a, \langle n_b, \mathsf{pub}(skb)\rangle\rangle, \mathsf{pub}(ska)) \end{array}$$

In words, the agent a (playing the role A) generates a new name n_a and sends it, together with her identity (here public key), encrypted with the public key of b. The agent b (playing the role B) replies by generating a new name n_b, sending it, together with n_a and his identity $\mathsf{pub}(skb)$, encrypted with the public key of a. More formally, using pattern-matching, and assuming that each agent a holds a private key ska and a public key $\mathsf{pub}(ska)$, which is publicly available, the protocol could be written as follows:

$$\mathsf{PrivAuth1} \quad \left\{ \begin{array}{ll} A(ska, pkb): & \nu\, n_a.\, \mathsf{out}(\mathsf{aenc}(\langle n_a, \mathsf{pub}(ska)\rangle, pkb)) \\ B(skb, pka): & \mathsf{in}(\mathsf{aenc}(\langle x, pka\rangle, \mathsf{pub}(skb))).\\ & \quad \nu\, n_b.\, \mathsf{out}(\mathsf{aenc}(\langle x, \langle n_b, \mathsf{pub}(skb)\rangle\rangle, pka)) \end{array} \right.$$

We will later write $A(a,b)$ for $A(ska, \mathsf{pub}(skb))$,$B(b,a)$ for $B(skb, \mathsf{pub}(ska))$, and $B(b,c)$ for $B(skb, \mathsf{pub}(skc))$.

This is fine, as long as only mutual authentication is concerned. Now, if we want to ensure in addition privacy, an attacker should not get any information on who is trying to set up the agreement: $B(b,a)$ and $B(b,c)$ must be indistinguishable. This is not the case in the above protocol. Indeed, an attacker can forge *e.g.,* the message $\mathsf{aenc}(\langle \mathsf{pub}(ska), \mathsf{pub}(ska)\rangle, \mathsf{pub}(skb))$ and find out whether $c = a$ or not by observing whether b replies or not.

The solution proposed in [21] consists in modifying the process B in such a way that a "decoy" message: $\mathsf{aenc}(\langle n_b, n_b\rangle, \mathsf{pub}(ska))$ is sent when the received message is not as expected. This message should look like B's other message from the point of view of an outsider. More formally, this can be modelled using the following process:

Please cite this article in press as: V. Cheval et al., A procedure for deciding symbolic equivalence between sets of constraint systems, Inf. Comput. (2017), http://dx.doi.org/10.1016/j.ic.2017.05.004