Information and Computation 239 (2014) 182–215

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Bounding messages for free in security protocols – extension to various security properties

Myrto Arapinis [a], Marie Duflot [b,*]

[a] University of Edinburgh, UK
[b] LORIA & Université de Lorraine, France

ARTICLE INFO

Article history:
Received 5 March 2013
Received in revised form 23 July 2014
Available online 2 October 2014

Keywords:
Cryptographic protocols
Formal methods
Verification
Secrecy
Authentication

ABSTRACT

While the verification of security protocols has been proved to be undecidable in general, several approaches use simplifying hypotheses in order to obtain decidability for interesting subclasses. Amongst the most common is type abstraction, *i.e.* considering only well-typed runs of the protocol, therefore bounding message length. In this paper, we show how to get message boundedness "for free" under a reasonable (syntactic) assumption on protocols, in order to verify a variety of interesting security properties including secrecy and several authentication properties. This enables us to improve existing decidability results by restricting the search space for attacks.

1. Introduction

Security protocols are short programs that describe communication between two or more parties in order to achieve security goals such as data confidentiality or identification of a correspondent for example. These protocols are executed in a hostile environment, such as the Internet, and aim at preventing a malicious agent from tampering with the messages, for instance, using encryption. However, encrypting messages is not sufficient to enforce security or privacy guarantees. History has shown that these protocols are extremely error-prone and careful formal verification is needed.

Despite the apparent simplicity of such protocols, their verification is a difficult problem and has been proved undecidable in general [21,7]. Different attempts [21,25,32,8,30,31,6] have successfully exhibited decidable subclasses, and tools for proving security have been designed [27,13,4,15,34]. However, termination of these tools is often not guaranteed. In the literature, only very few results consider an unbounded number of sessions and even fewer allow unbounded creation of new nonces (constants generated to ensure freshness of messages). Indeed, most existing decidability results were obtained either under the abstraction that only a bounded number of nonces are ever created, or considering only a bounded number of sessions of the protocol (see the recapitulative tables pages 6 and 10 in [10]).

In order to obtain decidability, many existing results bound the message length in adopting a typing abstraction [25,30,21,14], an assumption according to which one can always tell the type of a given message. While this appears as an unrealistic assumption in the general case, this paper presents a simple way of justifying it. This question has been addressed in [26,22] amongst others, and solved with tagging schemes.

In this paper, we show that when a protocol satisfies a reasonable syntactic condition based on non-unifiability of subterms of different types – and belongs to a class we will denote $\mathbb{P}$, then the typing abstraction is correct for several

* Corresponding author.
E-mail addresses: marapini@inf.ed.ac.uk (M. Arapinis), marie.duflot-kremer@loria.fr (M. Duflot).

http://dx.doi.org/10.1016/j.ic.2014.09.003
0890-5401/

security properties, including for example secrecy and several forms of authentication. Furthermore, the considered typing system here is much more fine-grained than the one considered in [26,22], thus refining existing results by reducing the search space to be handled. Indeed, we prove that a protocol in our class $\mathbb{P}$ admits an attack if and only if it admits a "well-typed" attack with respect to a stronger notion of typing.

Our class of protocols characterizes protocols in which an encrypted term cannot be mistaken for another, unless they are of the same type. This notion is often satisfied in protocols found in the literature (see [12]) and, even when the protocol is not well-formed, a light tagging like the one proposed in [6] permits one to comply with the property and thus use our result.

This paper is a revised and extended version of [2]. Unlike the previous one, this work tackles a larger class of security properties. They are now defined using the logic $\mathcal{PS}$-LTL and characterized by a newly defined class that includes not only secrecy but other interesting security properties related to authentication and fairness. We also present extensive proofs of the results.

The rest of the paper is organized as follows. Section 2 presents the formalism used throughout the paper, both for the description of the content of a message, and for the capabilities of the intruder to decrypt messages and forge new ones. Section 3 gives the model we consider for security protocols. In Section 4 we present the logic used to describe the security properties to verify. We also explain how to express, using this logic, several well known security properties. Section 5 is dedicated to the statement, first informally and then in more details of the main result of this paper, introducing in particular the typing system considered.Then the formal proof together with the tools needed to write it can be found in Section 6. Section 7 presents applications of this result, as well as a discussion on the (un)decidability of the secrecy problem for the class of protocols $\mathbb{P}$. More precisely, we show that the number of sessions to consider to find an attack on a protocol in $\mathbb{P}$ cannot be bounded in advance by a constant. We finally conclude in Section 8. In order to help the reader keeping track of all notations and technical terms used in the paper, an index can be found just after the bibliography.

Related work Our work presents a reduction result of the search space to well-typed runs for the class of protocols satisfying the criterion of "non-unifiability of sub-terms" stated in [3]. Many works take this reduction result as an assumption to prove decidability results, such as [25,30,21,14,23]. However, such an abstraction is not correct in general (this can be seen by the attack present in the protocol Π_{Toy} that we use as running example here).

The closest work to ours is the one of J. Heather *et al.* [22], where the authors also present a reduction result of the search space to well-typed runs. Indeed, they show that it is possible to label the terms of a protocol with static tags such that the typing assumption is correct. However, their result does not apply to protocols involving blind copies, *i.e.* variables with complex types, nor complex keys. It doesn't either apply to protocols that do not use static tags. Our result is more general as it applies to protocols that satisfy non-unifiability of subterms of different types, but not necessarily through the use of static tags. Moreover, the reduction result presented in [22] is more coarse grained, as all nonces have the same type.

2. Messages and intruder capabilities

This section is dedicated first to the presentation of the formalism used to describe the considered cryptographic primitives and messages. We then precisely characterize the intruder capabilities, that is how the intruder can decompose messages received in order to gain information, as well as how he can construct new messages in order to fool the other participants.

2.1. Messages

We use an abstract term algebra to model the messages of a protocol. For this we fix several disjoint sets. We consider an infinite set of *agents* $\mathcal{A} = \{\epsilon, a, b \ldots\}$ with the special agent ϵ standing for the attacker, and an infinite set of *agent variables* $\mathcal{Z} = \{z_A, z_B, \ldots\}$. Agent variables in $\mathcal{Z}$ will denote the roles of the protocol. We also need to assume an infinite set of *honest names* $\mathcal{N} = \{n, m \ldots\}$ to model atomic data such as nonces, an infinite set of *dishonest names* $\mathcal{N}^\epsilon = \{m^\epsilon, n^\epsilon, n^{\epsilon,1}, n^{\epsilon,2} \ldots\}$ standing for names generated by the attacker, and an infinite set of *term variables* $\mathcal{X} = \{x, y, x_1, x_2, \ldots\}$. In what follows, we will often refer to term variables simply as variables, in order to avoid unnecessary and cumbersome repetitions. We consider an infinite set of constants $\mathcal{C} = \{c, d, c_1, c_2, \ldots\}$ and the following *signature* $\mathcal{F} = \{\mathsf{senc}/2, \mathsf{aenc}/2, \mathsf{sign}/2, \langle\rangle/2, \mathsf{h}/1, \mathsf{pv}/1, \mathsf{sh}/2\}$. These function symbols model cryptographic primitives. The symbol $\langle\rangle$ represents pairing. The term $\mathsf{senc}(m, k)$ (*resp.* $\mathsf{aenc}(m, k)$) represents the message m encrypted with the symmetric (resp. asymmetric) key k whereas the term $\mathsf{sign}(m, k)$ represents the message m signed by the key k. The function h models a hash function whereas $\mathsf{pv}(a)$ is used to model the private key of agent a, and $\mathsf{sh}(a, b)$ is used to model the long-term symmetric key shared by agents a and b. In order to avoid reasoning modulo the equational theory $\mathsf{sh}(z_1, z_2) = \mathsf{sh}(z_2, z_1)$, we equip the set $\mathcal{A}$ with a total order denoted $<_{\mathcal{A}}$, and will only allow keys $\mathsf{sh}(a, b)$ with $a <_{\mathcal{A}} b$. We assume ϵ to be such that for all $a \in \mathcal{A}$, $\epsilon <_{\mathcal{A}} a$. The set of *terms* $\mathcal{T}$ is defined inductively by the following grammar: