

The CASPA Tool: Causality-based Abstraction for Security Protocol Analysis (Tool Paper)

Michael Backes^{1,2}, Stefan Lorenz¹, Matteo Maffei¹, and Kim Pecina¹

¹ Saarland University, Saarbrücken, Germany

² MPI-SWS

Abstract. CASPA constitutes a push-button tool for automatically proving secrecy and authenticity properties of cryptographic protocols. The tool is grounded on a novel technique for causality-based abstraction of protocol executions that allows establishing proofs of security for an unbounded number of concurrent protocol executions in an automated manner. We demonstrate the expressiveness and efficiency of the tool by drawing a comparison with T4ASP, the static analyzer for secrecy properties offered by the AVISPA tool. CASPA is capable of coping with a substantially larger set of protocols, and excels in performance.

1 Introduction

Proofs of security protocols are known to be error-prone and, owing to the distributed-system aspects of multiple interleaved protocol runs, awkward to do. In fact, vulnerabilities have accompanied the design of such protocols ever since early authentication protocols like the Needham-Schroeder protocol, to carefully designed de-facto standards like SSL and PKCS, up to current widely deployed products like Microsoft Passport. Formal methods have proved to be salient tools for dealing with such flaws, by helping both to securely design and to analyze security protocols, and even to formally establish their security properties.

A central intricacy that these tools have to tackle is to concisely treat the potentially very large number of concurrent protocol executions. We have developed CASPA (Causality-based Abstraction for Security Protocol Analysis), a tool for establishing formal security proofs of cryptographic protocols for an unbounded number of concurrent protocol executions in a mechanized manner. The tool is grounded on a recently proposed abstract interpretation of cryptographic protocols [3] based on causal graphs. Causal graphs are finite dependency graphs in which nodes represent process events and edges express the causality among events. These graphs constitute a sound abstraction of an unbounded number of protocol executions and, interestingly, they serve as a graphical illustration of the actual protocol behavior. A quick inspection of these graphs often suffices to identify unintended, and possibly harmful, interactions among parties. This facilitates protocol design and error detection even on the human level.

Related Work We demonstrate the expressiveness and efficiency of our tool by drawing a comparison with T4ASP [13,9], the static analyzer for secrecy properties offered by AVISPA [2], the well-known tool suite for security protocol analysis. CASPA is capable of coping with a substantially larger set of protocols than TA4SP, and it furthermore excels in terms of performance; moreover, CASPA is capable of verifying both secrecy and authenticity properties in contrast to only secrecy properties in the case of TA4SP. Note that TA4SP so far constitutes the only tool of the AVISPA tool suite that is capable of producing proofs of security; the remaining tools such as OFMC [5] rely on state-space exploration techniques and are constraint to either taking into account only a limited number of concurrent protocol executions, typically two or three, or giving up guaranteed termination in general. The same holds for other techniques based on state-space exploration such as Athena [16], CPSA [12], and Scyther [11]. ProVerif [6] is a tool based on Horn-clause resolution that verifies trace-based security properties, such as secrecy and authenticity, as well as observational equivalence of process behavior, is very efficient, but provides guaranteed termination only for tagged protocols [7]. The Lysa tool [8] implements a control-flow analysis that offers termination and security proofs for an unbounded number of protocol executions, but does not support security properties based on correspondence assertions [17].

2 The CASPA Tool

CASPA is written in Objective CAML and equipped with a graphical user interface, assisting the user in the specification and in the analysis of cryptographic protocols, see Figure 1. CASPA provides an *editor* for protocol specifications, offering a quick loading procedure for the protocols specified in underlying protocol libraries, and a convenient parsing procedure for user-defined protocol specifications. In addition, the tool features a *graph management system* that automatically generates and displays causal graphs.

Finally, CASPA offers a fully mechanized *analyzer* that verifies secrecy and authenticity properties on a given causal graph and displays the results. More precisely, CASPA allows for analyzing the security properties *secrecy*, *weak authenticity*, and *strong authenticity* [14]: as in AVISPA, authenticity properties are specified in terms of correspondence assertions [17]. Since causal graphs are of finite size, the analysis is assured to terminate. The analysis entails security proofs that establish the safety of the

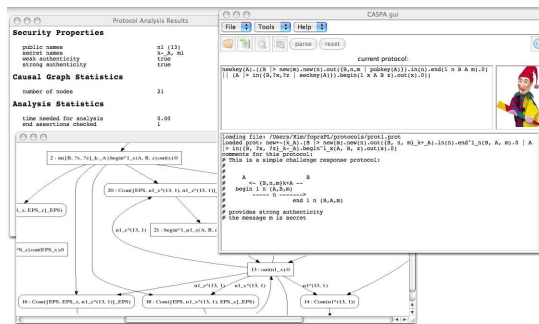


Fig. 1. The CASPA Tool.

Protocol	CASPA	TA4SP	OFMC	Protocol	CASPA	TA4SP	OFMC
CHAPv2	0,93s	10,59s	0,32s	NSPK	0,13s	7,56s	0,01s
CRAM-MD5	0,09s	-	0,71s	NSPK-KS	28m	-	1,1s
EKE	0,81s	7,56s	0,19s	NSPK-fix	0,08s	0,98s	0,18s
IKEv2-CHILD	0,31s	-	1,19s	NSPK-KS-fix	7m	-	24,86s
ISO1	0,05s	×	0,02s	SHARE	0,4s	14,38s	0,08s
ISO3	1,08s	×	0,04s	UMTS-AKA	0,04s	0,51s	0,02s
LPD-MSR	0,05s	-	0,02s	APOP	0,44s	×	2,94s
LPD-IMSR	0,37s	-	0,08s	DHCP-DA	1,03s	-	0,06

Table 1. Protocol results, conducted on a Pentium-IV 3GHz 1GB under linux.

protocol for a potentially unbounded number of protocol executions. As usual for static analysis techniques and due to the undecidability of the security problem, false positives may occur caused by an insufficient precision of the analysis, hence potentially classifying secure protocols as insecure. The CASPA tool is freely available at [4].

3 Performance Evaluation of CASPA

We evaluate the performance of our tool by running it on a subset of the AVISPA library. To facilitate our experiments, we developed a translator from the Intermediate Format protocol language [2] that the AVISPA suite is based upon into the dialect of the spi-calculus [1] used in our tool. The translation is only partially automated in that it requires some manual steps, such as specifying the owners of the keys and defining suitable correspondence assertions. These steps were straightforward in all protocols considered so far, but they admittedly require basic familiarity with our language and understanding of the protocol.

The results are reported in Table 1. The tool succeeded in the analysis of safe protocols (i.e., we did not get any false positives), and it failed to establish security proofs of flawed protocols as expected. For each protocol, the table reports the running time for CASPA and TA4SP. The performance evaluation shows that CASPA is capable of dealing with a substantially larger set of protocols than TA4SP: the symbol - means that the protocol is not supported by the tool, while the symbol × means that the protocol guarantees only authenticity properties, which can be verified by CASPA and not by TA4SP. In addition, the evaluation shows that even for the protocols that are in scope of both TA4SP and CASPA, the CASPA tool improves in terms of performance. Note that the subset of protocols we consider comprises all protocols of the AVISPA library for which an analysis with TA4SP succeeds. Some protocols in the AVISPA library employ non-standard equational theories that are not supported by our tool (e.g., NSPKxor that is based on xor) and for some other protocols the analysis did not succeed due to memory exhaustion (e.g., SET and TSL).

For comparison, we additionally depict the running times of OFMC [5], the most advanced model checker in the AVISPA tool, when the analysis is con-

strained to three executions. In this restricted setting, OFMC shows significantly better performances. Scaling this approach to more executions rapidly becomes infeasible, hence leaving potential attacks undetected (e.g., cf. [10,15]).

References

1. M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, 1999.
2. A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, L. Cuellar, P. Drielsma, P. Heám, O. Kouchnarenko, J. M. S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The avispa tool for the automated validation of internet security protocols and applications. In *Proc. Computer Aided Verification'05 (CAV)*, LNCS, pages 281–285, 2005.
3. M. Backes, A. Cortesi, and M. Maffei. Causality-based abstraction of multiplicity in cryptographic protocols. In *Proc. 20th IEEE Symposium on Computer Security Foundations (CSF)*, pages 355–369. IEEE, 2007.
4. M. Backes, S. Lorenz, M. Maffei, and K. Pecina. The CASPA tool. Available at www.infsec.cs.uni-sb.de/caspa.
5. D. A. Basin, S. Mödersheim, and L. Viganò. Ofmc: A symbolic model checker for security protocols. *IJIS*, 4(3):181–208, 2005.
6. B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW)*, pages 82–96. IEEE, 2001.
7. B. Blanchet and A. Podelski. Verification of cryptographic protocols: Tagging enforces termination. In *Proc. 6th International Conference on Foundations of Software Science and Computation Structures (FOSSACS)*, pages 136–152, 2003.
8. C. Bodei, M. Buchholtz, P. Degano, F. Nielson, and H. R. Nielson. Static validation of security protocols. *Journal of Computer Security*, 13(3):347–390, 2005.
9. Y. Boichut and T. Genet. Feasible trace reconstruction for rewriting approximations. In *Term Rewriting and Applications (RTA 2006)*, pages 123–135, 2006.
10. R. Chadha, S. Kremer, and A. Scedrov. Formal analysis of multi-party contract signing. In *Proc. 17th IEEE Computer Security Foundations Workshop (CSFW)*, pages 266–279. IEEE, 2004.
11. C. Cremers. *Scyther - Semantics and Verification of Security Protocols*. Ph.D. dissertation, Eindhoven University of Technology, 2006.
12. S. Doghmi, J. Guttman, and F. Thayer. Searching for shapes in cryptographic protocols. In *Proc. 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, LNCS, pages 523–538, 2007.
13. T. Genet and V. Tong. Reachability analysis of term rewriting systems with timbuk. In *Proc. Artificial Intelligence on Logic for Programming (LPAR '01)*, pages 695–706. Springer-Verlag, 2001.
14. G. Lowe. “A Hierarchy of Authentication Specification”. In *Proc. 10th IEEE Computer Security Foundations Workshop (CSFW)*, pages 31–44. IEEE, 1997.
15. J. Millen. A necessarily parallel attack, 1999. In *Proc. of Workshop on Formal Methods and Security Protocols*.
16. D. X. Song. Athena: a new efficient automatic checker for security protocol analysis. In *Proc. 12th IEEE Computer Security Foundations Workshop (CSFW)*, pages 192–202. IEEE, 1999.
17. T. Y. C. Woo and S. S. Lam. A lesson on authentication protocol design. *Operation Systems Review*, 28(3):24–37, 1994.