Integrated Cryptographic Facility of the Enterprise Systems Architecture/390: Design considerations

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

This paper reviews the considerations that shaped the design of the Enterprise Systems Architecture/390™ Integrated Cryptographic Facility. It describes design issues, alternatives, and decisions, and it provides the rationale behind some of the decisions. Issues related to performance, security, usability, and availability are covered.

Introduction

Protection of information is one of the most important issues facing computer users today. This is because of the increasing number of applications involving network communications and distributed systems, the growing amount of information processed by computers, the increasing dependence on shared databases, and the increasing number of applications critical for the existence and success of enterprises. Cryptography is an effective tool for providing data security (preventing unauthorized access to data), data integrity (ensuring the accuracy of data), and authentication (verifying the identity of a user or sender). The Enterprise Systems Architecture/390™ (ESA/390™) Integrated Cryptographic Facility (ICRF) was designed to provide the cryptographic functions necessary for users of large systems to protect their data.

The ICRF is a DEA-based [1] cryptographic architecture that is an extension to the IBM ESA/390 architecture. The facility provides cryptographic functions[1] for performing data encryption and decryption, handling message-authentication codes (MACs) [2] and personal-identification numbers (PINs) [3], and managing cryptographic keys. All cryptographic instructions are performed synchronously with central processing unit (CPU) instruction execution.

Implementation of the ICRF architecture on the water-cooled Enterprise System/9000™ (ES/9000™) computer

[1] In this paper, the term *cryptographic function* is used to refer to the capability provided by the ICRF to perform a particular cryptographic action. The term *cryptographic instruction* is used to refer to the interface provided in the CPU to permit the program to invoke cryptographic functions. The term *cryptographic operation* is used to refer to the execution of cryptographic functions or instructions; it is also used in a more general sense to refer to the actions in a cryptographic environment.

models provides a hardware cryptographic feature within a CPU that is capable of high performance in financial-transaction and large-file-encryption applications. The crypto unit[2] is physically implemented in a tamper-resistant enclosure. Secret information is always kept inside the enclosure. A manual-control panel is provided for manual entry of cryptographic keys, clearing the contents of internal registers containing secret quantities, enabling or disabling the facility, and controlling the use of certain special cryptographic functions.

Software support for the ICRF is provided by the IBM Integrated Cryptographic Service Facility (ICSF/MVS). This program provides an "interface" to application programs that is upward compatible with the interface provided by the Cryptographic Unit Support Program (CUSP)[3] and the Programmed Cryptographic Facility (PCF)[4].

A detailed description of the product objectives, the specific key-management scheme, and the structure of the ICRF was presented in a previous paper [4]. That paper also reviewed major functions and summarized the physical security provided by the first ICRF implementation.

This paper presents the considerations that shaped the design of the ICRF. It first compares the ICRF with the ESA/390 Vector Facility [5], from a system-design viewpoint, to highlight unusual design concerns caused by the ICRF. It then describes design issues, alternatives, and decisions, and it provides the rationale behind some of the decisions. Issues related to performance, security, usability, and availability are covered.

Comparison with vector facility

In the first implementation of the ICRF, because of the physical security and hard-wired logic, the hardware associated with the crypto unit took so much space that the number of crypto units that could be installed was limited. On some ES/9000 computer systems with up to three CPUs, only one crypto unit could be installed. On the largest systems, with up to six CPUs, only two could be installed. In addition, either a crypto unit or a vector unit,[5] but not both, could be installed on any particular CPU.

ICRF and vector are the only two optional CPU facilities that are permitted to be installed on some (but not necessarily all) CPUs in a multiprocessing system. The type of configuration in which the CPUs are not equivalent poses several unusual problems for the control program, and solutions for these problems must be addressed in the architecture. This led to a starting point in the design process in which an attempt was made to use the same type of solutions for ICRF as had been used for the vector facility. In many cases, the vector solutions also worked for ICRF; in other cases, they did not.

This section presents some of the similarities and differences between these two facilities. The comparison accentuates many complexities and implications for system design caused by the unusual characteristics of the ICRF.

- *Similarities to vector facility*

The ICRF and the vector facility have the following similarities:

- Both facilities have a requirement that the control program be able to disable any unit so that the application program cannot use it. When a vector unit is disabled, the control program does not have to save and restore the contents of the registers in the unit. This saves a significant amount of time when application tasks that do not use the facility are being dispatched. A crypto unit may have to be disabled because, for example, the master key (described below) has not been loaded, or the unit is in the error state. This requirement is met by means of a control bit in each CPU that enables or disables the use of the unit. When the unit is disabled, any attempt to execute an instruction associated with the unit results in a program exception which indicates that the unit is unavailable.
- Both facilities have a requirement for a special indication of the absence of the facility. For most facilities, an attempt to execute an instruction that is part of an uninstalled facility results in a program interruption that indicates an "operation exception." The action taken by the control program when an application task encounters an operation exception is to terminate the task. For the vector facility and ICRF, when the facility is installed on some but not all of the CPUs in the system, this is not the desired action. Rather, a task attempting to use a particular facility should be rescheduled from a CPU without the facility to a CPU with it. To assist the program in identifying this case, new program interruption codes were defined for both the vector facility and ICRF. When an application task attempts to execute an instruction associated with the vector facility on a CPU on which a vector unit is not installed, the vector-operation exception is reported. Similarly, when an application task attempts to execute an instruction associated with the ICRF on a CPU on which a crypto unit is not installed, the crypto-operation exception is reported.
- Both facilities must process data in very large quantities. This can result in excessively long execution time and

[2] In this paper, the term *crypto unit* is used to refer to a physical implementation of the ICRF architecture.
[3] The IBM Cryptographic Unit Support Program (CUSP) is the host software for supporting the IBM 3848 cryptographic unit, which is a channel-attached device.
[4] The IBM Programmed Cryptographic Facility (PCF) is software that simulates the 3848 functions and provides the CUSP application program interface.
[5] In this paper, the term *vector unit* is used to refer to a physical implementation of the vector facility architecture.

requires some sort of "sectioning" to divide the operation into smaller units. The considerations associated with sectioning are discussed in a later section of this paper.

- *Differences from vector facility*

There are many differences between the ICRF and the vector facility in handling certain functions and requirements. Among these functions and requirements described in the following sections are interchangeability, task scheduling, security restriction, and exception handling.

Interchangeability

All vector units in a system are identical and can be used interchangeably by all programs. The control program takes advantage of this characteristic to balance the workload on a multiprocessing system. It is very desirable to perform the same type of load balancing for programs using crypto units. Although crypto units in a system can normally be used interchangeably, there are some infrequent situations in which this is not the case.

Following are the two categories of situations in which CPU affinity, the association of a task with a specific physical CPU, is required:

- Some cryptographic support tasks require CPU affinity. These include manual key entry (because the process uses the manual-control panel, and each crypto unit has a separate panel) and error handling (because of the very nature of the process).
- The states of the crypto units do not appear to be identical to the program. Following are some examples: manual controls have disabled one crypto unit while the other is enabled; master-key entry has been completed on one crypto unit but not yet started on the other; and physical tampering has caused the master-key registers to be cleared in one crypto unit but not in the other.

When the crypto units are not interchangeable, it may be desirable to permit application programs to use one of the crypto units but not the other. To do this, the control program must have a simple means to determine whether the crypto units are interchangeable. Since the contents of most registers of crypto units must be kept secret, a means was provided to verify interchangeability without disclosing the contents of secret registers.

Even though a task may have a requirement to access a particular crypto unit, the task may execute many thousands of noncryptographic instructions between actual uses of the facility. By disabling the crypto unit (using the crypto-control bit), the control program can dispatch such a task on any CPU in the system and then reschedule the task on the proper CPU when the task attempts to issue a cryptographic instruction. The requirement for CPU affinity and the situations in which the crypto units are not interchangeable create additional considerations and complexities for Processor Resource/Systems Manager™ (PR/SM™)[6] and VM support, as explained in the section on PR/SM support below.

Task scheduling

The expected workload characteristics of the ICRF and the vector facility are quite different. Normally, vector instructions are executed in a batch environment and are used by only a small number of tasks; when used, vector instructions normally continue to be used over an extended period of time and represent a significant fraction of the instructions executed. Cryptographic instructions, on the other hand, are expected to be used primarily in a financial-transaction environment and to be used by most of the tasks in the system; even when used, cryptographic instructions represent an extremely small fraction of the total number of instructions executed.

As mentioned previously, both the vector facility and ICRF provide special program exceptions to indicate to the control program that an application task has attempted to execute an instruction associated with the facility on a CPU that does not have the facility installed. As a result of this indication, the control program places the task in a list associated with that facility (a "vector-scheduling" list or a "crypto-scheduling" list). The action for placing a task in the list is basically the same for the two facilities. However, the action for removing the task from the list is quite different.

When an application task first issues a vector instruction, the task is placed in the vector-scheduling list and left there for several time slices. Tasks on the vector-scheduling list are allocated time slices only on CPUs with vector units. The tasks in the vector-scheduling list are periodically tested to see whether they are still using vector instructions. If several time slices pass with no vector use, a task is marked as not using vectors, and the task is placed in the "non-vector-scheduling" list. To perform this type of action, the control program must be able to test whether the task has used vectors during the past several time slices. This information is provided by the "vector-activity-count" register.

The above-mentioned process for removal of tasks from the vector-scheduling list cannot be applied to the removal of tasks from the crypto-scheduling list. This is because expected workload characteristics indicate that nearly all tasks in crypto environments will be using crypto units. Unless some special action is taken, the scheduling list for those CPUs with a crypto unit will be overloaded, while

[6] PR/SM is a hardware feature that allows the resources of a machine (multiprocessor or uniprocessor system) to be shared dynamically among multiple, independent "partitions." Each partition can run a system control program, and all partitions can operate simultaneously [6].

those CPUs without a crypto unit may have a very light workload.

It is expected that customers will install as many vector units in their systems (up to one per CPU) as are required to meet the peak workload requirements. As mentioned previously, early implementations of the ICRF can provide a maximum of two crypto units, even in systems with up to six CPUs. More sophisticated scheduling algorithms involving smaller time slices are needed to efficiently balance CPU workload in this environment.

The fact that the vector facility is an optional feature and is typically used by a small fraction of tasks leads to the requirement to provide vector-usage information for billing purposes. The vector-activity count provides this information. Since cryptographic use is expected to be more pervasive, no corresponding cryptographic-activity count was deemed necessary.

Security restriction

There is no security restriction on the use of a vector unit. Any program that attempts to use the facility is given access to it. Most vector instructions are valid in the problem state (the state of the CPU when an application program is running).

Stringent security restrictions exist regarding the use of the crypto unit, and special authorization must be provided to control access to the facility. In most cases, this authorization is at a very fine degree of resolution; i.e., tasks are given authority to perform certain cryptographic functions but not others. In ESA/390, this type of control is normally provided by means of an access-control program that operates in the supervisor state (the state of the CPU when the control program is running). To facilitate this type of support, all cryptographic instructions are privileged (that is, they are valid only in the supervisor state).

Exception handling

Once a vector unit is made available by the control program, the only situation that may cause the vector unit to become unusable is a hardware failure. This is reported simply as a "machine-check" interruption.

For a crypto unit, many unusual situations must be reported. These include indications of hardware malfunction or improper setting of the manual controls. There are also indications that do not represent errors at all. For example, each crypto unit has its own tamper-detection circuitry which, upon detecting tamper, can cause the facility to become unusable; this situation must be reported to the control program as a special tamper indication.

These unusual situations are reported by the hardware as part of the cryptographic instruction execution by setting status bits in the CPU. In a multiprocessing environment, the cryptographic control program is normally scheduled to run on any of the CPUs in the multiprocessing system. Thus, when it tests the result of a cryptographic instruction, the cryptographic control program may be running on a different CPU from the one it was running on when it executed the instruction. In order for the program to determine which CPU reported an unusual situation, information in the status word includes the address of the CPU that executed the instruction.

Design considerations

The stringent requirement for security affected other design considerations, such as performance, flexibility, usability, and recovery, in a special way, resulting in many concerns and difficult trade-offs. Major design considerations are described in this section, and the rationale for each decision is discussed. The design considerations are divided into two categories: system-design considerations and cryptographic-design considerations.

- *System-design considerations*

CPU-integrated approach

Most commercially available hardware cryptographic products are implemented as I/O devices attached to I/O channels. In fact, the original goal of this project was to develop a follow-on product to the IBM 3848 Cryptographic Unit [7], which is also a channel-attached device. However, the performance objective of providing up to 1000 transactions per second[7] made the CPU-integrated approach a necessity.

Register operands

The execution time of many cryptographic operations is relatively long compared with the execution time of a typical ESA/390 computer instruction. This long execution time creates an additional design concern for a multiprocessing system.

Execution of certain ESA/390 instructions, such as INVALIDATE PAGE TABLE ENTRY and SET STORAGE KEY EXTENDED, requires that all CPUs in the multiprocessing system observe the effect of these instructions before the executing CPU can continue. This is implemented by means of special hardware interlocks. A CPU executing one of these instructions must wait for a response from all other CPUs in the system before completing the instruction execution. This response indicates that all storage accesses for instructions already in progress have been completed. Normally, each CPU signals the response at the completion of its current

[7] "Transaction" here means an "IMS fast-path" financial transaction, consisting of thousands of instructions and several I/O and cryptographic instructions.

instruction, and the wait is short because the execution time for a typical ESA/390 instruction is short. However, with the long execution time for typical cryptographic instructions, if no special action were taken to improve the interlock, the waiting time for all CPUs to signal completion would be substantially increased. In some implementations, all CPUs must wait until the CPU that takes the longest time has completed its current instruction. In these implementations, the significance of this increased waiting time is even more pronounced.

To reduce this wait, the results of most cryptographic instructions are placed in general registers. This allows the CPU to respond to an interlock request from any other CPU immediately after all storage operands of the instruction have been fetched. As a result of this additional use of general registers, some cryptographic instructions use as many as ten general registers. This extensive use of general registers does not impose any significant programming constraint, since 1) there is no compiler support for the instructions; 2) application programs cannot issue the instructions directly, but must invoke them by means of a privileged support program; and 3) the only intended support program, the IBM Integrated Cryptographic Service Facility (ICSF/MVS), uses the cryptographic instructions in a single subroutine that saves and restores all of the necessary general registers.

Sectioning instructions

The operands used in some cryptographic operations may be very long and must be located in main storage. MAC (message-authentication code) generation and MAC verification operate on a variable-length source operand. Encryption, decryption, and ciphertext translation operate on variable-length source operands and produce variable-length results. "Sectioning" is used in these operations to provide a reasonably short length of time between interruptible points.

By dividing the processing of a long operand into intermediate-length sections, the CPU can recognize and respond to an interlock or interruption request at the end of each section; thus, the waiting time of the interlock requests can be reduced. Sectioning could be achieved by defining the instructions to be interruptible (as for the System/370™ MOVE LONG [8]), by defining an explicit fixed section size (as for the ESA/390 MOVE WITH KEY [9]), or by defining an explicit implementation-dependent section size (like the vector section size). The cryptographic instructions use a different method [10] of dividing the processing.

Each variable-length storage operand for a cryptographic instruction is specified by means of an address and length in a pair of general registers. Execution of the cryptographic instruction processes an implementation-dependent amount of the operand, updates the general registers to show the address and length of the remaining operand, and sets a condition code (status bits in the CPU) to indicate whether or not the end of the operand was reached. The program can test the condition code set by the instruction in order to determine whether to continue the processing.

Unlike instructions based on an explicit section size, in which the program must know the section size, the program does not have to know the section size for cryptographic instructions, and the cryptographic section size does not have to be a fixed value, thus providing more implementation flexibility. Sectioning for cryptographic instructions achieves the same result as making instructions interruptible, with less implementation complexity.

Storage access

Two ESA/390 precedents in the area of storage access have been established by several cryptographic instructions.

Three storage keys The execution of most ESA/390 instructions involves the use of a single "access key," which is used to gain access to both instructions and operands in storage. Some ESA/390 instructions, such as MOVE WITH KEY, permit the specification of a second access key, to be used in accessing a particular storage operand. As mentioned before, encryption, decryption, and ciphertext translation operate on variable-length source operands in storage and produce necessary variable-length results in storage. Since the information associated with cryptographic operations may be considered to be quite sensitive, the locations associated with the instruction, the fixed-length operands, and the two variable-length operands must be fetch-protected, and the source and result operands must be protected by different keys. Accommodating these requirements requires three access keys: one used to access instructions and the fixed-length operands, and one each to access the source and target operands. No other ESA/390 instruction uses more than two access keys.

No boundary alignment Storage operands of most ESA/390 privileged instructions are constrained to be located on word or double-word boundaries. The storage operands for cryptographic instructions are normally specified by application programs, which do not necessarily recognize boundary alignment. In order to place short operands on the appropriate boundaries, ICSF/MVS moves the operands supplied by the application program to another location. However, since the variable-length storage operands may be very long, this technique becomes impractical. Therefore, even though they are

privileged instructions, the cryptographic instructions allow variable-length storage operands to be located on any boundary.

Error indication and recovery

The philosophy of error handling adopted for the ICRF is similar to that for I/O. The ICRF failure conditions are reported by means of a condition code and status word. No new machine-check interruption is defined for the ICRF. This is because cryptographic requests, like I/O requests, are routed through a single service program, ICSF/MVS. Since ICSF/MVS must examine the condition code and status-word setting after each cryptographic instruction for many other unusual situations, it was simple to add a bit to the status word for indicating errors; there was no merit in reporting hardware failure as a machine-check interruption.

Redundancy and automatic hardware recovery are implemented within each crypto unit. To achieve high availability, two crypto units can be installed in a system.

When hardware recovery for crypto-unit errors fails, an error indication is reported to the control program, and failing components are automatically disabled by the hardware. The control program can continue to use the CPU for noncryptographic tasks.

Special security mode

The import-clear-key and generate-clear-PIN functions are required for compatibility, migration, and some other special purposes, but these functions, if misused, can result in a significant reduction of overall system security. To maintain security, one approach considered was to provide clear-key and clear-PIN handling on an off-line system (a separate support system). Encrypted keys and encrypted PINs would then be exchanged between the main and off-line systems. That approach was discarded because moving those functions from one system to another does not solve the problem; it only moves it. The security of the main system depends on the security of the off-line system. In effect, from a security viewpoint, the off-line system is an extension of the main system. Also, the use of two systems, rather than one, to solve the problem is more complex and more costly.

Instead, a special-security mode was implemented to enable these functions. The special-security mode is controlled by means of a physical key. The mode must be active in order to perform the import-clear-key and generate-clear-PIN functions. Changing the mode does not disrupt normal applications. The security risk is reduced by disabling the functions when they are not in use.

For users who prefer to use two systems, two PR/SM partitions (see footnote 6, above) can be established, one of which runs in the special-security mode. This allows the user to implement the off-line system in a PR/SM partition.

Dynamic master-key update

ICRF uses a "master key," which resides inside the tamper-resistant enclosure, to protect other cryptographic keys in the system. The capability to update the master key dynamically and without disrupting application programs was required.

Keys encrypted under the master key are said to be in the "operational state." Most keys in the operational state are maintained by ICSF/MVS in a cryptographic key data set (CKDS). Other keys in the operational state are maintained by application programs.

The following list summarizes the major steps of the process chosen to satisfy the requirements for dynamic update of the master key:

1. A new master key is entered at the manual-control panel.
2. ICSF/MVS performs a batch job to convert keys in the current CKDS, which are encrypted under the current master key, to be encrypted under the new master key, and to place the result in a new CKDS. The re-encipher-to-new-master-key function is provided specifically to perform this conversion.
3. ICSF/MVS executes the set-master-key function and switches to the new CKDS. The set-master-key function causes the current master key to become the old master key and the new master key to become the current master key. The set-master-key function also changes the value in a register called the master-key-version-number register.
4. Whenever an application requests a cryptographic service involving a key in the operational state, it supplies the key and a master-key version number. The master-key version number supplied by the application is compared with the master-key-version-number register. If the two master-key version numbers match, the application has supplied an up-to-date key, and the function can be performed. If the two master-key version numbers do not match, ICSF/MVS is alerted that conversion is required. If the master-key version number supplied by the application matches the most recent previous value, ICSF/MVS automatically converts the key supplied by the user from being encrypted under the old master key to being encrypted under the current master key. A new function, reencipher-from-old-master-key, is provided specifically to perform this conversion.

Conceptually, the master-key-update process could be executed without step 2, since a key in the operational state may be converted by step 4 when the key is first used. However, the step was included for the following reason: Although the conversion process is transparent to application programs, this approach, if applied to a larger number of keys, could degrade performance significantly.

Multiple cryptographic domains

To achieve physical isolation and protection among PR/SM partitions, multiple cryptographic domains were adopted, each with its own master key and associated registers. A control program can use a cryptographic domain only if it knows the secret authorization pattern associated with the domain. Having multiple cryptographic domains permits PR/SM support of the following two configurations:
1) A cryptographic testing system and a cryptographic production system run on the same machine (uniprocessor or multiprocessor), each using a different master key.
2) Several backup systems run in different partitions on the same machine. Each partition must use a different master key, since the front-end production systems are independent cryptographic users, each using a different master key.

Multiple cryptographic domains are also provided when operating without PR/SM. This permits different control programs to be run on the machine (by means of initial program loading), each using a different cryptographic domain with a different master key.

PR/SM support

This section discusses the requirements and problems associated with PR/SM support.

The requirement for CPU affinity (defined previously in the subsection on interchangeability) makes support of ICRF for partitions more complex. For workload balance, PR/SM normally uses floating CPU scheduling—that is, a logical CPU in a partition can run on any physical CPU. This scheduling was extended to include crypto units. Floating crypto-unit scheduling can be used only when CPU affinity is not required.

PR/SM must be alerted when the crypto units have become noninterchangeable and when a partition attempts to execute an instruction which invokes a cryptographic function that must be performed on a specific physical crypto unit. This is accomplished by defining special signals for those cryptographic functions associated with master-key entry, error handling, and other processes that may cause the crypto unit to become noninterchangeable. Exception conditions that may indicate that the crypto units have become noninterchangeable are signaled to PR/SM. PR/SM uses these signals to turn off floating crypto-unit scheduling. When floating crypto-unit scheduling is off, each CPU in the partition must run on a predetermined physical CPU. Floating crypto-unit scheduling is resumed after all crypto units in the system become interchangeable again.

When a virtual machine is running under the VM control program (CP) and CP is running under PR/SM, floating crypto-unit scheduling can be used by both CP and PR/SM. This means that when CPU affinity is required, crypto-unit floating must be prohibited at both CP and PR/SM levels. This was achieved by having control returned to the appropriate control program when CPU affinity is required. The architecture was defined in such a way that ICSF/MVS need not know whether it is running natively, in a partition, or as a guest.

As part of the master-key-entry process, a secret quantity, the master-key authorization pattern, is returned by the hardware to the program. After the master key has been manually entered, the program cannot perform most cryptographic functions unless it can supply this secret quantity. For security reasons, it is undesirable for PR/SM to maintain a copy of this secret quantity. Without this information, PR/SM cannot simulate cryptographic functions for a partition. Therefore, it was necessary to define the ICRF architecture in such a way that simulation of cryptographic functions is never required. This was accomplished by defining special controls for each function or condition that must be signaled to PR/SM. These controls can be set to either cause a signal or permit execution of the function. PR/SM sets the controls to cause signaling when floating crypto-unit scheduling is on and to permit execution of the function without signaling when floating crypto-unit scheduling is off.

• *Cryptographic-design considerations*

In this section, some basic concepts of the ICRF key-management scheme are reviewed, and several key decisions with respect to the ICRF cryptographic design are discussed. A detailed description of these concepts and the ICRF approach are provided in [4].

Cryptographic keys can be divided into different types according to key usage. **Table 1** shows the five basic key types supported by the ICRF and their intended functions. Note that each basic key type has more than one intended function and that for some functions more than one key is involved. If a key of a particular type could be used for unintended functions, severe security exposures would exist. For example, if a key-protection key could be used as a data-protection key, cryptographic keys encrypted under a particular key-protection key could be disclosed in the clear by using the key-protection key and the decryption function.

When a key is shared by two systems, the key is normally used in one system for a particular function (e.g., encryption) and in the other system for a different function (e.g., decryption). These two functions are called complementary functions. To prevent misuse of keys between complementary functions, the ICRF further divides each basic key type into more detailed key types, as shown in the right-hand column of Table 1. The key types for complementary functions are called complementary key types.

When a key is generated, the intended usage is specified by the application program through ICSF, and the

Table 1 Summary of major ICRF functions and associated key types.

Basic key type	*Function*	*Detailed key type*
Data-protection	Encryption	Data-encrypting
	Decryption	Data-encrypting key
	Ciphertext translation	Data-translation
		Data-translation
Message-authentication	MAC generation	MAC-generation
	MAC verification	MAC-verification
PIN-protection	PIN translation	Input PIN-encrypting
		Output PIN-encrypting
	PIN verification	Input PIN-encrypting
Algorithmic PIN-generation	PIN verification	PIN-verification
	PIN generation	PIN-generation
Key-protection	Key export	Exporter key-encrypting
	Key import	Importer key-encrypting

hardware associates the corresponding key type with the key. The key type of the key remains unchanged thereafter and is enforced by the hardware so that meaningful results are not produced if the key is used for unintended functions.

Degree of key separation

Generally, using more key types provides better security. However, key-management processes become more complex when more key types are handled. A careful trade-off between security and usability was a major design consideration.

Conceptually, a different detailed key type could be provided for each unique use of cryptographic keys. This is impractical for the ICRF, however, because of the adverse effects on usability and the space constraint on hardware design. The following provides the rationale behind some decisions on key separation of certain detailed key types.

Data-encrypting and data-translation key types The ciphertext-translation function is provided to allow intermediate nodes to reencipher messages that are passing through without disclosing the messages in the clear outside the tamper-resistant enclosure. The function requires two cryptographic keys; it first deciphers the input message in the clear using one key and then enciphers the clear message using another key. The function can be paired with the encryption or decryption function, or with the ciphertext-translation function itself, to form complementary functions. **Table 2** summarizes the ICRF complementary functions for data protection. If a different key type were used by the function at each end of every complementary pair in the table, a total of eight different detailed key types would be required.

The ICRF provides only two detailed key types for data protection: The data-encrypting key type is used by both the encryption and decryption functions; the data-translation key type is used for both keys of the ciphertext-translation function. This decision was made for the following reasons: 1) It is a customer requirement that application programs written for the Cryptographic Unit Support Program (CUSP) (see footnote 3) or the Programmed Cryptographic Facility (PCF) (see footnote 4) must run under ICSF/MVS. For this compatibility, the same key type is used in both the encryption and decryption functions. 2) Separation between the data-encrypting key type and the data-translation key type is necessary to prevent programs at a switching node from being able to decipher messages that are passing through. 3) Additional key types would not have enhanced security significantly and would have entailed additional complexity of key management and the associated software support.

MAC-generation and MAC-verification key types A message authentication code (MAC) is a cryptographic check sum which can be used to verify that a message has been received without modifications; it can also be used to authenticate the message originator. A MAC is generated for a message, using a cryptographic key, and is sent with the message by the originator. To verify the integrity of the message, the recipient performs MAC verification. Verification consists of generating a MAC for the received message using the same key and comparing the generated MAC with the received MAC. If they match, it is highly likely that the message is genuine and has been received without modifications. The MAC-generation and MAC-verification functions are complementary functions.

If the key used by the recipient for MAC verification could also be used for the MAC-generation function,

anyone who has access to the key at the recipient's system would be able to generate a valid MAC for a bogus message. Separation between the MAC-generation key type and the MAC-verification key type removes this concern.

PIN-generation, PIN-verification, input PIN-encrypting, and output PIN-encrypting key types A secret personal identification number (PIN) is usually used to authenticate the holder of a debit card or credit card in an electronic-funds-transfer system. The PIN can be a random number assigned by the card issuer, or can be cryptographically derived from some information about the cardholder. The PIN-generation function is provided to derive a PIN using a cryptographic key, called an algorithmic-PIN-generation key, and a specific algorithm. The derived PIN is in the clear, and the function is available only in the special-security mode. Random PINs are normally encrypted and stored in a PIN database for PIN verification. Derived PINs need not be stored and can be regenerated at verification time.

After being entered at an automated teller machine (ATM) or point-of-sale terminal for host PIN verification, the PIN is encrypted. The encrypted PIN is sent to the host. This is called the PIN-entry function.

In some situations, a PIN entered by a cardholder may travel through several nodes before it reaches the system that performs PIN verification. The PIN-translation function is provided to allow switching nodes to reencipher PINs without disclosing the PINs in the clear. The function requires two keys—one for decrypting the incoming encrypted PIN and the other for encrypting the outgoing PIN.

The PIN-translation function may also be employed in PIN verification using a PIN database. In this application, the received PIN in the encrypted form is converted to become encrypted under the PIN-database key. The result is compared with an appropriate entry from the database.

The PIN-verification function, which uses a cryptographic algorithm, requires two keys—one for decrypting the received PIN and the other for deriving a reference PIN. The received PIN is deciphered and compared with the generated reference PIN.

Table 3 summarizes complementary functions for PIN processing. The ICRF provides four PIN-related key types for the following reasons:

1. A card issuer may authorize other institutions to perform the algorithmic PIN-verification process for the issuer. Separation between the PIN-generation key type (for the PIN-generation function) and the PIN-verification key type (for the PIN-verification function) prevents these institutions from generating valid PINs.
2. The PIN-translation function is defined to use two key types: an input PIN-encrypting key type for the key protecting the input PIN and an output PIN-encrypting key type for the key protecting the output PIN. This separation prevents the function from being used (with the PIN-database key) to reencipher a PIN database to become one encrypted under a key chosen by an adversary and potentially known to the adversary.
3. After a system has received an encrypted PIN sent from an ATM or point-of-sale terminal, the system may perform a PIN-verification function or a PIN-translation function, depending on whether the received PIN is to be processed by the system or forwarded to another system. Most current ATMs and terminals use only one cryptographic key for protecting PINs. To accommodate existing equipment, the same key type (input PIN-encrypting) is used to protect the input PIN for both the PIN-translation and PIN-verification functions.

Table 2 Complementary functions for data protection.

Encryption	Decryption
Encryption	Ciphertext translation
Ciphertext translation	Ciphertext translation
Ciphertext translation	Decryption

Table 3 Complementary functions for PIN processing.

PIN generation	PIN verification
PIN entry*	PIN verification
PIN entry*	PIN translation
PIN translation	PIN translation
PIN translation	PIN verification

*Pin entry is performed at an ATM or point-of-sale terminal and is not provided by the ICRF.

Importer and exporter key-encrypting key types A key-encrypting key (KEK) is used to protect other keys of any type. The key-export and key-import functions are provided for key distribution; they are complementary functions. The key-export function reenciphers a key from being encrypted under the master key of the sender to being encrypted under a KEK shared with the receiver; the key-import function reenciphers a key from being encrypted under a KEK shared with the sender to being encrypted under the master key of the receiver. An exporter key-encrypting key type is used for the KEK in the key-export function, and an importer key-encrypting key type is used for the KEK in the key-import function.

The ICRF provides a function that generates two copies of the same key with complementary key types. One copy can be used by the generation system, and the other is ready to be sent to another system. The key to be sent is encrypted under an exporter KEK. If separation between the exporter and importer key-encrypting key types were

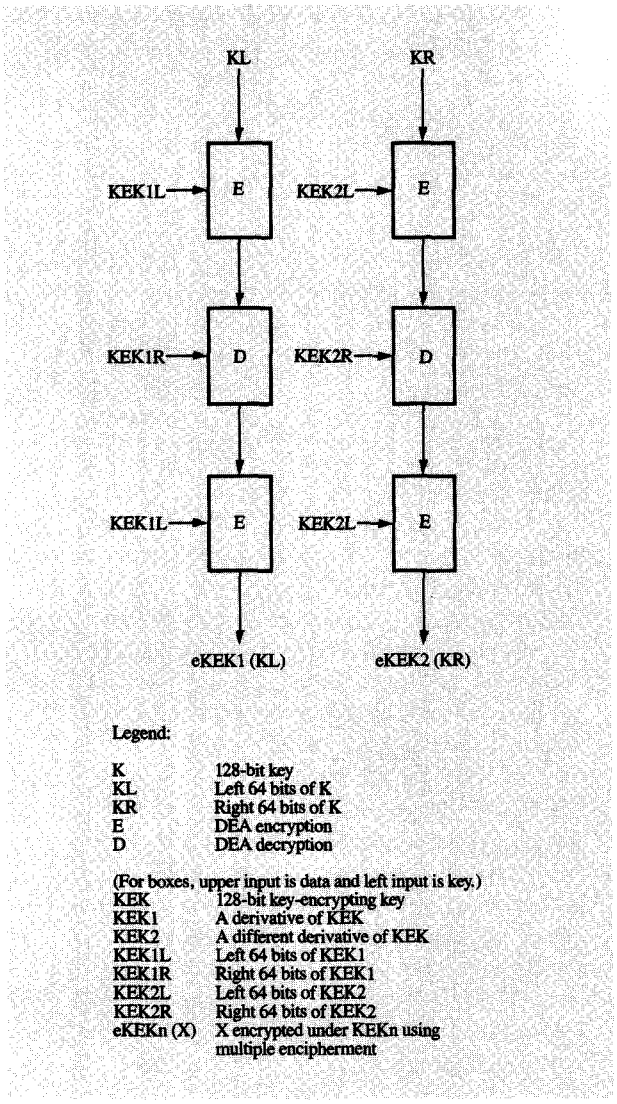

Figure 1

Multiple encipherment for key protection.

not provided, any key to be sent could also be imported back to the sender by an adversary. This would allow the adversary to obtain all pairs of complementary keys, thus compromising key-usage control.

Key length and key protection

The introduction of high-performance cryptographic functions provides a very powerful code-breaking tool. This leads to additional security considerations. In addition to supporting many key types, the ICRF offers the following additional security enhancements: All key-encrypting keys (KEKs) and PIN-related keys are 128 bits long, and the left and right halves of a 128-bit key are protected by different derivatives of a KEK. (The derivative of a KEK is obtained by performing the exclusive-OR of a value with the key.)

One of the many reasons why each half of a 128-bit key should be encrypted under different derivatives of a KEK is provided here for illustration. As can be seen in **Figure 1,** if the same derivative of a KEK were used to encrypt both halves of a 128-bit key (i.e., KEK1 = KEK2), and if both halves of the 128-bit key had the same value (i.e., KL = KR), the encrypted values of both halves of the key would be the same [i.e., eKEK1(KL) = eKEK2(KR)]. This allows 128-bit keys with equal halves to be detected and also allows 128-bit keys with equal halves to be created. One could create a 128-bit key with equal halves in the encrypted form by simply declaring a 64-bit random number as both halves of an encrypted 128-bit key. Even though the clear value of this randomly created key is unknown, the key has only the strength of 64 bits. This ability to create a 128-bit key with equal halves drastically reduces the strength of system security. For example, an adversary could create an exporter KEK with equal halves in the encrypted form. The adversary could then use the key-export function to reencipher all keys in the CKDS (cryptographic key data set) to become encrypted under that exporter KEK. Thus, the overall system security is effectively reduced to the level of 64-bit keys. Encrypting each half of a 128-bit key under a different derivative of a KEK eliminates the ability to create and detect equal halves of a 128-bit key.

Concluding remarks

The design of ICRF led to many unusual architectural solutions. The physical limitations imposed on the implementation resulted in special instructions to assist the control program in scheduling. The long execution time sociated with many of the cryptographic operations quired special handling of operands and sectioning. The continuous-operation requirement for large systems sulted in a requirement for dynamic master-key update.

The stringent requirement for security led to many unusual considerations that shaped the design of the ICRF. To provide for protection of operands in storage, three access keys were required for the execution of a single instruction. To achieve physical isolation and protection among logical systems running on the same machine, multiple cryptographic domains were provided. To obtain high availability, extensive redundancy and transparent hardware recovery were required to protect the secret quantities involved. To facilitate control of the usage of individual cryptographic functions, all cryptographic instructions were specified to be privileged.

The most significant difference between the ICRF implementation and other hardware cryptographic devices is that the ICRF is implemented on CPUs of general-

purpose mainframes. The high-performance characteristic of this CPU-integrated approach allows the facility, with little overhead, to use a 128-bit key length, thus enhancing security for most cryptographic keys. The high-performance characteristic also introduced many new challenges, requiring a careful evaluation of overall security. The results of the evaluation shaped many aspects of the facility.

Another unique characteristic of the facility is its ability to significantly enhance security by means of key separation. Although using more key types generally provides higher security, it also complicates key management. The number of key types provided by the ICRF was carefully investigated and determined in order to provide a balanced system that best fits the design criteria and market requirements.

Acknowledgments

Many individuals have contributed to the concept and development of this product. Among them, Walter F. Bankowski, Brian B. Moore, and Julian Thomas initiated the project and established hardware-design guidelines. Chris J. Holloway and Robert J. Rosenthal defined PIN-processing environments and customer requirements. Ernest T. Zooper constantly supplied market requirements. Stephen M. Matyas provided general cryptographic direction; Don B. Johnson frequently reviewed ICRF security aspects. Randall J. Easter, John T. Matcham, Chuck F. Megivern, Brian M. Moriarty, and Vincent A. Spano provided hardware-implementation input and feedback. Gina Bourbeau, Lucina L. Green, Michael J. Kelly, and R. Craig Larson specified software-design criteria. Peter H. Gum, Roger E. Hough, Sandy L. Rankin, Steve J. Schmandt, and Devon Yu supplied PR/SM-support requirements. Dennis G. Abraham, Ramesh K. Karne, Carl H. Meyer, An V. Le, Russ Prymak, and John D. Wilkins suggested security improvements.

References

1. *Data Encryption Algorithm*, American National Standard X3.92-1981, American National Standards Institute, New York, December 31, 1981.
2. *American National Standard for Financial Institution Message Authentication (Wholesale)*, American National Standard X9.9-1986, American Bankers Association, Washington, DC, August 15, 1986.
3. *Personal Identification Number (PIN) Management and Security*, American National Standard X9.8-1982, American National Standards Institute, New York, January 14, 1982.
4. P. C. Yeh and R. M. Smith, Sr., "ESA/390 Integrated Cryptographic Facility: An Overview," *IBM Syst. J.* **30**, 192–205 (1991).
5. *IBM Enterprise Systems Architecture/390 Vector Operations*, Order No. SA22-7207; available through IBM branch offices.
6. T. L. Borden, J. P. Hennessy, and J. W. Rymarczyk, "Multiple Operating Systems on One Processor Complex," *IBM Syst. J.* **28**, 104–123 (1989).
7. *IBM 3848 Cryptographic Unit Product Description and Operating Procedures*, Order No. GA22-7073; available through IBM branch offices.
8. *IBM System/370 Principles of Operation*, Order No. GA22-7700; available through IBM branch offices.
9. *IBM Enterprise Systems Architecture/390, Principles of Operation*, Order No. SA22-7201; available through IBM branch offices.
10. "Instruction with Long Operand Converted to Intermediate Length Operation by Central Processor," *IBM Tech. Disclosure Bull.* **31**, 78–79 (1989).

Received January 31, 1991; accepted for publication May 21, 1992

Ronald M. Smith, Sr. *IBM Enterprise Systems, P.O. Box 950, Poughkeepsie, New York 12602.* Mr. Smith is a Senior Technical Staff Member in the Enterprise Systems Central Architecture Department of the Mid-Hudson Valley Development Laboratory in Poughkeepsie. He received his B.E.E. degree in electrical engineering from The Ohio State University in 1957 and joined IBM at the Endicott Laboratory that same year, moving to Poughkeepsie in 1961. He worked on assignments in circuit design, central processor design, and programming before joining Central Systems Architecture in 1966. Mr. Smith has twelve patents, six patent applications on file, and thirteen published invention disclosures. He has received an IBM Outstanding Contribution Award, an IBM Outstanding Innovation Award, and an IBM Sixth-Level Invention Achievement Award.

Phil C. Yeh *IBM Enterprise Systems, P.O. Box 950, Poughkeepsie, New York 12602.* Dr. Yeh is a Senior Engineer in the Enterprise Systems Central Architecture Department of the Mid-Hudson Valley Development Laboratory in Poughkeepsie. He received an M.S. degree in computer science and a Ph.D. 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 assignments in architecture. He has five issued patents and three patent applications on file. He has also published several technical papers and has received an IBM Outstanding Innovation Award. Dr. Yeh is a member of the ACM and the IEEE Computer Society.