Protocol Indistinguishability and the Computationally Complete Symbolic Attacker (Extended Abstract)

Gergei Bana
INRIA
Paris, France
bana@math.upenn.edu

Hubert Comon-Lundh
Laboratoire Spécification et Vérification
École Normale Supérieure de Cachan
Cachan, France
comon@lsv.ens-cachan.fr

Recently, we proposed a technique [1] to define a computationally complete symbolic attacker for the verification of security protocols. Such a symbolic attacker can do everything computational attackers can, and hence it is possible to avoid the numerous usual restrictions computational soundness theorems require; in other words, unconditional soundness can be achieved. The main idea is that instead of listing all capabilities of the symbolic attacker (as the Dolev-Yao technique does), we list all properties (called axioms) the attacker is not allowed to violate. The attacker wins if the negation of the security property of the protocol is consistent with the axioms and agent checks of the protocol. In followup works (e.g. [2]), we created a small library of axioms, and illustrated that the technique is suitable for proving actual protocols. Construction of an automated tool is also under way [5]. However, these earlier works considered reachability properties only. In this work we make the technique suitable for proving equivalence properties as well. We discuss some of the difficulties that made this step non-trivial and how we solved them. We also present an example protocol and its verification with this technique.

In the Dolev-Yao approach, indistinguishability of protocols is defined via trace equivalence. Namely, two protocols are equivalent if the execution traces of one of the protocols are in correspondence with the execution traces of the other protocol (interacting with the same adversary) such that the corresponding traces are statically equivalent. It is easy to see however that this delivers attacks that computationally do not exist even in very simple situations. Consider for example the two processes

$\nu n'.\, \mathsf{out}(n')$
$\nu b.\nu n.\mathsf{if}\ b = 0\ \mathsf{then}\ \mathsf{out}(1 \cdot n)\ \mathsf{else}\ \mathsf{out}(0 \cdot n)$

where n is drawn uniformly at random in $\{0,1\}^\eta$, n' is drawn uniformly at random in $\{0,1\}^{\eta+1}$ and b is drawn uniformly at random in $\{0,1\}$. That is, the first process generates a nonce in $\{0,1\}^{\eta+1}$ and outputs it. The second generates a bit b and a nonce in $\{0,1\}^\eta$, checks the value of the bit b, then outputs its opposite concatenated with n. Both processes are indistinguishable, since they output a random bitstring of length $\eta + 1$. Symbolically, the first process has only one branch, while the second has two branches, all three inequivalent, they cannot be matched. The first idea would be to split the branch of the first process based on the value of the first bit. But it is unclear how to perform such a splitting automatically in general.

Instead of trying to match the execution branches in the two protocols, we fold the protocols, including the control structure into the message terms so that each protocol has only one trace. This trick is inspired by [3]. The branches of the process are merged such that at each step, instead of numerous possible outputs, a single term is produced, which (instead of the control structure of the original process) contains all the conditional branching leading to that point. For instance, the second process above is folded into

$$\nu b.\nu n.\mathsf{out}(\mathsf{if}\ b = 0\ \mathsf{then}\ 1 \cdot n\ \mathsf{else}\ 0 \cdot n\ )$$

introducing $\mathsf{if\_then\_else\_}$ as a function symbol on terms. This way, we are left to check the equivalence on terms:

$$n' \sim \mathsf{if}\ b = 0\ \mathsf{then}\ 1 \cdot n\ \mathsf{else}\ 0 \cdot n$$

where $\sim$ is indistinguishability (of terms), the single predicate we introduce in our language.

While axiomatizing computational security notions such as CPA in this framework turned out to be rather straightforward, it was more challenging to introduce convenient and computationally sound axioms for the relationship of $\sim$ and the function symbol $\mathsf{if\_then\_else\_}$. We present some of our axioms, and show how our technique works by verifying (but first correcting) the simple private authentication protocol from [4] on which these basic difficulties can be illustrated.

References

[1] G. Bana and H. Comon-Lundh. Towards unconditional soundness: Computationally complete symbolic attacker. In *POST'12*, LNCS, pages 189–208. Springer, 2012.

[2] G. Bana, K. Hasebe, and M. Okada. Computationally complete symbolic attacker and key exchange. In *CCS '13*, pages 1231–1246. ACM, 2013.

[3] V. Cheval and B. Blanchet. Proving more observational equivalences with proverif. In *POST'13*, Lecture Notes in Computer Science, pages 226–246. Springer, 2013.

[4] Vincent Cheval, Hubert Comon-Lundh, and Stéphanie Delaune. Trace equivalence decision: Negative tests and non-determinism. In *CCS'11*, pages 321–330. ACM Press, 2011.

[5] H. Comon-Lundh, V. Cortier, and G. Scerri. Tractable inference systems: an extension with a deducibility predicate. In *CADE'13*, LNAI. Springer, 2013.