Using Interpolation for the Verification of Security Protocols (Extended Abstract)

Marco Rocchetto[1], Luca Viganò[2], and Marco Volpe[1]

[1] Dipartimento di Informatica, Università di Verona, IT
[2] Department of Informatics, King's college London, UK

Interpolation has been successfully applied in formal methods for model checking and test-case generation for sequential programs. Security protocols, however, exhibit such idiosyncrasies that make them unsuitable to the direct application of such methods, most notably, the fact that, in the presence of a Dolev-Yao intruder, a security protocol is not a sequential program (since the intruder can freely interleave his actions with the normal protocol execution).

We have addressed this problem to develop an interpolation-based method for security protocol verification. Our method (thoroughly described in [2]) starts from a formal protocol specification, along with the specification of a security property (e.g., authentication or secrecy) and a finite number of protocol sessions. It creates a corresponding sequential non-deterministic program, where parameters are introduced in order to capture the behavior of an intruder, in the form of a control flow graph, and then defines a set of goals (representing possible attacks on the protocol) and searches for them by symbolically executing the program. When a goal is reached, an attack trace is extracted from the set of constraints that the execution of the path has produced; such constraints represent conditions over parameters that allow one to reconstruct the attack trace(s) found. When the search fails to reach a goal along a given path, a backtrack phase starts, during which the nodes of the graph are annotated (according to the IntraLA algorithm defined by McMillan [1] for sequential programs, which we adapted and slightly modified by omitting unnecessary parameters) with formulas obtained by using Craig interpolation. Such formulas express conditions over the program variables, which, when implied from the program state of a given execution, ensure that no goal will be reached by going forward. The annotations are thus used to guide the search: we can discard a branch when it leads to a state where the annotation of the corresponding node is implied. Summing up, the output of the method is a proof of (bounded) verification in the case when no goal location can be reached starting from a finite-state specification; otherwise, one or more attack traces are produced.

We have implemented our method (where we use the solver Z3 to model the deductive power of the intruder and check satisfiability, and its extension iZ3 to generate interpolants) and we have applied it to a number of concrete examples, e.g., the protocols NSPK, NSL, EKE. The first results are very promising, as the experiments show that the interpolation-based annotations allow for pruning the graph on which the search for goals is performed. The optimization seems to be especially efficient when graphs have a large size and contain nodes with a great number of incoming edges. For this reason, we are currently modeling more complex protocols, where we expect the advantages of our method to be more evident.

As future work, we will consider the automated extraction, from discovered attack traces, of test cases to be applied to security protocol implementations.

References

[1] McMillan, K. L.: Lazy annotation for program testing and verification. In *CAV'10*, LNCS 6174:104–118. Springer, 2010.

[2] Rocchetto, M., Viganò, L., Volpe, M., Dalle Vedove, G.: Using interpolation for the verification of security protocols. In *STM'13*, LNCS 8203:99–114. Springer, 2013.