The Caernarvon Secure Embedded Operating System

David C. Toll
toll@us.ibm.com

Paul A. Karger
karger@watson.ibm.com

Elaine R. Palmer
erpalmer@us.ibm.com

Suzanne K. McIntosh
skranjac@us.ibm.com

Sam Weber
samweber@watson.ibm.com

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

ABSTRACT

The Caernarvon operating system was developed to demonstrate that a high assurance system for smart cards was technically feasible and commercially viable. The entire system has been designed to be evaluated under the Common Criteria at EAL7, the highest defined level of assurance.

Historically, smart card processors have not supported the hardware protection features necessary to separate the OS from the applications, and one application from another. The Caernarvon OS has taken advantage of the first smart card processors with such features to be the first smart card OS to provide this kind of protection. Even when compared with conventional systems where the hardware protection is routine, the Caernarvon OS is noteworthy, because of the EAL7 assurance.

This approach facilitated implementation of a formally specified, mandatory security policy providing multi–level security (MLS) suitable for both government agencies and commercial users. The mandatory security policy requires effective authentication of its users that is independent of applications. For this reason, the Caernarvon OS also contains a privacy-preserving, two–way authentication protocol integrated with the Mandatory Security Policy.

The Caernarvon OS includes a strong cryptographic library that has been separately certified under the Common Criteria at EAL5+ for use with other systems. The Caernarvon OS implements a secure method for downloading trusted and untrusted application software and data in the field, with the assumption that all applications are potentially hostile. While the initial platform for the operating system was smart cards, the design could also be used in other embedded devices, such as USB tokens, PDAs, cell phones, etc.

Categories and Subject Descriptors

C.3 [**Special Purpose and Application-Based Systems**]: Smartcards; D.4.6 [**Operating Systems**]: Security and Protection—*access controls, authentication, cryptographic controls, information flow controls*; D.4.7 [**Operating Systems**]: Organization and Design—*Real-time and embedded systems*

General Terms

Security, Reliability

Keywords

Operating systems, mandatory access controls, embedded systems, smart cards, mobile phones, PDAs, Common Criteria evaluation

1. INTRODUCTION

The goal of this paper is to describe a high assurance embedded operating system developed in anticipation of a market need for secure, multi-application devices running arbitrary, field-downloadable applications. This paper concentrates on the issues arising when implementing such a system on embedded devices, including smart cards. Other aspects of the system are described in previous papers [21, 20, 26]. Although the technology is described in the context of smart cards, it is indeed applicable to other embedded devices with limited memory.

1.1 Motivation

The Caernarvon[1] operating system (OS) was developed as a result of a request to the IBM Research Division from the IBM Smart Card Solutions team in Germany to assist in the development of smart cards that were markedly more secure than those in existence at the time. It was common for a card application (such as banking or health) to be carefully intertwined with a set of OS-like functions, such as a file system and support for external communication protocols. The applications and the support functions together were known as the card operating system, and were typically written in hand-tuned assembler.

Because of limitations of the smart card processors in that era, there was no logical separation of application and OS software. When an application was executing, it had full control of the processor and unrestricted access to all persistent storage (the on-card file system) and memory. Errant applications inadvertently modified data intended to be under the exclusive control of the OS or other applications. Negative side-effects from this lack of separation wreaked havoc with applications, triggering the need for software corrections. Unfortunately, the software was written into Read-Only Memory (ROM), and could not be patched post-issuance.

Techniques emerged to deal with these problems, which are still in use today. The problem of patching the system

[1]Caernarvon is arguably the most magnificent of the castles built in North Wales at the end of the thirteenth century.

was addressed by having the OS jump to reserved, post-issuance, writable patch areas. Security concerns limit the use of this technique. Another technique which is still in common use today is to use a software interpreter to protect the OS from the applications and the applications from each other. While an interpreter-based approach mitigates some of the problems, it exposes others, such as the limited ability of the tiny on-card processor to carry out run-time security enforcement, and the lack of security and integrity of off-card development tool suites and *their* environments.

Historically, smart card processors have not supported the memory protection and user/supervisor states needed to support the OS in enforcing a strong security policy, even when executing native (machine) code and an arbitrary mix of untrusted applications. When the first smart card chip became available with such features (the Philips[2] Smart*XA*, later updated to the Smart*XA*2), we proposed building a high-assurance operating system to take advantage of those features. Now such features are available on a few other smart card chips, and Caernarvon could be ported to any chip which supports them. Features such as secure field downloading of applications, strong privacy-preserving authentication, and controlled sharing of data across applications were incorporated to meet the security requirements of the emerging multi-application embedded devices market.

1.2 Project Goals

Our goal of developing a high-assurance operating system for an embedded device, in our case, a smart card, required us to invent a security policy with sufficient features to compete in the open market. This policy was proven internally consistent, and our design and implementation reflected this policy. Additionally, we followed a rigorous development methodology, as mandated by the highest level Common Criteria evaluation level, known as EAL7.

The Caernarvon project objectives included the following key items:

- To use the preferred hardware–based approach for high assurance in which security features present in the hardware architecture are marshaled to implement a formally specified, proven correct, Mandatory Security Policy that provides the MLS capability suitable for multiple government agencies, including those resposible for national defense.
- To guarantee isolation of data belonging to different organizations and also facilitate controlled sharing of data between organizations, through hardware–based security,
- To provide built–in cryptographic functions that can be proven correct.
- To provide a privacy–preserving, two–way authentication protocol integrated with the Mandatory Security Policy.
- To devise a secure method for downloading untrusted code in the field—with the assumption that all applications are potentially hostile.
- To successfully complete a Common Criteria evaluation at assurance level 7
- To support both native and interpreted applications.
- To host applications including HSPD–12 [7], electronic passports, and proposed electronic visas.
- To obey the following platform–specific constraints:
  - To comply with the ISO 7816 standard.
  - To comply with smart card power limitations.
  - To have sufficient transaction speed for our anticipated applications.

[2]Philips Semiconductors is now the NXP Corporation.

1.3 Organization of This Paper

One goal permeated every activity—completing a Common Criteria evaluation at the very highest assurance level. Thus, a discussion of Common Criteria is helpful in understanding some of our decisions. A detailed description of the OS architecture follows. The Common Criteria influence appears again in a brief discussion of the extraordinary documentation and testing tasks we encountered. Finally, we close with a brief discussion of the impact of the work and its current status.

2. COMMON CRITERIA EVALUATION

The Common Criteria[12] is an ISO standard (ISO 15408: 2005) for evaluation of the security aspects of a computer system. It evolved from earlier evaluation criteria, including the US Trusted Computer System Evaluation Criteria (TCSEC) [13], and the European Information Technology Security Evaluation Criteria (ITSEC) [17]. The Common Criteria requires an independent third party evaluation of the product, examining both the security functional requirements and the evidence, dependent upon the assurance level, that the functional requirements are actually correctly implemented. The evaluation assurance levels (EAL) are measured from EAL1 (very low) to EAL7 (highest). Typical commercially available operating systems have received Common Criteria certificates at EAL4. Levels EAL6 and EAL7 are often called the *high assurance* levels, because systems evaluated at those levels under the Common Criteria (or equivalent levels under the TCSEC or ITSEC) are the only systems that have been shown to be generally resistant against sophisticated penetration attacks. These attacks are currently commonplace, but used to be considered only the concern of the military [9, p. 7–19], and include such problems as buffer overflows, incomplete argument validation, spyware, Trojan horses, and root kits. High assurance is specifically designed to address these problems.

To pass an EAL7 evaluation, the Common Criteria requires the strongest software engineering techniques known. These techniques include a formal security policy model, a full system design with a formal high–level design specification, and a formal proof of correspondence between the security policy model and the high–level design specification. It also requires a specification of all internal functions in a semi–formal low level design with a demonstration of correspondence between the high–level design, the low–level design, and the actual software code. The development cycle must include intensive design and code reviews and full configuration control. There must be comprehensive testing, including code coverage analysis that every path has been exercised, and that no dead code exists. Finally, there

must be an extensive vulnerability analysis for possible security loopholes, as well as a search for covert communications channels.

3. SYSTEM ARCHITECTURE

The Caernarvon kernel is monolithic and runs in supervisor state. The complete OS structure is shown in Figure 1. The other items in the figure are applications, all of which

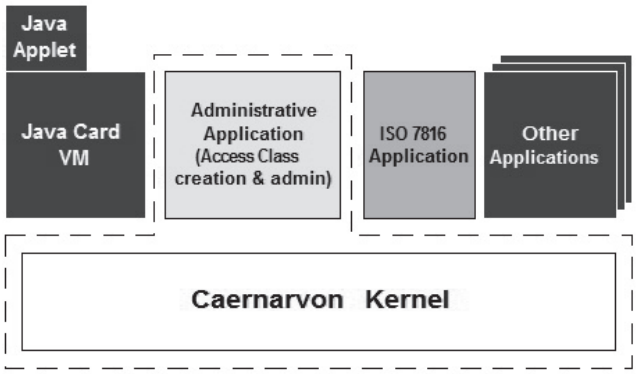

Figure 1: High Level Structure of Caernarvon

run in user mode. The kernel is protected from the applications, and the applications are protected from one another, by hardware protection features as described in Section 1.1. While this approach is standard in conventional operating systems, it is not common for smart card operating systems.

The dotted line that includes the Caernarvon Kernel and the mandatory Administrative Application (see section 3.5) indicates that these two components are those parts of Caernarvon that must be evaluated together under the Common Criteria. The other items need not be evaluated as part of Caernarvon, and they may or may not have their own Common Criteria evaluations. The ISO 7816 application is a system utility; no other "built–in" applications are currently planned. Figure 1 also shows how a Java Card[3] applet could be run under Caernarvon—the Java Card virtual machine is run as an application which executes the Java Card applets.

In theory, the Caernarvon kernel is capable of running multiple concurrent applications—much of the system design specifically allows for this. However, in practice, there is insufficient Random Access Memory (RAM) available in a smart card to store multiple stacks, data areas, crypto buffers, etc., and hence the current version of Caernarvon is limited to running just one application at a time.

Even though the Caernarvon kernel is monolithic, it is internally structured in layers as required by the Common Criteria and as originally proposed by Dijkstra [14], with defined interfaces between them. Many of these layers are shown in Figure 2. A component may call only those components in its own layer or in layers beneath it, calls to higher layers are not permitted.

3.1 Mandatory Security Policy

The Caernarvon kernel implements a mandatory security policy that is based on the Bell and LaPadula secrecy model [4], and a modified Biba integrity model [5]. This security

[3]Java Card is a trademark of Sun Microsystems, Inc.

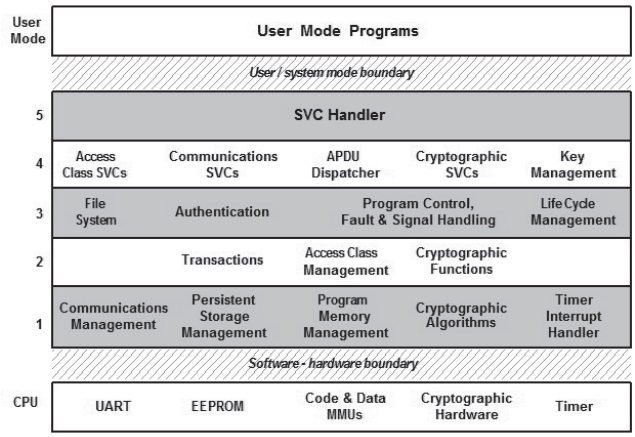

Figure 2: Caernarvon Component Layers

model applies to application download (in particular, after card issuance), the selection of the application program to run, and to file access (inter–application information sharing and exchange).

Access to files is permitted according to rules based on the comparison of *access classes*. An access class in Caernarvon is more correctly called a *Universal Access Class* (UAC); each UAC is comprised of one or more *Organizational Access Classes* (OACs). UACs and OACs come in two types, namely secrecy and integrity. The secrecy and integrity access classes are independent of each other. Examples of how secrecy and integrity access classes can be used can be found in [21, Section 4].

An OAC consists of a type field (indicating whether this is a secrecy OAC or an integrity OAC), an organizational identifier (OID), a sensitivity level, and an optional set of categories. The OID is similar to the organization identifier issued for Application IDs according to ISO 7816-5 [67]. In the case of secrecy OACs, the assignment and meaning of the sensitivity level and categories in the OACs for a given organization are completely controlled by that organization. Therefore, there can be no collision between categories with the same value chosen by other organizations because OIDs are unique. Categories correspond to internal groups, departments or other classifications chosen by the organization. The organization may alternatively regard the categories as a formalized "need to know" list.

Integrity OACs also have levels, which may be an integer in the range 0-7 inclusive. The value 0 means "normal" or "low" integrity. The values 1-7 indicate a higher level of integrity, and are assigned only to items that have been evaluated under the Common Criteria, and the OAC level then is the EAL level of the evaluation. Caernarvon allows for categories being present in integrity OACs, but currently such categories are not used for anything.

There are well–defined rules for the comparison of the access classes; these rules, together with a full description of the Caernarvon mandatory security policy, are in [21]. The Caernarvon mandatory security policy was formally specified and proven consistent [25] in a joint effort with the University of Augsberg and DFKI (the German Research Center for Artificial Intelligence).

3.2 Downloaded Applications

The Caernarvon operating system permits applications to be downloaded at any time, subject to optional approval requirements that can be imposed by the card issuer. Most applications are assumed to be untrusted and do not require any Common Criteria evaluation. The mandatory security policy *does* permit downloading of high–integrity applications that may perform limited downgrading of secrecy and/or upgrading of integrity. Such high–integrity applications must be Common Criteria evaluated and must be digitally signed by the Common Criteria certifying body, as described in [21, Section 3]. These high–integrity applications can work in conjunction with untrusted applications, so that the total amount of applications code requiring Common Criteria evaluation can be minimized. Furthermore, each of these trusted applications can be independently evaluated, as their security depends only on the underlying Caernarvon operating system.

3.3 Authentication

The mandatory security policy requires that the operating system perform an effective authentication of its users (in this case, the outside world) before allowing access to objects under the operating system's control. The Caernarvon authentication is application independent, and is enforced by the system kernel; it incorporates four mechanisms performed between the Caernarvon system in the smart card and the smart card reader:

- a device verifies the existence of a certified secret key on the other party
- the devices negotiate or exchange information to establish a common session (symmetric encryption) key for subsequent operations
- the devices negotiate or exchange information to establish a common access class for subsequent operations
- the authentication is mutual, that is it is two-way, and binds all of the above elements

The authentication protocol is based on IKE [16], which is one of the family of SIGMA protocols [23] that have been mathematically verified [8]. The authentication protocol uses a two-way Diffie-Hellman key exchange scheme in order to create a symmetric session key for the current session. Access class selection for the new session is performed as part of a two-way public key certificate verification—the access class selected here controls which files may be accessed in the current session. This protocol also is privacy-preserving, that is, it does not expose the identity of the card holder until it has been determined that the smart card reader is authorized to know the holder's identity.

This authentication protocol is described in [26]; it was submitted to the ESIGN-K working group that is designing a specification for the European signature application on Smart Cards, was published as a CEN standard [2] and is being further revised by CEN [3] prior to submission to ISO.

3.4 Data Storage Components

The data file storage facilities within the Caernarvon system are provided by two system components:

- The File System component provides a logical file structure in blocks of persistent storage.
- The Persistent Storage Manager (PSM) manages the physical blocks of persistent storage.

3.4.1 File System

The Caernarvon logical file system is as specified in ISO 7816-4 [19]. That is, the file system is tree–structured, with a single *Master File* (MF, the root directory), plus *Dedicated Files* (DFs, directories) and *Elementary Files* (EFs, terminal files). Files (both DFs and EFs) do not have text names, but instead have 16-bit numerical identifiers. Certain of the possible identifiers are reserved, for example the MF has the identifier 0x3F00.

It is interesting that the ISO 7816-4 file system is very similar to the file system of the MITRE PDP–11/45 security kernel [27] that was the world's first high-assurance system. The MITRE kernel also had only numerical identifiers, rather than text names. Smart cards avoid the text names to save space. The MITRE kernel did it for simplicity, since text names are usually variable length data structures. This similarity first convinced us that developing the Caernarvon operating system would be technically feasible.

While ISO 7816-4 specifies the external appearance of the file system, it does not dictate its internal implementation. ISO 7816 also specifies commands to be used from outside the smart card to access files within the card; however, ISO 7816 does not require or limit any internal Application Program Interfaces (APIs) provided by the file system within the card. Unlike most smart card systems, the Caernarvon kernel does not itself implement any of the commands coming from outside the card for file access; rather, such commands are passed to the currently selected application (see section 3.7). The operating system provides Supervisor Calls (SVCs) such as *open*, *close*, *read*, *write*, *seek*, *tell*, *create* and *delete* to enable the application to perform file operations. Files may also be created as *mappable*; such files can be mapped into the application's virtual address space and then can be read directly without the program having to issue *read* SVCs.

The files are stored in persistent *memory objects* that are managed by the PSM—see Section 3.4.2. Each file (DF or EF) has a header and data area. In many cases these two areas are contiguous and are held in a single memory object. However, if the file is to be mappable into virtual storage, (which includes all executable programs), the file header cannot be accessible by the user mode program; hence in these cases, the file header and the file data area are in separate memory objects. Further, the size and alignment of the data area of a mappable file must be appropriate for the memory management unit of the processor.

In a conventional operating system, a directory contains a pointer to each of its children. Caernarvon does not do this, but instead each child has a pointer to its containing directory. This saves storage space, avoids any possible problems of maintaining consistency in DF file pointers and removes the need to manage a variable length list of children. Such a design has the consequence that file searches require examining many or all of the file headers present in the smart card; this is of little consequence when the file system consists of only a few files in a small (e.g. 64K byte) persistent storage. However, this algorithm does not scale to large hard drives.

Each DF (and hence all the EFs within it) has an associated mandatory security policy access class, and the File System calls the Access Class manager (see section 3.5) to

verify, on each file *open* request, that it is permitted to access the file with the currently authenticated access class. There are facilities to change the current access class of a DF (and all the files within it). This may have the result that access to a currently open file is no longer valid: in this case, immediate revocation is implemented to remove access to the open file(s) in order to avoid possible covert channels.

A DF may also have an associated quota, which prevents one application from exhausting all available memory, thus avoiding denial of service attacks and covert channels. The file system "charges" all space occupied by the DF and all EFs within it to this quota, and enforces the quota limit. If a DF does not have a quota of its own, the space is charged against the quota of the next highest DF in the file tree that *does* has its own quota. This quota mechanism is based on the Multics quota mechanism [28, section 3.7.3.1].

3.4.2 Persistent Storage Manager - PSM

As previously stated, the PSM provides for the persistent storage of *Memory Objects*. The PSM primarily supports the File System, but also supports the Access Class Manager and the Key Management System. The persistent storage medium in smart cards is frequently EEPROM, but it may also be FLASH memory. In general, the available EEPROM memory size is small, typically 128K bytes or less in current smart card processors.

Each memory object is created with the size specified by the requester; there may also be alignment requirements, for example for the processor's Memory Management Unit (MMU) for memory objects that are to be mapped into virtual address space. The PSM creates each object as a single contiguous area of storage, moving other objects if necessary, and returns a memory object ID to the caller. Memory objects are referred to by these IDs, not by the address, which would change if the object moved. Additional control information is required for each memory object. This information is held in a separate area of memory for compact storage and for protection from unauthorized access when the memory object itself is being accessed.

Smart cards have a potentially serious problem in that the power to the device can be removed at any instant simply by withdrawing the card from the card reader. On other embedded devices, battery removal or exhaustion could cause similar difficulties. Further, writing (programming) EEPROM or FLASH memory can take several milliseconds per operation, and a PSM request may require multiple memory write operations to complete. Thus it is very easy for the holder of the card to inadvertently interrupt a PSM operation, which could potentially leave the card in an invalid or corrupted state.

Consequently, it is imperative that all PSM requests be atomic—that is, each operation is either completely executed (all of the necessary memory write operations are successfully performed), or it is not executed at all. To this end, the PSM implements sophisticated algorithms using a *back-trace buffer*, where, for each step of an operation, the old contents of memory are saved before they are overwritten. Then, if the PSM request was not successfully completed, when the smart card is next powered up, the PSM can use the contents of the back-trace buffer to undo every stage of the operation, in reverse order. Each entry in the back trace buffer cannot be removed until that step of the operation is known to have been successfully undone.

Another problem with electronic persistent memory (EEPROM and/or FLASH) is that only a limited number of write cycles can be performed on any individual memory cell before the cell starts to wear and fail. Hence the PSM maintains checksums on its control blocks and all memory objects, and can then verify the validity of the data on accesses to the memory object. If errors start to occur, the PSM can attempt to recover the data, move it elsewhere, and mark that block of memory as defective. In addition, every write to persistent memory is verified—that is, after each memory write, the data that is now in persistent memory is compared to the input data to ensure that it is correct.

3.5 AC manager and Admin Application

The Access Class manager and the Administrative Application enforce and administer the mandatory security policy (see section 3.1). The Access Class manager is divided into two system components:

- Access class functions—these functions perform access class comparison and determine if access is permitted to an item (file, etc.). These functions depend on the system authentication functions. (see section 3.3)
- Access class SVC functions—these functions implement certain special system calls that are restricted to the Admin Application and are used for the administration (creation, deletion, etc.) of access classes.

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

3.6 Cryptographic Services

Embedded devices, such as smart cards, are frequently required to perform cryptographic operations; various standards specify cryptographic requirements, for example those for electronic passports, HSPD–12, or for online financial transactions. To meet these requirements, a smart card system must implement appropriate cryptographic facilities.

Smart cards in general, and their cryptographic features in particular, may be subject to attack, in particular side channel attacks [22, 1]. Defenses have been developed, both in the processor cryptographic hardware and in the crypto functions in software, to significantly mitigate these problems. However, for these defenses to work effectively, they must be used correctly. The Caernarvon system is designed to be high assurance, which implies that its crypto code must be carefully developed and tested to prevent information leakage. However, Caernarvon can run untrusted, unevaluated applications written by anyone, and if such applications perform cryptographic operations, the operating system has no guarantee that this code will not compromise the security of the cryptographic keys. To be trustworthy, the application crypto code would need to be evaluated under the Common Criteria, which is an expensive process. In addition to the financial penalty, unnecessary duplication of crypto code would squander critical memory resources.

The Caernarvon system, for the reasons stated above, includes full cryptographic facilities for applications which the application writers can depend upon to be secure. Further, to avoid key exposure, Caernarvon implements secure key storage, whereby keys can be securely loaded into the card and used within the system without the key ever being visible to the application.

Caernarvon currently supports DES and triple-DES, RSA up to 2048 bits, DSA and SHA–1. There are also random number generator facilities. Some Common Criteria certification agencies have objections [15] to pure hardware RNGs, on the grounds that they degrade with age. Hence Caernarvon does not depend on the hardware RNG, but instead has defined a new hybrid RNG algorithm [10].

3.7 Communications—the APDU Dispatcher

The communications channel of smart cards is defined by ISO 7816-3 [18]. This standard also defines the communications protocols (named T=0 and T=1) to be used. Smart cards may also provide a contactless (radio) interface, either instead of or in addition to the contact interface. However, the Caernarvon system currently supports only the contact interface, as the underlying chip does not support contactless. If Caernarvon were used in other embedded devices, additional communication protocols would be necessary.

When the smart card is first powered (by inserting the card into the reader), it sends the "Answer to Reset" to the reader; subsequently the card waits for commands sent from the reader. These commands from the reader, and the responses from the card, are called Application Protocol Data Units (APDUs). Each command APDU from the reader consists of a header, which includes bytes that specify the command plus an optional data part. Many APDUs are defined in ISO 7816; it is also permissible for a smart card system or application to define its own proprietary APDUs.

The Caernarvon system receives all incoming APDUs into the APDU dispatcher, which examines the APDU header to determine the command to be executed and hence determine to whom it should be routed. In order to provide the equivalent of a secure login that you might find in a conventional operating system, the Caernarvon system implements and responds to certain of the APDUs, notably those for authentication at the start of a new session (immediately following power–up). The APDU dispatcher also traps the SELECT command which is defined in ISO 7816-4 [19] to start a new application, and (assuming the specified program file is valid under the rules of the mandatory security policy), causes the specified program to be run. This is the equivalent of starting the command interpreter or shell of a conventional operating systems. It is up to the program to issue the response to the SELECT APDU. Subsequent incoming APDUs are passed to the application program, and the program implements the APDU and responds to it. If a SELECT APDU is received while an application is running, then the current application is notified that it is being terminated. Once that program terminates, the new application is started. Thus, the SELECT APDU constitutes the equivalent of a trusted path or secure attention key in a conventional operating system.

4. DOCUMENTATION

The Caernarvon operating system was designed from the start to be evaluated under the Common Criteria [12] at EAL7, the current highest level of assurance. This requires that the system have comprehensive documentation—this documentation has been written and maintained from the very start of the project.

The documentation for Caernarvon can be regarded as falling into a number of categories, including the system specification, system internals documentation, test system documentation (see Section 5), and development tool documentation. Large portions of the internals documentation and test generation is actually embedded in comments in the source code and test scripts. One of the development tools is a program that extracts the documentation from the source code and produces the necessary Common Criteria documentation in book form.

Meeting the strict documentation standards of the Common Criteria proved very helpful to us. ISO 7816–4 [19] has ambiguities that all implementations have had to resolve. However, other implementations have not clearly and openly documented how those ambiguities were resolved, leading to interoperability problems between different readers and smart cards. The strict documentation requirements forced us to identify these ambiguities early and to clearly document exactly how we chose to resolve them. This will help end users at least understand where the problem areas lie and how they were resolved in the Caernarvon OS.

5. TESTING

Part 3 of the Common Criteria [12] has requirements for extensive testing of the target of evaluation, particularly when the target is to be evaluated at a very high level. As a result, our tests need to be far more extensive than is common practice. There are very limited tools available for smart card platforms. Therefore, we have had to develop tools to replace those that are today readily available on standard development platforms.

To this end, we have developed a comprehensive test system, written in Ruby, that runs sequences of tests against the system. The test system executes on an external computer and communicates with Caernarvon (running in a smart card emulator) over the normal smart card communications line. This way almost all of the required intelligence required for the tests is outside the system under test. The test suite generates sequences of commands which are sent to a special test application that runs in the Caernarvon system under test. The test application executes each of these commands, and returns the appropriate data and/or status information. These test commands may, for example, require a test of an external API of the system with the parameters detailed in the command; many of the tests specify parameters that are invalid in some way, to verify that the system correctly traps the error cases. There are also facilities to test internal functions of the Caernarvon system.

Full testing of such a system requires running many thousands of tests, and it is necessary to verify that each test completes as expected, that is, the test succeeds and performs some action, or fails with the expected error indication. It is an extremely time consuming and error prone proposition to expect someone to check that all of the tests gave the anticipated results. Hence all of the tests are self checking, that is, each of the test scripts knows its expected result, and calls a Ruby method to check that the returned data (if any) and status match the explicitly indicated expected results.

6. CAERNARVON PROJECT INFLUENCE

Through the Caernarvon project, IBM Research has had significant external impact in the areas of cryptography, mandatory security policy, and authentication.

The Caernarvon cryptographic library implementation has completed the Common Criteria evaluation process and has earned EAL5+ certification. This cryptographic library has been made available to the community of developers implementing software for the Smart*XA*2 processor.

The Caernarvon mandatory security policy has been incorporated into the design of System S [29]—a high throughput, large–scale, distributed stream processing system used for analyzing voluminous amounts of unstructured data such as audio, video and data feeds, sensor output, and news reports. In particular the System Fuzzy Multi–Level security model [11] is directly based on Caernarvon.

The Caernarvon mandatory security policy also forms the basis of the Simple Linux Integrity Module (SLIM) that implements a low–watermark integrity policy for a Trusted Linux Client [24].

Finally, the Caernarvon privacy-preserving authentication protocol [26] protects the identity of the smart card holder from being divulged to an untrusted party (card reader). This protocol has been adopted by CEN as discussed in Section 3.3.

7. PROJECT STATUS AND FUTURE

Much of the Caernarvon OS has been implemented and tested on a smart card hardware emulator. The OS consists of approximately 33,000 executable C statements and 14,600 assembler statements. In addition to the executable statements, the source code includes approximately twice as many lines of comments, both in C and in assembler. The Ruby tests, test framework, and the OS and test documentation, when printed and stacked, could conceal a small child. We have developed a working demonstration of an electronic visa application, which allows friendly countries to read biometric data from and write entry/exit time stamps on each other's passports. Unfriendly countries can read only the public data, and cannot write anything. All countries (including the passport–issuing country) can be prevented from over–writing data from other countries.

The Caernarvon operating system was developed to demonstrate that a high assurance operating system for smart cards was technically feasible and commercially viable. The technical feasibility was partially demonstrated through the completion of the initial implementation of the operating system and a successful evaluation of a portion of the system at EAL5+. The commercial viability is still undetermined, because the complete OS has not been released as a product. However, explicit requirements for the features and security provided by the Caernarvon OS have been described in applications such as electronic visas and the next generation of SIM cards for mobile phones. The next generation of mobile phones are expected to run multiple simultaneous financial applications as well as video applications that are likely to require strong digital rights management and security isolation. This paper [6] is one of many discussing the needs of these next generation SIM cards.

Future work will continue to address applications of this technology as described in Section 6.

8. 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[4], 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.

9. REFERENCES

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

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

[3] Application interface for smart cards used as secure signature creation devices – part 1: Basic requirements. prEN 14890-1:2007, Comité Européen de Normalisation, Brussels, Belgium, March 2007.

[4] 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.

[5] 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.

[6] F. C. Bormann, L. Manteau, A. Linke, J. C. Pailles, and J. van Dijk. Concept for trusted personal devices in a mobile and networked environment. In *Fifteenth IST Mobile & Wireless Communication Summit*, Myconos, Greece, June 2006.

[7] G. W. Bush. Policy for a common identification standard for federal employees and contractors. Homeland Security Presidential Directive HSPD-12, The White House, Washington, DC, 27 August 2004. URL: http://csrc.nist.gov/policies/Presidential-Directive-Hspd-12.html.

[8] R. Canetti and H. Krawczyk. Security Analysis of IKE's Signature-Based Key-Exchange Protocol. In *Advances in Cryptology - Crypto 2002*, pages 143–161, Santa Barbara, CA, 2002. Lecture Notes in Computer Science, Vol. 2045, Springer Verlag.

[9] M. G. Carter, S. B. Lipner, and P. A. Karger. Protecting data & information: A workshop in computer & data security. Order No. EY-AX00080-SM-001, Digital Equipment Corporation, Maynard, MA, 1982.

[4]now with Giesecke & Devrient

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

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

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

[13] Department of defense trusted computer system evaluation criteria. DOD 5200.28–STD, Washington, DC, Dec. 1985. URL: http://csrc.nist.gov/publications/history/dod85.pdf.

[14] E. W. Dijkstra. The structure of the THE multiprogramming system. *Communications of the ACM*, 11(5):341–346, May 1968.

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

[16] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC 2409, November 1998. URL: ftp://ftp.rfc-editor.org/in-notes/rfc2409.txt.

[17] Information technology security evaluation criteria (ITSEC). Version 1.2, Commission of the European Communities, Brussels, Belgium, June 1991. URL: http://www.ssi.gouv.fr/site_documents/ITSEC/ITSEC-uk.pdf.

[18] Identification cards - Integrated circuit(s) with contacts - Part 3: Electronic signals and transmission protocols, Second edition. ISO Standard 7816-3, International Standards Organization, Dec. 1997.

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

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

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

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

[23] H. Krawczyk. SIGMA: the 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and its Use in the IKE Protocols. In *Advances in Cryptology – CRYPTO 2003 Proceesings*, pages 399–424, Santa Barbara, CA, 17-21 August 2003. Lecture Notes in Computer Science, Vol. 2729, Springer Verlag.

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

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

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

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

[28] 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. URL: http://csrc.nist.gov/publications/history/whit74.pdf.

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