JavaCard Program Verification

Bart Jacobs
Computing Science Institute, University of Nijmegen
Toernooiveld 1, 6525 ED Nijmegen
`bart@cs.kun.nl` `http://www.cs.kun.nl/~bart`

This abstract provides some background information on smart cards, and explains the challenges these cards represent for formal verification of software.

Smart card trends

Increasingly, physical keys are being replaced by cryptographic keys, which are typically a thousand bits in size. Modern smart cards are the ideal carriers for such keys, because they have enough computing power to do the necessary en- or de-cryption on-card, so that the secret key never has to leave the card. Smart cards are meant to be tamper-resistant secure tokens, typically bound to individuals via a PIN, possibly in combination with biometric identification.

Modern smart cards contain a standard interface (API) for a mini-operating system which is capable of executing application programs (called *applets*) written in high-level languages. The standard language in this area is JavaCard [Che00], which is a "superset of a subset" of Java: it is a simplified version of Java (no threads, multi-dimensional arrays, floats or doubles) with some special features, like persistent objects and a transaction-commit mechanism. Two further characteristics of modern smart cards are:

- **Multi-application.** One card can hold multiple applets for different applications. These are typically centered around the basic cryptographic functions, in for example banking, telecommunication (GSM or UMTS SIMs), access to buildings or networks, identification, voting, *etc.*) Limited communication can be allowed between applets, *e.g.* for automatically adding loyalty points after certain commercial transactions.

- **Post-issuance downloading.** In principle, it is possible to add applets to a card after it has been issued. This option gives enormous flexibility, but is disabled for security reasons in all currently operational cards.

Security evaluation and certification are very important in the smart card area, because cards are distributed in large numbers for security critical applications. Correctness should be established both for the card platform, and for applets.

Challenges for formal methods

The so-called Common Criteria[1] form a standard for the evaluation of IT security. They are much used for smart cards. The Common Criteria involve seven different levels of evaluation, where the two highest levels (6 and 7) require the explicit use of formal methods. Currently available smart cards are evaluated at levels 4 and 5, but there is a clear pressure to move to higher levels. Therefore the smart card industry is open to the use of formal methods. For the formal verification community smart cards form an ideal target since they are within reach of existing verification tools, because of their limited size.

VerifiCard: aims

In 2000 a European research consortium called VerifiCard[2] was set up, with support from the European Union's IST programme. Its aim is to apply the existing verification tools (mostly theorem provers, but also model checkers) to establish the correctness of JavaCard-based smart cards. The approach is pragmatic: no new development of semantics of Java(Card) from scratch, but application of available expertise and experience in Europe in tool-assisted formal methods, to see what can be achieved in a concentrated effort in a small, well-defined area. This is a potential killer application for formal methods in software verification, which can cut in two directions: if this succeeds, the success may spread to other areas. But if this fails there may be a serious setback for formal methods in software: it is bad news if these methods cannot deliver for such relatively small systems as smart cards.

VerifiCard: work

The VerifiCard consortium consists of five academic partners, some of which are well-known in the TPHOLs community: Nijmegen (coordinator; Jacobs, Poll), INRIA (Barthe, Bertot, Jensen, Paulin-Mohring), Munich (Nipkow, Strecker), Hagen (Poetzsch-Heffter), SICS (Dam). Also, the consortium involves two smart card manufacturers: Gemplus and SchlumbergerSema (formerly Bull CP8). The planned work is roughly divided along two lines: source code / byte code, and platform / applets. The work involves for instance formalisation of the JavaCard virtual machine and byte code verifier, non-interference properties, and specification and verification of the API and of individual applets. Towards the end of the project the industrial partners will carry out a tool-evaluation, to see which approaches can contribute most to their evaluation needs.

Scientific work of Nijmegen

The talk will elaborate on the work done at Nijmegen, as part of VerifiCard. This involves verification of JavaCard programs, that are part of the API and

[1] See `www.commoncriteria.org`

[2] See `www.verificard.org`

applets. The correctness properties are specified using the interface specification language JML [LBR99], developed by Gary Leavens *et al.* in Iowa, see *e.g.* [PBJ01]. A Java(Card) program with JML annotation is translated to PVS using the LOOP tool [BJ01]. Actual verification in PVS proceeds via a tailor-made Hoare logic for JML [JP01]. See [BJP01] for a small case study, involving the `AID` class from the JavaCard API. Basically, the verification technology for Java is there, but scaling it up to larger programs is still a real challenge.

References

[BJ01] J. van den Berg and B. Jacobs. The LOOP compiler for Java and JML. In T. Margaria and W. Yi, editors, *Tools and Algorithms for the Construction and Analysis of Software (TACAS)*, number 2031 in Lect. Notes Comp. Sci., pages 299–312. Springer, Berlin, 2001.

[BJP01] J. van den Berg, B. Jacobs, and E. Poll. Formal specification and verification of JavaCard's Application Identifier Class. In I. Attali and Th. Jensen, editors, *Proceedings of the Java Card 2000 Workshop*, Lect. Notes Comp. Sci. Springer, Berlin, 2001.

[Che00] Z. Chen. *Java Card Technology for Smart Cards*. The Java Series. Addison-Wesley, 2000.

[JP01] B. Jacobs and E. Poll. A logic for the Java Modeling Language JML. In H. Hussmann, editor, *Fundamental Approaches to Software Engineering (FASE)*, number 2029 in Lect. Notes Comp. Sci., pages 284–299. Springer, Berlin, 2001.

[LBR99] G.T. Leavens, A.L. Baker, and C. Ruby. JML: A notation for detailed design. In H. Kilov and B. Rumpe, editors, *Behavioral Specifications of Business and Systems*, pages 175–188. Kluwer, 1999.

[PBJ01] E. Poll, J. van den Berg, and B. Jacobs. Formal specification of the JavaCard API in JML: the APDU class. *Comp. Networks Mag.*, 2001. To appear.