Smart cards and card operating systems

Pieter H. Hartel, Eduard K. de Jong Frz
phh@ecs.soton.ac.uk, edejong@integrityarts.com

Declarative Systems and Software Engineering Group
Technical Report DSSE-TR-95-5
November 30 1995

Department of Electronics and Computer Science
University of Southampton
Highfield, Southampton SO17 1BJ, United Kingdom

Smart cards and card operating systems

Pieter H. Hartel * Eduard K. de Jong Frz †

Abstract

The operating system of an IC card should provide an appropriate interface to applications using IC cards. An incorrect choice of operations and data renders the card inefficient and cumbersome. The design principles of the UNIX operating system are most appropriate for IC card operating system design. The actual design that we recognise as UNIX (or any other current operating system for that matter) is not appropriate for IC cards.

1 Introduction

An IC card (or smart card) is a complete computer housed in a piece of plastic the same size as a credit card. In most systems, the computer communicates with a terminal via half a dozen contacts on the surface of the card. Some of these contacts supply the power to the computer. The card currently finds its use in providing off-line security in widely deployed service systems like debit and credit cards. Its future use will also be to provide user centred personal security to Information Technology services as will be develop along side the infamous information super highway [6].

Embedded in a thin, flexible plastic card, an IC card computer has to be small to reduce the risk of mechanical problems. Because of these mechanical constraints, as well as aspects of cost, the current generation of IC cards typically contains only a small 8-bit micro processor, a few hundred bytes of RAM, a couple of Kbytes of ROM and a few Kbytes of non-volatile memory in the form of EEPROM. This small size constrains the freedom in design of software that has to run on an IC card processor, and also places restrictions on the development methods that can be used.

An IC card operating system provides the processing and storage facilities that enable an application provider to implement specific card applications with flexibility and ease. The term 'Application Provider' is used here to refer to the organisation that uses a smart card as part of a product or service, and is responsible for the operations performed by the card. Current IC card operating systems are modelled on DOS-like file systems, with files and directories. A set of commands interpreted by the operating system provides for basic data access operations. Security is supported by access conditions associated with files and directories.

A recently adopted ISO standard [4] based on this data structuring method has been complemented with industry standards such as the joint initiative by Mastercard, Visa and Europay or the mainly European GSM mobile phone industry. Commands are defined in these standards with particular focus on format; command semantics and consistency of the command set are poorly expressed. Actually available operating systems support arbitrary subsets of the standardised commands augmented with proprietary functions. Some operating systems offer the use of cryptographic protocols to satisfy access conditions. IC card operating systems based on the file system model tend to require lengthy sequences of commands to retrieve or modify a data element stored on the card.

We subscribe to a somewhat different view: the card operating system should foremost present the IC card as a processing element rather than as a storage component. This allows the card to communicate with its environment in a client-server fashion. In the client-server approach, card applications can be designed as objects containing persistent data, access methods and security procedures [1], leveraging on the wide experience in developing object based software. Such an operating system provides the application designer with a powerful set of instructions to effectively express the transactional nature of the card communication. Applications implemented in this way will require a few, short commands to implement a transaction.

2 Hardware

One of the most important factors that distinguish an IC card from a normal computer is that the latter has disks as non-volatile memory. In IC cards EE-prom is used instead because it is smaller and can be integrated on chip. EE technology in early IC cards at the begin-

*Dept. of Electronics and Computer Science, Univ. of Southampton, S017 1BJ Southampton, UK, Email: phh@ecs.soton.ac.uk

†Integrity Arts Inc., San Francisco, USA, Email: edejong@integrityarts.com

ning of the seventies made it difficult to store information reliably. The capacity of the non-volatile memory was also small. Over the past 20 years, EE technology has been improved considerably. Both the capacity of memory devices and the reliability of the devices has increased. An interesting observation on the hardware aspects of smart cards is that the price of IC cards has not changed much, the main reason being that the technological advances have not been returned in the form of a lower price but in the form of increased functionality and memory reliability and capacity.

A major technological challenge in building an IC card is to design non volatile memory such that it is small in size and can be rewritten as often as we like. The current state of the art does not allow more than between 1,000 and 10,000 rewrites of a single memory cell. Rewriting memory requires the use of high voltages (15V) which stresses the silicon especially the thin oxide insulation used to implement the memory cell. With the use of additional memory cells and of error correcting codes non-volatile memories can be built that allow storage to be rewritten over 100,000 times. Even with such relative high numbers it remains impossible to use non-volatile memory as we would use memory on our PCs and work stations, First of all, the operating system must at all times be aware which locations it is using and re-using. Secondly the software tools used to build IC card applications must minimise the use of non-volatile memory. It must certainly not be used as temporary storage for program variables and the like.

The advances in non-volatile memory technology represent a significant feat of engineering. Because these advances are hardly visible, they are not appreciated by most of us, who are working with PCs and work stations. It would simply be unthinkable that a memory or a disk would wear out after being written to about 1000 times. Clearly, limitations that the underlying hardware present should be taken into account by the IC card operating system. More interestingly, addressing such limitations should be confined to the kernel of the operating system. In particular the applications programmer should not have to worry about this problem.

3 Software

The hardware development in IC card design has been followed closely by the development of IC card operating systems. Before smart cards appeared, simple memory cards were used, either with magnetic stripes or with memory chips. There is not much that one can do about the interface to a memory card: it can be read and written at various different addresses. There is more freedom in the design of the interface to an IC card. Many systems designers have taken the view that really an IC card is still some kind of a glorified memory card. Addresses have become file names, and the words have become files. In short the interface is that of a file system. This is not only the next logical step up from a memory card to something more accomplished, but it is also inspired by the pervasive need for security. In the minds of many security and access permissions are associated with files. Therefore, to build a system that offers secure and private access to data means to build a file system. File systems come in many varieties, and a prominent one is of course the MS DOS file system. This system has basically been designed for a single user to control the entire file store. This suited early IC card designers nicely, as most early IC card applications required the exclusive use of the card. Two or more applications co-existing on a single card was not considered good business practice, because how can one card issuer trust another? To the ordinary users of IC cards (and other cards for that matter), to have a large collection of cards, all for different purposes, is not attractive.

Presently, a number of IC cards are being offered explicitly qualified as 'multi-function'. The ISO standard also claims to provide support for multiple applications on a single card. Clearly multi-functionality (or multi-programming in the more standard operating system jargon) makes entirely new demands on the IC card operating system, which are difficult to accommodate by an MS DOS style operating system. However the pervading view held by IC card system designers is to stay as closely as possible to tried and tested designs.

The strength of UNIX is that it offers conceptually simple interfaces to the facilities offered by the system through files. In principle, all you need to know about as an application programmer is that everything in UNIX is really a file, or, more accurately, a stream of bytes. Common operations on a stream of bytes are reading, writing, and positioning. Whenever possible, such operations have been given a sensible meaning for I/O devices, which are also accessed as a stream of bytes. Clearly, this uniform and conceptually clean interface comes with a price: The system is rather open and it is difficult to protect one user from another. In fact UNIX uses only a simple set of bits associated with each file to support access restrictions. It seems unlikely that imitating the functionality of the UNIX kernel on an IC card could offer the flexibility and security that we require.

Instead of imitating the result, one should embrace the design principles of UNIX and analyse the requirements of smart cards and the transactional applications they will be used for. Thus, an entirely new operating system should be built specific to the particularities of the use of cards. The result would be a system that offers clear and simple interfaces to the storage being held on the card,

whilst providing tight security.

4 System

From the point of view of an IC card system one should look not only at the card and its software but also at the integration of the card in the system as a whole. In many cases the following simplified client-server view on the system will suffice. This consists of a card (the server), a card reader with some other, larger computer connected to the reader (the client). In principle the card user carries the card from terminal to terminal, carrying out his or her business, and the terminals with their associated computers see only some of the transactions that a user makes. In the case of an electronic purse, or stored value card, for example, one terminal could serve to charge the card with electronic money, and an other terminal would be used to discharge the card, to make electronic payments.

The physical reality is thus that the card needs to communicate with a card reader, which in turn needs to communicate with some larger computer. This physical reality has caused many to think about an IC card as no more than a peripheral, with a peripheral interface: using the card means to connect the peripheral to the computer, which is then in complete control. The responsibility for the dialogue rests for the larger part with the terminal.

Memory cards show this quite clearly as really all we can do is communicate to the memory an address and a read command, so that it will return the data corresponding to that address (If no errors occur). This is how any memory device works. The case of an IC card is somewhat more complicated as the card has its internal interfaces, to the EEPROM, ROM and RAM as well as an external interface to the reader. From the point of view of the reader in a traditional approach, the card is nothing more than a file system device, which we can send a file name, and a read command so that the card may return the requested information (provided no errors are encountered). The terminal would then process the information read and it may subsequently decide to write information back into the file system. Setting aside the added intelligence of using cryptographic techniques to encode information and to enable authorisation to be certified, in this traditional approach the change is really small from a memory card to a 'smart' card.

We believe that a smart card should not be given a file system interface but a security interface. The commands that are issued to the card are not simply an unstructured sequence of commands but a structured dialogue. This consists of opening an application, conducting a dialogue with the opened application in terms meaningful to that application and then closing the application again. The crucial point is that the security is delivered by the protocol that governs the dialogue between card and terminal. When sending an 'open application' command from the terminal to the card, extra information should be sent along, so that the card may check that the terminal, or any communicating party it represents, is indeed what it pretends to be. In this manner the card can allow or disallow the terminal to enter into a dialogue with it. Then, in its answer to the open application command, the card presents its credentials to the terminal, which may decide to accept or reject the card. There are of course more detailed arrangements to be made, as the better cryptographic protocols require more than just one action/reaction step. The principle should be clear: security comes first, and it must be embedded in a clearly structured dialogue.

The client-server approach is widely practiced in distributed computing. Really, nothing is new here, for the card and the terminal form part of a distributed computing system, and even a heterogeneous one. The realisation of this simple fact allows us to import a whole body of knowledge about building distributed systems. The crucial extra factor in the case of building IC card based systems is that they are so tiny. It just would not be sensible to try to run even the smallest of distributed operating systems kernels on an IC card.

5 Standards and their implementation

The principles that we have laid out in the preceding sections are all well and good, but the reality is that IC card systems are not built according to these principles. The situation is actually rather worse as considerable effort is being spent on developing standards for IC cards in all sorts of application areas. Clearly standards are necessary to govern the behaviour of the cards, because otherwise they would all be incompatible and create a real nuisance to the user rather than to make life easier.

Developing standards is not an easy thing, as they are based on tried and tested technology and, more importantly, compromises driven by commercial interests. A novel view is sometimes embraced by standardisation committees but not often so. In the present domain there are a number of important standards all elaborating the file system interface:

ISO 7816-4 Part of the ISO 7816 series of standards on smart cards from ISO JTC1/SC17; after a four year process adopted in March 1995.

EN726 From working group TE9 of the European

Telecommunication Standardisation Institute (ETSI) in Sofia Antipolis (Nice, France) for pay phone and calling cards. Although its adoption predates the ISO standard, the file and directory structure and many of the commands it defines are compatible with ISO.

GSM The subscriber identification module of the GSM mobile telephone network, originating from ETSI. According to the application specific definitions in this standard the SIM smart card provides some of the security to the network. It is further extensively used as a personal storage device holding phone numbers and messages.

EMV The result of a joint working group consisting of Europay, Mastercard and Visa, which published its most recent documents in early 1995. This standardization effort aims to define a core set of ISO commands for use in the banking industry.

In contrast, an application specific standard developed in Europe, EN1546 [2], by TC224 WG10, on stored value cards completely ignores a possible file and record structure on the card and defines a comprehensive set of specific commands. A mapping of the application card data on a file structure is provided in an informative annex.

Most manufacturers of smart cards have implemented operational software for smart cards supporting a file and directory structure. Some of those currently found on the market are:

TB100 Made by BULL CP8 in France this operating system is one of the first to implement directories and files.

MCOS, MPCOS These operating systems are made by GEMPLUS in France and are of more recent date. They provide application specific coding through 'filters' loaded in the EEPROM.

OSCAR This operating system has been created by GIS in England and is used by OKI. It has a proprietary approach to multi application support in providing 'virtual cards',

MFC The Multi Function Card created by IBM in Germany is the most recent implementation of a file and directory structure in smart cards. It supports the basic command set defined in the ETSI standard and provides facilities to include additional application specific commands.

The commands supported by IC cards usually are a subset of the ISO definition, and often support these commands with part of the functionality. All operating systems provide additional commands for maintenance: creating files and directories, allocating memory and specifying access conditions. Most operating systems provide facilities to add executable code to modify the command behaviour or to define additional commands. Despite the multiple application structure officially adhered to in many cases the operating system is created with embedded application specific code. This effectively results in a single application smart card. As the OS code is placed in the card's ROM a new application often requires the definition of a new ROM mask and incurs the costs and delays of silicon fabrication.

An alternative approach has been taken by QC Technology of The Netherlands and Integrity Arts of San Francisco in the design and implementation of the Macsime smart card software technology. Based on a compact operating system kernel application code is loaded in EEprom in the form of applets. The applet is interpreted by the kernel to implement any number of application specific commands as required by the application designer. Portability of the kernel is realised by implementing it in an abstract processor language that is interpreted in the card. In addition to compactness, such an abstract interpreted language allows the inclusion of run-time security checks on the executed code. With this feature the application designer can be allowed to program the required functions in the card without risk to other applications loaded in the same card.

Interoperability between card terminals is provided by loading a 'script' in the terminal that is interpreted to perform the application functions in a terminal as requested by the user. The terminal script is compiled from the same source code as the card applet and therefore guarantees that card and terminal use the same application specific commands.

Interoperability between multiple applications is assured by the use of a single fixed command that is recognised by all applications. This 'open application' command is interpreted by the kernel first to create an execution context for the application, much like the UNIX process, and secondly to initialize the application by calling a procedure defined for that purpose in the applet code. Unlike a UNIX process however, an application can define a number of execution contexts, each designed to support a specific aspect of the application. Execution contexts can be defined for initialization of non-volatile application memory with cryptographic keys or a user name, for key management or for general use. To this end each execution context defines a specific set of access conditions for the application data shared by all contexts. The open application command specifies the particular execution context to be activated for the current terminal and may in addition provide data to (cryptographically) authenticate the activation of the application and execution context at that terminal. The Macsime technology is the first known implementation of an

open operating system for smart cards that securely provides for multiple applications sharing the card memory and processor. The application of formal methods to its design and implementation has been one of the design objectives [3].

6 Methodology

Card operating systems are different from most other operating systems in the sense that the need to provide security should pervade all levels of the system and all possible ways in which it is used. In particular this means that the development methods used in building a card operating system must take security into account right from the start and at all levels. The use of formal methods is the key to success here. An ideal design would consist only of formally specified hardware and software components. Verification procedures would make it possible for an independent body to certify that the implementations used indeed satisfy the specifications. Such verification procedures are described in the ITSEC guidelines [5].

The use of formal methods to support the construction of a fully fledged operating system does present problems that should not be underestimated. Firstly the software industry is perhaps more used to engineering methods using informal designs. Such designs may consist of diagrams and descriptions of processes in natural language. Replacing tried and tested engineering methods by something else will encounter difficulties. Not least people will have to see the benefits of a new method before they may consider embracing it. This is one hurdle that should be overcome. Secondly, the costs associated with the use of formal methods will often be higher than with the use of informal methods. The extra security and reliability naturally carries a price. Thirdly, systems should be designed for change. It is rarely the case that a first design is the ultimate design, so there will be changes. Formal methods can be applied in such a way that changes can be made. This can be accomplished by using adequate support tools that carry out reasoning about a design as well as the generation of support documents and even the code. Planning for change requires planning the development process, such that the development process can be reiterated at a reasonable cost.

7 Conclusions

As single chip computers embedded in a thin plastic card, smart cards only have limited resources for computation and data storage. Advances in silicon technology find their way in smart cards with a delay. They are moderated by the essential non volatile on chip memory and mechanical constraints. However, deployed as extension to the desk top computer the smart card promises enhanced security services available to IT individual system users.

Embedded smart card software is rapidly overcoming the initial preoccupation with hardware memory access procedures. The software evolves into a truly open smart card operating system supporting multiple independent applications. The different card applications provide for a range of security software products.

The operating system of a smart card can be seen as a security critical system component. Distributed in millions, smart cards with their restricted resources support applications where transferred values are mainly protected by the strength provided by the operating software.

A smart card operating system provides the processing and storage facilities that enable an application provider to implement specific card applications with flexibility and ease. Current smart card operating systems are modelled on classical operating systems, with files and directories. We argue that the resulting systems are too general and too large to be effective.

Our view on smart card operating systems regards the smart card as a co-processing element that communicates with a host in client-server fashion. Applications are designed as objects containing persistent data, access methods and security procedures. Our system offers a powerful and secure instruction set that requires a few, short commands to implement a transaction.

We use formal methods to support and verify the design of the system. The basic principle is to design the system in a strictly hierarchical fashion. The functionality at each level in the hierarchy can then be formally specified in a compositional fashion. Compilers and software tools to support the system and application development are built on the basis of the formal specifications. The use of formal methods is feasible and cost effective because a smart card operating system is simpler than a general purpose operating system.

References

[1] E. K. de Jong Frz. Objects in smart cards. In B. Struif, editor, *5th GMD-Smart card workshop*, pages 12.1–12.6, Darmstadt, Germany, Jan 1995. GMD, Darmstadt.

[2] CEN European Committee for Standardization. Identification card systems – inter–sector electronic purse. Draft standard prEN 1546, European Committee for Standardization, Brussels, Mar 1995.

[3] P. H. Hartel and E. K. de Jong Frz. Towards testability in smart card operating system design. In V. Cordonnier and J-J. Quisquater, editors, *1st Smart card research and advanced application conference (CARDIS 94)*, pages 73–88, Lille France, Oct 1994. Univ. de Lille, France.

[4] ISO. *Draft international standard 7816-4.2 "Inter industry command set"*. International Standards Organization, 1993.

[5] ITSEC. *Evaluation criteria for IT security – part 3: Assurance of IT systems*. INFOSEC central office, Brussels, Belgium, version 1.2 edition, 1993.

[6] J. L. Zoreda and J. M. Otón. *Smart Cards*. Arctech House Inc, Norwood, Massachusetts, 1994.