RC 24531 (W0804-034), 9 April 2008
Computer Science

Designing a Secure Smart Card Operating System

Paul A. Karger, David C. Toll, Elaine R. Palmer, Suzanne K. McIntosh, and Samuel M. Weber

IBM Research Division
Thomas J. Watson Research Center
P. O. Box 704
Yorktown Heights, NY 10598, USA

This paper has been submitted to the 13th European Symposium on Research in Computer Security (ESORICS).

Designing a Secure Smart Card Operating System

Paul A. Karger, David C. Toll, Elaine R. Palmer, Suzanne K. McIntosh, and Samuel Weber

IBM Thomas J. Watson Research Center,
P.O. Box 704, Yorktown Heights, NY 10598, USA
{karger|samweber}@watson.ibm.com, {toll|erpalmer|skranjac}@us.ibm.com

Abstract. The ever-increasing computational power of smart cards has made them feasible for applications like electronic passports and military id cards. However, these applications require a secure smart card operating system.
In this paper we argue that smart card platforms pose additional security challenges than traditional computer platforms. We discuss our design for a secure smart card operating system, called Caernarvon, and show it addresses these challenges, such as secure application download, protecting cryptographic functions from potentially malicious applications, resolving covert channel issues, assuring both security and data integrity in the face of arbitrary power losses, etc.

1 Introduction

Credit-card sized computers, called smart cards, are increasingly available and their computing ability is increasing at a fast rate. Their ability to perform computation, instead of just storing data like standard credit or memory cards, makes them suitable for applications where card readers need authentication before accessing card data, or where a transaction has to occur without access to a central authority. Currently, smart cards are being used for both civilian and military identification cards and for electronic passports. For a good overview of smart card technology in general, see [27].

As the prime motivations for the use of smart cards are identification, authorization or encryption, it is crucial that an appropriate amount of trust be established between different applications executing on the same card. The lack of a trusted secure operating system for smart cards has resulted in users having a "necklace of cards", each one hosting a different application. The Caernarvon project was started to create such a secure smart card operating system. An overview of the Caernarvon system can be found here [36].

In this paper we argue that smart cards provide security challenges as well as opportunities for operating system development. Two major areas of the security aspects of the Caernarvon operating system have already been published — the security model [30] and the authentication protocol [31]. However, in pursuing our goals, we found that it was like peeling an onion: when tackling each security problem we found a deeper one behind it. This paper will discuss both these smart-card specific challenges and the solutions developed by the Caernarvon team.

1.1 Background – Smart Cards

Historically, smart card chips have been very weak computers by today's standards. They typically have had no more than 128K of memory, slow 8-bit processors, and no memory protection or supervisor state. However, the latest generations of smart card chips now have 16-bit or 32-bit processors and increased memory capacities, although RAM memory still remains very restricted at typically only 8K to 16K bytes.

Due to physical limitations, smart cards are unable to contain their own power source and instead receive power from their readers. This power can be removed at any time, and this is even a common event as users pull their cards out of the machine. Cards cannot retain enough charge to complete any in-progress memory writes, and the result is that the target locations can become arbitrarily scrambled. Attackers can potentially make use of this fact by removing power either just before or during critical writes.

Smart cards have persistent memory to hold data when power is unavailable. Usually this is EEPROM, but can be flash as well. Both of these memory types have limitations on the number of write cycles (the number of times a given cell can be re-written), to between 500,000 and 1,000,000 cycles for modern EEPROM, and perhaps as few as 100,000 cycles for flash. Memory cells that are written too many times are subject to failure, so that the data values are not stored or retained correctly. However, chip vendors are significantly raising these write cycle limits, so the problems are becoming less severe.

Another problem is that the write cycle time for EEPROM and flash is of the order of 4 to 6 milliseconds, and many operations require multiple memory writes. This increases the probability that a user will remove power during a write.

The design of a smart card OS is constrained by pre-existing specifications, of which the most important is ISO 7816–4 [18]. This specification defines the communication protocol between a card and its reader using Application Protocol Data Units (APDUs). There are two kinds of APDUs — command APDUs that go from the reader to the card, and response APDUs that return results. Of particular note is the "SELECT" APDU which instructs the card to execute the application which is specified by the command's argument. (See Section 2.2.)

Although 7816–4 does not specify the internals of a smart card operating system, it has implications that affect the file system design. Smart card filesystems are hierarchic, like those of many conventional operating systems. However, the smart card industry uses its own unique terminology for many of the common components. The unique root of the file system is called the "MF" (Master File). DFs (Dedicated Files) act as directories, while EFs (Elementary Files) are basic data files. The MF and each DF can contain EFs and DFs. Each MF, DF and EF is given a 16-bit file id, rather than a text name as in most other operating systems. Each application is associated with a DF (not a file, like one might expect), but not vice-versa.

1.2 Feasibility and Related Work

When the Caernarvon project to build a highly secure operating system for smart cards began, the first question was "Would it be feasible at all to build such a system?". Early smart card chips did not have adequate hardware support for security, such as separate supervisor and user states and memory protection. The introduction of the NXP (formerly Philips) Smart$XA2$ chip met those needs. Karger, Toll and McIntosh [23] discuss these hardware requirements in much more depth. Furthermore, the memory management approach of the Smart$XA2$ was very similar to that of the DEC PDP–11/45 minicomputer [26].

The similarity to the DEC PDP–11/45 was very significant, because the very first high assurance security kernel operating system was developed by Lee Schiller [32] at the MITRE Corporation for the PDP–11/45. The Schiller kernel was only 900 lines of higher level language - a very small amount of code that could clearly fit into a smart card with limited memory. Also, the Schiller kernel supported a very simple hierarchical file system with files named by small integers in a manner not dissimilar to that used by smart cards.

2 Smart Card Applications Code and Security

2.1 Security Policy

The Caernarvon system builds on previous work on mandatory security policies to provide multi-level security within the system. Security within the Caernarvon system is enforced using a mandatory security policy [30] that is based on modifications of the Bell and LaPadula secrecy model [5] and the Biba integrity model [6]. The security policy is built around the concept of Access Classes. Each Access Class (AC) is either an Organizational Access Class (OAC) or a Universal Access Class (UAC). Details of an early design for UACs and OACs can be found in [19].

An OAC consists of a type field (which specifies if the OAC is a secrecy OAC or an integrity OAC), an organizational identifier (OID), a sensitivity level, and an optional set of categories. The assignment and meaning of the sensitivity level and categories in the OACs for a given organization are completely controlled by that organization—there can be no collision between categories with the same value chosen by other organizations because OIDs are unique. Categories correspond to internal groups, departments or other organizational structures. The organization may alternatively regard the categories as a formalized "need to know" list. Note that, in the case of an integrity OAC, the sensitivity level is replaced by an integrity level; the Caernarvon modification to the Biba model is to specify that the integrity level is an integer in the range 0-7, where 0 implies normal low integrity, and a value in the range 1 to 7 indicates a higher integrity level that is the level n of a Common Criteria [12] evaluation at level EALn.

A UAC consists of a set of OACs, each of which has the same type (that is, secrecy or integrity). Every OAC in a UAC must have a distinct organizational ID.

There are well defined rules for the comparison of Access Classes—these are discussed, together with many details of the Caernarvon security policy, in [21].

Authentication and Authorization Enforcement of a meaningful security policy requires that there be a secure mechanism to ensure that the use of the desired ACs is valid and correct. In Caernarvon, each time the smart card is inserted into a reader, the system and the outside world perform a two-way authentication, which verifies each party's identity to the other and then sets up a secure channel. This authentication must be performed by the operating system itself and not by an application, so that the operating system is guaranteed, and can guarantee to others, that the authentication has been correctly completed. The operating system then knows with high assurance the identity of the user, which is typically the host system behind the smart card reader. The operating system can use this knowledge to safely grant the user access to files and other system objects; conversely, it can also use this knowledge to ensure, with high assurance, that a user is denied access to anything he is not authorized to see or use. The full description of this authentication protocol is beyond the scope and space limitations of this paper; more information is available in [31].

Handling of Trusted Processes The Caernarvon security model [30] incorporates the concept of *trusted processes* or *guard processes* directly in the model, rather than as exceptions to the model as in most previous mandatory access control schemes. In particular, the Caernarvon model explicitly requires that a guard process have a high integrity access class, and that it be allowed to operate over a specified range of access classes, much as in the GEMSOS system [29]. The integrity level of the trusted code, together with the range of allowed access classes would be specified and digitally signed by the Common Criteria certifying body that certified the trusted application. The idea of using a range of access classes originated in an early design [34] of the US Air Force Strategic Air Command's secure packet switched network SATIN IV, later renamed SACDIN.

2.2 Application Selection Control

The Caernarvon system consists of a kernel, running in supervisor mode, and user-mode applications. These applications are stored in the file system by files marked as executables. In a multi-application smart card, with multiple access classes specifying different levels of access to files, the security policy applies equally to running a program file as it does to any other file access—insufficient access to a file implies that the program in that file cannot be run in the current session.

The Caernarvon kernel contains a component, the APDU Dispatcher, which examines every incoming command, and determines if it is an APDU to be handled by the system, for example those for the authentication process. Other APDUs (those that are not explicitly handled by the system) are passed to the currently selected application.

The SELECT APDU is handled directly by the APDU dispatcher. Assuming that the access class(es) for the current session, selected during authentication, allow the reader to execute the requested program, it is started by the APDU dispatcher. Subsequent APDUs are sent to this application, which reads the commands and returns responses through communications SVC calls. If, while the application is running, another SELECT APDU is received to run a new application, then this APDU is intercepted by the APDU dispatcher and is *not* passed to the application. Instead the current application is notified that it must terminate. If the current application continues to run and attempts any communication with the outside world, the program is forcibly terminated.

2.3 Cryptographic Facilities

Many modern smart cards contain cryptographic hardware to implement (or at least significantly ease the implementation of) algorithms such as DES and triple–DES, AES, RSA, DSA and ECC. Where such hardware is not provided, any cryptographic algorithms must be implemented entirely in software. Smart cards are physically small, with little protection from attacks of various types, including side-channel attacks or even direct probing of the memory. These attacks are well documented in the literature, e.g. [25, 1]. Protection of cryptographic functions against attack attack requires a combination of hardware counter measures and specific software coding techniques. For example, if a DES engine was in use by one process, and is now to be re–assigned to another process, it is necessary to ensure that any keys loaded into the hardware by the first process cannot be read by the second. An example of a poor implementation is the attack on DSA if the nonce k is not calculated correctly each time [33].

The Caernarvon system was designed from the very beginning to allow application programs to be written by anyone—indeed, the design allows entirely untrusted or even deliberately malicious code to be run. This design aim has repercussions, in that poorly written applications can, either accidentally or deliberately, directly or indirectly, leak cryptographic information (primarily keys). The Caernarvon system makes use of the processor's hardware protection to ensure that the crypto hardware can be accessed only in supervisor state—that is, applications have no access to the cryptographic hardware. Hence the kernel provides crypto functions, accessed by SVCs, to implement the various cryptographic algorithms for applications. This removes the possibility of an application mis-using the crypto hardware, for example "accidentally" seeing DES keys that belong to another process, or running the hardware in such a manner as to cause it to leak information via side-channels.

Cryptographic algorithms within applications may be poorly written, or not optimized for the particular processor on which they are executing. The result of this could be information leakage by means of side channels or from poor software design. Conversely, the cryptographic code in the Caernarvon kernel is carefully written to provide all possible protection from attacks and information leakage. The cryptographic code has completed an evaluation under the Common Criteria at EAL5+ (and would need to be re–evaluated at EAL7 when Caernarvon is evaluated). Thus if applications use the crypto functions in the kernel, they get the best possible protection for crypto on this processor. Using the kernel crypto functions also saves time and effort in implementing the application, and avoids duplicated code within the smart card.

Another potential problem with cryptographic functions, particularly in smart cards with their susceptibility to attack, is the key management. If the application handles the key itself, it may inadvertently leak information (for example, some bits of the key) by such simple operations as copying the key from one memory location to another. Further, there is nothing to prevent a malicious program from deliberately leaking the key to outside the smart card.

Caernarvon provides secure key management facilities within the kernel. Keys can be loaded into the card by the kernel, so that the application never see the key; the application refers to the key by a name of its choosing. The keys are effectively stored in the file system with file IDs for names and hierarchical file paths, the same as for regular files. This avoids covert channel problems that could arise in the names of keys, if the keys were stored in a flat file system. However, the key names are a separate name space from the file names, and, to ensure security of the key, these key "files" cannot be accessed as regular files. An application, when it wishes to use one of these secure keys, issues a request quoting the key's file path, and the operating system returns a key handle; the subsequent crypto requests use this handle to specify the key. All key operations are then kept within the kernel, where appropriate measures can be taken to protect against attacks. In addition, keys can be marked, for example, to be encryption keys or to be signing keys; the kernel can then prevent a signing key from being used for encryption, or vice versa. This prevents certain cryptographic weaknesses where a key is used for more than one purpose.

Unfortunately, a few smart card standards (such as the Global Platform standard [4]) require that the keys be visible to applications (or in the Global Platform case, the application security domain). To satisfy this requirement, Caernarvon also supports a "raw" key mode, where the keys are handled entirely by the application. Alternative crypto functions are provided in the kernel that have their keys supplied in buffers, instead of using a handle as is the case for a secure key. While applications may find this mode a necessity, its use is strongly discouraged, since the secure kernel cannot ensure any security for these raw keys.

Certain smart card application standards specify weak cryptographic protocols, such as the standards for cryptography for GSM phones that have been broken [3]. Currently the Caernarvon kernel contains implementations of only known secure, standardized, crypto algorithms, such as DES, AES and RSA. If implementations of weak algorithms were added to Caernarvon, the algorithm would still be weak—there is no way a high security system can magic a weak algorithm into a strong one. The only way to avoid such problems is to not devise standards with weak cryptography.

Another problem that can arise is that a program can develop its own cryptographic code, for example to implement an algorithm devised specially for that application. Running such code on top of a high security kernel provides no guarantee of the quality of the implementation of the cryptography, including particularly immunity to side channel attacks. Again, the only way to avoid the problem is to design the application to use only the strong crypto (and secure key management) provided by the Caernarvon kernel.

3 Security Design Challenges

3.1 File System

The Caernarvon system implements a smart card file system, as described in Section 1.1. Besides the inherent challenges caused by the specification and hardware restrictions, the filesystem must also conform to and enforce the system's security policy. Also, there must be a quota mechanism and support for memory-mapped files. We will now describe these challenges and our resultant design in more detail.

File System Integrity It is imperative to maintain the integrity of the file system in a smart card, even when the power source is unexpectedly removed, as described in Section 1.1. In Caernarvon, the functionality is divided into two separate components:

1. the Persistent Storage Manager (PSM), which handles the physical memory blocks, and ensures their integrity. The PSM is described in the section 3.2.
2. the File System, which handles the logical file system structure within memory blocks provided and maintained by the PSM.

This avoids the necessity of maintaining the integrity of the persistent storage within the code that is controlling the logical file structure.

ISO7816 File System In addition to the standard MF, DFs and EFs, the Caernarvon system extends ISO 7816 by defining another file type, an "XF" (Executable File), which are EFs that contain an executable program.

As an implementation optimization, the MF and DFs do not keep a table of the file names that they contain; instead, each DF and EF in the system has a pointer to its parent. This means that file system searches require examining every file in the system; however, since the amount of persistent memory is limited, there can be only a small number of files in any given smart card, so this extended search is not a problem. Obviously, this algorithm does not scale to large memories with lots of files. However, if there was a large amount of memory there would be no necessity to shrink directories in this manner.

ISO 7816-4 defines certain types of record structure files; the Caernarvon kernel does not implement these (this is left as a user program function); in the Caernarvon file system, all files are treated as unstructured sequences of bytes.

Although this is not required by ISO 7816, Caernarvon requires that the file ids be unique within a DF or the MF.

DFNames ISO 7816 defines the concept of a DF Name. These names are used to select executable programs from outside the card, without having to know the numeric file IDs. They consist of a string name assigned to the DF containing the application. The DF Names are unique to the card, and, (as defined in ISO 7816) constitute a global address space.

Global address spaces cause two different operational problems. First, if two different application developers happen to choose the same DFName, then the first such name loading onto a particular card will win. Since ISO 7816-4 assumed all applications would be preloaded onto the card, this was never a problem. However, once you have multiple application providers downloading applications to a card after the card has already been issued, the name collision problem can become serious. Second, such name collisions could be used as a covert channel to bypass mandatory access controls.

Caernarvon, to avoid these problems, stores the DF Names prepended with the current AC chosen during the authentication for the current session. This makes the DF Name space into a name space that is private to the current access class. The ISO 7816 rules for DF Names are then applied within each access class, rather than system wide. Within an access class, DF Names must be unique, but the same DF Name may be repeated in a different access class.

Short File Identifier ISO 7816-4 defines the notion of a Short File Identifier, or SFI, whereby a file can be specified by a single 5-bit identifier. The standard leaves the actual implementation of SFIs to the application, and existing implementations of SFIs differ substantially.

Just like DFNames, SFIs could form a global address space, and have the same collision and covert channel implications. In Caernarvon, SFIs are implemented as local to a DF (that is, to a current directory). Thus there are multiple local SFI address spaces, which removes the covert channel.

Access Classes and the File System In order to reduce memory usage, files (EFs and XFs) do not have associated access classes while DFs may but are not required to. DFs without an access class of their own, and EFs and XFs, inherit the access class of their immediate parent DF. Note that the MF has secrecy access class System Low, and the secrecy access classes are monotonically increasing as the file system tree is traversed away from the MF. Similarly, the MF has integrity access class System High, and the integrity access classes are monotonically decreasing as the file system tree is traversed away from the MF.

Access to a DF, and hence to the files within it, is controlled by comparing the current access class of the process (for example, the AC chosen during system authentication) to the access class of the target file.

A program, if it has appropriate access to a file, may change the access class of that file, for example to raise its secrecy level or to lower its integrity level. In either case, it is quite likely that, having changed the access class of the file, the program no longer has access to the file. In this case, any open file handles for the file in question are marked, and then the program can perform no more operations such as read and write using the file handle until it has closed the file and attempted to re-open it. If the program no longer has access to the file then the re-open will fail.

This facility to change the access class of a DF (and hence of all the files within it) can be used to move data from one access class to another. Thus if a program at AC a wishes to move a DF (also at AC a) to AC b then the program must change the DF to the AC $a+b$. Note that this is quite legitimate - a program may always change a file to a higher secrecy level. Having done this, the program, which is still at AC a, no longer has access to the DF. At this stage, a special guard process must be run; this is an evaluated application that has been certified as fit to perform the downgrade of secrecy level $a+b$ to a. This guard program would verify that it is indeed legitimate to downgrade the secrecy of the DF, and if all is well, change the AC of the DF to b.

Quota In order to protect against both inadvertent and malicious denial of service attacks when one application takes all the persistent storage on the card, Caernarvon provides a quota facility. This also enables the card issuer to control (and charge for) the amount of space on the card used by each application or, possibly more importantly to the card issuer, by each application provider. Covert channels whereby a Trojan horse could signal by either allocating all memory or freeing some are prevented.

The algorithms used for the quota allocation are basically those from Multics [37, section 3.7.3.1], except that in Caernarvon, due to the severe limitations of the platform on persistent storage space, we are more careful about including system overhead such as control blocks in the quota calculations.

Each DF may (but need not) have a quota; if the DF does not have its own quota then that DF and all the files within it are charged against the quota of the nearest parent DF that does have a quota. When a new top-level DF is created for an application, then that DF would normally be allocated its own quota. The application can quite legitimately give some of its quota to a DF below its own top-level DF. If a DF is moved from one AC to another (as described above), then the quota occupied by that DF and all the files within it is also moved to the new parent DF of the DF that has been moved.

Discretionary Security Policy - Capabilities The Caernarvon system, in addition to the mandatory security policy, provides *capabilities*—these are discretionary security policy rules that may be associated with an individual executable program (an XF). These capabilities take two forms:

1. a bit array that specifies whether that program is, or is not, permitted to issue certain SVCs or groups of SVCs. For example, there is a special, evaluated, Admin Application issued with the system that is used for the administration of (in particular, the creation of) Access Classes and top-level DFs for applications. This program uses certain special SVCs for the administration of ACs; the capability bit for this group of SVCs is unset for *every* other XF in the system, so that no other program can issue those SVCs.
2. there can be special access rules to allow or forbid access to individual files by this program. Note that any access granted by these rules is still subject to the mandatory security policy—that is, a capability rule cannot grant access to a file when the access class comparison would forbid it.

It is important to note that these capabilities are not a fully general capability system, as defined by Dennis and Van Horn [15]. In particular, Caernarvon capabilities cannot be passed from one process to another.

3.2 Persistent Storage Manager - PSM

In the Caernarvon system, the physical blocks of storage are managed by the Persistent Storage Manager (PSM). While the principal client of the PSM is the File System, the PSM is also used by the Access Class Manager and the Key Management system.

A solution to the errors that ensue when power is removed during a write is to ensure that all memory transactions, for example a request to extend a file and update its contents, be treated as a single atomic operation. That is, the entire transaction must be completed in its entirety, or not performed at all. There is a *back-trace buffer* where, when memory is to be updated, the old values are stored before the new data is written. This is done for every step of the operation - the backtrace buffer is cleared only when the entire transaction is completed. When the card is powered-up, if the backtrace buffer is not empty, the items in the backtrace buffer are restored one-by-one, in the reverse order to which the original steps were performed. In this case, when the backtrace buffer has been emptied, the state of the memory is as if the transaction had never been started.

The PSM takes two measures to prevent or recover from memory corruption due to the cells wearing out. The first is that, on every write to persistent memory, once the write is completed and before control is returned to its caller, the low-level code that wrote the data compares the updated contents of the memory with the data in the caller's buffer (in RAM). If a mismatch is detected, an error is returned; in this case, the data that was to be written is still available in the buffer. The second protective measure employed by the PSM is to place a checksum on every memory object under its management, including any control blocks or descriptors that define the memory blocks. This checksum is verified on read operations, and hence memory failures can be detected—an attempt is then made to recover the lost data byte(s). Once a memory error is detected, the block in question is marked as bad, and the data is re-written to a different location.

3.3 Application Download

A primary design aim of the Caernarvon system is to allow for the secure download of applications in the field. The download process is to be under the control of the card issuer—it is up to the card issuer to allow or forbid the download of applications, and when download is permitted, to control which organizations are or are not allowed to install their applications on the card and the amount of file quota to be allocated to each organization. It should be noted that, in the context of download, the term "application" is not limited to just XFs; it may also encompass DFs, EFs, file quota, keys, etc.

The download process can be divided into two main steps:

1. creation of access classes for organizations that currently are not present on the card, and allocation of file quota.
2. download of application files, including executable programs, for an organization that is present on the card.

Obviously, download of an application for a new organization requires the completion of both of these steps.

Creation of Access Classes The creation of a new access class is a tricky operation on a smart card, because the card is physically in the possession of an end user who may not be privileged to create access classes. The smart card also does not have a system administrator or security officer who can perform such operations. Requiring the card holder to carry the card back to the card issuer to have access classes installed would be unacceptable to most customers.

Instead, the Caernarvon operating system includes secure cryptographic protocols to perform the following operations:

1. create the Organizational Access Class (OAC).
2. create the Universal Access Class (UAC).
3. create the necessary top-level DF associated with the new UAC, and set its allocated quota.

A full description of these protocols is beyond the scope and space limitations of this paper, and will be the subject of a future paper.

File Download Once an organization has been authorized to be present on a Caernarvon card, that is, once any necessary access class(es) have been created, then that organization may download such files as it needs, subject to the file space the quota imposed by the card issuer.

A file is downloaded simply by authenticating at the appropriate access class, running a program to create any required DFs and EFs, and writing the appropriate data to those files. The Caernarvon system includes a utility program, the ISO7816 application, which implements many of the APDUs specified in ISO 7816-4 and which can be used under any access class to perform such file operations. An executable file, once it is downloaded, must be "activated" to convert the file from an EF to an XF.

The card issuer may wish to restrict the programs that are run. For example, only approved applications or Common Criteria evaluated applications might be allowed. In the former case the application would have a signature from the card issuer, while in the latter case a signature from the certification agency would exist. If such restrictions are in place, the Caernarvon kernel will validate the necessary restrictions when the activate operation executes.

3.4 Side Channel Issues

Just as other smart cards, the Caernarvon operating system could be subject to power analysis, RF analysis, or timing attacks against its cryptographic mechanisms. We made use of standard techniques for addressing these attacks, as described in [8, 1, 9].

We also had to consider the implications of side channel attacks on random number generators (RNGs). Common Criteria evaluation of hardware random number generators requires consideration of possible hardware failure modes. As a result, Germany requires [16] that the random numbers from a hardware RNG be tested prior to use. However, the act of testing the random numbers could easily leak the values of the numbers via power or RF analysis. To overcome this problem, the Caernarvon operating system includes a new kind of RNG that will be the subject of a future paper. At present, the only description can be found here [10].

3.5 Chip Initialization

Chip initialization is another design challenge that we had to address. Details can be found in Appendix A.

4 Common Criteria Problems Solved

During the Caernarvon development, we ran into several issues related to the Common Criteria itself. These included composite evaluation, lack of appropriate protection profiles, and dealing with testing.

4.1 Composite Evaluation

The purpose of composite evaluation is to allow an upper level product (such as a smart card operating system) to benefit from the results of an evaluation of a lower level product (such as a smart card chip) without requiring a full re-evaluation. Composite evaluation was not defined under version 2.3 of the Common Criteria [12], but rather by supplemental documents. These documents limited the amount of information available to the software developers and evaluators about the underlying smart card chip.

Limiting the interaction between the software developers and the hardware evaluators, often leads to security vulnerabilties being completely missed in the composite evaluation. In [22], we have described a number of problems with such an approach. Since that time, the Common Criteria version 3.1 [13] has been published that includes provisions for composite evaluation. However, smart card composite evaluation does not use the techniques specified in version 3.1, but rather has a new supplemental document [14] for smart cards only that does not appear to address the problems that we encountered and reported in [22]. A more complete analysis of the new smart card composite evaluation approach is beyond the scope of this paper.

4.2 Protection Profiles

One major problem we encountered is that none of the smart card protection profiles available at the time adequately described the requirements of the Caernarvon operating system. In particular, all the protection profiles assumed that all smart card software was fully trustworthy and installed prior to the issuance of the card to end users. To overcome this lack of appropriate protection profiles, Helmut Kurth (of atsec information security) helped us develop a security target that addressed the many new security requirements we faced.

4.3 Testing

Thorough testing is well-recognized as an important part of the development process for secure software and is required for Common Criteria certification. Full code coverage testing (that is, ensuring that all code is executed during testing) is often performed. Branch coverage, ensuring that both branches of every conditional are executed, should be the minimum standard of testing for high-assurance code. Ideally, one would like to make sure all possible execution paths are tested. This clearly can't be done for any code with a non-trivial number of conditional tests, as the number of execution paths is either exponential in the number of tests, or infinite (in the case of unbounded loops).

Even if full path testing were accomplished, this would still not be sufficient to prove system security, as Trojan horses have been demonstrated that exploit data flow, not just control flow. Testing is not a panacea, but it does serve as an independent check upon the design and code of the system.

Test strategies for smart cards must also consider that the persistent memories (EEPROM or flash) will fail if too many write cycles are carried out. The limits on write cycles have gotten much better in recent years, so that normal pre-deployment testing is not likely to be a problem, because many of the test cases will abort before actually performing a write operation. However, if the card issuer or any servers with which the card communicates require too many self-test sequences that involve writing to persistent storage, then the memories could begin to fail.

5 Applications and Impact of the Technology

A number of applications could benefit from a commercially available high assurance smart card operating system. In general terms, those applications have data or software from multiple parties co-residing on the same card, and require some level of data sharing between the parties. The trust relationship of those parties ranges from friendly (e.g. allies and business partners) to mistrustful (e.g. coalitions) to hostile (e.g. competitors or military combatants). The threats addressed range from honest mistakes in software to attacks by financially-motivated cardholders to industrial espionage to comprehensive logical and physical attacks by hostile adversaries and insiders. Below is a list of sample applications:

- an intelligent electronic passport issued by one government, with electronic entry/exit timestamps added by other governments, some potentially hostile
- one card for access to multiple security levels of government networks, ending the "necklace of cards"
- a corporate/school campus card, with multiple application providers for copiers, vending machines, canteens, public transit, and building and room access
- a frequent traveler ID card with loyalty software from multiple airlines, rental car companies, hotels, and restaurants
- an ID card for coalition military forces for access to physical and logical services
- a subscriber identity module for mobile phones to hold credentials for multiple institutions, e.g. financial institutions, governments, and phone service providers

There are roadblocks hindering the commercialization of a high assurance smart card operating system. First, the development, evaluation, and commercialization of such a system could not be described as "low hanging fruit." Significant investment in time and funding is required by multiple institutions. The skills required cross several domains, and are not typically found in any one organization: hardware design, operating system design, formal methods, software and hardware testing, vulnerability analysis, and evaluation methodology. Second, some existing smart card application specifications have mandated protocols that preclude a high level of security. For example, the electronic passports specified by the International Civil Aviation Authority (ICAO) require the use of weak cryptographic authentication protocols [24], and while the protocols of the Federal Employee Personal Identity Verification (PIV) program are cryptographically strong, they also require some very sensitive information to be transmitted in unencrypted form [20].

Although the Caernarvon OS is not commercially available, the technology created as part of the project has had impact in multiple areas. The privacy-preserving authentication protocol is now part of the European CEN standard for application of digital signatures in smart cards [2]. The Caernarvon cryptographic library has been certified under the Common Criteria at EAL5+ in Germany [7]. The mandatory security policy is a fundamental part of the Fuzzy Multi-Level Security Model [11] for System S, a large-scale, distributed, stream processing system for analyzing large amounts of unstructured data [38]. The mandatory security policy has also been incorporated into the Simple Linux Integrity Module (SLIM) in a Trusted Linux Client [28]. Lastly, the lessons learned from a developer's perspective were documented in [35].

6 Conclusion

In summary, the Caernarvon operating system project has shown the feasibility of building smart card systems with much higher levels of security. It is possible to support downloading of applications from multiple sources that may be mutually hostile, yet these applications can be prevented from interfering with each other or with the underlying operating system.

Accomplishing these goals required reconsideration of many of the traditional smart card software practices, as well as solving many security problems that are not present in larger-scale computers.

7 Acknowledgements

The Caernarvon project involved work by a number of people in addition to the authors of this paper, and we wish to acknowledge the contributions of Vernon Austel, Ran Canetti, Suresh Chari, Vince Diluoffo, Jonathan Edwards, Günter Karjoth, Gaurav Kc, Rosario Gennaro, Hugo Krawczyk, Mark Lindemann, Tal Rabin, Josyula Rao, Pankaj Rohatgi, Helmut Scherzer[1], and Michael Steiner from IBM, Helmut Kurth of atsec, Hans–Gerd Albertsen, Christian Brun, Ernst Haselsteiner, Stefan Kuipers, Thorwald Rabeler, and Thomas Wille of Philips Semiconductors (now NXP), Wolfgang Reif, Georg Rock and Gerhard Schellhorn of the University of Augsburg, Germany, Axel Schairer and Werner Stephan of the German Research Center for Artificial Intelligence (DFKI), and Stefan Wittmann of the Bundesamt für Sicherheit in der Informationstechnik (BSI) in Germany.

References

[1] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. The EM side-channel(s). In *Cryptographic Hardware and Embedded Systems - CHES 2002*, Lecture Notes in Computer Science, Vol. 2523, Springer Verlag, pages 29–45, Redwood Shores, CA, 13-15 August 2002.

[1] now with Giesecke & Devrient

[2] Application interface for smart cards used as secure signature creation devices – part 1: Basic requirements. CWA 14890-1, Comité Européen de Normalisation, Brussels, Belgium, March 2004. `ftp://ftp.cenorm.be/PUBLIC/CWAs/e-Europe/eSign/cwa14890-01-2004-Mar.pdf`.

[3] E. Barkan, E. Biham, and N. Keller. Instant ciphertext-only cryptanalysis of GSM encrypted communications. Technical Report CS-2006-07, Technion - Israel Institute of Technology, Haifa, Israel, 2006. `http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/2006/CS/CS-2006-07.pdf`.

[4] S. Z. Béguelin. Formalisation and verification of the globalplatform card specification using the B method. In *Construction and Analysis of Safe, Secure, and Interoperable Smart Devices, Second International Conference, CASSIS 2005*, pages 155–173, Nice, France, 8–11 March 2005. Lecture Notes in Computer Science, Vol. 3956, Springer.

[5] D. E. Bell and L. J. LaPadula. Computer Security Model: Unified Exposition and Multics Interpretation. ESD–TR–75–306, The MITRE Corporation, Bedford, MA, HQ Electronic Systems Division, Hanscom AFB, MA, June 1975.

[6] K. J. Biba. Integrity Considerations for Secure Computer Systems. ESD–TR–76–372, The MITRE Corporation, Bedford, MA, HQ Electronic Systems Division, Hanscom AFB, MA, Apr. 1977.

[7] Certification Report for Tachograph Card Version 1.0 128/64 R1.0 from ORGA Kartensysteme GmbH. Technical Report BSI-DSZ-CC-0205-2003, Bundesamt für Sicherheit in der Informationstechnik, Bonn, Germany, 22 August 2003. `http://www.bsi.de/zertifiz/zert/reporte/0205a.pdf`.

[8] S. Chari, C. Jutla, J. R. Rao, and P. Rohatgi. Towards sound countermeasures to counteract power analysis attack. In *Proceedins of Crypto '99*, Lecture Notes in Computer Science, Vol. 1666, Springer Verlag, pages 398–412, Santa Barbara, CA, August 1999.

[9] S. Chari, J. R. Rao, and P. Rohatgi. Template attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002*, Lecture Notes in Computer Science, Vol. 2523, Springer Verlag, pages 13–28, Redwood Shores, CA, 2002.

[10] S. N. Chari, V. V. Diluoffo, P. A. Karger, E. R. Palmer, T. Rabin, J. R. Rao, P. Rohatgi, H. Scherzer, M. Steiner, and D. C. Toll. Method, apparatus and system for resistence to side channel attacks on random number generators. United States Patent Application No. US 2006/0104443A1, Filed 12 November 2004.

[11] P.-C. Cheng, P. Rohatgi, C. Keser, P. A. Karger, G. M. Wagner, and A. S. Reninger. Fuzzy multi-level security: An experiment on quantified risk-adaptive access control: Extended abstract. In *Proceedings of the IEEE Sympsium on Security and Privacy*, pages 222–227, Oakland, CA, 20–23 May 2007. IEEE Computer Society.

[12] Common Criteria for Information Technology Security Evaluation, Parts 1, 2, and 3. Version 2.3 CCMB2005-08-001, CCMB2005-08-002, and CCMB2005-08-003, August 2005. `http://www.commoncriteriaportal.org/public/expert/index.php?menu=2`.

[13] Common Criteria for Information Technology Security Evaluation, Parts 1, 2, and 3. Version 3.1 CCMB–2006–09–001, CCMB–2006-09-002, and CCMB2006-09-003, September 2006. `http://www.commoncriteriaportal.org/thecc.html`.

[14] Composite product evaluation for smart cards and similar devices. Technical Report CCDB–2007–09-001, September 2007. `http://www.commoncriteriaportal.org/files/supdocs/CCDB-2007-09-001%20-%20Composite%20product%20evaluation%20for%20Smartcards%20and%20similar%20devices%20v1-0.pdf`.

[15] J. B. Dennis and E. C. Van Horn. Programming semantics for multiprogrammed computations. *Commun. ACM*, 9(3):143–155, Mar. 1966.

[16] Functionality classes and evaluation methodology for physical random number generators. AIS 31, Version 1, Bundesamt für Sicherheit in der Informationstechnik (BSI), Bonn, Germany, 25 Sept. 2001. `http://www.bsi.bund.de/zertifiz/zert/interpr/ais31e.pdf`.

[17] IBM 4764 Model 001 PCI-X Cryptographic Coprocessor. Data Sheet G221-9091-05. `http://www-03.ibm.com/security/cryptocards/pdfs/4764-001_PCIX_Data_Sheet.pdf`.

[18] ISO 7816-4, Identification cards - Integrated circuit(s) with contacts - Part 4: Interindustry commands for interchange, First edition. ISO Standard 7816-4, International Standards Organization, September 1995.

[19] P. A. Karger. Multi-Organizational Mandatory Access Controls for Commercial Applications. RC 21673 (97655), IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 22 February 2000. `http://domino.watson.ibm.com/library/CyberDig.nsf/home`.

[20] P. A. Karger. Privacy and security threat analysis of the federal employee personal identity verification (PIV) program. In *Proceedings of the 2nd Symposium on Usable Privacy and Security*, pages 114–121, Pittsburgh, PA, 12–14 July 2006. ACM Press. `http://cups.cs.cmu.edu/soups/2006/proceedings/p114_karger.pdf`.

[21] P. A. Karger, V. R. Austel, and D. C. Toll. A New Mandatory Security Policy Combining Secrecy and Integrity. RC 21717 (97406), IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 15 March 2000. `http://domino.watson.ibm.com/library/CyberDig.nsf/home`.

[22] P. A. Karger and H. Kurth. Increased information flow needs for high-assurance composite evaluations. In *Second IEEE International Information Assurance Workshop*, pages 129–140, Charlotte, NC, 2004. IEEE Computer Society.

[23] P. A. Karger, D. C. Toll, and S. K. McIntosh. Processor requirements for a high security smart card operating system. In *Proceedings of the Eighth e-Smart Conference*, Sophia Antipolis, France, 19–21 September 2007. Eurosmart. available as IBM Research Division Report RC 24219 (W0703-091), `http://domino.watson.ibm.com/library/CyberDig.nsf/Home`.

[24] G. S. Kc and P. A. Karger. Preventing attacks on machine readable travel documents (MRTDs). Technical Report RC 23909 (W0603-079), IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 10 March 2006. `http://domino.research.ibm.com/library/cyberdig.nsf/index.html`.

[25] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis: Leaking Secrets. In *Proceedings of Crypto '99*, Lecture Notes in Computer Science, Vol. 1666, Springer Verlag, pages 143–161, Santa Barbara, CA, August 1999.

[26] PDP–11/45 processor handbook. Order Code 67.00473.2743, Digital Equipment Corporation, Maynard, MA, 1973. `http://bitsavers.org/pdf/dec/pdp11/handbooks/PDP1145_Handbook_1973.pdf`.

[27] W. Rankl and W. Effing. *Smart Card Handbook: Third Edition.* John Wiley & Sons, Chichester, England, 2003. translated from Handbuch der Chipkarten, 4th edition, Carl Hanser Verlag, Munich, 2002.

[28] D. Safford and M. Zohar. Trusted computing and open source. *Information Security Technical Report*, 10(2):74–82, 2005.

[29] R. R. Schell, T. F. Tao, and M. Heckman. Designing the gemsos security kernel for security and performance. In *Proceedings of the 8th National Computer Security Conference*, pages 108–119, Gaithersburg, MD, 30 September - 3 October 1985.

[30] G. Schellhorn, W. Reif, A. Schairer, P. Karger, V. Austel, and D. Toll. Verification of a formal security model for multiapplicative smart cards. In *6th European Symposium on Research in Computer Security (ESORICS 2000)*, Lecture Notes in Computer Science, Vol. 1895, Springer Verlag, pages 17–36, Toulouse, France, 2000.

[31] H. Scherzer, R. Canetti, P. A. Karger, H. Krawczyk, T. Rabin, and D. C. Toll. Authenticating Mandatory Access Controls and Preserving Privacy for a High-Assurance Smart Card. In *8th European Symposium on Research in Computer Security (ESORICS 2003)*, pages 181–200, Gjøvik, Norway, 13–15 October 2003. Lecture Notes in Computer Science, Vol. 2808, Springer Verlag.

[32] W. L. Schiller. The design and specification of a security kernel for the PDP–11/45. ESD–TR–75–69, MTR–2709, The MITRE Corporation, Bedford, MA, HQ Electronic Systems Division, Hanscom AFB, MA, May 1975. `http://csrc.nist.gov/publications/history/schi75.pdf`.

[33] G. J. Simmons. Subliminal communication is easy using the dsa. In *Advances in Cryptology — Eurocrypt '93*, pages 218–232, Berlin, 23–27 May 1993. Lecture Notes in Computer Science, Vol. 765, Springer Verlag.

[34] The feasibility of a secure communications executive for a communications system. Technical Report MCI-75-10, Information Systems Technology Applications Office, HQ Electronic Systems Division, Hanscom AFB, MA, Aug. 1974.

[35] D. Toll, S. Weber, P. A. Karger, E. R. Palmer, and S. K. McIntosh. Tooling in Support of Common Criteria Evaluation of a High Assurance Operating System. Technical report, 3 April 2008. `https://buildsecurityin.us-cert.gov/daisy/bsi/articles/knowledge/lessons/961.html`.

[36] D. C. Toll, P. A. Karger, E. R. Palmer, S. K. McIntosh, and S. Weber. The caernarvon secure embedded operating system. *Operating Systems Review*, 42(1):32–39, 2008.

[37] J. Whitmore, A. Bensoussan, P. Green, D. Hunt, A. Kobziar, and J. Stern. Design for Multics security enhancements. ESD–TR–74–176, Honeywell Information Systems, Inc., HQ Electronic Systems Division, Hanscom AFB, MA, Dec. 1973. `http://csrc.nist.gov/publications/history/whit74.pdf`.

[38] K.-L. Wu, P. S. Yu, B. Gedik, K. Hildrum, C. C. Aggarwal, E. Bouillet, W. Fan, D. George, X. Gu, G. Luo, and H. Wang. Challenges and experience in prototyping a multi-modal stream analytic and monitoring application on System S. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 1185–1196, Vienna, Austria, 23–27 September 2007. `http://www.vldb.org/conf/2007/papers/industrial/p1185-wu.pdf`.

APPENDIX

A Chip Initialization

Smart card chips containing the Caernarvon system are intended to be high security devices. To this end, it is imperative that each individual chip be secure right from the point of manufacture, with no opportunity for the chip or its contents to be compromised while in the factory, nor between the factory and the end user. Manufacture and initialization are the most security-sensitive stages in the chip's entire lifecycle, because the chip is in its most vulnerable, exposed state, and it is during these stages that important roles and security parameters are set for the remainder of the chip's lifecycle. A fundamental assumption is that the manufacturing line is secure, which requires the chip manufacturer to assure that it is safe from tampering, collusion, theft, and other threats, including those from insiders.

In a typical smart card chip manufacturing facility, manufacturing test software is built into each chip to assure the viability of the chip. The test software tests the processor, memory subsystem, internal peripherals, and other subsystems such as cryptographic accelerators. These tests typically destroy the contents of writable memory, thus, the chips cannot be initialized with unique persistent data until all manufacturing tests are complete, and the chip is known to be good. Once the manufacturing tests have completed successfully, the test software downloads a copy of the initial file system *for that chip*, decrypting it with a triple-DES key held in read-only memory, and used only once (during manufacture). The image of each chip's initial file system is pre-calculated by the chip manufacturer by filling in the values of security-relevant data items in predefined locations. Some of these items include certificates, private and public keys, Diffie-Hellman key parameters, a chip-unique seed for random number generation, initial access classes, and uncertified application binary files. Because it is difficult for the smart card chip's processor to meet the demanding speed required by the manufacturing line, these security-relevant items are not typically generated on-chip. Instead, they must be generated and digitally signed in advance in hardware security modules such as the IBM 4764 [17], and injected into each smart card chip at very high speeds. Additionally, the chip manufacturer digitally signs a certificate unique to each chip, thus enabling external (off-chip) applications to verify (as part of an interactive authentication protocol) that communications emanate from an authentic Caernarvon chip, and not an imposter. This chip certificate includes a serial number and public key unique to each chip, a chip type / configuration code, the Caernarvon software hash value, version number, and evaluation assurance level.

The chip makes use of a public key hierarchy to establish identities and public keys of the actors that set the final configuration of the chip's software and data. Actors include the chip manufacturer, the smart card enabler, the smart card personalizer, the smart card issuer, the application certifying body, and others. During initialization, some of the public keys and roles of these actors are set by the chip manufacturer. Others are initialized later in the chip's lifecycle, and can only be set by an actor authenticated in a specific role.

After the test code has completed the initialization of a chip, it disables itself so that it can never be run again. At this point in the chip's lifecycle, the OS is fully functional and secure. Thus, when the chip is first powered up for any purpose outside of the manufacturing line (for example, for personalization of the smart card for the end user), the Caernarvon system is in control. In particular, full system authentication is required to perform any operations such as personalization or the installation of applications.