RC 24687 (W0811-037), 5 Nov 2008
Computer Science

IBM Research Report

Design of a Secure Smart Card Operating System for Pervasive Applications

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

Research Division
Almaden – Austin – Beijing – Delhi – Haifa – T.J. Watson – Tokyo – Zurich

This paper has been submitted to the Seventh Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2009)

Design of a Secure Smart Card Operating System for Pervasive Applications

Paul A. Karger, David C. Toll, Elaine R. Palmer, Suzanne K. McIntosh and Sam Weber
IBM Corporation, 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 use in pervasive applications such as electronic passports, military and public sector identification cards and cell-phone based financial and entertainment applications. However, such applications require a secure environment, which can only be provided with sufficient hardware and a secure operating system. In this paper we argue that smart card platforms pose additional security challenges when compared to traditional computer platforms. We discuss our design for a secure smart card operating system, named Caernarvon, and show that it addresses these challenges, which include secure application download, protection of cryptographic functions from potentially malicious applications, resolution of covert channel issues, and assurance of both security and data integrity in the face of arbitrary power losses.**

I. 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, for electronic passports, and cell-phone based financial and entertainment applications. For a good overview of smart card technology in general, see [28].

As the prime uses of smart cards are identification, authorization and encryption, it is crucial that sufficient 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 [34].

In this paper we argue that secure operating system development provides challenges as well as opportunities for pervasive systems. 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], both of them describing the platform-independent system aspects. This paper, in contrast, will focus on the challenges and solutions related to pervasive platforms in general and smart-card devices specifically.

A. Background – Smart Cards

Secure operating systems research for traditional computers has a long history. However, hardware and protocol issues pose additional challenges for secure pervasive systems.

Besides the weak processing capabilities and small memories of these devices, until recently their CPUs did not offer memory protection or separate supervisor and user states [22]. Without these features, an operating system cannot prevent untrusted code from taking over the entire device. Although some currently-used chips do have these features, hardware issues regarding power and memory technology remain.

Pervasive devices, by their nature, have limited power sources. Smart cards do not have any self-contained power and are entirely reliant on current supplied by their readers, which can be removed at any time. Even devices which do contain batteries are subject to sudden failures. When power is removed, any in-progress storage writes will fail, scrambling the relevant storage locations. Attackers can make use of this fact by removing power at critical moments.

Pervasive devices generally use EEPROM or flash memories to hold data when the device is off. Memory locations in both of these technologies will fail if too many write operations are performed, and secure systems have to defend against attackers deliberately causing failures at critical locations.

Turning our attention to protocol issues, smart cards are constrained by pre-existing specifications, of which the most important is ISO 7816–4 [19]. This specification defines the application programming interfaces (APIs) 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 II-B.)

Although 7816–4 does not specify the internals of a smart card operating system, it has implications on the file system design. Smart card filesystems are hierarchic, like those of many conventional operating systems. However, the smart card industry uses its own 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, as one might expect), but not vice-versa.

B. Background – Security Policy Models

Security policy models can be broken down into three major categories, listed in order of complexity:

- Preventing unauthorized disclosure of information,
- Preventing tampering or sabotage, and
- Preventing denial of service.

1) Preventing Unauthorized Information Disclosure: The first requirement of most security systems is preventing unauthorized disclosure of information. This section examines two classes of mechanisms: discretionary access controls and mandatory access controls.

Discretionary access controls are found in most operating systems to determine the access rights users get to files. These are typically access control lists or permission bits, based on the fully general Lampson access matrix [25]. They are called discretionary, because the access rights to an object may be determined at the discretion of the owner or controller of the object. The presence of Trojan horses in the system can cause great difficulties with discretionary controls. The Trojan horse could surreptitiously change the access rights on an object or could make a copy of protected information and give that copy to some unauthorized user.

Harrison, Ruzzo, and Ullman [17] have shown that the confinement problem, which is that information cannot flow to unauthorized recipients, is undecidable for general discretionary access controls. They show that solving the confinement problem is equivalent to solving the Turing-machine halting problem.

Mandatory access controls have been developed to deal with the Trojan horse problems. The distinguishing feature of mandatory access controls is that the system manager or security officer may constrain the owner of an object in determining who may have access rights to that object.

Lipner [26] and Denning [12] have shown that lattice security models, unlike the general Lampson access matrix, can solve the Trojan horse problem. Most mandatory controls have been based on lattice security models.

A lattice security model consists of a set of access classes that form a partial ordering. Any two access classes may be less than, greater than, equal to, or not ordered with respect to one another. There is no requirement for strict hierarchical relationships between access classes. The U.S. military services use a set of access classes that have two parts: a secrecy level and a set of categories that represent compartments of information for which an individual must be specially cleared. To gain access to information in a category, an individual must be cleared, not only for the secrecy level of the information, but also for the relevant categories.

Lattice models were first developed by Bell and LaPadula [5] to formalize the military security model and to develop techniques for dealing with Trojan horses that attempt to leak information. At the time, no one knew how to deal with Trojan horses at all, and it came as quite a surprise that two quite simple properties could prevent a Trojan horse from compromising sensitive information.

First, the simple security property says that if a subject wishes to gain read access to an object, the access class of the object must be less than or equal to the access of the subject. This is just a formalization of military-security-clearance procedures that one may not read a document unless one is properly cleared.

Second, the *-property requires that if a subject wishes to gain write access to an object, the access class of the subject must be less than or equal to the access class of the object. Enforcing the *-property means that any Trojan horse that attempts to steal information from a particular access class cannot store that information anywhere except in objects that are classified at least as high as the source of the information.

2) Preventing Tampering and Sabotage: To address the problems of tampering and sabotage, Biba [6] developed a model of mandatory integrity that is a mathematical dual of the Bell and LaPadula mandatory-security model. Biba defines a set of integrity access classes that are analogous to security access classes and defines simple-integrity and integrity-confinement properties that are analogous to the simple-security and confinement properties. The difference between integrity and secrecy is that a program of high integrity is prevented from reading or executing low integrity objects that could be the source of tampering or sabotage.

3) Preventing Denial of Service: Preventing denial of service is much more difficult than all of the other security requirements, as a fully general solution would appear to require solving the Turing machine halting problem. However, as this paper will show, denial of service issues are addressed in a number of areas of the Caernarvon operating system, including particularly the area of quota enforcement. (See section III-A5.)

C. Background – Covert and Side Channels

The paths over which a Trojan horse leaks information are called *covert channels* [16]. Covert channels can be divided into two major categories: *storage channels* and *timing channels*. Information can be leaked through a storage channel by changing the values of any of the state variables of the system. Thus, contents of files, names of files, and amount of disk space used are all examples of potential storage channels. A Trojan horse can leak information through a storage channel in a purely asynchronous fashion. There are no timing dependencies. By contrast, information can be leaked through a timing channel by modifying the length of time that system functions take to complete. For example, a Trojan horse could encode information into deliberate modifications of the system page-fault rate. Timing channels all use synchronous communication and require some form of external clocking.

Side channels are closely related to covert channels, but with one major difference. Side channels do not require the presence of a Trojan horse to leak information. Side channel attacks can be mounted either external to the system, such

as power analysis attacks [24], or by software, such as cache timing analysis attacks on cryptographic algorithms [27].

D. Background – Common Criteria

With the growing needs for better computer security, industry and governments have developed a concept of independent third-party evaluation to enable purchasers of computer systems (both hardware and software) to have an unbiased assessment of the quality of the security implementations. In concept, these evaluations would be similar to the unbiased evaluations carried out by consumer magazines, such as **Consumers Reports**. A variety of countries developed national criteria, culminating in the Common Criteria for Information Technology Security Evaluation [11] which has become an ISO standard. The Common Criteria specifies both functional requirements that specify security feature and assurance levels running from EAL1 to EAL7 that assess how much you can trust the quality of an implementation.

II. Smart Card Applications Code and Security

A. 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 modifications provide three major improvements. First, the integrity policy is changed to permit high integrity programs to examine and possibly sanitize low integrity data while still preventing executing of low integrity program code or scripts. Second, both the secrecy and integrity models are generalized to support broad commercial organizations that may not agree on any common security authorities. Third, the Caernarvon model provides a framework for downloading code based on objective standards for determining the level of trustworthiness of that code. These security policies are all discussed in detail in [21].

1) 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].

B. 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 as 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. Calls from user-mode applications to system-level code are referred to as Supervisor Calls (SVCs) in this environment.

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 SVCs. 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 existing application. Otherwise, the existing application could masquerade as the new one. 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, that program is forcibly terminated.

C. Cryptographic Facilities

Many modern smart card processors 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. [24], [1]. Protection of cryptographic functions against 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 [32].

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.

III. Security Design Challenges

A. File System

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

1) 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 I-A. 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 III-B.
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.

2) 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 does not create a performance 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.

3) 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.

4) 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.

5) 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 operating system [35, 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.

6) 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 exe-

cutable 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 [13]. In particular, Caernarvon capabilities cannot be passed from one process to another.

B. Persistent Storage Manager - PSM

In the Caernarvon system, the physical blocks of storage are managed by the Persistent Storage Manager (PSM). 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.

C. Application Download

A primary 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.

1) 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 Access Class.
2) create the necessary top-level DF associated with the new access class, 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.

2) 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.

D. Side Channel Issues

Just as for other pervasive devices, 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].

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 [15] 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. Currently, a description can be found here [9].

E. 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 strong cryptographic 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 [14] key parameters used for authentication, 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 [18], 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 off-chip applications to verify (as part of an interactive authentication protocol) that communications come 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.

IV. Common Criteria Problems Solved

During the Caernarvon development, we ran a number of Common Criteria related issues, including the lack of appropriate protection profiles and the extensive testing requirements.

A. Protection Profiles

The Common Criteria uses *protection profiles* to define the set of functional and assurance requirements appropriate to a class of products. Standardized protection profiles make it easier to compare evaluations of different products.

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.

B. 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.

V. 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. Caernarvon enables extensions of electronic passports to include e-Visas; indeed, our demo for the Caernarvon security policy was of e-Visas on a passport.
- 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, and hotels, etc.
- 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 [23], 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 [10] for System S, a large-scale, distributed, stream processing system for analyzing large amounts of unstructured data [36]. The mandatory security policy has also been incorporated into the Simple Linux Integrity Module (SLIM) in a Trusted Linux Client [29]. Lastly, the lessons learned from a developer's perspective were documented in [33].

VI. Conclusion

The Caernarvon operating system project has shown the feasibility of building smart card systems with much higher levels of security. It is possible to download applications from multiple sources that may be mutually hostile, yet still prevent these applications from interfering with each other or the operating system.

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

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 (now with Giesecke & Devrient), 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*. LNCS Vol. 2523, Springer, 13-15 August 2002, pp. 29–45.

[2] "Application interface for smart cards used as secure signature creation devices – part 1: Basic requirements," Comité Européen de Normalisation, Brussels, Belgium, CWA 14890-1, 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," Technion - Israel Institute of Technology, Haifa, Israel, Tech. Rep. CS-2006-07, 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 Conf.* LNCS Vol. 3956, Springer, 8–11 March 2005, pp. 155–173.

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

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

[7] "Certification Report for Tachograph Card Version 1.0 128/64 R1.0 from ORGA Kartensysteme GmbH," Bundesamt für Sicherheit in der Informationstechnik, Bonn, Germany, Tech. Rep. BSI-DSZ-CC-0205-2003, 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 *Proc. of Crypto '99*. LNCS Vol. 1666, Springer, August 1999, pp. 398–412.

[9] 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.

[10] 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 *Proc. IEEE Symp. on Security and Privacy*. Oakland, CA: IEEE Computer Society, 20–23 May 2007, pp. 222–227.

[11] "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.

[12] D. E. Denning, "A lattice model of secure information flow," *Comm. ACM*, vol. 19, no. 5, pp. 236–243, May 1976.

[13] J. B. Dennis and E. C. Van Horn, "Programming semantics for multiprogrammed computations," *Comm. ACM*, vol. 9, no. 3, pp. 143–155, Mar. 1966.

[14] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. on Information Theory*, vol. IT-22, no. 6, pp. 644–654, Nov. 1976.

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

[16] V. D. Gligor, "A guide to understanding covert channel analysis of trusted systems," National Computer Security Center, Fort George G. Meade, MD, Tech. Rep. NCSC-TG-030, Version 1, Nov. 1993, http://www.radium.ncsc.mil/tpep/library/rainbow/NCSC-TG-030.pdf.

[17] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman, "Protection in operating systems," *Comm. ACM*, vol. 19, no. 8, pp. 461–471, Aug. 1976.

[18] "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.

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

[20] P. A. Karger, "Privacy and security threat analysis of the federal employee personal identity verification (PIV) program," in *Proc. 2nd Symp. on Usable Privacy and Security*. Pittsburgh, PA: ACM Press, 12–14 July 2006, pp. 114–121.

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

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

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

[24] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis: Leaking Secrets," in *Proc. of Crypto '99*. LNCS Vol. 1666, Springer Verlag, August 1999, pp. 143–161.

[25] B. W. Lampson, "Protection," *Operating Systems Review*, vol. 8, no. 1, pp. 18–24, Jan. 1974, Proc. Fifth Princeton Conf. on Information Sciences and Systems, Princeton, NJ, USA, March 1971, pp. 437–443.

[26] S. B. Lipner, "A comment on the confinement problem," *Operating Systems Review*, vol. 9, no. 5, pp. 192–196, Nov. 1975, Proc. of the Fifth Symp. on Operating Systems Principles, Unversity of Texas at Austin, Austin, TX, USA, 19–21 November 1975.

[27] C. Percival, "Cache missing for fun and profit," Tech. Rep., 2005, http://www.daemonology.net/papers/htt.pdf.

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

[29] D. Safford and M. Zohar, "Trusted computing and open source," *Information Security Technical Report*, vol. 10, no. 2, pp. 74–82, 2005.

[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 Symp. on Research in Computer Security (ESORICS)*. LNCS Vol. 1895, Springer, 2000, pp. 17–36.

[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 Symp. on Research in Computer Security (ESORICS)*. LNCS Vol. 2808, Springer Verlag, 13–15 October 2003, pp. 181–200.

[32] G. J. Simmons, "Subliminal communication is easy using the DSA," in *Advances in Cryptology — Eurocrypt '93*. Berlin: LNCS Vol. 765, Springer, 23–27 May 1993, pp. 218–232.

[33] 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," Tech. Rep., 3 April 2008, https://buildsecurityin.us-cert.gov/daisy/bsi/articles/knowledge/lessons/961.html.

[34] D. C. Toll, P. A. Karger, E. R. Palmer, S. K. McIntosh, and S. Weber, "The Caernarvon secure embedded operating system," *Operating Systems Review*, vol. 42, no. 1, pp. 32–39, 2008.

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

[36] 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 *Proc. of the 33rd Intl. Conf. on Very Large Data Bases*, Vienna, Austria, 23–27 September 2007, pp. 1185–1196.