Secure64 Cryptographic Module

Non-Proprietary Security Policy

FIPS 140-2 Level 2

For Version 1.2

December 2, 2009

SECURE64

Table of Contents

Introduction

Document Purpose

This document specifies the ***Cryptographic Module Security Policy*** for the Secure64® Cryptographic Module. The companion document, "*Secure64 Cryptographic Module – Cryptographic Module Specification*", provides a complete description of the Cryptographic Module and its compliance with FIPS 140-2 level 2 specifications.

The purpose of this document is to define a set of security rules and guidelines under which the Secure64 Cryptographic Module shall operate. Sufficient detail is presented to allow individuals and organizations to determine whether the Secure64 Cryptographic Module satisfies their stated security policy. A description of the capabilities, protections, and access rights provided by the module is provided to allow an assessment of whether the module will adequately serve organizational security requirements. This description is given in terms of identities, roles, security functions, cryptographic keys, and critical security parameters. The following items are specified:

- Identification and authentication policy
- Access control policy
- Physical security policy
- Design assurance policy
- Security policy for mitigation of other attacks

The FIPS Publication 140-2 specification states in its introduction: "While the security requirements specified in this standard are intended to maintain the security provided by a Cryptographic Module, conformance to this standard is not sufficient to ensure that a particular module is secure. The operator of a Cryptographic Module is responsible for ensuring that the security provided by a module is sufficient and acceptable to the owner of the information that is being protected and that any residual risk is acknowledged and accepted."

Table 1 on the following page lists each of the FIPS 140-2 Security Requirements sections, the FIPS 140-2 Security Level testing target for each section, and the properties and capabilities to be tested. These properties and capabilities of the Secure64 Cryptographic Module are described in greater detail in the subsequent sections of this document.

Testing and Certification Goals

FIPS 140-2 Section	Security Level	Properties and Capabilities to be Tested
Cryptographic Module Specification	Security Level 2	Cryptographic boundary; NIST approved algorithms; FIPS approved-mode-only operation; Documents of Security Policy and Cryptographic Module Design.
Crypto Module Ports and Interfaces	Security Level 2	Data in/out, Control in, Status out - over API and SPI interfaces controlled by Crypto Module; compliant with Level 2 constraints.
Roles, Services, Authentication	Security Level 3	Identity-based authentication for securityadmin Crypto officer, *SourceT* Crypto officer, and Application User.[1]
Finite State Model	Security Level 2	Specification and implementation of the finite state model.
Physical Security	Security Level 2	Standard HP H/W platform with intrusion detection tape on locking cover. Encrypted & signed on disk. Non-modifiable in RAM.
Operational Environment	N/A	Limited Operational Environment. No direct human operator or Crypto officer control of crypto module except reboot.
Cryptographic Key Management	Security Level 2	No manual or direct entry of CSPs. All CSPs are generated internally or are entered/removed in encrypted form.
EMI/EMC	Security Level 2	Operates on a standard HP platform. Shielded rack can increase RF shielding to exceed FCC regulations Part 15, Subpart A.
Self-Tests	Security Level 2	Power-on self-tests for integrity, cryptographic functions, and critical internal system functions. Continuous self-tests for PRNGs and asymmetric key pairwise consistency.
Design Assurance	Security Level 3	High Level Language; Tuned A/L for H/W system control and performance critical functions; Source code scanning; Build time module scanning.
Mitigation of Other Attacks – Crypto	Security Level 2	RSA Timing Attacks – RSA encryption/decryption is constant time for 1024-bit keys; blinding enabled for other key sizes.
Mitigation of Other Attacks - *SourceT*	N/A	Micro O/S immune to all forms of malware. Integrated network attack defenses. Physical intrusion response. All are outside of the Cryptographic Module. See Table 9, page 45 for details.

Table 1

[1] For the details of identity based authentication please see Section 1.4

Cryptographic Module Overview

The Secure64 Cryptographic Module is a static, non-modifiable firmware module designed for use only in systems based on *Secure64® SourceT®*, a static, non-modifiable limited operational environment running on an Intel Itanium- based server platform[2]. The Secure64 Cryptographic Module provides cryptographic functions that can be used by applications running in this environment. Example applications include DNSSEC signing (secure DNS using digital signatures), certificate management applications, etc. Example functions include key generation, secure key storage, encryption, decryption, hashing, and digital signing.

The Secure64 Cryptographic Module may be thought of as a firmware implementation of a Hardware Security Module (HSM) except that the interfaces to the device are a firmware API (Application Programming Interface) and a firmware SPI (Systems Programming Interface) rather than physical ports. Commands, data, and encrypted keys are sent from application programs to the Cryptographic Module by means of protected data paths. Results are also returned to the application using protected data paths. *Therefore, the comparison to a HSM is more than a simple analogy.* The Secure64 Cryptographic Module acts as a completely separate and independent "device" that shares the host server as its hardware platform.

The Cryptographic Module is installed at each system boot and is completely isolated by hardware protections from all other applications and code running on the server platform. While operating, the Module uses only minimal services from the underlying *Secure64 SourceT* micro operating system. These services act only in response to the Cryptographic Module, only when the module is active, and cannot in any way affect the contents of the Cryptographic Module when invoked by any other application or system component. Complete isolation of the Cryptographic Module is made possible by automatically enforcing configuration constraints during system design and build, and employing advanced Itanium execution-time hardware protections.

Configuration constraints are enforced by designing critical system components to operate only when called by the proper Cryptographic Module firmware, and by scanning intermediate modules at build time. These enforcements ensure that critical system components are only called by, and function only when called by, the correct Cryptographic Module firmware components.

Itanium hardware protections include memory PK's (Protection Keys). An Itanium protection key is NOT a cryptographic key. It is a 64-bit value containing a 24-bit field that can be used as a tag value for one or more memory pages in a system. If the virtual address mapping control for a page specifies a PK, the hardware may read, write, or execute code to/from that page only when the matching 24-bit tag value exists in one of the processor's 16 PKR's (Protection Key Registers). Erasing the PK from the processor register renders all the tagged pages inaccessible, no matter what the hardware privilege level. Restoring the PK to the hardware register reopens access to all the correspondingly tagged pages. Protection keys can be thus used to configure isolated compartments[3] within memory.

[2] We will refer to *SourceT* herein as a "micro operating system" or "operating system", even though it is a limited, but highly secure, control program providing a static, non-modifiable limited operational environment.

[3] The set of all pages tagged with the same 24-bit Protection Key value.

Itanium also permits marking all executable firmware and software loaded into memory as execute-only (not readable, and not writable). Further, Itanium manages its large set of processor registers using a hardware RSE (Register Save Engine). The RSE manages working general registers that contain all of the critical program control flow information as well as other data. The RSE performs register renaming, saving register contents to backing store RAM when more registers are needed, and restoring those values when the previous register contents are needed later. The backing store used by the RSE for information being saved/restored from/to the general registers is located in higher privilege level memory, and is inaccessible to applications and nearly all *SourceT* software.

These Itanium mechanisms allow the *Secure64 SourceT* micro operating system to configure an isolated memory compartment for the Cryptographic Module within the main memory of the host server platform. All CSPs and other data for the Cryptographic Module are managed within this compartment. It is not possible for any other applications or non-critical system software to access the Cryptographic Module memory compartment. Further, operations within the Cryptographic Module switch to and use an RSE reserved for code executing at a higher hardware privilege level, and the physical processor registers are cleaned before returning from every computation within the Cryptographic Module. This ensures that no critical information leaks back to the calling application via general registers. This level of hardware protection allows the Secure64 Cryptographic Module to operate in a manner similar to that of a separate hardware cryptographic device.

Limited Operational Environment

The *Secure64 SourceT* micro operating system is a static, non-modifiable limited operational environment. It is booted, together with its operator commands and product applications, by its loader that also, at each boot, installs the Secure64 Cryptographic Module. The Cryptographic Module conforms to all the requirements of FIPS 140-2 level 2.

- The software components of the *Secure64 SourceT* operating system, operator commands, and product applications, together with the firmware components of the Secure64 Cryptographic Module, are linked together into a cryptographically signed and encrypted monolithic load image on disk.
- Intermediate system build modules are scanned during the load image build process to ensure adherence to design constraints. These constraints include, in particular, specific limitations on function calls and data references among executable and data components.
- At boot time, the entire monolithic load image is decrypted and integrity checked using NIST certified digital signature algorithms (certificates #426, #874) to protect from unauthorized disclosure and modification.
- Once all software components are loaded, and the Cryptographic Module firmware installed, no other general-purpose operating system is present and no other applications can be dynamically linked or loaded.

- Once the system is loaded, and the Cryptographic Module installed, all executable software and firmware are protected by hardware. Executable code is not even readable, let alone writable, by any firmware or software.
- To protect plaintext data within the Cryptographic Module memory compartment, the access control mechanisms of the operating system and the Cryptographic Module each employ identity-authentication-based access control. The operating system also enforces additional role-based access control.
- The system architecture and implementation prevent operators , executing commands, or applications from observing or interfering with cryptographic computations within the Cryptographic Module. Similarly, hardware protections prevent such software from reading or writing cryptographic firmware, Critical Security Parameters (CSPs), or any other data within the Cryptographic Module.
- An audit system is present.
- All plaintext cryptographic keys, authentication data, and other CSPs are generated within the Cryptographic Module, or are written at boot time into the newly installed Cryptographic Module.[4] They never appear in plaintext outside the Cryptographic Module.
- All control inputs and status outputs are communicated by trusted firmware paths, controlled completely by Cryptographic Module firmware, as described in the Secure64 Cryptographic Module Specification.

The Host Hardware Platform

The Secure64 Cryptographic Module is designed to run on commercially available, general-purpose Itanium based server platforms. The server platforms have physical enclosures that provide a signal to the operating system when the cover is opened while the system is running, and also contain a Trusted Platform Module (TPM) chip.

If a hardware platform cover is opened while the system is running, the host cover-opened signal provides the opportunity for the operating system to take appropriate action to mitigate this attack, even though such mitigation is beyond that required by FIPS 140-2 level 2. The use of the TPM chip also falls outside of the requirements of FIPS 140-2 level 2 and is described in the next section.

The Secure64 Cryptographic Module has been operationally tested on the following hardware computing platforms:

- HP Integrity Server rx2660
- HP Integrity Server rx3600

Other platforms (both HP and non-HP) may be available in the future that provide the necessary

[4] As sanctioned by Section 7.7, *Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program* .

host functionality. They will be operationally tested before being recommended.

Role of The TPM Chip

The TPM chip is NOT a FIPS 140-2 certified component. As such, all of its use described herein occurs outside of the Cryptographic Module, and only when downloading the system image or booting the system. Because the TPM chip is completely outside of the Cryptographic Module, it's functionality is automatically excluded from FIPS testing. There is NO direct transfer of information between the TPM and the Cryptographic Module. Either can function without the other; neither requires a security function of the other.

The TPM was designed by the Trusted Computing Group[5] to perform a rich set of functions, but there are only three used within the *SourceT* limited operational environment. None of these uses is required by FIPS 140-2, and all uses occur completely outside of the Cryptographic Module. Secure64 employs them to offer additional protection to a system load image, and to ensure that an image can be loaded only upon a specific hardware platform. The three uses of the TPM are:

1. TPMs are manufactured and certified to contain a unique RSAES OAEP[6] 2048-bit public/private key pair, called the Manufacturer's EK (Endorsement Key). The private EK never leaves the TPM chip; the public EK can be read from the TPM. This public EK thus provides a unique identifier of the hardware platform containing the TPM chip.
2. When the TPM in a hardware platform is first initialized, a second key pair of the same type as the EK, called the SRK (Storage Root Key), is generated.[7] The private SRK also never leaves the TPM chip; the public SRK can be read only with restrictions. The public SRK is returned to Secure64 and used to wrap another RSA 2048-bit PKCS#1 v1.5 key called the BBK (Boot Binding Key) for use by the TPM. The BBK is used to protect both a 256-bit AES symmetric key and a 128-bit IV generated by a NIST certified PRNG (certificate #507). More details will follow in Section 2.3.
3. The TPMs in the hardware platforms supported by *SourceT* contain a hardware, non-deterministic random number generator. This provides an additional source of entropy when initializing PRNGs.

SourceT uses the TPM only for these three functions. All three are used only when downloading the system load image or booting the system. None are required by FIPS 140-2. All uses occur outside of the Cryptographic Module. The goals of these uses are:

1. Uniquely identifying a specific hardware platform.
2. Providing additional, non FIPS 140-2 required, protection for the AES key and IV used to decrypt the system image when booting the system.
3. Having an additional source of entropy for initializing the Cryptographic Module.

[5] www.trustedcomputingroup.org.

[6] PKCS#1 v2.1, RSA Laboratories, June 14, 2002. This recommends that all new application use this form of RSA key.

[7] The initialization process is known as "Taking Ownership" and will be described in Section 2.3.

Guidelines for Users

The Secure64 Cryptographic Module components are embedded within the system monolithic load image of a product based on the *SourceT* micro operating system. Once the system boots, and the Cryptographic Module is installed, a human operator who wishes to initiate computations that utilize the security functions of the *Secure64* Cryptographic Module must first authenticate his identity to the *SourceT* operating system, then enable a *SourceT* role he is authorized to assume. Having assumed a particular *SourceT* role, he then is limited to issuing the commands authorized for that specific role. Some commands function individually, such as the file editor or directory display commands. Other commands start, stop, or control execution of specific embedded applications.

There are no commands for a human operator to issue a specific API (Application Programming Interface) call to the Cryptographic Module. API calls are issued only as functionally required by embedded commands and applications[8].

Guidelines for Crypto Officers

The *SourceT* role, "**securityadmin**", is reserved for the *securityadmin* Crypto officer. Once the system boots, and the Cryptographic Module is installed, the security administrator must authenticate his identity to the *SourceT* operating system, and assume the *SourceT securityadmin* role. There are no commands for the *securityadmin* Crypto officer to issue API calls to the Cryptographic Module. There is one command for the *securityadmin* Crypto officer that issues the SPI (System Programming Interface) call to query the status of the Cryptographic Module. All other SPI calls are issued only by the *SourceT* operating system.

The Secure64 Cryptographic Module is NOT delivered as a stand-alone unit to be separately installed by an administrator. Its components are ALWAYS packaged together with those of the *Secure64 SourceT* limited operational environment. The operation of the Module has been designed to be as automatic as possible. There is no maintenance role or maintenance interface. *SourceT* automates routine tasks: there is no need for operator intervention when the system boots or shuts down. Only in the event of an unlikely internal self-test failure, or a system event that forces the Cryptographic Module to shutdown, is operator action required from the *securityadmin* Crypto officer– and then only to shutdown and reboot the system.

Each time the system boots, the Cryptographic Module is automatically installed and initialized, executes its self-tests, is populated with master keys, and begins operation in the FIPS Approved mode of operation. The Cryptographic Module always operates in the FIPS Approved mode – NO other operating mode is provided. In the event of system shutdown, or any failure of the Cryptographic Module while the system is booting or operating, zeros are written over the CSPs and all other contents of the Cryptographic Module memory compartment, and the Module is shut down. Here too, no *securityadmin* Crypto officer action is required, except possibly to issue the system reboot command.

There are NO standalone manuals for the Cryptographic Module. Instead, each product bundle which embeds the Secure64 Cryptographic Module has an administrator manual and system

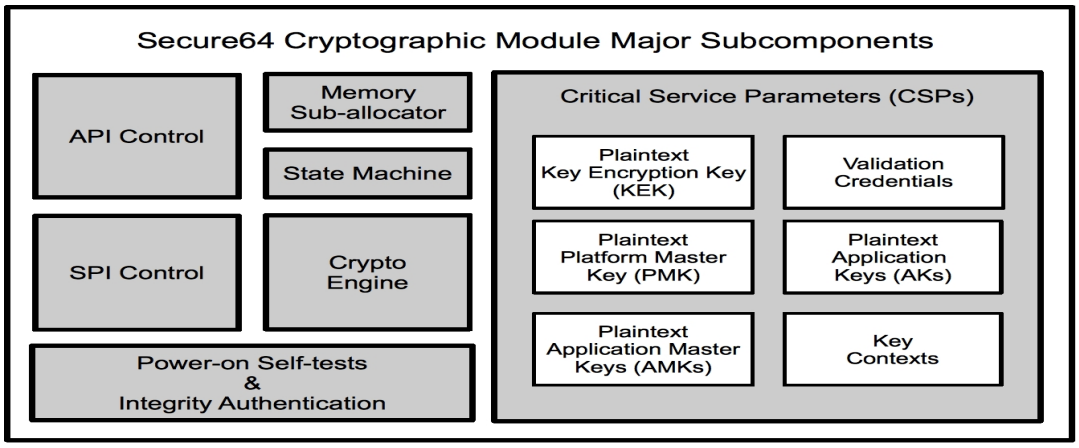
[8] The term "application" may henceforth be used to designate either a command or an application that makes calls to the Cryptographic Module.

operation guide which describes how to administer and operate that product bundle. Each product manual will give appropriate guidance for use of the Cryptographic Module – relevant to *securityadmin* Crypto Officers, as well as to other system administrators and application users. Product documentation contains information which describes:

- The purpose of the *SourceT securityadmin* role.
- The method for assigning the *SourceT securityadmin* role to a *SourceT* operator identity.
- The commands and files available to the *SourceT securityadmin* role; for example:
  - The "**enable securityadmin**" command.
  - The query Cryptographic Module status command.[9]
  - The commands to manage encrypted master keys.
  - The command to reboot the system.
  - Application-specific commands.
- The responsibilities of the *securityadmin* Crypto officer, which include:
  - Security administration compliant with local security policies.
  - Periodic inspections of physical security mechanisms:
    - Tamper-resistant seals.
    - Inspection of the host platform.
  - Examination of audit logs for security events.
  - Implementation of local response policies. Local security policies may optionally specify other steps to be taken when an event, such as evidence of damage to a tamper-resistant seal, occurs.
  - Implementation of disaster recovery procedures for key backup and restoration.

Figure 1 shows the major subcomponents within the Cryptographic Boundary.

Figure 1



[9] The only *securityadmin* command that makes a call to the Cryptographic Module.

1. Identification and Authentication Policy

1.1 Identification and Authentication

Access to the Secure64 Cryptographic Module is controlled in TWO stages – (1) human operator identity authentication and role assumption to the *Secure64 SourceT* micro operating system; and (2) *SourceT* Crypto officer, *securityadmin* Crypto officer, or embedded application User identity authentication to the Cryptographic Module. Note that each of these two stages authenticates an *identity*.

At stage 1 the identity and role are for a human operator. The login credentials supplied by the operator include the operator's login name, together with one, two, or three factor authentication information (see Table 3 in Section 1.2). The operator next must issue a separate command to enable an authorized *SourceT* role. This then permits an operator to issue one or more of the commands defined for that particular *SourceT* role.

At stage 2 the identities are for either the *SourceT* Crypto officer, *the securityadmin* Crypto officer, or the specific Application User in the monolithic system load image. The credentials supplied to the Cryptographic Module to authenticate the identity of the caller to the API or SPI interfaces of the Module are called "**caller's**" credentials. The caller's credential supplied in each API or SPI call is authenticated within the Cryptographic Module before access to the requested security function is granted. In all cases, this caller's credential is confirmed to be accessible only to the proper system code, command, or application at system build time. The caller's credential is authenticated by using a "**validation**" credential that is a CSP built into the Cryptographic Module at Module install time. More details of caller's and validation credentials are described below.

Human Operator Authentication

- First, a human operator must be able to log into the host platform. This is controlled by the *Secure64 SourceT* micro operating system using identity-based authentication.
- Second, once an operator has logged in, access to subsets of functionality (e.g., being able to use files, or run specific commands and applications) is controlled by *SourceT* using role-based authorization. An operator must have authorization for the proper role before the role functionality, whether commands or an application, becomes available to him.
- More details follow below in section 1.2. Although this stage of authentication is only for the *Secure64 SourceT* system, and not directly for authentication to the Cryptographic Module, it is explained here to aid the reader's understanding of the entire security framework. Of particular interest are the strength of the *SourceT* authentication schemes, and the means by which use of commands for the *securityadmin* Crypto officer are restricted to a specific *securityadmin* Crypto officer.

Operating System, Command, and Application Authentication

- Authentication to the Cryptographic Module is accomplished by inclusion of an appropriate caller's credential in each API or SPI call to the Cryptographic Module. A caller's credential is a 64-bit word, of which 56 bits are random data and the other eight bits contain an integer encoding the caller's identity. The 56 random bits are used for strong authentication of the caller's credential. This is summarized in Table 2.
- There are three types of caller's of the Cryptographic Module, each with a distinct caller's credential:
    - *SourceT* Crypto officer – this credential identifies the *Secure64 SourceT* micro operating system, and authorizes it to initialize or shutdown the Module, or to manage master keys. *SourceT* initiates operation of the Module after the system boots. The Module initializes itself, completes its power-on self-tests, and commences operation in FIPS approved mode. *SourceT* next imports encrypted platform and application master keys to prepare the Cryptographic Module to accept API calls from applications. The identity authenticated by this credential is: *SourceT* Crypto officer. The role authenticated by this credential is:

      SYS CRYPTO OFFICER.
    - securityadmin Crypto officer – this credential identifies a *securityadmin* Crypto officer function, and permits querying the status of the Cryptographic Module. The identity authenticated by this credential is: *securityadmin* Crypto officer. The role authenticated by this credential is: ADM CRYPTO OFFICER.
    - Application User – this credential identifies a specific application, and enables this application alone to use the sessions and cryptographic keys generated for or by it. The identity authenticated by this credential is that of the particular application. The role authenticated by this credential is: USER.
- The credentials for these three types of caller's are generated uniquely for each loadable system image. They are generated at product build time, and kept encrypted and digitally signed within the monolithic system load image. Build time code scanning ensures that when the system is loaded, each credential can be loaded only by its authorized function.

Identity	Role	Caller's Credential	Usage
SourceT Crypto officer	SYS CRYPTO OFFICER	Accessible only to the *SourceT* system.	Crypto Module initialization, Crypto Module shutdown, Master key mgmt.
securityadmin Crypto officer	ADM CRYPTO OFFICER	Accessible only to get status command.	Query Crypto Module status command.
Application User	USER	Accessible only to proper application.	API security functions.

Table 2

- A second category of credentials also is generated during the same build time process. This category of credentials is called "**validation**" credentials, and the randomly generated credentials are written into the Cryptographic Module when it is installed. Each validation credential within the Cryptographic Module comprises both a 64-bit **value** in the same format as a caller's credential, and a 64-bit **mask** word of 8 zero bits and 56 randomly generated bits. The mask random bits are aligned with the 56-bit random bits in the validation credential value. The value of each caller's credential is equal to the XOR of the validation credential value and the validation credential mask word. This defines how each caller's credential is calculated from the components of the corresponding validation credential. Mathematically, the caller's credential then can be authenticated within the Cryptographic Module by first decrypting it by XOR'ing it with the validation credential mask, then comparing the result with the validation value. In other words, the caller's credential is valid if and only if:

  Caller's credential XOR Validation mask = Validation value
  because
  (Validation value XOR Validation mask) XOR Validation mask = Validation value

1.2 Identity-Based Authentication to the Operating System

This section elaborates authentication to the *SourceT* limited operational environment, not to the Cryptographic Module. Operators of the host platform for the Cryptographic Module must have a login name before they can log in. They establish a connection to the host platform over a secure transport channel established with the SSH-2 protocol. The SSH-2 protocol is implemented in *SourceT* completely independently of the Cryptographic Module. Authentication processes are then invoked that make use of either a password or public key authentication to determine if the operator is allowed access to the host platform (illustrated in the following Table 3).

If the operator authenticates properly, the *Secure64 SourceT* micro operating system will then determine which *SourceT* roles are available. Recommended usage for a system containing a Cryptographic Module is to create only one operator who is assigned *securityadmin* Crypto officer responsibilities. Only this user would have the *SourceT securityadmin* role.

Having a login name and being properly authenticated by *Secure64 SourceT* only allows access to the host platform. Operators also must enable the proper *SourceT* role before they can proceed to issue commands or launch applications that use the Cryptographic Module security functions. Enabling a role will require the *SourceT* role name. No *SourceT* role, including the *securityadmin* role, ever enables direct access to a key[10] or CSP within the Cryptographic Module.

Table 3 summarizes identity authentication to the *SourceT* micro operating system:

[10] Except for applications handling the public portion of an asymmetric key pair.

Identity	Type of Authentication	Authentication Data
All human operators	Password or public-key authentication over an SSH-2 connection.	Login Name Password or public key

Authentication Mechanisms	Strength of Mechanism
Login Name and Password	Single factor authentication. Password policy requires minimum length of eight characters, three of which must be alphabetic and one of which must be numeric.
Login Name & public key authentication	Two factor authentication. Public/Private Key mechanism with public key stored on host platform and private key held by operator.
Optional 3rd party devices to provide 3-factor authentication data. Example is a Privaris key fob that releases public key data only when self-contained mechanisms such as biometric data allow release of the public key data.	Three factor authentication. Operator-supplied key fob releases RSA private key from flash memory only when authenticated with fingerprint. RSA key is then released to complete the authentication.

Table 3

1.3 Role-Based Authorization by the Operating System

Assuming Roles

This section elaborates roles for the *SourceT* limited operational environment, not for the Cryptographic Module. Each human operator will have one or more available roles. However, once an operator has logged into the host platform, only one role at a time may be active for that operator. An operator must explicitly enter and exit a role to enable and disable, respectively, particular access rights. The "**enable <role>**" command is used to enter a role, and the "**exit**" command is used to exit a role. For example, an operator would enter the command "**enable securityadmin**" to be allowed access to the directories and commands associated with a *securityadmin* Crypto officer.

Notes on Specific SourceT Roles

There is **no** "super-user" role available in the *Secure64 SourceT* micro operating system. While an operator could conceivably be assigned all available roles, it is NOT a recommended practice.

There is **no** maintenance role, in the sense defined by the FIPS 140-2 specification, section 4.3.1. The firmware-only implementation of the Cryptographic Module does not require any handling that would necessitate including a *SourceT* maintenance role.

The *SourceT* role permitting execution of the command that authenticates itself to the

Cryptographic Module as the *securityadmin* Crypto officer identity is the "***securityadmin***" role.

1.4 Identity-Based Authentication to the Crypto Module

This section elaborates authentication to the Cryptographic Module. As described earlier, having logged in and enabled an appropriate *SourceT* role, an operator may now issue a command or run an application that invokes one or more of the Secure64 Cryptographic Module security functions. Each application or command will supply its identity (which application or command) to the Cryptographic Module, using its caller's credential, to establish its access rights for API calls. Similarly, the *SourceT* Crypto officer functions and *securityadmin* Crypto officer command will supply their identities (*SourceT* Crypto officer, *securityadmin* Crypto officer) using their respective caller's credentials, to establish their access rights for SPI calls.

As described in section 1.1, an application's identity and role is authenticated within the Cryptographic Module on each API call, by XOR'ing the caller's credential with the corresponding validation credential mask, then ensuring that the result is equal to the built-in validation credential value. The same identity authentication process is used for *SourceT* Crypto officer and *securityadmin* Crypto officer caller's credentials in SPI calls.

In effect, each credential mask is a one-time pad, that is used only for one specific message in one specific system. Only a single value in the system is encrypted by this one-time pad, namely, the 64-bit value word of the validation credential built into the Cryptographic Module. Because each Credential within the Cryptographic Module has its unique corresponding mask, no one outside of the Cryptographic Module ever sees two values encrypted by the same one-time pad. From this point of view, the application caller's credential supplied in an API call is its Cryptographic Module validation credential encrypted by a one-time pad.

A one-time pad used in this manner is acknowledged to have an encryption strength of 56 bits. Additionally, the credential values and masks are generated uniquely for each system image, using the NIST certified PRNG algorithm (certificate #507), so the values of credentials in one system provide no information about the values in other systems. The cryptographic strength of this approach is shown in Table 4.

Identity	Role	Caller's Credential	Authentication Strength
SourceT Crypto officer	SYS CRYPTO OFFICER	Accessible only to the *SourceT* system.	For all cases: if we assume a > 2-core guess rate of 4 x 10^9 guesses/sec:
securityadmin Crypto officer	ADM CRYPTO OFFICER	Accessible only to get status command.	P(correct guess) < 1 in 7.2 x 10^{16} << 1 in 10^6
Application User	USER	Accessible only to proper application.	P(correct guess in 1 minute) < 1 in 300,239 < 1 in 10^5

Table 4

The probabilities in Table 4 were calculated in the following manner. A correct caller's credential guess requires guessing a seven-bit application ID, a one-bit API/SPI flag, and a 56-bit random number. To be very conservative, we tabulated only the probabilities of guessing the 56 random bits. The probability of a correct guess of a 56-bit random number would be 1 in 2^{56} = 1 in 7.2057594 x 10^{16}, which by itself is significantly less than 1 in 10^6. If it were possible to sustain 4 x 10^9 guesses per second, the probability of a correct guess in one minute would be (4 x 10^9 x 60) / 2^{56}, or 1 in 300,239, which is less than 1 in 10^5. These calculated probabilities are upper bounds. The actual probabilities are far smaller – the other eight bits also need to be guessed, the assumption of a guess rate of four billion per second is well above that achievable on the HP platforms supporting the Cryptographic Module, and additional cycles would be required to test each guess to see if it is correct.

Application User authentication also establishes access to a corresponding 256-bit AES AMK (Application Master Key, one key per application) also located within the protected Cryptographic Module memory compartment. While an application is active, all keys generated by that application may be used only by that application. If the application needs to save a key it generated, the key will be wrapped by its corresponding AMK, using the AES key wrap specification, and exported from the Cryptographic Module for external storage.

If two applications need to share a cryptographic key, or if one application needs to generate a key to be used by another application, API calls have been provided to meet these needs. An application may wrap a key it has generated specifically for another application and export it from the Cryptographic Module. The other application then may import the wrapped key back into the Cryptographic Module for its own use. An application also may import a copy of a wrapped key already in its possession to be re-wrapped for a different application and exported from the Cryptographic Module. More details follow in the next Section.

2. Access Control Policy

2.1 Commands and Applications: Identity and Roles

At a programmatic level, as explained earlier, the Cryptographic Module API (Application Program Interface) and SPI (System Program Interface) require the operating system, commands, and applications to present authorization credentials. These “caller's” credentials convey the caller's identity to the Cryptographic Module. This mechanism enables the Cryptographic Module to prevent each caller from performing restricted operations, and applications from utilizing cryptographic keys that belong to some other application. An application's use of its corresponding AMK ensures that all keys wrapped and stored externally can subsequently be imported into the Cryptographic Module only for use by the same application.

Table 5 on the next page summarizes the services available via API and SPI calls for the Cryptographic Module security roles. The following legend of abbreviations is for this table:

1. “CM” = Cryptographic Module
2. “ACR-OFF'R” = ADM CRYPTO OFFICER
3. “SCR-OFF'R” = SYS CRYPTO OFFICER
4. “CR” = Caller's Credential,
5. “SH” = Session Handle
6. “CS” = both Credential and Session Handle
7. “KH” = Key Handle
8. “PMK” = Platform Master Key
9. “AMK” = Application Master Key
10. “WK” = Wrapped Key
11. “wrap'd” = wrapped
12. “PK” = Public Key
13. “DS” = Digital Signature
14. “DV” = Digest Value
15. “ED” = Encrypted Data
16. “DD” = Decrypted Data
17. “RD” = Random Data
18. “Alg” = Algorithm
19. “args” = The remaining arguments to a Cryptographic Module API or SPI call (normally specifying algorithms, key lengths, pointers to keys or data, lengths of data buffers. Etc.).

Services	Role	Service Inputs	Service Outputs	Description
Open session	USER	CR, API args	Session handle	Open session with CM
Close session	USER	CS, API args	OK/Error	Close session with CM
Key generation	USER	CS, API args	KH/Error	Crypto key of spec'd type
Delete key	USER	CS, API args, KH	OK/Error	Delete application key
Encryption	USER	CS, API args, KH	ED/Error	Spec'd key length & Alg
Decryption	USER	CS, API args, KH	DD/Error	Spec'd key length & Alg
Random # gen	USER	CS, API args	RD/Error	ANSI X.9.31 PRNG data
Hash Digest	USER	CS, API args	DV/Error	SHA-1 or SHA-256 digest
HMAC	USER	CS, API args, KH	DV/Error	HMAC digest
Digital Signing	USER	CS, API args, KH	DS/Error	Spec'd digital signature
Signature Verify	USER	CS, API args, KH	Verified/Error	Verification result
Wrap key	USER	CS, API args, KH	WK/Error	Wraps Application key
Rewrap key	USER	CS, API args, WK	WK/Error	Wrap for other appl
Wrap app key	USER	CS, API args, KH	WK/Error	Wrap 4 other application
Unwrap key	USER	CS, API args, WK	OK/Error	Unwraps Application key
Import Public key	USER	CS, API args, PK	OK/Error	Moves PK into CM
Export Public key	USER	CS, API args, KH	PK/Error	Moves PK from CM
Query CM Status	ACR-OFF'R	CR	CM state	Current CM state
Initialize CM	SCR-OFF'R	CR, SPI args	OK/Shutdown	Init + power-on self-test
PL/2 Shutdown CM	SCR-OFF'R	CR	OK/Error	Zeroize & set failed state
Wrap PMK	SCR-OFF'R	CR, SPI args	w'd PMK/Error	Gen & wrap new PMK
Unwrap PMK	SCR-OFF'R	CR, wrap'd PMK	PMK OK/Error	PMK is 256-bit AES key
Wrap AMK	SCR-OFF'R	CR, SPI args	w'd AMK/Error	Gen & wrap new AMK
Unwrap AMK	SCR-OFF'R	CR, wrap'd AMK	AMK installed	
PL/1 Shutdown CM	SCR-OFF'R	CR	OK/Error	Zeroize & set failed state

Table 5

Note 1: Shading denotes services invoked by an SPI call. No shading denotes services invoked by an API call.
Note 2: The USER role is reserved for API calls from embedded commands and applications within a *SourceT*-based product. All API calls are executed only by embedded commands or applications.
Note 3: The *securityadmin* Crypto officer has only one command that calls an SPI service in Table 5: the *SourceT* **status** command calls the "Query CM Status" service.
Note 4: The *SourceT* system issues all other SPI calls for services requiring the SYS CRYPTO OFFICER role. This is to automate tasks (see Sec 2.3) that otherwise would be done manually.
Note 5: The PL/2 shutdown SPI call is issued by *SourceT* components running at PL/2 in an application processor core. The PL/1 shutdown command is issued by physical-monitoring software executing at PL/1 on a separate I/O processor core. Shutdown logic for the two calls differs.

2.2 Security Functions

The Secure64 Cryptographic Module provides confidentiality, integrity, and message digest services. As shown in the table below, the module supports the following NIST approved algorithms: AES, Triple-DES, RSA, DSA, SHA-1, SHA-256, HMAC-SHA-1, and PRNG. Appendix A lists the details of the NIST certificates for approved algorithms.

No non-approved algorithms are supported by the API calls; consequently the Cryptographic Module always operates in FIPS 140-2 approved mode.

Secure64's products boot from a single monolithic load image, encrypted and digitally signed at the Secure64 facility. This image contains:

1. The *SourceT* micro operating system.
2. Cryptographic Module components, with the Module's own digital signature for internal Cryptographic Module integrity validation.
3. A single 256-bit AES key (the KEK) CSP to be placed into the installed Cryptographic Module.
4. Validation credential CSPs (one for each authenticatable identity) to be placed into the installed Cryptographic Module.
5. Caller's credentials (one for each identity that invokes the Cryptographic Module) embedded within *SourceT* and accessible only by its proper caller - *SourceT* Crypto officer, *securityadmin* Crypto officer, or specific Application User.
6. All the statically linked applications and commands for the Secure64 product bundle.

The KEK and credentials are generated using a NIST certified PRNG algorithm (certificate #507). Each KEK is generated uniquely for all systems acquired by a single customer, to permit the customer to migrate master keys among his systems for backup, restore, and key migration purposes. Validation credentials are generated uniquely for each system. Corresponding caller's credentials are computed from the validation credentials. The internal signing of each Cryptographic Module, and the encryption and signing of each full load image also utilize NIST certified algorithms – RSA (certificates #426, #874, #507), and AES (certificate #956). The 256-bit AES encryption keys, for the system image for each target hardware platform, are generated by the NIST certified PRNG algorithm (certificate #507). The Secure64 signing key is generated by NIST certified algorithms (certificates #426, #507).

These measures ensure that only an authenticated image of the operational environment gets loaded onto a target hardware platform, and that no other firmware or software has been injected into the system. Once loaded properly – i.e., the image decrypts properly, the full load image digital signature is verified, and the Cryptographic Module is installed – the operating system will then invoke initialization routines for all of its subsystems, including the Cryptographic Module. The Cryptographic Module initialization routines will properly set all initial values, allocate and

zero initial dynamic memory, execute the random number and algorithmic power-on self-tests, and validate the internal Cryptographic Module digital signature – assuring its internal firmware and embedded CSP integrity. This internal validation of the Cryptographic module covers only the Cryptographic Module executable code and the initialized data within the Cryptographic Module memory compartment (components 2, 3, and 4, listed on page 20). Once all self-tests complete successfully, the Cryptographic Module may be called by *SourceT* to generate or install the platform and application master keys, and set itself to the READY state to accept command or application API calls in FIPS approved mode.

As sanctioned by the FIPS 140-2 specifications, a single non-approved security function is also present within the Cryptographic Module. This function is only used internally and cannot be accessed outside of the Cryptographic Module boundary. This sole function is a non-approved high speed deterministic pseudo random number generator (PRNG). This fast PRNG is used only for generating initialization vectors (IV's) and for seeding the NIST-approved deterministic PRNG (certificate #507). The fast PRNG also is required to pass both a known answer power-on self-test, and the same continuous conditional self-test required of the NIST-approved PRNG.

The Secure64 Cryptographic Module performs random number generation using an approved pseudo-random number generator as specified in “NIST Recommended Random Number Generator Based on ANSI X9.31, Appendix A.2.4, using the 3-Key Triple-DES and AES Algorithms”. The certified PRNG (certificate #507) must pass a known answer test during power-on self test. An initial seed is then obtained from the fast PRNG, which, in turn, was seeded by non-deterministic random data from the TPM, mixed with data from a ~1 GHz Itanium timer register, after completing its own known answer power-on self-test.

Several key sizes for RSA and DSA algorithms are listed in Table 6 as not recommended by NIST. The NIST Cryptographic Module Validation Program (CMVP) allows the use of NIST non-recommended key sizes as long as at least one recommended key size is supported and the documentation lists these non-recommended key sizes. The Cryptographic Module complies with NIST SP800-57 which specifies that the non-recommended DSA and RSA key sizes shall not be used to generate new digital signatures.

Secure64 Cryptographic Module Security Services • All algorithms are FIPS approved algorithms • Key-sizes include both NIST-recommended and non-recommended sizes				
Algorithm	**Key Sizes (NIST-recommended)**	**Key Sizes (allowed, but not NIST-recommended)**	**Modes**	**NIST Algorithm Certificates**
AES	128 / 192 / 256		ECB CBC	#882, #956
Triple-DES	2-Key / 3-Key		ECB CBC	#722
PRNG	AES 128 / 192 /256		ANSI X9.31	#507
SHA-1			BYTE	#874
SHA-256			BYTE	#936
HMAC			SHA-1	#580
DSA	1024			#350
RSA	1024 / 1536 / 2048 P-Exp: 3, 7, 65537	1152 / 1280 / 1408 / 1664 / 1792 / 1920 P-Exp: 17, 35, 37		#495
Algorithms used in the system image build and load processes				
RSA	2048, for SIG(ver)			#426
SHA-1	for load image digest		BYTE	#874
AES	128 / 192 / 256		CBC	#956

Table 6

2.3 Cryptographic Keys and Critical Security Parameters

As stated earlier, the Secure64 Cryptographic Module is functionally a firmware implementation of a Hardware Security Module (HSM), except that interfaces to the device are a firmware API and SPI rather than physical ports. Commands, data, and encrypted keys are sent to the Module via API calls from application programs. Outcomes are returned to the application. Cryptographic Module control directives are sent to the Module via SPI calls from the system, and status is returned to the system. All transfers of information to/from the Cryptographic Module are completely controlled by the Module firmware.

The primary key management policy for the Secure64 Cryptographic Module is that Cryptographic

Keys[11] and Critical Security Parameters are NEVER in the clear or accessible by any command, application, or operator outside of the Cryptographic Module. This security policy applies to all system components and applications. Cryptographic keys and critical security parameters residing outside of the Cryptographic Module memory compartment, either in RAM or on disk, always are encrypted. NO plaintext keys ever enter or leave the Cryptographic Module.[12]

Key Management

1. The Key Encryption Key (KEK) is a 256-bit AES key used only to wrap[13] the PMK to be exported for backup or migration, or when importing and unwrapping an externally saved PMK back into the Cryptographic Module. All master key wrapping is done using the AES key wrapping specification. The KEK key is placed into the Cryptographic Module at the boot-time Module installation. It is a unique key that is shared among all the systems for a single customer. If the Cryptographic Module is zeroized, only a system reboot and Cryptographic Module reinstall will restore the KEK. To fully zeroize the KEK the monolithic load image must be erased from the disk drive and the disk reformatted.
2. The Platform Master Key (PMK) is a 256-bit AES key used only to wrap an AMK to be exported for backup or migration, or when importing and unwrapping an externally saved AMK back into the Cryptographic Module. The PMK key is generated within the Cryptographic Module at initial system boot, or at any subsequent boot when the system file containing the system's encrypted master keys is absent. Once generated, *SourceT* saves the wrapped PMK in the system file of encrypted master keys. It is imported and unwrapped from the system master keys file for subsequent system boots.
3. The Application Master Keys (AMKs), are 256-bit AES keys, one for each turnkey application built into the *Secure64 SourceT* load image. AMKs are only used to wrap application generated keys to be exported for backup or migration, or when importing and unwrapping externally saved application keys into the Cryptographic Module. The AMKs are generated within the Cryptographic Module at initial system boot or at any subsequent boot when the system file containing the system's encrypted master keys is absent. They are imported and unwrapped from the system master keys file for subsequent system boots.
4. Application keys (AKs) are additional symmetric or asymmetric keys for particular applications. They may be generated and exist within the Cryptographic Module. They too may be wrapped and exported for external storage. This is done by the wrap API call. Symmetric and asymmetric application keys always will be wrapped by using their corresponding AMKs. The public portions of asymmetric AKs may be freely moved into and out of the Cryptographic Module.
5. PMK' is defined to be the PMK wrapped by the KEK. PMK' is generated within the

[11] Except for the public portion of an application key which is an asymmetric key pair.

[12] Except for the public portion of an application key which is an asymmetric key pair.

[13] Wrapping a key is done inside the Cryptographic Module. An encrypted copy of the key is returned. The plaintext key remains in the Module.

Cryptographic Module, wrapped by the KEK using the AES key wrap specification, and exported - by the wrap PMK SPI call.

6. AMK' is defined to be an AMK wrapped by the PMK. Each AMK' is generated within the Cryptographic Module, wrapped by the PMK using the AES key wrap specification, and exported - by the wrap AMK SPI call.
7. AK' s are defined to be AKs wrapped by their corresponding AMKs. AK' s are generated within the Cryptographic Module, wrapped by their AMKs using the AES key wrap specification, and exported – by the wrap Key API call.

Platform master keys always are generated within the Cryptographic Module, and never appear outside of a Cryptographic Module in the clear, even to a *securityadmin* Crypto officer. After installation CSPs and cryptographic keys NEVER enter or leave the cryptographic memory module in plaintext, with the one exception of the public portion of an AK which is an asymmetric key pair.

Figure 2 shows the Secure64 Cryptographic Module key hierarchy. Table 7 on the following page elaborates the Cryptographic Module CSPs.

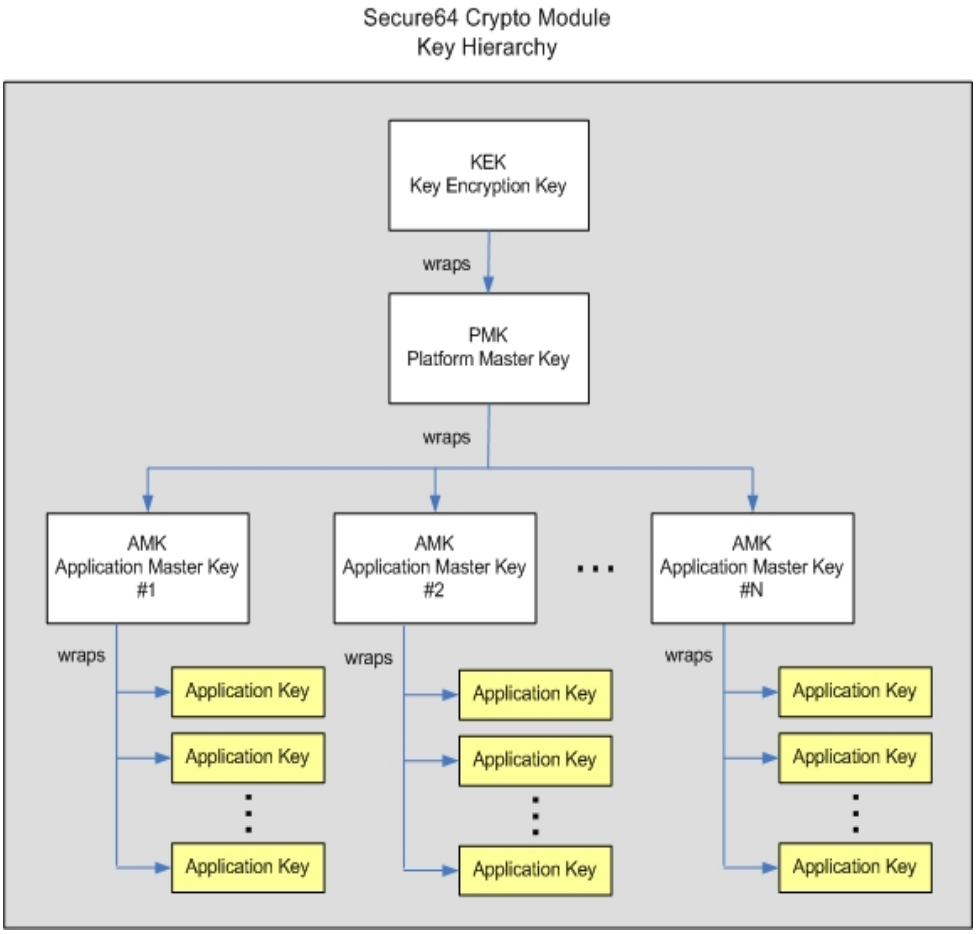


Figure 2

Definition of Critical Security Parameters (CSPs):

CSP	Description	Creation	Storage[14]	Entry[15] / Output	Zeroization
Key Encryption Key (KEK)[16]	Used only for the safe import and export of the PMK	Secure process at Secure64 facility using an approved random number generator	Embedded in the encrypted load image, Plaintext in the Crypto-Module MC[17]	Placed into Crypto-Module MC after installing Crypto-Module	Zeroized before any Cryptographic Module shutdown
Platform Master Key (PMK)	Used only for the safe import and export of AMKs	256-bit AES key created at initial boot using approved algorithms.	Stored as PMK' as described earlier, plaintext in Crypto MC	SPIs: Wrap + export; import + unwrap, using KEK.	Zeroized by Cryptographic Module shutdown
Application Master Keys (AMKs) One AMK per application built into *Secure64* product	Used only for the safe import and export of application keys	256-bit AES keys created at initial boot of the system; using approved algorithms.	Stored on disk as AMK's after being wrapped using the PMK.	SPIs: Wrap + export; import + unwrap, using PMK.	Zeroized by Cryptographic Module shutdown
Application Keys (AKs) AES keys 3-DES keys RSA key pairs DSA key pairs HMAC keys	Keys used to protect application specific data	Generated in CM by API calls from applications	Stored on disk as AK' s after wrap by AMKs	APIs: Wrap + export; import + unwrap	Zeroized by Cryptographic Module shutdown
Approved PRNG Seed, Seed Key	Seed values for approved PRNG	Generated in CM to seed the PRNG	Stored within Crypto MC	Never entered or exported	Zeroized by CM shutdown
Validation Credentials	Authenticate SPI & API calls	56-bit random data + 56-bit random mask	Embedded in the encrypted load image.	Placed into installed Crypto-Module MC	Zeroized by Cryptographic Module shutdown

Table 7

[14] Always plaintext within the Cryptographic Module; Encrypted outside of the Cryptographic Module.

[15] Encrypted master keys are always loaded from the system master key file. Encrypted application keys are loaded from application files.

[16] The entire key tree uses 256-bit AES keys except for AKs, which may be for any of the algorithms or key sizes in Table 6 on page 22.

[17] MC = memory compartment.

On Initial Use of the Host Platform

The first time a product based on the *Secure64 SourceT* micro operating system is downloaded to the disk of a customer hardware platform, an encrypted and digitally signed monolithic system load image must be prepared for that specific hardware platform. This image is built only at the Secure64 facility. It must contain all needed initial CSPs and, although not required by FIPS 140-2, be encrypted specifically for the target machine. The latter requires using the plaintext unique identifier read from the TPM on the target platform. All of the steps of this build process: securely reading information from the target platform, building the system image at the Secure64 facility, securely transmitting the system image back to the target platform, and loading it onto the target platform are performed without utilizing the Cryptographic Module in any manner. However, NIST certified algorithms are used outside of the Cryptographic Module wherever possible.

The product installation system will first execute a pre-install utility on the target host platform. This pre-install program, over an SSL session with the Secure64 facility, will:[18]

1. Generate random data using a NIST certified algorithm (certificate #507). This data is hashed by SHA-1, and the result is used as TPM authorization data.
2. Obtain the unique public EK from the platform's TPM chip.
3. Take “ownership” of the TPM using the generated authorization data. In this process two parts of the hashed random data from step 1 become the authorization data for the TPM owner, and for authorizing use of the TPM SRK.
4. Communicate the TPM authorization data, public EK, and public SRK back to the Secure64 facility. The public SRK is used to wrap for the TPM a generated BBK (certificates #426, #507) that, in turn, encrypts the AES key and IV used to encrypt the load image. At each boot, the wrapped BBK is unwrapped by the TPM SRK, and the BBK itself is then used to decrypt the AES key and IV needed to decrypt the full load image.

A product image, including an encrypted system load image, is then prepared at the Secure64 facility and transmitted back to the disk of the target host platform.

Part of the encrypted load image – specifically, for placement into each newly installed Cryptographic Module memory compartment – is the Key Encryption Key (KEK), a 256-bit AES key that is never wrapped and exported from the Cryptographic Module. Each KEK is securely created at the Secure64 facility, as described in Section 2.2, for a particular customer. Immediately after being generated, it is encrypted by the public Secure64 RSA signing key. Thereafter it is saved in a data base only in this encrypted form – a form unintelligible to all Secure64 personnel.

When a system image is being produced, at an intermediate stage of the process, the KEK is decrypted by the Secure64 RSA private signing key, and immediately re-encrypted as part of the AES CBC encryption and signing of the entire system load image. As described earlier, the secure64 signing key was generated by NIST certified algorithms (certificates #426, #507), and the

[18] Please refer to the TPM section in the Introduction.

256-bit AES image encrypting key was generated by the NIST certified PRNG (certificate #507). Thus, the KEK is never exposed in plaintext within the Secure64 facility, and never leaves the Secure64 facility until it has been encrypted for a specific Itanium-based server. Its integrity is again validated by the power-on start-up integrity self test of the Cryptographic Module. The KEK is the root of the key tree within the Cryptographic Module, and is critical to the process of generating, backing up, migrating, and restoring Platform Master Keys (PMKs) among a customer's systems.

There is a unique different PMK for each Secure64 Cryptographic Module. After the system boot installs the Cryptographic Module, on first use of the Module, the PMK is generated within the Cryptographic Module as a 256-bit AES key, generated with a FIPS certified random number generator (Certificate #507). This is then wrapped by the KEK, using the AES key wrap specification, which produces PMK'. PMK' is then exported from the Cryptographic Module, and stored safely in a previously non-existent system master key file. PMK' is reimported into the Cryptographic Module and unwrapped whenever the system is subsequently restarted and the system master key file can be found.

Generated AMKs are handled in a similar fashion. They are wrapped by the PMK, using the AES key wrap specification, producing AMK's. AMK' keys are then exported from the Cryptographic Module and appended to the previously non-existent system master key file. On future system boots, this file will be found and will supply the wrapped forms of application master keys which can be reimported into the Cryptographic Module and unwrapped.

On Restart After Initialization

When the *Secure64 SourceT* limited operational environment is restarted, it will check the file system for the existence of the master key file containing the PMK' and the AMK's. This file will be stored in a portion of the file system that is accessible only to the *SourceT* system. At this point PMK' can be imported back into the Cryptographic Module and unwrapped using the KEK.

Once the PMK has been restored to the Cryptographic Module, the AMK's also can be imported back into the module, and then unwrapped using the PMK. This is done using the AES key wrap specification, and restores the individual AMKs.

At this point, the Cryptographic Module is ready to import and unwrap AK' s. Application files containing AK' s can be read by applications, and the encrypted AKs can be imported back into the Cryptographic Module and unwrapped, using unwrap key API calls.

Note that this logic enables the *securityadmin* Crypto officer to roll over master keys at the next system boot by simply deleting the system master key file. This may be required as a matter of security policy or when redeploying a *SourceT* system. Deleting the system master key file is a *SourceT* system function that takes place entirely outside of the Cryptographic Module.

Key Migration and Disaster Recovery

Key migration among systems belonging to the same customer is an essential part of any disaster

recovery plan. Wrapped master keys are encrypted by either the 256-bit AES KEK, or the 256-bit AES PMK. Wrapped application keys are encrypted by a 256-bit AES AMK. These encrypted keys may safely be stored or backed up, locally or remotely, in this encrypted form by various system functions outside of the Cryptographic Module. They subsequently can be reimported and unwrapped into the Cryptographic Modules of the originating system or of any system sharing a KEK with the originating system.

It also is possible, outside of the Cryptographic Module, to apply one or more additional wrapping layers of protection to encrypted keys that can restrict the systems to which they can be migrated, or that require pass phrases or other additional authenticating means when removing the outer wrapping layer(s) of protection. Means utilizing the unique TPM identifier of a second hardware platform, for example, can be employed to wrap the keys so that they can be migrated only to that specific second platform.

In all cases, the encrypted keys can be migrated to another system, and imported into that system's Cryptographic Module, if and only if that other machine shares the KEK of the originating system. Note that the migrated AMKs will be usable only for the same applications as they were on the originating system.

Application Usage of the Cryptographic Module

Secure64 turnkey applications, statically linked with the *SourceT* micro operating system, provide an example of using the Secure64 Cryptographic Module. The DNS Signer application, which implements DNSSEC, can require the use of thousands of zone signing keys. DNS Signer also needs to keep a record of each zone signing key used, and have these keys on hand for varying periods of time. As a new zone signing key is needed, DNS Signer will invoke another application that has been using the the Cryptographic Module, at a lower usage level, to create a supply of keys in advance of the times they are needed. These pre-generated keys will have been wrapped using the DNS Signer's AMK, so that they immediately can be imported into the Cryptographic Module and unwrapped for use by the DNS Signer.

2.4 Security Audited Events

An audit system is present within *SourceT* which logs Cryptographic Module events and related *SourceT* events that occur outside of the Cryptographic Module. Unencrypted log messages are written circularly to a read-only disk file. The maximum size of this file is settable by the *SourceT sysadmin.* This log file may be read, but never modified, by authenticated *SourceT* operators. In addition, if the customer elects, the log file also will be copied over a network link to a site configured by the customer. Protection of the network link and access to the log at the receiving site are the responsibilities of the customer. Receiving sites are expected to range from a simple facility management system, to a more complex management system such as HP's Open View. Reading the log from the read-only disk file assures that the log information has not been altered in any way.

None of the information written to the *SourceT* event log contains information that could be used to compromise the security of the Cryptographic Module. All log messages are constant text strings, describing only which particular event occurred. No log message contains specific information about an API or SPI caller's identity, credentials, handles, or requested cryptographic service. Power-up and conditional self-test log messages, however, do identify the failing cryptographic function.

The types of logged Cryptographic Module events are:

- Cryptographic Module begins operation in FIPS approved mode.
- Cryptographic Module shutdown, no longer running in FIPS approved mode.
- Invalid caller's credentials or handles.
- Invalid attempts to open or close a Cryptographic Module session.
- Invalid inputs for USER , ADM CRYPTO OFFICER , or SYS CRYPTO OFFICER role services.
- Power-on self-test passed, or failed.
- Conditional self-test failures.

Related *SourceT* logged events are:

- Valid and invalid attempts to enable the *SourceT securityadmin* role.
- Use of *SourceT* encrypted key migration, backup, or restore functions.
- Use of *SourceT* application-specific *securityadmin* Crypto officer functions.
- Addition/deletion of *SourceT* users and roles, including the *SourceT securityadmin* role.
- Detection of open hardware case before starting the Cryptographic Module.
- Detection of open hardware case after starting the Cryptographic Module.
- Normal and Abnormal system shutdowns.

A second log is maintained internally to the management processor in the host hardware platform (The HP iLO 2 – Integrated Lights Out – processor). The HP iLO 2 Operations Guide[19] describes a range of means for providing secure access to this log. It is recommended that access to the iLO 2 processor, and its log be restricted to *SourceT sysadmins* and *SourceT securityadmins*. The log includes entries for hardware platform events such as system power- on/power-off, system reboots, power and temperature warnings, cooling fan failures, and the hardware case being opened while the system is running.

2.5 Self Tests

2.5.1 Startup Tests

The Secure64 Cryptographic Module executes the following self-tests during startup and

[19] HP Integrity and HP 9000: iLO MP Operations Guide, HP Part Number '5991-6006', Publication Date 'January 2008'

initialization:

- Integrity Test
- Cryptographic Algorithm Test
- Random Number Generator Test

Successful completion of these tests will place the Cryptographic Module into the INITIALIZE state. Once *SourceT* has made the SPI calls to initialize the platform and application master keys, the Module will transition to the READY state, and applications may then initiate sessions. A failure in any of these start-up tests will result in the module being zeroized, entering the FAILED state, and returning a failed status to the system. Any subsequent attempts to use the Cryptographic Module (SPI or API calls) would then return an S64CM_SFAILED return code. Appropriate log messages are generated as part of this testing.

2.5.2 Integrity Test

During each and every boot process both the entire system load image, including all Cryptographic Module components, and the executable code and embedded data of the Cryptographic Module itself are authenticated by digital signatures to ensure their integrity.

The system image is a single monolithic image. During the build process, the system binary image was signed using NIST certified algorithms (certificates #426, #874), then encrypted with an AES key and IV generated by a NIST certified algorithm (certificate #507). Once decrypted, the digital signature also is validated by NIST certified algorithms (certificates #426 and #874). Using the same algorithms, the Cryptographic Module itself also is separately signed.

The system boot loader will validate the digital signature for the entire system load image each time the system is booted. After the Cryptographic Module is installed and started, its self-tests will also independently validate the internal digital signature covering only the executable code and embedded data components of the Module itself. This digital signature validation is done as a start-up self-test using certified algorithms (certificates #495, #874) to internally assure its integrity.[20] If this test fails, the Cryptographic Module will zeroize its contents, shut itself down, and a log message will be written to SYSLOG. The *SourceT* system also will be notified.

The following details are not required by FIPS 140-2. They describe measures wholly outside of the Cryptographic Module that Secure64 has adopted to ensure greater protection of the confidentiality and unique hardware platform destination for the product system load image. This explains more fully the boot use of the TPM.

At boot time, the system self-tests the TPM, reads the TPM public EK, and uses the TPM SRK and the BBK key to recover the AES IV and encryption key for the encrypted system image. This is done in the following manner:

1. In addition to the encrypted and signed system load image, the downloaded product also

[20] This requires a system function, callable only from the Cryptographic Module, to briefly permit reading the Cryptographic Module executable code. Once the digital signature hash been computed, the memory protection for the executable code is reset to "execute only".

contains two data structures that enable the recovery of the AES IV and encryption key for the system load image.

2. The first of these data structures, called the SDB (Secret Data Blob), contains the encrypted AES key and IV used to encrypt the signed system load image. The SDB is encrypted with the public portion of the RSA PKCS#1 v1.5 BBK generated at the Secure64 facility for initial product installation.
3. A second data structure is used to recover the wrapped BBK and the TPM owner and SRK credentials. This structure is a large encrypted table called the “crypto pad”, and permits these data to be recovered using the TPM public EK. This is done in the following steps:
    1. The TPM public EK is hashed to a 20 Byte SHA-1 digest.
    2. This digest is input to the multiple hashing algorithm defined in RFC 2898. This repeatedly hashes the input digest N times, where N is a function of the input digest value itself.
    3. The high order 16 bytes of the N-1'st hash are used as a 128-bit IV; the entire N-1'st hash is appended, as four 4-byte words in reverse order, as a MAC for the circular AES/CBC encryption and decryption of the crypto pad.
    4. The high order 16 bytes of the Nth hash are taken as a 128-bit AES key; the low order four bytes are used to derive an odd-valued stride and and a starting index into the crypto pad table. The crypto pad must be a power of two in size (presently 1024 x 16 bytes), and at least twice as large as the embedded data.
    5. The entire table is AES/CBC circularly decrypted beginning at the starting index. Matching the last decrypted value to the reversed N-1'st hash is used to validate the decryption. After decryption of the table, beginning at the starting index, the TPM authorization data and wrapped BBK can be recovered – from 4-byte words located a stride apart from each other in the table.
    6. The wrapped BBK is loaded into the TPM, using the SRK credentials. Within the TPM, the wrapped BBK is unwrapped and used to decrypt the SDB. This recovers the AES decryption key and IV for the signed system load image.

Note that this process is self checking. If the decryption of the crypto pad fails, it is detected by the final MAC validity check after decrypting the entire table. If the SRK credentials are bad, the TPM will refuse to unwrap the BBK. If the SDB is modified the TPM will refuse to decrypt it. If the BBK value is incorrect, the digital signature of the decrypted system image will fail. An error at any step will halt the boot process.

The loader then decrypts the system load image and validates the digital signature for the decrypted image, using the NIST certified algorithms (certificates #426, #874, #956). If the signature authenticates, the integrity of the system load image has been strongly validated, and the system proceeds with installing, initializing, power-up self-testing, importing master keys, and placing the Cryptographic Module in full operation in FIPS approved mode. If the system load

image signature validation fails, the load terminates, a log message is sent to SYSLOG, and the system boot is halted. If any subsequent Cryptographic Module self-test fails, a log message is sent to SYSLOG, and the Cryptographic Module is zeroized and shut down.

2.5.3 Cryptographic Algorithm Tests

Both pairwise consistency tests and known answer tests (KATs) are run as part of the cryptographic algorithm tests. RSA and DSA are both tested using pairwise consistency tests. The remainder of the algorithms are tested using known answer tests. Each mode is tested as part of the cryptographic algorithm tests, but only one key size is used for each FIPS approved algorithm.

The following cryptographic algorithm tests are performed during power-on testing:

- Triple-DES 3 key ECB mode, encrypt/decrypt KAT
- AES, 128 bit keys for ECB mode, encrypt/decrypt KAT
- SHA-1 and SHA-256 KAT
- PRNG X.9.31 using AES 128 bit key KAT
- HMAC SHA-1 KAT
- RSA 1024 bit key pairwise consistency test
- DSA 1024 bit key pairwise consistency test

The cryptographic algorithm tests are automatically run at system start-up. In order to initiate the power-on tests, an operator must boot, reboot, reset, or cycle the power on the system, as sanctioned by the FIPS 140-2 specification.

2.5.4 Random Number Generator Test

The FIPS approved random number generator will perform a known answer test (KAT) using a known seed value. The following test is performed:

- PRNG X.9.31 using AES 128 bit key KAT

The fast PRNG will perform a known answer test using a known seed value.

After completing the known answer tests, the PRNGs are reinitialized in the following manner:

1. Random data, read from the non-deterministic built-in RNG in the TPM, was passed in by *SourceT* when its SPI initialization call was made to the Cryptographic Module.
2. That data is then mixed with the hardware high-frequency interval timer, circularly shifted based upon its own value, and used to seed the proprietary deterministic fast PRNG.
3. The output of the fast PRNG is used to seed the FIPS approved deterministic PRNG.

In accordance with FIPS requirements, the output from the initialization of the FIPS approved PRNG is then stored for use in the continuous random number generator test. Similarly, the

initial output from the fast deterministic PRNG is saved for similar testing.

2.5.5 Conditional Tests

The following conditional tests are performed whenever the Secure64 Cryptographic Module is executing the associated function:

- Continuous deterministic FIPS approved PRNG conditional test
- Continuous deterministic fast PRNG conditional test
- RSA pairwise consistency test for generated keys
- DSA pairwise consistency test for generated keys

The continuous random number generator test is performed on both of the two deterministic random number generators that are contained within the boundary of the Cryptographic Module. The PRNGs generate sixteen and eight bytes of random data, respectively, each time they are called. If two successive calls to a specific random number generator produce identical results, then the conditional test has failed and an error has occurred. To detect this error the output of each random number generator is saved in units of sixteen bytes for the NIST approved deterministic PRNG (certificate #507), and in units of eight bytes for the fast internal deterministic PRNG. Each subsequent generation of random bytes is compared with the preceding generated bytes, and then saved for the next comparison.

Asymmetric keys generated for use in digital signatures are immediately tested for pairwise consistency. A test value is first signed and then verified using each newly generated key. Should the result of these operations not equal the original test value, the test has failed.

Whenever any of these start-up or conditional tests fail, the Cryptographic Module takes these actions:

1. The Module puts itself into the FAILED state, precluding any further use.
2. Zeros are written over all CSPs and all memory allocated from the system.
3. The Cryptographic Module memory compartment is closed.
4. A SYSLOG entry is written describing the failure.
5. The *Secure64 SourceT* operating system is notified that the module has shutdown.

At this point, the Cryptographic Module is no longer usable. Any subsequent attempts to access the Cryptographic Module security functions will return a S64CM_SFAILED return code. Once notified of the shutdown, the operating system must then decide how to react to the module's failure, and take appropriate action. For this version of the Secure64 Cryptographic Module, the only remedial action possible is to reboot the host platform. This will result in reinstalling, reinitializing, and re-power-up-self-testing the Cryptographic Module.

3. Physical Security Policy

As stated in the Introduction Section, the hardware host platforms for the Secure64 Cryptographic Module are standard Itanium based HP servers. The hardware contents of these standard platforms are published and widely known. There is no additional information to be learned from observing or peeking into the hardware platforms.

The Cryptographic Module is firmware that executes on these standard HP server platforms. This firmware is fully encrypted and digitally signed when stored on disk, and is decrypted and made operational only after being decrypted, installed into RAM, and integrity tested. Once operational, the executable code is both unreadable and unwritable by any other firmware or software.

Probing or observing the firmware in the clear would require attaching high-speed, multi-lead hardware analyzers to internal hardware buses, or reading the contents of RAMs containing keys, executable firmware, or CSPs in the clear. The hardware cases of the HP platforms each have a removable cover. Any of these attacks would require removing the cover of the HP server hardware case. Attempts that would not remove the cover would cause considerable and evident damage to the hardware case.

Each removable cover is fastened by a lockable lever, as shown in photographs in Sections 3.1 and 3.2. Once a lever is opened, the cover must be slid back in order to be removed. The inside edges of the backs and sides of these covers contain metal posts that slide into retention grooves on the sides and backs of the hardware cases whenever the covers slide forward for closing the lockable lever. Prying any cover edge open would cause evident damage both to the cover and to the hardware case.

The Secure64 Cryptographic Module security policy also specifies that tamper evident seals must be attached to the hardware case before the product system image is down-loaded to the customer. This assures that the lockable latch cannot be opened, nor the case cover removed, without leaving visual evidence of the attempt.

Tamper evident seals also must be attached to the two power supplies, or to one empty power supply bay cover and one power supply, on the rear of the hardware case. This ensures that attempts to gain access through the power supply bays also will leave visual evidence of the attempt.

The placement of the tamper-evident seal on the top of the HP rx2660 covers the movable end of the latch lever. The placement of the tamper-evident seal on the top of the HP rx3600 also covers the movable end of the latch lever, but also is required to extend over boundary between the main cover and an adjacent small cover. The latter is removable only if the main cover has been removed. The placements of these seals are shown in the photos in Sections 3.1 and 3.2.

The placement of tamper-evident seals over the HP rx2660 power supplies or empty power supply bay cover requires up to three seals. For a configuration with a single power supply, a first horizontal seal is used to cover the thumb holes in the cover of the empty power supply bay. Two

additional vertical seals are placed at right angles to, covering each end of, the first seal. These vertical seals are affixed to the top cover, and extend beneath the power supply bays. This prevents the empty power supply bay cover from being removed without leaving evidence, and prevents the single power supply from being removed without leaving evidence. When both power supplies are present, a single vertical seal, covering the adjacent sides of the two power supplies is used. The placements of these seals also is shown in the photos in Section 3.1. For the HP rx3600, a single seal covers either both power supplies, or an empty bay cover and a single power supply. This seal is shown in the photos in Section 3.2.

Customers are expected to acquire hardware platforms directly from HP, or from their preferred distributor. Secure64 can establish agreements with HP and leading distributors that will assure that the hardware platform reaches the customer intact, with the required tamper-evident seals already in place. Customers who acquire their own hardware platforms will be notified that in order to fully trust the FIPS certification, they themselves are responsible for ensuring that the platform is delivered to them in a manner not permitting prior physical intrusion, and that the prescribed tamper evident seals are applied prior to downloading the Secure64 product load image. For these customers, the seals and directions for applying them will be included in the installation package.

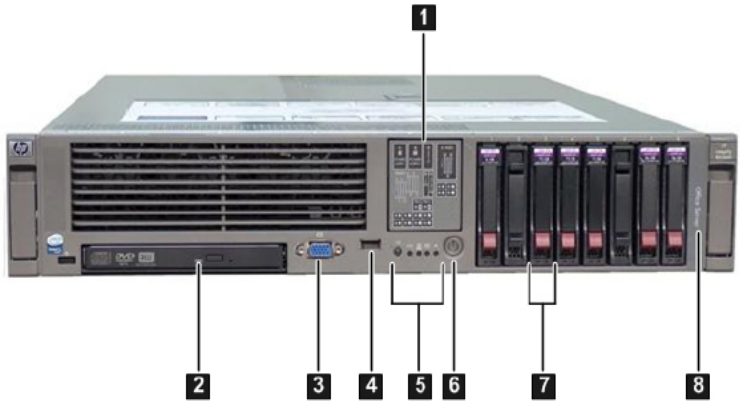
Note that the standard HP sever hardware platforms were not designed to be used in a hostile physical environment. They were designed to be used in a “data center” environment, and it is the customer's *securityadmin* Crypto officer’s responsibility to ensure the physical security of the host platform. This will require performing periodic inspections of the tamper-evident seals, and may require other tests, as required by a particular organization's security policies. Such a customer security policy may even require even stronger physical protection measures - such as placement of the HP hardware platform in a locked and RF-shielded rack. This would preclude any physical contact with the server platform itself without seriously damaging the rack.

Photographs and more details about the HP rx2660 and rx3600 server platforms follow on pages 36-41. Table 8, starting on page 42, summarizes the Security Policy.

3.1 HP rx2660 server physical hardware

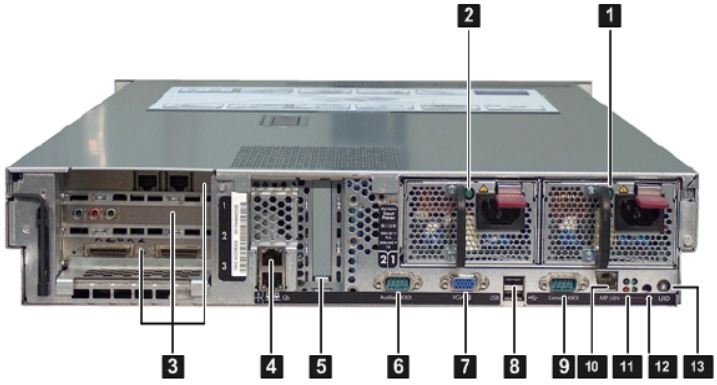
3.1.1 Tamper Detection

The HP Integrity Server rx2660 is a commercially available hardware platform illustrated below. The only method to access the physical components of the crypto module is to unlock and remove the top cover on the hardware case. The cover plate is locked in place with a latch and is not removable due to metal posts attached to the cover that slide into grooves on the side plates. The Physical Security Policy requires that tamper-evident tape be placed over the locking lever such that any attempt to unlock the case would be detected by inspection. The tape meets the tamper detection requirements of FIPS 140-2. Any attempt to open the case otherwise would result in highly visible physical damage.



1 System Insight Display LEDs
2 DVD LED
3 VGA port
4 USB port
5 Server status LEDs
6 Power button
7 SAS disk drive LEDs
8 Office Friendly server label

Illustration 1: front



1 Power supply 1 and LED
2 Power supply 2 and LED
3 PCI-X/PCIe slots
4 Core LAN ports
5 Smart Array P400 controller slot
6 Auxiliary serial port
7 VGA port
8 USB ports
9 Console serial port
10 iLO 2 MP LAN port
11 iLO 2 MP status LEDs
12 iLO 2 MP reset
13 UID button/LED

Illustration 2: back

To remove the top cover, follow these steps.

1. Unlock the cover release lever (if necessary) by turning the cam approximately 90 degrees counterclockwise with the Allen wrench provided on the rear panel of the server (1).
2. Pull up on the cover release lever to disengage the top cover from the chassis (2).
3. Slide the cover toward the rear of the server until the tabs release from the slots in the chassis (3).
4. Lift the cover off the chassis (4).

Figure 3-5 Removing the Top Cover

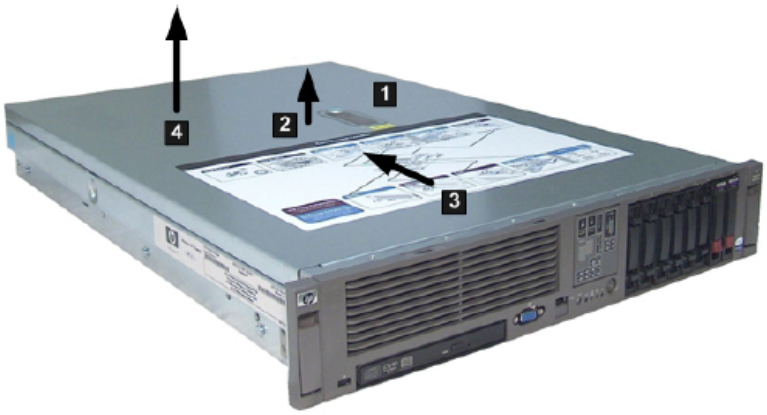


Illustration 3: removal of locking cover is only method of access

Illustration 4: tamper-evident security tape covering lock and cover latch, two power supplies

Illustration 5: tamper-evident security tape covering lock and cover latch, power supply bay and power supply

3.1.2 EMC/EMI

The HP rx2660 server is a "class A" electronic device with respect to electromagnetic emissions. If placed in an RF-shielded locking cabinet, the server could meet "class B" radiation requirements.

For both the HP rx2660 and rx3600 server platforms the only hardware components which are part of the cryptographic boundary are the CPU and DIMM memory modules, as indicated in the general component block diagram shown in the following illustration 6.

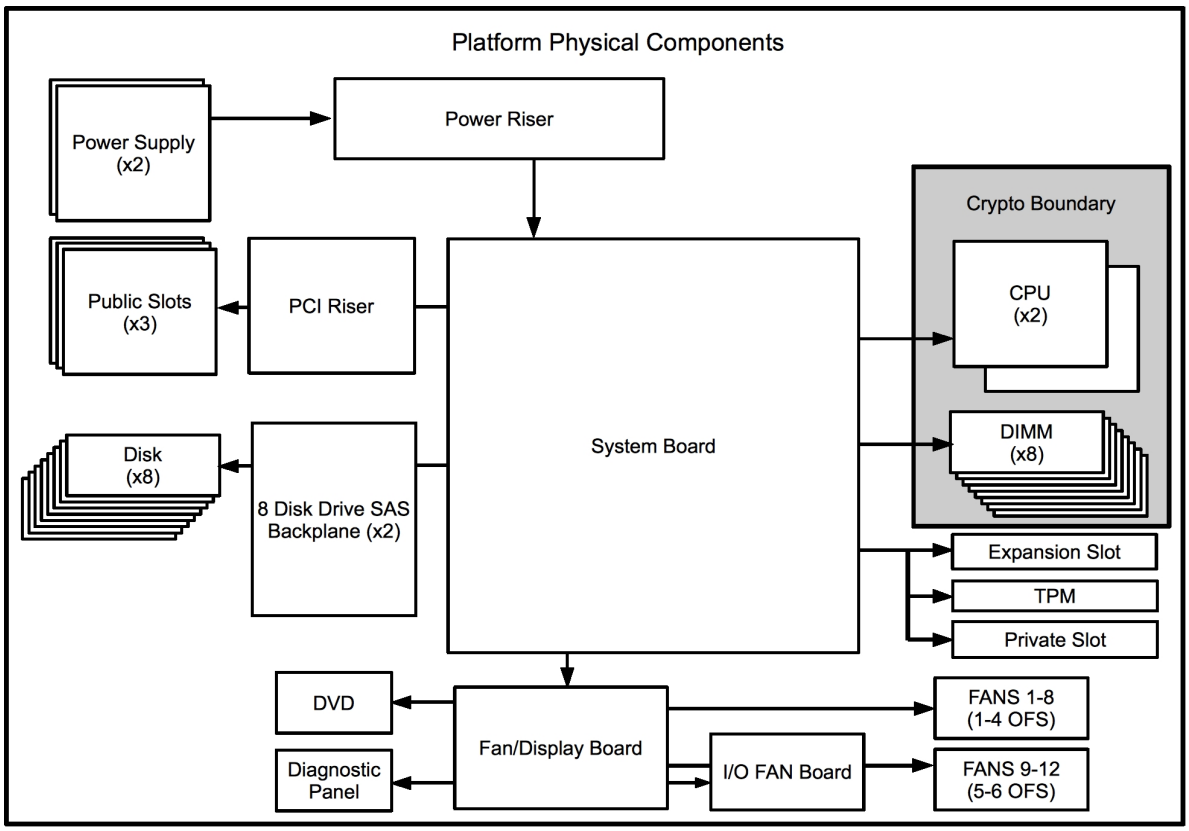


Illustration 6: Block Diagram of platform Physical Components indicating Crypto Boundary

3.2 HP rx3600 server physical hardware

3.2.1 Tamper Detection

The HP Integrity Server rx3600 is a commercially available hardware platform illustrated below. It is similar to the rx2660, but supports larger H/W configurations. The only method to access the physical components of the crypto module is to unlock and remove the top cover on the hardware case. The cover plate is locked in place with a latch and is not removable due to metal posts attached to the cover that slide into grooves on the side plates. The Physical Security Policy requires that tamper-evident tape be placed over the locking lever such that any attempt to unlock the case would be detected by inspection. The tape meets the tamper detection requirements of FIPS 140-2. Any attempt to open the case otherwise would result in highly visible physical damage.

Illustration 7: front plastic cover is vented. Aluminum cage behind plastic has small vent holes.

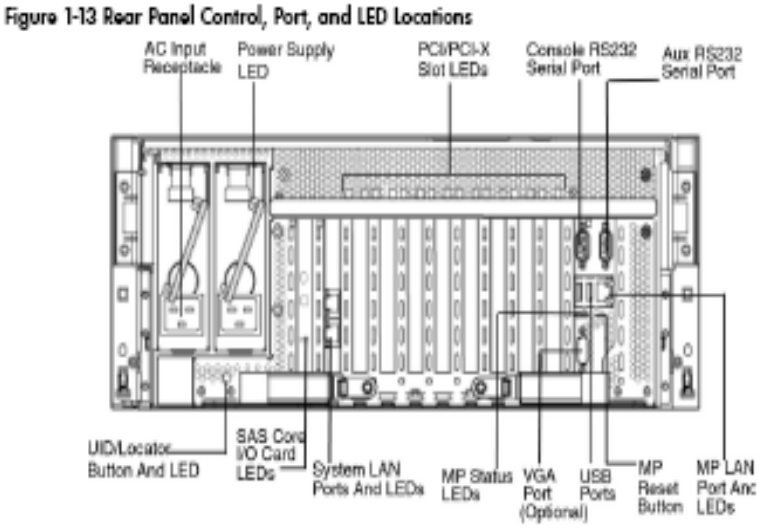


Illustration 8: back

Illustration 9: rx3600 showing locking lever secured with tamper-detection tape

Illustration 10: tamper-evident security tape covering power supply bay and power supply

3.2.2 EMC/EMI

The HP rx3600 server is a "class A" electronic device with respect to electromagnetic emissions. If placed in an RF-shielded locking cabinet, the server could meet "class B" radiation requirements.

Physical Security Mechanisms	Recommended Frequency of Inspection/Test	Inspection/Test Guidance Details
Prevention of Cryptographic Module modification or replacement. ● The System will not boot if any tampering has been done to modify the monolithic load image. ● Self-tests will ensure the integrity of the cryptographic module, algorithms and critical system functions.	System and Cryptographic Module Integrity are checked automatically at each system power-up or re-boot. Cryptographic Module self-tests execute automatically during module initialization. The *securityadmin* Crypto officer may optionally reboot or power-cycle to run self-tests on a periodic manual basis depending on organizational security policies.	Failure to boot due to system load image tampering will require a re-download of the complete system to ensure that the Cryptographic Module, operating system and applications are intact. A self-test failure requires the operator to re-boot the system and if failure continues, a re-download to ensure system integrity.
Tamper Evidence ● Periodic inspection of host platform and required tamper-evident seals. ● Periodic inspection of security audit logs ● Optional: Depending on organizational security policy, the *securityadmin* Crypto officer may place additional tamper-evident seals around the platform cover and disk drives during physical system installation and periodically inspect these seals.	Periodic inspection is dependent on organizational security policy. The inspection may be performed monthly, weekly or daily depending on the isolation of the host platform from unauthorized access and the degree of risk containment desired. A system management platform (Nagios, HP OpenView, CA, Tivoli, etc) is highly recommended to monitor the host platform for unexpected outages in which an unauthorized person may have access to the host platform.	Host Platform inspection should examine for traces of tampering with the server platform, extraneous cables, missing covers, faulty cover switch, broken tamper-seals etc. Audit Log inspection should check for any entries indicating unauthorized access, backup, recovery or tamper-evidence (opening the server cover while system is powered on is logged to the server's management processor log) Any unexpected or maintenance outage should be followed by a physical inspection of the host platform.

Physical Security Mechanisms	Recommended Frequency of Inspection/Test	Inspection/Test Guidance Details
Physical Inaccessibility ● Optional, place the hardware platform in a locked and shielded rack. This precludes any physical contact with the platform.	Periodic inspection should be performed to monitor the integrity of the locked and shielded rack.	Any unexpected attempt to compromise the locked and shielded rack should be followed by a physical inspection of the hardware platform, and a reinstallation of the Secure64 product.

Table 8

4. Mitigation of Other Attacks Policy

This section is for information only. It is simply descriptive of the capabilities of the *SourceT* limited operational environment and actions taken by *SourceT*. FIPS 140-2 Level 2 compliance does not require mitigation of other attacks, such as timing attacks, Internet attacks, TEMPEST, physical intrusion detection, and power attacks. Nonetheless, mitigation of a variety of such attacks is provided by the Cryptographic Module and the *Secure64 SourceT* micro operating system with which the module is statically linked. These security and self-defense mechanisms are listed in the following Table 9.

Attack Mitigation provided by Secure64 Cryptographic Module		
Other Attacks	**Mitigation Mechanism**	**Specific Limitations**
1. Timing Attacks	RSA signature generation uses a constant-time algorithm for 1024 bit keys and hence does not require blinding. Blinding is enabled for all other key sizes by default.	
Attack Mitigation provided by Secure64 SourceT Micro Operating System		
Other Attacks	**Mitigation Mechanism**	**Specific Limitations**
1. Prevention of Root Kits	a) Each monolithic load module is uniquely encrypted to its server platform using TPM keys. This prevents forgery. b) The System Loader uses digital signatures to verify the integrity of each step of the load/CM-install process to ensure no modifications to the system have been made.	
2. Resistant to Malware Injection (viruses, trojans, worms)	OS designed to use Itanium RSE (register save engine) to prohibit buffer overflow attacks. The OS architecture prohibits the injection of executable attack code. This architecture has been analyzed by Matasano Security (independent researchers) who have concluded "no architectural flaws that would allow for the injection of foreign code to the *SourceT* system were identified."	

3. Resistant to Memory Scanning Attacks – Secrets can be kept isolated from application or system code that does not have authorization privilege.	No application can read or write any executable code, nor data placed in a protected memory compartment utilizing an Itanium hardware memory protection key. The Secure64 Cryptographic Module runs in its own compartment to protect its data.	
4. Resistant to Escalation of Privilege Attacks -- Defense in Depth Protections	OS uses all 4 levels of privilege with only minimal required mechanisms defined at the highest privilege level. (Typical operating systems such as Linux use only two privilege levels, user & kernel).	
5. Denial of Service Mitigation	Mechanisms in the TCP network stack prevent a variety of attacks such as TCP-SYN floods, reflected floods, amplified floods, etc.	Good traffic can get through to the applications under DDoS attack up to line saturation.
6. Physical Intrusion Response	An intrusion signal is generated by the hardware whenever a removable cover is opened while the system is running. The Secure64 *SourceT* system contains logic to detect this signal and immediately call the "PL/1 Shutdown CM" service to zeroize and shut down the Cryptographic Module. The event is also recorded in the system log and the *SourceT* system is notified.	

Table 9

Appendix A – NIST Approved Algorithms

NIST Certified Algorithms

Cert Num	Algorithms	Certificate Number	Date	Key Sizes	Referenced Certificates
1	AES - ECB (e/d) - CBC (e/d)	882	10/07/08	128 / 192 / 256	none
2	AES - CBC (e/d)	956	12/12/08	128 / 192 / 256	none
3	DSA - PQG(gen) - Keygen(Y) - SIG(gen) (ver) all: MOD 1024	350	03/31/09	1024	RNG - #507 SHA-1 - #874
4	HMAC SHA-1	580	03/31/09	KS < BS, KS=BS, KS>BS	SHA-1 - #874
5	RNG ANSI X9.31	507	10/07/08	AES-128 / 192 / 256	none
6	RSA – Keygen - Alg ANSI X9.31 - SIG(gen) (ver) RSASSA PKCS #1 v1.5 - SIG(gen) (ver)	495	03/31/09	1024 / 1536 / 2048 pub keys 3, 7, 65537	RNG - #507 SHA-1 - #874
7	RSA – Keygen - Alg ANSI X9.31 - SIG(gen) (ver) RSASSA PKCS #1 v1.5 - SIG(gen) (ver)	426	10/07/08	1024 / 1536 / 2048 pub keys 3, 7, 65537	RNG - #507 SHA-1 - #874
8	SHA-1 (Byte Only)	874	10/07/08	N/A	none
9	SHA-256 (Byte Only)	936	12/24/08	N/A	none
10	Triple DES - TECB(e/d): - TCBC(e/d): - - KO 1,2	722	10/07/08	2-key, 3-key	none

Appendix B – Firmware Configuration Control

Source Code Control and Defect Tracking

Product integrity requires that the relevant components of firmware and software development are identified and placed under configuration control. Secure64 uses the configuration management system *Subversion* (subversion.tigris.org), an open source version control system. Subversion provides a source-code control system for managing code updates, check-ins, version control, etc. Each Secure64 product release is represented as a 'tag' in the Subversion control system.

Because the Cryptographic Module will be used in multiple Secure64 products, additional protections are used to ensure that the source files for the Cryptographic Module remain unchanged. A file of the SHA-256 hashes of each Cryptographic Module source file is added to the source tree. This permits validating that no Cryptographic Module source file has changed, and identifying which particular files have changed when the Module is subsequently improved and requires re-certification.

Delivery and Installation

As described in this document, the components of the Secure64 Cryptographic Module are included in the signed and encrypted product load image that is downloaded from the Secure64 facility into the target platform's disk. At each system boot time, the boot loader installs the Cryptographic Module from these components, and places the initial KEK and authorization CSPs into the Cryptographic Module's memory compartment. This is sanctioned by the FIPS 140-2 Implementation Guidance document, Section 7.7, because all of the code of the limited operational environment is contained within the Cryptographic Module software physical boundary. The installation, at each boot, of the Cryptographic Module into RAM is the final step of delivery and installation to the target machine.