Verification of Security Protocols*

Véronique Cortier
LORIA, CNRS, Nancy, France

Security protocols are short programs aiming at securing communications over a network. They are widely used in our everyday life. They may achieve various goals depending on the application: confidentiality, authenticity, privacy, anonymity, fairness, etc. Their verification using symbolic models has shown its interest for detecting attacks and proving security properties. A famous example is the Needham-Schroeder protocol [7] on which G. Lowe discovered a flaw 17 years after its publication [5]. Secrecy preservation has been proved to be co-NP-complete for a bounded number of sessions [8], and decidable for an unbounded number of sessions under some additional restrictions (*e.g.* [1, 3, 4, 9]). Many tools have also been developed to automatically verify cryptographic protocols like [6, 2].

In this tutorial, we first overview several techniques used for symbolically verifying security protocols that have led to the design of many efficient and useful tools. Various formal models have been proposed for representing security protocols. They all have in common that messages are represented by terms, preserving the structure of messages but abstracting away all implementations details of the functions such as encryption, signatures or Exclusive Or. We will see how the analysis of security protocols then reduces to solving constraint systems or resolving (fragment of) Horn clauses.

However, the guarantees that symbolic approaches offer have been quite unclear compared to the computational approach that considers issues of complexity and probability. This later approach captures a strong notion of security, guaranteed against all probabilistic polynomial-time attacks. In a second part of the tutorial, we present recent results that aim at obtaining the best of both worlds: fully automated proofs and strong, clear security guarantees. The approach consists in proving that symbolic models are *sound* with respect to computational ones, that is, that any potential attack is indeed captured at the symbolic level.

References

[1] R. Amadio and W. Charatonik. On name generation and set-based analysis in the dolev-yao model. In *Proc. of the 13th International Conference on Concurrency Theory (CONCUR'02)*, LNCS, pages 499–514. Springer Verlag, 2002.

*This work has been partially supported by the ANR-07-SESU-002 project AVOTÉ.

[2] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proc. of the 14th Computer Security Foundations Workshop (CSFW'01)*. IEEE Computer Society Press, June 2001.

[3] B. Blanchet and A. Podelski. Verification of cryptographic protocols: Tagging enforces termination. In A. Gordon, editor, *Foundations of Software Science and Computation Structures (FoSSaCS'03)*, volume 2620 of *LNCS*, April 2003.

[4] H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In *Proc. of the 14th Int. Conf. on Rewriting Techniques and Applications (RTA'2003)*, volume 2706 of *LNCS*, pages 148–164. Springer-Verlag, 2003.

[5] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In T. Margaria and B. Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, volume 1055 of *LNCS*, pages 147–166. Springer-Verlag, march 1996.

[6] G. Lowe. Casper: A compiler for the analysis of security protocols. In *Proc. of 10th Computer Security Foundations Workshop (CSFW'97)*, Rockport, Massachusetts, USA, 1997. IEEE Computer Society Press. Also in Journal of Computer Security, Volume 6, pages 53-84, 1998.

[7] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communication of the ACM*, 21(12):993–999, 1978.

[8] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions and composed keys is NP-complete. *Theoretical Computer Science*, 299:451–475, 2003.

[9] H. Seidl and K. N. Verma. Flat and one-variable clauses: Complexity of verifying cryptographic protocols with single blind copying. In *Logic for Programming and Automated Reasoning (LPAR'04)*, LNCS, pages 79–94. Springer-Verlag, 2005.