A Formal Analysis of Authentication in the TPM
(extended abstract)

Stéphanie Delaune[1], Steve Kremer[1], Mark D. Ryan[2], and Graham Steel[1]
[1]LSV, ENS Cachan & CNRS & INRIA Saclay Île-de-France, France
[2]School of Computer Science, University of Birmingham, UK

1 Introduction

The Trusted Platform Module (TPM) is a hardware chip designed to enable commodity computers to achieve greater levels of security than is possible in software alone. To this end, the TPM provides a way to store cryptographic keys and other sensitive data in its shielded memory. Through its API, one can use those keys to achieve some security goals. There are 300 million TPMs currently in existence, mostly in high-end laptops, but now increasingly in desktops and servers. The TPM specification is an industry standard [11] and an ISO/IEC standard [9] (more than 700 pages) coordinated by the Trusted Computing Group.

Several papers have appeared describing systems that leverage the TPM to create secure applications, but most of these assume that the TPM API behaves correctly and provides the high-level security properties required [6, 7]. Lower level analyses of the TPM API also exist and several vulnerabilities in the TPM API have been discovered: offline dictionary attacks on the passwords or 'authdata' used to secure access to keys [5], attacks exploiting the fact that the same authdata can be shared between users [4], an attacker can also in some circumstances illegitimately obtain a certificate on a TPM key of his choice [8], ...These attacks highlight the necessity of formal analysis of the API specification. We perform such an analysis in this work, focusing on the mechanisms for authentication and authorisation.

2 Our Contributions

We model a collection of four TPM commands, concentrating on the authentication mechanisms. We identify security properties which we will argue are central to correct and coherent design of the API. We formalise these properties for our fragment in the applied pi calculus [1], and using ProVerif [3], we rediscover some known attacks on the API and some new variations on them. We propose some fixes to the API, partly inspired by ongoing discussions in the Trusted Computing Group, and prove our security properties for the modified API.

One of the difficulties in reasoning about security APIs such as that of the TPM is *non-monotonic state*. If the TPM is in a certain state s, and then a command is successfully executed, then typically the TPM ends up in a state $s' \neq s$. Commands that require it to be in the previous state s will no longer work. We chose ProVerif after first experimenting with the AVISPA tool suite [2], which provides support for mutable global state. However, of the AVISPA back ends that support state, OFMC and CL-AtSe require concrete bounds on the number of command invocations and fresh nonces to be given. It is possible to avoid this restriction using SATMC, but SATMC performed poorly in our experiments

Tools such as ProVerif are not optimised to work with non-monotonic state. We address this restriction by introducing the assumption that only one command is executed in each authorisation session.

Submitted to:
SecCo 2010

This assumption appears to be quite reasonable. Indeed, the TPM imposes the assumption itself whenever a command introduces new authdata. Moreover, tools like TPM/J [10] that provide software-level APIs also implement the assumption. Again to avoid non-monotonicity, we do not allow keys to be deleted from the memory of the TPM. However, we allow an unbounded number of keys to be loaded.

The TPM specification does not detail explicitly which security properties are intended to be guaranteed, although it provides some hints. For example, the specification [11, Part I, p.60] states that: "*The design criterion of the protocols is to allow for ownership authentication, command and parameter authentication and prevent replay and man in the middle attacks.*" We will formalise these security properties as *correspondence properties* that state:

1. If the TPM has executed a certain command, then a user in possession of the relevant authdata has previously requested the command.
2. If a user considers that the TPM has executed a certain command, then either the TPM really has executed the command, or an attacker is in possession of the relevant authdata.

The first property expresses authentication of user commands, and is achieved by the authorisation HMACs that accompany the commands. The second one expresses authentication of the TPM, and is achieved by the HMACs provided by the TPM with its answer. We argue that the TPM certainly aims at achieving these properties, as otherwise there would be no need for the HMAC mechanism. The above mentioned properties can be expressed by injective correspondence properties. In all experiments, the security properties under test are the correspondence properties explained above.

Our methodology was to first study some core key management commands in isolation in order to analyse the weakness of each command. This leads us to propose some fixes for these commands. Then we carried out an experiment where we consider the commands TPM_CertifyKey, TPM_CreateWrapKey, TPM_LoadKey2, and TPM_UnBind together. We consider the fixed version of each of these commands and we show in our last experiment (Experiment 10) that the security properties are satisfied for a scenario that allows:

- an attacker to load his own keys inside the TPM, and
- an honest user to use the same authdata for different keys.

All the files for our experiments are available on line at:

`http://www.lsv.ens-cachan.fr/~delaune/TPM`.

In our first six experiments, we model the command TPM_CertifyKey in isolation. Then, in Experiments 7-9, we model the command TPM_CreateWrapKey only. Lastly, in Experiment 10, we consider a model where the commands TPM_CertifyKey, TPM_CreateWrapKey, TPM_LoadKey2, and TPM_UnBind are taken into account.

3 Conclusion and Future Work

In this work, we proposed a detailed modelling of a fragment of the TPM in the applied pi calculus. We model core security properties as correspondence properties and use the tool ProVerif to automate our security analysis. We were able to rediscover several known attacks and some new variants of these attacks.

As future work we foresee to extend our model with more commands such as the key migration part. We also plan to include a modelling of the TPM's registers (the PCRs) which allow to condition some commands on the current value of a register. PCRs are crucial when using the TPM for checking the integrity of a system. Modelling the PCRs and the commands for manipulating these registers for automated verification seems to be a challenging task.

Acknowledgments. Mark Ryan gratefully thanks Microsoft and Hewlett-Packard for interesting discussions and financial support that contributed to this research.

References

[1] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. 28th Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115. ACM Press, 2001.

[2] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, 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 *Computer Aided Verification, 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6-10, 2005, Proceedings*, pages 281–285, 2005.

[3] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *14th IEEE Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Computer Society Press, 2001.

[4] L. Chen and M. Ryan. Attack, solution and verification for shared authorisation data in TCG TPM. In P. Degano and J. D. Guttman, editors, *Formal Aspects in Security and Trust*, volume 5983 of *Lecture Notes in Computer Science*, pages 201–216. Springer, 2009.

[5] L. Chen and M. D. Ryan. Offline dictionary attack on TCG TPM weak authorisation data, and solution. In D. Grawrock, H. Reimer, A. Sadeghi, and C. Vishik, editors, *Future of Trust in Computing*. Vieweg & Teubner, 2008.

[6] A. Datta, J. Franklin, D. Garg, and D. Kaynar. A logic of secure systems and its application to trusted computing. In *Proceedings of 30th IEEE Symposium on Security and Privacy*, pages 221–236, May 2009.

[7] Y. Gasmi, A.-R. Sadeghi, P. Stewin, M. Unger, and N. Asokan. Beyond secure channels. In *Scalable Trusted Computing (STC'07)*, pages 30–40, November 2007.

[8] S. Gürgens, C. Rudolph, D. Scheuermann, M. Atts, and R. Plaga. Security evaluation of scenarios based on the TCG's TPM specification. In *ESORICS*, pages 438–453, 2007.

[9] ISO/IEC PAS DIS 11889: Information technology – Security techniques – Trusted platform module.

[10] L. Sarmenta. TPM/J developer's guide. Massachussetts Institute of Technology.

[11] Trusted Computing Group. TPM Specification version 1.2. Parts 1–3, revision 103. `http://www.trustedcomputinggroup.org/resources/tpm_main_specification`, 2007.