Implementing a High-Assurance Smart-Card OS

Paul A. Karger[1], David C. Toll[1], Elaine R. Palmer[1], Suzanne K. McIntosh[1], Samuel Weber[1,⋆], and Jonathan W. Edwards[2]

[1] IBM Thomas J. Watson Research Center,
P.O. Box 704, Yorktown Heights, NY 10598, USA
`karger@watson.ibm.com, (toll|erpalmer|skranjac|jone)@us.ibm.com`
[2] IBM Global Business Services
1500 Aristides Blvd., Lexington, KY 40511, USA

Abstract. Building a high-assurance, secure operating system for memory constrained systems, such as smart cards, introduces many challenges. The increasing power of smart cards has made their use feasible in applications such as electronic passports, military and public sector identification cards, and cell-phone based financial and entertainment applications. Such applications require a secure environment, which can only be provided with sufficient hardware and a secure operating system. We argue that smart cards 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 malicious applications, resolution of covert channels, and assurance of both security and data integrity in the face of arbitrary power losses.

1 Introduction

The design of higher security operating systems has been studied since the 1960s. However, most of these designs have been for relatively large computer systems. This paper examines a series of issues that a high-security operating system must face to be able to run in an extremely memory-limited environment, such as a smart card, a cell phone, a small PDA, or other constrained pervasive devices. For a good overview of smart card technology in general, see [11].

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 some users having a "necklace of cards", each one hosting a single application. The Caernarvon project was started to create such a secure smart card operating system. A very high-level overview of the Caernarvon system can be found here [14]. In contrast, this paper focuses on a

⋆ Now with the National Science Foundation, 4201 Wilson Boulevard, Arlington, Virginia 22230, **sweber@nsf.gov** number of challenges faced in actually implementing a high-assurance operating system on a smart card.

Most existing smart card systems have required that all applications be written together and loaded onto the card prior to the card being issued, because smart card processors did not support internal security controls, and all applications had to be mutually trusting. However, with the development of new smart card processors with internal security features (see Section 1.1), much stronger security could be provided. Thus, one primary goal of the Caernarvon project was to build a smart card operating system capable of supporting the download of applications that might be actively hostile, both to each other and to the underlying operating system. To be able to protect against such potentially hostile applications, the Caernarvon security policy was chosen to be mandatory access controls (MAC), because only such controls can effectively deal with applications that may contain Trojan horses. (See Section 3.)

This paper will focus on the specific challenges that must be faced to build a high-assurance operating system for such memory-constrained devices as smart cards. Sections 4 and 5 examine in detail the following security aspects of the operating system that are particularly different from previously described work on high-assurance operating systems: a hierarchical file system structure to reduce memory consumption, elimination of global address space covert channels that are unique to smart cards, reducing memory consumption of mandatory access classes, capability-based discretionary access controls, reliable persistent storage in the presence of power failures and memory write errors, secure application download, secure cryptographic implementations without trusted applications, and secure chip initialization without slowing down manufacturing lines.

1.1 Feasibility

The first question in the Caernarvon project was "Would it be feasible at all to build such a system?". When IBM Research first considered the project, the answer was, "No." Early smart card chips did not have adequate hardware support for security, such as separate supervisor and user states and memory protection.

The project only became possible when the Philips (now NXP) Smart*XA* chip was introduced as the first smart card processor to meet those needs. Karger, Toll and McIntosh [9] discuss these hardware requirements in much more depth. Since then, other vendors have also introduced chips that meet the requirements to support the Caernarvon operating system. However, as discussed in [9], not all chips that claim to support memory protection can do so without introducing covert channel problems.

The surrent Smart*XA*2 chip supports a relatively large amount of memory for a smart card chip: 7 Kbytes of RAM, 256 Kbytes of ROM, and 144 Kbytes of EEPROM. However, compared to most contemporary secure computer system projects, that amount of memory is extremely tiny. Note that the numbers are in kilobytes, not megabytes or gigabytes, and there is no external peripheral memory, such as a disk. All memory must fit on the single chip.

2 Applications of the Technology

Many applications could benefit from a high assurance smart card operating system. Generally, 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 to mistrustful to hostile. 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 electronic passport issued by one government, with electronic entry/exit timestamps added by other (possibly hostile) governments, described in [8].
- a corporate/school campus card, with multiple application providers for copiers, vending machines, public transit, and building access
- an ID card for coalition military forces for building or computer access
- a subscriber identity module for mobile phones to hold credentials for financial institutions, governments, and phone service providers, etc.

There are roadblocks hindering the commercialization of a high assurance smart card operating system, such as: significant investment in time and funding is required by multiple institutions; the skills required cross several domains; 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 require the use of weak cryptographic authentication protocols, and the protocols of the Federal Employee Personal Identity Verification program require some very sensitive information to be transmitted in unencrypted form. In attempting to resolve these issues, the Caernarvon development led to a clearer understanding that in order to meet many security goals, privacy goals must also be met [8] .

3 Background – Security Policy and Authentication

The Caernarvon system builds on previous work on mandatory security policies to provide multi-level security. Mandatory security was chosen specifically because a major goal of the Caernarvon system is to support downloading of multiple applications from multiple application providers, who may be mutually hostile. Caernarvon system security is enforced using a mandatory security policy described more completely in [12] that is based on modifications of the Bell and LaPadula secrecy model [2] and the Biba integrity model [3].

Enforcement of a meaningful security policy requires that there be a secure mechanism to ensure that the use of the desired access classes is valid and correct. This authentication must be performed by the smart card's 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 smart card operating system can use this knowledge to safely grant or deny the host system access to files and other system objects on the card. Space does not permit including the full description of this authentication protocol which is available in [13].

4 Security Design Challenges

4.1 File System

The Caernarvon system implements a smart card file system. Besides the challenges caused by the specification and hardware restrictions, the file system must also enforce the system's security policy. There also must be a quota mechanism and support for memory-mapped files. The file system is the major repository of system state, and hence security is crucial to its design and implementation.

Caernarvon implements an ISO 7816-4 File System, with certain security extensions described in this section. This file system is implemented by two separate components, namely the Persistent Storage Manager (PSM) and the file system abstraction layer.

The PSM provides and manages memory objects, that is, blocks of persistent storage. Smart cards can have their power sources removed unexpectedly, corrupting in-progress memory writes. An important purpose of the PSM is to maintain the integrity of the memory objects, allowing other system components to ignore power interruption issues.

The PSM is not exposed to user mode applications. The file system layer sits on top of the PSM, and it is visible to applications.

File System Structure The ISO 7816-4 standard defines a hierarchical file system, which has a single MF (Master File, equivalent to "root" in Unix), which contains DFs (Dedicated Files, otherwise known as directories) and/or EFs (Elementary Files). The Caernarvon system extends this model by defining another file type, an "XF" (Executable File), which are executable programs.

Unlike most other file systems, the Caernarvon MF and DFs have no table of the files that they contain; instead, each DF and EF (including each XF) in the system has a pointer to its parent, as shown in Figure 1. This saves the space that

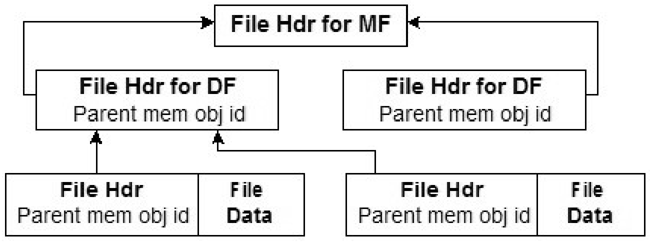

Fig. 1. Directory Structure Implementation

would be occupied by the list of file names and the reference to each file; it also means that, when files are created or modified, there is no need to update the corresponding DF entry for that file. However, this design has the drawback that file system searches require examining every file in the system. Since the amount of persistent memory is extremely limited (< 256 Kbytes), there can be only a small number of files in any given smart card, and this extended search cannot create a performance problem. Obviously, this space/time tradeoff algorithm does not scale to large memories with lots of files. Eventually, smart cards will have much larger persistent memories available. In that case, the highly modular and layered design of Caernarvon would make switching to a more conventional directory structure quite easy.

An obvious, but incorrect optimization for the linear search would be to cache the information about recently opened files. However, such a cache could not fit in RAM (only 7 kbytes total) and would itself consume the scarce persistent memory that we are trying to save. Furthermore, the open file table itself serves as a small cache, as long as the file remains open. Due to the slow communications speeds of smart cards, switching applications consumes more time than any file system cacheing would save.

The MF, each DF, and the headers and data areas of all files, are each held in a PSM memory object, the size and location of which are defined by the PSM's object descriptors, as shown in Figure 2.

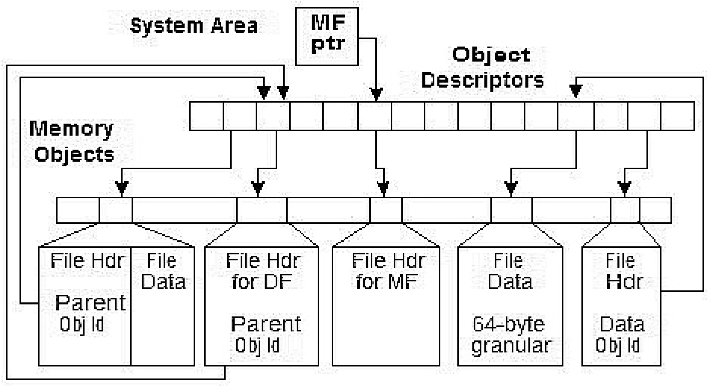

Fig. 2. PSM Memory Objects for Files

In general, files are packed so as to minimize the space and number of memory blocks required for their storage. In the case of most data files, the file header and the file data area are packed together into a single contiguous memory area. However, Caernarvon provides a facility for files to be memory mappable - this is used, in particular, for the code of executable programs. Data that is memory mappable must lie on a physical memory boundary and have a size granularity specified by the processor's memory protection unit. This means that mappable files have one memory object for the header and another for the data. Figure 2 shows the arrangement of the memory objects for both types of file.

Global Address Spaces—DFNames The ISO 7816-4 file system names all files as numbers. While that simplifies the file system, it makes it difficult for end users to select an application by name. Remembering the file numbers is not likely to be acceptable. As a result, ISO 7816-4 defines the concept of a DF Name that is used to select an executable program from outside the card. A DF name is a string name assigned to the DF containing the application. The DF Names are unique to the card, and, (as defined in ISO 7816-4) 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 loaded onto a particular card will win. Since ISO 7816-4 assumed all applications would be preloaded onto the card, this was never a problem. Once you have multiple application providers downloading applications to a card after issuance, the name collision problem can become serious. Second, such name collisions could be used as a covert channel to bypass mandatory access controls.

Caernarvon avoids these problems by making the DF Name space into a name space that is private to the current access class. The ISO 7816-4 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.

Storing Access Classes in the File System In most previous operating systems that supported mandatory access controls, the access classes of files and directories are stored with their associated files and directories, either in file headers or in the directory branches. Furthermore, Caernarvon access classes are designed to support multi-organizations, as discussed in [14, Section 3.1]. These multi-organizational access classes can be quite large to represent, and every file and directory may have its own unique access class. In a smart card with very limited amounts of memory, storing large numbers of access classes could be a severe problem.

To reduce access class memory usage in Caernarvon, two steps have been taken. First, we note that due to restricted memory for storing applications and data; no one smart card will need to use more than a small number (< 256) of distinct access classes. Therefore, the File System only stores each multi-organizational access class once in a table, and need store only an 8-bit index into that table with each file or directory.

Caernarvon follows the Multics practice [15] that while a child directory (DF) may have a different (higher) access class than its parent DF, ordinary files (EFs and XFs) must have the same access class as their parent DF. This allows us to save additional memory by requiring that files (EFs and XFs) do not have associated access classes, while directories (DFs) may but are not required to. DFs, EFs and XFs without an access class inherit the access class of their nearest parent DF that does have its own access class.

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, but doing so may remove the program's 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 re-opened it. If the program no longer has access to the file then the re-open will fail. This re-open approach avoids the kernel complexity of the Multics revocation approach and originates in the design of the DEC A1-secure virtual machine monitor [10].

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. Note that moving a DF must also move disk quota as discussed below.

Quota In order to protect against 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 provider. Covert channels whereby a Trojan horse could signal by either allocating all memory or freeing some are prevented.

Most contemporary operating systems use special quota accounts to charge for file space. However, such quota accounts require a great deal of management software. Instead, the Caernarvon system uses a very old approach (described below) from the Multics operating system [15, section 3.7.3.1], that is simpler to implement. The Multics approach was replaced by quota accounts, because such accounts were easier for time-sharing system users to manage. However, Caernarvon quota will be managed by application providers, not end-users, so the slightly harder management interface should not be a problem.

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. The combined atomic operation of moving a DF from one AC to another together with its quota is totally new and is needed to avoid covert storage channels. Most existing mandatory access control systems avoid the channels by restricting such moves to human users through a trusted path, but a smart card has no direct human interface.

Discretionary Security Policy - Capabilities As discussed above in Section 3, the primary security policy of the Caernarvon system is a mandatory access controls. This is because mandatory access controls are specifically designed to deal with malicious applications code. However, mandatory access controls do not provide the very fine-grained control that users can get from discretionary access controls, such as access control lists or permission bits. As the Caernarvon design progressed, we became concerned that including both a powerful mandatory access control mechanism as well as a full access control list implementation would exceed the very limited amount of memory that can be committed to the smart card operating system.

As a result, we chose to implement a compromise discretionary access control system, based on a very restricted form of capabilities [4]. These are discretionary security policy rules that may be associated with an individual executable program (an XF). These capabilities take two forms that are based on our assessment of the most common needs of smart card developers and are easy to implement in only a small amount of code.

1. a bit array that specifies whether that program is permitted to issue certain supervisor calls (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.

It is important to note that these capabilities are not a fully general capability system, as defined by Dennis and Van Horn [4]. In particular, Caernarvon capabilities cannot be passed from one process to another.

4.2 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.

Figure 3 shows how the PSM divides persistent storage. The System Area at the beginning of EEPROM contains information that the operating system uses to locate its internal data structures and persistent state. Items in the System Area are expected to be set up when the initial EEPROM image is

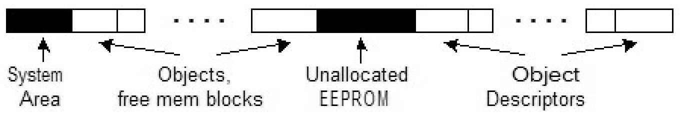

Fig. 3. PSM Use of Persistent Storage

built, and subsequently is never changed. Everything else is held in a memory object; every object has a unique ID, and is always referenced via this ID. Every memory object is in turn described (in particular, its location and size) by an object descriptor; a memory object is located by searching the object descriptors for the descriptor with the requested ID. This allows for the possibility of either the memory object and/or its descriptor moving in physical storage.

Management of Storage Objects Figure 2 shows how file system objects are stored in PSM memory objects. In older smart card systems, where the file system was created during card personalization or initialization, the file system was effectively static, that is there was no file creation or deletion after issuance. However, a Caernarvon file system is dynamic—the number of files and/or their sizes will change since the Caernarvon system allows for the installation or removal of applications, and the creation and deletion of files after the card has been issued.

This means that the PSM must manage any possible fragmentation of memory objects. The Caernarvon PSM avoids this completely by always ensuring that any memory object is stored in a single contiguous area of memory. However, this in turn causes another problem, in that as files are created, extended (or shrunk), or deleted, free storage will become fragmented. When a file is to be created, there may not be a single block of free memory available that is large enough. Alternatively when enlarging a file, there may be no immediately adjacent free memory.

A traditional approach to this problem would be to do a garbage collection: move in-use objects to contiguous storage and coalesce the free areas. In Caernarvon, we do not do a complete collection. Instead, when a memory allocation is attempted but no suitably-sized memory block is available, the PSM determines the minimal number of occupied blocks that need to be moved to create a large enough free memory space. After this compaction operation, memory may still be fragmented, but the number of memory writes will have been kept low in order to preserve the lifetime of the memory locations. It should be noted that compaction will certainly require multiple read and write operations to persistent storage, both of the memory objects and of the descriptors. These operations are time consuming, so minimizing their number and frequency is also important to maintain acceptable performance.

Weaknesses of Persistent Storage Write operations to EEPROM or Flash memory are slow, usually taking 4 to 6 milliseconds each. Further, the write block size for EEPROM is limited, for example, to 128 bytes. Thus writing any significant amount of data to a file is likely to take multiple write operations, plus additional writes to update the control block information. There is also the problem that the smart card is powered only when it is in the reader. These factors mean there is a significant risk of a file write operation being interrupted and not fully completed.

A solution to the power interruption problem 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, including all of the multiple write operations required, must be completed in its entirety, or not performed at all. The PSM maintains a *backtrace buffer* where, when persistent storage 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 entries are 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. When this is complete, the state of the memory is as if the transaction had never been started. There is one additional complication, in that, when powered up, the smard card cannot immediately start these backtrace operations, but is required to respond to the smart card reader within 40,000 processor clocks. This allows no time to perform the pending backtrace buffer operations. However, when the reader sends the first command to the card, the card can request a "waiting time extension" to delay its response to the reader; this request can be repeated as many times as are necessary to complete the pending backtrace operations.

Smart card persistent memory, EEPROM or Flash, has a limited number of write cycles before it starts to fail, for example between 100,000 and 500,000 for EEPROM, and only 10,000 for Flash. The PSM takes two measures to compensate for memory corruption due to the cells wearing out. First, once every write to persistent memory is completed and before control is returned, the low-level code that did the write compares the updated contents of 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. Second, the PSM places a checksum on every memory object under its management, including the control blocks or descriptors that define the memory objects. This checksum is verified on read operations to enable the detection of memory failures—an attempt is then made to recover the lost data byte(s). Once a memory error is detected, the memory area in question is marked as bad, and the data is re-written to a different location.

The backtrace buffer for atomic transactions is well known in data base design [6], but its use throughout a file system in conjunction with mechanism to recover from memory wearing out is unprecedented in smart cards. Of course, the backtrace buffer is itself an area of memory that has frequent write operations, and hence can wear out. When errors occur within the backtrace buffer, the PSM will attempt to move the backtrace buffer to a new area of memory.

4.3 Implementing Application Download

A primary aim of the Caernarvon system is to allow for the secure download of applications in the field. The card issuer can allow or forbid the download of applications, and when download is permitted, can control which organizations are allowed to install their applications on the card and how much file quota they may use. Note that in the context of download, the term "application" is not limited to just the XFs; it may also encompass the associated DFs, EFs, file quota, keys, etc.

The download process can be divided into two main steps, namely the creation of access classes (with the appropriate file quota) for organizations that currently are not present on the card, and then the 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 would be unacceptable to most customers.

Instead, the Caernarvon operating system includes secure cryptographic protocols to (a) create the Access Class and (b) to create the necessary top-level DF associated with the new access class, and set its allocated quota. These protocols will require a full future paper to explain and are not further discussed here.

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. 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. To implement this level of control, the Caernarvon kernel will check for digital signatures, either from the card issuer or the Common Criteria certifier or both.

4.4 Cryptographic Challenges

The Caernarvon system includes a cryptographic library to ensure that the cryptographic algorithms, such as DES, triple-DES, AES, RSA, DSA, and ECC are correctly and securely implemented in a side-channel free fashion. This is discussed in section 3.6 of [14].

In addition to proper cryptographic algorithm implementation, it is essential that cryptographic key management also be implemented securely. 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 sees the key; the application refers to the key by a name (actually a handle) 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. In addition, keys can be marked by purpose, such as to be encryption keys or 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, some smart card standards (such as the Global Platform standard [1]) 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. Access to the crypto co-processor must still be mediated by the kernel to avoid both object re-use and covert channel issues. While applications may find this mode a necessity, its use is strongly discouraged, since Caernarvon cannot ensure any security for these raw keys.

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, in particular 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.

5 Chip Initialization

Smart card chips containing the Caernarvon system are intended to be high security devices. Therefore, 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 or during delivery. 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. 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 [5] 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 [7], and injected into each smart card chip at very high speeds. The chip manufacturer also 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 and 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 must securely disable itself so that it can never be run again, even in the face of physical attacks on the chip. 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.

6 Conclusion

The Caernarvon operating system project has shown the feasibility of building smart card systems with much higher levels of security than is common practice. In particular, it is possible for a smart card to download and execute applications from multiple mutually-distrusting sources, but still prevent them from interfering with each other or with the operating system itself.

These goals required reconsideration of many traditional smart card software practices (see Section 4.1), as well as solving many security problems that are not present in larger-scale computers. In particular, new security approaches had to be developed to deal with the extreme memory constraints of a smart card, and these new techniques had to be applied throughout the operating system. Despite these challenges, the Caernarvon operating system was completely free of covert storage channels.

The present status of the Caernarvon system is discussed in [14, Section 7]. An alpha test version has been implemented and runs on a smart card hardware emulator. However, commercial viability is still undetermined, because the complete OS has not been released as a product. Several major pieces of the design have also been transferred to other IBM projects, as discussed in [14, Section 6].

What this paper has shown is how such a high-security system can actually be implemented in the extremely memory constrained environment of a smart card, yet still support a very general mandatory access control model that can protect against essentially arbitrary malware attacks.

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, from IBM: 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, and Michael Steiner; from atsec: Helmut Kurth; from Philips: Hans–Gerd Albertsen, Christian Brun, Ernst Haselsteiner, Stefan Kuipers, Thorwald Rabeler, and Thomas Wille; from the University of Augsberg: Wolfgang Reif, Georg Rock and Gerhard Schellhorn; from the DFKI: Axel Schairer and Werner Stephan; and from the German BSI: Stefan Wittmann. We must also thank Wietse Venema for his very useful suggestions on improving the paper.

References

[1] 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 (CASSIS), pages 155–173, Nice, France, 8–11 Mar. 2005. Lecture Notes in Comp. Sci., Vol. 3956, Springer.

[2] 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. `http://csrc.nist.gov/publications/history/bell76.pdf`.

[3] 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.

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

[5] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. on Information Theory*, IT-22(6):644–654, Nov. 1976.

[6] J. N. Gray. *Notes on Data Base Operating Systems*, pages 393–481. Lecture Notes in Comp. Sci.9, Vol. 60. Springer Verlag, Berlin, Germany, 1978.

[7] 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`.

[8] P. A. Karger, G. S. Kc, and D. C. Toll. Privacy is essential for secure mobile devices. *IBM Journal of Research and Development*, 53(2), 2009.

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

[10] P. A. Karger, M. E. Zurko, D. W. Bonin, A. H. Mason, and C. E. Kahn. A retrospective on the VAX VMM security kernel. *IEEE Trans. on Software Eng.*, 17(11):1147–1165, Nov. 1991.

[11] W. Rankl and W. Effing. *Smart Card Handbook: Third Edition.* John Wiley & Sons, Chichester, England, 2003.

[12] 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)*, Lecture Notes in Comp. Sci., Vol. 1895, Springer Verlag, pages 17–36, Toulouse, France, 2000.

[13] 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)*, pages 181–200, Gjøvik, Norway, 13–15 Oct. 2003. Lecture Notes in Comp. Sci., Vol. 2808, Springer Verlag.

[14] 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.

[15] 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`.