Transaction Security System

by D. G. Abraham
G. M. Dolan
G. P. Double
J. V. Stevens

Components of previous security systems were designed independently from one another and were often difficult to integrate. Described is the recently available IBM Transaction Security System. It implements the Common Cryptographic Architecture and offers a comprehensive set of security products that allow users to implement end-to-end secure systems with IBM components. The system includes a mainframe host-attached Network Security Processor, high-performance encryption adapters for the IBM Personal Computer and Personal System/2® Micro Channel®, an RS-232 attached Security Interface Unit, and a credit-card size state-of-the-art Personal Security™ card containing a high-performance microprocessor. The application programming interface provides common programming in the host and the workstation and supports all of the Systems Application Architecture™ languages except REXX and RPG. Applications may be written to run on Multiple Virtual Storage (MVS) and PC DOS operating systems.

Competition for development resources motivates the design of long-lasting systems that contain common elements. Developing a comprehensive security system presents unique challenges in architecture, hardware, and programming.

Controlling access to the system capabilities is fundamental to a comprehensive design for a security system. If it is relatively easy to alter the parameters that control the system, security could be compromised. Such access is usually based on verifying the identity of a specific individual. Verification can be done through testing for something that the person knows, for example, a secret password; something that the person possesses, such as a brass key to a physical lock; or something that biometrically identifies the person, such as a personal signature.

Secret passwords and passphrases do not satisfactorily prove that the person entering the information is the legitimate owner of the password rather than merely someone who successfully discovered the secret. Passwords and personal identification numbers (PINs) can be guessed. In addition, the owners of passwords or PINs often write them down in convenient places in case they forget them, thus exposing them to unauthorized use.

Similarly, using something the person owns, such as a key or token, as the sole means for granting access does not prove the person presenting the key or token is the legitimate owner rather than merely the one who possesses it at that moment. If the token also contains additional information, such as a photograph of the owner, that may strengthen the proof, but such cards are routinely forged.

Using a human characteristic that biometrically identifies a person provides the strongest and

most reliable means of user identification and authentication. Systems that verify fingerprints have been developed, but to many people there is a stigma associated with the use of fingerprints for identification. Voice recognition has also been used, although readily available high-fidelity recording and playback equipment can accurately replay voice information. Identification of the blood vessel patterns on the rear of the retina of the eye has been proposed as well, but there seems to be a reluctance to adopt such devices for general application.

Attaching a written signature to a transaction as a form of authorization is a common practice, and many times is a required part of transacting business in the financial community. The signing of a name by an individual, if done in a "normal" manner, is a dynamic action. The signature flows from the pen to the paper without the individual thinking about it, and this action occurs in a remarkably repeatable fashion. The visible signature is vulnerable to forging, but the dynamic variables such as pressure and acceleration associated with producing the signature are much less so.

The IBM Transaction Security System, announced October 24, 1989,[1] was developed to meet requirements of the financial industry. It implements the Common Cryptographic Architecture.[2] The system, shown in Figure 1, offers a comprehensive set of security products that allow users to implement end-to-end secure systems with IBM components. The Transaction Security System includes an IBM 4753 Network Security Processor for attachment to a host computer, high-performance encryption adapters for the IBM Personal Computer and the Personal System/2® (PS/2®) Micro Channel®, an RS-232 attached IBM 4754 Security Interface Unit, a credit-card size state-of-the-art Personal Security™ card containing a high-performance microprocessor, and a signature verification pen and associated signal processor. The application programming interface[3] (API) is common on the host and the workstation, and it supports all of the Systems Application Architecture™ (SAA™) languages except REXX and RPG. Applications may be written to run on the Multiple Virtual Storage (MVS) and PC DOS operating systems.[4] The Transaction Security System also implements several extensions to the Common Cryptographic Architecture Cryptographic Programming Interface.[5] In addition to the cryptographic extensions, several services unique to the Transaction Security System are implemented.[6]

IBM's existing cryptographic products, 3848-CUSP[7] and the Programmed Cryptographic Facility (PCF),[8] are used by a number of banks. However, this equipment does not adhere to all applicable standards of the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO) because such standards were nonexistent when the equipment was developed. The Transaction Security System was specifically designed to meet all applicable ANSI and ISO standards and to provide a common base for the future development of related products and applications.

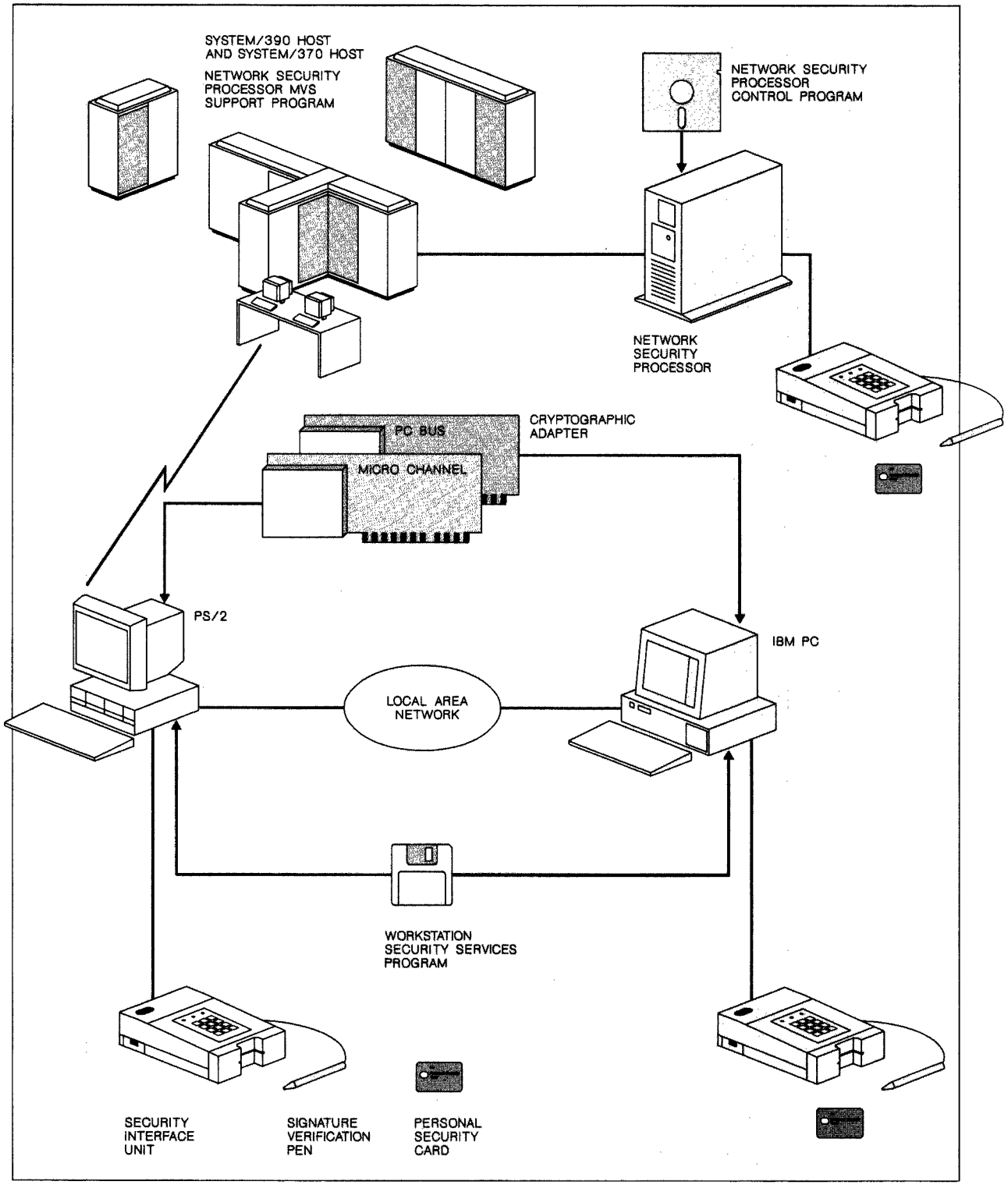
The Transaction Security System was specifically designed to meet all applicable ANSI and ISO standards.

History

Prior to the introduction of automated teller machines (ATMs) the market for cryptography was essentially limited to the military environment or to host file and workstation communications. Cryptography for the financial industry involves the use of the Data Encryption Algorithm (DEA).[9] The security obtained with the DEA is much like that obtained with use of a combination lock. The details of the construction of the lock are not secret, but the combination used to open the lock is. Similarly, the details of the DEA algorithm are public, and the security of the data is entirely dependent on keeping a secret value, called the *key*, secure from unauthorized individuals.[10]

The personal identification number (PIN) was introduced into the financial industry as the accepted means of identifying a bank customer and approving a transaction when the customer was not in the presence of a bank employee. Since knowledge of the PIN coupled with the possession

Figure 1 Transaction Security System products



of the ATM plastic card was sufficient proof for a bank to approve a transaction, protection for the PIN was required whenever it was entered or transmitted. Cryptography provides the only practical means of protection for the PIN at all points in the network where the PIN might be subject to hostile interception. Although cryptography provided protection for the PINs, it presented the bankers with the problem of how to manage the cryptographic keys. Bankers were not really interested in becoming cryptographic experts; thus an acceptable and competitive solution was needed.

The IBM financial products developed earlier have been used in many different environments and were all designed at separate times according to then-emerging requirements. At their inception, PIN processing was still relatively new, and standards had not yet been developed or even requested. New ideas and constantly expanding knowledge led to independent solutions to the same problem, not only among competitors but also among IBM organizations. Interoperability was either difficult or totally impossible. In their zeal for market acceptance, product developers implemented vastly different philosophies and techniques in large numbers. Such strategies led to expensive and long development cycles. These strategies can also create security weaknesses since different key-management systems need to be accommodated. In many implementations, security depended on the integrity of the designers and programmers, thus enabling insiders to launch attacks against equipment without detection. Clearly, improvements were needed if IBM were to continue to be a leader in the financial industry marketplace.

The Transaction Security System addresses these specific problems and concerns voiced by customers. The system includes several novel physical security features that are designed to fend off all but the most determined adversaries who are supported with unlimited resources. The system has been designed to minimize any advantage that the system designers might have by not relying on the secrecy of designs or algorithms in any way. Only the cryptographic keys and the PIN or password that is used to gain system access need to be maintained as secrets, and an optional signature verification feature removes the need for keeping the PIN or password secret. With this option, the dynamic variables involved in producing a signature are stored on a "smart" card for users, called the Personal Security card. System access is granted when a signature is entered, using a specially designed pen, that satisfactorily matches the signature dynamics stored on the Personal Security card.

IBM has worked on the problem of information asset protection for a long time. From the earliest days it was clear that cryptography was the best solution to the problem. In some early IBM equipment, the LUCIFER algorithm[11] was employed to provide cryptographic protection. IBM's response to a request from the United States government for a suitable general-purpose cryptographic algorithm led to the development of the Data Encryption Standard (DES).[12] Transparent session level encryption (SLE) was included in the Advanced Communications Function/Virtual Telecommunications Access Method, or VTAM, along with the introduction of the Programmed Cryptographic Facility.[8] VTAM SLE provides transparent cryptographic protection to all information flowing between a terminal and a host computer or between hosts without the explicit involvement of the sending or receiving application. After the application "sends" a message, VTAM encrypts the information before transmitting it to its destination. When an application "receives" a message, VTAM decrypts the information before passing it to the application. The IBM 3848 channel-attached cryptographic unit and the corresponding support program (3848-CUSP) were introduced a short time later to provide higher performance and a greater (hardware) level of security than PCF provided. These products also had an application-level interface that allowed user-written applications to encipher and decipher data. Among early product offerings in the financial industry was the IBM 3600 Financial Transaction System, which included primitive DEA functions. The IBM 3624 ATM, the 4700 Financial Branch System, and the 4730 Personal Banking Machine provide additional cryptographic capabilities to meet the requirements of more complex financial transaction processing.

In 1985, it was decided that a unified security strategy and architecture would be an important enhancement to the business strategy of the IBM Consumer Systems Business Unit (CSBU). Definition of the security strategy included the development of a pervasive security architecture that was to be followed by all CSBU product implementers. As one of the efforts to reduce product

development costs, product building blocks were defined to be reused in the development of future CSBU products.

During the development of the security strategy, it was noticed that there was little or no commonality among the various predecessor products. This finding led to a decision that a set of common functions should be defined to provide the same cryptographic functions at all points in the network where cryptographic processing was required. This set of functions became known as the Common Cryptographic Function (CCF) set.

A set of implementation-independent cryptographic function definitions was proposed to provide interoperatability between products without defining the implementation details to be followed. These definitions were submitted to various product development groups as a statement of the cryptographic processing requirements of the CSBU products with a request that they be included in current and future machines to which CSBU products might be attached.

Several other non-CSBU products also had requirements for cryptographic processing. Therefore, to obtain one design, responsibility was transferred to a neutral group not having a specific product interest, but which would have a strong interest and the capability to complete the development and definition of a comprehensive and complete common cryptographic architecture. Thus the IBM Cryptographic Center of Competence was chosen. The result of work by the center was the Common Cryptographic Architecture Cryptographic Application Programming Interface. The Common Cryptographic Architecture is to be used as the corporate strategic cryptographic architecture, and any IBM products employing cryptographic capabilities are required to adhere to it.

With the center developing the Common Cryptographic Architecture, CSBU personnel were able to spend their full time defining and developing the product set that would be used for securing financial transactions throughout the network. Thus began the definition and development of the Transaction Security System.

Objectives

The objectives for development of the Transaction Security System were derived from extensive joint studies with IBM customers in the banking industry over a period of several years. Additional objectives were formulated to be consistent with IBM strategic directions and business objectives. Participation in several ANSI and ISO financial security standards development projects ensured that the Transaction Security System would be consistent with emerging financial security standards.

A survey of customers produced a list of their needs. Among them were an unobtrusive product, a common programming interface for programs written to run on the host and those written to run on the workstation, SAA if available, and an "acceptable" level of physical security. Most networks are operated 24 hours a day, so continuous availability was important. Most customers do not have a full-time security staff and looked for compliance with applicable national and international standards as a first-level measure of "goodness" of the equipment. Customers also needed turnkey solutions and the ability to control access to the various system capabilities, as well as a well-defined path for migration from current to new equipment. Finally, they wanted assurances that whatever the product, it would be strategic, i.e., it would have IBM's commitment to use significant resources for its development as a product with potential enhancement and growth.

The IBM business objectives were to satisfy the customer requirements while developing a cost-effective product. These objectives usually mean minimizing the development expense. IBM management was interested in developing a product that was low in cost, could be developed quickly, and had the maximum number of common components that could be used in future products. Products that contained parts usable in other products were far easier to justify than were products that had all unique and unreusable components. Conformance to IBM and industry standards, as well as to the Common Cryptographic Architecture, was high on the list of desirable qualities. Also, it was more desirable to develop strategic instead of tactical products.

The final objectives to be used to develop the Transaction Security System were defined as a common set of requirements taken from a series of disclosures and studies with customers from around the world. The Transaction Security System was to conform to SAA design requirements,

even if the product plans for the various required systems could not all be met during the development cycle. The security functions were to be compartmentalized, that is, made separate and independent from one another, along with a granular and customer-selectable level of security. The Transaction Security System was to be interoperable with existing and planned IBM security products, and a well-defined migration plan was to be made available. Finally, the basic design assumption was to be secure from insider attacks. Although it is true that an insider seemingly always has the advantage, there were to be no weaknesses in the design that might be exploited by an insider having such knowledge. No "trap doors," undocumented "features," or other secret ways to gain access to the system were to exist.

The Transaction Security System would implement the Common Cryptographic Architecture.

Implementation challenges

All product development programs are challenging. Many design choices and compromises must be made. The Common Cryptographic Architecture clearly defined the cryptographic services that were to be implemented but did not cover such other aspects of the design as data entry, file formats, physical security, number of keys in key storage, frequency or method of key change, and equipment maintenance procedures. Several of these parameters are usually determined by existing equipment and environments.

Since IBM customers have major investments in the 4700 Financial Branch equipment and specifically the 3624 ATM, it was necessary to protect these investments as much as possible, yet also provide them with additional and improved function. In some cases, as new requirements were studied and understood, it was necessary to be careful to ensure that the Transaction Security System design provided support for existing equipment.

It was decided early that the Transaction Security System would faithfully implement the Common Cryptographic Architecture. The question was "when," because the Common Cryptographic Architecture and the Transaction Security System were on parallel development paths. Both were being tuned in response to new knowledge, new customer demands, interoperability issues, and sometimes just ordinary "bugs." In the final analysis, the Transaction Security System implements the Common Cryptographic Architecture, and it implements many compatible extensions. Choices had to be made concerning what additional customer requirements were to be met.

The Personal Security card was being developed while related industry standards were constantly being updated and altered. Some of the standards for smart cards (similar to credit cards but containing programmable circuitry) were not fully compatible with the Common Cryptographic Architecture. As a result, the product developers attempted to anticipate the direction the standards would take and incorporate this information into a design for the Personal Security card that would meet requirements of both the standards and the Common Cryptographic Architecture. In addition, IBM participated in the development of the smart-card standards.

There was an established market for security cards, and other manufacturers were the acknowledged leaders in the marketplace. Whatever IBM did needed to be compatible or at least interoperable with the other cards while still maintaining product differentiation to make the Transaction Security System desirable and marketable.

Software for the system had to take into account the fact that program code was to reside in the same machine as other applications, with allowable code space in customers' machines ranging from 5K to 600K. Most customers had PC DOS with Operating System/2™ (OS/2®) being their next logical step. Therefore, the logical plan was for the product developers to implement the PC DOS version first. The OS/2 version would follow as soon as resources permitted the work to be done. Likewise, the IBM PC I/O bus version of the Cryptographic Adapter was given priority in development to accommodate most customers' existing hardware.

Programming

As the product requirements emerged, it became clear that the hardware would have to be usable both in a control unit attached to System/370™ and System/390™ processors based on personal computer technology and as individual products connected to the IBM PC bus, Micro Channel, and serial interface. The computing platforms would include PC DOS and MVS with a strong desire to support additional platforms including:

- OS/2
- System/88
- Operating System/400® (OS/400®)
- Advanced Interactive Executive™ (AIX®) for PS/2 and the RISC System/6000™

There is a procedure call for each basic service.

Furthermore, the ability to offer cryptographic services from one or more server machines on a local area network (LAN) to other stations on the LAN was felt to be desirable.

It was decided that the initial software offerings would support the hardware as a set of tools—leaving the application development up to users and to providers of application software. Thus, the software consists of an application programming interface with underlying function to control the hardware and with utilities to configure the hardware and provide other basic support.

Application programming interface and support. Applications have access to the capabilities of the hardware through a series of callable services at the programming interface. Calls to the interface result in requests that are routed to a security server for processing. The requests can have cryptographic functions performed, manage data on the Personal Security card, perform I/O operations with the Security Interface Unit and Personal Security card, and manage the hardware access controls.

Programming interface. Since the computing platforms of interest are generally the SAA platforms, it was decided to adhere to SAA CPI (common programming interface) practices. Work by John Ehrman at the IBM Santa Teresa laboratory had identified a set of practices that can result in a "universally usable" programming interface. The Ehrman guidelines were adopted so that a single API could be defined for obtaining services consistently from any likely programming language on any of the computing platforms. The programming interface available with the Transaction Security System is a superset of the Common Cryptographic Architecture common API.[2,3,5]

Requests for service by applications or by the Transaction Security System utilities are communicated through a procedure call. There is a procedure call for each basic service, such as Encipher or Profile_Activate, with a fixed number of parameters per service call. The call parameters are simple address pointers to the variables that are shared with the service. The variables are either four-byte, twos-complement integers, or strings. The parameters can point to single variables or to one-dimensional arrays. All communication between an application and a service is via the call-identified variables—there are no side effects. Also, there is no concept of "open" and "close"; a service is presumed to be always available.

Applications can be written in a wide variety of programming languages so long as the language supports the standard calling sequence for the computing platform. The applications are linked with code supplied in an interface library. The linkage conventions are well standardized in the MVS environment. Conventions for the PC DOS environment are adapted from the dynamic link library (DLL) conventions of OS/2. PC DOS application programmers must take these conventions into consideration when choosing a language compiler. The IBM "/2" compilers and assembler for PC DOS and OS/2 are specifically supported—other compilers may also be usable.

Request processing. The hardware can be in the same machine as the using application or in another machine on a LAN. In the case of the IBM 4753 Network Security Processor, the machine hardware is channel-attached to a System/370 or System/390 processor. In other possible imple-

mentations the hardware could be in a coprocessor configuration.

In order to accommodate the various connections from applications to hardware, we selected a *client-server* system concept. (See Figure 2.) Applications and utilities obtain service by issuing a procedure call. The procedure-call name is an external reference which is resolved by the linkage editor as an entry point in the product interface code library. The interface code performs a preliminary analysis of the request, then packages the request in control blocks and data areas for communication to the server code. Each platform has a unique request communication scheme for moving a request to the server.

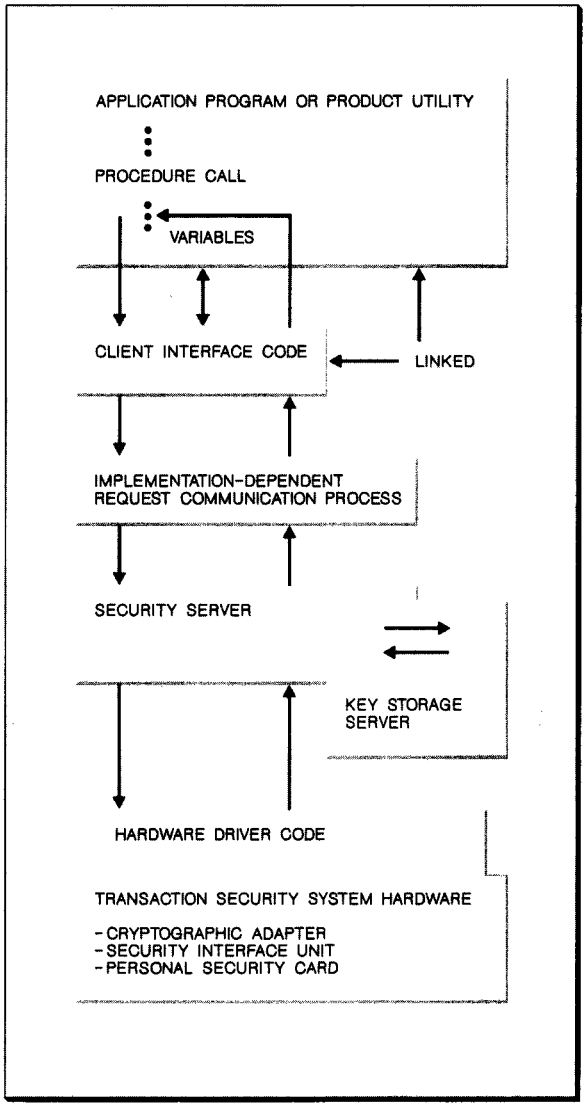
PC DOS request process. In the single-station PC DOS environment, the server is supplied address pointers to the control blocks and data areas, and control is transferred from the interface code to the server through the use of an interrupt.

In the PC DOS LAN environment, the interface code prepares request control blocks with pointers to the data and issues an interrupt. In conjunction with the Financial Branch System Services (FBSS) program, the control blocks and data are concatenated into one or more transmission blocks and sent over the LAN to the machine with the server code.

MVS request process. In the System/370 or System/390 environment, the interface code issues a program call instruction to transfer control to protected code within the Network Security Processor Support Program access method. In the access method the individual requests are transformed into control blocks and data and concatenated into a string. An exit is provided to the MVS Security Access Facility where the Resource Access Control Facility (RACF) can be used to authorize performance of the service or use of a key in key storage. Then multiple request strings can be combined into a single channel record for transmission to the IBM 4753 Network Security Processor in order to reduce system overhead associated with channel I/O activity. The access method can route the blocked requests to one of several attached 4753s to distribute the processing load among multiple 4753s.

Server process. At the server, the control blocks representing individual requests are examined by the security server code. Cryptographic requests can involve secondary requests to a key-storage server to obtain cryptographic keys. The security server prepares the detailed, bit-oriented control blocks required by the hardware driver code and initiates one or more I/O actions to perform the request. The results are packaged into updated control blocks and data areas, and the response is communicated back to the interface code which places the results into the variables of the application and returns control to the application.

Figure 2 Client-server request processing



Types of service. The software supports the following types of callable service requests:

- Cryptographic services—An application can request cryptographic services such as data ciphering, generation and verification of Message Authentication Codes (MACs),[13] and various key-management activities.

  Individual cryptographic keys can be supplied by the application, or the security server can obtain the cryptographic key from a key-storage server using a key label supplied by the application. Certain types of keys can also be stored within the hardware in special key registers. The keys are packaged in a data structure known as a key token which contains the key, the control vector,[14] and various processing flags such as the type of data cipher processing. Many different services are provided for managing key generation, supporting key distribution, and storing long-life keys in a server-managed key storage.

  The data ciphering operations support several different methods for processing data that are not eight bytes in length. Although the design point is oriented to ciphering and authenticating short data strings common in transaction processing, very long strings can be processed in a single call or group of related calls.

- I/O service for the Personal Security card and Security Interface Unit—Requests can be issued to power-up a Personal Security card, allocate, read and write data blocks, eject the card, and read information from the Security Interface Unit keypad and access-protected clock-calendar.

- Hardware access-control management—Each of the hardware units contains an access-control mechanism that determines which hardware commands can be performed. Services are provided for activating the various profiles, causing the hardware to check passwords and PINs or verify a signature against prestored signature reference information.

Utilities. The hardware is supported with several utilities that are part of the package. The utilities provide the tools needed to:

- Initialize the hardware with customer-specified user and application command authorization profiles
- Manage the installation of master keys and initial key-encrypting keys[14]
- Customize the support software for different memory environments
- Allocate Personal Security card data blocks, and read and write data in the blocks
- Provide support for the distribution of cryptographic keys via paper, diskette, and Personal Security card media
- Migrate cryptographic keys from older host products to the Network Security Processor

The batch initialization utility may be used with a control file created in the previously mentioned utility to quickly initialize the hardware with access control information and cryptographic keys.

The hardware is supported with several utilities.

This utility also supports fast initialization of Personal Security card groups that differ only in ID values, PINs, etc. A similar process is supported in the Network Security Processor for setting up that machine in a secure manner.

The building blocks

Designing and developing a cryptographic system requires special skills in addition to those skills normally required for any development project. Cryptographic equipment by its very nature becomes very important to an enterprise in that the equipment is used to protect the most valuable resources of an enterprise. Since so much important data may be protected by such a system, the user wants some assurance that the equipment has been carefully and responsibly implemented. By using common building blocks, we design equipment that works together. In addition, the development of functions represented by the building blocks does not need to be repeated for the next product.

The Transaction Security System is designed to operate with a very diverse set of systems and the technologies used in these systems, which in-

clude System/370, System/390, System/88, AS/400™, IBM PC, and PS/2. Since each of these systems has its own standard I/O bus architecture and available voltages and packaging technologies, defining a common hardware building block so "one size fits all" was extremely difficult. It was decided to define the Common Cryptographic Function (CCF) set as the common building block. It was a new idea to have a set of functions as a building block. All previous building blocks had been hardware components; this building block was the first that was a set of rules. The Engineering Design System (EDS) had sets of rules as building blocks, but these "rules" were sets of logic gates that a designer put together to perform a function. The implementation of the underlying component was predetermined and fixed by EDS. The implementation of CCF was left to individual design engineers.

Implementers using the CCF building block were free to select a technology most appropriate for their environment and requirements. It was only important that the definition of CCF be unchanged. If the rules were followed, the new product when completed would be cryptographically interoperable with other equipment also implementing CCF. Additional requirements led to the expansion of CCF into IBM's strategic Common Cryptographic Architecture.

The "ultimate" implementation would be a single-chip implementation of the architecture, including the required support functions such as memory, timers, I/O paths, etc.

The Shield module. The first implementation of CCF was called the Shield module. Its components were separate bare integrated-circuit chips interconnected on a module substrate.

Read-only memory (ROM) of the microprocessor component contained customized microcode including a DEA facility[9] performing basic DEA functions. Higher-level operations were implemented using the DEA facility and other utility functions contained in the ROM. However, it was difficult to obtain the bare chips or the die for them from manufacturers in a useful form prior to having them packaged.

This same chip set was also used in the experimental Personal Security cards. These cards were used only under the control of IBM since there were certain security exposures inherent in the multichip design. For example, a knowledgeable adversary could attach probes to the wires between the chips and monitor the flow of information. In this case the keys were in erasable programmable read-only memory (EPROM) storage, and the DEA executed in the microprocessor component. The secret cryptographic keys could be easily obtained in this way. The design goal was to develop a single chip that would fit the needs of both the Personal Security card and the reader for the card.

The HPS module. The high-performance Shield (HPS) module is the cryptographic facility for executing the Common Cryptographic Architecture services. All cryptographic operations take place inside the secured environment. Other services and extensions to the Common Cryptographic Architecture are also executed in the HPS.

Although a single-chip implementation would still be best, results of the cost vs function tradeoff study to build such a chip were not favorable, and all required technologies were not available. Using the available technologies would result in a very large chip. Such a large chip would not be suitable for use in the Personal Security card. It was decided to take a less aggressive approach with respect to technology while adding significant performance capabilities to the Shield module concept.

The HPS implements the entire Common Cryptographic Architecture function set. In addition to the common functions, HPS contains several compatible cryptographic extensions not implemented in other devices. Control vector[14] extension type 0 allows a single key to be used for a purpose such as generating a MAC, where use of the key can be linked to individual employees and multiple application divisions. The HPS module will also contain the User Defined Function (UDF) facility. The UDF facility is used to implement unique functions and proprietary algorithms to meet unique customer requirements. The functions of Receive Session Key and Verify Cryptographic Service Message provide a means of implementing a key-distribution system where the receiving terminals can run unattended.

HPS also implements a comprehensive set of noncryptographic functions for the purpose of supporting additional Transaction Security System

functions such as signature verification, functions for secure session establishment, and initial key loading procedures and functions.

The lessons of the first Shield module taught us to not use individual chips for the second pass. It was decided to use surface mount technology and off-the-shelf modules on a normal circuit card substrate. The second Shield module also was required to have much higher performance than the initial Shield module since it was intended to be used in the workstation adapter as well as the host product. It therefore required a hardware DEA processor. The physical security associated with the HPS is discussed in the next section.

The Transaction Security System utilizes many secret keys and authorization numbers.

Physical security for transaction systems

The Transaction Security System utilizes many secret keys and authorization numbers. Effective implementation of a secret-key cryptographic facility, along with the high value of assets that the keys are protecting, requires significant physical security to prevent the keys from being compromised. It was necessary, therefore, to define and implement some special physical security features to protect the encryption keys for the application environments expected for the Transaction Security System, and to meet ANSI and ISO security standards.

Design methodology. Implementation of effective physical security requires that the design pass through a number of phases.

The first phase, which precedes the actual design, consists of understanding the application environment or how the system will be used, where it will be used, and by whom. This phase provides details of what is to be protected in the system.

The second phase, also preceding the design, consists of a study of known physical security protection methods and the experience of others with attacks against cryptographic systems, transaction systems, and computer systems in general. The study then shifts to the specific system to be protected. Attack scenarios are proposed, and tests are conducted that will lead to an estimation of the expected adversaries and possible attacks against the system. Results from these preliminary studies are a great aid in knowing what to protect against. These studies also help to identify possible weak points during the early development phase of the system. Early feedback to the system designers allows design modifications to be made that can enhance overall system security.

The third phase uses the results of the preliminary studies to conceive and propose tentative designs of physical security and controls. The designs are built as prototypes and characterized to determine their potential effectiveness.

In the fourth phase, the physical security prototypes are developed into reliable, manufacturable, and cost-effective physical protection hardware which will be integrated into the Transaction Security System.

The fifth phase involves evaluation of the final product. The effectiveness of the physical security design is evaluated through analysis, characterization, attack testing, and reliability testing to ensure that the original objectives have been met.

Adversaries and attacks. The preliminary studies laid the groundwork to define the classes of adversaries expected and the types of attacks that might occur against the Transaction Security System. Adversaries were grouped into three classes, in ascending order, depending on their expected abilities and attack strengths.

Class I (clever outsiders)—They are often very intelligent but may have insufficient knowledge of the system. They may have access to only moderately sophisticated equipment. They often try to take advantage of an existing weakness in the system, rather than try to create one.

Class II (knowledgeable insiders)—They have substantial specialized technical education and experience. They have varying degrees of under-

Table 1 A security menu

Security Level	Definition
ZERO	No special security features added to the system. Example: a standard IBM PC in a room with free access.
LOW	Some security features in place. They are relatively easily defeated with common laboratory or shop tools such as pliers, soldering iron, or small microscope.
MODL	More expensive tools are required, as well as some specialized knowledge. Tool cost may range from $500 to $5000.
MOD	Special tools and equipment are required, as well as some special skills and knowledge. The tools and equipment may cost from $5000 to $50,000. The attack may become time-consuming but will eventually be successful.
MODH	Equipment is available but is expensive to buy and operate. Cost may range from $50,000 to $200,000 or more. Special skills and knowledge are required to utilize the equipment for an attack. More than one operation may be required so that several adversaries with complementary skills would have to work on the attack sequence. The attack could be unsuccessful.
HIGH	All known attacks have been unsuccessful. Some research by a team of specialists is necessary. Highly specialized equipment is necessary, some of which might have to be designed and built. Total cost of the attack could be one million dollars or more. The success of the attack is uncertain.

standing of parts of the system but potential access to most of it. They often have access to highly sophisticated tools and instruments for analysis.

Class III (funded organizations)—They are able to assemble teams of specialists with related and complementary skills backed by great funding resources. They are capable of in-depth analysis of the system, designing sophisticated attacks, and using the most sophisticated analysis tools. They may use Class II adversaries as part of the attack team.

Attacks of the kind that could occur against the Transaction Security System were proposed, studied, and evaluated for level of difficulty over a period of two years with the use of the facilities and services of a number of IBM locations. The attacks, some of which involved very sophisticated techniques and equipment, fell into three categories:

1. *Microcircuit attacks* are aimed at the hardware components where sensitive data are stored. A successful attack bypasses all software controls and directly reveals encryption keys or allows data to be altered.
2. *Counterfeiting and hardware simulation* substitutes hardware (and may include some special software) that is capable of subverting software control and allows unauthorized access into the system.
3. *Eavesdropping* is the process in which sensitive information is learned by picking up radiated signals from various points in a system or a network of systems.

The attack study provided information that led to a proposed scheme of levels of security that are related to the strength of an attack required to overcome a given security implementation. By relating the level of security to the difficulty of the attack, the scheme, shown in Table 1, helped to clarify the likelihood of an attack and to determine the physical security needed to minimize the threat.

Aspects of the design. A basic design objective of the Transaction Security System was that it should be secure from Class II adversaries. Even with detailed design information, a single insider should not be able to successfully compromise the system. That level of physical security corresponds to the MODH level in Table 1. One might justifiably ask why we should not protect against all adversaries, including those of Class III. A primary consideration was cost-effectiveness of the design. Protection at the MODH level may result in only a small increase in the cost of the

overall system, but protection at the HIGH level could conceivably *double* the cost of the system. The physically secure modules in which encryption keys are held could generally be augmented with a variety of additional security controls, depending on the level of assets to be protected and the security environment in which the system resides. Any additional security features would combine to provide the necessary overall security for the system.

The physical security design concept implemented in the secure modules of the Transaction Security System consists of a primary security layer backed up by a secondary defense which protects against attacks that try to circumvent the primary security layer. The concept is shown in Figure 3.

Encryption keys are stored in battery-backed semiconductor memory. The primary security layer is made up of a flexible membrane, containing a fine screened conductive ink pattern that surrounds the key-storage devices and encryption circuitry. The membrane is coated with a more durable material of similar chemistry. Attempts to break through the material are very likely to break the ink pattern. A detection circuit, based on an original design from the IBM Thomas J. Watson Research Center,[15] detects the break in the ink pattern and causes the keys to be thoroughly erased, thus preventing disclosure of the keys and other secret data. The secondary defense consists, for example, of such features as a temperature detection circuit that will also cause keys to be erased if an adversary attempts to "freeze" the keys into memory and prevent the erase circuit from working if the screen is breached.

Some standards, for example ANSI X9.17, recommend overwriting the key-storage memory a number of times with unrelated data to minimize the chance of key recovery by an adversary. This may be prudent, but it is also expensive in terms of the actual cost of additional circuitry and space on the circuit card. In addition, the overwrite circuitry may be subject to attack. It is suggested that if the semiconductor key-storage memory device cells are sufficiently characterized and understood, and the erase mechanism is effective, simple key erasure should be sufficient to protect the keys from compromise at the MODH level.

Reliability studies within IBM give confidence that the DES security module should function normally through the expected life of the system. Analysis of the design as well as some limited attack studies within IBM on the security hardware incorporated into the Transaction Security System provide the confidence that the MODH level of physical security has been achieved. An independent security evaluation by an external certified security organization should provide additional confidence to customers that the encryption keys are sufficiently protected. It is important that the external security organization be certified to ensure that it has the necessary knowledge and experience required for accurate physical security design evaluations, and is trustworthy. After the security standard FIPS (Federal Information Processing Standard) 140 is established, the National Institute of Science and Technology (NIST) has indicated an intention to certify approved organizations in the United States which would provide certified evaluations of commercial cryptographic modules for physical security effectiveness.

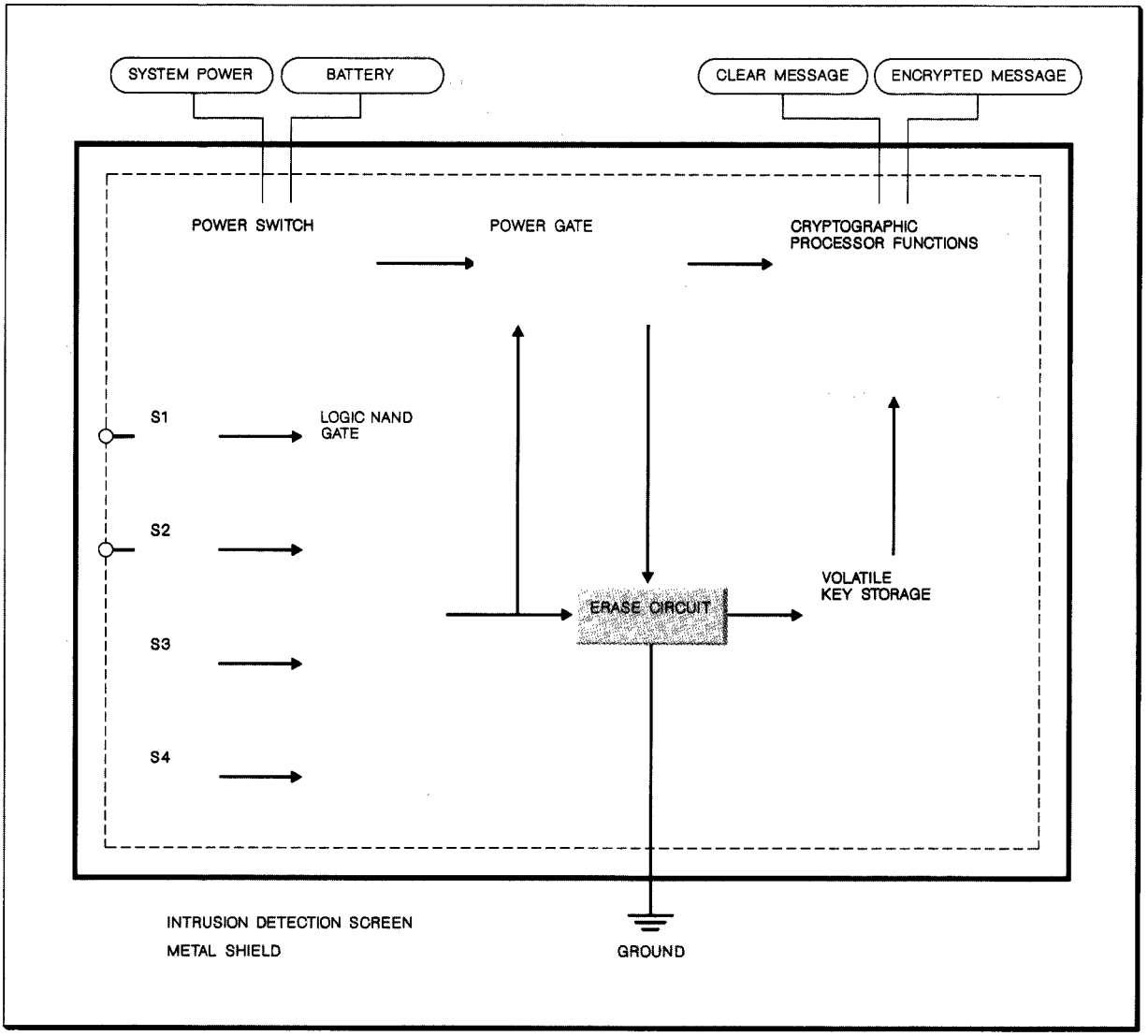
The cryptographic products

The Transaction Security System consists of several products which together can provide the security needed in transaction processing systems and networks. Each of the hardware products has a cryptographic *engine* that executes the Common Cryptographic Architecture and extensions. The hardware products are the only components where clear cryptographic keys are stored and used.

Cryptographic Adapter. The IBM 4755 Cryptographic Adapter provides all of the cryptographic functions in a high-performance package usable in either a workstation or the Network Security Processor. It can be used for supporting applications such as Systems Network Architecture (SNA) Session Level Encryption, transaction "MACing,"[13] and processing for the signature verification feature.

The Cryptographic Adapter is the highest-performance member of the Transaction Security System components. It comes in one of two models. One is an IBM PC I/O bus version for use in machines such as the IBM PC, PC/XT™, and PC/AT®. The other model is a Micro Channel version for

Figure 3 Conceptual block diagram of the intrusion protection method



use in machines such as the IBM PS/2 Models 50Z, 60, 70, and 80.

The Cryptographic Adapter consists of a set of building blocks which are assembled onto a raw adapter card. The primary building block is the HPS module that was described earlier. It is the heart of the Cryptographic Adapter. All of the controlling microcode for the adapter and the microprocessor upon which the microcode is executed are contained in the programmable read-only memory (PROM) modules that are encapsulated inside the HPS.

The 64K bytes of random access memory (RAM) inside the HPS is where the microcode maintains all of the data needed by the adapter to perform its function. Examples of the types of data stored in this RAM are a master key that is used for encrypting or decrypting data keys and key-encrypting keys, user authorization tables, com-

mand authorization tables, and global configuration data.

Besides the HPS, there are several additional blocks on the Cryptographic Adapter. It has an additional 128K bytes of RAM that are currently unused but are intended for future use by user-written programs. A battery maintains power to the RAM inside the HPS whenever the power for the personal computer is off. Finally, there is a serial communications chip and an RS-232 interface for communicating to a Security Interface Unit and a socket for attachment of a signature verification signal processor board.

The Security Interface Unit is a stand-alone box with its own power supply.

The Cryptographic Adapter microcode communicates with a device driver via several I/O ports and direct memory access (DMA). A set of commands and control blocks is defined. These commands and control blocks are transferred into the Cryptographic Adapter and contain the information necessary to perform the cryptographic and other security-related functions.

The Cryptographic Adapter can perform a comprehensive set of the cryptographic functions defined in the Common Cryptographic Architecture. Some examples are: Encipher and Decipher in CBC[16] mode, Generate or Verify MAC, key-management functions such as Re-encipher To or From Master Key and Generate Key Set, and financial PIN functions such as Verify Encrypted IBM 3624 PIN and Generate Formatted and Encrypted IBM 3624 PIN.

Security Interface Unit. The role of the IBM 4754 Security Interface Unit is to provide communications to and from the Personal Security card and support the secure entry of user authentication information via either the keypad or the signature verification pen. Finally, the Security Interface Unit can be used as a functional substitute for the Cryptographic Adapter when the customer is interested in a lower-cost solution and is not concerned with lower performance.

The Security Interface Unit is a stand-alone box with its own power supply. It has a tamper-resistant enclosure, an integrated-circuit chip card (i.e, *smart card*) reader, a 12-key keypad (similar in appearance to a telephone keypad), a connector for the signature verification pen along with analog-to-digital circuitry for converting the analog signal from the pen, and a nine-pin RS-232C communications port for attachment to a Cryptographic Adapter or directly to the RS-232C adapter of a workstation.

Inside the tamper-resistant enclosure is the logic card for the Security Interface Unit. It contains the microprocessor, 8K bytes of RAM which is battery-backed, and 32K bytes of PROM which contains the microcode. The DEA is implemented in software in the Security Interface Unit. The tamper-resistant features are similar to those for the HPS module previously described.

The Security Interface Unit provides a communications path to the Personal Security card via its integrated-circuit chip card reader. Through its keypad, it allows the secure entry of Cryptographic Adapter and Personal Security card PINs. It supports a subset of the Common Cryptographic Architecture services and stores in its RAM much the same type of information that the Cryptographic Adapter does.

Personal Security card. The Personal Security card provides a secure, portable cryptographic processor that is capable of performing all of the user authentication and authorization functions defined in the Transaction Security System. In addition, it can be used to store any type of information about or concerning the holder of the Personal Security card in its *data blocks*—user-definable data structures into which users can store any type of data that they wish. These data blocks can be protected via a number of security methods such as session key encryption. The card is used to store the data containing the signature dynamics for use by the signature verification feature discussed next.

The Personal Security card is about the same size, shape, and feel as a typical credit card, but

that is where the similarity ends. It incorporates a single integrated-circuit chip containing a processor and storage facilities, and it conforms to the evolving ISO standards for the physical characteristics of integrated-circuit chip cards.

The Personal Security card communicates with the Security Interface Unit via block protocol ISO 7816. Its single integrated-circuit chip contains a CPU, 10K bytes of ROM that contains the base microcode, 256 bytes of RAM, and 8K bytes of electrically erasable programmable read-only memory (EEPROM). It implements the DEA in software.

The Personal Security card stores basically the same type of data in its EEPROM as the 4754 and 4755 do in their RAM; however, the card has the capability of performing microcode patches via its EEPROM. Neither of the other two devices has any patch capability. In addition, the Personal Security card can use part of its EEPROM for data blocks. Data blocks give the Personal Security card a portable file capability much like a diskette has.

Signature verification feature. The signature verification feature consists of three pieces. First, there is the signature verification pen which is connected to the Security Interface Unit. It records signature dynamics by measuring the acceleration and rate of change of the pressure of the pen tip.

The second piece of the signature verification feature is the signature-processing daughter card which plugs into the socket on the Cryptographic Adapter. On the daughter card are a signal processor and 128K bytes of RAM. The signal processor does numerically intensive computations such as correlations, Fourier transforms, and floating point calculations which are part of the signature verification process. The RAM is used to contain additional signature verification microcode which is downloaded to the RAM by the Cryptographic Adapter device driver.

The third piece is the signature verification microcode. This microcode is in two parts. The main controlling code, including all code to communicate with the signature verification pen in real time and to read the signature reference data from the Personal Security card, is located inside the encapsulated module as part of the 256K PROM. The rest of the code is downloaded to the RAM on the daughter card.

Network Security Processor. The IBM 4753 Network Security Processor provides DEA cryptographic support for System/370 and System/390 host processors. It can also perform financial PIN cryptographic functions. It connects to a System/370 or System/390 through a high-speed block-multiplexer channel. The application programming interface is used by the user-written applications to implement secure transaction processing system applications. Internal key storage of the 4753 can hold 70 000 keys. There is an internal cache that holds the first 10 000 keys for fast access. Keys not in cache are retrieved from the fixed disk of the workstation, and a replacement algorithm swaps the newer key for an older one in the cache. Multiple 4753s can be attached to a single host mainframe. The Network Systems Program (NSP) MVS control program can control up to 16 4753s on a single host.

The Network Security Processor is based on an IBM 7531 Industrial PC AT with the following items added and changes made to convert it into a Network Security Processor. A Cryptographic Adapter is added, and a Security Interface Unit is attached to it. Two million bytes of additional memory is added to be used as a cache for cryptographic keys. A monochrome display is attached, and a System/370 or System/390 channel adapter card is added. A tamper-resistant enclosure and service access door with lock are added. The ROM BIOS (Basic Input Output System) is removed and replaced with an altered version to support operation without a keyboard (the Security Interface Unit keypad is used for any key input) and to prevent booting from the A-drive once installation is complete. Finally, the keyboard is removed. A Personal Security card is used for operator access, key transportation, and initialization.

Operation of the Network Security Processor is controlled by the Network Security Processor Control Program inside of it. In the System/370 and System/390, the Network Security Processor MVS Support Program provides application support for the Network Security Processor under MVS/370, Multiple Virtual Storage/Extended Architecture (MVS/XA™), or Multiple Virtual Storage/Enterprise Systems Architecture (MVS/ESA™).

The access controls

The Transaction Security System has controls for accessing its various functions and capabilities. The controls are required to prevent unauthorized individuals from altering the system configuration and to discourage attackers from attempting to use the capabilities of the system against itself. The controls are used to *authenticate* the user, *authorize* what a user is permitted to do, and *exclude* alien components from being introduced into the system by requiring a secure session to be established before certain functions may be executed.

User authentication is the ability to determine that users are who they say they are.

User authentication. User authentication is the ability to determine with high probability that users are who they say they are. An example in use today is the magnetic stripe card and PIN number that an ATM user possesses. With these two pieces of "information," the ATM is able to authenticate the user's identity. The importance of this function is readily apparent with the realization that it is undesirable for users to gain access to a system by falsely stating that they are someone else, since that other user may very well have different authorities and capabilities within the system.

The Transaction Security System supports three different types of user authentication, each of which has a different level of security.

1. The Cryptographic Adapter supports a VERIFY PIN command that accepts the PIN in the clear (not encrypted) from the personal computer application. The input PIN is compared inside the secure area of the Cryptographic Adapter against the stored PIN for that particular user. The Cryptographic Adapter returns a status code that indicates whether the verification was successful or not. In addition, if the verification was successful, the Cryptographic Adapter makes that user active and employs the user's authority table for authorization of future commands until the user "logs off."

   Obviously, this authentication method is not totally secure since the user's clear PIN is available in the personal computer.

2. The Cryptographic Adapter and the Personal Security card, in conjunction with the Security Interface Unit, support secure entry of the user's PIN via the keypad on the Security Interface Unit. In order for this method to work, a secure session must be in place between the Security Interface Unit and the device[17] on which the user's identity is being authenticated. Secure sessions are discussed in more detail later in this paper.

   In the case where the user's identity is being authenticated via the Personal Security card, a VERIFY PERSONAL SECURITY CARD PIN command is sent to the Security Interface Unit. It enables its keypad and gathers keystrokes until the Enter key is pressed. It then encrypts the entered keystrokes under the session key shared between it and the Personal Security card. The Security Interface Unit sends the encrypted value to the Personal Security card where it is decrypted and compared against the stored PIN. The Personal Security card returns an encrypted response to the Security Interface Unit, which indicates success or failure.

   In the case of authentication on the Cryptographic Adapter, the process is slightly different and involves more application interaction. The application first sends a command to the Security Interface Unit which asks it to do a secure read of the keypad and return the results encrypted under the session key, which it shares with the Cryptographic Adapter. The application then sends the encrypted result to the Cryptographic Adapter for an ENCRYPTED VERIFY PIN command. The Cryptographic Adapter decrypts the input and compares it to the stored PIN. It returns a response to the application that indicates whether the user's identity was verified. This authentication method offers more security than the first since the clear PIN is never outside any of the physically secure areas of the device. However, there is still the problem of the user exposing

the PIN while entering it on the keypad of the Security Interface Unit.

3. Signature verification is the most secure method of user authentication. To perform signature verification, the signature verification pen must be attached to the Security Interface Unit, the signature verification processor card must be installed on the Cryptographic Adapter, and the additional signature verification microcode must be downloaded to the RAM on the signature verification card.

   Because the signature verification process is based on signature dynamics which are widely believed to be unique to an individual, it is not sufficient to forge an individual's signature in terms of appearance; instead the potential forger must match the dynamics of the signature. The user signs his or her name, and a similarity score is calculated by comparing the entered signature against reference signatures from the user's Personal Security card. On the basis of the similarity score, the user is accepted or rejected.

Signature enrollment and verification. Signature verification is actually a two-step process. First, a user of the process must "enroll" his or her signature into the system by signing his or her name a minimum of five times. From among these five signatures, two are chosen as the primary references and the other three are temporarily designated the most recent signatures. This information is stored in data blocks on the user's Personal Security card for future reference during the verification process. Communication of these data between the Cryptographic Adapter and the Personal Security card are protected by encryption under the session key. A description of secure sessions and session keys is given later.

The verification process, shown in Figure 4, works in the following way. The reference signatures are read by the Cryptographic Adapter from the user's Personal Security card. This information is protected by encryption under the session key shared by the Cryptographic Adpater and the Personal Security card. The user is prompted to sign his or her name with the signature verification pen. The analog data from the pen are digitized and encrypted by the Security Interface Unit and sent to the Cryptographic Adapter. These data are then compared against the primary references and a similarity score is generated. If the similarity score is above an acceptance threshold, the user's signature is verified. If the similarity score is below a closeness threshold, the user's signature is rejected. Upon acceptance, the new signature replaces the oldest signature on the user's Personal Security card.

If the similarity score is between the closeness threshold and the acceptance threshold, the new signature is compared against the three most recent signatures. If the similarity of one of these signatures to the new signature is above the acceptance threshold, the user's signature is accepted; however, the system goes into an adaptation phase.

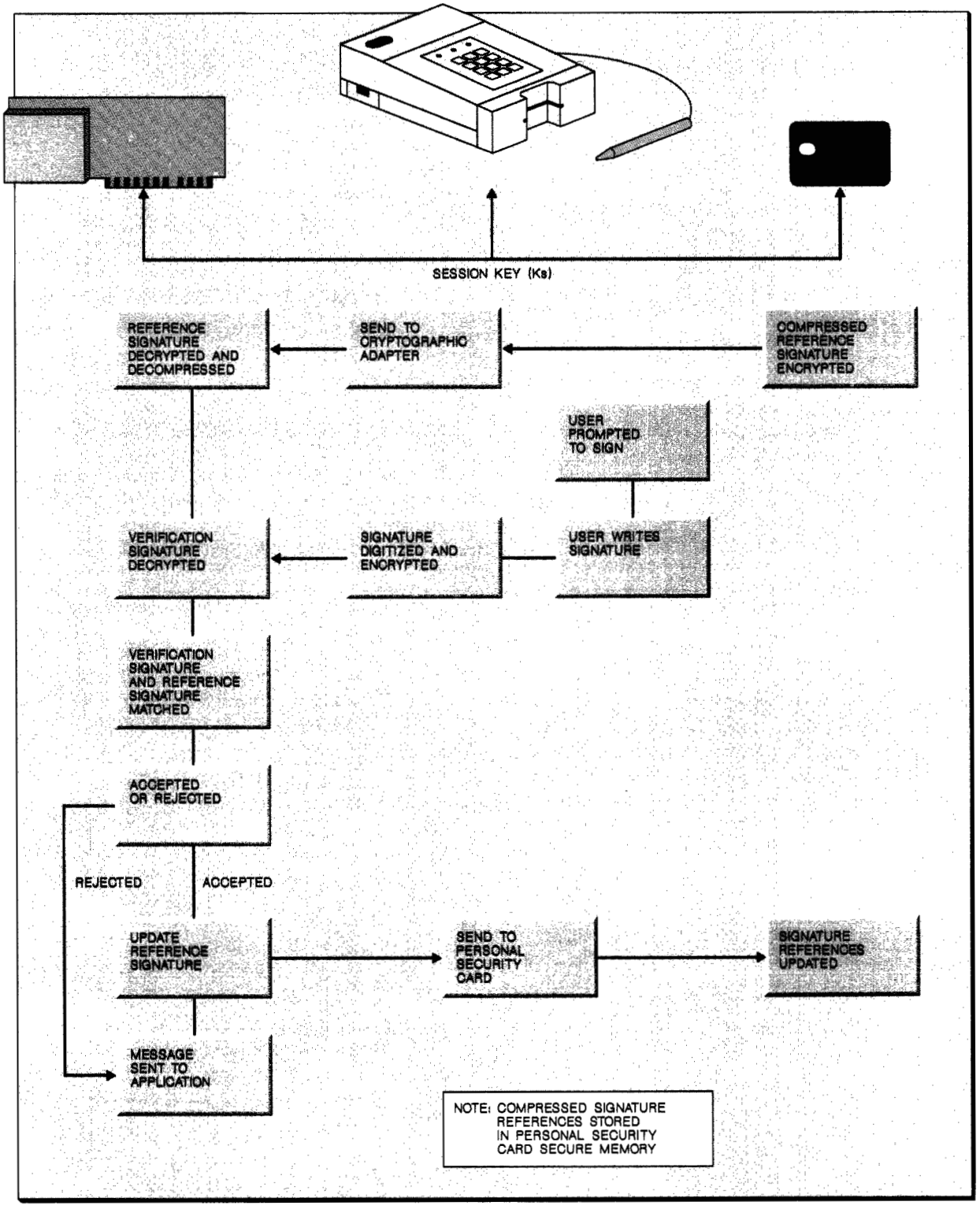
Three types of tables control user authorization within the hardware components.

In the adaptation process, the system will regenerate the primary references just as it does during the enrollment process. By means of the adaptation process, the system keeps the signature references up to date, permitting gradual changes that take place in a user's signature over time.

User authorization. User authorization is a method of determining whether a specific user has the authority to perform whatever function is being attempted. Obviously, such a determination is highly important. For example, a bank teller would not normally have the authority to perform multimillion-dollar electronic funds transfers, whereas a high-level bank official could have the authority. The Cryptographic Adapter, Security Interface Unit, and Personal Security card are highly configurable with regard to user authorization.

Three types of tables control user authorization within the hardware components. One defines the authority necessary to perform each hardware command. The second defines the authority that an individual user has. The last defines a table of 16 holidays when the execution of most commands is not allowed. All of the tables are configurable by the user if authorized.

Figure 4 The signature verification process



Each of the three devices has a *Command Configuration Table* (CCT). The CCT has a two-byte entry for each command that the hardware device supports. The first byte of each entry contains a series of flags that define certain attributes about the command as follows:

- *Command unavailable*. This option permits the command to be completely disabled under all circumstances.
- *ESS required*. This option permits the command to be processed only if a secure session is in effect between the sender of the command and the device that executes the command.
- *Enable date and time checking*. This option permits the command to be disallowed when the user is attempting to execute the command and the date or time is outside of the user's limit.
- *Initial verification required*. This option permits requiring that the user's identity be authenticated (via one of the methods discussed previously) before the command can be executed.
- *Pre-execution authentication required*. This option permits requiring the user to make an authentication each time before executing the command.
- *Exact authorization level required*. This option permits requiring the user's authority level to exactly match that of the command.

The second byte of each CCT entry contains the *required authority level* of the command. Depending on the value of the exact authorization-level-required flag, the user's authority level must either exactly match or be greater than or equal to that of the command in order for the user to be permitted to execute the command.

Each of the devices has some number of *User Data Blocks* (UDBs). The number of UDBs varies from device to device. There are three different types of UDBs as follows, although all of them contain the same type of data:

- *Regular* UDBs that correspond to individuals
- *Public* UDBs that are active when no other type of UDB is active
- *Guest* UDBs that are downloaded from the Personal Security card to both the Cryptographic Adapter and the Security Interface Unit after a Personal Security card user has been authenticated

The Cryptographic Adapter contains a public UDB, a guest UDB, and four individual UDBs. The Security Interface Unit contains a public UDB and a guest UDB, and the Personal Security card contains four individual UDBs.

A User Data Block contains a number of fields that define the authority privileges of the user. The fields are:

- *User ID*—An eight-byte field that identifies the user
- *PIN*—An eight-byte field that contains the value the user must enter in order to authenticate his or her identity when not using signature verification
- *Verification method*—A two-bit field that indicates which methods of user authentication are valid for this user. The possible values are: PIN only; signature only; PIN or signature, signature required if it is present.
- *Verification failure count*—A one-byte field that records the number of times the user has failed PIN authentication. It is reset to zero when the user's identity is successfully verified via a PIN.
- *Verification failure limit*—A one-byte field that contains the maximum number of invalid PIN authentication attempts the user is permitted before being locked out
- *User authority level*—A one-byte field that contains the authority level of the user. It is compared against the required authority level of a command whenever the user attempts to execute a command.
- *Command authorization flags*—A series of flags. There is one flag for each command. If the flag is on, the user is permitted to execute the command (given that the user passes all other authority checks). If the flag is off, the user is not permitted to execute the command.
- *Expiration date*—A date defining the last date on which the user will be permitted to use the device
- *Valid days of week*—A series of flags that define which days of the week the user is permitted to use the device
- *Time limits*—Made up of two fields: a *lower time limit* and an *upper time limit*. The user can only use the device when the current time is within these limits.

Secure sessions. Secure sessions are a concept wherein two entities temporarily connect and es-

tablish a session key that the two entities share and no other entity knows. The Cryptographic Adapter, Security Interface Unit, and Personal Security card support secure sessions among themselves and also between themselves and some outside entity. Each device supports a different number of secure sessions. In addition, some number of the secure sessions are reserved for use among the devices themselves.

The session key protects information which devices transmit across the interfaces between one another.

The Cryptographic Adapter supports three secure sessions. The Security Interface Unit supports up to eight secure sessions, and the Personal Security card supports two. The first two secure sessions supported by the Cryptographic Adapter and the Security Interface Unit are reserved, whereas just the first secure session of the Personal Security card is reserved.

The secure session establishment process results in a randomly derived secret session key that the two entities share. In the case of the secure session established among the devices, the session key has the following properties:

- It is a data-compatibility key with Encipher, Decipher, MAC-Generate, and MAC-Verify privileges.
- Only internal functions of the device can access it. It cannot be accessed as a normal key.
- It changes each time a new secure session is established. The Cryptographic Adapter and the Security Interface Unit each attempt to establish a secure session with a Personal Security card whenever it is inserted into the Security Interface Unit, and the Cryptographic Adapter and the Security Interface Unit attempt to establish a secure session whenever they are powered on.

The devices use the session key to protect information that they transmit across the interfaces between one another. The information can be protected by either encryption or "MACing." For example, the Cryptographic Adapter does not contain a clock, so to obtain the current date and time (in order to do date and time checking), it sends a READ CLOCK command to the Security Interface Unit. The Security Interface Unit appends a MAC to the date and time which it returns, and the Cryptographic Adapter then does a MAC Verify on the date and time before accepting it.

Secure session process. We now describe the process that the devices use to establish a secure session. The process consists of two main parts: (1) establishing a session key and (2) verifying that the two devices have established identical session keys.

In the following discussion, the secure session establishment process will be examined from the point of view of two devices establishing the secure session between themselves. The process is slightly modified if a third party is orchestrating the establishment of the secure session between two devices.

As a prerequisite for the establishment of the secure session, a shared key-encrypting key (KEK) must be loaded into the appropriate entry in the KEK table of each device.

In establishing the session key, in all cases of secure-session establishment, one device has to be in control of the process and driving it. For the secure session between either the Cryptographic Adapter and the Security Interface Unit or the Cryptographic Adapter and the Personal Security card, the Cryptographic Adapter is the controlling device. For the secure session between the Security Interface Unit and the Personal Security card, the Security Interface Unit is the controlling device.

First, the controlling device generates a random eight-byte key using its random number generator. This random key is the session key. The controlling device stores the session key in the appropriate entry in its session key table. The entry used depends on the device with which the controlling device is attempting to establish the secure session.

Next, the controlling device will triple encrypt the session key under the appropriate KEK stored in its KEK table. It will use a control vector that specifies data compatibility with Encipher, Decipher, MAC-Generate, and MAC-Verify privileges.

Finally, it will load the session key into the device with which it is attempting to establish the secure session via the LOAD SESSION KEY command. The parameters for this command are the encrypted session key, the control vector, the entry number in the KEK table of the target device to use for decrypting, and the entry number in the session key table of the target device in which to store the clear session key. The target device will decrypt the session key using the supposedly shared KEK and store it into its session key table.

After the session key is loaded, both devices need to verify that in fact, they do share the same session key. This verification is done by using a three-step process performed first in one direction and then in the other.

Assume that the two devices are called A and B, and further assume that A is the controlling device. The steps in the process work as follows:

1. Device A sends a GENERATE CHALLENGE QUANTITY command to device B. Device B returns an eight-byte random number in the clear (i.e., not encrypted). B also saves this random number for later use.
2. Device A encrypts the random number using the session key that it supposedly shares with B. This step is effectively a GENERATE CHALLENGE RESPONSE command.
3. Device A sends the encrypted random number back to B as part of a VERIFY CHALLENGE RESPONSE command. B decrypts the value and compares it to the original random number it generated in Step 1. It then returns a response to A that indicates whether the comparison matched.

If the comparison in Step 3 matched, device B knows that device A shares the same session key as it does; however, A cannot be positive that the session key of B is identical to its own. Therefore, the process is repeated in the following manner:

1. Device A generates an eight-byte random number and saves it for later use. This step is effectively a GENERATE CHALLENGE QUANTITY command.
2. Device A sends the clear random number to device B as part of a GENERATE CHALLENGE RESPONSE command. B encrypts the random number with the session key that it shares with A and returns the result to A.
3. Device A decrypts the value received from B with the session key, which it supposedly shares, and compares the result with the original random number that it had saved in Step 1.

If the comparison in Step 3 was a match, device A knows that device B shares the same session key as it does, and the secure session establishment process is complete.

Summary

The IBM Transaction Security System has been described and some of the challanges associated with its development have been discussed. The development of a security system presents unique challanges for which there exists no exact paradigm. Common sense along with good solid engineering judgment provide the best guidance for such an undertaking.

Such a system will continue to evolve with additional enhancements and improvements in response to customer demand. The same principles will guide the development of those enhancements as were used for the original work.

Acknowledgments

The authors of this paper wish to acknowledge the following people who made significant contributions to the development of the Transaction Security System: S. G. Aden, T. W. Arnold, G. Bourbeau, T. J. Chainer, S. Deskevich, M. B. Forbes, G. B. Fryer, D. Hrelic, D. B. Jacobs, D. B. Johnson, M. R. Kelly, J. W. Lamb, A. V. Le, S. M. Matyas, E. H. Nachtigall, S. W. Neckyfarow, R. Prymak, W. S. Rohland, S. L. Schifano, P. D. Smith, D. J. Sundberg, A. B. Wadia, S. H. Weingart, and S. E. Wince.

vanced Interactive Executive, RISC System/6000, AS/400, PC/XT, MVS/XA, and MVS/ESA are trademarks, of International Business Machines Corporation.

Cited references and notes

1. See the appropriate announcement letters: No. 189-174 (IBM 4754 Interface Unit, 4755 Cryptographic Adapter, Personal Security Card and Signature Verification Pen), No. 189-171 (IBM 4753 Network Security Facility), No. 289-585 (NSP MVS Host Support Program Product No. 5706-028).
2. *Common Cryptogtaphic Architecture, Cryptographic Programming Interface,* SC40-1675, IBM Corporation; available through IBM branch offices.
3. D. B. Johnson et al., "Common Cryptographic Architecture Cryptographic Application Programming Interface," *IBM Systems Journal* **30**, No. 2, 130–150 (1991, this issue).
4. IBM has issued a statement of direction for support of System/88 and Operating System/2.
5. D. B. Johnson and G. M. Dolan, "Transaction Security System Extensions to the Common Cryptographic Architecture," *IBM Systems Journal* **30**, No. 2, 230–243 (1991, this issue).
6. *Transaction Security System Programming Guide and Reference*, SC31-2934, IBM Corporation; available through IBM branch offices.
7. 3848-CUSP stands for the IBM 3848 Cryptographic Unit and Cryptographic Unit Support—OS/VS2 MVS Program Product No. 5740-XY6.
8. The IBM Programmed Cryptographic Facility Program Product No. 5740-XY5.
9. For information about the Data Encryption Algorithm, see ANSI X3.92 or FIPS-46.
10. For additional information on the DEA, see S. M. Matyas, "Key Handling with Control Vectors," *IBM Systems Journal* **30**, No. 2, 151–174 (1991, this issue).
11. S. M. Matyas and C. H. Meyer, *Cryptography: A New Dimension in Computer Security,* John Wiley & Sons, Inc., New York (1982), p. 115.
12. W. F. Ehrsam, S. M. Matyas, C. H. Meyer, and W. L. Tuchman, "A Cryptographic Key Management Scheme for Implementing the Data Encryption Standard," *IBM Systems Journal* **17**, No. 2, 106–125 (1978).
13. For more information about the Message Authentication Code, see ANSI X9.9 or ISO DP 8730.
14. *Op. cit.*, Reference 10.
15. S. Weingart, "Physical Security for the μABYSS System," *Proceedings of the 1987 Symposium on Security and Privacy* (1987), pp. 52–58.
16. CBC is the Cipher Block Chaining mode of the DEA; see ANSI X3.106.
17. In the discussion that follows in the text, the term "device" is understood to signify one of the components of the Transaction Security System.

General references

D. G. Abraham, G. P. Double, and S. W. Neckyfarow, *Secure Component Authentization System,* U.S. Patent No. 4,799,061 (January 17, 1989).

IBM Cryptographic Subsystem Concepts and Facilities, GC22-9063, IBM Corporation; available through IBM branch offices.

IBM Data Security Through Cryptography, GC22-9062, IBM Corporation; available through IBM branch offices.

IBM Transaction Security System: General Information and Planning Guide, GA34-2137, IBM Corporation; available through IBM branch offices.

IBM Transaction Security System: Programming Guide and Reference, GC31-2934, IBM Corporation; available through IBM branch offices.

IBM Work Station Security Services Installation and Operating Guide, SA34-2141, IBM Corporation; available through IBM branch offices.

IBM Work Station Security Services Licensed Program Specification, GC31-2720, IBM Corporation; available through IBM branch offices.

IBM 4753 Network Security Processor Installation and Operating Guide, SA34-2139, IBM Corporation; available through IBM branch offices.

IBM 4753 Network Security Processor MVS Support Program Installation and Operating Guide, SA34-2139, IBM Corporation; available through IBM branch offices.

IBM 4753 Network Security Processor MVS Support Program Licensed Program Specification, GC31-2933, IBM Corporation; available through IBM branch offices.

USA Federal Information Processing Standard, Data Encryption Standard, 46-1-1988, National Bureau of Standards (now NIST), U.S. Department of Commerce, Washington.

Dennis G. Abraham *IBM Services Sector Division, 1001 W. T. Harris Boulevard, Charlotte, North Carolina 28257.*

Mr. Abraham is a Senior Technical Staff Member in the Security System Architecture area, where he has been a leader in establishing the architecture and function definitions for the Transaction Security System. He graduated from Fairleigh Dickinson University with a B.S.E.E. degree in 1964, and he received his M.S.E.E. from Syracuse University in 1972. He joined IBM in June 1964 at Endicott, New York, where he held assignments in various product and service groups, among which were circuit design and logic design with a strong speciality in servomechanisms, including a special expertise in stepper motor control and design. He was the lead architect of the IBM 3890 OCR machine and, after moving to Charlotte in 1979, he worked in developing image technology as it applies to check processing. After an assignment in the former National Marketing Division headquarters where he provided technical expertise for the marketing force, Mr. Abraham joined the advanced technology group and was assigned to develop a security strategy and architecture for the Consumer Systems Business Unit. This work led to the development of the Transaction Security System and the Common Cryptographic Architecture. He holds nine issued patents, ten patents on file, and 23 published invention disclosures.

George M. Dolan *IBM Services Sector Division, 1001 W. T. Harris Boulevard, Charlotte, North Carolina 28257.*

Mr. Dolan graduated from Lehigh University with a B.S. in electrical engineering. Since joining IBM at Endicott, New York, in 1961, he has had design responsibilities for various communications hardware and software products, which in recent years have been principally for the worldwide finance industry. Mr. Dolan is a senior engineer in the IBM Secure Workstation Development department. His work on the Transaction Security System has involved integrating cryptographic

processors into IBM PS/2 and MVS systems, and integrating the result into customer applications for the protection of data and user identification. His responsibilities include specifying the user programming interface and software structure in support of the Transaction Security System.

Glen P. Double *IBM Services Sector Division, 1001 W. T. Harris Boulevard, Charlotte, North Carolina 28257.* Mr. Double is an advisory engineer/physicist and is currently responsible for physical security on the Transaction Security System. He received a B.S. degree with honors in engineering physics from the University of Toledo. He attended Wayne State University under a National Science Foundation traineeship and received an M.S. in solid state physics. Before his present assignment in Charlotte, he worked in various areas at the IBM facilities in East Fishkill, New York, and the former Instrument Systems Division. He has a broad background in integrated circuits and electronic packaging technologies, magnetics, electro-optics, lasers, and radiation effects in solids.

James V. Stevens *IBM Services Sector Division, 1001 W. T. Harris Boulevard, Charlotte, North Carolina 28257.* Mr. Stevens is a staff programmer in Secure Workstation Development. He joined IBM in Charlotte in 1983 after receiving a B.S. in computer science from the University of Missouri at Rolla. He is nearing completion of an M.S. in computer science from the University of North Carolina at Charlotte. For the past three years, he has been involved with the design and development of the device drivers and microcode for the IBM 4755 Cryptographic Adapter. His current assignment is the coordination of the OS/2 software development effort for the Transaction Security System.

Reprint Order No. G321-5431.