S/390 CMOS Cryptographic Coprocessor Architecture: Overview and design considerations

by P. C. Yeh
R. M. Smith, Sr.

This paper describes the design objectives and presents an overview of the design for the IBM S/390® CMOS Cryptographic Coprocessor, also known as the S/390 cryptographic module (SCM). The SCM is fully compatible with the earlier S/390 cryptographic module, ICRF (Integrated Cryptographic Facility), and has been certified by the National Institute of Standards and Technology at the highest level of security qualification. The principal features and unique characteristics of the SCM are summarized in the context of the architecture design.

Introduction

With the explosive growth of Internet applications, demand for information protection has become prevalent. Lack of network security has been recognized by many as the single most significant barrier to the progress of e-business. Cryptography is an effective means of protecting information and authenticating users, and is commonly used to improve network security. Furthermore, hardware cryptographic devices are often used for higher security and better performance. It is well recognized in the security community that hardware cryptographic devices provide additional security because of hardware-enforced protection and control of security-related data.

The S/390* CMOS cryptographic coprocessor, also known as the SCM (S/390 cryptographic module), provides high performance, reliability, and security for various applications. The SCM is fully compatible with the Integrated Cryptographic Facility (ICRF) [1, 2], a previous S/390 cryptographic module, and the National Institute of Standards and Technology (NIST) has granted it an overall Level 4, the highest security level of Federal Information Processing Standards (FIPS) 140-1 certification [3].

The SCM is available on G3 and subsequent S/390 enterprise servers, and supports both DES-based and public-key applications, including data encryption using DES or triple-DES (TDES), message-authentication-code (MAC) processing using one or more DES keys, personal-identification-number (PIN) processing, DES-key

0018-8646/99/$5.00 © 1999 IBM

management, 1024-bit RSA and Digital Signature Standard (DSS) [4] digital signature, 1024-bit Diffie–Hellman (DH) key-agreement protocol, and private-key management.

This paper first outlines the design objectives and the intended operating environments; an overview of the SCM is then presented. Main features and unique characteristics of the SCM are summarized, and a detailed discussion of the pseudorandom number generator (PRNG) design is provided. Major considerations that shaped the SCM design are discussed, and the rationale behind some of the decisions is also described.

Note that the FIPS 140-1 Level 4 certification of the SCM specifically excludes the PRNG, not because of any known weakness in the PRNG but because it does not implement a FIPS-approved algorithm. The PRNG was designed before the FIPS-approved algorithms were selected. To clearly explain and justify the PRNG design in light of this exclusion, the section on the PRNG includes more detail than the rest of the paper.

The SCM provides a rich set of functions and a wide range of controls to support customers with various security requirements, ranging from the highest, for the very sophisticated security-conscious customer, to the simplest. The SCM can be configured to any desired level of security by establishing proper controls and disabling unused functions through a secure workstation. Note also that not every function mentioned in this paper is currently supported by the software and the secure workstation, and the terminology used in this paper is not necessarily consistent with that of other IBM manuals or publications.

Design objectives

In addition to meeting the general requirements of high performance and reliability for enterprise servers, the SCM is designed to achieve the following specific objectives as well:

1. *To be a follow-on product to ICRF*. This objective requires that the SCM implement all ICRF application functions so that existing applications can run with no changes.
2. *To enhance the DES-based functions*. Several functions provided in ICRF needed improvement. Among these were better control of the DES key export function, more flexibility of control vector usage [1], and support of TDES for data encryption and MAC using multiple DES keys.
3. *To support PKA (public-key algorithm)* applications. To support applications for e-business, it is essential to support commonly used PKAs and newly emerging standards, including RSA, DSS, and DH.
4. *To improve manual operations for high-security users*. With ICRF, control of the cryptographic module and installing of initial keys are manual operations which must be performed at a control panel attached to the server. In most installations, the server is placed in an area (sometimes referred to as a "darkroom") which is kept secure and is not easily accessible to security officers. In such installations, it is desirable to provide the "manual" functions by means of a remote capability, rather than to require security officers to enter the darkroom area.
5. *To facilitate the establishment of mirror images for high-security users*. In many enterprises, multiple systems are established with a mirror image for backup or performance. For ICRF, this requires the same manual operations to be performed repeatedly. This problem creates operational complexity and increases cost for managing cryptographic systems, particularly for enterprises that have multiple geographically separated systems. A solution that allows security officers to create mirror images at a remote site with security is desirable.
6. *To simplify manual key entry for normal users*. Many customers do not require the maximum security provided by the SCM, and have requested a simple means to install initial keys.
7. *To be the most secure cryptographic product commercially available*. In many installations, S/390 enterprise servers are used to perform critical operations and high-value transactions that require extremely high security. This requires both cryptographic strength and physical security. Cryptographic strength is provided by using the most secure cryptographic algorithms currently known. The goal for physical security was to comply with the implementation security requirements for Level 4 of FIPS 140-1.
8. *To be exportable*. Cryptographic devices are export-controlled. ICRF is an optional feature and is included only in machines for entitled customers. This solves the export problem for ICRF. For the SCM on CMOS machines, however, the cost of the cryptographic module is smaller than the cost to provide additional part numbers, and it is cheaper to include the SCM as a standard part of all CMOS machines. This requires a means of enforcing the export regulations in hardware so that all machines are exportable.

Operating environments

The SCM is designed for IBM S/390 Parallel Enterprise Server* computers. It is initialized, customized, and configured by customer security officers for use by the control program. Applications must invoke the control program to obtain cryptographic services. The control program issues cryptographic functions on behalf of

applications. Security-relevant cryptographic functions used by the control program can be partially or entirely disabled by security officers. This allows security officers to configure the SCM for various control-program authorities, including any combination of SCM use and SCM update. With regard to manual operations, the SCM is designed to be used in three different environments: interactive secure workstation, control-program protected, and "load and lock."

- *Interactive secure workstation environment*

The primary thrust of the SCM design is a direct follow-on to the ICRF, but the manual controls of the ICRF are replaced with a secure workstation. In this environment, security officers can monitor the status of the SCM; enable and disable a particular group of functions, a cryptographic domain, or the entire module; and perform key entry concurrently with other operations—all from a remote location with complete security and integrity. In this case, security and integrity are enforced by the SCM. Additionally, multiple control is provided for performance of critical functions. In some documentation, this is called the TKE (trusted key entry) environment.

In this environment, the secure workstation becomes a logical extension to the SCM and can be used in conjunction with it to provide the optimum balance between performance and flexibility by implementing the high-usage cryptographic operations within the SCM while permitting the less frequent operations to be performed in the associated secure workstations. Thus, the secure workstation may be used in the management of DES and private keys, including generation, split-knowledge splitting and restoration, distribution, preservation, and migration of private keys from smaller servers.

- *Control-program-protected environment*

For customers who require a simple means for cryptographic system management, a set of clear master-key entry functions is provided to allow security officers to install the master keys without requiring a secure workstation. In this case, security and integrity are enforced by the control program. Each key part is entered "in the clear" into the SCM through the control program. The clear master-key entry functions can be disabled by setting proper controls in the SCM. Separate controls are provided for each of the three master-key registers in each of the sixteen domains. In some documentation, this is called the non-TKE environment.

- *Load and lock environment*

The SCM provides the security-conscious customer who has limited resources with a simple mechanism to protect the SCM. Again, no secure workstation is required in this environment. The customer can perform customization of the SCM as a special step by physically securing the machine room, loading the master keys with the clear master-key entry functions, and then locking the SCM by properly setting up controls in SCM so that these functions are disabled and cannot be re-enabled without an SCM reset, which clears all customer information.

- *PR/SM environment*

Under PR/SM,[1] the SCM is shared among systems in different partitions, running simultaneously on the same machine. Inside the SCM, special hardware is provided to support sixteen independent cryptographic domains in order to achieve high protection and isolation among partitions. Outside the SCM, PR/SM controls are provided so that, for a given partition, the cryptographic capability may be totally disabled, or may be entirely or partially enabled by the PR/SM operator. Critical functions that reset the SCM or change the SCM contents can also be disabled by the operator.

SCM overview

- *Performance and availability*

For performance and availability, up to two SCMs are provided in each machine. Each SCM is physically attached to a different CPU (central processing unit).

Synchronous, asynchronous, and concurrent operations

A cryptographic operation may be synchronous or asynchronous with respect to the requesting CPU. Functions requiring a significant amount of processing time are performed asynchronously with respect to the requesting CPU in order to avoid tying up the CPU. Typically, these functions use public-key algorithms or other complex algorithms. All other functions are performed synchronously with respect to the CPU to avoid the overhead of asynchronous functions. Synchronous functions can be requested only by CPUs that have an SCM attached. To handle asynchronous execution, special hardware queues are provided and shared by all CPUs. New instructions are provided to permit all CPUs to send asynchronous functions (called requests) to the SCM by means of the hardware queues. The CPU attached to the SCM sends these requests from the hardware queues to the SCM and places the results (called replies) back in these queues. Except for a small time period required for initiation and completion, the asynchronous operations are

[1] Processor Resource/Systems Manager* (PR/SM*) is a hardware feature that allows the resources of a machine to be shared dynamically among multiple, independent "partitions." Each partition can run an operating system, and all partitions can operate simultaneously.

performed without interfering with synchronous operations.

In the simplest preliminary design, while the SCM was performing asynchronous functions, it would have appeared busy to any synchronous functions. However, it is desirable to permit concurrent execution of synchronous and asynchronous functions to avoid blocking effects. To permit this type of overlap, the SCM includes two processors, a synchronous processor and an asynchronous processor.

Note that all ICRF functions are synchronous. It would have been possible to change these original functions to use the hardware queues, thus making them available to all CPUs, but this was not done for the SCM because it would have resulted in a substantial incompatibility, and the performance benefit would have been marginal.

- *Function and control*

The SCM provides two types of functions, PKSC (public-key security control) and normal. PKSC functions are usually issued by security officers at secure workstations; normal SCM functions are issued by the control program.

Hardware-enforced authentication and authorization are provided for security-relevant PKSC functions. Identification is achieved by means of RSA digital signature. These signature-controlled PKSC functions are further divided into single-signature or multiple-signature functions. The latter allow multiple control of the function performance. A set of PKSC authorization masks in the SCM provides control at a finer granularity than the function level. Several query functions, which read out SCM status, are the only PKSC functions that do not require authentication and authorization to be performed by the SCM.

Security-relevant normal SCM functions can be disabled by means of a profile register in SCM. The control program can issue one of these functions only if the function is enabled, as specified in the profile register. The contents of the profile register can be changed only by authorized security officers using PKSC functions. The following normal SCM functions have no reliance on any secret data, are not subject to profile control, and are always available:

1. Generate hash.
2. Verify DSS digital signature.
3. Modular exponentiation.

- *Cryptographic strength*

The lifespan of S/390 enterprise servers can be more than ten years. The security provided by the SCM must not only meet today's highest requirement but also accommodate future advances in technology.

A design guideline to pursue the highest security possible was developed, and a minimum security level was established. Since the DES master key (DMK), carried over from ICRF, was 128 bits, the minimum security level was established as a work factor of 2^{128}. It was also a goal that no new portion added should be the weakest link. Thus, all new secret values maintained inside the SCM had to be at least 128 bits.

- *Physical security*

Strong cryptographic algorithms alone do not provide overall system security; physical security is also required, but the effectiveness of such a design cannot be easily validated by customers. ICRF was designed to meet the physical security requirements of FIPS 1027 [5], the only standard available at the time. In the past, many customers had suggested that the physical security of ICRF be evaluated by independent laboratories. This was not done because of a lack of standardized evaluation criteria and processes.

During the design of the SCM infrastructure (1992–1994), FIPS 140-1 was proposed to replace FIPS 1027 and to include a validation process. Even though the basic design of the SCM was completed before FIPS 140-1 was adopted, the goal was to comply with the highest security level of the standard.

The SCM is a single-chip cryptographic module, and the chip itself forms a physically secure boundary. The tamper-resistant design protects security-related information in the SCM against various physical intrusions and probing. When the SCM is removed from the machine, all customer information inside the module is automatically erased.

- *Secure boundary*

The design prevents security-related information maintained inside the SCM from being compromised through subversion of any system component outside the SCM, including the control program, CPU microcode, and system diagnostic tools. All cryptographic operations are performed entirely by hardwired circuitry inside the secure boundary. The CPU microcode, which interprets instructions, does not perform any security-relevant cryptographic operations. No intermediate value of any cryptographic operation ever leaves the boundary. Secret data maintained within the boundary never leaves the boundary unencrypted.

- *Nonvolatility*

The SCM has two power sources, a primary power and a backup battery. When the primary power is turned off, customer security-related data is maintained by the backup battery. When the primary power is turned back on, the

SCM resumes the state that existed before the primary was turned off.

- *Module structure*

Figure 1 visually summarizes the major portions of the SCM. The SCM implements special hardware for DES, PIN, SHA-1 [6], PRNG, and modular exponentiation. In addition to its use by the more complex RSA, DSS, and DH operations, the modular exponentiation operation is also provided as an uncontrolled normal SCM function. The modulus size for the modular exponentiation, RSA, DSS, and DH functions can be up to 1024 bits. The SCM includes registers for maintaining customer security-related data, including master keys, identification data, and authorization controls.

Burned-in values

As part of the manufacturing process, each SCM is assigned a 128-bit unique cryptographic module ID (CMID). A unique 1024-bit RSA key pair is also generated for each SCM; the RSA key pair includes a 1024-bit cryptographic module secret exponent (CMSE) and a 1024-bit cryptographic module public modulus (CMPM). During the manufacturing process, the CMID and CMSE are burned into the SCM. Because of the technology used, there is not sufficient room on the SCM for the entire CMPM as a burned-in value, so a 128-bit MDC-4 hash value [7] of the CMPM is burned in.

Initialization and export controls

During the initialization process, a cryptographic configuration control (CCC) and the 1024-bit CMPM are loaded into the SCM. The CCC is used to enforce the availability of algorithms and key lengths controlled by U.S. export regulations. The burned-in MDC-4 hash value is used by the SCM to verify the public modulus loaded during the initialization process.

PKSC registers

A 128-bit cryptographic module signature sequence number (CMSSN) register is included in the SCM. The CMSSN is included in each message signed by the SCM and is incremented each time it is used. It provides an audit trail of all activity performed on the SCM.

Each security officer has a unique 1024-bit RSA key pair assigned. The SCM provides 16 security officer identification (SOI) registers; each holds a security officer public modulus (SOPM) and a 128-bit transaction sequence number (TSN) register. The TSN in the request message eliminates the possibility of replaying a previously signed PKSC request.

To facilitate support for multiple-signature PKSC functions, the SCM provides a signature requirement array (SRA). For each of these functions, the SRA specifies the number of signatures required and the security officers who are authorized to sign. A requested multiple-signature PKSC function is placed in a pending-request register (PRR) until all requirements in the SRA have been satisfied.

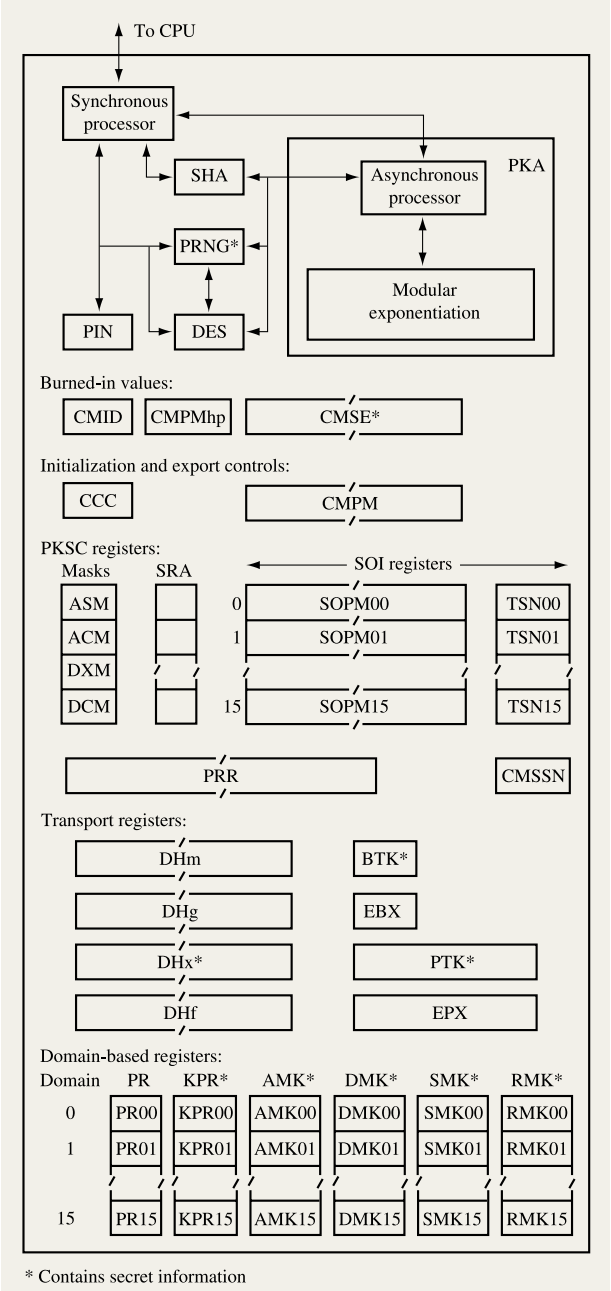

Figure 1

SCM overview.

Four PKSC authorization masks in the SCM control the use and update of each SOI register and the effects of certain PKSC functions on domains.

Transport registers
Secret information entering or leaving the SCM by means of PKSC functions is protected using the transport registers. The transport mechanism is based on the Diffie–Hellman (DH) protocol. All DH registers are 1024 bits long and include a modulus (DHm), a generator (DHg), a secret exponent (DHx), and the public result of the exponentiation (DHf).

The DH protocol results in two secret transport keys: a 128-bit basic transport key (BTK) and a 320-bit public transport key (PTK). The use of two transport keys provides better separation between DES-based and PKA-based operations and also reduces the problems involved in making the SCM exportable.

Several transport operations involve extracting a master key, or a value encrypted under a master key, and encrypting the value under a transport key. These operations use the encrypted basic extracted key (EBX) and encrypted public extracted key (EPX) registers to hold the results.

Domain-based registers
Domain-based registers are replicated for each of the sixteen cryptographic domains. Three master keys are used to protect application keys maintained outside the secure boundary. The DES master key (DMK) protects DES keys, the signature master key (SMK) protects private keys for digital signature, and the receive master key (RMK) protects private keys for importing DES keys. The use of three master keys provides maximum separation among the three types of applications. An auxiliary DES master key (AMK) is provided to facilitate dynamic update of the DES master key.

A profile register is used to enable or disable certain normal SCM functions. These functions are divided into groups. A bit in the profile register is assigned to each of these groups. A set of key part registers is provided to enhance the manual key entry process. This allows up to three key parts to be stored inside the SCM before the control program accepts them.

PKSC facility

The Integrated Cryptographic Facility (ICRF) implemented on bipolar machines included a manual control panel for use by security officers. The control panel, placed on the server, included a key entry device and manual switches. The manual switches, operated by brass keys, provided cryptographic controls with dual control. Secure communication between the control panel and the tamper-resistant cryptographic module was provided by means of a tamper-resistant cable. The cost of this approach, especially with regard to the secure cable, was significant in previous implementations, but with the CMOS technology used for SCM, the entire cryptographic module is implemented in a single chip, and it is not practical to attach a tamper-resistant cable to it.

With the SCM, the manual control panel and the key part entry device are provided by means of remote secure workstations, the secure cable is replaced by the use of public keys and digital signature over a public channel which does not require security or integrity, and the dual control function is enhanced by means of an *N*-of-*M* voting control mechanism.

The PKSC facility consists of a number of functions and registers for supporting module initialization, key entry, and module control from a remote site with extremely high security and integrity in the interactive secure workstation environment. This facility is a replacement of the tamper-resistant cable used in ICRF and provides a great enhancement for cryptographic system management.

- *PKSC functions*

PKSC functions are divided into the following categories: initialization, query, transport-key management, multiple signature, and co-sign. The initialization functions are used only during the initialization process and are not discussed here in detail. The query functions are provided to examine the SCM status, and query requests are not signature-controlled. All other PKSC functions are signature-controlled. The transport-key management functions are used to establish transport keys using the Diffie–Hellman protocol and are single-signature functions. The multiple-signature functions are provided for performing key entry, control setup, and other critical functions. Performance of multiple-signature functions is subject to control specified in the signature requirement array (SRA) and is achieved by digital signature using the co-sign function, which itself is a single-signature function.

- *PKSC security*

PKSC public audit
Public audit of the SCM status and the results of previous operations is provided by means of query functions. Query functions are provided to read all public information and the MDC-4 hash value for each secret value in the SCM. Queries can be requested without requiring any authentication and authorization to be performed by the SCM. This allows public audit of the SCM to be performed by anyone to ensure integrity at each step of all security-critical processes, such as installation of identification data,

authorization controls, and master keys, and transfer of master keys or PKA objects.

PKSC authentication

Messages transmitted between the SCM and security officers at secure workstations using PKSC functions are authenticated for originator and integrity in both directions. This is achieved by using the RSA digital signature. A request message for a security-relevant PKSC function must be signed by the requesting security officer using the security officer's private key. The signature is verified by the SCM using the public modulus in the SOI register associated with the requesting security officer before the request is accepted. A security-relevant reply message for a PKSC request is signed by the SCM using the SCM's private key. The signature is verified by the receiving security officer. In either direction, an ISO-9796 signature [8] of an MDC-4 hash value of the message is used, and a fixed value of 65537 ($2^{16} + 1$) is used as the public exponent for all SCMs and security officers.

Message freshness is also ensured in both directions to eliminate the possibility of replaying any signed message. Query requests include a 128-bit value called a query ID, which is returned in the signed reply. The security officer can ensure the freshness of reply messages by placing a fresh random number in each query request. Replay of signed requests is prevented by means of the TSN field in the SOI register. When the security officer public modulus is loaded into the SOI register, a fresh 128-bit random number is placed in the TSN field. After the SOI register has been loaded, the security officer must query the current TSN. The current TSN must be included in the signed portion of the request; if the TSN in the request does not match the TSN in the SOI register, the request is rejected by the SCM. Each time a signed request is accepted, the SCM increments the TSN for the requestor. Thus, each request must be different. Each signed reply contains the MDC-4 hash value of the original request in the signed portion of the reply message, thus ensuring freshness of the reply message.

When a multiple-signature request is pending in the PRR, the MDC-4 hash value of the entire original request, called a PR hash, is included in the PRR. Co-sign requests by security officers, who are authorized to supply additional signatures, must include this PR hash. This ensures that the pending request has not been changed between the time the security officer queries the PRR and the time the request is co-signed.

Certain operations must be performed using multiple requests. For example, several query functions must be performed to obtain a complete report of the SCM status. As another example, multiple transport-key management functions must be performed to achieve the Diffie–Hellman key agreement protocol. To solve the atomicity problems, the cryptographic module signature sequence number (CMSSN) is used. This number is set to a random number during the initialization process. Each time, after a reply message is signed by the SCM, the CMSSN is incremented. Since performance of any PKSC function that changes the SCM status provides a signed reply message, consecutive CMSSNs returned by sequential PKSC executions ensure that no such intervening function execution has occurred.

PKSC authorization

A security officer can request signature-controlled functions only if the public modulus for that security officer is in an SOI register enabled for verifying the signature of a signed request. After the signature of the request is verified by the SCM, a single-signature request can be performed immediately, but a multiple-signature request is placed in the PRR for further authorization checking using the SRA.

PKSC authorization masks

Four PKSC authorization masks provide additional control of certain PKSC functions. They are summarized as follows:

1. *Security officer signature mask (ASM)* This mask specifies which SOI registers can be used to verify the signature for a signed request. This allows activation and deactivation of security officers without requiring the corresponding SOI registers to be updated.
2. *Security officer identification register change mask (ACM)* This mask specifies which SOI registers can be updated by the load SOI register function. This provides a safeguard to prevent security officers of critical systems from being removed from the SCM.
3. *Domain-extraction mask (DXM)* This mask specifies the domains whose master keys are allowed to be transferred out by means of the extract-and-encrypt functions. This mask also controls whether PKA objects are allowed to be transferred in or out of a particular domain by means of the import PKA object and export PKA object functions.
4. *Domain-change mask (DCM)* This mask specifies the domains whose registers (master keys, key part registers, and profile register) can be changed by multiple-signature functions. This mask can be used to protect the domains for critical systems from being modified.

Handling of multiple-signature request

Performance of security-critical PKSC functions may require multiple signatures, as specified in SRA, from different security officers. Before all requirements specified in SRA are fulfilled, the request is placed in the pending-request register (PRR). Authorized security

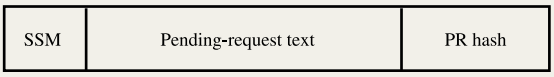

Figure 2

Pending-request register.

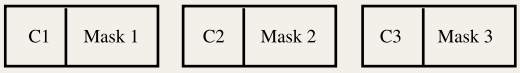

Figure 3

Entry of signature requirement array.

officers can provide needed signatures by using the co-sign function to sign the pending request.

The following summarizes the multiple-signature functions:

1. Load SOI register.
2. Load SRA and PKSC masks.
3. Zeroize domain.
4. Load profile register.
5. Create transport keys using DH.
6. Load DES key or DMK.
7. Load RMK or SMK.
8. Extract and encrypt DMK.
9. Extract and encrypt RMK or SMK.
10. Import PKA object.
11. Export PKA object.

Pending-request register

The pending-request register has the format shown in **Figure 2**. The signature summary mask (SSM) contains 16 bits and indicates which security officers have signed or co-signed the pending request. The pending-request text contains portions of the pending request. PR hash is the MDC-4 hash value of the entire original request and is included in each co-sign request for the pending request.

Signature requirement array

The signature requirements for multiple-signature functions are specified in the signature requirement array inside the SCM. There is one entry for each multiple-signature function. **Figure 3** shows the entry format. Each entry contains three requirement fields, and each requirement field contains a 4-bit count and a 16-bit mask. The count specifies the number of signatures required, and the mask specifies the security officers whose signatures are counted. If the count is zero, the requirement is considered to be satisfied and the mask is ignored. All three requirements must be met before the pending function is performed.

The three *N*-of-*M* controls can be used in several ways. The three requirement fields could be set up, for example, one for each of three departments and requiring a signature by at least one member of each department; or for two departments and a group of secure workstations, requiring a certain number of signatures from each department and at least two different workstations. In another use, a requirement field can be assigned to a group of automated audit and verification programs, each operating in the role of a notary, recording and then signing the function only if a predefined set of conditions have been met.

Note that, in an SRA entry, if a counter value is greater than the total number of authorized security officers specified in the corresponding mask, the requirement can never be satisfied, and the associated PKSC function is effectively disabled. In the load-and-lock environment, after master keys have been installed using the clear master-key entry functions, the profile register is set to disable these functions. To prevent these functions from subsequently being enabled, and to prevent master keys from being changed by any other means, the load-profile-register, load-SRA-and-PKSC-masks, load-DES-key-or-DMK, and load-RMK-or-SMK PKSC functions must also be disabled by means of specifying the SRA entries so that signature requirements for these four PKSC functions can never be satisfied. The only way to change master keys is to perform an SCM reset, which erases all customer data and places the SCM in the noninitialized state.

Performance of multiple-signature function

When a multiple-signature request is received, the signature is verified. If verification is successful, the requested function is loaded into the PRR, and the bit in the signature summary mask corresponding to the requestor is set to one. If all SRA requirements for the pending function are satisfied, the function is performed; if the requirements are not all satisfied, the request is left in the PRR waiting for additional signatures.

Each time a co-sign request is received, the SCM verifies the signature of the requestor and compares the hash value in the co-sign request with the one in the PRR. If verification is successful and the PR hash values match, the bit corresponding to the co-signing officer in the signature summary mask is set to one. When all signature

requirements are satisfied, the pending function is performed.

PKSC secrecy

Secret information transmitted between the SCM and another external system using PKSC functions is protected under a transport key. Transport keys are derived by using the Diffie–Hellman key agreement protocol. The SCM includes a set of DH registers, two transport-key registers, and several transport-key management functions to support the protocol. This entire process can be audited by using query functions to ensure that the correct transport keys are established between intended systems by authorized security officers. The DH modulus size can be up to 1024 bits, and each execution of the DH key agreement protocol creates two transport keys: one contains 128 bits and the other, 320 bits.

When secret information is protected using the 128-bit basic transport key (BTK), the encryption uses the two-key TDES TECB mode of encryption [9]. When secret information is protected using the 320-bit public transport key (PTK), 192 bits of the PTK are used as three DES keys, and the remaining 128 bits are used as two 64-bit secret initial chaining values. Secret information is first encrypted using CBC-mode DES encryption [10] with the first key and the first secret initial chaining value. The result is then decrypted using CBC-mode DES decryption with the second key and the initial chaining value of zero. The result is again encrypted using CBC-mode DES encryption with the third key and the second secret initial chaining value. This process, sometimes referred to as inner chaining, is much stronger cryptographically than outer chaining, that is, the TDES TCBC mode of operation [9], which might be considered first in our particular use.

The RSA key pair of the SCM is used for authentication only; it is not used for protecting the secrecy of sensitive data. Thus, compromising the key does not itself cause significant exposure, since it is nontrivial to impersonate the SCM in a real-time environment. If the key were also used for encryption, compromising the key could compromise all secret values ever protected under the key.

Using the transport key established by the Diffie–Hellman key agreement protocol is a better approach, because 1) a different transport key can be used for each transaction and 2) the transport key can be erased when the transaction is complete.

PKA facility

The PKA facility provides functions to perform digital signature using RSA and DSS, and DES key distribution using RSA and DH. Private keys in operational form are called PKA objects and are protected under either of two master keys: The signature master key (SMK) protects private keys for digital signature, and the receive master key (RMK) protects private keys for importing DES keys.

Functions using the RMK and SMK are called RMK-based and SMK-based functions, respectively. These functions are profile-controlled, with separate profile bits assigned to RMK-based normal use, RMK-based PKA object import, SMK-based normal use, and SMK-based PKA object import.

Protection of private keys

Since keys used in most public-key algorithms cannot be changed frequently, their lifespan is usually much longer than that of DES keys. In addition to the use of 192-bit RMK and SMK, other special considerations were given to the PKA design to ensure that extremely high security is provided for private-key protection, so that maintaining them outside the SCM is not a weakness.

Information about a private key consists of both secret and public parts. The secret part is strongly encrypted when it is outside the SCM. Although the public part does not require encryption, its integrity affects the secrecy of the corresponding private key. To ensure that no data can be practically substituted or modified, information about a private key is entirely encapsulated. Encapsulation is achieved by placing all information about a private key in an object. The public part is stored in the clear; the secret part is encrypted. The SHA-1 secure hashing algorithm is performed on the clear value of both public and secret parts in the object. The hash value is stored as part of the object; it is generated by the SCM when the object is created and is used by the SCM to verify object integrity when the object is used.

When a PKA function using a private key is requested, the designated object is provided to the SCM as an input operand of the function. Before the object is accepted, the integrity of the object is verified by the SCM. The SCM first recovers the clear value of the private key and generates an SHA-1 hash value on the clear contents of the object. The generated SHA-1 hash value is then compared with the referenced one in the object. If the comparison fails, the object is rejected, and the requested function is not performed.

Secret parts in an RSA, DH, or DSS object are encrypted under a unique 320-bit object protection key. The encryption algorithm is the same as that used by the 320-bit public transport key. The object protection key is encrypted under the 192-bit RMK or SMK using the same algorithm, except that all initial chaining values are zeros. A unique object protection key is used in each object, and the key is internally generated by the SCM when the object is created. This enhances security, because compromising an object protection key compromises only one object. This also enhances the security of the RMK

and SMK, because no plaintext–ciphertext pair of any data under RMK or SMK is available to applications. The encrypted object protection key is encapsulated in the object.

Object and key generation
Functions are provided to create RSA, DH, or DSS private-key objects from clear input, and to import them protected under a DES KEK with a special control vector. Functions are provided to generate DSS and DH private-key objects with the secret information generated internally within the SCM. A field in the object indicates how the object was created.

RSA key generation is not provided in the SCM; rather, the SCM must depend on an external system to generate RSA key pairs for applications. RSA key generation is a complex process and is not suitable for a hardware-only implementation. Additionally, in an enterprise server, it is expected that the number of RSA private keys used will be small, and that RSA key generation will be a very infrequent event. In the interactive secure workstation environment, these keys can be generated in the secure workstation.

Master-key update
It is expected that there will be cases in which the RMK and SMK must be updated. This could be the case, for example, when enterprises are consolidated, when a security officer leaves, or when a key part is compromised. Dynamic master-key update is provided for the DMK. This is accomplished by means of an auxiliary DES master key (AMK) register. The new master key can be loaded into the DMK register, and the old value of DMK can be placed in the AMK register. DES keys encrypted under the old value of DMK can be converted to become encrypted under the new master key by means of the reencipher-from-old-master-key function, which is a profile-controlled, normal SCM function.

Because of the nature, use, and lifetime of private keys, it is expected that control and handling of objects protected under RMK or SMK is a more sensitive matter than that of DES keys, and that a more granular control is required. It is also expected that the number of private keys used in the server environment is quite small—perhaps only one or two. To provide the appropriate granularity, special PKSC controls were provided to permit PKA objects to be exported and imported individually with multiple control. These controls also provide for PKA-object transfer between two cryptographic systems. When the SMK or RMK is updated, affected PKA objects become invalid, and a copy of these objects must be imported by means of the import-PKA-object PKSC function.

Object transfer
For backup and recovery, a means must be provided for object distribution with tight control. An object can be exported from one system and imported to another by means of the export-PKA-object and import-PKA-object PKSC functions. Public auditing and multiple control ensures that only designated objects are moved. These PKSC functions reencipher the object protection key from being encrypted under RMK or SMK to being encrypted under the 320-bit public transport key, or vice versa. Since encapsulation is based on the clear value of the object protection key, object integrity can be verified by both the sender and the receiver.

DES-based facility

DES-based functions use the DMK and are profile-controlled. This section describes only the new functions and features. Separate profile bits are assigned for functions that reset domain, for each type of verification pattern supported for DES-based keys, for hash values on each of the master keys, for zeroizing the old, new, and current DMK, and for clear master-key entry of each of the three master keys in each domain.

The enhancements which are new in the SCM are summarized as follows.

Improved encryption
New encryption functions include TCBC and TECB modes of triple-DES (TDES) cipher.

Improved MAC
New MAC functions include a two-key MAC defined in ANSI X9.19 [11] and a three-key MAC, which is the last block of ciphertext using three-key TCBC-mode TDES encryption.

Improved key types
ICRF had ten key types. The SCM adds additional variability in the control vector (CV), resulting in more flexibility but also reducing the total number of key types. Control vector is a means of implementing the key-separation concept, which requires that keys be used only in the prescribed operations according to the key type. A detailed description of this concept and some implementations are provided in [1, 2].

The MAC generation key and MAC verification key types are merged into a single MAC key type with two CV bits: a MAC generation bit and a MAC verification bit. Similarly, the PIN generation key and PIN verification key types are merged into a single PIN derivation type with two CV bits: a PIN generation bit and a PIN verification bit. One new key type has been added—a data-encryption message-authentication (DEMA) key type with four CV bits: encipher, decipher, MAC generation, and MAC

verification. CV bits were added to the exporter KEK and importer KEK to control whether the key could participate in key generation, key translation, and key import and export functions.

With this change, the SCM includes a larger number of the control vectors supported by IBM 4753.

Improved control of key state change
The most significant CV change is the key distribution control, which can be set to prohibit a key encrypted under DMK from being exported. In ICRF, any key in the operational state could be converted to the exportable state. A CV bit has been added to all key types, except the data-encrypting key type, to control whether a key in the operational form can be exported. This bit cannot be applied to the data-encrypting key type because the CV for this key is all zeros. The data-encryption message-authentication key provides this control.

- *Processes*

This section describes several processes which use the functions described earlier in the paper.

Initialization process
When an SCM is shipped with the enterprise server to the customer, when the primary power is turned on after both power sources have been removed, or when an SCM reset is performed, the SCM is placed in the not-initialized state. In this state, all security-relevant functions are disabled, but non-security-relevant functions are enabled because they are not subject to the control of U.S. export regulations. The initialization process is performed to change the SCM state from the not-initialized state to the initialized state, after which the customization process can begin.

The initialization process uses a set of initialization functions to establish a set of legitimate cryptographic configuration controls (CCC). The performance of initialization functions is similar to that of multiple-signature PKSC functions, except that signatures for initialization functions are provided by IBM. The initialization functions are currently delivered to the customer on an enablement diskette. Without the enablement diskette, security-relevant functions cannot be enabled and, thus, can be exported and be included in all machines. Only the enablement diskette must be export-controlled. Note that an SCM test mode can be established by loading a particular CCC. In this mode, the pseudorandom number generator can be placed and maintained in a deterministic state to produce predictable outcomes. This mode is used during the manufacturing process for testing and is disabled for all SCMs by initialization functions.

Customization process
After the SCM has been initialized, security officers can install customer security-related data, including identification data, authorization controls, and master keys. In the interactive secure workstation environment, public identification data and authorization controls, including security officer identification, specifications for SRA and PKSC authorization masks, and profile registers should be installed first. This can be done as a single step and then authenticated. Customer secret master keys are installed in the SCM only after identification data and authorization controls have been properly established. In other environments, master keys may be installed before final controls in the profile register or signature requirement array are set.

Master keys and initial DES keys are installed using split knowledge. That is, the key is split into multiple key parts, and each key part is installed separately. Additionally, each key part may be encrypted under a transport key while being transmitted from a secure workstation to the SCM. The load-DES-key-or-DMK function is provided to install key parts of a DES key or DMK using the 128-bit basic transport key; the load-RMK-or-SMK function is provided to install key parts of the RMK or SMK using the 320-bit public transport key.

After this process is completed, the SCM is ready to perform all enabled functions for cryptographic applications.

Master-key transfer
The PKSC extract-and-encrypt functions permit the master keys of one system to be securely copied into another. For backup or performance, this master-key transfer capability simplifies cryptographic system management for establishing the same image on multiple systems. By using the DH protocol, a common transport key can be established inside the secure boundary of each of two SCMs. The source information used to establish this can be queried. This permits an auditing program to verify that the proper DH parameters have been used and thus that the transport key is not known outside of the two SCMs. This permits a master key to be encrypted under the transport key extracted from one SCM, transferred to the other SCM, decrypted in the second SCM, and installed with no exposure to compromise. For DMK transfer, the 128-bit basic transport key is used; for RMK or SMK transfer, the 320-bit public transport key is used.

A similar PKSC process is used to import or export PKA objects as previously described in the subsection on object transfer.

Pseudorandom number generator (PRNG)

The overall security of the SCM depends on a secure and reliable random number generator. Implementation of a

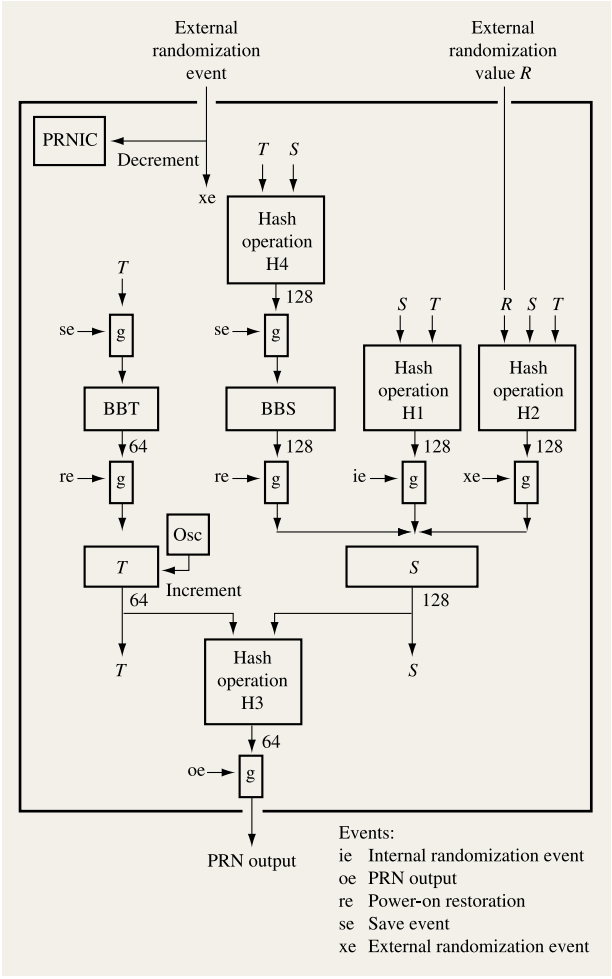

Figure 4

Pseudorandom number generator (PRNG) overview.

true random number generator, thermal noise or Geiger counter, for example, was out of the question; such circuits are not compatible with the CMOS circuit design and do not meet the requirements for reliability, testing, and diagnostics for this product.

Much special attention was devoted to the design of the pseudorandom number generator (PRNG). There is a broad spectrum of uses for pseudorandom numbers (PRNs) by the SCM. Random numbers are generated and used internally for the crypto module signature sequence number and the transaction sequence numbers (both are 128 bits and public) and for Diffie–Hellman values (1024 bits and secret). Random numbers are generated and provided to applications in a protected form for use as DES keys (64, 128, and 192 bits); DSS private keys and message secret numbers (both are 160 bits); object protection keys (320 bits); confounders (64–192 bits); and padding (8–192 bits). Random numbers are also provided to applications in the clear for additional uses such as initial chaining values and the generation of prime numbers.

Such diversified uses for PRNs indicated that multiple PRNGs customized to individual requirements were impractical; rather, a single very strong PRNG was required.

Since random numbers are used during the initialization process, the PRNG must be initialized before the remaining stages of the process can be performed. Thus, the PRNG had to be self-initializing rather than being dependent on other means for the installation of an initial seed. This requirement differs from the environment assumed by most standards, where initialization is outside the scope of the standard.

The generate pseudorandom number function provides a 64-bit pseudorandom number in the clear. In ICRF, this function was not available until after the master key had been installed. In the SCM design, the PRNG is the first secure portion of the machine to become available from the not-initialized state. Since the output of the PRNG is designed to be unbreakable, the generate pseudorandom number function is not profile-controlled and is available whenever the PRNG is initialized.

- *Design criteria*

The design criteria for the PRNG were as follows:

1. *Randomness* The output of the pseudorandom number generator must appear to be a string of random independent bits with no bias. By "no bias," it is meant that the values zero and one are equally probable for each bit. By "independent," it is meant that given any number of output bits, it is computationally impractical to compute (or predict) the value of any unknown bit.
2. *Cryptographic strength* Since the output bits of a PRNG are not truly independent, we define the "work factor" of a PRNG as the order of magnitude of the most efficient algorithm capable of computing the value of unknown bits. For any particular use of the system, the PRNG should not be the weakest link. All secret values in the SCM have at least 128 bits, thus implying a cryptographic strength of 2^{128} work factor; the PRNG must have a matching strength.

   In some PRNG designs, the work factor to compute the value of future bits may be different from that to compute the value of previous bits. These are referred to as the forward and backward work factors, respectively. It should be expected that the use of a hash operation in the feedback path may result in the backward work factor being larger than the forward work factor. It should also be noted that for some types

of compromises, the backward work factor is the more important of the two. This would be the case, for example, if the action of compromising were to cause the unit to become no longer usable or placed in a tamper state.

3. *Secure initialization and re-initialization* The PRNG must contain adequate entropy before it becomes operational. Because of the tightly controlled manufacturing processes used to build the SCM, unless special action is taken, the PRNG will have a tendency to repeat the same sequence each time it is initialized. The probability of causing the same PRNG (or different PRNGs) to return to the same state by repeatedly forcing initializations is required to be no larger in magnitude than the probability of repetition of the state in normal operation.
4. *Recovery from compromise* Each value output by the PRNG gives away information and reduces the entropy. Also, although the PRNG has no known design weaknesses, it is not a good security design principle to assume that no weaknesses exist. To ensure long-term security, and to minimize exposure from compromises due to unknown reasons, a mechanism to add entropy is needed.
5. *Prevent backtracking from compromise* The SCM is tamper-resistant, and it is highly unlikely that a perpetrator could compromise the PRNG state. But, in the unlikely event that the PRNG state should be compromised, it is likely that the SCM would no longer be operational. Thus, the backward tracking problem is more serious but also much more easily solved.
6. *Reliability and testability* Hardware reliability in the S/390 Parallel Enterprise Server* environment is extremely important. To meet these objectives requires extensive testing. Testing of the design must be performed during the development process, and testing of each individual product must be performed during the manufacturing process and also in the field. Testing presents unique problems because it is normally based on predictability and repeatability, which are contrary to the PRNG output in normal operation.
7. *Restoration after power-off* Initialization of the PRNG is a sensitive, complex, and time-consuming operation. Once it has been initialized, subsequent initialization should not be necessary when power is dropped and restored. This includes the case in which power is unexpectedly lost without time to react as the power is going down. Thus, information must be saved in advance in preparation for a potential restoration.

- *PRNG operation*

Figure 4 shows an overview of the implementation of the PRNG. **Figures 5** through **10** show more detail for specific situations.

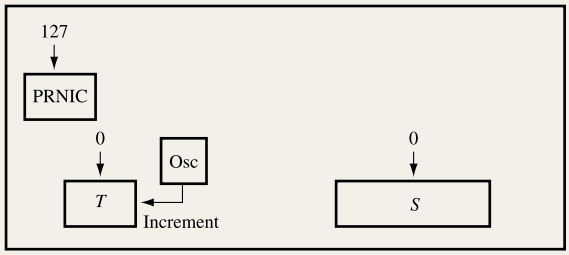

Figure 5

PRNG SCM reset.

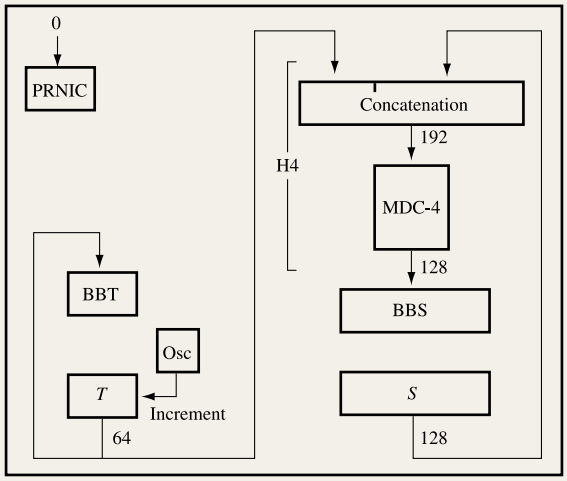

Figure 6

PRNG save event.

PRNG working registers

The PRNG includes a real-time counter (*T*), a randomization state (*S*), and a 7-bit pseudorandom number initialization count (PRNIC). The PRNIC indicates the remaining number of external randomization events required before the PRNG is considered to be initialized. The PRNIC is initially set to 127, and the PRNG cannot be used for generating PRNs until the PRNIC is zero. When the PRNIC is nonzero, it is decremented by one each time an external randomization event occurs.

The 128-bit randomization state *S* provides the basic strength of the PRNG. When initialization is complete, *S* contains secret information.

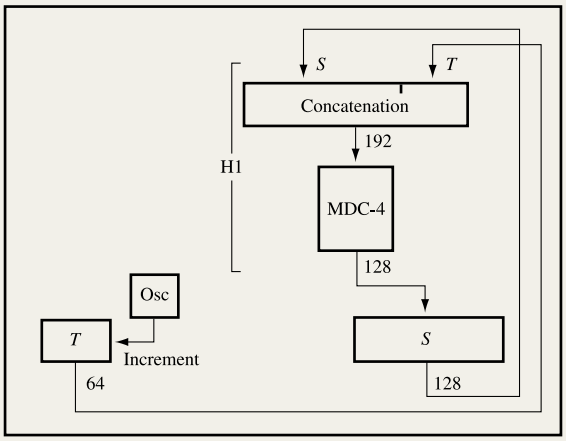

Figure 7

PRNG internal randomization event.

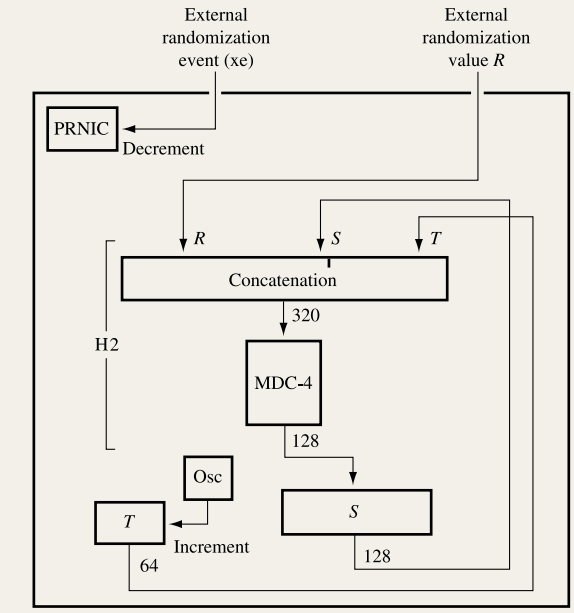

Figure 8

PRNG external randomization event.

The 64-bit real-time counter T is incremented continuously at the fastest rate possible for the machine. T provides three important factors:

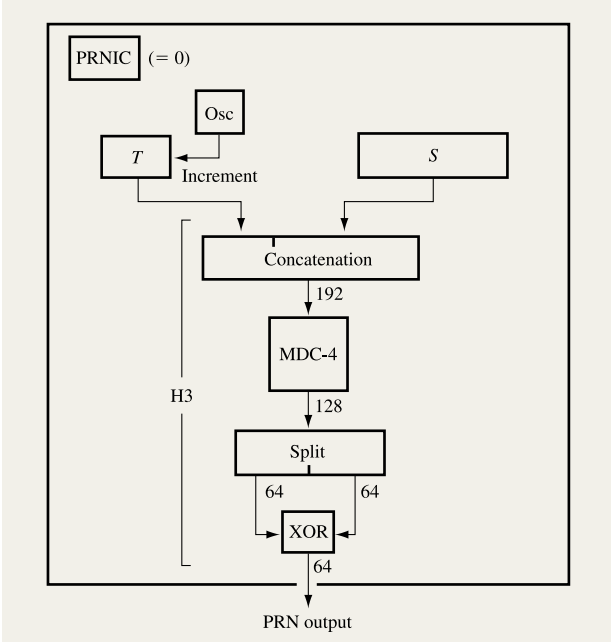

Figure 9

PRNG output.

1. T is used as the basis for providing entropy. This is based on the assumption that although the value of T is known approximately, it cannot be known to the exact cycle. The amount of entropy added by T depends on how accurately the difference between several values of T can be known or controlled. This value is called "jitter" and should be distinguished from two other values associated with T, "drift" (variation in stepping rate of T) and "epoch" (time at which T was zero).
2. The use of T ensures that the output of the PRNG does not have a short-term cycle of repetition. (On the G5 processor, the time for T to wrap around is several hundred years.)
3. When more than 64 bits of PRNs are required with no intervening idle period, the changing value of T provides different output information for each execution.

The reliability of T is crucial to the PRNG security. Error-checking and correction (ECC) code is used not only on the counter data but also on all control signals. Additional independent checking is also continuously performed to ensure that the counter is incrementing; otherwise, the checking logic will shut down the entire SCM.

PRNG battery-backup registers

The SCM is powered by two sources, primary power and battery-backup power. Most of the logic circuitry and active registers in the SCM are designed for speed at the expense of additional power consumption and are powered only by the primary power source. Registers *T*, *S*, and PRNIC are of this type, and information in these registers is lost when primary power is removed.

Associated with registers *T* and *S* are registers BBT and BBS (battery-backup registers). BBT and BBS are powered by both the primary power and battery-backup power sources. This storage operates at lower speed, but has lower power consumption, and the information is maintained when primary power is removed. (There is also battery-backup information, not shown in the figures, indicating the initialization state of the PRNG.) The battery-backup registers maintain sufficient information that after the PRNG has been initialized, initialization is not required during subsequent power-on sequences.

SCM reset

It is assumed that the state of the registers in the SCM after power-on is not random, but rather may be quite repeatable. Thus, it is necessary to introduce randomness into the PRNG in response to some external event. Since randomness is introduced in this way, SCM reset is defined to set all registers to a known state, thus permitting validation of error-checking and correction codes in these registers. SCM reset causes *T* and *S* to be set to zero, and PRNIC to be set to 127.

Internal randomization event

During normal operation, the randomization state *S* may be updated as the result of either of two randomization events. (Updates of registers are shown in Figure 4 by means of a signal, such as "ie," controlling a gate, shown as "g" in the figure.)

The internal randomization event (ie) causes *S* to be updated from the output of hash operation H1. Whenever the SCM is not performing other activities, a continuous sequence of internal randomization events is performed until some other activity is requested, at which point the internal randomization event in process is canceled. The sequence of internal randomization events performed without any other activity is called an "idle window." There are no limitations on the size of an idle window, and it may be small enough that no internal randomization events occur. Hash operation H1 is the MDC-4 of *S* and *T*.

The internal randomization event adds entropy on a continuing basis without requiring any special action on the part of the program.

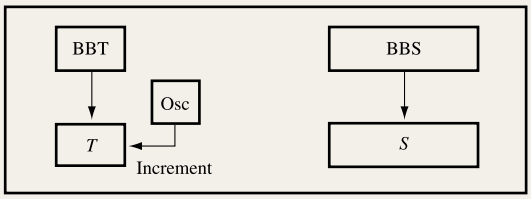

Figure 10

PRNG power-on restoration.

External randomization event

As part of the execution of any PKSC query function, an external randomization event (xe) is performed which causes *S* to be updated from the output of hash operation H2. Each PKSC query function includes a 128-bit value *R*, which is used as a randomization value. Hash operation H2 is the MDC-4 of *R*, *S*, and *T*.

Execution of any PKSC query function continues to cause the external randomization event to be performed even after the PRNIC is zero. This permits the cautious user to add additional entropy and perhaps randomization to the PRNG at any time. In this context, we view "entropy" as the amount of unpredictability added to the PRNG as a result of jitter, assuming that an observer knows the randomization value along with the epoch and drift values of *T*. "Randomization" is the complete reseeding accomplished by this action if it can be accomplished without the observer's knowledge of the randomization value.

Initialization complete

When 127 external randomization events have occurred and the PRNIC is set to zero, a save event occurs. The save event provides information for restoration after power loss; it causes the output of hash operation H4 to be placed in BBS and the current value of *T* to be placed in BBT. Hash operation H4 is the MDC-4 of *T* and *S*.

PRN output

PRN output is produced 64 bits at a time using hash operation H3. H3 produces a 128-bit intermediate value using the MDC-4 of *T* and *S*. The left 64 bits of this intermediate value are exclusive-ORed with the right 64 bits to produce a 64-bit result. When more than 64 bits are required during execution of certain cryptographic operations, H3 is invoked multiple times, with no intervening update of *S*. Each of these outputs is different because *T* is incremented at approximately a 100-MHz rate, while it takes approximately one microsecond to produce a 64-bit PRN output.

Power-on reset

The action taken by power-on reset depends on whether the SCM has been previously initialized. If not, power-on reset causes the same action as SCM reset. If the SCM has been previously initialized, power-on reset causes T and S to be restored from the values in BBT and BBS; then, after several cycles, a save event occurs.

Periodic save

During normal operation, a save event may be performed periodically. This keeps the value of BBT more or less current and avoids the possibility of restoring T to a very old value.

Test facilities

The design of the PRNG includes special diagnostic facilities which permit extensive testing of the PRNG registers, data path, and controls. These facilities permit each of the hash and feedback operations associated with the PRNG to operate independently and in a completely deterministic manner. They also permit the counter T to be stopped or single-stepped. These diagnostic facilities are operational only when the test mode bit in the CCC is a 1. The program can ensure that the SCM is not in test mode by means of the query module information PKSC function.

- *Security analysis of PRNG*

If we start with the simplifying assumption that T can be known exactly, there is a relatively simple algorithm to compute unknown bits on the PRNG. If several output values, with no intervening randomization event, are available and T is known exactly for each of these values, S can be found by means of an exhaustive search. The work factor for this search is 2^{128}. From this value for S, and exact values for T, all outputs generated using this S can be computed. Additionally, if the exact value of T is known for future randomization events, the future values of S can also be computed.

Given exact values for T, the same process can be extended with the same work factor, even if the known values are separated by intervening randomization events. In the same way, given exact values for T, the process can be extended to compute values previous to the known values by picking an earlier starting point to begin the exhaustive search.

The above approach can be extended to the case for which T is known only approximately by including all possible values for T as part of the exhaustive search. The unknown value of the epoch of T adds a simple increase to the work factor. It is assumed that the effect of drift is insignificant in comparison to the unknown value of the epoch and can be ignored. Jitter in the value of T associated with known output values results in a simple increase in the work factor, and the exact value of T for these events will be revealed as a result of the search. Jitter in the value of T associated with unknown output values results in uncertainty in the unknown values, but does not increase the work factor. Jitter in the value of T associated with randomization events has a compound effect on the work factor; not only must additional values for T be included in the exhaustive search, but the effective size of S over time is increased, thus increasing the space that must be searched.

The amount of entropy added by T depends on the jitter. It is assumed that the jitter for an external randomization event adds at least one additional bit of entropy. Thus, the fact that 127 external randomization events must be performed before the PRNG is initialized increases the entropy to the maximum amount containable in S.

After initialization is complete, both external and internal randomization events continue to add entropy. For internal randomization events, the timing between events within a single idle window may be repeatable, but the unpredictability of the timing for the entire window adds at least one additional bit of entropy.

While measurements and controls accurate to within one or two cycles would be possible in a laboratory environment, it is unrealistic to expect this type of accuracy in a production environment. On the G5 processor, for example, T steps at a rate of the order of 100 MHz, the stepping rate of the time-of-day clock is 64 MHz, and the time to perform one internal randomization event or generate a 64-bit PRN is of the order of 1 μs. But the path length for the operating system to send an application request to the SCM is several microseconds, long enough to permit several internal randomization events between requests. Thus, when no asynchronous operations are in progress, at least one bit of entropy is added for each synchronous request, but as viewed by the application program, the effective jitter is much larger. Asynchronous operations may reduce the number of internal randomization events, but only at the expense of increasing the effective jitter, since the execution time of PKA and PKSC functions is measured in milliseconds.

Each of the operations H1, H2, H3, and H4 is the cryptographically strong MDC-4 hash algorithm. The main difference between the operations is the order in which S, T, and R are fed into the MDC-4 operation. The use of strong hash algorithms rather than an encryption algorithm increases the backward work factor in the case of certain compromises.

In the restoration of T and S from BBT and BBS, no attempt is made to eliminate repetition of values in T. Instead, S is updated using a hash operation different from those used to update S in the normal process. This reduces the probability of repetition of S to the same order of magnitude as the strength of the PRNG.

In summary, the PRNG security has a security level that is higher than a work factor of 2^{128}. From a given output, it takes a work factor of about 2^{128} to derive the PRNG state. However, even if the state could be compromised, there would be no long-term effect, because entropy is continually being added.

Recently, Keisey et al. [12] studied attacks on several PRNGs and suggested six design guidelines. Our PRNG design was completed four years before that analysis appeared, but it complies with all suggested guidelines.

Conclusions

As has been shown in this paper, the SCM is a high-performance, highly secure product designed for the server cryptographic environment. It was designed to be used in conjunction with several secure workstations operated by security officers. The SCM provides an optimum balance between performance and flexibility by implementing high-usage cryptographic operations with high security and high performance, while permitting less frequent operations to be performed in the associated secure workstations.

As we have shown in this paper, for a hardware implementation the time to market after design "freeze" presents a major difficulty in currency in the standards area. Standards will continue to become more important in the future. Keeping track of existing and emerging standards requires much effort; interpretation and application are even more difficult tasks.

Much has been learned during the four and one-half years it took to develop, design, test, and ship the SCM. If we were starting over again, we would use DSS rather than RSA for the PKSC digital signatures; DSS has several advantages over RSA:

1. With RSA, the difficult mathematical operations (choosing two strong primes) must be performed secretly; this creates an exposure to lack of auditing. With DSS, the difficult mathematical operations (choosing a strong prime and a generator) need not be performed secretly and can be publicly audited.
2. With RSA, key generation is too complex to be implemented in pure hardware; thus, this operation cannot be provided inside the SCM. With DSS, key generation is a simple operation of picking a 160-bit pseudorandom secret value and can be performed by the SCM inside the secure boundary.
3. RSA requires formatting, adding an additional step in the signature generation and verification processes. With DSS, SHA-1 the hash value of the message can be used directly.
4. With RSA, the SCM private key had to be generated during the manufacturing process and burned in at the factory. If DSS were used, the private key could have been generated in the field and placed in tamper-responsive volatile storage.

*Trademark or registered trademark of International Business Machines Corporation.

References

1. P. C. Yeh and R. M. Smith, Sr., "ESA/390 Integrated Cryptographic Facility: An Overview," *IBM Syst. J.* **30**, No. 2, 192–205 (1991).
2. R. M. Smith, Sr. and P. C. Yeh, "Integrated Cryptographic Facility of the Enterprise Systems Architecture/390: Design Considerations," *IBM J. Res. Develop*. **36**, No. 4, 683–693 (1992).
3. *Security Requirements for Cryptographic Modules*, Federal Information Processing Standards Publication 140-1, National Institute of Standards and Technology, Washington, DC, January 11, 1994.
4. *Digital Signature Standard (DSS)*, Federal Information Processing Standards Publication 186-1, National Institute of Standards and Technology, Washington, DC, December 15, 1998.
5. *Telecommunications: General Security Requirements for Equipment Using the Data Encryption Standard*, Federal Information Processing Standards Publication 1027, National Institute of Standards and Technology, Washington, DC, April 14, 1982.
6. *Secure Hash Standard*, Federal Information Processing Standards Publication 180-1, National Institute of Standards and Technology, Washington, DC, April 17, 1995.
7. IBM Corporation, *Common Cryptographic Architecture: Cryptographic Application Programming Interface Reference*, Order No. SC40-1675; available through IBM branch offices.
8. *Digital Signature Scheme Giving Message Recovery*, ISO/IEC 9796, International Standards Organization/ International Electrotechnical Commission, Geneva, Switzerland, July 1991.
9. *American National Standard for Financial Services*, *Triple Data Encryption Algorithm, Modes of Operation*, ANSI Standard No. X9.52-1998, American National Standards Institute, Washington, DC, 1998.
10. *American National Standard for Financial Services*, *Data Encryption Algorithm, Modes of Operation*, ANSI Standard No. X3.106-1983, American National Standards Institute, Washington, DC, 1983.
11. *American National Standard for Financial Institution Message Authentication (Retail)*, ANSI Standard No. X9.19-1986, American National Standards Institute, Washington, DC, 1986.
12. John Keisey, Bruce Schneier, David Wagner, and Chris Hall, "Cryptanalytic Attacks on Pseudorandom Number Generator," *Fast Software Encryption, Fifth International Workshop Proceedings*, Springer-Verlag, New York, March 1998, pp. 168–188.

Received November 4, 1998; accepted for publication August 5, 1999

Phil C. Yeh *IBM System/390 Division, 522 South Road, Poughkeepsie, New York 12601 (pyeh@us.ibm.com).* Dr. Yeh is a Senior Engineer in the Systems Architecture Department of the IBM Mid-Hudson Valley Development Laboratory in Poughkeepsie, New York. He received an M.S. degree in computer science and a Ph.D. degree in electrical engineering from the University of Illinois at Urbana–Champaign in 1977 and 1981, respectively. In 1981, he joined IBM at Poughkeepsie, where he has worked on several architecture assignments.

Ronald M. Smith, Sr. *IBM System/390 Division, 522 South Road, Poughkeepsie, New York 12601 (rmsmith1@us.ibm.com).* Mr. Smith is a Senior Technical Staff Member in the Systems Architecture Department of the IBM Mid-Hudson Valley Development Laboratory in Poughkeepsie. He received his B.E.E. degree in electrical engineering from Ohio State University in 1957 and joined IBM at the Endicott Laboratory the same year, moving to Poughkeepsie in 1961. He worked on assignments in circuit design, central processor design, and programming before joining the Systems Architecture Department in 1966.