Model-Based Testing of Electronic Passports

Wojciech Mostowski[1], Erik Poll[1], Julien Schmaltz[2], Jan Tretmans[1,2], and Ronny Wichers Schreur[1]

[1] Radboud Universiteit, Nijmegen, the Netherlands
[2] Embedded Systems Institute (ESI), Eindhoven, the Netherlands

Introduction

Electronic passports, or e-passports for short, contain a contactless smartcard which stores digitally signed data. To rigorously test e-passports, we developed formal models of the e-passport protocols that enable model-based testing using the TorXakis framework.

E-passport protocols

Access to e-passports involves several protocols. All e-passports should conform to the standards of the International Civil Aviation Organization (ICAO) [3]. To prevent surreptitious access to the e-passport chip, ICAO specifies the Basic Access Control (BAC) protocol, which establishes a secure channel based on a symmetric session key. To prevent cloning of e-passports, ICAO specifies an optional Active Authentication protocol. The second generation of e-passports being introduced in the EU in the summer of 2009 contains more sensitive biometric data, namely fingerprints. To protect this information, the EU mandates the use of a stronger security mechanism called Extended Access Control (EAC) [1]. EAC requires the passport terminal to authenticate itself (with a chain of PKI certificates) to the passport before any sensitive biometric data can be read.

Formal Models and Model-Based Testing

The official standards give long and detailed descriptions of the individual protocols. Understanding the combination and possible interaction of the various protocols is difficult. To better understand the protocols, we developed formal models in the form of finite state diagrams. Initially we just drew these on a whiteboard to understand the specifications, but then we realised that they could be used to test e-passports using *model-based testing.*

In model-based testing a formal model is used to automate the testing process: a formal model specifying the desired behaviour is used to generate tests and analyse the test results. One theory for model-based testing uses labelled transition systems as models, and an implementation relation called **ioco** that formally defines when an implementation is correct with respect to its specification model [5].

We expressed our state diagrams as labelled transition systems and then used the TorXakis model-based testing tool to test actual e-passports. TorXakis is based on the model-based testing tool TorX [6] extending it with symbolic test generation capabilities [2]. TorXakis performs random walks through the model, sending commands to the passport chip and verifying that the responses conform to the model. TorXakis is implemented in Haskell[3]. For lower-level communication with the passport chip using a card reader, we used the open source passport terminal software that we helped develop in the JMRTD project[4].

Results and Conclusions

The most difficult part of the testing process was understanding the official specifications and constructing a formal model for them. Finite state diagrams turn out to be a very effective and perspicuous way to specify the combination of passport protocols. Indeed, it amazes us that the official specifications do not use finite state diagrams anywhere.

Once we had the model, the actual testing, including developing the extra software, only took about a week. Here we did take advantage of the existing e-passport terminal software we had. But note that this software, like any passport terminal software, only executes one particular sequence of protocol steps, whereas to rigorously test the security of a passport one has to try all different possible sequences of these steps. This is what we let TorXakis do for us.

The tests were run fully automatically: an overnight test was able to perform over 100 000 protocol steps on a passport. Once the test infrastructure has been set up, model-based testing simply amounts to playing with models. By slightly changing the model, new tests are derived automatically by a simple key stroke. By refining and tweaking the model we could quickly find out how any underspecification or unclarities in the specifications had been resolved in the implementation we tested.

We will publish our formal models of the e-passport protocols as an aid to anyone implementing or testing e-passport software, or indeed anyone just trying to understand the specifications. We plan to do the same for the ‘twin brother’ of the e-passport, the new electronic driving license [4].[5]

References

1. Advanced security mechanisms for machine readable travel documents – Extended access control (EAC) – Version 1.11. Technical Report TR-03110, German Federal Office for Information Security (BSI), Bonn, Germany, 2008.

[3] `http://www.haskell.org`

[4] `http://jmrtd.org`

[5] For which we developed the first (open source) reference implementation, `http://isodl.sourceforge.net`.

2. L. Frantzen, J. Tretmans, and T. Willemse. Test Generation Based on Symbolic Specifications. In J. Grabowski and B. Nielsen, editors, *Formal Approaches to Software Testing – FATES 2004*, volume 3395 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2005.
3. Doc 9303 – Machine readable travel documents – Part 1 - 2. Technical report, ICAO, 2006. Sixth edition.
4. ISO/ICE. ISO-compliant driving license – Part 1, 2, 3. Technical report, 2009. ISO 18013.
5. J. Tretmans. Model Based Testing with Labelled Transition Systems. In R.M. Hierons, J.P. Bowen, and M. Harman, editors, *Formal Methods and Testing*, volume 4949 of *Lecture Notes in Computer Science*, pages 1–38. Springer, 2008.
6. J. Tretmans and E. Brinksma. TorX : Automated Model Based Testing. In A. Hartman and K. Dussa-Zieger, editors, *First European Conference on Model-Driven Software Engineering*. Imbuss, Möhrendorf, Germany, December 11-12 2003. 13 pages.