Analysis of LySA -calculus with explicit confidentiality annotations

Han Gao, Hanne Riis Nielson
Informatics and Mathematical Modelling, Technical University of Denmark
Richard Petersens Plads bldg 322, DK2800 Kgs. Lyngby, Denmark
-{hg,riis}@imm.dtu.dk

September 29, 2005

Abstract

In modern society, communication protocols are frequently used for significant tasks such as banking, shopping, and for personal communications. With the massive increase in use of communication protocol, it is demanding that the security of protocols must be ensured, meaning that protocol should be able to guard against malicious behavior and meanwhile guarantee the necessary amount of confidentiality, authenticity, message integrity and availability.

Protocol analysis is a hard problem for several reasons. One is that protocols are often described somewhat informally using protocol narrations that are imprecise about some of the finer details concerning the deployment of the protocol. Aiming this problem, several process calculi were developed for precisely describing protocols, i.e. π, Spi and Applied-π calculus, which greatly facilitate analyzing protocols.

In [1], a new process calculus, called LySA , was proposed and annotations were used to assist validating authentication property. Moving further along this way, we add explicit annotations to LySA for handling confidentiality property of communication protocols.

LySA is a relatively new process algebra, which is developed based on π- and Spi-calculus. It inherits most of the features of its predecessors, while still keeps its own characteristic mainly in two aspects. One is the absence of channels: in LySA all processes have only access to one single global communication channel. The second characteristic is that tests associated with input and decryption are expressed in pattern matching.

LySA consists of terms and processes. The syntax of terms E and processes P are as follows:

$E ::=$	*terms*
$n^l[within\ \mathcal{L}]$	name ($n \in \mathcal{N}$)
x^l	variable ($x \in \mathcal{X}$)
$\{E_1, \ldots, E_k\}^l_{E_0}[within\ \mathcal{L}]$	symmetric encryption ($k \geq 0$)

For each name and encryption, we add to it a label, $l \in Lab$ which serves as an unique identity, hence avoid using canonical name.

$P ::=$	*processes*
0	nil
$P_1 \mid P_2$	parallel composition
$(\nu\ n^l[within\ \mathcal{L}])P$	restriction
$!P$	replication
$decrypt\ E\ as \{E_1, \ldots, E_j; x_{j+1}^{l_{j+1}}[from\ \mathcal{L}_{j+1}], \ldots, x_k^{l_k}[from\ \mathcal{L}_k]\}_{E_0}\ in\ P$	symmetric decryption (with matching)
$(E_1, \ldots, E_j; x_{j+1}^{l_{j+1}}[from\ \mathcal{L}_{j+1}], \ldots, x_k^{l_k}[from\ \mathcal{L}_k]).P$	input (with matching)
$\langle E_1, \ldots, E_k \rangle.P$	output

In addition to the classical constructs for composing processes, LySA contains an input construct with matching and one decryption operation with matching. The idea behind the pattern matching in LySA is that whenever matching a k-tuple of values we allow the match on a prefix of j $(0 \leq j \leq k)$ values and then to bind the remaining $k - j$ values to variables. To describe the intentions of protocols in LySA , whenever a name or an encryption is used we add to it the annotation $[within\ \mathcal{L}]$, where $\mathcal{L} \subseteq Lab$ is a set of labels pointing to the variables which they may be bound to and at each binding occurrence we add the set of labels pointing to the values that may be bound to the variable.
Having explicitly specified the intention of the protocol, we then develop a control flow analysis aiming at giving a safe over-approximation of all possible behaviors of target protocols, which include the sets of values which may be bound to each variables and the set of messages that are successfully being received at each relevant point. Meanwhile, the analysis will record all the possible $within/from$ violations, which indicate the involvement of a potential attacker. Due to the over-approximative nature of our control flow analysis, it is clear that the analysis might catch a too large set of messages but no successfully received messages are left out.

References

[1] C. Bodei, M. Buchholtz, P. Degano, F. Nielson, H. Riis Nielson, Automatic Validation of Protocol Narration, the 16th Computer Security Foundations Workshop (CSFW 03)., pp. 126-140, IEEE Computer Society Press, 2003.