MULTOS

KEYCORP LIMITED

Technical Paper

Composite Evaluations: Evaluations of Applications in High Security Multiple Application Modules

Authors: Brian McKeon, PhD, & Raymond Makewell

Abstract: *In this paper we discuss the emergence of Multi-Application High-Security Modules, in particular smartcards, and the impact on FIPS140 security evaluation techniques. The FIPS140-2 standard has been used for evaluation of such modules however the evaluations typically involve a single application or a fixed set of applications, preventing the mix-and-match of applications that are desired by some end-users. This paper describes product security requirements, derived from FIPS140-2, that would allow security evaluations of different application configurations to be accepted as a composite evaluation spanning a set of applications.*

Introduction

Security design principles require that security modules have the minimum of functionality required for the intended purpose. This ideal is often compromised with modern High Security Modules (HSMs) because of the desire of HSM designers to offer single multi-purpose architectures to cater for a wide variety of applications. This compromise does have a trade-off benefit for security in that it allows HSM designers to concentrate on the security of a single architecture rather than the effort being diluted across many dedicated designs. Good examples of flexible HSM products are multi-application smartcards such as JavaCard [JavaCard] and MULTOS [MULTOS] and personal

computer-based products such as the IBM 4758 PCI card [IBM4758].

Some HSMs address the multi-purpose requirement by assuming that fundamentally all that is required is a key storage sub-system which allows various applications to store their keys within the HSM and perform simple cryptographic transforms within the HSM. This does address many typical requirements however there are some applications which require that a sequence of operations, a "secure transaction", be performed within the HSM which leads to the requirement that the HSM be capable of being programmed with some form of executable code.

When the HSM environment needs to support a number of such transaction applications, the requirements on such an environment are the same as those on any secure multi-application operating system [GASSER-88].

Current HSM security evaluation standards such as FIPS140 originated in the early to mid-1990s [FIPS140-1] and are based on fixed-functionality (when first introduced, the IBM 4758, which had 486 processor hardware capable of multi-application support, had a software architecture only able to support a single application). If a multi-application HSM is presented for security evaluation it is typically presented with an application or set of applications configured for an end-customer. When the set of applications is varied the HSM would require to be presented for evaluation again, a significant cost penalty, and delaying the introduction of security solutions.

In particular, modern smartcard systems are being offered with a variety of applications and the ability to combine security evaluations of different applications on a smartcard to produce a composite evaluation would be a very desirable outcome.

This paper suggests a basis whereby security evaluations of different applications can be combined without requiring a further evaluation. This does have some impact on the security architecture of the underlying operating systems and hardware.

Smartcard Platforms

Multiple-application smartcard technology has now matured to the point where it is in widespread use in mobile phones, banking, user authentication and user data protection. The dominant platforms are currently JavaCard [JAVACARD] and MULTOS [MULTOS]. Both platforms have a similar core architecture with a small virtual machine interpreting application bytecode. This model allows applications to be run, without change, on different hardware. [SC2002] is a good introduction to smartcard technology and a number of different platforms, including JavaCard and MULTOS.

To illustrate the similarity of the JavaCard and MULTOS platforms the following text describing the MULTOS system was taken from [MULTOS-ST].

MULTOS is an operating system for integrated circuit cards (also known as smartcards). It is designed to allow multiple smartcard applications to be securely loaded and executed on a smartcard.
The user of the smartcard accesses the applications loaded on it via an Interface Device (IFD), which could be a Point-of-Sale terminal, Automatic Teller Machine, or some other device which supports ISO 7816 [ISO7816] smartcard protocols.

Communications across the IFD-MULTOS interface comprise a message transmitted by the smartcard when it is reset (the Answer-to-Reset or ATR message), followed by command-response pairs, where a command is a message from the IFD to MULTOS and a response is a message from MULTOS to the IFD.

By means of these command-response pairs, MULTOS allows:

a) Applications to be loaded onto and deleted from the smartcard.
b) An IFD to access data and applications which are loaded on the card.
c) Information specific to the card to be retrieved by an IFD.

MULTOS is a single-threaded operating system. Only one application can be executing at any given time. MULTOS does not provide mechanisms for concurrency or multi-tasking. Following power-on of the smartcard and initialisation, the basic execution sequence for MULTOS is as follows:

a) Wait for input from the IFD.
b) Parse the input.
c) If the input is a MULTOS command, process the command and write a response to the IFD.
d) Otherwise, execute the currently selected application and write to the IFD any output created by the application.
e) Loop back to a).

Applications to be loaded on MULTOS-based smartcards are written in a hardware-independent language called MULTOS MEL. MEL applications

are interpreted by MULTOS, rather than being compiled and executed directly on the smartcard processor.

If the word "MULTOS" in this text was replaced by "JavaCard" and the "MULTOS MEL" by "JavaCard bytecode" the text is then just as valid for JavaCard, highlighting the similarity of the two architectures.

Analysis Method

The intent of this paper is to determine potential revisions of the FIPS140 standard. The analysis starts with a review of operating system platform requirements from FIPS140-2, including the referenced Controlled Access Protection Profile [CAPP]. These requirements are compared with typical security targets of the MULTOS and JavaCard platforms and referenced protection profiles. A cross-check on the outcome is then performed against core smartcard operating system requirements as identified in a multi-application smartcard protection profile [PP/0010] referenced by the JavaCard and MULTOS Security Targets.

Review of FIPS140-2

The FIPS140-2 documentation has specific requirements for operating systems that support the cryptographic functions.

In the following analysis different sources use the terms process, thread and application for similar concepts. As this paper is primarily interested in smartcard operating systems which are single-thread, single-process for any one command-response pair (ref [ISO7816]), the term application will be used. This is more familiar in the smartcard world.

The following extracts are intended to highlight aspects that are relevant to multi-application, multi-function operating systems that can be reconfigured with different application functionality:

The principles behind the evaluation at the different security levels of FIPS140 are:

- Plaintext key material of one application within the HSM will not be accessible to other applications
- The code and data of an application within the HSM will not be readable nor writeable by another application or by an entity external to the HSM.
- Applications spawned by an application within the HSM are owned by the spawning application and no other
- During execution, an application of the HSM cannot be interrupted by another application (not relevant to the smartcard application execution model).
- Application software will be installed in the HSM in a form that prevents tampering and allows confidentiality
- A confidential channel mechanism will be provided to applications, the confidentiality being protected from other applications and from entities external to the HSM, excepting the intended communicating entity
- A role-based authentication mechanism will be provided
- The loading/installation of applications to the HSM will be restricted to authorized roles
- The execution of applications in the HSM will be restricted to authorized roles
- The (re)loading of key material to the HSM will be restricted to authorized roles
- The use of key material in the HSM will be restricted to authorized roles
- Recording of the above actions or rejection of failed actions to be recorded in an audit log.

Controlled Access Protection Profile

For evaluation at FIPS140-2 security levels 2 and above, a supporting operating system is required to be evaluated under Common Criteria [CC] to the Controlled Access Protection Profile [CAPP] developed by the NSA.[1] [2]

The [CAPP] Protection Profile states that the “assurance level is EAL 3 and the minimum strength of function is SOF-medium.”. This EAL3 level is really only directly applicable to FIPS140-2 evaluation up to level 3 which specifies that the supporting operating system be evaluated to CC EAL3. As a level 3 evaluation is tending to the desired target of smartcard evaluations the CAPP is directly relevant to the analysis of this paper.

The CAPP profile clarifies the generalised FIPS140 “roles” into three fundamental divisions, (unauthorised) users, authorised users and (authorized) administrators.

[1] FIPS140-2 provides for “an equivalent evaluated trusted operating system” to a Common Criteria evaluated operating system. The analysis in this paper focuses on the more regular Common Criteria-based path.

[2] In any revision of FIPS140 it is likely that Appendix A of FIPS140 would reference the multi-application smartcard protection profile PP/0010 as an alternative to CAPP.

Tests will be run at startup, periodically or on administrator request, to demonstrate correct operation of security assumptions including domain separation [CAPP, s5.5.1.1]. CAPP also states that this generally refers to the hardware platform with the stated assumption that the hardware platform was providing the domain separation mechanism. FIPS140-2 allows that administrator request can be implemented as the ability to reset or power cycle the HSM to trigger a power-on self test.

Domain separation is further clarified [CAPP s5.5.3] as providing a separate domain for the operating system and separate domains for each application. This prevents the operating system from interference from untrusted subjects and prevents applications from interference from other applications.

Security Policy enforcing functions should be invoked and succeed before each attempt to perform an action between a subject and an object that requires security enforcement [CAPP s5.5.2]. The intent here seems to be to avoid use of security enforcing functions that are invoked either remote in time or code position, to the point of object access. Such remoteness allows more likelihood of design flaws and opportunities for security attacks.

Object reuse is addressed in CAPP s4.1 which states that information in protected resources must be destroyed when the resource is recycled.

FIPS140 Requirements Compared to Smartcard Security Targets

It is expected that next revisions of the FIPS-140 will continue to align with Common Criteria for generic requirements such as operating systems and will probebly include a reference to the smartcard-specific protection profile [PP/0010] as well as [CAPP].

The above emphasis towards Common Criteria and the increasing availability of smartcard security targets based on common protection profiles means that the Common Criteria is a useful basis for comparison of the different solutions against FIPS-140 requirements.

Although the Common Criteria appears to be a common basis for the different security targets, the available security targets for Javacard and MULTOS cannot be directly compared. They do not all span the same lifecycle phases and, although the security targets would be expected to be individually consistent with referenced protection profiles etc, the different security targets have chosen difference approaches. So there needs to be some degree of interpretation when trying to compare the different documents. Previous attempts to develop smartcard security targets addressing multiple protection profiles have identified similar problems [GAMMA-02].

For example, the multi-application smartcard lifecycle is defined with seven phases from design through to end use and end of life [PP/0010]. The heavily-referenced multi-application smartcard protection profile [PP/0010] evaluation scope only covers phases 1-3 up to the end of IC manufacture and test. The Global Platform Security Target Guidelines [GP-ST], even though recent and heavily based on the multi-application Javacard, also only span lifecycle phases 1-3. The Gemplus 2002 (EAL4) ST [JC-ST-GP-1] scope does not include application loading after card production as this seems to have been intended for a SIM market with fixed application set. The Oberthur 2002 (EAL4) ST [JC-ST-OCS-1] covers only phases 1 (prepersonalisation requirements and software design) and 7 (end-use). The MULTOS ST [MULTOS-ST] spans the full product lifecycle.

[MULTOS-ST] and [JC-ST-OCS-1] were chosen for further analysis as both documents spanned the lifecycle phases to do with product design and end-use including application loading and deletion in an unprotected environment.

In the following text, when a Common Criteria heading abbreviation such as FDP_ACC is referenced it will be prefixed by a “CC:” to distinguish from references to documents.

Plaintext key material of one application within the HSM will not be accessible to other applications

Smartcard operating systems generally have no mechanisms to distinguish critical data stores of applications.

The Javacard architecture allows for applications to store key material within Security Domain applications however this is more of a convenience for sharing of keysets across applications where a number of applications may share a security domain because they share a common application issuer for instance. A JavaCard application is free to store key or other sensitive material within its dataspace.

The Security Domain architecture of JavaCard has an impact on application separation in that, when an application is deleted, it's key material should also be deleted. If this key material is stored in a separate application, a Security Domain, then this

key material should also be deleted. At present this is the responsibility of the application owning the key material and the operating system platform does not manage this.

MULTOS has not defined a Security Domain application concept although it could support this model. Present MULTOS applications store key material within their data spaces. Therefore, like JavaCard, the underlying operating system cannot determine what data of an application is key material or otherwise sensitive material.

The operating system needs to treat all application code and data as sensitive. This FIPS-140 key material requirement has therefore been generalized to the next FIPS-140 requirement concerning protection of code and data of an application.

The code and data of an application within the HSM will not be readable nor writeable by another application or by an entity external to the HSM

There are three aspects to be covered here. The first concerns protection of application code and data while it is loaded on the HSM, the second concerns protection of code and data when the application has been deleted from the HSM and application resources released and the third concerns access by an external entity.

Application Separation

There is some ambiguity in in [MULTOS-ST] and [JC-ST-OCS-1] with respect to the use of the term "Security Domain" and it is useful to first examine [CAPP] to try for a basis of comparison.

CC:FPT_SEP.1, Domain Separation, is the CAPP requirement for separation of the domains of the operating system and applications, each consisting of code and data resources.

[JC-ST-OCS-1] addresses CC:FPT_SEP.1 TSF, but specifically states that this is for Security Domains, specialized JavaCard applications that are intended for keys and passwords. Other applications call the Security Domain applications for specific services. [JC-ST-OCS-1] does introduce a firewall concept but this is not under Domain Separation and is introduced under CC:FDP_ACF.1 to do with access control. JavaCard needs to introduce a firewall at this more abstract level as application structure in JavaCard can be distributed into a number of objects with objects capable of being in a number of states, e.g. active, inactive, sharable etc.

A recent formal model of JavaCard [JAVACARD-FM] attempts to determine security requirements of this platform. However it is focussed on the Security Domain model where Security Domains are repository of all cryptographic keys, cardholder PINs (CHVs) etc. This leaves any critical application state variables in an unclear position. The need or otherwise for off-card application bytecode verification (see next) was also not stated in this proof.

Current JavaCard technology is based on an off-card bytecode verifier which "ensures that the CAP file has the correct format. The bytecodes are verified using a simple theorem prover which establishes a set of structure constraints on the bytecodes" [JC-ST-OCS-1]. [JAVACARD-SEC] states that, without this off-card verification, a malicious application could run and domain separation could be threatened. With applications loaded over a trusted path from the card issuer the card issuer can ensure that the verifier is run on all applications that are to be loaded on the JavaCard. This does place reliance on the off-card bytecode verifier and maintenance of the integrity of that tool. This is further complicated by bytecode verification not being 100% specified [JAVACARD-SEC].

The MULTOS application architecture is more simplified and application separation has a direct mapping to CC:FPT_SEP.1 subdivided into 1.1 and 1.2. Subsection 1.1 defines that the Operating System will maintain a security domain for its own use and subsection 1.2 states that the Operating System will maintain security domains for each of its subjects (applications). Note the use of the term "security domain". This is an abstract term for [MULTOS-ST] but describes applications with specific functionality in the [JC-ST-OCS-1] model. The concept of an application firewall in MULTOS would be directly traceable to FPT_SEP.1.

The MULTOS domain separation model is fully implemented on-card. MULTOS places no requirement on any off-card pre-processing of application code.

Release of Application Resources

[CAPP] s4.1 indicated that information in protected resources must be recycled when the resource is recycled (CC:FDP_RIP). Both [MULTOS] and [JC-ST-OCS-1] directly address CC:FDP_RIP.1 with similar techniques.

See also the later section on confidential channel mechanism for object reuse of the communications buffer.

The release of application-owned key material was discussed under the previous FIPS140 topic.

Access to Application Code and Data by External Entities

As all access to application code and data is mediated by the operating system, this requirement falls under access controls provisions, CC:FDP_ACF. An authorised external entity will be allowed to load an application, perform certain state changes on an application, and delete an application. Any other access to application code or data by entities external to the HSM is under control of the application itself.

With JavaCard or MULTOS the applications are written in an interpreted bytecode and domain separation is commonly performed in software as part of the operation of a virtual machine running the application bytecode. CAPP assumed hardware-enforced domain separation and required testing of the domain separation hardware. This is not relevant with software-enforced domain separation, if the domain separation software has not been tampered-with then the separation mechanism will be operational. With virtual machine-based domain separation the CAPP test of s5.5.1.1 can therefore be translated into a consistency check on the domain separation software. Typically this would be part of a general operating system code check on startup.

Applications spawned by an application within the HSM are owned by the spawning application and no other

This requirement is not obviously traceable into [CAPP]. Most multi-application smartcards do allow one application to spawn another. The smartcard model is a single-threaded model and therefore the calling application waits until normal completion of the spawned application. This FIPS140 requirement is therefore naturally satisfied by current smartcard environments.

JavaCard applications can publish interfaces as shared and such interfaces are callable by other applications on the card. When the interface is called the current context is switched, and access permissions are then those of the called (spawned) application. [JC-ST-OCS-1] does not explicitly link this delegation of control to any specific Common Criteria clauses and the only implied clause would be access control of different applications ie CC:FDP_ACF.

MULTOS has a less complex spawning method. Smartcard applications in JavaCard or MULTOS accept data in a command buffer from an external entity, process the data, and assemble a response which is then transmitted back to the external entity. MULTOS allows one application to assemble data in the command buffer and then delegate processing to another application. The other application processes the data as if it were received from an external entity and returns the response to the delegating application. Applications need to be designed with one interface method for either external entities or intra-application calls. [MULTOS _ST] does discuss the concept of switch of application context during delegation (see “Application Execution Management SF”) but this is not linked to any specific Common Criteria clauses. As with JavaCard the only implied clause would be access control of different applications together with context switching ie CC:FDP_ACF.

There is no direct impact of this aspect of FIPS140 on the mix and match of application security evaluations.

Application software will be installed in the HSM in a form that prevents tampering and allows confidentiality

Javacard and MULTOS both provide mechanisms to allow an issuer to encrypt and sign applications to provide confidentiality and integrity of loaded applications.

The JavaCard loading fundamentally relies on FTP_TRP.1, Trusted Path.

MULTOS loading is based on authentication/identification of an authorised external entity (CC:FIA_UAU.1 and CC:FIA_UID.1) and encryption for confidentiality (CC:FDP_ITC.1 with support from CC:FCS_COP.1, CC:FCS_CKM.3 and CC:FCS_CKM.4).

The two approaches are logically equivalent with the trusted path of JavaCard more closely associated with on-line application loading techniques whereas MULTOS allows applications to be encrypted and signed on-line or off-line if the smartcard is not on-line at that time.

There is no direct impact of this aspect of FIPS140 on the mix and match of application security evaluations.

A confidential channel mechanism will be provided to applications, the confidentiality being protected from other applications and from entities external to the HSM, excepting the intended communicating entity

The FTP_TRP.1, Trusted Path, function of JavaCard is available to applications and provides a confidential channel to applications.

The MULTOS platform does not provide confidential channel support due to the lack of a standardized mechanism in the smartcard market. Applications construct their own confidential channel mechanisms using MULTOS cryptographic functions.

Both MULTOS and JavaCard have a portion of memory that is used for communication with external entities, is shared between applications, may contained unwrapped (decrypted) data, and is therefore subject to object reuse. Both platforms have policies of clearing this buffer on application selection or card reset.

A role-based authentication mechanism will be provided
Operating system roles with JavaCard and MULTOS fit with the CAPP model ie (unauthorised) users, authorised users and (authorized) administrators.

At least one smartcard security target [JC-ST-OCS-1] abstracted the concept of an application as a user. This looks correct as the applications are clearly users of operating system resources but the domain separation aspects of these users as applications tends to become confusing.

Although essential for controlling access there is no direct impact of this aspect of FIPS140 on the mix and match of application security evaluations.

The loading/installation of applications to the HSM will be restricted to authorized roles
In both JavaCard and MULTOS this requirement is satisfied by the mechanisms used to ensure secure loading of applications (see earlier).

There is a requirement placed on the entity authorised to load applications to ensure that application IDs are allocated in a controlled manner. This to avoid the possibility of one application masquerading as another. This requirement would typically fall on the card issuer.

FIPS140 does not specifically mention control of application deletion. The mechanisms for deletion mirror those of loading for both JavaCard and MULTOS and fall under the access control provisions, CC:FDP_ACC.

The recycling of application resources has been discussed in a previous section and this requirement from FIPS140 does not otherwise impact the mix and match of application security evaluations.

The execution of applications in the HSM will be restricted to authorized roles
Neither JavaCard nor MULTOS directly enforce the concept of restriction of execution of applications. Any such restriction is the responsibility of individual applications.

This requirement from FIPS140 does not directly impact the mix and match of application security evaluations.

The (re)loading of key material to the HSM will be restricted to authorized roles
The JavaCard Card Manager application can have key material loaded via a trusted path (FTP_TRP.1) to the Card Issuer.

The MULTOS operating system key material is configured at one point, the loading of the MULTOS Security Manager (MSM) Controls. After this point the key material cannot be reconfigured. This loading is under the control of the card issuer with generation of the MSM controls delegated to the MULTOS Key Management Authority (KMA). Refer to Management of TSF data (CC:FMT_MTD).

This requirement from FIPS140 does not directly impact the mix and match of application security evaluations.

The use of key material in the HSM will be restricted to authorized roles
Javacard and MULTOS operating systems use key material for validating the loading of applications and, in the case of JavaCard, allowing card issuer authentication to allow certain application state transitions. This operating system key material can therefore be exercised by unauthorised users but with only a success/fail outcome. This key material is therefore open to cryptographic attack and is protected by audit mechanisms that restrict the possible number of unsuccessful attempts, blocking the card from further attack (CC:FAU_ARP.1, Security Alarms).

For application key material the JavaCard architecture with its concept of Security Domain applications would require that key material within a Security Domain application be owned by certain application(s). This is not a direct requirement on the JavaCard operating system platform and therefore must be managed by the application(s).

Recording of the above actions or rejection of failed actions to be recorded in an audit log
[CAPP] states that reliable timestamps are required for audit records however no current commercial smartcard technology offers real-time clock

functionality. Administrative alerts are also a difficult concept where smartcards are intended to be used for off-line authentication. The Department of Defense PKI Smart Card [DOD-PP-PKI] concluded that auditing functions were not relevant to system requirements on a smartcard operating system.

JavaCard maintains an audit log of certain exceptions (CC:FAU_LST). JavaCard and MULTOS maintain counters of critical security events that indicate a potential security threat and will either terminate an application or terminate the card under specific conditions (CC:FAU_SAA.1).

FIPS140 Requirements Compared to Smartcard Protection Profile PP/0010

The following list of requirements was identified as core for a multi-application Smartcard Operating System in [PP/0010].

- Integrity and confidentiality of Native and or Loaded Applications. 3
- Prevention of encroachment of loading and unloading of applications on Loaded-Applications
- Maintaining secure Domain separation of Loaded-Applications
- Integrity and confidentiality of Native and Loaded-Application TSF data.
- Integrity and/or confidentiality of End User Data which have been stored on the TOE when it is required. (For example result of health check-up, audit tracks of financial transactions…),
- Correct operation of arithmetical functions (e.g. incrementing counters in electronic purses, calculating currency conversation in electronic purses...) which are part of the security chain of the system using the TOE.
- Correct operation of application cryptographic functions when required (e.g. electronic signature for legal recognition , e-commerce...) which are part of the security chain of the system using the TOE.
- Contribution to secure data communication,
    - Ciphering and/or stamping of exported data
    - Deciphering and/or origin verification of imported data

Although not in the above table, management of unexpected reset or power loss is implied and is a critical function of smartcard design (refer “tearing” in [SC2002]).

The current analysis is intended to achieve the mix-and-match of security evaluations of different applications running on an evaluated operating system. This relates to the first five points above.

The only point that appears additional to requirements identified from the FIPS140 analysis is the second bullet point to do with “encroachment” of loading and unloading of applications on loaded applications. Unfortunately the text of this item is not directly linked to subsequent analysis in [PP/0010] and the term “encroachment” does not appear elsewhere in that document. A cross-check based on the threats to card lifecycle stages 6 and 7 in [PP/0010] show that this point does not introduce any new requirements to those already identified from the FIPS140-2 analysis.

Card tearing is not well addressed by FIPS140-2 nor CAPP. Power resetting is mentioned but in the scope of initiation of self-test. The maintenance of system transaction information across inadvertent power loss, if required, seems to be expected to be managed by backup power. Smartcards often need to maintain transaction integrity and without the option of backup power sources.

The security-enforcing functions of the operating system should be resistant to interruptions in supply of external power. Any critical activities of the security-enforcing functions should be atomic such that the HSM remains in a controlled state at all times (a valid state might be a shutdown HSM if the HSM determines that it is unable to recover from a certain reset condition). This should include security-enforcing functions involved in domain separation and application loading and unloading/deletion. This is not core to the mix-and-match of applications but is a general requirement on smartcard operating systems.

[3] For the purposes of this analysis Native applications are not considered further as they bypass the virtual machine and therefore should be considered as part of the operating system. The conclusion would be that such components should be part of the operating system security evaluation and beyond the scope of any application mix-and-match. In this document an “application” is defined as containing code that is only executed by the virtual machine of the operating system.

Conclusions

The mix and match of security evaluations of applications on a given HSM operating system is feasible if the following constraints can be achieved:

- All resources associated with an application must be clearly identified. This would include application code, application data, application-owned keystores or other stores that may reside outside the application data space, operating-system application context, and any references to the application resources that may be allowed from other applications.
- The management of these resources during the application lifecycle from creation through to application deletion must be identified.
- The operating system must have a robust firewall mechanism that supports the separation of application resources. This should not be dependent on any off-card preprocessing of application code or data.
- If execution of application code is based on an operating system virtual machine then the relevant operating system code of the virtual machine must be subject to an integrity check.
- Resources shared across applications, such as communications buffers, must be clearly identified and appropriate defenses identified.

References

- [CAPP] Controlled Access Protection Profile (CAPP), Version 1.d, Protection Profile NoPP006, (United States) National Security Agency (NSA), 8 October 1999.
- [CC] Common Criteria for Information Technology Security Evaluation, Version 2.1, August 1999.
- [DOD-PP-PKI] Department of Defense Public Key Infrastructure and Key Management Infrastructure Token Protection Profile (Medium Robustness) Version 3.0 22 March 2002.
- [FIPS140-1] FIPS PUB 140-1, Security Requirements for Cryptographic Modules, US DOC/NIST, January 11, 1994.
- [FIPS140-2] FIPS PUB 140-2, Security Requirements for Cryptographic Modules, US DOC/NIST, May 25, 2001. (also - Change Notices 2, 3 and 4: 12/03/2002).
- [GAMMA-02] A Security Architecture for Global Platform Smart Cards, Brewer et al, e-Smart, Nice, France, 2002.
- [GASSER-88] Building a Secure Computer System, Gasser, M, Van Nostrand Reinhold, New York, 1988.
- [GP-ST] GlobalPlatform Smart Card Security Target Guidelines, Draft v0.99c for Public review, February 2005
- [IBM4758] Building a High-Performance, Programmable Secure Coprocessor, S Smith, S Weingart, IBM Technical Report RC21102, Feb 17, 1998.
- [ISO7816] Identification cards -- Integrated circuit cards – Parts 1-4
- [JAVACARD-FM] Formal Specification of GlobalPlatform Card Security Requirements, Beguelin, S, INRIA, Sophia Antipolis, 15 Dec 2004.
- [JAVACARD-SEC] Computer Security from a Programming Language and Static Analysis Perspective, Xavier Leroy, The European Joint Conferences on Theory and Practice of Software (ETAPS), Warsaw, Poland, 2003.
- [JAVACARD] JavaCard V2.2.1 Platform Specification, http://java.sun.com/products/javacard/specs.html
- [JC-ST-OCS-1] Cosmopolic 2.1 Version 4, JavaCard Open Platform Security Target, Oberthur Card Systems, 2002
- [JC-ST-GP-1] ASE - Security Target, Java Card Platform Embedded Software V3 (Core). GemXplore Xpresso V3, Version A00P, Gemplus, 2002
- [MULTOS] MULTOS, The high Security Smartcard, http://www.multos.com/
- [MULTOS-ST] Keycorp MULTOS Common Criteria Security Target, Version 2.2, 23 April 2003.
- [PP/0010] Common Criteria for Information Technology Security Evaluation, Protection Profile, Smartcard Integrated Circuit with Multi-Application Secure Platform, Version 2.0, November 2000, registered by French Certification Board under number PP/0010.
- [SC2002] Smart Cards, The Developers Toolkit, Jurgensen, T, Guthery, S, Prentice Hall, NJ, 2002.

Keycorp Limited
Level 5 Keycorp Tower
799 Pacific Highway Chatswood
NSW 2067 SYDNEY Australia
Tel +61 2 9414 5200 Fax +61 2 9415 1363
Email: info@keycorp.net www.keycorp.net

Authors Contact Details
Brian McKeon
brian.mckeon@sentrypm.com
+61 (0)2 9410 1300

Raymond Makewell
Rmakewell@keycorp.net
+61 (0)2 9414 5200