To the Graduate Council:

I am submitting herewith a thesis written by William Chester Hood entitled "Encryption as a File Security Measure in Large Operating Systems." I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Michael G. Thomason

Michael G. Thomason, Major Professor

We have read this thesis and recommend its acceptance:

Accepted for the Graduate Council:

Vice Chancellor
Graduate Studies and Research

ENCRYPTION AS A FILE SECURITY MEASURE IN LARGE OPERATING SYSTEMS

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

William Chester Hood

August 1980

ABSTRACT

The purpose of this study was to examine the use of encryption as a file security measure in large operating systems. The IBM Corporation's Operating System/Virtual Storage 2 Multiple Virtual Storages was chosen because of IBM's involvement in development of the National Bureau of Standards Data Encryption Standard (DES) and the availability of both software and hardware products to implement the DES under the MVS operating system.

A general review of the need for security, physical security measures, personnel practices, and administrative procedures was used to establish a foundation for the use of computer technology to supplement all other techniques. Hardware and software technologies were reviewed and several existing computer systems were briefly examined from a security standpoint. Historical developments leading up to the modern day interest in encryption were summarized. The DES was described and major criticisms of this algorithm enumerated, prior to examination of IBM's implementations and support of the algorithm in software and hardware.

The Programmed Cryptographic Facility, a software version of the DES, was found to have two major limitations: lack of accessibility to the high level language programmer and inability to achieve high speed. Previous software product developments have suggested that the first limitation may disappear in the future; the second can only be overcome by hardware.

The 3848 Cryptographic Unit is the primary hardware device which implements the DES and was also found to have two limitations. It is a channel-attached device, which increases the load on the input/output subsystem, and it can handle only four kilobytes of data at one time. Recommendations were made to incorporate the DES circuits into the input/output channel and also to provide a pair of encryption instructions for the central processing unit.

The increasing use of computers and data processing services in sensitive areas such as electronic funds transfer systems requires that some form of data encryption be included to assure adequate security. Developments in areas such as very large scale integration, data base management systems, international data flows, and language extensions are likely to provide capabilities to meet this requirement. Encryption is a powerful capability that must be included in security plans, and security must be a major concern in the design of any new system.

TABLE OF CONTENTS

CHAPTER I

THE NEED FOR SECURITY

Throughout history man has felt a need to protect his assets. Those possessions which have required protection include money, property, and family. A prudent man sought something called "security" for these assets: that is, a sense of safety from danger, injury, and damage. Security measures taken depended on the form of the asset and the type of risk involved. For example, monetary assets were placed under combination lock at a banking institution, thus transferring the responsibility, and the risk of theft, to someone else who was paid to assume that responsibility and risk. A home or other structure was normally protected by locks on all doors and windows; sophisticated burglar alarms were added if necessary. Electrified fences with barbed wire were used to restrict close access to the structure. Family members were protected from physical assault or kidnapping by bodyguards.

In a data processing environment, security is also a significant concern. Valuable assets and sensitive resources, which may take the form of hardware, software, or data, require protection against loss or destruction. Some legislation which requires access controls to prevent misuse of accumulated data exists, and certainly much more will emerge.

Threats against data processing can be divided into two categories: forces (water, lightning, wind, fire, earthquake, and other acts of God) and people (those with the skills, knowledge, resources, and appropriate access). (90) Many installations are substantially reducing the risks of threats from forces by moving their data processing equipment from glass-enclosed "fish-bowl" showplaces to steel-reinforced concrete block-houses. Unfortunately, the risks of threats from people are not usually given the same attention because losses sustained due to misuse or abuse by people are not as visible as physical damage. Most corporations are understandably reluctant to publicly discuss any such occurrences.

Parker has defined two types of computer abuse committed by people: "inspec" and "nonspec." (90) "Inspec" abuse occurs when the computer is used within specifications, but false data is entered. An example of this type of abuse occurred at the Union Dime Bank in New York. (116) A chief teller embezzled approximately one and one half million dollars in cash by transferring money among savings accounts. He had no computer science background, only special authority due to his position. This man was discovered after a raid on a book-making establishment where he was losing some thirty thousand dollars daily betting on horse races while earning only eleven thousand dollars yearly. As a humanitarian gesture, he first took only twenty thousand dollars, later forty thousand dollars, from each account so that the losses would be covered by the Federal Deposit Insurance Corporation. "Nonspec" abuse occurs when unauthorized program changes are made to alter data. A head accountant set up an independent service bureau

(not in his own name) and later recommended that his employers contract with "his" service bureau for computerized accounting systems. Disappointed that a promised bonus was not what it should have been, he developed an econometric model of the company's operation to siphon off money in the form of higher costs paid to dummy companies. He even made suggestions to management to reduce costs; of course, improvements would always follow. Later, he became worried that the company would be investigated by Internal Revenue Service auditors. This man expected to get off lightly and began to salt executives' personal bank accounts, but he had to overdraw one of his dummy companies by seventy-eight thousand dollars to arouse suspicion by a bank officer. To his surprise, company management signed a criminal complaint and he served five and a half years of a ten-year prison sentence in San Quentin Penitentiary. (116)

The computer can be used in most types of crime--the most common being embezzlement, fraud, and theft; but it is also possible to involve the computer in blackmail, extortion, terrorism, espionage, and sabotage. The computer may itself be the victim of crimes ranging from malicious mischief by computer operators and programmers to destruction of hardware or storage media by vandals and terrorist groups. The computer may create an environment for crime: data may be copied, but the original data remains intact; therefore, there may be no evidence of theft. The computer may also be a silent accomplice in crime: in a case of embezzlement, for example, a few minutes of effort by a programmer may illegally divert thousands of dollars. (90, 116)

Computer crime has a nice clean quality: there is normally no overt violence, just an intellectual challenge for some programmer to succeed. The perpetrator has a certain anonymity, and the crime is often difficult to detect. Victimized corporations are usually unwilling to talk about these frauds and often pursue less-than-intensive litigation. (116) Evidence to support prosecution cannot be easily collected due to the lack of knowledge of the attorneys. The cases that do make it into court find judges and juries that do not understand what is presented to them. The results are numerous undetected and unpunished computer crimes. Corporations are concerned about publicity if they choose to prosecute. However, any negative publicity is usually temporary, and prosecution has a positive effect on other employees because the detected crimes do obtain widespread attention. One famous case was that of a bright young engineering student who stole over one million dollars' worth of equipment from the Pacific Telephone and Telegraph Company. (116) A two hundred fifty thousand dollar civil suit was settled for eighty-five hundred dollars, and this man is now a computer security consultant with a thriving business.

Most companies that acquired the early computers were proud of their new status symbol and generously offered tours to customers, stockholders, and the public. Of course, no employee was denied access to the computing facilities; in fact, employees were encouraged to learn as much as they could about this new technology. Management perceived no threats as the computer replaced traditional manual record-keeping functions with more accurate administrative controls

over inventories and accounts payable. Soon the computer became critically important to day-to-day operation and, therefore, vulnerable to threats from political activists or disgruntled employees as well as such natural sources as floods and fires. (19)

This new environment was causing problems that management had not previously encountered and could not handle. Internal threats from programmers and operators ranged from stealing a little computer time or selling programs to large-scale theft by manipulating manufacturing shrinkage allowances. External threats from burglars (or organized crime) took on new dimensions as data processing employees became involved in tampering with the computer. Most computer manufacturers have offered their customers only limited assistance in preventing these types of abuse. Crime insurance premiums continued to rise; difficulty in obtaining or renewing crime insurance and steadily increasing deductible amounts have led many companies to self-insure, thus spending money formerly used to pay insurance premiums to purchase protection against the most seriously perceived threats. (19, 73)

Stringent security measures are imposed on classified data processing systems used by government agencies and their contractors. (4) These requirements are necessary because today greater reliance is placed on the results of computer use in sensitive functional areas. The user population is typically composed of persons who do not all possess either the same level of security clearance or the same "need-to-know" requirements. Currently available off-the-shelf operating systems are recognized as having

potential security exposures when several programs are executing concurrently as in the typical multiprogramming environment. There is some limited protection against accidental violations, but little protection against an intentional threat to security controls.

Typical government security requirements might include a number of general purpose measures designed to restrict access to classified data by a user without the proper security clearance or "need-to-know." (30) Access would be defined to include retrieval of classified data, insertion of false or misleading information into the system, or subversion of security measures. Personnel security would insure that only authorized operators, system programmers, maintenance personnel, and users are allowed access to any portion of the computer system. Physical security measures would be applied to the central computer facility and any remote terminal areas to prevent access by unauthorized personnel who are not properly escorted. All communications lines would be protected by approved cryptographic devices. Computer hardware itself must prevent the user from executing privileged instructions or referencing memory not assigned to him by the operating system. Operating system software must prevent a user from performing executive control functions, must verify file access by an authentication mechanism, and must log attempts to penetrate the system or otherwise bypass security controls. Administrative security measures would be prescribed for secure handling of classified information by personnel, including access to computer facilities, appropriate marking of classified output data, adequate erasure of magnetic media, and destruction of

classified material.

In the course of their everyday operation, some businesses accumulate vast amounts of data on individuals. For example, credit bureaus routinely ask for, and nearly always receive, a large amount of information in the process of approving a credit application. Modern computers and data processing techniques have enabled the credit bureaus to collect and process more information than ever before. Government is following the pattern and example set by private industry, but much faster and on a larger scale. The Federal Bureau of Investigation is exchanging data with state and local police agencies as the National Crime Information Center (NCIC) becomes the cornerstone of a nationwide information-swapping network. Perhaps in the future, each individual will be assigned a unique identification number at birth to be used for all future transactions, both with the government (social security, taxes, military service) and with private enterprise (banking, education). This endeavor could have life-saving implications: if a person were injured far from his home, medical personnel could retrieve the appropriate portions of his medical history to prevent errors in treatment due to previously existing, but otherwise unknown, conditions.

A brief consideration of the implications of the data banks mentioned above reveals that a large amount of sensitive personal information can be assimilated in a short period of time. Most of these information collecting efforts are well-intentioned and may achieve some degree of success, but an individual's right to privacy must be safeguarded. In some cases, there is considerable reluctance

on the part of these information-collecting agencies and bureaus to erase or eliminate data that is in error. Furthermore, the "verdict" handed out by the computer may come at the hands of an inexperienced or inept programmer who has inadequate supervision and direction from his management and who is unconcerned about the human being represented by the data records. It is not enough to expect (or demand) superior standards of conduct and moral judgment from the people involved in data processing; therefore, there must be regulations prescribing limits on the data which may be collected, restrictions on its use and dissemination, and the necessary periodic house-cleaning to purge records that are no longer correct or relevant to current conditions.

The Privacy Act of 1974 reaffirms an individual's constitutional right to privacy by protecting him from invasions of privacy by agencies of the federal government. (109) An individual may inspect his records, make copies of most of them, and place a statement in the file if he disputes the information that is there. The agencies are required to provide adequate safeguards to prevent misuse; furthermore, officials are personally liable for violations and subject to prosecution. The National Bureau of Standards has prepared guidelines for compliance with the Privacy Act. One shortcoming of this legislation is that state and local governments do not come under its jurisdiction. There is, however, other legislation for protection in some limited areas. (12) For example, the Fair Credit Reporting Act is an attempt by the federal law-makers to help individuals in their requests for credit. (108) It provides for access to files

maintained by these agencies and specifies an error-correcting procedure, but there is no mention of a time limit that undesirable information can be kept on file or any standards for the proper collection of information. A more far-reaching statute was introduced in the 94th Congress by Goldwater and Koch. H. R. 1984 was not passed, but it provided the groundwork for the development of further legislation.

The computer does not commit crimes alone; man is always the controller. These employees are not machines; their competence, loyalty, and integrity are system parameters that cannot be quantitatively measured. (9) The hardware and software remains vulnerable to operators, programmers, and maintenance men. These personnel are not orthodox company staff members, but rather, professional technologists and technicians (often underpaid) with undistinguished backgrounds. (75) The perpetrators of many crimes are "bright, eager, highly motivated, courageous, adventuresome, and qualified people, willing to accept a technical challenge." (90) It is not a coincidence that these characteristics are among those most desirable in data processing employees. For some of these people, merely saying that a system is safe from penetration evokes a challenge response that overshadows any question of morality. (92) However, these people are amateurs; the professional criminals have not yet acquired the necessary expertise to commit computer crimes and may never do so. It is much cheaper and easier to buy off a programmer through blackmail, bribery, or even extortion. In this manner, both the professional criminal and the professional

programmer stay in their respective elements and can operate for very high stakes. In England, Guy Parker, who has uncovered many computer crimes, went into hiding after receiving numerous threats on his life and those of his family. (75)

Despite this environment of potential crime, industry and government are highly dependent on the computer. The banking industry, credit sales, corporations (sales, inventories, accounts receivable and accounts payable), the Social Security Administration, the Internal Revenue Service, and the Bureau of the Census would all be severly impaired if forced to operate for even one day without their computers. (116) Therefore, it is necessary for an installation to be constantly alert to threats to data processing resources. Various types of security will be discussed in Chapter II, but computer security is first and foremost a people problem which cannot be corrected by rules, regulations, or legislation. Technology can be used to reduce the number of people who must be in positions of trust (who can therefore commit a crime or an abuse) and also to reduce the degree of trust necessarily placed in these people. (90) It is important to remember that the determined technical penetrator will get through any safeguards. (42) It may, however, be possible to make the cost of circumventing these safeguards greater than the value of the benefits gained and thus prevent unauthorized acts before they occur.

CHAPTER II

TYPES OF SECURITY

There are many types of security, and their degree of effectiveness varies considerably from one installation to another. In a large corporation, there is usually a full-time security director (possessing superior administrative and diplomatic skills) who reports to the chief executive officer. Although it is the chief executive who is ultimately responsible to the shareholders of the corporation, the security director will be the primary individual concerned with safeguarding the corporation's assets. (73) The security director and his staff must be given the authority and sufficient resources (people and machines) for executing all security arrangements to fulfill their responsibilities. One goal must be to make each employee aware of the need for security and the commitment of management to support all security measures. Constant monitoring is required to insure compliance as a standard operating procedure. After each person has accepted the fact that he is involved in this problem, then the implementation of security policies may proceed more smoothly. (103)

Prevention should be the major goal of any security program, but not all problems can be anticipated. In some cases, the cost of preventing a security breach may exceed the cost of recovery and therefore not be economically justified. It is also important that precautions taken do not impose impossible constraints on normal operation. Uncertainty and human fallibility will contribute to

failure of security procedures, as the most carefully prepared plans may collapse at precisely the moment they were intended to function. (19) A security program is started by enumerating all assets (cash, goods, machines) and the risks to each. A quantitative risk assessment program can then be used to identify the most critical resources and to select an appropriate combination of safeguards to establish sufficient risk reduction for each at a reasonable cost. (73, 80)

There are two popular, but false, threats which some unscrupulous firms or salesmen use to exploit nervous executives; accordingly, it is important to identify these circumstances so that they will be properly eliminated from consideration before examining security measures for more reasonable types of risk. (73) One of these is the story of a tour group of boy scouts who destroyed a fifty-thousand reel tape library in a matter of minutes with small magnets about the size of a quarter. A study performed by the Stanford Research Institute proved that if a magnet of this size was to do any damage, the tape would have to be unwound and the magnet physically touched to the tape. Simple arithmetic indicates that at the rate of one reel per second, this process would have taken fourteen hours. The second story is that of the "superspy" truck that pulls up across the street to pluck electromagnetic emanations out of the air and reconstruct the data being processed by the computer. When pressured for details, the American firm that circulated this story could offer no substantiating evidence for this electronic feat. These two examples show that if a firm decides to purchase security products or services, more care must

be exercised than with other products or services. The ensuing false sense of security could be disastrous, as well as a tremendous waste of money.

Human ingenuity knows no bounds: after one elementary course on computers, anyone can know how and what to destroy to render a computer inoperable. (113) If vandals or terrorists can gain access to a data processing establishment, the computer can be held hostage. Therefore, it is appropriate to first consider physical security.

Designing and building a data processing facility from the ground up with the opportunity to impress security, not aesthetics, on the architect is a luxury most firms do not enjoy. (73) An ideal location for the facility would be well away from potentially explosive locations (such as airports or chemical plants) and on a busy street close to fire and police protection. It is also good common sense to avoid flash flood plains and earthquake areas. Often a portion of an existing building will be leased as a computer facility. This is undesirable because the lessee has little or no control over a constant flow of maintenance men, janitors, and customers of other firms in the building. The most common location for computer facilities is within an existing building owned by the company; but regardless of the structure, an internal location on an upper floor away from main traffic areas is preferred. The goal is to avoid accessibility, especially from outside walls and entrances. (19) The best example of physical security may be the San Luis County (California) computer center because it is located in the old county jail. (47)

Entry and exit doors are often assumed to be adequate because they are locked. Another concern is the addition of alarmed crash-bars on some doors so they can qualify as emergency exits. Completely overlooked is the fact that keys may be duplicated by construction workers or employees. Locks and keys may be purchased from a lock company with the guarantee that the key pattern is not available as a standard commercial blank to prevent duplication at the corner hardware store. (73, 113) After construction is complete, all locks should be changed and outside door knobs removed from non-entrance doors. Programmable digital locks or digitally encoded cards may be used to restrict passage through internal doors. (18, 19)

This confinement of the computer behind walls, floor, and ceiling of steel-reinforced concrete with limited entrances and exits must be balanced against the most serious physical threat to any structure: fire. Methods to prevent fire and its rapid spread include flame-resistant carpets and locating storage rooms for forms and other supplies away from the machine room. However, much more effort is expended to detect and fight fire after it breaks out than to prevent it in the first place. Elaborate electronic detection systems can report to the nearest fire station and discharge chemicals such as Halon 1301 to extinguish a blaze. Fire extinguishers, power-off switches, and waterproof covers should be strategically located for quick access in an emergency. Of course, fire prevention and detection efforts should include all adjacent rooms (around, above, and below) to be effective. (19, 90, 113)

The computer is also dependent on an adequate power supply and air conditioning. The sources of these critical utilities should be independent of the remainder of the building, and their entrances should be hidden from public view, much like the computer is hidden. An uninterruptible power source (UPS) can be used to isolate the computer from fluctuations in the line voltage. Batteries can maintain power for up to ninety minutes; if service is required for a longer period of time during a power outage, a diesel generator should be installed. (19, 90)

Surveillance devices can be used to guard against intrusion during off-hours. Active or passive motion detectors, photographic or closed circuit television cameras, and intrusion detectors may be used if a reliable power source is available. Delivery personnel should not be permitted to enter the building and employees should be prohibited from receiving packages at company facilities. Visitors should be required to wear a visitor's badge and be escorted at all times, even to the rest room. Unescorted strangers should be tactfully and discreetly questioned. (18, 43, 73) One of the best security measures is placing a guard, who may double as a receptionist, at or near building entrances. He can verify that employees have a need to enter the computer center and make positive identification of vendors, building maintenance personnel, and others who are not employees of the computer center. (19) A guard is also an active observer in two aspects: he may detect that something is wrong (that cannot be predicted ahead of time) and he may use force to resist a potential violation of security.

Employees are the key factor to success of any data processing endeavor, and security is no exception. A solid organizational structure provides the framework to minimize opportunities for security breaches by employees within this sensitive area. Appropriate administrative procedures covering all aspects of operation and behavior should be established and well documented in a standards manual or employee handbook. Job descriptions with clear designations of duties, responsibilities, and authorities can reduce opportunities for collusion and access to unauthorized areas. Dependable and stable employees can never be guaranteed, but initial security checks of references, military records, law enforcement agencies, and educational records may eliminate some undesirables. (19) The careful selection of employees is a one-time cost. An application for employment tailored to the position can identify those critical areas that deserve special attention. It is important to note that it is illegal to ask certain questions; for example, it may not be possible to refuse employment to a convicted felon upon learning that fact. If lengthy checks are required, a probationary period may be used and the employee discharged without trauma to either the employee or the company. During this period, any investment, such as education costs, should be withheld, and the employee should be denied access to any sensitive resources. If an employee has been bonded, it may be a deterrent to dishonesty because he has a reputation to uphold. (73)

The mobility of the professional data processing work force indicates that lasting loyalties to a particular employer are not necessarily developed. Communication is important to insure that employees continue to be persons of integrity. Security awareness may be maintained by regular briefings by the head of security or a high-ranking executive to continually remind employees that management relies on them, as individuals, to preserve the integrity of the installation. Trained supervisors should continually observe employees for potential problems. An educational or professional development program may help to keep up morale, and maintain effectiveness and efficiency. (19, 43)

Termination of employees, both voluntary and involuntary, is a fact of life. In data processing, employees should be terminated immediately without the option of working a customary two-week notice. All keys and identification cards should be surrendered; locks and passwords should be changed. At the exit interview, the employee should be queried about any dishonesty, theft of supplies, leaking of trade secrets, or kickbacks. If he has signed any non-disclosure agreements, he should be reminded of his legal responsibility to guard proprietary information. (19, 73)

There are two weaknesses readily exploited by the professional criminal: inadequate separation of duties and lack of sufficient physical controls over facilities and supplies. (116) The simplest form of embezzlement may occur when a single individual is authorized to certify input that he has prepared for the computer. This situation contains a serious flaw because there are no checks and

balances to insure adequate review by a responsible person. (113) Any competent auditor involved in development of the system would detect this type of weakness and demand separation of functions and rotation of duties. The likelihood of criminal exploitation of input data can be reduced by limiting access to data and using time-honored techniques such as batch numbers and hash totals. To discourage data or program manipulation, computer operators should only have access to documentation required to run programs on the computer. Access to the machine room by others, especially programmers, should also be limited to prevent tampering with input data or program execution. (18, 73, 92)

In some installations, input data preparation and submission to the computer are rigorously controlled but programmers and program changes are ignored. The potential for damage by an unauthorized program change is virtually unlimited. Appropriate separation of functions may be more difficult to achieve in a program development and maintenance environment, but additional benefits may accrue from this organizational structure in offsetting losses due to constant turnover of employees and changing responsibilities. All program changes should be verified by another individual; programmers should not be permitted to do acceptance testing on their own programs, particularly after changes have been made to correct errors or enhance function. Published documentation standards should be established and enforced. Appropriate forms should be used to maintain a history of authorized program changes and to facilitate periodic audits to insure that no unauthorized modifications have occurred. (18, 19, 43)

The second weakness exploited by criminals is perhaps more in their traditional style of gaining access to facilities or supplies. Controls on tape and disk libraries should severely limit access to these sensitive areas. Remote storage sites should be subject to the same environment and protection afforded the computer room. Tapes returned to the scratch pool should be degaussed or overwritten to obliterate prior contents. Serious consideration should be given to avoiding the common practice of conspicuously affixing "CLASSIFIED" labels to tapes or disks. This technique does provide a constant security reminder to those personnel who handle these items, but also immediately identifies sensitive materials to intruders. (19)

Data processing forms are also subject to theft: blank checks, bills of lading, and stock certificates have always been a prime target of thieves. The procedures for handling these types of forms should include bonded couriers, receipts, and accounting for spoiled or unused forms. Check handling procedures should be constantly reviewed, tested, and strictly enforced. Discarded printout, punched cards, and continuous-form carbon should be either shredded or incinerated. (19)

A recovery/restart plan should also be included in the standards manual to provide for orderly reaction during a major physical disaster. It is desirable to be able to quickly remove critical software and data (disks and tapes) from the machine room or library and transfer these items to a safe location. Regardless of the effectiveness and speed of the above procedures, duplicates of critical software, data (master files and transaction files), and

pertinent documentation should be stored off-site. Emergency arrangements should include adequate quantities of forms and supplies to assure continued operations. Alternate plans, such as printing headings on stock paper to convert it to temporarily acceptable special forms, should be considered. (19, 115)

Insurance companies have long marketed policies to cover data processing equipment and media from perils such as fires, floods, and earthquakes. Usually an endorsement to the policy will extend media coverage to include the reconstruction of data. These policies normally cover the loss in income (or cost) of a business interruption due to suspended data processing operations. Another covered expense is extra expenditures required to contract with a service bureau or to hire temporary personnel to recover from a disaster. Some executives have declined to purchase comprehensive insurance policies such as those described above, preferring a gentlemen's agreement with the firm down the street. Most companies will cooperate on modest requests for a few hours of computer time, but must refuse long-term requests for many weeks of twenty-four hour a day, seven day a week support. Lack of insurance in this type of situation could be a costly experience. (19)

Management emphasis is often centered on making systems operational rather than insisting on adequate systems controls. (116) To be effective, security must be a major top management concern, and security measures must be constantly tested and enforced. Written reports and irregular spot checks must be used to continually attempt to close the gap between written procedures and

actual employee performance. Insurance policies are necessary to cover physical damage but are not a solution to security violations because nearly all policies will exclude any dishonest or criminal acts by employees. The physical security, personnel practices, and administrative procedures described in this chapter are all basic requirements of a well-managed organization, but they are significantly dependent upon people for their effectiveness. Hardware and software technology occupies an important place in a data processing environment and can augment the security measures described above by functioning as an impartial partner to any use of the computer. Chapter III will review the security capabilities of hardware and software, as well as examine current work in these areas.

CHAPTER III

HARDWARE AND SOFTWARE TECHNOLOGY

The original motivation for including hardware and software protection mechanisms in computer systems was primarily to prevent one user's unintentional errors (or malicious intent) from harming another user and, to a lesser degree, himself. These mechanisms were intended to provide data security: protection against unauthorized intentional or accidental disclosure, modification, or destruction of data. Additionally, some of the software protection mechanisms existed to guard against the degradation of service due to unfair consumption of resources such as CPU time or public, temporary disk space. The ultimate degradation is to cause the system to crash, making it unavailable to all users. (68, 89, 110) Typically, multiple protection mechanisms are found in a given hardware and software environment. They are not necessarily related, but in most cases their functions will overlap. (68)

It is necessary to define two terms before proceeding. A subject is a user of the computer system and is considered an active entity: a process, procedure, or user job. An object is an identifiable resource, generally the target of a subject: data files, programs, or hardware resources such as real memory or space on a disk volume. In this chapter, references to specific hardware and software will be avoided in favor of identifying methods and techniques. Specific hardware and software will be addressed in Chapter IV.

There are several principles which should be considered in the design of computer hardware and software: (47, 98)

1. The impact of the inevitable occasional security break should be minimized by requiring that any access control mechanism default to access denial. In this manner, design or implementation flaws would also result in access denial. This access denial default requires that users justify their needs before access is granted.
2. The design should not be secret. This makes the protection mechanism available for study and review without compromising sensitive parameters such as passwords.
3. It must be easy to use. If the users do not accept the mechanisms, they will search for ways around them rather than routinely, automatically applying them in daily work.
4. All access to objects must be checked. After any change in authority, future requests must be rechecked.
5. Only the necessary privileges should be allowed (least privilege). The user should be denied access from all elements of the system for which he cannot justify a need for access.

Designers must also resist the temptation to make the system too powerful. Limiting the flexibility of the system results in a more simple and understandable system and eases the task of verifying security. (101) Two broad classes of system failures should be

expected and planned for in advance. Algorithmic failures are those which result from design or implementation flaws. Probabilistic failures are due to circumstances in a random process, such as a hardware component failure, and can only be exploited or corrected after they are recognized. (93)

Penetration testing is intended to uncover integrity flaws which could permit unauthorized access to a protected resource such as a master password list or another user's data. Other system penetrations might be the ability to execute user code in supervisor state or achieving unauthorized access to physical equipment. Of course, failure in penetration testing does not mean that there are no flaws in the computer system. (114)

The traditional idea of protection in a computer system is that of an impenetrable wall, which appears as a barrier, or fence, to the access of some object. This fence is considered to be of infinite height and width so that there is no way to go around, over, or under it; the fence must be penetrated to achieve access. Presumably, as access to more sensitive objects is requested, there are correspondingly more fences to penetrate. In addition to limiting the path to the object, the fence may even function to keep the very existence of the object a secret, thus providing some additional protection, but only from the casual observer. Gates are placed in these fences to allow authorized communications with the objects under the supervision of a control mechanism, sometimes called a protection monitor. Optionally, this monitor may provide any necessary synchronization or log accesses (and access attempts) for later

analysis. Integrity exposures may be viewed as either illegal possession of the key to a gate or as a hole in the fence. (14, 110)

There are several hardware features that are designed to isolate subjects from one another. One of the most common features is memory protection, which may be achieved by several methods. (49) Memory bounds registers delimit a contiguous area of memory that a subject is permitted to reference, but this technique has several limitations. If multiple sets of bounds registers are not provided, performance may suffer. Without associated attribute registers, more elaborate types of control such as read-only or execute-only access cannot be implemented. A lock and key arrangement is superior to bounds registers because memory areas need not be contiguous. This mechanism allows the development of a hierarchy among keys so that the operating system may have access to all memory. The primary limitation is the number of locks as compared to the size of the memory protected and the number of simultaneous users that are permitted. In virtual systems, segment and page tables provide control over memory references. Addresses in user programs are translated by the CPU according to the contents of these address translation tables. Memory which has not been assigned to the user is not mapped by his set of tables and, therefore, cannot be referenced. This scheme also allows users to share the same areas of memory but to have different protection attributes.

Another hardware feature is the so-called system state bit which distinguishes between supervisor mode and user mode. Typically, users may execute most of the machine instructions in user mode, but there

will be several privileged operations which are reserved for supervisor use only. Input/output instructions and those instructions which directly alter the system state bit or other hardware protection mechanisms are in this class. The operating system thus has control over the execution of privileged instructions. When the user requests a supervisor function that requires the execution of privileged operations, the call to the supervisor is trapped and the system state bit is changed on behalf of the user, but only for the duration of the supervisor function. Normal user mode will be re-established when the supervisor function returns control to the user program. This mechanism is designed and implemented (usually in a combination of hardware and software) so that the user is never allowed to gain control in the supervisor state.

Parity checking has been a well-established technique since the earliest computers for detecting hardware errors as data is processed by the CPU, memory, and input/output devices. (19) The simplest form of parity check is the vertical redundancy check (VRC) which requires that each character have an odd number of one bits. Longitudinal redundancy checking (LRC) performs the same function, but in the other direction, by inserting an extra character every nth character. Finally, cyclical redundancy checking (CRC) adds a large number of bits but provides the best checking because single bit errors may be corrected and double bit errors detected. Other hardware features that can contribute to enhanced security control are a program-readable clock for time-stamping transactions, the ability for a program to determine the exact hardware environment in which it is

operating via obtaining hardware serial numbers, and the assignment of all operation codes so that no spurious instructions cause unpredictable results.

Recent advances in microprocessors allow these devices to be used in constructing hardware where duplication was previously prohibitive due to cost and size. One example is the possibility of using a content-addressable microprocessor for each track of a data base on disk storage. (49) The microprocessors search the data base in parallel and output only that data which does not violate the security specification. The cost of providing a microprocessor on each track of a disk cylinder is very small and is unlikely to degrade performance. For more elaborate protection, or for higher volumes of data, it is necessary to step up to minicomputers. They can be used to provide switching capabilities for classes of input/output devices (tapes and disks). This ability is useful for implementing the periods processing required by the Department of Defense; in a multiprogramming environment, only one classification of work (confidential, secret, top secret) is allowed to execute concurrently. For example, access to top secret disk drives must be prohibited if secret work is executing on the CPU. (47) A minicomputer can also be used as a monitor for a larger computer and sound an alarm if a security violation is detected.

A protection domain establishes a limited set of access rights that a subject has to objects, implementing the principle of least privilege. (70) Normally, a small program would only require access to a small number of objects, but a large program might require access

to correspondingly more objects. Consequently, this larger program would run in many protection domains and require a considerable number of domain switches. A capability-based system could be used to implement this protection model. A capability is a system-wide name, created by a low-level portion of the system, to correspond to an object and is not modifiable by the user. An example would be a thirty-two bit binary number which allows for over two billion names. Possession of a capability establishes a subject's access rights to an object (read, write, update, destroy). If a subject does not have a capability for an object, then the subject is denied access to that object. (41, 44, 68, 110) The idea of a conditional capability, implying that a matching condition or state must exist in addition to possession of a capability for an object, has also been advanced. (81) At run-time, the capabilities possessed by a given subject might be organized into a list or tree structure to provide better access for enforcement of the protection domain by the hardware portion of the protection mechanism. Given any reasonably sized system, there are definitely access and management problems due to the sheer volume of these data structures. Since the user must be denied access to the capabilities, it might be possible to use a microprogrammed associative device as a cache memory for the capabilities and then use addresses as a key to locate a capability register to check bounds and access rights. Such a system would also include instructions to access or alter data in this device, but only under proper authority. This implementation overcomes one of the greatest difficulties in any capability-based system: speed. The

micro-processor would only see an active table which would be backed up by a larger table (probably on a disk device) and most likely use a reference scheme not unlike paged memory systems to determine which capabilities to retain in the cache memory. Devices of this type do exist today in the implementation of fine-grained memory protection and virtual address translation offered on most current large systems. (44)

Another popular idea concerning protection is that of a security kernel which is a hardware mechanism, augmented by software, that enforces access controls within a computer system. The function of this isolated, formally verified security kernel is to provide the nucleus for a correct implementation of the primitives which are an essential foundation for secure operation of the operating system. (49) In the abstract kernel design, each object is given a fixed security classification when it is created. If read access is requested, then the security classification of the subject must be at least as high as the object (simple security condition). If the subject is untrusted and granted read access to one object, it will not be granted write access to a second object unless the security classification of the second is at least as great as the first (*-property). No access is permitted to inactive (deleted) objects. (79) The kernel approach leads to a modular system with the policies and mechanisms separate, thus permitting the proof, or certification, of the access control mechanism. (81)

A well-organized modular system written using structured programming techniques and top-down development achieves many interrelated benefits. Of prime concern is the ability to isolate and exhaustively test the most sensitive security routines in the system. In accomplishing this goal, the accuracy and reliability of these routines will be dramatically increased, as will their intelligibility. These attributes then feed back and contribute to higher levels of privacy and security. (84)

There are a number of well-defined functions performed by operating system software that may be classified as protective measures or security features. Identification of the user is almost always required before access to any resource is granted. Until proven otherwise, this identification should be treated as an assertion of identity and not fact. The authentication of identity is the proof and may be costly in terms of time (hardware resources consumed) or user frustration. Generally, passwords are used to verify the claim that the user has correctly identified himself. This process is critical and should be subject to strict regulations. The user should not be allowed to choose his own password; those can usually be easily guessed. The password should never exist in clear text once it has been acquired for verification. A critical point not to be overlooked is that authentication is not authorization. Authorization is an access control mechanism, typically a table or tree structure that specifies the privileges of a particular user but may also be delimited as a function of the services that he requests. For example, utility programs for file handling might be restricted

from the general user population. Requests may be screened as a function of data to be accessed or time of day. These types of restrictions will vary among installations and applications but in most cases some type of log, or audit trail, will be kept. The logging function may also be parameterized as a function of user, time of day, type of access, or even success or failure of the request. Given the availability of a comprehensive logging function, it is not too difficult to justify transcription of everything that occurs in support of system tuning, capacity planning, or backup and recovery of critical data files. A surveillance program can then examine the data searching for access to specific data items, a series of unusual requests, or a series of access failures that could suggest a possible penetration attempt. A more sophisticiated use of this data would be to develop threat monitoring routines to detect patterns of access that could signal attempts to collect data illegally. (47, 89)

The access control mechanism must be general, simple, and easy to understand, but flexible enough to support extension to user-defined subsystems. This mechanism must fail in a safe manner (access denial) and embody elaborate abilities--for example, to differentiate between the capability of a user to add records to a data base under control of a data base management system as opposed to allowing the actual physical input/output operations on the same data base.

There are several generic faults that exist in operating system software which could be used to subvert any control mechanism. The incompleteness or incorrectness of a design is a gaping hole which is next to impossible to remedy. For the millions of lines of code that

do exist, common flaws have surfaced again and again. Parameters supplied by the user on a supervisor request may not be adequately checked (or cross-checked). Properly verified parameters may be accessible to the user after they have been checked but before they are used, thus permitting a clandestine modification. Work areas acquired in the user's area of memory may contain sensitive residue after execution of a system function; in fact, bad input data may be used to force an error. Error recovery routines are notoriously incorrect and unreliable. System utility programs which may require an intentional opening in the operating system may not be subject to the same tight control as an application program. Checkpoint routines must record vast quantities of sensitive system data on an external storage medium in order to permit restart. (18, 47, 89) And, of course, there is always the ever-present system programmer with still another local modification to make the system function a little bit better or run a little bit faster. The systems programming staff is on very intimate terms with the software and knows where there are weaknesses in the software protection mechanism or easily exploited security exposures. They do not need to resort to creating trapdoors or inserting Trojan horses into the system during regular maintenance. Often they will disable any protection mechanism while applying maintenance, claiming greater efficiency as justification.

Verification of operating system software is now an art. There is adequate motivation for developing tools to advance these test and evaluation methods to a science. Analytic tools are available to examine source code to identify modules for further study. The

determination of the characteristics or profile of a module and the generation of supplemental documentation, such as extensive cross-reference lists, form the basis for more complex testing and modelling approaches. If the flow of control can be determined, then exercisers can be developed to overload critical subsections of code or to perform exhaustive tests. It may also be possible to automate the search for generic flaws such as those enumerated above. Asynchronous input/output and time dependencies are more difficult to handle, but the benefits to be gained will be commensurate with the expenditure required to solve these types of problems.

CHAPTER IV

SECURITY IN EXISTING COMPUTER SYSTEMS

The hardware and software protection mechanisms of several existing computer systems are described in this chapter. The first two systems to be discussed are primarily research systems but have also become viable computing resources in their respective user communities as they have evolved and stabilized into working systems. Hydra is the operating system for C.mmp, a multi-mini-processor system at Carnegie-Mellon University. A protection mechanism, implemented by capabilities, was a basic design specification of this system. Multics, a follow-on to Project MAC at the Massachusetts Institute of Technology, is another research system which provided a hierarchical protection mechanism known as "rings of protection." The next two systems presented are included to show the vast differences between file security on a large number-crunching system and a powerful time-sharing system. The Control Data Corporation's Network Operating System is used to control CDC's 6000 series and has a minimal file security system. The DECsystem-10 file protection mechanism supports eight levels of access for three categories of users: the file owner, another member of the owner's project, and all other users. Finally, three of IBM's operating systems will be reviewed. Virtual Machine Facility/370 provides an effective virtual machine hypervisor which can be used recursively; that is, VM/370 can be run as a guest operating system in a virtual machine under control of the real

VM/370. It is this capability that is under study by the Department of Defense in an attempt to build a secure VM/370 system called KVM/370. Operating System/360 Multiprogramming with a Variable number of Tasks at one time was the most advanced general purpose operating system known, but it contained numerous integrity exposures. OS/360 MVT was the compatibility target for Operating System/Virtual Storage 2 Multiple Virtual Storages, which provides an operating system with guaranteed integrity upon which secure subsystems and applications can be built.

Hydra

The Hydra operating system was designed to support up to sixteen of Digital Equipment Corporation's PDP-11's as a multi-mini-processor system. The multiprocessor, C.mmp, and its operating system, Hydra, have grown from a large research project for operating system implementers to include all facets of operating system and multiprocessor research, as well as attracting many curious users. Protection was successfully integrated into the lowest level of the operating system to provide a flexible set of mechanisms to implement an arbitrary set of facilities and policies. (17, 69, 118, 119, 120) Access to objects is controlled by the use of capabilities, which are manipulated only by the Hydra kernel. Generic operations on capabilities (get, load, store, copy, delete) are implemented by instructions which trap to the kernel. A procedure is an abstraction of the idea of a subroutine and establishes a template of prototype capabilities to be replaced by actual parameters. A local name space,

or LNS, provides a naming function and defines the instantaneous protection domain. Associated with each LNS is a capability list which contains capabilities for all objects which may be accessed. All objects can contain capabilities for other objects; the Hydra kernel follows the path to the target object. A subsystem may require different access to an object than the current LNS; rights amplification provides a method to expand (or contract) the rights allowed to a subsystem when necessary.

These mechanisms can be used to solve a number of protection problems, either by direct solution or by providing a protected environment for code which implements a solution. (17) The call mechanism, whereby the LNS is pushed onto the stack, solves the mutual suspicion problem (both the calling and called programs are concerned about how much damage the other can do) directly because the called LNS does not inherit access to the caller's LNS. A called procedure can also be prevented from modifying an object passed to it if the caller does not pass an appropriate capability and the amplification template does not also allow the modification. Called procedures can also be prevented from passing a capability to a third user. A similar approach guarantees that access is revoked when a procedure returns to the caller by destroying the LNS for the called procedure along with the LNS's for any subsequent calls made. The confinement problem (prevention of a called program from passing data to a third, unauthorized program) is addressed by creating a confined LNS (which can only create other confined LNS's) which can not pass data directly to another procedure. Covert channels (disguised signals to another

program, such as amount of CPU time consumed or number of input/output operations performed) are not closed, however, so there is a possibility of some data being transmitted to another program, but only at a low bandwidth. Selective temporal revocation of rights is accomplished by creating an alias, which resides between a capability and the object it represents. The called procedure is then passed a capability for the alias.

Multics

The Multics operating system began as an experiment and is now available commercially. The primary objective of Multics (as a private computer utility) is to provide protection facilities for the controlled sharing of information so that the user may place appropriate controls on the release of private information with confidence. (20, 98) A tree-structured hierarchical administrative system is headed by a system administrator, who allocates resource quotas to projects. Project administrators control the way users use the system by allocating resources to the users. This decentralization eliminates an administrative bottleneck by establishing lines of authority and requiring that the user take some responsibility for his work. (59, 98) A process in Multics is known by the user's name (verified at login) and is the combination of a program and an address space (the segments available to the user). All user authentication is done interactively; batch jobs must be released from an authenticated interactive user.

All information in Multics is mapped into the virtual storage of some process, and the protection responsibility is a function of the access control mechanism of the storage system. The descriptor-based, primary memory protection system is implemented in hardware so that there is no performance loss in a call to either a protected or unprotected subsystem. This structure is called "rings of protection" and establishes the typical supervisor-user protection relationship; it is also extendable to user-written subsystems. This mechanism is a refined version of the general access controls to a segment which provides read, write, or execute access. The user is assigned to a ring for normal use (usually ring 4) by the project administrator and is then entitled to use any segments residing in that ring or a higher-numbered ring (lower privilege). Transition to a lower-numbered ring (higher privilege) is controlled by providing gates, or entry points, to procedures so that data of parameters passed may be thoroughly checked. Access lists, enumerating those users who may reference a segment, are organized from specific to general and may be specified by user name, groups (projects), or compartments (subareas). The access mode (read, write, execute, none) is associated with the use of an object, not the object itself.

There are two major weaknesses in the Multics design. (98) The first is the amount of system code (15%) in the most protected area (ring 0). This was not intended in the design but evolved during the early development stages as a result of eagerness to have a functioning system and a lack of knowledge of better techniques. The ring structure can provide a solution to this problem by placing

supervisory routines which require protection, but not all supervisor privileges, into an intermediate ring. This change would reduce the amount of code with privileges and be a step toward reducing the size of the supervisor (kernel) so that it could be formally verifed as correct. (20, 99) The second weakness is the complexity of the user interface. Defaults hide the complex underlying structure of access lists, access modes, backup copies, bulk input/output, and implied access through higher level directories. A user with an unusual protection requirement must decide for himself how to establish the desired environment. A project administrator has a similar situation but with more defaults and more possibilities.

CDC NOS

The Control Data Corporation's Network Operating System (NOS) controls the CYBER 170 Series, the CYBER 70 Series, and the 6000 Series Computer Systems. (83) References to main memory are controlled by memory bounds registers. The operating system is permitted access to all memory by setting its starting address to zero and the field length to the size of the memory. Tape files are protected by a security character in the ANSI header label. They may be read by the owner or anyone who knows the security character. Disk files are accessible either by the owner, the public, or by anyone who knows the filename and password, which may be supplied by an executing program in the macro call issued to gain access to the file. Accesses are recorded for review by the owner. File access modes allowed are read-write, read-only, execute-only, append, and combinations of the

above. The PERMIT macro is provided for the file owner to allow access to his files in a specific mode by another user.

DECsystem-10

The DECsystem-10 is a large-scale computer system which may contain multiple PDP-10 central processors. (25) Protection and relocation hardware map the user's virtual address space into two segments. A high (pure) segment contains reentrant programs which are usually write-protected and shared by many users. The low (impure) segment is unique to each user and is available to the user for any purpose. Most programs operate in user mode, which guarantees the integrity of the monitor and other user programs. In this mode, memory protection and mapping is automatic and all illegal instruction codes are trapped. User mode is subdivided into public mode and concealed mode. A program running in public mode cannot access the concealed area except to transfer control to it at predefined entry points (execute-only access). A program running in concealed mode may access all of a user's virtual memory. In supervisor mode, all instructions are valid and all memory is accessible, but some pages may be write-protected. The supervisor mode is used for general management of the system including all interrupt processing, performing input/output operations, and the construction and maintenance of page maps.

The primary control program for the DECsystem-10 is the TOPS-10 monitor, which identifies users by a combination of a project number and a programmer number. Typically, a file owner is considered to be

any user with the same programmer number as the file directory, regardless of whether the project numbers are the same. An installation option may be specified which requires a match on both the project number and the programmer number. The TOPS-10 monitor implements a complete file access protection mechanism with eight levels of access defined: none, execute-only, read, append, update, write, rename, and change protection. A protection field establishes the access privileges for three classes of users: the owner of the file, other members of the owner's project, and all other users. Programmer numbers zero to seven are special and will be considered the owner of all files with that programmer number, independent of the project number. All privileges are assigned to the user running under the project number-programmer number combination [1,2].

A licensed software product, the File Daemon (FILDAE), is available to provide extended file protection. This program is invoked if the owner's protection field is at least append access, even if it is the owner accessing the file. A user (or the system administrator) may create a file containing the names of a user's files and specifying who may access each file and the highest type of access that that user is allowed. This access request may optionally be logged in the owner's disk area. The software supplied is intended to be an example but is typically used unmodified by installations desiring this additional protection mechanism.

VM/370

Virtual Machine Facility/370 (VM/370) is a general purpose time-sharing system which provides multiple software replicas of a real System/370 environment so that each guest operating system appears to have dedicated real resources (CPU, memory, and input/output devices). (100) The main component of VM/370 is the control program (CP), the resource manager which resolves differences between the virtual machine and the real system. The other component of VM/370 is the Conversational Monitor System (CMS), which provides an environment for interactive program development and personal computing. The services provided by CP are transparent to the virtual machine because CP is operating in its own address space, isolated from all other users. This technique minimizes the impact of software failures and also provides integrity to the virtual machine. Several improvements in hardware (microcode assists) and software (VM handshaking) have increased VM/370 performance significantly, but the isolation and integrity of the individual virtual machines have been preserved. (72)

IBM has provided two products which allow virtual machines to communicate directly with each other rather than using previous methods such as spool file transfer, virtual channel-to-channel adapter, or 370X transmission control units. The VM Communication Facility (VMCF) is a software interface which provides an asynchronous logical connection between two or more otherwise isolated virtual machines. Authorized users communicate with each other via a privileged instruction (diagnose) interface to CP. A user field

(doubleword) is provided in the parameter list and could be managed as a password by the user, thus implementing some security. There are potential integrity problems due to the fact that this asynchronous process can result in long delays due to page faults or the user can violate the rules of the interface. (60) Virtual Control Storage (VCS) provides a synchronous interface between virtual machines via a new instruction (X'FFxx') which is interpreted by CP. This implementation is intended to provide the capability to implement a central server type of application such as a data base management system. The protection and scheduling functions are separated to improve performance and take advantage of all existing protection mechanisms in VM/370. (6)

Penetration studies of VM/370 have succeeded in identifying weaknesses in the complex input/output interface simulation which could allow a user to take over the system or deny resources to other users. User channel programs are copied into the CP address space for execution, but the protect key is the only restriction to the real channel. Non-terminating channel programs may also be written to deny channel usage to other users. VM/370's strengths include its inherent simplicity, different file management systems for VM/370 and the users, and the fact that system control information is not stored in the user address space. (7)

Efforts have been sponsored by the Department of Defense to identify and build a security kernel for VM/370 which will implement the military security policies of the simple security condition and the confinement condition, historically known as the *-property (star

property). (38, 39) This system, KVM/370, will achieve further security by combining all security-relevant modules (privileged operation instruction users and global system data users) into a kernel which will run in real supervisor state. A non-kernel CP (NKCP) will run in real problem state for each security level in the system. Trusted modules (logon, directory update, operator processes, unit record and spool allocators) will receive the same automated formal verification as the kernel. Semi-trusted resource management modules will be audited for Trojan horses. It is expected that these modules could only communicate through covert channels at a twenty or twenty-five baud bandwidth.

OS/360 MVT

Operating System/360 Multiprogramming with a Variable number of Tasks (OS/MVT or MVT) has been a very popular operating system for installations with medium to large System/360 CPU's. One of the reasons for this popularity is that the system programmer can customize the system to fit the needs of a particular installation by a tailoring process known as system generation. Further selection of options, within a limited range, is possible each time the system is loaded from disk. The primary design objectives of OS/360 in general, and MVT specifically, were:

1. balancing the hardware, software, and human resources to provide the performance and facilities required by the installation, and
2. modularity to provide the ability to grow without

disruption in areas of performance, applications, technology, compatibility, and device independence. (53)

The MVT supervisor, which executes in the supervisor state (all instructions are valid), controls allocation of memory, starts input/ouput operations, loads programs for execution, and initiates execution of programs. All other programs run in the problem state, which does not allow the execution of the privileged instructions to initiate input/output or change system state. The master scheduler controls overall computer system operation, and the job scheduler enters work into the system and schedules this work under control of the supervisor. While these two routines do not operate in supervisor state, they do run with a storage protection key of zero, so all memory is accessible for reading and writing. A number of language translators and utility programs are also supplied with the system.

One area of the design of OS/360 which was emphasized was the capability of the system to share resources: data, programs, and areas of memory. Unfortunately, adequate controls were not designed into the protection mechanisms, and the inherent flexibility of the system was shown to contain a large number of integrity exposures. The user could achieve supervisor state, key zero, or access another user's data without permission. On the System/360 Model 50 and above, hardware main storage fetch protection is provided, but OS/360 does not use it; so memory is not shielded from unauthorized reading but is protected from unauthorized writing by the storage protection key. OS/360 provides minimal data set security with a password mechanism

which requires that the system operator supply the correct password during the execution of OPEN routines before access to data is granted. (52) Most installations using this facility have modified it so that the password is supplied via another method and the operator does not have to know it. The only indication of a password-protected data set is that a bit is set in the data set control block in the volume table of contents on the disk volume (or in the volume label on a tape volume) containing the data set. Given the existence of integrity exposures in the basic system, this password security is minimal. An access control mechanism was developed at the IBM San Jose Research Center in 1973. (37) This Inventory Control System, or Installation Management Facility, provided controlled access to data sets and decentralized the accompanying administrative responsibility. The system was not impenetrable when attacked by a determined opponent, due to the lack of a secure base.

In May 1972, IBM announced a major study of data security requirements. The Resource Security System (RSS), an experimental version of OS/360, was updated to support release 20.7 of OS/360 and was installed at four locations for evaluation of a secure system in a data processing operation. (22, 23) RSS addressed the following functional areas:

1. A security officer is provided commands for defining controlled users and resources, grouping users and resources for authorization, limiting a user's access according to his security profile, and requesting reports on the authorization structure of the system.

2. A code word is used to identify the user each time he submits work to the system. Authorization schemes are established for terminal usage, volume integrity, data set space allocation, data set security, program security, removal of data sets, and printed output labeling.
3. Extensive statistics and an audit trail of attempted violations are maintained.

Modifications to OS/360 by RSS were made to many portions of the system at the cost of performance degradation ranging from 1% to 31%. (85) This is not normally an issue in a security environment because typically the security requirements are mandated by a government agency or top level management; therefore, any cost in throughput or response time must be absorbed as additional overhead. RSS did close known integrity exposures in OS/MVT, and it is now obvious that an unstated purpose of this study was to gain valuable experience with this type of operating system before announcing OS/VS2 MVS and a corresponding security product, the Resource Access Control Facility.

OS/VS2 MVS

The design of Operating System/Virtual Storage 2 Multiple Virtual Storages (OS/VS2 MVS or MVS) recognized the need for system integrity in support of security and that the concept of an adversary had not historically been considered in general purpose operating systems. Specifically, OS/MVT was not designed to prevent deliberate user

interference with the operating system, but the MVT philosophy did attempt to protect itself from accidental user errors. (5, 76) The system integrity support in MVS states that there must be no way for an unauthorized program to:

1. bypass store or fetch protection
2. bypass password checking
3. obtain control in an authorized state (supervisor state or a system protect key).

Furthermore, IBM will accept and respond to an Authorized Program Analysis Report (APAR) which describes any potential violation of any of the above conditions. The elimination of the integrity exposures provides a foundation on which additional security capabilities such as the Resource Access Control Facility (RACF) and the Cryptographic Subsystem may be built. (76, 105) Several responsibilities must be assumed by the installation to insure continuance of this secure environment: appropriate system libraries must be password-protected, and local modifications must be carefully scrutinized to insure that integrity exposures are not introduced. (76)

Several general classes of integrity problems have been identified, and techniques have been developed to close the exposures they create. Some of these are obvious and only require additional code to eliminate: system data in the user area, non-unique identification of system resources, and the user supplied address of protected control blocks. Others may require significantly more effort to correct: system violation of storage protection, the concurrent use of serial resources, and uncontrolled sensitive system

resources. The RACF program product (a descendant of the Inventory Control System) builds a further access control mechanism on the secure base of MVS and is designed to enhance an installation's control over who is allowed to use what data and how it may be used. (37, 88) Profiles stored on direct access storage devices are used to perform user identification and verification, authorization checking, and logging. RACF is sufficiently generalized so that any resource can be protected, such as account numbers used to charge for use of the computing resource or a job class. This product has extended the security capabilities of MVS and its subsystems far beyond the limited password facility which the system initially provided.

The Cryptographic Subsystem provides communications and file security to protect data in the hostile environment of a communications line or portable media such as magnetic tape or disk. This subsystem consists of a number of hardware and software components which implement the National Bureau of Standards Data Encryption Standard and will be examined in detail in later chapters.

CHAPTER V

ENCRYPTION

The hardware and basic operating system software discussed in the preceding chapter have provided several access control mechanisms that are intended to protect data by denying access to unauthorized persons. It is possible that a computer operator or system administrator may be able to bypass these controls to gain access to sensitive data. Flow controls are often an extension to the access control mechanism and restrict the passage of information from one security class to a lesser class (confinement condition implementation). Inference controls may attempt to limit the extraction of data from a statistical data base or increase the cost to compromise a specific system. These methods remain largely insecure because a single error may allow sensitive data to be disclosed. Examples of such errors are a password list left on a desk or confidential data copied to offline media for backup or maintenance. Some method of encryption may significantly enhance the security of data where other controls are faulty, by concealing information. (26)

A cipher is a transformation process which renders a plaintext message (said to be "in the clear") unintelligible to someone who does not know how to reverse the transformation. A code is different from a cipher because a code is typically a dictionary-like system in which syllables, words, or entire phrases in the plaintext are replaced by

code words or numbers. A cipher normally operates on single letters (or a block cipher on fixed-length groups of letters) and is driven by a key of some type which specifies the pattern of the transformation. The optimum cipher, known as the one-time pad, uses a key which is longer than the message only for the encryption of that single message. Cryptography is the part of cryptology that is concerned with scrambling the plaintext, and cryptanalysis is the attempt of someone to break a cipher without prior knowledge of the transformation applied or the key used. Reference material on cryptanalysis is limited; Gaines (36) and Sinkov (104) are the two volumes most often cited and are relatively old but good works.

There are two major transformations (and others more exotic) that can be applied to a plaintext message. Simple transposition shuffles the letters of the plaintext; for example, the word COMPUTER might become URMCTEOP. These systems are not too interesting and may be solved by anagramming until syllables or words appear. Substitution is more complicated and offers a wider range of possibilities to confuse an adversary. In a substituion scheme, COMPUTER might become XQAZBMSD. These two transformations may be combined to form a product cipher, which is stronger than the sum of its parts.

The science of cryptology has a rich history which has been documented by Kahn. (62) The essential element of cryptography, deliberate transformation, was first demonstrated in the main chamber of the tomb of Khnumhotep II where unusual hieroglyphics were used in place of the standard ones. The early Egyptians, Chinese, Indians, and Mesopotamians all employed secret writing as a method for

transmitting information and delaying its comprehension, even if only for a short period of time. The first cipher, or substitution system, which is well-known today, is the Caesar cipher, in which the plaintext letters are replaced by letters three places farther down the alphabet. This name is now used to refer to this entire class of ciphers, regardless of whether the letter substituted for "A" is "D". This is a form of monoalphabetic substitution because there is only one cipher alphabet in use. A polyalphabetic substitution uses two or more alphabets and is the type most often implemented in modern cipher machines used by the military. The Vigenère tableau is the most famous polyalphabetic cipher system; it is composed of twenty-six Caesar ciphers, one for each letter of the alphabet.

In World War I large volumes of enciphered messages implied that valuable information was concealed within, and the availability of multiple messages enhanced the probability of finding solutions. (62) The German field cipher, the ADFGVX system, was a block product cipher which incorporated daily changes in two key variables. A six-by-six checkerboard contained the characters to be enciphered; each was replaced by the row and column labels of ADFGVX and then a transposition was applied.

Interest in product ciphers declined and by World War II machines which used multiple rotors to generate lengthy keys (approaching the one-time pad) were in military use. The Japanese employed a system of this type, named PURPLE, to send diplomatic messages. The PURPLE encipherment was often a superencipherment; that is, the message was enciphered twice in two different systems. American cryptanalysts

broke this cipher and built a machine to decrypt messages in August 1940 after approximately eighteen to twenty months of effort. Unfortunately, a number of events were overlooked which might have prepared the United States for the Pearl Harbor attack: the reassignment of twenty thousand radio call signs on November 1, 1941, and December 1,1941, an occurrence usually separated by six months; the transmission to the Japanese embassy in Washington, D. C., of instructions for destroying their machines and codes in early December 1941; and the message describing a break in negotiations between Japan and the United States.

Much of the work done in cryptography during World War II was, and still is, classified. A paper by Shannon, declassified and published in 1949, examined the mathematical structure of secrecy systems, the theoretical limits of secrecy, and practical secrecy in terms of work required to solve a cipher. (102) He observed that a good cipher is one represented by a problem known to be difficult to solve, such as a large system of simultaneous non-linear equations. Two methods were identified that would impair or discourage any statistical attacks on cryptograms, both of which should be used in a good cipher. Diffusion is intended to spread any redundancy out over large strings of text, thus requiring greater quantities of enciphered text for analysis. Confusion requires that the statistical relationships between the key variables and the enciphered text be involved and complex. Furthermore, he suggested that a good mixing transformation is desirable. Good mixing transformations may be developed by combining simple systems--for example, a preliminary

transposition followed by a sequence of alternating substitutions and linear operations. These are strong ciphers when viewed from the standpoint of their work characteristic; they result in a complex system of simultaneous equations which is difficult to solve. The only drawback of such systems is that an error in transmission will affect several characters when deciphering. These principles were demonstrated in an early IBM system called Lucifer. (34) The observation was made that caution should be exercised in selecting the substitution functions so they do not degrade into simple permutations. Additionally, the idea of selectable substitution functions was advanced, where the actual substitution function used depends on the text or key being processed.

The capabilities of modern digital computers enable almost anyone to use some form of cryptography to safeguard stored data. Normally, unless the enciphered data can be deciphered, there is nothing to be gained. The system password file is a good counter-example of this belief. If the password is enciphered by a non-linear function which is difficult or impossible to invert, then compromise of the password file is not an issue. This method requires that the user's password be enciphered each time the user signs on the system before the comparison is made to the stored version. The speed and flexibility of the hardware allow the choice of a combination of methods to implement a good mixing function for this purpose. Permutations, substitutions, table lookups, or complicated mathematics such as sparse polynomials can be used without imposing undue hardship on users or system administrators. (32, 94)

The availability of inexpensive digital hardware to implement secure communications over an unsecure channel by using cryptography has spawned a great deal of interest. It has been suggested that the use of electronic mail and the lack of a written signature pose significant problems that may be solved as the ancient art of cryptography evolves into a science. (29) Public-key cryptosystems are one possible solution to this problem. (28, 29, 78, 95) In this method, different keys are used for encryption and decryption. Each pair is a related set and typically they are two large prime numbers, on the order of two hundred decimal digits. A user's encryption key (E) is published in a directory, but the decryption key (D) is kept secret. Encryption is performed by representing the message as a number between 0 and N-1 and then raising it to the Eth power modulo-N. Decryption is a similar process: raise the ciphertext to the Dth power modulo-N. This system is also considered to be a set of trap-door one-way functions, since it is computationally infeasible to determine the inverse functions. (29) The alternative solution is the older conventional key encryption, which may function just as well as the public-key cryptosystems. It is the strength of the algorithm that is important for the class of problems usually considered: private communications, internal authenticators, datagrams, and digital signatures. (63) The strength of the public-key cryptosystem described suggests that it may be used to safely transmit keys which would be used in other systems. After this preliminary exchange, a method may be used for bulk data transfer which does not impose such a high computational overhead.

The growing need for a uniform method of encryption for use in the data processing industry led the National Bureau of Standards to seek a crytographically strong algorithm. On January 15, 1977, an algorithm submitted by the IBM Corporation was selected as the Data Encryption Standard (DES). (21) This algorithm is a recirculating block product cipher, which operates on blocks of sixty-four bits under control of a fifty-six bit key variable. (13, 21) After an initial permutation, the input block of sixty-four bits is processed by a complex key-dependent function and then subjected to the inverse of the initial permutation. The complex function is sixteen iterations (or rounds) of operations on the left half and right half of the sixty-four bit permuted input. At a given iteration, the left output half is the right input half. The right output half is the modulo-2 sum (exclusive-or) of the left input half and a function of the key and the right input half. At each iteration forty-eight different key bits are selected for modulo-2 addition with the right half, which is expanded to forty-eight bits. This sum is divided into eight groups of six bits each to select eight groups of four output bits from eight substitution tables. The resultant thirty-two bits is then permuted prior to modulo-2 addition to the left input half. The two halves are not interchanged after the sixteenth iteration. One property of the DES is that each ciphertext bit is a function of all plaintext bits and all key bits. Meyer has shown that this "intersymbol dependence" is fully established after only five of the sixteen rounds which make up the heart of the DES. (77)

The standard has been subjected to strong criticism by Diffie and Hellman that it is too weak for some applications and that a twenty million dollar machine could be built to break the standard in an average of twelve hours of computation by exhaustive search. (27) Rapidly decreasing costs of hardware could lower the estimated five thousand dollar cost per solution to only fifty dollars by the late 1980's. The known-plaintext attack described is based on brute force, but there is no doubt that it would be successful. This approach was selected because most successful, professional cryptanalysis is based on variations of this method; it can be modified to a ciphertext-only attack; and finally, perhaps most importantly, the National Bureau of Standards has stated that the algorithm should be secure against the known-plaintext attack.

The technique used in the machine which Diffie and Hellman proposed is to search one million keys in parallel, which should yield a solution in about half a day, assuming that on the average only half the keys need to be searched. This type of machine could be easily designed since the DES was intended to be implemented on an LSI chip, but the actual construction would have to be economically justified. It is assumed that agencies such as the Central Intelligence Agency or the National Security Agency could easily produce such justification, and indeed, Tuchman initially stated that by 1981 IBM could deliver this machine at a cost of two hundred million dollars. (65) IBM officials later repudiated Tuchman's statement and agreed with the NBS estimate that current technology would not support such a machine and, furthermore, that it could not be built until 1990. (27, 65) The NBS

specifically responded with a number of objections to refute Diffie and Hellman's proposal of a design for such a machine. They are listed below, followed by Diffie and Hellman's counter-arguments. (27)

1. Objection: Design and control costs of such a parallel processor overshadow the hardware costs.

   Reply: This would be true of a real parallel processor which requires interaction between processors, such as Illiac IV. The architecture is more like a large memory due to low input/output and lack of component interaction.

2. Objection: The mean time between failures would be much less than one day; an error-free search would not be feasible.

   Reply: The architecture would specify sixty-four racks, each with a minicomputer controller to switch to a spare rack (three or four needed) to replace the failing component.

3. Objection: A ten dollar, one microsecond chip cannot be built; however, a one hundred dollar, forty microsecond chip is possible.

   Reply: This statement assumes MSI TTL implementation when in fact N-channel depletion load or CMOS/SOS technology would be used. One hundred dollars is reasonable for small lots, but if a million chips (plus spares) were ordered, ten dollars is not an unreasonable

assumption.

4. Objection: Six thousand six-foot racks would be required (physical size).

   Reply: Assumptions of search chip geometry and control microprocessors required for input/output are grossly overestimated. Only sixty-four racks would be needed.

5. Objection: A large amount of power would be consumed by a million-chip machine operating at high speed.

   Reply: Assuming N-channel depletion load devices were used and adding a factor of two for power supply losses, cooling and control logic, only two megawatts are required. At three cents per kilowatt-hour, this is about fifteen hundred dollars a day, or fifteen percent of the five-year amortization cost of the entire machine.

6. Objection: Doubling or quadrupling the key size would increase the cost of the machine.

   Reply: Only ten percent of the encryption chip area requirements are related to key size. Additionally, the device would handle a variable length key since shorter keys would suffice in many applications.

7. Objection: Frequently changing keys will thwart exhaustive search cryptanalysis.

   Reply: A change often enough (hourly) would reduce the determination of timely solutions only to about one per day. In a financial application the ten thousand dollar

cost per solution would still be useful.

8. Objection: There is no guarantee that a solution found is unique.

   Reply: Even if multiple solutions were found, a second ciphertext-plaintext pair would probably determine a unique solution.

Diffie and Hellman have also asserted that this machine is capable of being modified to mount a ciphertext-only attack. This statement is based on the assumption that since both the DES and ASCII are Federal Information Processing Standards, the plaintext will be in ASCII. In this case, every eighth bit will be a parity bit, the exclusive-or of the preceding seven bits. If an attempt to decode the ciphertext block yields bits with correct parity, a second block is attempted. Additional blocks are tried as the solution is passed up through board and rack levels. Only if correct parity persists is a solution likely. This assumption of the use of ASCII may be valid for certain communications of federal agencies, but falls short of being a serious matter because the DES has not been accepted for use in enciphering classified data, nor are commercial firms bound to the use of ASCII.

A major concern supported by NBS is that the standard may have to be revised relatively soon (in the 1982 to 1987 range) due to changes in technology. (121) In fact, it may be that long before the DES gains widespread use. Another significant danger in changing technology is that ciphertext that cannot be broken can simply be retained until it is feasible to decipher it. Additionally, one of

the cardinal rules of cryptosystems has been broken in the design. The government has asked IBM not to divulge the internal structures of the substitution tables used in the DES and has classified some of the design principles. This implies that the possibility exists that one of the designers may turn against the standard and be able to break it or that a trapdoor may exist. In addition to IBM, the Central Intelligence Agency and the National Security Agency are noticeably silent on evaluations of the DES. It has been alleged (27, 65) and denied (111) that the NSA had a strong influence on IBM in designing the algorithm that is now a Federal Information Processing Standard. The United States Senate Select Committee on Intelligence has concluded that the NSA convinced IBM that a reduced key size was sufficient. (46) The IBM prototype Lucifer which preceded the DES used a one hundred twenty-eight bit key. The committee also determined that the NSA assisted in developing the substitution functions and certified the DES free of statistical and mathematical weaknesses. Requests to the NBS that evaluation studies done by IBM and NSA be released have been ignored. (27)

A final issue is the generation and storage of keys. Since a lost or incorrect key renders the enciphered data useless, this is a significant problem and will be addressed in Chapter Six because it is strongly related to software support of the DES.

The next two chapters are examinations of the hardware and software implementations of the DES. Since IBM played such a major role in development of the standard, the focus will be primarily on the products which IBM has brought to the marketplace. Deficiencies

in these products will be pointed out and recommendations given for improvements to them.

CHAPTER VI

SOFTWARE ENCRYPTION

There are several problems associated with use of the DES for protection of data, whether it be for communications security or file security. These problems will be referred to collectively as the key management problem and include the selection (generation), distribution (handling), protection, and installation of the cryptographic keys used to encipher or decipher data. If encrypted data is to remain secure, then the key used to encrypt the data must also be secure. There must be available an automated mechanism to provide a secure supply of keys for encrypting new data. These requirements dictate that all keys themselves be encrypted but still remain readily available for decrypting data by authorized users. Due to speed and availability constraints, these keys are best stored on the computer system itself. Such a key management scheme has been described (31, 74) and provided in an IBM Program Product, the Programmed Cryptographic Facility. (86) This product allows the enciphering of data before it is written and the deciphering of data after it is read.

Another cryptographic system is available only within IBM and will not be examined closely. (66) The Information Protection System (IPS) uses the DES and ciphertext feedback to protect confidential data. There are two major differences between this system and the Programmed Cryptographic Facility. First, key management is totally a

user responsibility with IPS and more flexibility is allowed in choosing a key. The requirement that the user provide the key when it is necessary to decrypt data insures that the key is not available in the computer system (even in encrypted form) and subject to compromise. A long character string may be supplied and IPS converts it to a fifty-six bit quantity, as required by the DES. Second, a header record is ordinarily placed at the front of a file to indicate how the data should be deciphered (chaining method used) and to contain a verification field to determine if the user is attempting to decipher the file with an incorrect key. The latter feature is undoubtedly advantageous because an enciphered output file may not readily be recognized as incorrect; the capability to easily verify the ability to decipher a file is valuable. It is also interesting to note that files encrypted by IPS are fully compatible with those encrypted by either of the IBM products which are commercially available.

The Programmed Cryptograhic Facility has defined six types of keys which are grouped into three categories as shown in Table I. The first category is primary key-encrypting keys which are used to encipher other cryptographic keys. There are two types of keys in this category. The host master key (the only key that must be supplied by the user) is used to encipher operational keys (used to encipher and decipher data) and as a base for deriving the two host master key variants, by complementation of selected bits. The host master key is the only key that exists "in the clear" in the cryptographic facility. Host master key variant-1 is used to encipher

TABLE I

Categories, Types, and Uses of Cryptographic Keys

Category	Type	Use
Primary key-encrypting keys	Host master key	Encipher cryptographic keys
	Host master key variants	
Secondary key-encrypting keys	Local keys	Encipher cryptographic keys
	Remote keys	
	Cross keys	
Data encrypting keys	Operational keys	Encrypt and decrypt data

Source: OS/VS1 and OS/VS2 Programmed Cryptographic Facility General Information Manual. GC28-0942-2. White Plains: IBM Corporation, 1978, p. 10.

both master key variants, all local keys and cross key-1. Host master key variant-2 is used to encipher all remote keys and cross key-2. The second category of cryptographic keys is secondary key-encrypting keys which are used to encipher data-encrypting keys. There are three types of keys in this category. Local keys are used to encipher operational keys that are written to a file or sent to a terminal. Remote keys are used to encipher an operational key before it is written to a file. Cross key-1 is used to encipher an operational key that will be sent to another host processor. Cross key-2 is used to encipher an operational key that will be written to a file or to

decipher an operational key read from a file or received from another host processor. The third category of cryptographic keys is data-encrypting keys which are the operational keys used to encipher and decipher data. Security for all keys is provided by enciphering them before they are stored or allowed to pass outside the system either in a file or on a transmission link. Additionally, it is recommended that the value of the stored cryptographic keys be changed at least yearly.

A key generator utility program is provided to create and maintain the necessary cryptographic keys. Two data sets are required for key storage: a master key variant data set (MKDS), which contains the two variants of the host master key, and a cryptographic key data set (CKDS), which contains the host master key and all secondary key-encrypting keys. When supplied with a new host master key, the key generator utility program derives the new host master key variants and re-enciphers all keys in the CKDS. This utility program can also create or change the secondary key-encrypting keys.

The Cryptographic Key Resource Manager is used to create the operational keys that are used to encipher data and to manage the MKDS and CKDS. The interface to this resource manager is via two assembly language macro instructions: GENKEY and RETKEY. The GENKEY macro requests the generation of an operational key, which is returned to the user enciphered by the host master key and the remote key. The RETKEY macro requests the translation of the operational key which is returned to the user enciphered under the host master key.

The following procedure would be used by an application program to encipher a file as it was written to magnetic tape or disk.

1. Issue the GENKEY macro to request that a new operational key be generated.
2. Issue the CIPHER macro to encipher the data record, passing as parameters the data to be enciphered and the operational key enciphered under the host master key.
3. Write the operational key enciphered under the remote key and the enciphered data to the file.

A similar procedure would be used to decipher a file for processing by an application program.

1. Read the enciphered operational key and enciphered data from the file.
2. Issue the RETKEY macro to request translation of the operational key.
3. Issue the CIPHER macro to decipher the data to be processed, passing as parameters the data to be deciphered and the operational key enciphered under the host master key.

There are a number of significant positive observations that can be made about this method of encrypting files. (31, 74) The operational key is encrypted under a secondary key-encrypting key which is not dependent on the host master key. Therefore, the file does not have to be re-written when the host master key is changed, nor is the host master key derivable and thus vulnerable to compromise. Recovery of the file is possible at another location

because the secondary key-encrypting key does not assume the importance of a master key and thus may be divulged, if necessary, without fear of wide-ranging compromise. At the user's option, the enciphered operational key does not have to be written on the file or stored in the system. The key then becomes a personal key, but it must be provided to the system whenever it is desired to recover the data. Operational keys are not generated until they are needed, so they are not exposed to compromise, nor is additional storage required.

There are also two serious deficiencies in this method. The first is that all of the defined external interfaces are accessible only to the assembly language programmer. An exception to this is the Access Method Services REPRO command which may be used to encipher or decipher an entire file. The REPRO command is supported via the Access Method Services Cryptographic Option by using the services of the Programmed Cryptographic Facility for enciphering and deciphering operations and the key management function. This option is only intended to enhance the security of data while it is offline for storage purposes (onsite or offsite) or for transportation from one site to another. The enciphered file is always a sequential file; the input to the enciphering process or the output of the deciphering process may be any data set organization that is supported by Access Method Services. The exposure in using this feature is that all records exist in clear text, rather than just the record being processed currently; this method is clearly unacceptable for data base use.

The second serious deficiency in the Programmed Cryptographic Facility is the lack of speed. As with any complex algorithm there will be many possible implementations. Timing estimates will be difficult to obtain without actually coding the algorithm and then running it under tightly controlled conditions. Precise instruction timings are not available on large processors due to the use of pipelining in either or both the instruction fetch unit and the execution unit, and the use of a cache memory. The actual implementation of the DES in the Programmed Cryptographic Facility is covered by a license agreement and, therefore, is not available for study. However, it is reasonable to assume that sufficient effort was expended to develop an implementation of the algorithm that is as efficient as possible. The ultimate in speed, of course, is to implement the algorithm in hardware; that is the subject of Chapter Seven.

Most commercial installations have standards that require application programs to be written in a high-level language such as COBOL, FORTRAN, or PL/I to improve the understanding of these programs by others for maintenance purposes and to protect against loss of expertise due to unplanned personnel turnover. Furthermore, there is a strong and growing trend to the use of data base management systems where the application programmer is expressly prohibited from executing his own input/output commands. Therefore, in order to use this product, assembly language subroutines must be written (and maintained) to accept data records from a calling program and issue the appropriate macro instructions to perform the enciphering and

deciphering operations. This introduces the possiblity of additional errors, and the assembler routines must be closely monitored for Trojan horses, trapdoors, and covert channels. In any case, changes would be required in the application programs that would require recompilation.

In some installations, recompilation of all affected programs presents numerous logistics problems. The actual recompilation may be done concurrently with use of the data which is to be converted to encrypted form; the only requirement is that program changes must be prohibited during this operation. Another problem area is the actual changeover: after the file has been encrypted, all programs must be thoroughly tested to insure proper functioning and that undesired modifications have not been introduced. The complexity of these problems is increased if any online, real-time system is involved or if the time required to make the changes impairs the regular functioning of the system. For example, a daily cycle which requires sixteen hours to execute leaves little room for errors, much less the opportunity to install any major modifications.

There is evidence that future enhancements to the language translators might include support for the Cryptographic Facility. When the Virtual Storage Access Method (VSAM) was first announced, a direct interface was available only to assembler language programmers through the ACB (access control block), RPL (request parameter list), and EXLST (exit list) macro instructions. VSAM was originally designed to take advantage of the virtual storage capabilities of the virtual operating systems, and certain system data sets (page space

and the system catalog) were specialized, modified versions of the standard VSAM organizations available at that time (Keyed Sequential Data Set, or KSDS, and Entry Sequenced Data Set, or ESDS). Since the KSDS organization was functionally a superset of the Indexed Sequential Access Method (ISAM) and outperformed ISAM easily by at least a two to one ratio, an interface was developed which would map ISAM input/output operations into VSAM requests, issue the request, and then map the VSAM return codes and feedback information into ISAM return codes and status bits. The ISAM Interface Program was activated during OPEN processing by examining each ISAM open request to determine if the target data set was ISAM or VSAM. No application program changes were required and no new VSAM features were accessible to the ISAM program. This support was only a compatibility mapping and further required that the ISAM program conform to all ISAM restrictions and requirements. There was no special support for debugging new ISAM programs, and some errors would not necessarily produce predictable results.

At a later date, the VS/COBOL compiler was announced and provided support for the VSAM data set organizations directly by implementing extensions to the ANSI standard. Therefore, it would not be surprising if an additional clause were to appear for the file definition, such as "DATA IS ENCRYPTED". It would not be difficult to parse this extra phrase during compilation and then at code generation time include the appropriate calls to the Cryptographic Facility to encipher or decipher data. The only requirement would be to standardize the location of the key, since it would be desirable to

write the encrypted key with the data. The goal is to make these changes transparent to the programmer, but in this case he would probably be required to reserve eight bytes in the logical record description for the encrypted key. It might be possible to avoid this overhead by writing a special record at the beginning of the file containing only the encrypted key, since it would be required when reading the data at a later time. At first glance, this approach would seem to complicate several procedures such as the need to sort the file and other utility program operations. Unless only a portion of the logical record is encrypted, most utility programs could do no more than copy the file, because it is unlikely that the encryption algorithm would preserve such characteristics as collating sequence.

Unfortunately, this approach would not be suitable for a language such as FORTRAN, where the programmer does not provide an explicit file declaration. It would also not be acceptable to the assembler language programmer because his interface is at a lower level, through the DCB (data control block) macro instruction, which specifies physical characteristics of the data and processing options. In order to be maximally transparent, the specification of encryption should be implemented as a job control language (JCL) data definition (DD) statement parameter, possibly as a DCB subparameter. This is not an unreasonable modification; new JCL parameters have continued to appear as new options and devices have become available. VSAM is a good example of this type of change: the parameter "AMP=AMORG" is specified in lieu of the DCB parameter to indicate that the required physical definition of the data set must be retrieved from the

appropriate VSAM catalog. JCL conversion routines are table driven for the express purpose of being easily modified; many reserved fields exist in internal control blocks which could be used to support encryption.

The issue of how to handle the key may be resolved in a very simple and direct manner by including the encrypted key in the standard data set label accompanying each data set. Standard labels are already required for data sets that reside on direct access devices, so requiring a tape volume to have standard labels if it contains an encrypted data set is not a serious drawback. Most large commercial installations have long recognized the benefits of standard labeled tape volumes and have either eliminated or severely restricted the use of unlabeled volumes. The problem of reserving space for the key in each record or in a special record at the beginning of the file is eliminated, and the logical records are in the same format as specified by the application program. Using this approach, the access methods (which perform physical input/output for the programmer) would issue appropriate calls to the Cryptographic Facility to encipher and decipher records on a timely basis. On input, records would be deciphered immediately prior to their being made available to the application program, whether they are moved to a buffer area in the program (move mode) or processed directly in the physical file buffer (locate mode). If processing in locate mode, it might be necessary to obliterate the previous logical record when a request for the subsequent record is received in order to provide maximum protection. Similarly, on output, logical records should be enciphered immediately

when passed to the access method for inclusion in an output buffer. The entire encryption process is almost totally transparent to the programmer; his only contact with the data is in clear text form.

The utility programs used to manipulate data would also require some minor modifications to support these encrypted files. The most obvious problem is that of propagation of the encrypted key from the standard label when the file is copied. The default for most utilities is to propagate DCB parameters, such as blocksize, unless instructed otherwise. Since the encrypted key would reside in a control block in main storage during normal processing, it would not be too difficult to pass this key to the output file. For this type of program, it would also be advantageous, from both performance and security viewpoints, to allow the processing of a file without the decryption and re-encryption overhead. It is still assumed that it is virtually impossible to sort an encrypted file.

The method outlined above does not address the need to support encrypted data bases. Furthermore, it would be a convenient capability to be able to encrypt different portions of a data base record with different keys. In this manner, only the payroll data could be used by the payroll organization because the payroll key would not be known to any other groups. Similarly, medical or personnel data could be restricted to those programs which were authorized access. Because a data base usually contains multiple record types, encryption would have to be specified with each record type in the schema for the data base. Since schemata are generally tightly controlled, they may safely contain encrypted keys so that the

keys will be available to the data base management system to decrypt existing records or to encrypt new records.

Two software implementations will be examined here. To an inexperienced programmer, the DES appears to be a formidable challenge, but it is really only a moderately complex programming exercise, given an adequately detailed specification such as in FIPS Publication 46. (21) Some of the difficulties in writing this program may also be attributed to the clerical effort involved in including all tables required to implement the many permutations, shifts, and table lookups. The first approach is to employ brute force. The System/370 instruction set has several byte-oriented instructions (MVI, CLI, STC, IC, TR) but lacks flexibility at the bit level. Therefore, a simplistic approach is to expand all quantities (input, key, subkeys) so that each bit is isolated in a byte by itself. After performing the calculation required, the resultant sixty-four byte quantity can be easily compressed to sixty-four bits. This implementation further aggravates any considerations of speed by requiring frequent memory references (both store and fetch) but requires limited memory (less that two kilobytes) for code, tables, and temporary workareas. One advantage of this approach is that debugging of the routine is simplified, because it is quite easy to print intermediate results for hand simulation to verify correct execution.

The second approach is to attempt to perform as much of the computation as possible in the general purpose registers. The only memory references required should be for the substitution functions,

some of the permutations, and fetching subkeys at each of the sixteen rounds. Shifting and exclusive-or operations can be performed much faster in registers and should also contribute to a slightly reduced program size. It is in this method that it becomes evident that the family of rotate (circular shift) instructions is conspiciously missing from the System/360 and System/370 processors, while it is common on most miniprocessors and microprocessors.

Both of these approaches were coded and tested in order to compare their relative speeds under the best possible conditions on the fastest IBM processors available. The System/360 Model 195 is a fully pipelined central processor with a basic cycle time of fifty-four nanoseconds. (51) Memory references handled by the cache memory require only one cycle; if the data is not in the cache, fourteen cycles are required. Input and output data sets were on 3330-equivalent direct access devices with a rated data transfer rate of eight hundred six kilobytes per second. The System/370 3033 Processor Complex is a large general purpose central processor organized into an instruction preprocessing function (IPPF) and execution function (E-function). (1, 55) The IPPF prepares instructions and fetches operands prior to passing them to the E-function for execution. The CPU cycle time is fifty-seven nanoseconds and an access can be made to the high-speed buffer on each machine cycle; true memory references require eight machine cycles. Input and output data sets for this test were also on 3330-equivalent devices. System software on the two processors was OS/360 MVT and OS/VS2 MVS respectively, the most powerful general purpose batch

operating systems available for each. The benchmark results are summarized in Table II.

The benchmark consisted of five runs of both the memory and register algorithms, each run composed of enciphering and deciphering twenty-five hundred four hundred byte logical records (one million bytes of data) with ciphertext feedback. No claims are made as to the absolute efficiency of these particular implementations. The register implementation was faster on both machines (as expected); the improvement was more significant (33.3%) on the Model 195 due to core memory. The 3033 Processor has N-channel MOSFET semiconductor memory and the improvement was almost negligible (2.0%). A relative measure of the time required to encipher the data is the enciphering time coefficient (the ratio of the time to encipher to the time to do an ordinary move). (35) Reruns of the benchmark with calls to enciphering and deciphering routines removed showed that one million bytes may be read and written in approximately five to seven seconds. This yields enciphering time coefficients ranging from thirty-two (for the 3033 register algorithm) to forty (for the 195 memory algorithm). It is appropriate to observe that in the average commercial environment, one megabyte of data is a relatively small amount, occupying only about four cylinders on a 3330-equivalent disk volume or less than two cylinders on a 3350-equivalent disk volume. Even on the 3033 Processor, the enciphering or deciphering operation would require in excess of two and one half minutes per megabyte of data. Thus a maximum of five hundred forty megabytes of data could be enciphered or deciphered in one day of operation. Even if these times

TABLE II

Benchmark Results

	Memory		Register	
Run	Encrypt	Decrypt	Encrypt	Decrypt
195, 1	278.75	278.70	185.92	185.87
195, 2	278.70	278.59	185.90	185.90
195, 3	278.33	278.57	185.89	185.92
195, 4	278.38	278.62	185.93	185.89
195, 5	278.62	278.51	185.96	185.87
195, Average	278.556	278.598	185.920	185.890
3033, 1	163.31	163.36	159.85	159.78
3033, 2	163.15	163.28	159.87	159.80
3033, 3	163.21	163.11	159.90	159.89
3033, 4	163.34	163.35	160.02	159.88
3033, 5	163.24	163.07	159.94	159.61
3033, Average	163.250	163.234	159.916	159.992

could be improved by a factor of ten, it would still triple the amount of time required to process a given amount of data. This is a significant limitation which can be overcome only by implementing the algorithm in hardware, despite claims of software algorithms which perform at hardware speeds. (117) This approach will be examined in Chapter VII.

CHAPTER VII

HARDWARE ENCRYPTION

The software described in the preceding chapter can solve most of the logistics problems which arise when dealing with a requirement to encrypt data. Keys may be generated and assigned automatically and the keys themselves are encrypted before they are stored on a direct access device. These functions have, out of necessity, been implemented in software but it is undesirable that the encryption algorithm also be a software function. There is a problem of safeguarding the key and data while the encryption algorithm is operating. One solution is to perform encryption only in protected areas of memory and to disallow any interruptions during execution of the encryption routine. A greater problem is the intrinsic limitation of the speed of the central processing unit. Regardless of the efficiency or sophistication of the coding of the algorithm, the volume of data that can be processed is limited. Additionally, considering that the algorithm was designed to be implemented on an LSI chip, a hardware implementation would be far superior to a software implementation.

Several semiconductor manufacturers have developed chips which implement the Data Encryption Standard for varying applications. The Motorola MC6859 Data Security Device is a monolithic MOS n-channel integrated circuit. (82) It is compatible with the M6800 microprocessor family and appears as an interface adapter device.

This twenty-four pin device has an eight-bit data bus, thus requiring eight cycles to load key, load data, or read data. Encryption or decryption requires three hundred twenty cycles, a minimum of three hundred twenty microseconds. Therefore, maximum throughput is fifty kilobytes per second. The Fairchild 9414 Data Encryption Set is a bit-sliced four-chip set of forty-pin integrated injection logic devices. (33) Separate inputs and outputs permit a block to be processed in twenty-four clock pulses at five megahertz, resulting in slightly over sixteen hundred kilobytes per second throughput.

Both devices have similar circuit requirements but aside from the obvious differences in speed, the two devices have different potential audiences. The Motorola device is very attractive to the microprocessor user; suggested methods for using the device are supplied in the data sheet, including the necessary connections to implement ciphertext feedback. Only a summary of the chip organization is given for the Fairchild device, presumably due to the assumed skill of anyone who would use their device.

Several other manufacturers have also announced data-encryption chips with a variety of features. (10, 15, 50) Selection of possible chips for a specific application may be easier after establishing the number of keys required, speed, power supply voltages available, microprocessor compatibility, and an estimate of effort required to integrate the device into a system. (45) One other factor exists which may be significant. The NBS has established a testing facility to verify that a chip complies with the standard. Both a short test (two hundred thirty-five encryptions and fifty-six decryptions) and a

lengthy Monte-Carlo procedure (eight million encryptions and six million decryptions) are available to insure correct operation and that plaintext or key never appear as ciphertext output. (46) If a chip is to be used in a government application, it is possible that NBS certification may be required.

IBM has announced several hardware devices which implement the DES. The initial offering was for a pair of similar devices to encrypt transmitted data in communications systems. (56) The 3845 Data Encryption Device (table-top model) and the 3846 Data Encryption Device (rack mounted) are used to provide link encryption at the remote site and host site, respectively. These devices are almost totally transparent to the user of the data link, except for the additional overhead of transmitting a synchronizing message. This message varies in length from eight to forty-eight bits and is transmitted after each line turnaround or period of idle time, as a function of the line protocol used. The maximum end-to-end delay is sixty-eight bits for the start-stop protocol; sixty-six bits for the synchronous data link control (SDLC) protocol; and fifty-eight bits for the binary synchronous protocol. The actual time delay will be dependent upon the line speed; the maximum speed supported is nineteen thousand two hundred baud with modem clocking and a synchronous protocol. The start-stop protocol is limited to two thousand baud when internal clocking is used; modem clocking raises this limit to ninety-six hundred baud. True functional transparency is achieved because the encryption devices are placed between the transmission control unit or terminal and the modem. Specifically, no special

carrier requirements are imposed and nine, ten, or eleven bit transmission codes are supported. Both half and full duplex communications are supported over switched lines and point-to-point or multipoint networks on leased lines. The key used to encrypt data, as well as specification of the specific communication environment, is entered through a hand-held Personalization/Key Entry Unit (a calculator-sized sixteen digit keypad and display).

These two devices can significantly enhance the security of a communications environment, provided the user can maintain physical security over the devices and has adequate procedures for generation, selection, and distribution of the key variables.

The IBM 3848 Cryptographic Unit is a channel-attached device which provides the capability to encrypt data prior to writing it to an external medium or transmitting it over a communications link. (57) The Cryptographic Unit Support Program Product provides an access method for the 3848 Cryptographic Unit, including error recovery and error recording. (87) In addition, The Cryptographic Unit Support includes all key management functions of the Programmed Cryptographic Facility and provides a compatible interface to application programs (they do not have to be recoded, recompiled, or re-linkedited). It is possible to operate the 3848 Cryptographic Unit without the Cryptographic Unit Support because it adheres to the standard System/370 input/output interface rules. This is not recommended, since the error recovery and error recording functions are assumed to enhance the availability and maintenance characteristics of the device. (58) Furthermore, the key management

functions and interface to the application programs would have to be developed, certainly a non-trivial task. Therefore, in this discussion of the 3848 Cryptographic Unit, it will be assumed that the Cryptographic Unit Support Program Product is providing the necessary software support to drive the unit.

The 3848 Cryptographic Unit consists of a control function and an encryption unit. The control function manages data movement to and from the encryption unit and handles the input/output interface protocols required by the System/370 channel circuitry. The encryption unit contains the master key storage, parity check circuits, block chaining circuits (ciphertext feedback is used to further disguise encrypted data) and a pair of DES circuits. The two DES circuits process all data in parallel and return an error indication (unit check and appropriate sense bits) if the outputs of the two circuits do not agree. Since an understanding of the operation of the 3848 Cryptographic Unit requires an understanding of System/370 input/output channel programming, a brief review of input/output channel programming is provided in the Appendix.

The 3848 Cryptographic Unit supports thirteen command codes, which may be divided into five control commands and eight commands for enciphering or deciphering data. The procedure for deciphering data is the same as enciphering with the exception of the channel command code, so only the enciphering operation will be described. The channel program to encipher data will normally consist of multiple channel command words in the following sequence:

1. load operational key and initial chaining value (ICV)

2. encipher data chain

3. read data

4. read output chaining value (OCV)

All keys and data must be multiples of eight bytes. Short blocks of data may be padded by the user or will be handled automatically by the Cryptographic Unit Support.

When the channel program begins, the first CCW transmits the operational key (eight bytes enciphered) and the initial chaining value (eight bytes clear) to the 3848. The enciphered operational key is deciphered, using the master key in the 3848, and placed in the working key registers. The clear ICV is placed in the chaining buffer and becomes the current chaining value (CCV). The second CCW transmits from one to five hundred twelve eight-byte blocks of data to the 3848 main buffer. The encryption unit processes the first two blocks of data and places the results in the output buffer. A third block of data is processed but not transferred to the output buffer until the previous data is read by the channel. The third CCW returns the processed data to the user. As the output buffer becomes available, the encryption unit processes additional eight-byte blocks of data until the main buffer is empty. The enciphering process occurs faster than the channel can transfer data and therefore is overlapped with channel operations. The second and third CCWs can be repeated if necessary, provided the command chaining bit is set in the CCWs. The command chaining bit instructs the 3848 to continue using the current operational key and current chaining value to encipher data. If the command chaining bit is off, then the 3848 must receive

a new operational key and initial chaining value. The last CCW in the chain reads the final chaining value, which is actually the last enciphered block of data enciphered again. This output chaining value is intended to be used in those situations where a short block of data must be enciphered.

There are several advantages of the 3848 Cryptographic Unit over pure software encryption. The channel command words allow for a great deal of flexibility in using the device. Since operational keys and initial chaining values are supplied in each request to the device, it is a relatively simple matter to share the device among multiple users. The Cryptographic Unit Support will also support multiple 3848 Cryptographic Units to provide greater capacity and redundancy. In fact, it is likely that an installation would have multiple units because on-site maintenance service is not available. The use of the command chaining bit to specify continued use of the operational key and current chaining value provides the capability to mix encryption and decryption operations in the same channel program by inserting a load operational key and initial chaining value command at appropriate places. CCWs are checked for valid sequencing to insure correct operation and to prevent misuse of the device by unauthorized users; this is one reason for the small output buffer and the lack of a diagnostic read capability. It is possible that one goal of the Cryptographic Unit Support might be to perpetuate a channel program to the device by using the transfer in channel command code. In a large system with massive volumes of data to be encrypted, this is not an impossible circumstance, given some of the restrictions on the device.

Significant, but not overwhelming, is the limitation of the one and a half million byte per second channel transfer rate; this figure is a design specification of the channel and only applies to actual data transfer. A recent IBM announcement has increased this limit to three million bytes per second. (24) The process of device selection and CCW fetching from the main memory will lower the effective data transfer rate.

A second limitation of the 3848 Cryptographic Unit is that only four thousand ninety-six bytes of data may be transferred to the device at one time. While this may be adequate to handle data base records or physical blocks of data for direct access devices, most applications would prefer that encryption be performed at the logical record level. This is a matter of risk in deciphering data. If an error is introduced into the enciphered text, the block chaining causes that error to be propagated. A fresh start at the beginning of each logical record would insure that one error would only cause the loss of one record. The fields of the logical record could still be arranged so that ciphertext feedback disrupts any patterns that could be used for a statistical analysis attack.

The biggest disadvantage of the 3848 Cryptographic Unit is the additional load on the input/output subsystem. If a user wishes to encipher a block of data before writing it to an external medium, he must issue additional instructions to encipher it. This request to encipher the data will result in an additional input/output request and will triple the volume of data transferred because the entire block must be written to the 3848 Cryptographic Unit, read back into

main memory, and then written to the storage medium. Since many performance problems are directly traceable to the input/output subsystem, it does not seem wise to impose an additional load on these components. There are two alternatives which should be considered to provide better performance in this critical area.

The first option is to place the necessary encryption circuits into the channel itself and encrypt the data as it passes through the channel from main memory to a device or vice-versa. Since the 3848 Cryptographic Unit has already established that it is possible to encrypt data faster than the channel can process it, it should not be too difficult to determine how to add these circuits to the channel and make the necessary changes to the channel program to request encryption.

The apparent data path from main memory to an input/output device is only one byte wide; this immediately presents a severe problem because the DES is a block cipher, requiring eight bytes of data in order to compute the cipher text. The channel would, therefore, be required to accumulate eight bytes of data, execute the encryption algorithm (which is only a combinatorial circuit), and then pass the eight bytes one at a time to the device. This modification represents a conversion from serial to parallel to serial data flow and is a very common circuit in many digital computers. The actual data transfer rate of the channel must be matched to the device; this matching would be done by the serial interface with the aid of a small buffer (sixteen bytes) to support the parallel processing of the eight bytes by the encryption circuits. Buffering would also permit ciphertext

feedback to be used, a technique that is known to strengthen the DES. The only problem remaining is how to communicate to the channel the proper key value and the operation to be performed, encryption or decryption.

There are several different methods that could be used to transmit the key to the channel. It is necessary that the key be transmitted as close as possible to the time that it is required. This raises the consideration of whether to send the key to the channel in the same CCW as the data. Since the cryptographic program products suggest writing the (enciphered) data-encrypting key with the data, this would be a possibility. In this manner, the key would be readily available for decrypting the data at a later time when it is read from the device. Unfortunately, this method increases the length of each block of data by eight bytes, which could amount to a large quantity of wasted space in a large data set. It is also not obvious why it is desirable to package the key and data together, even if the key is encrypted. It would be preferable to store the key separately under control of some strong protection mechanism where it is not so readily available. This suggests that the key and initial chaining value should be transmitted to the channel by a separate CCW, for example, transmit cryptographic key (TCK). The channel would then load the working key register and the chaining buffer and require that the next CCW contain either a read or write data command code. Write count, key, and data or write key and data would also be accepted, but only the data portion of the record would be encrypted. Count fields must remain in clear text to verify proper hardware operation and key

fields to allow hardware searches for the data block required. Command chained reads and writes would also be permitted; and at the end of the data transfer sequence, the working key register would be reset to some standard value such as all zeros or all ones. Keys transferred to the channel should be enciphered themselves, as in the cryptographic products. Therefore, a similar mechanism to transmit the host master key to the channel would be required and would be performed at each initial program load (IPL) of the system and whenever the host master key was changed.

It is very easy to make the assumption that if data is being written from main memory to a device, then the desired operation is encryption, and if data is being read the desired operation is decryption. This is not the case, however, if the application is designed to use the inverse method, as it is in implementing digital signatures in public-key cryptosystems. Therefore, the only conclusion that can be drawn is that under normal circumstances data being written to a device should be enciphered and data being read from a device should be deciphered. Since a separate CCW is used to transmit the key to the channel, this CCW may also specify if the operation to be performed is not the standard one. There are many flag bits in the CCW and it is tempting to assign one of the unused (reserved) bits to indicate the inverse function. If the bit were a zero the standard function would be performed; and if it were a one, the opposite function would be selected. Examination of the other flag bits reveals that their meanings are not unique to a particular operation code but rather are instructions to the channel on how to

execute this CCW (suppress incorrect length indication, request indirect data addressing, suppress transfer of data to main storage) or what to do when the operation indicated by this CCW is complete (command chaining, data chaining). Therefore, it would not be a good design to appropriate another bit which would apply only to one operation code, nor would it be safe to corrupt an existing bit by assigning a second meaning to it. The only reasonable option available is to define a second operation code to specify the cryptographic key to be used and to indicate that the inverse of the standard operation is to be performed, for example, transfer cryptographic key inverse (TCKI).

The insertion of this additional CCW (TCK or TCKI) at the appropriate place in the channel program with the main storage address of the enciphered key and data length of eight bytes would be done by the access method which already builds the channel program. A user who wishes to code his own channel programs would be responsible for the encryption CCWs just as he is responsible for the remainder of the channel program. If the system were to provide totally automatic and transparent encryption and decryption services, then there would be no need for them because the user would not be involved and would always see only clear text. The need for encryption and decryption would be specified to the access method along with all other parameters used to describe the data characteristics as a data control block (DCB) subparameter. Mechanisms for doing this were described in the previous chapter and will not be repeated here, but some additional justification for the access method to assume the burden of

constructing the proper channel program is appropriate.

Channel programs are built dynamically during the OPEN process by routines called executors. These routines collect information from a number of sources and then acquire main storage for an assortment of control blocks, including channel programs. An example of a decision made by the access method in constructing a channel program for a data set on a direct access device is whether the device supports Rotational Positional Sensing (RPS). Typically, direct access channel programs contain two more CCWs for RPS devices than for non-RPS devices. Therefore, it would not be difficult to insert an additional CCW (TCK, TCKI) during channel program construction to request that the channel perform encryption or decryption operations.

The above method of encryption and decryption by the channel reduces the volume of the data transferred almost back to its original level, except for the additional CCW fetch and sixteen-byte key and chaining value transfer. For practical purposes, then, this method is as nearly transparent as it can be. A large benefit may accrue due to the elimination of sensitive residue on direct access temporary work files. Many installations must overwrite these temporary files at the conclusion of a compiling or sorting step, a process that can add considerably to the input/output load on the system. If the data is enciphered, then it is unusable by anyone, and the key exists for such a short time that it would be very difficult to capture both the key and the temporary data file. Therefore, in data for which the DES provides adequate security, encrypted residue is as safe as overwritten residue. This low-cost method would also permit

encryption of permanent data sets that are somewhat sensitive, but not sensitive enough to require encryption at a higher cost.

A disadvantage of this method is that data must be written to a device in order for it to be encrypted. If the purpose of encryption is to conceal information, then there are essentially no applications which would require that data be encrypted while it resides in main memory, given effective memory protection and isolation of users from each other. Another similar limitation of this method is that the entire physical record is encrypted by the same key. This is clearly unacceptable in the data base environment where access is permitted to only a portion of a record by different users of the data (medical, payroll, personnel). The second hardware option addresses this problem by recommending an instruction for the central processor, rather than the channel, to perform the encryption and decryption operations.

An instruction to encrypt or decrypt data would provide a great deal of flexibility to the application programmer. In addition to solving the data base type problem described above, it would also permit the encryption of only sensitive portions of data records, thus allowing less sensitive data to remain in clear text. This selective encryption capability would reduce overhead to the minimum by not encrypting data unnecessarily. One potential problem of this type of instruction is the complexity of the operation to be performed and the time that would be required to execute it. There are a significant number of extremely complex instructions that are already implemented by existing large central processors, but probably the best examples

are those which exist to support hardware emulation. These instructions are not defined in the basic operation manuals for a given processor but are usually provided in supplementary documentation that is particular to the precise emulator feature selected. The function of this type of instruction usually is to support a data organization not intrinsic to the real hardware, such as the variable-length data words permitted on the IBM 7070 central processor, which are supported by a "buffer unload" instruction on the System/370 Models 165 and 168 in the 7070 emulator feature.

On the System/370, a pair of instructions, encrypt and decrypt, would be a preferred implementation. Of course, due to the length of data to be enciphered, these obvious storage-to-storage instructions should be implemented as are the move long (MVCL) and compare logical long (CLCL) instructions, using four general purpose registers. The four registers would contain input address, output address (which may be the same as the input address), key address (which is a sixteen-byte field for encrypted key and ICV), and data length (which must be a multiple of eight). Like the two existing instructions, these new instructions would be interruptable at the completion of a unit of operation (eight bytes). Ciphertext feedback would be handled by placing the current chaining value (CCV) in the storage location (+8) addressed by the key register; at the end of the operation, this location would contain the output chaining value (OCV), re-enciphered, just as with the Cryptographic Unit Support.

Neither of the methods described above provides an adequate mechanism for all users who require encryption of data. Together, these methods do provide the tools to permit selection of the proper method to use. Encryption and decryption by the channel would typically be used for communication lines and entire data files. The encrypt and decrypt instructions would be used for selective encryption of data records. These two hardware capabilities would be exploitable by users in many areas and significantly extend the ability to encrypt data without unduly impacting performance.

CHAPTER VIII

CONCLUSIONS AND THE FUTURE

Technological innovations are often credited with solving substantial problems, but may also be blamed for the creation of new problems. (67) Several years ago, electronic funds transfer (EFT) systems were regarded as an innovation which would rapidly lead to a checkless, cashless society. These systems have not achieved widespread acceptance due to resistance from a nervous consumer and a reluctant financial community. Incremental improvements in services, such as automated teller machines and the automated clearinghouse, have assured the existence of checks and cash for many years to come. Proponents of EFT systems assert that computer scientists should allow the existing laissez-faire development to continue and not interfere with the process. (64, 67) The vulnerabilities of computer systems are underestimated by many financial executives, and the need for additional security measures is not recognized. (64, 91)

Parker has drawn upon his studies of computer abuse and documented the potential for intentional acts against consumers and institutions. (91) He further observed that although the crimes committed have traditional names ("fraud, theft, larceny, embezzlement, conspiracy, extortion, sabotage, and espionage"), the modus operandi, technical safeguards, and controls are all new. Electronic funds transfer systems have moved the protection of assets from (relatively) secure vaults into the realm of computers and data

processing. Thus, security is a proper concern for these systems.

There are three elements of security in payment systems:

1. The relationship of the financial institution and its customers, especially how "orders to pay" will be executed
2. The identification method for the customers
3. The financial integrity of the institutions. (71)

The last two items may be potentially more fraud-proof if the capabilities of innovative technology are used. Of primary concern is that the identification of customers and institutions (to one another) and the codes that they use be difficult to forge. If these values are stored in encrypted form within the computer, these private relationships may be more secure than paper systems. Development and testing of microprocessor driven "black boxes" to encrypt customer account numbers and personal identification numbers (PINs) is being done by the Interbank Card Association, which operates Master Charge. (8) The system uses the Data Encryption Standard and is designed so that the customer account number and PIN are never both available in plaintext outside the "black box."

Nearly all computer hardware contains some mechanisms that can be considered security features, but to be effective, they must be under the absolute control of an operating system designed to exploit these built-in features. In the past, these operating systems attempted to protect themselves from accidental errors; today the trend is to design and build in security and integrity from the beginning and even to offer additional software to enhance security. These are proper

and noble goals for a computer system, but may not resist determined attack by a knowledgeable opponent. Many file security systems depend on an indicator in the machine-readable labels that are associated with the data, or on an access code in a directory-type structure. An opponent who must operate within the realm of such access control mechanisms will have a more difficult time achieving his goal than one who can acquire a physical volume or gain access to the computer system in such a manner as to subvert the security controls. The only protection against such an intruder is to render the data unintelligible by some strong cryptographic transformation.

This was the goal of the National Bureau of Standards when it initiated a search for such an algorithm, now known as the Data Encryption Standard (DES). It is not clear that the goal has been met because the winning algorithm, submitted by the IBM Corporation, has not been approved for the protection of classified data. It may, however, be satisfactory for the commercial user because the only known approach to break this cipher is to attempt all possible key values until a solution is found. Despite the strong criticism of this particular algorithm, encryption does offer data protection at an ever-decreasing cost and, hopefully, continually increasing convenience. There are promising developments in many areas which may contribute to better acceptance of encryption techniques. These will be briefly discussed.

VLSI

Very large scale integration, or VLSI (very high speed integrated circuits, or VHSIC to the Department of Defense), has provided a new capability to circuit designers which has not yet been fully exploited. (16) Initial use of the one hundred thousand-plus gate chips has been in answering the need for larger and faster random access memories, the traditional proving ground for new technologies. Many familiar problems exist in propagating this technology to broader applications: fabrication resolution, size of production run, heat dissipation, and power supply voltages. A trade-off is available by producing uncommitted logic arrays (ULAs) containing standard cells which are interconnected to become a semi-customized chip based on the customer's design specifications. (3, 11) These gate arrays are called an interim measure by some of the manufacturers because they do not use all of the gates on a chip, thus wasting some of the silicon; the availability of a prototype chip in a few days or weeks outweighs this concern for many users of this technique. Other limitations are physical limits in lithography and packaging. (11, 107) Submicrometer geometries dictate a change from optical to X-ray lithography using laser interferometry to align two gratings to achieve the high resolution required. The problems associated with heat removal, power distribution, and interconnection have not benefited from LSI advances and are yet to be solved.

Considering the history of the semiconductor industry, it is only a matter of time before VLSI takes its place in products on the commercial market. The capability to process more data faster will

certainly be applied to cryptographic circuits to perform encryption and decryption more cheaply than ever before. Encryption will then be a choice for providing data security to anyone who requires it at little additional cost in processing time. The DES may not necessarily be the algorithm in use; in fact VLSI may be the technology which breaks the DES and leads to the development of a new standard.

DBMS

Data bases and data base management systems have achieved popularity due to several inherent advantages over previous data organizations where each record contained all relevant data. (2) These advantages include flexibility to add new data and associate it with existing records, the elimination of costly duplication of updates and file storage space, and easier program maintenance. These new techniques, however, also have some disadvantages. The systems are often complex, requiring highly trained personnel to reap the benefits described above, and also requiring some changes in traditional organizational structures. A data base administrator is responsible for logical and physical design of the data base, including security aspects. The application programs must be prohibited from performing their own reads and writes to the data base; rather, they must invoke a central access mechanism to request the necessary input/output operations. Typical data base management systems include an audit trail feature (often called journaling or logging) which records before and after images of record

segments that are updated. Historical files may be maintained to satisfy legal requirements for retention of data. Both of these methods provide data for auditors to verify the proper functioning of the system and to analyze activity.

Another common feature in a data base management system is a data dictionary which contains descriptive information about each data element. (106) An extension of the data dictionary may include the names of programs or users that are allowed to access or update a given data element. Both the contents of the data base and the data dictionary are susceptible to unauthorized access. If the data is of sufficient value, then it may be necessary to protect it by cryptography.

There are two ways to use cryptographic transformations to achieve data base security. (40) The first is to include a user-controlled transformation (the user holds the key) as a part of the standard access control. The second is a system-controlled transformation to protect against illegal access paths such as browsing. User-controlled transformations are complex to implement because the desire to share data among users conflicts with their requirements for different levels of protection. System-controlled transformations are much easier to implement but may be less secure because the keys must exist inside the system.

International Data Flows

A large number of data processing systems operate international networks which provide computing services to users in several

countries. (112) These systems include the obvious multi-national corporations, airline reservation systems, an international banking network, and service bureaus. It is the latter that arouses concern in countries which must entrust large volumes of sensitive data to a foreign data processing operation. Many of the problems are viewed in terms of privacy and sovereignty but in reality these issues take second place to economics. (97) A developed nation does not want to be dependent on another for data processing, or for any raw data.

Data security in this environment is also a concern. (112) Physical and software security is still the basis for new developments such as cryptography. The use of encryption is questionable because of a nation's sovereign right to regulate telecommunications and monitor transmissions. No standards for international data security are being developed and the wide variety of technical characteristics and quality of equipment is sure to complicate any efforts. The existence of multiple regulatory authorities may impede the use of any new technology that emerges; it is a difficult task to stimulate innovation without a set of fair, streamlined regulations. (48)

Language Extensions

An alternative to depending on the operating system to provide an access control mechanism is to incorporate that facility into a programming language. (61) By integrating access declarations into the program, a number of benefits will accrue. In addition to enhancing readability and understanding of the program, some checking

of the programmer's intent may be performed at compile time. This type of access control mechanism in a programming language is similar to a capability-based protection mechanism that might be found in an operating system which allows the controlled sharing of data.

Summary

The data processing industry is changing rapidly due to improvements in both hardware and software. The power to process ever-increasing amounts of data has created more possibilities for potential misuse of the computer. (96) Legal measures will not convince the public that their privacy is assured. The data processing industry must take the lead in demonstrating the safety of stored data in existing systems before new systems can be justified. Encryption offers a significant enhancement to all other security measures in force, whether they be physical, hardware, or software. Above all, to be effective and lasting, security must be a primary design objective in any new endeavor. Future systems must be designed with built-in protection for data and processing power to thwart potential adversaries; unless adequate safeguards are included, that determined opponent will succeed in perpetrating another computer abuse, further destroying public confidence in the industry.

LIST OF REFERENCES

LIST OF REFERENCES

1. A Guide to the IBM 3033 Processor Complex, Attached Processor Complex, and Multiprocessor Complex of System/370. GC20-1859-4. White Plains: IBM Corporation, 1979.

2. Adams, D. L. "The Data Security Aspects of Data Base Systems," paper presented at IBM Data Security Forum, Denver, Colorado, September 1974.

3. "An Old Cost Cutter Catches On," Business Week. Industrial Edition Number 2633, April 21, 1980, pp. 134D-134J.

4. Anderson, J. P. Computer Security Technology Planning Study. Volume II, Report ESD-TR-73-51, 1972.

5. Attanasio, C. R. "Operating System Architecture and Integrity," paper presented at IBM Data Security Forum, Denver, Colorado, September 1974.

6. Attanasio, C. R. "Virtual Control Storage--Security Measures in VM/370," IBM Systems Journal. Volume 18, Number 1, 1979, pp. 93-110.

7. Attanasio, C. R., Markstein, P. W., and Phillips, R. J. "Penetrating an Operating System: a Study of VM/370 Integrity," IBM Systems Journal. Volume 15, Number 1, 1976, pp. 102-116.

8. "Banking by Encryption," IEEE Spectrum. Volume 17, Number 1, January 1980, p. 24.

9. Beardsley, C. W. "Is Your Computer Insecure?" IEEE Spectrum. Volume 9, Number 1, January 1972, pp. 67-78.

10. Beaston, J. "One-chip Data-encryption Unit Accesses Memory Directly," Electronics. Volume 52, Number 16, August 2, 1979, pp. 126-129.

11. Bernhard, R. "Solid State Looks to VLSI," IEEE Spectrum. Volume 17, Number 1, January 1980, pp. 44-49.

12. Bigelow, R. P. and Nycum, S. H. Your Computer and the Law. Englewood Cliffs: Prentice-Hall, Inc., 1975.

13. Branstad, D. K. "Encryption Protection in Computer Data Communications," Fourth Data Communications Symposium. October 1975, pp. 8-1--8-7.

14. Broadman, I. S. "Protection Techniques in Data Processing Systems to Meet User Data Security Needs," Second International Conference on Computer Communications, 1974, pp. 485-489.

15. Budzinski, R. "Single-Chip Computer Scrambles for Security," Electronics. Volume 52, Number 15, July 19, 1979, pp. 140-144.

16. Christiansen, D. "Technology '80," IEEE Spectrum. Volume 17, Number 1, January 1980, pp. 30-31.

17. Cohen, E. and Jefferson, D. "Protection in the Hydra Operating System," Operating Systems Review. Volume 9, Number 5, November 1975, pp. 141-160.

18. "Computer Security and Controls," Datapro 70. Delran: Datapro Research Corporation, 1979.

19. Computer Security Research Group, D. B. Hoyt, Chairman. Computer Security Handbook. New York: Macmillan and Company, Inc., 1973.

20. Corbato, F. J., Saltzer, J. H., and Clingen, C. T. "Multics--The First Seven Years," AFIPS Conference Proceedings, 1972 Spring Joint Computer Conference. Volume 40, pp. 571-583.

21. "Data Encryption Standard," Federal Information Processing Standard (FIPS) Publication 46, National Bureau of Standards, U. S. Department of Commerce, Washington, D. C., January 1977.

22. Data Security and Data Processing Volume 2 Study Summary. G320-1371. White Plains: IBM Corporation, 1974.

23. Data Security and Data Processing Volume 6 Evaluations and Installation Experiences: Resource Security System. G320-1376. White Plains: IBM Corporation, 1974.

24. "Data Streaming Feature (#4850) for 3033, 3032, 3031 Processors and 3042 Attached Processor Model 2," IBM Data Processing Division Product Announcement, June 11, 1980.

25. DECsystem-10 Assembly Language Handbook. DEC-10-NRZB-D. Maynard: Digital Equipment Corporation, 1971.

26. Denning, P. J. and Denning, D. E. "Data Security," Computing Surveys. Volume 11, Number 3, September 1979, pp. 227-249.

27. Diffie, W. and Hellman, M. E. "Exhaustive Cryptanalysis of the NBS Data Encryption Standard," Computer. Volume 10, Number 6, June 1977, pp. 74-84.

28. Diffie, W. and Hellman, M. E. "Multiuser Cryptographic Techniques," AFIPS Conference Proceedings, 1976 National Computer Conference. Volume 45, pp. 109-112.

29. Diffie, W. and Hellman, M. E. "New Directions in Cryptography," IEEE Transactions on Information Theory. Volume IT-22, Number 6, November 1976, pp. 644-654.

30. DOE Manual 5636. Washington: Department of Energy, 1979.

31. Ehrsam, W. F., Matyas, S. M., Meyer, C. H., and Tuchman, W. L. "A Cryptographic Key Management Scheme for Implementing the Data Encryption Standard," IBM Systems Journal. Volume 17, Number 2, 1978, pp. 106-125.

32. Evans, A., Jr., Kantrowitz, W., and Weiss, E. "A User Authentication Scheme Not Requiring Security in the Computer," Communications of the ACM. Volume 17, Number 8, August 1974, pp. 437-442.

33. Fairchild Camera and Instrument Corporation 9414 4-Chip Data Encryption Set, Preliminary Data Sheet, March 1979.

34. Feistel, H. "Cryptography and Computer Privacy," Scientific American. Volume 228, Number 5, May 1973, pp. 15-23.

35. Friedman, T. D. and Hoffman, L. J. "Execution Time Requirements for Encipherment Programs," Communications of the ACM. Volume 17, Number 8, August 1974, pp. 445-449.

36. Gaines, H. F. Cryptanalysis. New York: Dover Publications, Inc., 1956.

37. Gladney, H. M., Worley, E. L., and Myers, J. J. "An Access Control Mechanism for Computing Resources," IBM Systems Journal. Volume 14, Number 3, 1975, pp. 212-228.

38. Gold, B. D., Linde, R. R., Peeler, R. J., Schaefer, M., Schied, J. F., and Ward, P. D. "A Security Retrofit of VM/370," AFIPS Conference Proceedings, 1979 National Computer Conference. Volume 48, pp. 335-344.

39. Gold, B. D., Linde, R. R., Schaefer, M., and Scheid, J. F. "VM/370 Security Retrofit Program," ACM 77 Proceedings of the Annual Conference, pp. 411-417.

40. Gudes, E., Koch, H. S., and Stahl, F. A. "The Application of Cryptography for Data Base Security," AFIPS Conference Proceedings, 1976 National Computer Conference. Volume 45, pp. 97-107.

41. Hansen, P. B. Operating Systems Principles. Englewood Cliffs: Prentice-Hall, Inc., 1973.

42. Hebbard, B. and others. "A Penetration Analysis of the Michigan Terminal System," Operating Systems Review. Volume 14, Number 1, January 1980, pp. 7-20.

43. Hemphill, C. F., Jr. and Hemphill, J. M. Security Procedures for Computer Systems. Homewood: Dow Jones-Irwin, Inc., 1973.

44. Herbert, A. J. "A New Protection Architecture for The Cambridge Capability Computer," Operating Systems Review. Volume 12, Number 1, January 1978, pp. 24-28.

45. Hindin, H. J. "Encryption Chips Sort Themselves Out," Electronics. Volume 53, Number 13, June 5, 1980, pp. 96-97.

46. Hindin, H. J. "LSI-based Data Encryption Discourages the Data Thief," Electronics. Volume 52, Number 13, June 21, 1979, pp. 107-120.

47. Hoffman, L. J. Modern Methods for Computer Security and Privacy. Englewood Cliffs: Prentice-Hall, Inc., 1977.

48. Horton, D. J. "Challenges in the Planning of International Communications," AFIPS Conference Proceedings, 1978 National Computer Conference. Volume 47, pp. 713-715.

49. Hsiao, D. K., Kerr, D. S., and Madnick, S. E. Computer Security. New York: Academic Press, Inc., 1979.

50. Humphrey, T. and Toth, F. L. "Two-chip Data-encryption Unit Supports Multi-key Systems," Electronics. Volume 53, Number 2, January 17, 1980, pp. 136-139.

51. IBM System/360 and System/370 Model 195 Functional Characteristics. GA22-6943-2. White Plains: IBM Corporation, 1971.

52. IBM System/360 Operating System Concepts and Facilities. GC28-6535-8. White Plains: IBM Corporation, 1971.

53. IBM System/360 Operating System Introduction. GC28-6534-4. White Plains: IBM Corporation, 1972.

54. IBM System/370 Principles of Operation. GA22-7000-4. White Plains: IBM Corporation, 1974.

55. IBM 3033 Functional Characteristics. GA22-7060-4. White Plains: IBM Corporation, 1979.

56. IBM 3845 Data Encryption Device IBM 3846 Data Encryption Device. GA27-2865-2. White Plains: IBM Corporation, 1978.

57. IBM 3848 Cryptographic Unit Product Description and Operating Procedures. GA22-7073-0. White Plains: IBM Corporation, 1979.

58. "IBM 3848 Cryptographic Unit, an Addition to IBM Security Products Using Cryptography," IBM Data Processing Division Product Announcement, November 13, 1979.

59. Jarvis, J. E. "The Many Faces of Multics," The Computer Journal. Volume 18, Number 1, February 1975, pp. 2-6.

60. Jensen, R. M. "A Formal Approach for Communication Between Logically Isolated Virtual Machines," IBM Systems Journal. Volume 18, Number 1, 1979, pp. 71-92.

61. Jones, A. K. and Liskov, B. H. "A Language Extension for Controlling Access to Shared Data," IEEE Transactions on Software Engineering. Volume SE-2, Number 4, December 1976, pp. 277-285.

62. Kahn, D. The Codebreakers. New York: The New American Library, Inc., 1973.

63. Kline, C. S. and Popek, G. J. "Public Key vs. Conventional Key Encryption," AFIPS Conference Proceedings, 1979 National Computer Conference. Volume 48, pp. 831-837.

64. Kling, B. "Introduction to the EFT Symposium," Communications of the ACM. Volume 22, Number 12, December 1979, pp. 639-640.

65. Kolata, G. B. "Computer Encryption and the National Security Agency Connection," Science. Volume 197, Number 4302, July 29, 1977, pp. 438-440.

66. Konheim, A. G., Mack, M. H., McNeill, R. K., Tuckerman, B., and Waldbaum, G. "The IPS Cryptographic Programs," IBM Systems Journal. Volume 19, Number 2, 1980, pp. 253-283.

67. Kraemer, K. L. and Colton, K. W. "Overview of the EFT Symposium," Communications of the ACM. Volume 22, Number 12, December 1979, pp. 641-643.

68. Lampson, B. W. "Protection" in Proceedings Fifth Princeton Symposium on Information Sciences and Systems, Princeton University, March 1971, pp. 437-443, reprinted in Operating Systems Review. Volume 8, Number 1, January 1974, pp. 18-24.

69. Levin, R., Cohen, E., Corwin, W., Pollack, F., and Wulf, W. "Policy/Mechanism Separation in Hydra," Operating Systems Review. Volume 9, Number 5, November 1975, pp. 132-140.

70. Linden, T. A. "Operating System Structures to Support Security and Reliable Software," Computing Surveys. Volume 8, Number 4, December 1976, pp. 409-445.

71. Long, R. H. "Public Protection and Education with EFT," Communications of the ACM. Volume 22, Number 12, December 1979, pp. 648-654.

72. MacKinnon, R. A. "The Changing Virtual Machine Environment: Interfaces to Real Hardware, Virtual Hardware, and Other Virtual Machines," IBM Systems Journal. Volume 18, Number 1, 1979, pp. 18-46.

73. Mandell, M. Handbook of Business and Industrial Security and Protection. Englewood Cliffs: Prentice-Hall, Inc., 1973.

74. Matyas, S. M. and Meyer, C. H. "Generation, Distribution, and Installation of Cryptographic Keys," IBM Systems Journal. Volume 17, Number 2, 1978, pp. 126-137.

75. McKnight, G. Computer Crime. New York: Walker and Company, 1973.

76. McPhee, W. S. "Operating System Integrity in OS/VS2," IBM Systems Journal. Volume 13, Number 3, 1974, pp. 230-252.

77. Meyer, C. H. "Ciphertext/plaintext and Ciphertext/key Dependence vs. Number of Rounds for the Data Encryption Standard," AFIPS Conference Proceedings, 1978 National Computer Conference. Volume 47, pp. 1119-1126.

78. Michelman, E. H. "The Design and Operation of Public-Key Cryptosystems," AFIPS Conference Proceedings, 1979 National Computer Conference. Volume 48, pp. 305-311.

79. Millen, J. K. "Security Kernel Validation in Practice," Communications of the ACM. Volume 19, Number 5, May 1976, pp. 243-250.

80. Mills, R. G. "Management of the EDP Risk/Safeguard Tradeoff," paper presented at IBM Data Security Forum, Denver, Colorado, September 1974.

81. Mohan, C. "Survey of Recent Operating Systems Research, Designs and Implementations," Operating Systems Review. Volume 12, Number 1, January 1978, pp. 53-89.

82. Motorola Semiconductors MC6859 Advance Information, April 1980.

83. NOS Version 1 Reference Manual. 60435400 and 60445300 Revision J. St. Paul: Control Data Corporation, 1979.

84. Orr, K. T. "Systems Design, Structured Programming, and Data Security," paper presented at IBM Data Security Forum, Denver, Colorado, September 1974.

85. OS/MVT with Resource Security General Information and Planning Manual. GH20-1058-0. White Plains: IBM Corporation, 1971.

86. OS/VS1 and OS/VS2 MVS Programmed Cryptographic Facility General Information Manual. GC28-0942-2. White Plains: IBM Corporation, 1978.

87. OS/VS2 MVS Cryptographic Unit Support: General Information Manual. GC28-1016-1. White Plains: IBM Corporation, 1979.

88. OS/VS2 MVS Resource Access Control Facility (RACF) General Information Manual. GC28-0722-4. White Plains: IBM Corporation, 1978.

89. Palme, J. "Software Security," Datamation. Volume 20, Number 1, January 1974, pp. 51-55.

90. Parker, D. B. Crime by Computer. New York: Charles Scribner's Sons, 1976.

91. Parker, D. B. "Vulnerabilities of EFTs to Intentionally Caused Losses," Communications of the ACM. Volume 22, Number 12, December 1979, pp. 654-660.

92. Parker, D. B. and Nycum, S. "The New Criminal," Datamation. Volume 20, Number 1, January 1974, pp. 56-58.

93. Pinchuk, P. L. "Management Considerations of Installing a Secure Operating System," paper presented at IBM Data Security Forum, Denver, Colorado, September 1974.

94. Purdy, G. B. "A High Security Log-in Procedure," Communications of the ACM. Volume 17, Number 8, August 1974, pp. 442-445.

95. Rivest, R. L., Shamir, A., and Adleman, L. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM. Volume 21, Number 2, February 1978, pp. 120-126.

96. Rosenberg, J. M. "Human and Organizational Implications of Computer Privacy," AFIPS Conference Proceedings, 1976 National Computer Conference. Volume 45, pp. 39-43.

97. Safirstein, P. "How Do We Best Control the Flow of Electronic Information Across Sovereign Borders?" AFIPS Conference Proceedings, 1979 National Computer Conference. Volume 48, pp. 279-282.

98. Saltzer, J. H. "Protection and the Control of Information Sharing in Multics," Communications of the ACM. Volume 17, Number 7, July 1974, pp. 388-402.

99. Schroeder, M. D. "Engineering a Security Kernel for Multics," Operating Systems Review. Volume 9, Number 5, November 1975, pp. 25-32.

100. Seawright, L. H. and MacKinnon, R. A. "VM/370--A Study of Multiplicity and Usefulness," IBM Systems Journal. Volume 18, Number 1, 1979, pp. 4-17.

101. Sevcik, K. C. "Lessons on Security from an Operating System Design Project," paper presented at IBM Data Security Forum, Denver, Colorado, September 1974.

102. Shannon, C. "Communication Theory of Secrecy Systems," Bell System Technical Journal. Volume 28, Number 4, October 1949, pp. 656-715.

103. Short, G. E. "Establishing a Company Security Program," paper presented at IBM Data Security Forum, Denver, Colorado, September 1974.

104. Sinkov, A. Elementary Cryptanalysis. Washington: Mathematical Association of America, 1966.

105. "Statement of OS/VS2 (MVS) System Integrity," IBM Data Processing Division Programming Announcement, April 21, 1980.

106. Statland, N. "Data Security Considerations within a Data Base Management System," paper presented at IBM Data Security Forum, Denver, Colorado, September 1974.

107. Sumney, L. W. "VLSI with a Vengeance," IEEE Spectrum. Volume 17, Number 1, April 1980, pp. 24-27.

108. The Fair Credit Reporting Act, Public Law Number 91-508, 84 Stat. 1128 (1970).

109. The Privacy Act of 1974, Public Law Number 93-579, 88 Stat. 1896 (1974).

110. Tsichritzis, D. C. and Bernstein, P. A. Operating Systems. New York: Academic Press, Inc., 1974.

111. Tuchman, W. L. "Computer Security and IBM," Science. Volume 197, Number 4307, September 2, 1977, p. 938.

112. Turn, R. "Privacy and Security in Transnational Data Processing Systems," AFIPS Conference Proceedings, 1979 National Computer Conference. Volume 48, pp. 283-291.

113. Van Tassel, D. Computer Security Management. Englewood Cliffs: Prentice-Hall, Inc., 1972.

114. Webb, D. A. "Analytic Tools for the Examination of Security Aspects of Operating Systems," paper presented at IBM Data Security Forum, Denver, Colorado, September 1974.

115. Weiss, H. "Computer Security: An Overview," Datamation. Volume 20, Number 1, January 1974, pp. 42-47.

116. Whiteside, T. Computer Capers: Tales of Electronic Thievery, Embezzlement, and Fraud. New York: Thomas Y. Crowell Company, 1978.

117. Williams, D. "Can Software Do Encryption Job?" Electronics. Volume 53, Number 15, July 3, 1980, pp. 102-103.

118. Wulf, W., Cohen, E., Corwin, W., Jones, A., Levin, R., Pierson, C., and Pollack, F. "HYDRA: the Kernel of a Multiprocessor Operating System," Communications of the ACM. Volume 17, Number 6, June 1974, pp. 337-345.

119. Wulf, W. A. and Harbison, S. P. "Reflections in a Pool of Processors--An Experience Report on C.mmp/Hydra," AFIPS Conference Proceedings, 1978 National Computer Conference. Volume 47, pp. 939-951.

120. Wulf, W., Levin, R., and Pierson, C. "Overview of the Hydra Operating System Development," Operating Systems Review. Volume 9, Number 5, November 1975, pp. 122-131.

121. Yasaki, E. K. "Encryption Algorithm: Key Size is the Thing," Datamation. Volume 22, Number 3, March 1976, pp 164-166.

APPENDIX

APPENDIX

SYSTEM/370 INPUT/OUTPUT CHANNEL PROGRAMMING

System/370 input/output operations are implemented by a sophisticated combination of hardware and software. (54) The channel provides a standard interface for all device types to the central processing unit and main storage and allows overlap of instruction execution with input/output operations. Control units accept signals from the channel, map these channel requests into orders for the particular characteristics of each device, and return device status indicators to the channel. Individual devices are normally attached to one control unit and are addressable only through a single channel. In electromechanical devices, the control unit typically handles multiple devices, but in electronic devices the control unit may not be distinguishable from the device. In either case, the control unit is normally transparent to the channel program.

Input/output operations are initiated by the START I/O or the START I/O FAST RELEASE instruction. Both of these instructions specify the channel, control unit, and device address and cause the channel to fetch the Channel Address Word (CAW) from a fixed low memory address. The CAW contains a storage protection key and points to the first Channel Command Word (CCW), which is subsequently fetched by the channel to begin execution of the channel program. Each CCW contains a command code, a data address in main storage, the length of that data area, and several flag bits.

The command code of the CCW identifies to the channel the operation to be performed and is one of the following classes of operations: output forward (write, control), input forward (read, sense), input backward (read backward), or branching (transfer in channel). The most common operations (read, write, control) are specified by the two low order bits of the command code and the other operations (sense, read backward, transfer in channel) by the four low order bits. The remaining bits of the command code are modifiers which are interpreted by the device to which the command is addressed.

Many input/output operations may be accomplished by specifying a single CCW, such as the transfer of data from magnetic tape to a contiguous area of main storage. For other devices, such as direct access, several CCWs may be required because the device must perform positioning operations prior to data transfer. In the case of multiple CCWs, the automatic fetching of new CCWs as they are required by the channel is called chaining. Data chaining causes the operation in progress to continue and selects a new storage area from the new CCW. Command chaining initiates a new operation at the successful completion of the current operation. Other flags that may be specified include the capability to suppress an incorrect length indication, to suppress the transfer of data to main storage, to generate an interrupt when this CCW takes control of the channel, and to specify indirect data addressing.

VITA

William Chester Hood was born in Helena, Arkansas, on August 29, 1949. He attended elementary school in Wonder Lake, Illinois, and was graduated from Giles County High School in Pulaski, Tennessee, in May 1967. The following September he entered the University of Tennessee, Knoxville, and in June 1972 he received a Bachelor of Science degree in Electrical Engineering. After graduation he accepted employment with the Maremont Corporation and began study toward a Master's degree. His graduate studies were interrupted when he was employed by Provident Life and Accident Insurance Company in March 1974 and were resumed when he was employed by Union Carbide Corporation, Nuclear Division in August 1978. He received the Master of Science degree with a major in Computer Science from The University of Tennessee, Knoxville, in August 1980.

The author is a member of Eta Kappa Nu, the Institute of Electrical and Electronics Engineers Computer Society, and the Association for Computing Machinery.

He is married to the former Betty Self of Knoxville, Tennessee.