Making Smartcards Secure

Susan Stepney

Logica UK Ltd. (stepneys@logica.com)

In the early 1990s, *platform seven* (at that time part of the National Westminster Bank) started developing an innovative Smartcard-based electronic cash system, Mondex. A Smartcard is a credit-card-sized plastic card with a computer chip, which allows it to be programmed for various applications. Mondex is an application designed to work like electronic *cash*, which, unlike say an electronic cheque, has no third party authentication or authorisation involved in a transaction. The cost of third party clearing of an electronic transaction cannot be brought much below tens of pence, which makes it infeasible for low value cash-like transactions, such as buying a pint of milk or a newspaper, or use in vending machines, or in public telephone boxes. Mondex was designed to solve this problem, by having no third party involvement, and no per-transaction cost. Once the issuing bank has loaded electronic cash into the system, it has very little further control over it -- just like real cash.

Because of this lack of third party control, it is vitally important that the security of the card can not be broken, otherwise computer-literate criminals could "electronically print" money with ease. So *platform seven* decided to develop the full Mondex product to the very highest standards.

There are many standards and guidelines for developing software and systems to high levels of assurance. In the past, these have been mainly the domain of the military and security services, with their interest in safety critical and security critical systems. But now that safety critical and financially critical systems are becoming everyday consumer products, these standards are being used to develop some of them.

One such standard is the *UK ITSEC* scheme (UK IT Security Evaluation and Certification scheme), initially devised to provide levels of assurance for secure operating systems. The scheme, recognised throughout the EU, is run in the UK by CESG (Communications-Electronics Security Group), who licence *CLEFs* (Commercial Evaluation Facilities) to evaluate products. (*Common Criteria*, ISO 15408, is the emerging global counterpart of ITSEC.) The highest certification level in this scheme, ITSEC level E6, mandates stringent requirements on the design, development, testing and documentation procedures. It also mandates the use of formal methods (FM): to specify the high level abstract security policy model, to specify the lower level concrete architectural design, and to provide a formal proof of correspondence between the two levels, in order to show that the concrete design enjoys the abstract security properties.

platform seven decided to develop the full Mondex to this highest ITSEC level, E6. This was despite the fact that such a high level of certification had never previously been achieved (not even for Government sponsored secure computing developments, let alone for a commercial product), and there was a perception by many in the

industry that it was beyond the state of the art. This perception was mainly due to the FM requirements, as many equate "mathematics" with "impossible". Also the large overhead in the conventional part of the development process was thought to be too onerous and costly for a full commercial product. (A much smaller product, the "One Way Regulator", had previously been certified to the earlier UKL6. The technical challenges experienced on that successful small project helped the common perception of infeasibility for larger commercial projects.)

platform seven performed the challenging task of designing the secure cash-transfer protocol, and correctly implementing it, given the space and speed constraints of a Smartcard. These cards have very limited memory, with no built-in operating system support for tasks such as memory management. Also, careful programming techniques are needed to ensure correct functioning if power is withdrawn at any point during processing a transaction, if say the card has been withdrawn from the reader.

platform seven were determined to use only the best for all aspects of this advanced development. So, early in 1994, they contacted Logica to deliver the specification and proof requirements of the E6 development. Logica's formal methods specialists chose the widely used Z language in which to write the two specifications and to perform the correspondence proof. The team had little difficulty formalising the concrete architectural design from the existing semi-formal design documents, but the task of producing an abstract security policy model that both captured the desired security properties (in particular "no value is created", and "all value is accounted for"), and provably corresponded to the lower level specification, was much harder. A very small change in the design (very small from the perspective of the FM team, that is) would have made the abstraction much easier, but was deemed to be too expensive, as the parallel implementation work was already beyond that point. Eventually, after much experimentation with different approaches, the original design *was* successfully abstracted.

At that stage, the correspondence proof hit an unexpected snag. Although the Logica team were confident that the proof obligation implied by the Mondex specifications was true, they were unable to prove it using the standard proof rules available in the literature at the time. Consultancy with Oxford University Computing Laboratory explained why: those particular proof rules are suitable for only some classes of specification, and Mondex was of a different class.

Oxford researchers had been aware for some time of the full underlying refinement theory, and that the existing Z proof rules were incomplete, but until this project, they had never come across a major, real-world example of a system that required the full generality. The underlying refinement theory did not apply directly to Z, and considerable work was required to establish it in Z. In addition, because these new rules were not established in the literature, their derivation had to be presented to a standard that would convince the ITSEC evaluators who were certifying the system. So Oxford went back to first principles, and helped produce an 80-page derivation of a complementary set of Z proof rules that were applicable to the Mondex specification. Logica then took these raw rules, and applied them to the specific case, and successfully discharged the proof obligation, with 200 pages of mathematics.

The proof discovered a small flaw in one of the minor protocols – the design was changed to rectify that.

The development of these new proof rules exposed many previously hidden assumptions and restrictions of the classical Z refinement rules, which has subsequently led to a boom in academic research in this area. This development demonstrates a fruitful synergy between academia and industry: each doing what they do best, whilst providing interesting problems and valuable results to the other. The complementary rules have since been published, and are now available as part of the standard literature, for others to use.

All three institutions involved in this development brought unique skills, without which the full development of the products would not have been possible.

Once it became clear that E6 was indeed going to be achievable for Mondex (a fact not necessarily obvious from the outset), the decision was taken to develop Multos, a secure Smartcard Operating System, also to E6. Multos allows several applications to co-reside securely on a single Smartcard, guaranteeing to each its own secure operating environment. This permits different application providers to be confident that another application cannot harm their own, either by accident or by malice. Without such guarantees, application providers would be highly reluctant to allow their applications, with their secure algorithms and data, to co-reside with others; yet consumers would be reluctant to carry a separate Smartcard for each application. A secure operating system is essential for Smartcard-based e-commerce.

platform seven performed the design and implementation of the virtual machine that would provide all the required security features. There were major technical challenges in fitting Multos into the very small memory space available, whilst still leaving room for applications such as Mondex to be loaded into the remaining space, all without compromising the clarity of the design needed to achieve correct implementation and E6 certification.

Multos has a seemingly paradoxical security policy: applications are segregated from each other, yet are allowed to communicate in certain well-defined ways. Also, the implementation is forced to use shared areas of memory, yet the applications must not be able to use this to communicate covertly, no matter how maliciously the application may be written. Logica, again handling the FM development requirements, took ideas from the formal languages CSP and Z (both from Oxford), and explained precisely what the required segregation property means mathematically, and showed that it did in fact make sense. This segregation property was eventually formalised in just two lines of Z, but it then had to be shown to hold for the full-scale Multos application. This required an involved argumentation structure and several hundred pages of Z proof. This sufficed to show that, although the various applications do indeed share the same areas of memory, they do so in a way that means they cannot communicate with one another.

For Multos, Logica were in close consultation with *platform seven*'s design team from the beginning. This had the effect of making life differently hard. The FM team were able to influence the design in order to make specification work cleaner and clearer. As a result, they had to rework their specifications and proofs, in order to

keep them in line with the changing design. While this did result in some frustration, the improvement in clarity of final design was well worth the effort, and it did ease the final proof task considerably.

Given the existence of a secure platform, Mondex was then re-engineered to be a Multos application.

In 1999, both Mondex and Multos achieved their ITSEC E6 certificates -- the very first products ever to do so. Mondex is now being used around the world in a variety of application areas. Multos is internationally recognised as the most secure Smartcard operating system available on the market, suitable for hosting financially- or otherwise security-critical applications. Other Smartcard application developers recognise that these two products have raised the standard, and are themselves beginning to develop their own products to ITSEC (and now the global Common Criteria) certification levels. *platform seven* is now developing further Smartcard products.

In the finance sector, the security of products, and public confidence in that security, is all important. Being able to demonstrate conformance with ITSEC E6, the most stringent of Government-backed security standards, provides a high level of quality and confidence.

Both Mondex and Multos were developed to the highest non-formal industry practice, in parallel with the development of the Z specifications and proofs, in order to gain maximum assurance and quality from the entire process. This stringent development resulted in an initial system design and implementation with astonishingly few bugs. So the E6 certification process, both in its informal and formal requirements, demonstrably resulted in higher quality products. This provided the increased security and consumer confidence necessary to make these products possible in the marketplace.

Some formal development, outside the E6 process, was also applied to the definition of the intermediate programming language used to write Multos applications, which provides the basis of ensuring secure segregation. The formal development resulted in a programmer's manual that had a clearer description of the language (derived in part from the formal specification) and an implementor's manual that had a clearer definition of the necessary security checks. So the formal methods work demonstrably resulted in a higher quality product, and higher quality documentation.

Interestingly, in neither development were the FM tasks on the critical path. All the specification and proof work occurred in parallel with the rest of the development, and was ready for evaluation on time. (The cynic might speculate whether the other requirements for E6 certification merely slowed the development *even more*, of course.)

So, the benefits that formality gave to these particular products include: increased assurance; discovery of errors; cleaner design; a better understanding of the security properties being claimed. Usability, robustness, fitness for purpose, these are *all* of paramount importance for products. And so is correctness. FM can help make software correct. (See the box "what FM can do for you".)

Dr Susan Stepney, MBCS, CEng, is a Principal Consultant in Logica Security Consulting, specialising in the application of formal methods, and a Visiting Professor of Computer Science at the University of York.

Other things FM can do for you

Full-blooded formal specification and correspondence proof, as used in the E6-certified products described in the main text, represents only one of many ways the FM can be used to support and give benefit to critical systems developments. FM may be applied in a variety of diverse ways to develop products faster, cheaper, better:

- **Specify the wider system** : clarify the relationships with non-software parts of the system, and make explicit the requirements on them for correct functioning of the entire system. It can help ensure the proposed product will actually be secure in a realistic industrial environment.

- **Capture high level requirements and explore the design :** encourage clarity of thought during requirements elicitation and high level design. This helps eliminate some of the most expensive errors: those made in the earliest development stages.

- **Low level algorithm analysis :** demonstrate the correct design of an algorithm, that a loop always terminates, that an optimisation is correct, that some proposed assembly code correctly implements a high level algorithm. This helps eliminate subtle bugs in complex algorithms.

- **Specify and develop test suites** : the test plan is thus is available early on, so it can be used in a uniform manner to test prototypes, emulations, etc, as well as the final implementation. This helps produce better test plans that can catch more bugs.

- **Animation :** produce a rapid prototype of a high level specification to provide an early instrumented demonstration. This allows early exploration and validation of designs, and experiments with behaviours, before the detailed coded implementations or specific hardware platforms are available, and can include behaviour not intended to be implemented in the final delivery.

- **Non-functional analyses** : analyse security and safety properties, which may be functional or non-functional; analyse performance/memory use/scaling issues. This helps quantitative investigation of a variety of properties of systems.

Further reading

- Anthony Hall. “Seven Myths of Formal Methods”. IEEE Software, **7**(5). 1990.

- Bertrand Meyer. “On formalism in specifications”. IEEE Software, **2**:6–26, January 1985.

- Susan Stepney. “A Tale of Two Proofs”. In *BCS-FACS Third Northern Formal Methods Workshop*, Ilkley, 1998. http://public.logica.com/~stepneys/bib/ss/z/2proof.htm

- Susan Stepney, David Cooper, and Jim Woodcock. “More powerful Z data refinement: Pushing the State of the Art in industrial refinement.” In *ZUM '98: 11th International Conference of Z Users, Berlin 1998*. Volume 1493 of Lecture Notes in Computer Science, pages 284-307. Springer-Verlag, 1998.

- Susan Stepney, David Cooper, and Jim Woodcock. “An Electronic Purse: Specification, Refinement, and Proof”. Technical Monograph PRG-126, Oxford University Computing Laboratory. July 2000.

- Susan Stepney and David Cooper. “Formal Methods for Industrial Products.” In *ZB2000: First International Conference of B and Z Users, York, August 2000*. Volume 1878 of Lecture Notes in Computer Science, pages 374-393. Springer-Verlag, 2000.

- DefStan 00-55, “Requirements for Safety Related Software in Defence Equipment”, 1996. DefStan 00-56, “Safety Management Requirements for Defence Systems”, 1996. http://www.dstan.mod.uk/

- The UK ITSEC scheme. http://www.itsec.gov.uk/

- International Common Criteria Project website. http://www.commoncriteria.org/