Formal analysis of security APIs using Tamarin

Laboratory, institution and university The internship will be located at IRISA (Rennes).

Team or project of the Lab Team EMSEC at IRISA.

Name and email address of the advisor Stéphanie Delaune, Stephanie.Delaune@irisa.fr and Cyrille Wiedling (DGA)

Indemnisation The internship is supported by **POPSTAR** (ERC Starting Grant).

Keywords security protocols, formal verification, symbolic model

Context. Security APIs can be seen as sets of security protocols or commands [7]. They typically arise in systems where certain security-critical fragments of a program are executed on some tamper resistance device (TRD), such as smartcard, USB security token or hardware security module (HSM). They aim at ensuring a security policy, i.e. no matter what commands are received from the untrusted code, certain security properties will continue to hold, e.g. the secrecy of a sensitive cryptographic key. Such protocols are widely deployed and have been shown to have flaws, e.g. Yubikey [9], and PKCS#11 [6].

Formal methods have demonstrated their usefulness when designing and analyzing security protocols. They indeed provide rigorous frameworks and techniques that have allowed to discover new flaws. However, it is well-admitted that security APIs have some specificities to use tools designed for classical cryptographic protocols [8]. One of the difficulties in reasoning about security APIs is the need for non-monotonic global states. If the tamper resistant device is in a certain state s, and then a command is successfully executed, then typically the TRD ends up in a state $s' \neq s$. Commands that require it to be in the previous state s will no longer work. Security API models typically contain non-monotonic global state, which must be modelled accurately to get reasonable precision (i.e. not too many false attacks).

Nowdays several efficient verification tools exist, *e.g.* ProVerif [2], and Tamarin [1]. They can be used to analyse security protocols in a more or less completely automatic way. Of course, this requires to specify the protocol in the input language of the tool. This modelling step has to be done with a lot of care and requires some expertise. In the context of security APIs, the tool Tamarin is particulary interesting since it allows one to model stateful protocols.

Objectives of the internship. The goal of this internship is to analyse some security APIs using the verification tool Tamarin. To start on this proposal the intern will first study the generic security API proposed in [3] and for which a manual proof has been done. The aim is to propose a mechanized proof of this relatively simple API to better understand the subtlelties of using a verification tool like Tamarin. In a second phase, this work will be extended to analyse a more complex security API which allows revocation and update of long-term keys [4].

Such a security API is a bit more complex in its design and also regarding the security properties it is supposed to achieved. A third API of interest is the one proposed by Daubignard *et al.* [5] and which extends the one proposed in [3] to deal with asymmetric cryptography.

Expected skills. We are looking for candidates with good skills in Foundations of Computer Science (logic, automatic deduction, ...) Some knowledge in security is not mandatory. The candidate will assimilate this knowledge during the internship. This internship may also lead to a PhD thesis on similar topics.

Références

[1] D. Basin, C. Cremers, J. Dreier, S. Meier, R. Sasse, and B. Schmidt. Tamarin. https ://tamarin-prover.github.io/.

[2] B. Blanchet, V. Cheval, X. Allamigeon, B. Smyth, and M. Sylvestre. Proverif. https ://prosecco.gforge.inria.fr/personal/bblanche/proverif/.

[3] V. Cortier and G. Steel. A generic security API for symmetric key management on cryptographic devices. *Inf. Comput.*, 238 :208–232, 2014.

[4] V. Cortier, G. Steel, and C. Wiedling. Revoke and let live : a secure key revocation api for cryptographic devices. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 918–928. ACM, 2012.

[5] M. Daubignard, D. Lubicz, and G. Steel. A secure key management interface with asymmetric cryptography. In *Principles of Security and Trust - Third International Conference, POST 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings*, volume 8414 of *Lecture Notes in Computer Science*, pages 63–82. Springer, 2014.

[6] S. Delaune, S. Kremer, and G. Steel. Formal security analysis of PKCS#11 and proprietary extensions. *Journal of Computer Security*, 18(6) :1211–1245, 2010.

[7] R. Focardi, F. L. Luccio, and G. Steel. An introduction to security API analysis. In *Foundations of Security Analysis and Design VI - FOSAD Tutorial Lectures*, volume 6858 of *Lecture Notes in Computer Science*, pages 35–65. Springer, 2011.

[8] J. Herzog. Applying protocol analysis to security device interfaces. *IEEE Security & Privacy Magazine*, 4(4) :84–87, July-Aug 2006.

[9] R. Künnemann and G. Steel. Yubisecure ? formal security analysis results for the yubikey and yubihsm. In *Security and Trust Management - 8th International Workshop, STM 2012, Pisa, Italy, September 13-14, 2012, Revised Selected Papers*, volume 7783 of *Lecture Notes in Computer Science*, pages 257–272. Springer, 2012.