INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

MUTUAL AUTHENTICATION PROTOCOLS USING SMART CARD:

A New Approach for Authentication protocols

Tackling the Masquerader's Problem

by

Sikiru A. Fadairo

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

1998

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

7/6/93
Date

Chair of examining committee

7/14/93
Date

Executive Officer

Professor Steven Lucci

Professor Anthony Galatianos

Professor Stefan Burr
Editorial Reviewer

THE CITY UNIVERSITY OF NEW YORK

Abstract

MUTUAL AUTHENTICATION PROTOCOL USING SMART CARD

BY

SIKIRU A. FADAIRO

Adviser: Professor Michael Anshel and Professor Steven Lucci

It is widely recognized that the use of the existing authentication protocols only provides for one way authentication. In this paper we present a new model for mutual authentication protocol, which is applicable both in general secure network environments, and as a specific example the Smart card scenario. Our new protocol represents a significant technical advantage over existing similar protocols regarding performance, accuracy, dependability, and degree of redundancy. In particular, this is the first time the Blum-Blum-Shub generators have been utilized to maintain a highly secure authentication protocol environment.

Our protocol could be used in a distributed workstation environment, or by customers making financial transactions over the Internet. It could be implemented in hardware as an add-on board or as a co-processor; or as embedded communications, operating systems, or application software. We present a software implementation of our Blum-Blum-Shub authentication protocol.

The primary application we chose to exhibit our protocol is the Smart card. In doing so, we cover the hardware and software involved in implementation of our architecture based upon current and future technology. In addition, we plan to discuss the general issues of our protocol in future papers.

Acknowledgements

I am grateful to the many people who have helped me during my writing of this dissertation. I would like to express my deepest appreciation to Professor Michael Anshel who has been my mentor for many years and whose support and guidance have been crucial to the completion of this dissertation. He introduced me to the world of Cryptography and Smart card technology. His intellectual spirit sparked many of the ideas, which formed an important part of this dissertation. I could not have finished without his support and encouragement. This work was greatly influenced by Jose Luis Zoreda and Jose Manuel Oton's work on plastic cards, particularly their masterpiece, "Smart Card." I would also like to express my special thanks to Professor Steven Lucci, and Professor Anthony Galatianos for their knowledgeable advice and guidance. I could not have finished without their supports. I would also like to thank Professor Stanley Habib for his support. I should express my lifelong gratitude to my parents who passed on to me their values and attitude towards life. While they might not full understand my dissertation, they strongly believe that whatever their son is doing is also important to them. It is unfortunate that both of them are not alive today to see my completed work. Finally, I must thank my wife, Afotan, for all the sacrifices she has made for me during my many years as a student, and my children, Solabomi, Akinkunmi, Tinuke, and Olamide for their love, support and understanding. They fill my life with joy and hope.

CONTENTS

PART III

List of Figures

List of Tables

INTRODUCTION

There are many protocols that have been proposed recently by cryptologic researchers. Protocol formulation has recently gained new momentum and has become one of the most active areas of current cryptologic research, as well as one of the most difficult, especially when one seeks particular cryptographic functions to include in the protocol without compromise of their security. Authentication protocol is one of the most active research areas in computer security. It is increasingly gaining ground in a wide range of applications, especially from access to sites and commerce on the internet, to various kinds of financial transactions in the business world involving plastic cards. This protection system depends on its ability to identify to program and processes that are executing

Computer networks are susceptible to a variety of threats mounted by intruders as well as legitimate users of the system. Legitimate users are more powerful adversaries, since they possess internal state information not usually available to an intruder (except after a successful penetration of a host). File systems are also susceptible to threats because they contain information that is highly valuable to their users. Protecting this information against unauthorized usage is therefore a major concern of all file systems. These issues apply equally well to timesharing systems as to networks of personal computers connected to shared servers via local area networks.

In this dissertation, computer security is viewed as having at least two branches: data loss and intruders. At present, research in the field of computer security is heavily concentrated on intruders. They come in two forms: passive intruders who just want to access files they are not authorized to access. They read the files without changing their contents. Active intruders on the other hand are more malicious; they want to make unauthorized changes to the files. This includes message modification, message repetition, and delaying or deleting messages.

Based on recent research in computer security, Smart cards, credit-card-size devices with microchips embedded in them, have recently joined the world of plastic cards, covering different areas both in financial and non-financial applications. Security is one of the main features of the Smart cards. Data stored in the card's memory are managed by the microchips, which continuously supervises data flow and access restrictions. There is strong evidence that Smart cards may also be able to "challenge" the legitimacy of the external device being used for data reading. The motivation for this research is to show that built-in data enciphering in the card make the Smart cards the best solution for communicating sensitive information over unsecured channels and for applications in restricted environments. This built-in enciphering process can also help reduce card fraud in countries where Smart cards are extensively used, for example, France.

We present mutual authentication protocols using Smart card in this dissertation to show its possible use to solve both authentication and verification problems. The

need for the mutual authentication protocol using Smart card developed in this dissertation was due to the fact that security is a concern of organizations with assets that are controlled by computer systems.

This dissertation is organized into three (3) parts. Part I (chapters 1 and 2):
In chapter 1, Introduction to Computer Security, basic concepts about the need for computer security and the use of cryptography and encryption applications are discussed.

In chapter 2, Computer Security Overview, a general system security is introduced discussing software threats, security measures, authentication procedures, access controls, and auditing functions.

Part II (chapters 3, 4, and 5):
In chapter 3, Identity Verification, a framework for identity verification is defined. It encompasses identity verification structures and different verification methods employed.

In chapter 4, Cryptography, basic concepts and definitions about cryptosystems and cryptanalysis are introduced for the purpose of defining cryptography. We also discuss cryptography in a way to emphasize encryption coding as a basis for security in Smart card and cryptographic algorithm implementations in general.

In chapter 5, Key Management in Cryptography, general key management systems are discussed. Following that, a framework for key management in cryptography is defined which encompasses the overview of generating secure keys, key verification, key transfers, dealing with issues of compromised keys, and destroying keys. We also discuss in general, the conventional (one-key) system.

Part III (chapters 6-21):

We present our contribution in this part of the dissertation:

In chapters 6-9, we presented our motivation, design, development, and the schematic description of our Smart card.

In chapters 10 and 11, we identify major types of authentication networks in a computer network system and presented the general principles of authentication protocols.

In chapter 12, we present the main ideas that underline authentication protocol design by presenting various protocol examples. We further identify five areas of network system design and the related authentication needs.

In chapter 13, we present secure bootsrap protocol which is usually initiated when host hardware attempts a remote initialization. This protocol is necessary especially after a malicious attack by an adversary who attempts to penetrate the host. The

secure bootsrap protocol allows a re-initialized host to attain a "safe" state prior to resuming normal operation.

In chapter 14, we present User-Host authentication protocol. Because the potential for forgery is great in this environment, our contribution is the use of Smart card for mutual authentication. This is the first time mutual authentication has been used for authentication protocol. We further propose, as an alternative, a User-Host mutual authentication protocol using the Blum-Blum-Shub generators.

In chapters 15-17, we discussed another type of mutual authentication (Peer-Peer) and the cryptographic parameters needed to negotiate a secure connection oriented protocol. We further established that since both clients and servers are implemented as processes, basic protocol for Peer-peer authentication can also be applied to Client-Server authentication (chapter 16). In chapter 17, we discussed authentication protocols which distinguishes exchanges between principals belonging to the same domain (Intradomain) and exchanges between principals belonging to different domains (Interdomain).

In chapters 18-19, Case Study: Kerberos, a case study in Client-Server needs, is presented to show client-server interactions. We limit our discussion to the Kerberos authentication protocols and omit various administrative issues because Kerberos is not by itself a complete authentication framework required for secure distributed computing in general.

In chapters 20-21, we presented the advantages and summary of our design to show that with mutual authentication protocol using Smart card, fraud against plastic cards will almost impossible, because Smart cards are not susceptible to the frauds perpetrated against conventional plastic cards.

Part IV: In chapter 22, Conclusion and Open Areas fo~ Future Research, the dissertation is concluded with a brief discussion of some open areas for future research, such as using this design as benchmark for further research relating to the new CUNY Smart card project.

1.1 Computer Security

The security of computer systems is a serious issue due to the expanding role of computation, distributed databases, and telecommunication applications such as electronic mail and electronic fund transfer. While the trend of technological progress antedates the 1980s, and will certainly go on, this period will also be remembered for a qualitative evolution. The continuing proliferation in information technology has enabled vast amounts of information to be collected, processed, and utilized for various purposes. A large number of individuals have personal computers, many of which can communicate through modems over ordinary telephone lines. Most businesses, small or large, depend on computers for financial record keeping functions. The widespread introduction of information technology into business inevitably leads to its misuse for crime. Furthermore, individual privacy has been eroded as tremendous amounts of personal data are also more easily subjected to usage fraud because wireless networks can be accessed without a physical "tap." The need to control access in computers first became serious when shared system, such as time-sharing began to operate. The need is more severe for systems that can be accessed over a public telephone or data network. In the absence of counter measures, the user enjoys less privacy, as is demonstrated by the fact that eavesdropping prevails on wireless channels on a scale never heard of in the context of wire-line communications. The National Research Council Reports:

"Security is a concern of organizations with assets that are controlled by computer systems. By accessing or altering data, an attacker can steal tangible assets or lead an organization to take actions it would not otherwise take. By merely examining data, an attacker can gain a competitive advantage, without the owner of the data being any the wiser." [133].

The General Accounting Office further states:

"With the end of the cold war and the resulting shift in focus from military to economic power, there is evidence that economic espionage has become a growing problem for U.S. companies at home and abroad." [38].

In 1983, the film "War Games" introduced people to a new trend in computer security, the computer hackers who spend their time persistently trying to break into computer systems. In the same year, juveniles from Wisconsin, called the "414s", broke into many computing systems, including Sloan-Kettering Hospital and Security Pacific Bank. In 1987, a group of West German youths gained an authorized access to a NATO computing System with links to U.S. computers. Attacks such as these are not uncommon. The attacker's time is apparently unlimited and uncost; however, their devotion to a basically tedious pastime is extraordinary. Bruce Sterling [173] expressed it very well:

"The term "hacking" is used routinely today by almost all law enforcement officials with any professional interest in computer fraud and abuse. American police describe almost any crime committed, with, by, through, or against a computer as hacking. Most important, "hacker" is what computer intruders choose to call themselves. Nobody who hacks into systems willingly describes himself (or herself) as a "Computer Intruder," "Computer Trespasser," "Cracker," "Wormer," "Dark Side Hacker," or "High-tech Street Gangster." Several other demeaning terms have been invented in the hope that the press and public will leave the original sense of the word alone. But few people actually use these terms."(pp. 55-56).

Attacks by modifying software have increased-the 'Trojan Horse' attack. Software can be destroyed maliciously, modified, deleted or misplaced. The result is the same regardless of the motive. The types of attacks on the security of a computer system are best characterized by viewing the function of the computer system as providing information. Simmons [166] succinctly states that:

".... information security is about how to prevent cheating, or failing that, to detect cheating in information-based systems wherein the information itself has no meaningful physical existence."

The nature of the attack that concerns an organization varies from one set of conditions to another. A special feature of illegal attacks on computer systems is

that there is no tradition of associated guilt-feelings. The law, in most countries, has not begun to define the new kinds of crime. With no likelihood of punishment, the probing of the defenses of computer systems is considered to be a game. It has become the tool of organized crime. Protecting software against corruption must be a concern of all those who need to make their computer systems secure. Wood [185] states that:

"What is missing in many instances if the involvement of concerned users, enlightened developers, experienced EDP auditors, experienced information security specialists, or other parties who understand how to incorporate controls into systems while the systems are still in development." (p.13-24)

User authentication is a serious issue which becomes even more serious when unacquainted users seek to share facilities by means of computer networks. The traditional authentication device is the password. A plain text password file presents a serious vulnerability for a computing system. These files are usually either heavily protected or encrypted. The more serious problem, however, is establishing administrative procedures that make users' passwords adequately secure. Additional protocols are needed to perform mutual authentication in an atmosphere of distrust. Improving the security provided by secure protocols against attacks is a continuous problem for computer scientists as well as society at large. Designing cryptographic requirements for secure protocols against various classes

of attacks that ensure confidentiality in international data traffic becomes impractical for several reasons.

First, the security of a communications system lies in its ability to meet specific operating requirements despite the actions of a knowledgeable and determined attacker. Until recently, research efforts on secure protocols concentrated on the security of the underlying cryptographic transformations, but new protocols have arisen whose security properties are not readily apparent. In these protocols, it is not only the security of the cryptosystem that plays a role; we might assume that the perfect concealment of messages is axiomatic, but the logic of the protocols or even the implementation of the cryptosystem may be flawed. Such flaws, when undetected, are as damaging to the overall security as a compromised cryptosystem.

Second, there are significant problems in protocol design. Historically, cryptography is a fundamental tool of computer security and is a widely used communication channels encryption and for local file encryption. Although advances in cryptography have led to cryptosystems whose security implementation requires the design of complex protocols, applications have become more demanding. Finally, there is a basic problem of proving the correctness of cryptographic exchanges. The automated control of complex systems that require sophisticated authorization protocols, the trend toward maintaining distributed

databases of sensitive information, and the availability to the general public of powerful computer and communications networks all point toward significant problems in protocol design. Cheswick [37] states that:

".... In the cryptographic world, finding holes in protocols is a popular game. Sometimes, the creators make mistakes, plain and simple. More often, the holes arise because of different assumptions. Providing correctness of cryptographic exchanges is a difficult business and is the subject of much active research. For now, the holes remain, both in academe and in the real world as well." (pp. 164).

Furthermore, in an OSI-based network, encryption can be done in any of the seven layers. The higher the layer at which encryption is performed, the greater security it provides to the user. However, data link encryption can mask traffic characteristics, and that by itself may be of interest to an unauthorized party. Pfleeger [144] succinctly states that:

"... In summary, then, the security of the encryption system should be appropriate to the degree of confidentiality of the data being preserved. Users of encryption systems should consider the value of the data being encrypted when selecting an encryption system." (pp. 153).

When designing a secure protocol for a computer network, several sources of data insecurity need to be considered. Prominent among these are spurious message injection, message reception by unauthorized receivers, transmission disruption, and re-routing data to fake nodes. To maintain security against these hazards, a combination of encryption algorithms on the data and appropriate protocols for message exchanges is utilized. These technique also facilitate the handling of other problems in computer communication networks, such as key distribution, authentication, privacy, digital signatures, network mail, and transaction verification.

Most security vulnerabilities have been exploited by people who broke encryptions, tapped lines, circumvented access control, and wrote malicious programs. It should be emphasized that although people may be the biggest source of security problems for computing systems, people are not the only source. While some of these vulnerabilities can be controlled with procedures or system design, others are inherent weaknesses in certain kinds of computer systems or applications. For example, each communications link in a computer network is assumed to leak data. Satellite and terrestrial microwave leak more than the others; while optical fiber does not leak data, the non-fiber junctions are repeaters and can be tapped. Communications links within a building can also be controlled, but once communications are routed to a common carrier, there is no longer any control of medium, routing multiplexing, or leakage. The lack of such control can be viewed

from two perspectives. The positive side is that the unpredictability is equal for the potential intruder. On the other hand, it is almost impossible to limit the exposure.

Most professional organizations have ethical codes. Various nations and industries have codes of fair information practice. Teaching and reinforcement of computer-related values are very important, alerting system purveyors, users, and would-be misuser to community standards, and providing guidelines for handling abusers. Generally, access to confidential information should be limited to only those who "need to know." Dorothy Denning [50] states that:

"In military systems, access controls enforce both a non-discretionary policy of information flow based on military classification scheme (the multilevel security policy), and a discretionary policy of access control based on "need -to-know" (that is, on the principle of least privilege)." (pp. 286-287).

In every computer system, there are privileged individuals who in some sense have to be more trustworthy than others, for example, system programmers, database administrators, and operators. Most system designs do not permit or encourage carefully compartmentalized privileges, and instead provoke the use of omnipotent mechanisms (for example, superusers). Such mechanisms are potent, even if used with good intent, particularly if they are abused by privileged users or if they can be subverted by someone masquerading as a privileged user.

When solving security problems, it is important not to do so at the cost of increased vulnerability on the network side. For example, a solution that requires adding an on-line database of secret information (e.g. Secret Keys), which must be heavily protected against computer system intrusion, is inferior in that respect in comparison to a solution that only requires adding a key revocation data base (which does not contain any secrets and therefore need not be protected as heavily).

In general, providing increasing levels of security carries increasing financial cost. Higher levels of security clearance cost greater amounts of money. Businesses in general must be prepared to spend greater amounts of money to provide higher levels of security or run the risks of neglect. While concerns for white-collar crime remains controversial, there is no doubt that financial institutions which have not taken adequate precautions do suffer from major frauds.

As already indicated, this paper presents a new approach to cryptographic requirements of secure protocols against various classes of attacks.

2.1 Computer Security: An Overview

Computer security includes properties such as 1) confidentiality, 2) integrity, 3) availability, 4) prevention of denial of service, and 5) prevention of generalized misuse. Security is a major concern in most distributed systems. While the system administrator may have definite ideas about who can use which resource (for example, no student should have an e-mail address in the Faculty Computing Lab), the physical environment needs to be adequately protected from intrusion also; additionally, many users may want their files protected from any form of intrusion.

In a school, making information in a computer system accessible for reading by authorized parties only is an issue of confidentiality. This type of access includes displaying, printing, and other forms of disclosure including revealing the existence of such information. Preventing school employees from an unauthorized modification of student academic records, whether intentionally or otherwise, is an issue of integrity. Such modifications include writing, deleting, changing and creating. Ensuring that computer system assets be available to authorized parties is an issue of availability.

Intentional disruptions of a network link so as to make the network or a portion of the network unusable, is a denial of service. This can be done in many ways, such as by an act of sabotage, introduction of a software virus to the network,

or by improper actions by inside authorized personnel. Generalized misuse may be caused by individuals or teams, acting in a variety of roles such as requirements' specifiers, designers, implementers, operators, users, and other external groups. These security issues are not mutually exclusive. For example, if a hacker penetrates a school's information system but he/she is not able to read the data or effect changes due to the encryption of the data, the hacker is likely to cause enough damage to the system so as to make it crash and hence become unavailable.

"Security" in a computer network is a measure of confidence that the integrity of a system and its data will be protected. Security requires not only adequate system protection, but also consideration of the external environment within which the system operates. For example, if an operator's workstation is exposed to unauthorized personnel, or if stored files can be removed from the computer system and processed at a remote site on a system with no protection, then, internal protections are not useful. These security problems are essentially management problems.

It is reasonable to say that a system is secure if its resources are accessed and utilized as intended in every situation. Security violations (misuse) of the system can be categorized as being either accidental or intentional (malicious). It is easier to protect against accidental misuse than to protect against malicious misuse. Example of malicious access includes unauthorized modification of data, unauthorized theft of information and unauthorized destruction of data. Absolute protection of the

system from malicious abuse is not possible, but the cost to the perpetrator can be made sufficiently high to deter most if not all attempts to access, without proper authority, the information residing in the system.

2.2 Security Measures

To protect the system, we must take security measures at two levels:

- Perimeter (Physical) security- that prevents those on the "outside" from getting "inside" a system. A system, and the site or sites containing the computer system must be physically secured against armed or surreptitious entry by intruders.

- Internal (Human) Security- that keeps users inside from interfering with one another or otherwise violating the security policy of the system. Users must be authorized carefully to reduce the chance of any such user giving access to an intruder in exchange for a bribe or other favors.

Security at both levels must be maintained to ensure operating system security. A weakness at a high level of security (perimeter or internal) allows circumvention of strict low-level (operating system) security measures. For example, certain security measures may be desirable to hinder malicious penetrators, but do relatively little to reduce hardware and software faults. Certain

reliability measures may be desirable to provide hardware fault tolerance, but do not increase security. On the other hand, properly chosen system architectural approaches and good software engineering practice can enhance both security and reliability. Thus, it is highly advantageous to consider both reliability and security within a common framework, along with other properties such as application survivability and application safety.

Designing systems and networks that are secure and reliable is a difficult task, and requires an overall system perspective. It is worthwhile, in many applications, to devote a considerable effort to the security of the computer system. Security problems increase in scope and magnitude as computer systems and networks are increasingly linked together and as communications become a fundamental part of our lives, interconnecting telephones, fax machines, interactive video systems, and databases, without adequate security. Many causes of security breach stem from inadequate system security, the absence of differentiated user privileges (all-or-nothing is the general default), and inherent ambiguities.

The most insidious cases involve misuse or inadvertent use by individuals whether or not those people have authorized system access. There are various potential motivations for abuse of rights, including espionage, revenge, curiosity, and more importantly financial gain. For example, large commercial systems containing credit card account numbers, payroll, or other financial data are inviting targets to thieves. Systems containing data pertaining to corporate operations may

be of interest to unscrupulous competitors. Furthermore, loss of such data, whether via accident or fraud, can seriously impair the ability of the corporation to function. Various classes of misuse are summarized in Figure 2.1 and Table 2.1 [135].

MODE	TYPE OF MISUSE
EX: External misuse	Computer-system access
HW: Hardware misuse	Computer-system use
MQ: Masquerading	Apparently authorized use (even if clandestine)
PP: Pest program for deferred misuse	Direct use
BY: Bypass of intended controls	Use apparently conforming with intended controls
AM: Active misuse of resources	Active use
PM: Passive misuse of resources	Apparently normal use
IM: Misuse resulting from inaction	Apparently proper use
IN: Use as an aid to other misuse	Proper use

Figure 2.1 Classes of techniques for computer misuse

(Adapted from P.G. Neuman and D.B. Parker, "A summary of computer misuse techniques." Twelve National Computer Security Conference, Baltimore, 1989.)

The order of categorization depicted is roughly from the physical world to the hardware to the software, and from unauthorized use to misuse of authority. The physical spans of networks create a number of unique problems. The larger area requires more extensive physical protection, provides more availability for interceptors or jammers, results in more places from which TEMPEST emanation may occur, and assures more potential diversity of user backgrounds and their controlling security policies. Dial-up networks introduce the potential for access to be attempted from most any of the world.

MODE	MISUSE TYPE
External (EX)	
1. Visual spying	Observing of keystrokes or screens
2. Misrepresentation	Deceiving operators and users
3. Physical scavenging	Dumpster-diving for printout
Hardware misuse (HW)	
4. Logical scavenging	Examining discarded/stolen media
5. Eavesdropping	Intercepting electronic or other data
6. Interference	Jamming, electronic or otherwise
7. Physical attack	Damaging or modifying equipment, power
8. Physical removal	Removing equipment and storage media
Masquerading (MQ)	
9. Impersonation	Using false identities external to computer systems
10. Piggybacking attacks	Usurping communication lines,

	workstations
11. Spoofing attacks	Using playback, creating bogus nodes and systems
12. Network weaving	Masking physical whereabouts or routing
Pest Programs (PP)	Setting up opportunities for further misuse
13. Trojan horse attacks	Implanting malicious code, sending letter bombs
14. Logic bombs	Setting time or event bombs (a form of Trojan horse)
15. Malevolent worms	Acquiring distributed resources
16. Virus attacks	Attaching to programs and replicating
Bypasses (BY)	Avoiding authentication and authority
17. Trapdoor attacks	Utilizing existing flaws
18. Authorization attacks	Password cracking, hacking tokens
Active misuse (AM)	Writing, using, with apparent Authorization
19. Basic active misuse	Creating, modifying, using, denying service, entering false or misleading data
20. Incremental attacks	Using salami attacks
21. Denials of service	Perpetrating saturation attacks
Passive misuse (PM)	Reading, with apparent authorization
22. Browsing	Making random or selective searches
23. Inference, aggregation	Exploiting database inferences and traffic analysis
24. Covert channels	Exploiting covert channels or other data leakage

Inactive misuse (IM/25)	Willfully failing to perform expected duties, or committing Errors of omission
Indirect misuse (IN/26)	Preparing for subsequent misuses, as in off-line Preencryptive matching, factoring large numbers to Obtain private keys, autodialer scanning

TABLE 2.1 Types of Computer Misuse

(Adapted from P.G. Neumann and D.B. Parker, "A summary of computer misuse techniques" Twelfth National Computer Security Conference, Baltimore, 1989.)

Networks transversing international boundaries usually accrue problems in coordinating laws, policies, and protocols. Malicious misuse of computer systems can never be presented completely, particularly when perpetrated by authorized users. Ultimately, the burden must rest on more secure computer systems and networks, on more enlightened management and responsible employees and users, and sensible social policies. Monitoring and user accountabilities are also important.

Additionally, expensive measures such as hiring computer security consultants to oversee not only the privileged user but all users on the network can be implemented. Numerous surveys show the surprising result that insiders may provide the most common threat. The well-known security consultant Robert Courtney estimates that 90% of computer crime is instigated by insiders [21].

Insiders can include computer personnel (operators, system analysts, programmers, users), security personnel (security designers and security implementers), management (not only highly trusted, but highly likely to go unchallenged if deviating from expected procedures), and other employees (workers located in proximity to computing operation or source of information, maintenance and cleaning personnel). Telling examples of the potential power for misdeeds available to insiders are the 1987 Ivan Boesky insider trading scandal and the 1987 $259 million insider frauds at Volkswagen AG in West Germany [83].

Other examples more directly related to computers [22] are: In June of 1988, two small computers were stolen on two consecutive nights from the Strategic Defense Initiative Organization (SDIO) in the Pentagon. The area struck was under stringent physical security protection, including a guard, a security card-locked door system, and a videotape surveillance. Investigators believe that the security bypasses accomplished could have been done only by an insider. In 1981, a major state lottery was victimized by insiders with information on unclaimed winning tickets. The printed duplicate tickets counterfeited the proper information in order to claim winning shares. Other methods for dealing with threats from human agents can be classified into two categories: access control and cryptography.

2.3 Access control

Controlling access has continued to be a challenge. Exposure to threats worsened because of remote users and widely dispersed communications media.

This was balanced by security methods such as computer access passwords, database security features, and security software (which included auditing software and specific access control features in addition to passwords). Access control determines how authorizations are handled. It is implemented for a specific user and a specific object. An access control policy determines whether a specific user should have access to a specific object.

2.4 Cryptography

Cryptography is the practice of using encryption to conceal text. It is the science of encoding and decoding messages with the purpose of preventing the interceptor from gaining knowledge of the text. Both encryption and decryption are controlled by a cryptographic key or keys. David Kahn's book "The Code breaker" [87] presents a comprehensive treatment of the history of cryptography.

Access controls and cryptography deal with different aspects of security issues. For example, while cryptography protects against browsing by making the information unintelligible, access controls are needed for protecting data from disclosure while it is being processed in the clear. Cryptography and access controls must be supplemented with information flow controls to control information dissemination. Current security implementations employs a combination of methods from each category.

For all practical purposes, all multi-user computer network systems use encryption to protect password information. In the Athena System, authentication is the task of Kerberos, a set of distributed softwares that employs a series of encrypted exchanges of information to allow a user access to servers. Kerberos protocols rely on encryption keys, known only to the appropriate parties in a transaction, to protect information sent across an open network. The basic Kerberos protocol is secure. In operation, it provides authentication to users on the network by having trusted servers issue valid "tickets" that enable users to obtain services from the network. The Athena System presents a sophisticated example of a distributed computing network system that utilizes cryptography to complement access controls procedures.

2.5 Security Threats - Cost/Benefits Analysis

The question one must answer before deploying a security mechanism is: how much security can one afford? The cost/benefit analysis applies when analyzing computer security threats and the benefits of countermeasures against these threats. Part of the cost of security are direct financial expenditures, such as the extra routers and computers to build a defense mechanism. Often administrative costs of setting up and running the defense mechanism are overlooked. But, there is a more subtle cost, a cost in convenience and productivity, and even morale.

Too much security can hurt as assuredly as too little. Finding the proper balance is tricky, but utterly necessary. It can only be done if one has properly assessed the risk to the organization from either extreme. The idea behind this assessment is that a comparison of the loss expectancy with the cost of providing protection will be a guide for the managers in deciding what security measures to install. The primary problem with conducting a cost benefit analysis is the lack of certainty associated with assessing each type of threat. For example, the data that enter into a risk assessment are known only with very low accuracy, rarely better than an order of magnitude and sometimes with even less precision.

Consequently, risk analysis can hardly be considered a scientifically based procedure. However, management needs some guidance and the many risk-assessment methodologies on the market offer them a degree of help. All successful attacks are not publicized in the news media because organizations have a vested interest in not divulging security weaknesses on their systems. Worse yet, in the event of crime, some companies will not investigate or prosecute, for fear that it will damage their public image. For example, an investor would not feel safe investing money in a company that has just suffered a loss of millions of dollars through computer security weaknesses. It is a common belief that publicizing computer security breaches makes the system more vulnerable to further attacks. While the extent of white-collar crime remains controversial, there is no doubt that organizations and financial institutions that have not taken adequate precautions do suffer from major frauds.

2.6 Authentication

A major security threat for operating systems is the authentication problem. Generally, authentication is based on some combination of three sets of items: user possession (a key or card), user knowledge (user identifier and password), and a user attribute (signature, fingerprint, or retina pattern). The most common approach to authenticating a user identity is the use of user passwords.

Passwords are often used to protect objects in the computer system, in the absence of more complete protection schemes. Different passwords may be associated with different access rights. For example, different passwords may be used for reading, appending, and updating a file. Although there are some problems associated with the use of passwords, they are nevertheless extremely common, because they are easy to use and understand. The problem with passwords is related to the difficulty of keeping passwords secret. Passwords can be compromised by being guessed, accidentally exposed, or illegally transferred from an authorized user to an unauthorized one.

The failure of password security due to exposure can result from visual or electronic monitoring. Exposure is a particularly severe problem if the password is written down where it can be read or lost. Furthermore, password security can be compromised due to human nature. For example, if one user-ID was shared by several users, and a scrutiny breach occurred from that user-ID, then it is

impossible to know who was using that user-ID at the time, or even if it was one of the authorized users.

2.7 Software Threats

In an environment where a program written by one user may be used by another user, there is an opportunity for misuse. The two common methods used to facilitate the misuses are: trap door and Trojan horse.

- Trap door: The designer of a program or system might leave a "hole" in the software that only he is capable of using. For example, the code might check for a specific user identifier or password and circumvent normal security procedures. This type of security breach was demonstrated in the movie "War Games".

A clever trap door could be included in a compiler. The compiler could generate a standard object code as well as a trap door, regardless of the source code being compiled. The activity is villainous because a search of the source code of the program would not reveal any problems. Only the source code of the compiler would contain the information. Trap doors pose a serious problem because, to detect them, all the source codes for all components of the system would have to be analyzed. Since software systems may consist of millions of lines of code, this analysis is not done frequently.

- **Trojan horse: Many systems have provisions that allow programs written by one user to be executed by other users. If these programs are executed in a domain that provides the access rights of the executing user, they may misuse these rights. Inside a text-editor program, for example, there may be a code to search the file to be edited for certain keywords. If any are found, the entire file may be copied to a special area accessible to the creator of the text editor. A code segment that misuses its environment is called a Trojan horse. The Trojan-horse problem is further heightened by long search paths (such as are common on UNIX systems). All the directories in the search path must be secure or a Trojan horse could be slipped into the user's path and executed accidentally.**

2.8 System Threats

Most operating systems provide a means for processes to spawn other processes. In such an environment, it is possible to create a situation where operating system resources and user files are misused. The two most common methods for doing this are worms and viruses:

- **Worms: A worm is a process that uses the spawn mechanism to defeat system performance. The worm spawns copies of itself, using up system resources and locking out system use by all other processes. On computer networks, worms are very potent, since they may reproduce themselves among systems and thus**

shut down the entire network. Such an event occurred in 1988 to UNIX systems on the worldwide Internet network, causing millions of dollars of lost system and programmer time.

Although the self-replicating program was designed for rapid reproduction and distribution, features of the UNIX networking environment provided the means to propagate the worm throughout the system. The worm program exploited flaws in the UNIX operating system's security routines and took advantage of UNIX utilities that simplify resource sharing in local area networks to gain unauthorized access to thousands of other connected sites. The very features of the UNIX network environment that assisted the worm's propagation also helped to stop its advance.

- Viruses: Another form of computer attack is a virus. Like worms, viruses are designed to spread into other programs and can wreak havoc in a system, including modifying or destroying files, and causing system crashes and program malfunctions. Whereas a worm is structured as a complete, standalone program, a virus is a fragment of code embedded in a legitimate program. Viruses are a major problem for computer users, especially users of microcomputer systems. Multi-user computers, generally, are not prone to viruses because the executable programs are protected from writing by the operating system. Even if a virus does infect a program, its powers are limited because other aspects of the system are protected. Single-user systems have no

such protections and, as a result, a virus has free run. Viruses are usually spread by users downloading viral programs from public bulletin boards or exchanging floppy disks containing an infection. Although viruses are not designed to destroy data, they could spread to application files and cause such problems as long delays and program malfunctions.

Occasionally, upcoming viral infections are announced in high-profile media events. Such was the case with the Michelangelo virus that was scheduled to erase infected hard disk files on March 6, 1992, the Renaissance artist's five hundred seventeenth birthday. Because of the extensive publicity surrounding the virus, most U.S. sites had located and destroyed the virus before it was activated, so it caused little or no damage. Anti-virus programs are currently very good sellers.

The best protection against computer viruses is prevention, or the practice of safe computing. Purchasing unopened software from vendors and avoiding free or pirated copies from public sources or floppy-disk exchange is the safe route to preventing infection. However, even new copies of legitimate software applications are not immune to virus infection. Because they usually work among systems, worms and viruses are generally considered to pose security, rather than protection problems.

2.9 Threat Monitoring

The security of a system can be improved by two management techniques. One is *threat monitoring*. The system can check for suspicious patterns of activity in an attempt to detect a security violation. A common example of this scheme is a time-sharing system that counts the number of incorrect passwords given when a user is trying to log in. More than a few incorrect attempts may signal an attempt to guess a password.

Another common technique is an *audit log*. An audit log records the time, user, and type of all accesses to an object. After security has been violated, the audit log can be used to determine how and when the problem occurred and perhaps the amount of damage done. This information can be useful, both for recovery from the violation and, possibly, in the development of better security measures to prevent future problems. Unfortunately, logs can become large, and logging uses system resources that are then unavailable to the users.

Rather than log system activities, we can scan the system periodically for security holes. These scans can be done when the computer is relatively unused, and therefore, have less effect than logging. Such a scan can check a variety of aspects of the system:

- **Short or easy-to-guess passwords.**
- **Unauthorized set-id programs, if the system supports this mechanism.**
- **Unauthorized programs in systems directories.**
- **Unexpected long-running processes.**
- **Improve directory protection, for both the user and system directories.**
- **Improve protection on system data files, such as the password file device drives,**
- **or even the operating-system kernel itself.**
- **Dangerous entries in the program search path (for example, a Trojan horse).**
- **Changes to system programs; unexpected changes can be detected by keeping a list of the checksum values of all system programs, and comparing this list against the current checksum values of the programs-checksums do not change unless the contents of the file have changed.**

Any problems found by a security scan can either be fixed automatically or be reported to the managers of the system. Networked computers are more prone to security attacks than are stand-alone systems. Rather than attacks from a known set of access points, such as directly connected terminals, attacks come from an unknown and very large set of access points: a potentially severe security problem. Furthermore, systems connected to telephone lines via modems are also more exposed to attacks.

2.10 Access Controls

Access controls assure that all direct accesses to objects are authorized. Regulating the reading, changing and deleting of programs and data files, access controls offer protection against accidental and malicious threats to secrecy, authenticity, and system availability. The effectiveness of access controls rests on two premises. The first is the proper user identification - that is, no one should be able to acquire the access rights of another. This is accomplished through authentication procedures at login. The second premise is that information specifying the access rights of each user or program is protected from unauthorized modification. This is accomplished by controlling access to system objects as well as to user objects. Many access controls incorporate a concept of ownership; that is, users may dispense and revoke privileges or objects they own. This is common in file systems intended for the long-term storage of user data sets and programs. Not all applications include this concept; for example, students do not own their records in an academic information system.

In studying access controls, it is useful to separate policy and mechanism. An access control policy specifies the authorized accesses of a system, whereas an access control mechanism implements the policy. This separation is useful for three reasons:

- **It facilitates discussion of the access requirements of systems independent of how these requirements may be implemented.**
- **It allows comparisons of different access control policies as well as different mechanisms that enforce the same policies.**
- **It allows the design of mechanisms capable of enforcing a wide range of policies. Furthermore, these mechanisms can be integrated into the hardware features of the system without infringing on the flexibility of the system to adapt to different policies.**

Access control policies are classified into two categories: mandatory and discretionary. Discretionary access controls (DACs) give a user the ability to specify the types of access to individual resources that are considered to be authorized. Examples of discretionary controls are the user/group/world permissions of UNIX systems and the access-control lists of Multics. These controls are intended to limit access, but are generally incapable of enforcing copy protection. Attempts to constrain propagation through discretionary copy controls tend to be ineffective. The "Department of Defense Trusted Computer System Evaluation Criteria", otherwise known as the "Orange Book" [55], attempts to define security requirements for discretionary access controls, and specifies that access controls must be capable of providing or prohibiting access on protected objects down to the granularity of individual users.

A further requirement specified for discretionary access controls in the Orange Book is that authenticated data must be protected so that it cannot be accessed by an unauthorized user, for example, password data. In practice, almost all password data are encrypted using a one-way transformation. If done properly, knowledge of the transformation is of no value to an adversary, because the actual password is necessary for access.

Mandatory access controls (MACs) are typically established by system administrators, and cannot be altered by users; they can help to restrict the propagation of copies. The best-known example is multilevel security, which is intended to prohibit the flow of information in the direction of decreasing information sensitivity. Flawed operating systems may permit violations of the intended policy. Unreliable hardware may cause a system to transform itself into one with unforeseen and unacceptable modes of behavior. This gap between access that is intended and access that is actually authorized represents a serious opportunity for misuse by authorized users. Ideally, if there were no such gaps, misuse by authorized users would be a less severe problem.

In military systems, access controls enforce both a non-discretionary policy of information flow based on the military classification scheme (the multilevel security policy), and a discretionary policy of access control based on "need-to-know" (that is, on the principle of least privilege). For example, a process running with a secret clearance is permitted to read only from unclassified, confidential, and

secret objects and to write only to secret and Top Secret objects (although integrity constraints may prevent it from writing into Top Secret objects).

Several Kernel-based systems that have been developed or designed - including the MITRE security kernel for the DEC PDP-11/45 [158],[116]; MULTICS with AIM [161]; the MULTICS-based system designed at Case Western Reserve [177]; the UCLA Data Secure UNIX system 9DSU) for the PDP-11/45 and PDP-11/70 [148]; the UNIX-based Kernelized Secure Operating System (KSOS) developed at Ford Aerospace for the PDP-11/70 (KSOS-11),([108], [18]), and at Honeywell for a Honeywell level 6 machine (KSOS-6 or SCOMP) [31]; and Kernelized VM/370 (KVM/370) developed at the System Development Corporation [76] - use access to enforce multilevel security for user and untrusted system processes.

With the exception of UCLA Secure UNIX, these systems were all developed to support the Department of Defense multilevel security policy which states that classified information must not be accessible to subjects with a lower security clearance. For example, a user having a Secret clearance must not be able to read from Top Secret files (e.g., "read up") or write Secret information in Confidential or Unclassified files (i.e., "write down"). One of the first systems to use access controls to enforce multilevel security was the ADEPT-50 time-sharing system developed at Systems Development Corporation (SDC) in the late 1960's [178].

2.11 Auditing Function and Tools

The verification of system security features and system security performance is a difficult task requiring constant surveillance of the system. Distributed control among heterogeneous systems or among homogeneous systems with different administrative organizations can complicate the problem of audit-trail collection (that is, monitoring) and audit-trail analysis (that is, auditing). In the absence of a suitably global perspective, disruptive activity may go undetected far longer than it would with centralized auditing, simply because events across different system components are difficult to correlate. For example, at the University of Texas, an employee used a Dean's password to divert $16,200 over a 1-year period. He awarded fellowship stipends to students who were not eligible. The diversion was detected when a student wrote to the Dean of Recruiting to thank the Dean for his generosity [2].

Another case [184], demonstrates the importance of carefully selecting data for audit. A supervisor of a major city income tax refund section noticed that refund payments were assigned a reference number according to the case, with no direct tie to a person or business. The supervisor selected reference numbers, increased the check amount, and changed the name and address for payment to an accomplice. After the check was issued, he would change the data back. There were no authorization barriers except for the authority of the supervisor. The team received more than $120,000 before being apprehended.

During system development, auditors serve as advisors as to the effectiveness of internal controls, such as data logging, value totaling, and access restrictions. Auditors generally look for: the adequacy of internal control, sufficient feedback procedures (measures for obtaining meaningful data), and features that make data auditable with efficient use of resources.

Auditors assist in the test process by helping check on the adequacy of controls. Operational audits help assure compliance with standards on performance, media labeling, handling, and storage. Usually audits occur at the input, output, and internal level. Careful cooperation is required to ensure that real time or retrospective analysis can identify the people, processes, and systems involved, without violating confidentiality and privacy requirements.

Auditing functions are generally performed at regular intervals or phase points during system development and at frequent but random times during operation. The "Orange Book" mandated that each time an auditable event occurs, the system should write the following information to the audit trail:

- **Date and time of the event.**
- **Origin of the request (e.g., terminal/workstation ID).**
- **Unique ID of the user who initiated the event.**
- **Type of event.**
- **Name of object involved (e.g., file being deleted).**

- **Success or failure.**
- **Description of modifications to security databases.**
- **Subject or object security levels.**

The magnitude of the auditing function generally must be balanced between the degree of threat and the availability of auditing resources. Thus, while awareness of sophisticated threat potential is essential, the concentration is on routine control. For example, depending on how much support the audit function receives, it may not be productive for auditors to search for Trojan horses (Ths) and covert channels. For this reason, the audit approaches used tend to be oriented toward the most common problems. Some of the tools and techniques are:

- **User command log - This control is designed to detect unauthorized actions and monitor command activity by users.**
- **Sensitive file access log - This tool detects unauthorized accesses to sensitive files and to monitor file access activity.**
- **Crash log - This log is used to detect a "denial of service" attack.**
- **Program change log - This log facilitates detection of unauthorized modification of programs.**

- **Data adjustment/correcting log - This is to detect unauthorized modification of data.**

- **Extended records information - This capability in an application program appends audit trial information to all transaction records. The purpose is to detect unauthorized transactions.**

- **Parallel simulation - Processing production transactions with programs that simulate critical situations.**

Part II Cryptography: General Concepts

3.1 Introduction

The security of a system depends on identifying correctly the identity of each user. The ability of the system to identify users when they log in is called user authentication. A common example could be found in a bank's "Automatic Teller machine" (ATM) which, based on "blind trust", dispenses cash to account holders that it can identify while interactive computing; that is, man-machine interaction, is good for efficiency, it also opens possibilities for fraud by a masquerade; that is, the tendency to adopt false identity. Although we envisage that access to computers and in some cases access to secure buildings should be controlled in conformity with the identity of the person, but like many human skills taken for granted (such as language and vision) recognizing a person is surprisingly difficult for a computer. In most cases, practical systems do not identify the individual rather; they verify the individual's identity.

Means of personal identity verification can be classified into four general types. Each of the four types make use of (a) something known by the individual, (b) something acquired by the individual, (c) a physical attribute of the individual, and (d) the effect brought about by an unconscious action of the individual.

Categories (a) and (b) can be analogous to a password and a passport as examples. Categories (c) and (d) are not always distinguished. For instance, fingerprint is an example of physical attribute and a signature is the result of the effect brought about by an unconscious action, because each movement is not individually controlled.

In order to be meaningful and be acceptable to the system users, any successful identification system must, in addition, be economically viable and adequately secure. Any unmanageable procedure is likely to aggravate users into finding means to avoid the procedures, thereby making the effort of any potential intruder much easier.

3.2 Methods of Personal Identity Verification

3.2.1 Password

Passwords are one of the best known examples of identity verification by something known. Passwords are assumed to be known only to the user and the system. Passwords have been used in a wide variety of identity verification schemes ranging from protocols for accessing computer systems to military applications. In some cases a user chooses his or her own passwords, while in other cases they are assigned by the system. Methods of generating and managing passwords have been discussed in various publications. A standard for PIN management was published by ANSI in 1982 [143].

Passwords can be classified into four main groups:

(1) Passwords which are unique to individual users,

(2) Passwords which are not unique to individual users, but serve to confirm a claimed identity,

(3) Group passwords which are common to all users on a system, and

(4) Passwords which change every time a system is accessed.

Which type is used form these group is dependent on the environment of the application.

3.2.2. Unique passwords

Passwords which are unique to individual users provide a much higher level of security. For example, if there is unauthorized access, the system log should be able to indicate under what password the access took place. This kind of control can also detect the use of a loaned password. On the other hand, improper handling or storing of a password may lead to an unintentional disclosure. It is often suggested that users should memorize their unique passwords, rather than keeping a printed copy. In cases where unique passwords are implemented, a separate user identifier need not be used. However, when required, an added confirmation of identity can be generated.

3.2.3 Non-unique passwords

Passwords which are not unique to individual users, but which serves to confirm a claimed identity are used in an environment with a large user population; For example, the users of a bank's automatic teller machines. In this example, the use of unique passwords is impracticable because such a password would be too long to memorize. In this instance, a short password is allocated to each user. Identification of each user depends on a much longer password (number) which the user is not required to memorize.

3.2.4 Shared passwords

Assigning the same passwords to a number of different users should present no difficulty if the password assignment conveys no relationship to the user identity. For example, by using non-unique passwords in an on-line system, a central database can be created to give users identities and their corresponding passwords; to verify an identity, presented identity and password is compared with the information stored in the central database. On the other hand, in an off-line system, encipherment is employed; that is, the presented password must be connected to the identity token, e.g. the magnetic striped card, such that while the off-line terminal can interpret the enciphered information, the cardholder or an intruder cannot.

3.2.5 One-time passwords

Another type of password is what can be construed as "one-time stamp pad". These are passwords which change every time a system is accessed. The password used in this case changes every time the system is accessed. Every time the system is accessed, a different password is used. This is necessary to prevent an intruder from staging a playback attack. While this system is more secure than those, which uses the same password(s) continuously, it has disadvantages too. For example, the central database need to be properly secured at all times. The amount of password materials depends on the frequency of access. One-time passwords of this type are in use in the Society for Worldwide Interbank Financial Telecommunications (S.W.I.F.T.) network of the international banking community. For greater security the S.W.I.F.T. password tabulations are prepared in two halves, so that each list contains only half passwords. The two halves are sent separately to users, thereby reducing the possibility of complete passwords being intercepted.

3.2.6 Password protection

Password handling capability by the system should be designed to protect against any form of disclosure. For example, passwords transmitted from a terminal to a central database should be protected during transit by encipherment; so also, the response from the central database to the terminal accepting the validity

of the identity and password must be protected by encipherment. This is necessary to prevent against recognizing and reproducing the accepting response from the central database by an eavesdropper. As a matter of fact, if the terminal were an ATM, for example, this would allow the eavesdropper to corrupt the system without actually having the knowledge of any passwords.

Additionally, some flexibility must be included in both request and reply (for example, a time stamp), to prevent against a replay attack. A replayed request would deceive the central database and could deceive the ATM. For this reason, it is important that the passwords are not stored in clear form. The usual solution to this problem is encipherment. The original idea for using enciphered password lists is due to Needham, and is cited in [182].

By using an algorithm such as Data Encryption Standard (DES) for encipherment, the risk of compromise is transferred to potential disclosure of the encipherment key; hence, the encipherment key must be adequately protected. Bell Laboratories used the DES algorithm for protection of passwords in their Unix System [119]. It is important to note that, by itself, encipherment is not a solution. The fact is that verifying the correct identity of the receiver presents the same problem that the password is intended to solve in the first place. Hence, unless the passwords are transported is a secure manner, for example, by using the letter post for password transmission, their value would be diminished.

In banking, for example, where customers are verified by PINs, in order to prevent intervention of the password during transit, the password is recorded inside a sandwich envelope at the time of generation by the computer. The password does not appear on the outside because the character printer which imprints the password on the sandwich has no ribbon. Hence, the password is printed internally and is only readable when the envelope is open. Because the PIN is used in combination with a plastic card sent separately by ordinary mail; in order to authenticate the PIN, the authorized user must acknowledge safe receipt of the bank card. Therefore, an intruder must not only have access to the PIN but also have the physical possession of the bank card in order to benefit from his/her efforts. While this system offered some degree of protection, it is not necessarily suitable in situations where a high degree of security is required.

Another widely embraced idea called for selection of passwords unknown to the issuing organization. An example of a system that allows the use of personal choice password unknown to the system is discussed in details in [10]. In general, the system uses a one-way function using a combination of personal account numbers and passwords. The user chooses his/her password the first time the account is established by the bank from a computer keyboard shielded from any bank official, and combined with the personal account number, it is stored in the bank's computer database. In order to identify the user afterwards to the bank's computer system, the same one-way function is employed. Since the password is not transparent to the bank at any time; if the user forgets his or her password, a new

password would have to be chosen and activated by the bank. Furthermore, because the password is unknown to the employees of the bank, some degree of protection is provided to the user, hence, ordinary manipulation of the account cannot take place except only by unauthorized access by privileged system operators who could use commands to change a selected password by a user to a new one.

The exchange of passwords between a user and the system is conducted in "blind faith"; that is, when the system requests the user's password and it's supplied, there is no assurance at that brief time period that the system requesting the password is not a masquerade. The reason being that the system has not, at this point during the transaction process, authenticated itself to the user, and hence the user might be seeing a masquerade of the real system designed only for the purpose of capturing users' passwords. For example, in 1995, a phony ATM was installed in a Connecticut shopping mall to collect unsuspecting user's passwords. The scam works by having unsuspecting users supply their passwords to this phony ATM. The ATM collects the passwords and rather than dispense cash to the user, it gives out a message indicating that the system cannot process the user's request at this time. Eventually, the perpetrators were apprehended by the police, but the damage was done.

3.3 Methods of identity verification by a physical Key

3.3.1 Using a token for password

Besides passwords, another popular access control is by means of something acquired by the authorized user, for example, a token. A token is described as something possessed by a person to assist in identifying him or her. Its form could be a card, pen, key, etc. and it can have functions besides identification, for example, storage, access control, payment, etc. [46]. An example of a token widely used in the plastic card is the magnetic stripe.

Magnetic stripe cards are increasingly being used in applications such as for identification purposes, for credit validation, for automatic teller machine (ATM), other related business transactions, and for access control to secure sites. The physical dimensions for card and stripe are given by the International Organization for Standardization (ISO) [84]. The user identity is stored on the magnetic stripe with other relevant information.

In most cases, the magnetic stripe card is issued in combination with a special type of password called a personal identification number (PIN). The PIN is usually stored on the magnetic stripe in enciphered form in off-line systems and in the central database in an on-line system. In either situation, access is only permitted if there is a match between the claimed identity and the reference PIN.

Unfortunately, magnetic strip cards can be easily counterfeited. One method used to prevent the production of forged embossed cards is the hologram covering of some digits of the embossed number. The purpose is to ensure that any attempt to modify the embossing would destroy the hologram and hence reveal the forgery. Additionally, the contents of the magnetic stripe can be transferred from one card to another card's stripe simply by using the appropriate inexpensive hardware available to consumers today. Different methods are currently being tried to hinder the reproduction of the data on the magnetic stripe, of these; the two prominent ones are the watermark tape and the sandwich tape. Emidata uses the trade name "watermark" [24].

3.3.2 Smart Cards

A more recent development is the smart card (see chapter 6). A smart card is a plastic card, the size of a regular credit card, with a microprocessor embedded in the card. By incorporating a microprocessor chip into the card, it not only extends the storage capacity beyond what is obtainable on a magnetic stripe card (approx. 250 bytes), but also gives the smart card some data processing capabilities.

Until recently, the basic problem has been the difficulty of introducing the chip into the card. However, developments in chip technology have now advanced to the point where the chip is small enough to fit into the designated space on the

card, thickness being the critical dimension; new methods of mounting and connecting to the chip allow sufficient flexibility. Plated areas on the card surface provide the contacts to the card's circuitry.

Additionally, power supply is applied throughout the same circuit contacts in order to activate the card whenever it is placed in an input terminal device. A typical current smart card chip has an 8-bit microprocessor working at 5MHZ. The user memory is seldom higher than 8k bytes of electrically alterable memory. On the other hand, other cards contain only memory, hence, there is a limitation to the kind of operation that may be carried out by the card. It is worth noting that the clock frequency in PCs has increased up to 500Mhz and 64 bit and 128 bit microprocessors are now available. The user memory is usually expressed in megabytes, not kilobytes.

Information stored in the smart card's ROM may be semi-permanent; thus making it function as a typical magnetic stripe card used for some functions, for example, user identification. In spite of this, illegal alteration of the ROM contents is more difficult if not impossible because smart card's memory is scrambled to counter such practices, which could specifically erase vital information such as secret codes. Therefore, security of such a card is enhanced. Additionally, sentinel bits are randomly placed in the memory area. The microprocessor checks these bits from time to time, and stops working if they have been erased.

One application of a card with stored information is for viewing pay-per-view television. For example, if the transmission signal is sent out enciphered, such that a key is required to decipher it in order to receive the signal, then the decipherment key, which is changed periodically, say every 15 seconds, can be broadcast with the television signal but encoded under a Key Enciphering Key (KEK) which is stored on the card of the viewer. The transmitted key is deciphered in the key. Periodically, the KEK is altered, thus enhancing security. Additionally, having a valid card allow the consecutive transmission encipherment keys intelligible and used in the receiver.

The security of data on a smart card is further enhanced by storing all sensitive data on the same chip that holds the processing and encipherment facilities. For example, if the card has the ability to perform encipherment, this capability can be exploited in an identity verification procedure. The advantage here is that any attempt to eavesdrop on the channel between the card and verification terminal by an intruder would not yield any substantial result to aide an attack on the system.

3.4 Methods of identity verification by the physical attribute

Because it is possible, in practice, to compromise passwords due to negligence and to counterfeit a token; hence, the overall security of the system, other methods of identity verification using the physical attribute of the authorized user are being

explored with the hope that such methods will greatly reduce tampering if not totally prevent it. One very important issue is the need for an identity verification system to be acceptable to those required to use it.

Fingerprint verification, among many of such systems, is one such physical attribute we would consider because it has been the subject of much research and development especially in what is now known as biometric or automated recognition system. There is no perfect biometrics system for all the applications. Rather, some biometrics techniques may be more suitable for certain environments, based on the desired level of security and the number of users.

Fingerprints have long been employed as an identification tool especially in law enforcement because individuals have different patterns on the skin of the fingers. The study of these patterns is known as dermatoglyphics and presence of inherited physical disorders can be detected form these fingerprints. Unless plastic surgery is undertaken, the geometry of the fingerprints does not change. To match a user against stored records, it is essential to be able to classify fingerprints according to their characteristic features such as the arch, loop and whorl. Additionally, we have mixed types made up of elements of the other types. Each fingerprint has one or more of these characteristics. Other smaller features (minutiae); such as ridge bifucation and ridge end also play important roles in classification of fingerprints. There are between 50 and 200 such features on one finger.

Positive identification depends on the location and orientation of minutiae in fingerprints. Generally, detection of twenty minutiae can lead to positive identification of an unknown with a known print.

When a user presents himself or herself for identity verification, more dependable impression of the prints is obtained because the condition is controlled. Usually, fingerprint readers used for personal identity verification requires no ink pads, rather, they use total internal refection on a glass plate. Additionally, depending on the occasion, precise orientation of the print may vary; hence, it is essential that the reader allows the image to be rotated over a small angle. In general, when a user presents his/her finger to the fingerprint reader for identification, the reader checks for the minutiae which identifies their position and orientation. This data is compared with the original fingerprint information stored in the computer system database and based on the result of the comparison, access is either granted or denied.

Unfortunately, fingerprint verification for identity purposes is not widely accepted by users because it has a psychological disadvantage; that is, fingerprinting for identity purposes is usually associated with criminal elements in the society. A lot of law-abiding citizens are reluctant to have their fingerprints stored in any computer system for that matter. Another potential problem revolves around the possibility of someone losing a fingertip due to an accident or suffering

an injury on the fingertip. In either case, this might prevent the system from verifying the user. Overall, acceptance of the system must relate to ease of use.

Due to the difficulty of the verification process and the complexity of the fingerprint reader, the fingerprinting system is expensive. Optical and ultrasonic devices have been proposed for fingerprint detection because they have low False Rejection (FR) and low False Acceptance (FA) rates. However, problems such as dirty surfaces, among others, especially in optical device, may hurt the performance of the system, hence, the system may be unsuitable for an outdoor or unsupervised applications.

In the U.S., Dept. of Immigration and Naturalization services use fingerprints, among other biometric techniques, for alien identification programs. Also, the Los Angeles Department of Public Social Services is using a fingerprint-based identification system called Automated Fingerprint Image Reporting and Match (AFIRM) to identify eligible members within its relief benefit program. It should be noted that in countries where fingerprints have been traditionally used in identification cards for citizens, social rejection is considerably reduced.

3.5 Method of Identity verification by handwritten Signature

Dynamic signature verification is one of the several identity verification systems often referred to as behavioral features. This is due to the fact that

signatures have become a reflex action not subject to deliberate muscular control. Historically, signatures have always been accepted as a form of proof of identity or appended to a document indicating agreement as in the case of a contract. The introduction of magnetic stripe cards in the form of credit cards and identification can have also witnessed the widespread use of signatures as a means of personal identity verification. In cases where the validity of a signature is in dispute, opinions of signature experts are often solicited.

Other methods of signature verification besides visual inspection have been explored. Using machine to process handwriting is one of the methods gaining support because not only would this system be able to recognize the writer from the style of the handwriting, it would be able to interpret the state of mind of the writer.

Instruments for recording signature range form digitizing pads to bar code scanners. Signatures are not usually analyzed as prints; rather, they detect motion, speed, relative trajectories, and the relative motion of the pen device given to the user. The exact algorithm used by the manufacturer for this process is usually kept secret. Some specific drawbacks of this system include unsuitability for unattended outdoor applications, for obvious reasons, and for large crowds wanting to use the system, it is not time effective. Rather, for best results, controlled environment with manageable crowd is recommended, especially for indoor accesses to restricted areas.

Handwritten forgery detection by machine is based on an analysis of the original signature in terms of dimension ratio and pitch angles, measured for the signature as a whole and also for specific letters in it. These measures are then stored in a computer database against the identity of the user. Forged signatures on documents such as bank check is presented for verification, the signature on the check is compared against the profile of the original signature of the user stored in the bank's computer database, thereby allowing rejection of the handwritten forgeries. Encouraging results have been obtained with a system designed on this basis [123].

Selecting an identity verification system requires careful considerations. Unsubstantiated performance claims by manufactures need to be validated. It is essential that the user consider the attributes of the relationship between the user and system. For example, users in a constrained security environment can be made to adhere to defined procedures, whereas other users, say bank customers, can always relocate their accounts to other banks if they are unnecessary being burdened.

The National Bureau of Standards [124] has published a useful guide to principle of assessing and choosing identity verification systems, entitled "Guidelines on Evaluation of Techniques for Automated Personal Identification". This is a useful publication that details twelve points a user should consider before adopting an identity verification system. It is not unreasonable to expect that if the

users find any system unacceptable for reasons of inadequate convenience or slow speed of response, they will occupy themselves in finding means to circumvent the control, which the system provides.

4.1 Introduction

One of the primary methods to advance communication privacy is cryptography. This is the procedure of taking a plaintext communication and then encoding it into ciphertext. Messages encoded in this manner should be useful to only the sender and the intended recipient. All cryptographic schemes rely upon a "key." The key allows the person in possession of the message to decode it. Generally speaking, there are two different key encryption schemes. The first is the private key scheme. The second approach is the public key scheme.

The private key scheme relies on the existence of a single, secret key, which is used both to encrypt and decrypt messages. An example of private key encryption (PKE) is the Digital Encryption Standard (DES), which is a popular secret-key encryption algorithm originally released in 1977 by the National Bureau of Standards in the U.S. It was the first cryptographic algorithm openly developed by the U.S. government. The problem with DES and PKEs generally is transferring the key. There must be pre-existing channel to transfer the keys which is itself secure.

In the public key scheme a user has both a public key, which is published in a directory similar to a telephone book, and a private key, from which the public key

is derived. This has an advantage over private key systems since it is not necessary to exchange keys before messages can be decrypted. A popular public key system is RSA.

A related use for cryptography is the authentication of messages. Using public key encryption, a user can encrypt a message using his or her own private key. The recipient of the message can then determine the authenticity of the messages by using the sender's public key.

To be effective, standards must be established so that users in different networks will be able to exchange messages. Anything less than a full implementation makes it inconvenient and inefficient for users and reduces use and diminishes privacy. Because there are strong incentives to interconnect networks, it is expected that standards for encryption will develop rapidly.

A separate problem in the deployment of cryptographic methods is the prospect that national governments will seek to restrict privacy-enhancing technologies where they conflict with communications surveillance activities. Under the current U.S. law, the government is authorized to intercept the wire, electronic, or oral communications of a criminal subject by obtaining a special court order, which has been designed by Congress and approved by the Supreme Court.

Most governments today exercise some control of cryptographic apparatus if not of cryptographic research. The United States, for example, applies the same

export/import controls to cryptographic devices as to military weapons. Commercial applications revealed an urgent need for cryptography in the private sector. Today vast amounts of sensitive information such as health and legal records, financial transactions, credit ratings, and the like are routinely exchanged between computers via public communication facilities. Society turns to the cryptographer for help in ensuring the privacy and authenticity of such sensitive information.

A new challenge confronts the public practice of cryptography. The government has improved the widely published and available Data Encryption Standard (DES), with a secret algorithm implemented in tamper-resistant chips. These chips will incorporate a codified mechanism of government monitoring. The negative aspects of this "key escrow" program range from a potentially disastrous impact on personal privacy to the high cost of having to add hardware to products that had previously been encrypted in software. For example, providing an intercept capability would jeopardize security and privacy in two ways, first because the remote monitoring capability would make the systems vulnerable to attack, and second because the intercept capability itself would introduce a new vulnerability into the system.

The second part of the concern, that the intercept capability itself could introduce a new vulnerability, is at least potentially more serious. For instance, if the intercept capability is programmed into the switches and an unauthorized

person can break into a switch, then that person might be able to eavesdrop on a line or find out if a particular line is being tapped.

Indeed, "hackers" have broken into poorly protected computer switches and eavesdropped on communication lines. The switches can and must be designed and operated to prevent such break-ins independent of any intercept capabilities. Security is essential not only to protect against unlawful eavesdropping but to ensure reliable service and protect against other types of abuses.

To protect against possible abuses by employees of the service providers, access to the software for activating an intercept should be minimized and well protected through appropriate authentication mechanisms and access controls. The intercept control software might be left off the system and installed in an isolated partition only when needed prior to executing an authorized tap. Furthermore, implementing the intercept requirements could harm the competitiveness of U.S. cryptographic products in the global market.

It is now possible to purchase at reasonable cost a telephone security device that encrypts communications and to acquire software that encrypts data transmitted over computer networks. Even if law enforcement retains its capability to intercept communications, this capability could be diminished if criminals begin to hide their communications through encryption and law enforcement is unable to obtain access to the "plain text" or unscrambled communications. For example,

although it is technically feasible to intercept digital communications, not all systems have been designed or equipped to meet the intercept requirements of law enforcement.

If encryption becomes cheap and universal, this could pose a serious threat to effective law enforcement and hence to the public safety. Currently, the use of cryptography in this country is unregulated, though export of the technology is regulated. Cryptography is regulated in some of the major European countries. In order to assess whether cryptography can or should be regulated we need some idea of how it might be done. Three points [52] are offered as a starting point for discussion.

4.1.2 Weak Cryptography

While weak cryptography would offer adequate protection against most eavesdropping when the consequences of disclosure are not particularly damaging, it could be unacceptable in many contexts such as protecting corporate communications that are seriously threatened by industrial espionage. However, it is worth noting the general migration from analog to digital communications itself provides a high level of protection in the area of telecommunications, since such communications are only understandable with the aid of very sophisticated technology unlike the relative ease with which eavesdroppers can understand analog concepts.

4.1.3 Escrowed Private Keys

Ron Rivest has proposed using high-security encryption with "Escrowed secret keys" [153]. Each user would be required to register his or her secret key with an independent trustee, and cryptographic products would be designed to operate only with keys that are certified as being properly escrowed. The trustee could be some neutral entity such as the U.S. Postal Service, a bank, or clerks of the federal courts.

Additional protection can be obtained by distributing the power of the trustee. For example, two trustees could be used, and the keys could be stored with the first trustee encrypted under a key known only to the second. Alternatively, using Silvio Micali's "fair public-key cryptography," each user's private key could be split into five pieces, and each piece given to a different trustee [117]. The splitting is done in such a way that all five pieces are required to reconstruct the original key, but each can be independently verified, and the set of five can be verified as a whole without putting them all together.

In order to implement an approach based on Escrowed keys, methods would be needed for registering and changing keys that belong to individuals and organizations and for gaining access to the transient "session keys" that are used to encrypt actual communications. Key registration might be incorporated into the

sale and licensing of cryptographic products. To facilitate law enforcement's access to session keys, the protocols used to distribute or negotiate session keys during the start of a communications could be standardized. Once law enforcement has acquired the private keys on a given line, they would then be able to acquire the session keys by intercepting the key initialization protocol.

One drawback to this approach is the overhead and bureaucracy associated with key registration. Another is that it is limited to crytographic systems that require more-or-less permanent private keys. Although some such as the RSA public-key cryptosystem fit this description, others do not.

4.1.4 Direct Access to Session Keys

Ultimately a session key is needed to decrypt a communications stream, and this approach would give the service provider direct access to the session key when an intercept has been established in response to a court order. The service provider can then make the session key available to law enforcement along with the communications stream.

One way of making the session key available to the provider is for the provider to participate in the protocol used to set up the key. For example, the following three-way extension of the Diffie-Hellman public-key distribution protocol could be used to establish a session key that would be known only to the two communicants and the service provider: Each party independently generates a

random exponent x and computes $y = g^x \bmod p$ for a given g and prime p. All three parties then pass their value of y to the right. Next, using the received value of y, they compute $z = y^x \bmod p$ and pass it to the right. Finally, using the received value of z, they compute the shared session key $k = z^x \bmod p$, which will be the value g raised to all three exponents. An eavesdropper, who sees only the values of y and z, cannot compute k because he/she will lack the requisite exponent.

This approach has the advantage over the preceding ones of allowing the use of a strong cryptosystem while not requiring the use and registration of permanent keys. It has the disadvantage of requiring the service provider to be brought into the loop during the key negotiation protocol, which might also be difficult or costly to implement.

4.1.5 Protecting Privacy and Proprietary Interests

The last two approaches suggest that it is possible to regulate cryptography without compromising the privacy and proprietary interests of the citizens. Some people have argued that the citizens have a right to absolute communications secrecy from anyone, including the government, under all circumstances, and that requiring people to make the plaintext of their encrypted communications available to the government directly or indirectly would be nothing short of censorship, hence forbidding them from having private conversation in a secret place or using an obscure foreign language, or making them carry a microphone.

Viewed narrowly, cryptography offers the possibility for absolute communications protection or privacy that is not available to us in any other area of our lives. For example, although illegal eavesdropping posses a threat to corporate security, the communications network is not the weak link. Employees and former employees have posed a bigger threat. If companies themselves do not regulate cryptography, their employees would have a means of transmitting company secrets outside the company with freedom and without detection. Hence, corporate security is not necessarily best served by an encryption system that offers absolute secrecy to its employees.

4.1.6 Enforcing Cryptography Regulation

There have been concerns that criminals would violate cryptography regulations and use cryptosystems that the government could not decrypt, thereby also obtaining an absolute privacy beyond that of law abiding citizens. In other words "if encryption is outlawed, only outlaws will have encryption."

Cryptography can be embedded in a device such as a secure phone or security device attached to a standard phone that encrypts communications transmitted between phones (or fax machines), or it can be embedded in software packages or modules that run on computers and encrypt the communications transmitted over computer networks. It appears easier to regulate and control

telephone encryption devices than software. For example, if an approach based on escrowed keys is adopted, then the keys embedded in the products could be given to one or more trustees at the time of sale, and the products could be designed so the keys could not be changed without bringing the product in for service or negotiating a new key with a trustee on-line.

Similarly, if an approach based on direct access to session keys is adopted, a suitable key negotiation protocol could be built into the products. Software encryption, performed on personal computers or servers, could be much more difficult to regulate, especially since strong cryptographic methods have been distributed through networks such as the Internet and cryptographic algorithm can be implemented by any competent programmer.

Granger Morgan [118], has observed that controversy over the proposed digital telephony legislation is symbolic of a broader set of conflicts arising from several competing national interests: individual privacy, security of organizations, effective domestic law enforcement, effective international intelligence-gathering, and secure worldwide reliable communications.

Because the balance among these becomes hardwired into the design of our telecommunications system, it is difficult to adjust the balance in response to changing values. Technology has been drifting in a direction that could shift the

balance away from effective law enforcement and intelligence gathering toward absolute individual privacy and corporate security.

Less is known about the implications of regulating cryptography since no specific legislative or other proposal has been seriously considered. Although government regulation of cryptography may be inconvenient and subject to evasion, we should give it full consideration. Regulated encryption would provide considerably greater security and privacy than no encryption, which has been the standard for most personal and corporate communications. We must balance our competing interests in a way that ensures effective law enforcement and intelligence gathering, while protecting individual privacy and corporate security.

Finally, from a technical point of view, our communications future should not viewed from a narrow perspective. For example, in the future, communications are likely to be packet-based as much as circuit-oriented; are likely to be one-way as much as interactive; and are as likely to be between computers or electronic agents as between people. The ease with which effective cryptography can be implemented means anyone with a minimum of resources can achieve truly private communications.

More importantly, telecommunications networks security is fundamentally important for reasons of reliability and integrity in general. If someone can gain unauthorized entry into a telecommunications switch and bring a portion of the

communications infrastructure down, the consequences would be far worse than if someone listens in on a few conversations. However, most break-ins occur through sloppy practices, for example, no passwords or weak passwords, and not through holes in the technology. Such break-ins can be avoided through more robust and exacting authentication and access mechanisms.

4.2 Attacks on Cryptosystems

4.2.1 Cryptosystems and Cryptanalysis

Cryptology can be divided into two subdivisions: Cryptography and Cryptanalysis. While the cryptographer tries to ensure secrecy and/or authentication of his or her message, the cryptanalyst is trying to break a cipher or corrupt the coded messages by trying to present them as authentic. The cryptographer employs a secret key to control the enciphering process. The original message is called the plain text message, while the encrypted message is called the cipher text message.

The universal assumption of cryptography is that the cryptanalyst has full access to the cipher text. hence, the cryptographer adopts a principle, first stated by the Dutchman A. Kerckhoffs (1835-1903), that the security of the cipher must reside entirely in the secret key. Similarly, Kerckhoffs' assumption is that the entire process of encipherment, except for the value of the secret key, is known to the cryptanalyst.

Cryptography deals with the transformation of ordinary text (plaintext) into coded form (ciphertext) by encryption, and transformation of ciphertext into plaintext by decryption. These transformations are limited by one or more keys. The purpose of encrypting text is to provide security for transmission over insecure channels. Encryption procedures are generally said to make use of two types of transformations: substitutions and permutations (or transpositions). Both of these types of transformations can be combined in one application and most popular encryption methods such as RSA and DES use both substitution and permutation.

4.2.2 Types of Attacks

There are different levels of attacks on cryptosystems. The most common types are:

- Ciphertext-only attack - This is a situation where the cryptanalyst possesses a string of ciphertext, y. That is, based on the previously mentioned Kerckhoffs' assumption, the cryptographer will design the system for security against a ciphertext-only attack by the cryptanalyst.

- Known-plaintext attack - This is a situation where the cryptanalyst possesses a string of plaintext, x, and the corresponding ciphertext, y. That is, if the cryptographer further assumes that the cryptanalyst will have acquired some

plaintext cryptogram pairs formed with the actual secret key, then he will design the system for security against a known-plaintext attack.

- **Chosen-plaintext attack - This is a situation where the cryptanalyst has acquired temporary access to the decryption process. Hence by choosing a plaintext string, x, he or she can construct the corresponding ciphertext string, y. That is, if the cryptographer assumes that the cryptanalyst will submit any plaintext message of his or her own choice and in return, receive the correct cryptogram for the actual secret key, then he will design the system for security against a chosen-plaintext attack.**

- **Chosen-ciphertext attack - This is a situation where the cryptanalyst has acquired temporary access to the decryption process. Hence, by choosing a ciphertext string, y, he or she can construct the corresponding plaintext string, x. That is, if the cryptographer assumes that the cryptanalyst can submit declared "cryptosystem" and in return, receive the unintelligible garble which the cryptanalyst will decrypt using actual key, then he will design the system for security against a chosen-ciphertext attack.**

- **Chosen-text attack - Furthermore, if the cryptographer assumes that there is a possibility of combined chosen-plaintext and chosen-ciphertext attacks, then he will design the system for security against a chosen-text attack.**

In each case, the goal is to determine the key that was used for the encryption process. Most cipher systems used today are intended by their designers to be secure at least against a chosen-plain text attack.

These five levels of attacks are listed in increasing order of strength. Besides being the weakest type of attack, the ciphertext attack is also relevant to public-key cryptosystems. Additionally, it is reasonable to assume that the plaintext string is ordinary English text, without punctuation or spaces. This makes cryptanalysis more difficult than if punctuation and spaces were encrypted. Many techniques of cryptanalysis use statistical properties of the English language. In order to break a cipher, a cryptanalyst considers the frequencies of the individual letters and letter combinations of the supposed language of the plaintext.

Three of the most important services provided by cryptosystems are secrecy, authenticity, and integrity.

- **Secrecy refers to denial of access to information to unauthorized individuals.**
- **Authenticity refers to validating the source of a message; that is, that the message was transmitted by a properly identified sender and is not a replay of a previously transmitted message.**
- **Integrity refers to assurance that a message was not modified deliberately or accidentally in transit, by replacement, insertion, or deletion.**

A fourth service that may be provided is non-repudiation of origin, that is, protection against a sender of a message later denying transmission. Unfortunately, no single technique proposed to date has met all three criteria. Conventional systems such as the Data Encryption Standard (DES) require management of secret keys. Systems using public key components may provide authenticity but are inefficient for bulk encryption of data due to low bandwidths.

Fortunately, conventional and public key systems are not mutually exclusive; in fact they can complement each other. Public key systems can be used for signatures and also for the distribution of keys used in systems such as DES. Hence it is possible to construct hybrids of conventional and public key systems that can meet the previously stated goals: secrecy, authenticity, and ease of key management.

For survey of the preceding and related topics, (see [101], [109], [114], [146], [147], [154], [165]). More specialized discussions of public key cryptography are given, for example, in ([61], [115]). Mathematical aspects are covered, for example, in ([96], [139]).

The process of encryption can be defined as a functional operation where, E and D represent encryption and decryption transformations, respectively. It then follows that:

$$M = DE(M))$$

where M is the plain text message.

Alternatively,

$$M = E(D(M))$$

In this case, E or D can be employed for encryption. It is reasonable to assume that E and D are easy to compute when they are known. However, D is assumed to be secret, but E may be public.

Secrecy requires that a cryptanalyst should not be able to determine the plaintext corresponding to a given ciphertext, and should not be able to reconstruct D by examining ciphertext for known plaintext. This means that for a cryptosystem to provide secrecy, it must have the following properties:

- A cryptanalyst should not be able to determine M from E(M); that is, the cryptosystem should be immune to ciphertext-only attacks.
- A cryptanalyst should not be able to determine D given $(E(M_i))$ for any sequence of plaintexts $(M_1,M_2,...,M_n)$; that is, the cryptosystem should be immune to known-plaintext attacks. This should remain true even when the cryptanalyst can choose (M_i) (chosen-plain text attack), including the case in which the cryptanalyst can inspect $(E(M_1),...,E(M_j))$ before specifying M_{j+1} (adaptive chosen-plaintext attack).

Although adaptive chosen-plaintext is often regarded as the strongest attack, secrecy ensures that decryption of messages is infeasible. However, the enciphering

transformation E is not covered by the above requirements; it could even be public. Hence, secrecy leaves open the possibility that an intruder could masquerade as a legitimate user, or could compromise the integrity of a message by altering it. This simply means that secrecy does not imply authenticity/integrity. Authenticity requires that an intruder should not be able to masquerade as a legitimate user of a system. Integrity requires that an intruder should not be able to substitute false ciphertext for legitimate ciphertext. In order for the cryptosystem to provide these services, the following provisions should be made:

- It should be possible for the recipient of a message to ascertain its origin.
- It should be possible for the recipient of any message to verify that it has not been modified in transit.
- A sender should not be able to deny later that he sent a message (non-repudiation).
- It should be possible for the recipient of a message to detect whether the message is a replay of a previous transmission.

In conventional cryptosystem E and D are restricted by a single key K, so that we have:

$$D_K(E_K(M)) = M.$$

Although both E_K and D_K are secret, the algorithm for obtaining D_K and E_K from K are public. In this case the security of a conventional system depends entirely on keeping K a secret. Secrecy and authentication are both provided if two

parties share a secret key K, then they can send messages to one another that are both private (since an eavesdropper cannot compute $D_K(C)$) and provide authentication (since a would-be masquerader cannot compute $E_K(M)$). In certain instances (for example, transmission of a random bit string), this does not assure integrity; that is, modification of a message in transit may be undetected. In a similar way, integrity is provided by sending a compressed form of the message (a message digest) along with the full message as a check. Conventional cryptosystem are also known as symmetric (single key) systems.

4.2.3 Smart Card Security

Smart cards are equipped with both hardware and software security features. During the manufacturing process, hardware protection is included in the smart card's integrated circuits (IC). On the other hand, software protection is based on the use of encryption techniques and on data access controls.

The encryption method we proposed uses symmetric (private key) algorithm employing the simplest and most efficient complexity-theoretic generator called the Blum, Blum, and Shub (BBS) generator [25] (see figure 14.1b). As with other symmetric algorithm cryptosystems, key distribution is best handled by asymmetric (public key) algorithm cryptosystem. A popular asymmetric algorithm currently in use is RSA. RSA is capable of handling not only key generation, but also encryption.

Encryption techniques, many of which are currently in use, are based on secret algorithms. The purpose of the encryption process is to determine the use and scope of the secret algorithms. For example, in asymmetric cryptosystems, two keys (one is public and the other is private) are used. The private key is known only to the user whereas the public key is publicly known. These keys are linked to each other; that is, the private key is used for encryption and the public key is used for decryption. The best known asymmetric algorithm system is the Rivest-Shamir-Adleman (RSA) system.

In another example, the symmetric cryptosystems has one secret key, which is shared by the sender and the receiver of the message. Because the same key is used for encryption and decryption of the message, confidentiality is assured as long as the key is kept secret. The best known example of a symmetric algorithm cryptosystem is DES.

In zero knowledge cryptosystems, the cryptosystems have one or several keys known only to the authority granting access or communication rights - the third party. Although these systems are less suitable for messages in the real world, they are however often used for identification purposes; that is, neither the party demonstrating his or her identity nor the verifier needs to know the encryption algorithms or the secret keys.

The contents of the Smart cards are protected by secret codes. Encryption algorithms (usually DES) are also stored in protected memory areas. Secret keys for encryption algorithms are stored by the manufacturers or by selected issuers. The manufacturers also store some private information concerning personalization. This process allows cards of the same model to be split into unrelated families, thus avoiding security leaks coming from peer cards manipulated by unauthorized users. For example, suppose a manufacturing company is preparing a new service consisting of a syndicated lottery combined with the state lottery.

All transactions are made electronically. In this case, the Smart Card will undergo double personalization: a general process to characterize the cards pertaining to the application, and a specific process to identify every customer. Personalization data are vital for card security; these are protected by one or several manufacturer keys and possibly by encryption processes

Usually, further personalization is included in the cards when the issuing company designs applications for any given customers. It is not unusual to have at least one special key in the card, which belongs to the card issuer. Also, it is usual to reserve one or more issuer keys within the memory space for protecting areas where sensitive information is stored. Depending on the card model, issuer keys may be taken from the stock of regular secret codes, or be special keys prepared by the manufacturer. This is called a master key.

Master keys are usually encrypted. Some of the master key privileges include authorization to perform some restricted commands, such as partial or total card erasure (excluding the manufacturer's areas), loading of electronic purses, PIN reactivation, or utilization of encryption/decryption facilities.

The remaining memory and secret codes are left for users. User/Application memory is divided into files. Each file may be protected by one or more secret codes for reading and/or writing processes. Depending on the card model, one of the user's secret codes may have some privileges over the others. These codes are called PINs except that the Smart card PINs are not numbers, but alphanumeric strings. Giving access to the entire application or reactivating the card by a master key-protected command are some of the privileges of the PIN.

Other user's secret codes may be employed for protecting partial aspects of the application. Logic functions are implemented in many cards for secret code handling, thus allowing combined protection of specific areas. The use of secret codes becomes more involved when several applications share the same card. Because access to smart card memory is always controlled by the microchip, which checks that all the authorizations requested for writing or reading have been fulfilled, the final decision of carrying out any transaction is adopted by the card itself and not by the system.

4.2.4 Implementing Cryptography in Smart Cards

Encryption in smart cards is implemented as a means of carrying out several tasks. As a matter of fact, many Smart card designs incorporate the most widely used encryption scheme, the data encryption standard (DES) as a standard feature. Other manufacturers have developed other cryptosytems for communications, data handling, or telecommand (for example, Bull's TELEPASS). DES is a commercially available encryption scheme adopted in 1977 by the National Institute of Standards and Technology (NIST) and the National Security Agency (NSA), as Federal Information Processing Standard 46 [5]. The algorithm, based on a former IBM cryptosystem called LUCIFER, provides 2^{56} possible combinations. Two areas are of concern to experts regarding the security of the DES; the first being the key length. The key length in IBM's original LUCIFER algorithm was 128 bits, but that of the proposed system was only 56 bits, an enormous reduction in key size of 72 bits.

Experts feared (and still fear) that this key length is too short to withstand brute-force attacks. The second area of concern was that the design criteria for the internal structure of DES, the S-boxes, were and still are classified. Thus, users cannot be sure that the internal structure of DES is free of any hidden weak points that would enable NSA to decipher messages without benefit of the key. Whatever

the merits of the case, DES has flourished in recent years and is widely used, especially in financial applications.

Although more powerful cryptosystems (e.g., International Data Encryption algorithm (IDEA)) have been proposed, enhanced security can also be achieved with DES by using triple DES; that is, encrypt with key #1, decrypt with key #2, encrypt with key #1. By using triple DES, the number of combinations increases to 2^{112}.

In symmetric cryptosystem (e.g., DES), the sender and recipient share the same key. One of them is the card and the other is an external system. Used in this manner, the DES can be used for a number of services. For example, access control is accomplished by entering one or several secret codes. The master key (issuer's keys) are stored in the system. If the master key is needed for any transaction, it is encrypted and the resulting string is sent from the originating point. The card decrypts the string and compares it with its own copy. To avoid repetitions in the encrypted string, the master key is usually padded with random numbers.

The Smart card, through a process called challenge, may also verify that the external system has the appropriate authorization to work with it. For example, the Smart card interrogates the external system by generating a random number and sending an encrypted version to the external device. The external system decrypts the number and returns the correct answer to the Smart card. This exchange is only possible if the external system holds the same keys as the smart card. This

process can be reversed by the external system in order to verify the authenticity of the smart Card.

Message integrity and authentication are achieved by computing a cryptographic checksum over the message using the shared encryption key. For example, communications by modem or networks may be encrypted by the sender using a smart card and decrypted by the recipient with another smart card of the same family. The current batch of smart cards seldom use asymmetric cryptosystem. While the asymmetric cryptosystems are much slower than symmetric cryptosystems, they are capable of performing complex cryptographic processes; hence, we can expect to see widespread use of such system in the near future. Several semiconductor manufacturers, such as Atmel and Phillips, already include public-key encryption hardware in their chips [189]. The U.S. National Institute of Standards and Technology (NIST) Datakey Smart card, built on Hitachi's H8/310 microprocessor, implements an asymmetric cryptosystem. Similar implementations have been announced by both Cylink and Phillips.

Two linked keys, one public and one private, are used for encryption and decryption in these systems. Secured destination is achieved by the sender using the recipient's public key to encrypt the message. This message can only be decrypted by using the associated private key. Therefore, only the recipient can decrypt the message. Sender authentication is performed by using the sender's private key for encryption. As the recipient receives the message, he/she decrypts it using the

sender's public key. The recipient then knows that the message has been sent by the person whose public key has successfully decrypted it.

Because the keys in a symmetric system are too long (for example, 128 hexadecimal digits), users cannot memorize them. Furthermore, storing the private key in a computer, protected by a password, would make the system vulnerable to password crackers. Using the memory of a smart card to store the key or an enciphered version of it makes it much more secured. The smart card DES algorithm can also be used for message confidentiality (for example, specific message(s) can be decrypted by a specific recipient). In order to achieve this goal, the DES keys chosen are asymmetrically encrypted with the public key of the recipient. Once the keys are decrypted, they are shared only by the sender and the recipient. Hence, it is possible to use DES to send messages that cannot be decrypted by anybody else besides the intended recipient, including other Smart card holders from the same Smart card batch. This procedure is similar to the secured destination described earlier; the advantage is that symmetric systems are much faster than asymmetric ones (e.g., Hitachi's microprocessor takes 20 sec to perform a 512-bit operation).

It is possible to use these same procedures to produce digital signatures (i.e., the electronic equivalent to handwritten signatures for authenticating documents). Digital signatures are appended to digested versions of a message, that is, only a reduced version (known as a digest) of the message is linked with a digital signature.

Any message with a digital signature may be checked for authentication, integrity, as well as for nonrepudiation, that is, a procedure by which the recipient proves the identity of the sender to a third party. In asymmetric systems, this can be accomplished by means of the sender's public key.

Recently, the U.S. government proposed a digital signature standard called DSS. Although the standard has been widely criticized in various quarters by experts in the field of cryptography, if widely accepted, would allow real-time financial transactions and commercial agreements.

The public keys themselves may be questioned, especially if the sender and the recipient have never met. This situation is avoided with public key certificates containing the name of the key issuer, the public key, a validity period, and additional data from the user and his/her entity. These data are encrypted with the user's private key introducing a digital signature. In this way, the recipient may check at any time that the public key used to decrypt the message is the same as the key included in the message itself.

4.2.5 Cryptographic Algorithms: Data Encryption Standard (DES)

The most notable example of a conventional cryptosystem is Data Encryption Standard. The DES, also known as the Data Encryption Algorithm (DEA) by ANSI, the DEA-1 by ISO, has been a worldwide standard since 1976. DES is widely used

in businesses and for unclassified government applications around the world today. DES is well documented in [51], [63], [124], [125], [159], [168], [169] and is suitable for implementation in both hardware and software. DES is a block cipher, opearting on 64-bit blocks using a 56-bit key; that is, the key is usually expressed as a 64-bit number but every 8th bit is used for parity checking and is discounted. These parity bits are the least-significant bits of the key bytes. Essentially the algorithm is used to encipher or decipher, that is, the algorithm is fully reversible.

The key can be any 56-bit number and can be changed at any time. Because the system's security depends on the security of the key, a handful of numbers that are considered weak keys are avoided. The transformation employed using the same algorithm to encipher or decipher can be written as $P^{-1}(F(p(M)))$, where P is a certain permutation and F is a certain function that combines permutation and substitution. Substitution is accomplished by means of table lookups in so-called S-boxes.

The important features of DES are its single-key feature and the transformations performed during encryption and decryption. The combination of a substitution followed by a permutation on the text, based on the key is known as a round. DES has 16 rounds; it applies the same combination of substitution and permutation on the plain text block 16 times. Both permutation and table hookups are easily implemented, especially in hardware.

Encryption rates in excess of 40M-bits/Sec have been achieved [12], [107]. For this reason, DES is considered and efficient bulk encryptor, especially when implemented in hardware. The security of DES is produced by the alternation of substitutions and permutations. The function F is obtained by forcing a certain function f(x,y), where x is 32 bits and y is 48 bits. A sequence of 48-bit strings K_i is generated form the key.

Let L(x) and R(x) denote the left and right halves of x, and let XOR denote exclusive-or. Then if M_i denotes the output of stage 1, we have:

$$L(M_i) = R(M_i - 1)$$

$$R(M_i) = L(M_i\text{-}1) \text{ XOR } f(L(M_i),Ki)$$

It is expected that after 16 rounds, all patterns in the initial data will be undetectable; that is, the output will be statistically flat. The repetitive nature of the algorithm makes it ideal for use on a special purpose chip. Shannon [164], identified two features, which he said would not only be useful to incorporate into the design of a small key cipher but would also make crytanalysis much more difficult. These features he called confusion and diffusion. With confusion, the relationship between the statistical profile of the cipher text produced by the cipher and the description of the keys are made difficult.

On the other hand, diffusion broadens the statistical structure of a message over an extended portion of the message thus making each bit of the ciphertext dependent on a considerable portion of the plaintext message.

Although current implementations are better, the DES algorithm is not capable of providing meaningful ambiguity. Brute force cryptanalysis of DES cryptograms yield unique solutions for even short cryptograms. The security of DES is dependent upon the extent of confusion and diffusion the cipher provides. The design of the DES supports a high degree of diffusion and confusion.

Different modes for DES operation have been specified, including ECB (Electronic Code Book), CBC (Cipher Block Chaining), OFB (Output Feed Back), and CFB (Cipher Feed Back) modes [127]. The ANSI banking standards specify ECB and CBC for encryption and CBC and n-bit CFB for authentication [5]. Due to ease of implementation, ECB is often used in off-the -shelf commercial software products, although it is then vulnerable to attack. For example, although it is a little more complex than ECB, CBC is used every now and then, and it provides more security.

In the ECB mode, a block of plaintext encrypts into a block of ciphertext, hence, it is analytically possible for a cyrptanalyst to compile a codebook of plaintexts and corresponding ciphertexts for several messages without knowing the key. In CBC mode, the results of the encryption of previous blocks are fed back into the encryption of the current block; that is, each block is used to modify the encryption of the next block. For example, 64-bit block initialization vector (iv) is XORed with the first plain text data block before encryption is done with the DES

algorithm. The ciphertext output encrypting each data block is subsequently used in a preliminary XOR operation with each succeeding plaintext block in the same manner as was done with the initialization vector. Aside form DES, the modes do not have the same structural regularity that ECB mode displays. However, errors propagation in them extend beyond the immediate block of origin.

There has been much controversy on the security of DES regarding the key size, the number of iterations, and design of the S-boxes. Experts believed that the NSA installed a trapdoor into the algorithm to enable them easy access of decrypting messages, but IBM claimed otherwise. In 1981, Diffre theorized that for S50 million dollars, a special purpose parallel machine could be built that would be able to execute a brute force attack on DES encrypted text and successfully derive the plaintext and key within two days. While the cost and intent of the undertaking would undoubtedly deter most individuals or organizations, it surely would be within the means of the intelligence agencies.

Since 1981, however, while costs of implementation have steadily dropped, the speeds of processors in different hardware has increased dramatically. Brute force attacks using software on networked general purpose hardware (for example, workstations) are now possible. While such an attack would not guarantee a successful cryptanalysis in a reasonable amount of time, the odds are better now than before. It is also possible that a local area network with hundreds or even thousands of machines could be dedicated at off-hours to cryptanalysing DES at

absolutely no cost. If such distributed sites were to aid cryptanalysis of this kind, the odds of a successful cryptanalysis would even be better.

Besides being vulnerable to a dedicated crypt analyst, DES popularity is ascribed to the following properties:

- Besides having been subjected to years of public scrutiny, its specifications are public.
- It is much faster than public key cryptography algorithms. For example, a software implementation of DES on an IBM 3090 mainframe can perform 32,000 DES encryption's per second. In other words, DES is over 1000 times faster than RSA.
- It can be used in super ciphers; that is, triple-DES to obtain much greater security. Super ciphers involve multiple applications of cipher algorithms, and require that multiple keys be used.

4.2.6 Rivest, Shamir, and Adleman (RSA)

Rivest, Shamir, and Adleman (RSA) [151] obtained the best-known and most flexible public key cryptosystem. It supports both secrecy and authentication, and hence can provide complete and self-contained support for public key distribution and signatures; that is, it works for encryption and digital signatures. The security of the private components in the RSA cryptosystem depends on the difficulty of

factoring large integers. No efficient algorithms are known yet for solving this problem.

The RSA public key algorithm consists of two elements, a public key and a private key. the public and private keys are functions of a pair of large prime numbers. Obtaining the plaintext from the public key and the cipher- text is assumed to be analogous to factoring the product of the two primes. Besides being published for easy access, public keys are also exchanged in public. It is the private key that is kept secret and is not shared with anyone else besides the user.

Public key cryptography also provides solution to the problem of key management. For example, if two users, that is, users A and B respectively, want to communicate, private key algorithms (DES) require that they exchange secret keys prior to establishing the communication connections. Hence, user A and user B must then find a secure way to exchange their private key without giving out any secrets to a potential adversary or an eavesdropper.

To use the RSA system, user A encrypts a message to user B, using user B's public key. (The public keys are listed in a public key directory similar to telephone directory for ease of lookup by prospective users, or in the alternative, it could be sent over an insecure channel, for example, a radio communication channel. Once the encrypted message is sent out by user A, the only person who can successfully decrypt the message would be user B who has the proper decryption (secret) key to perform such decryption.

For example, suppose a user chooses primes p and q (see [166]) and computes n = p * q and m = (p - 1)(q - 1), then e is chosen to be an integer in [1, m-1] with GCD (e,m) = 1. The user then finds the reciprocal of e, modulo m ; that is, finding d in [1,m-1] with e * d = 1 (mod m). Now n and e are public; d, p, q, and m are secret. That is, for a user A, na and ea form the public element, and da, pa and qa form the private element. Following the computation of p, q, e, and d by the user, the private transformation and public transformation E are defined by:

$$E(M) = Me \bmod n$$

$$D(C) = Cd \bmod n$$

In the preceding equation, M and C are in [0,n-1]. We have D(E(M))M and E(D(C)) = C; D and E are inverses. Since d is private, so is D; and since e and n are public, so is E. This constitutes a cryptosystem that can be used for both secrecy and authentication. that is, for secrecy, user A sends Eb(M) to user B as expected; for authentication, user A sends DA(M) as expected.

For both secrecy and authentication, assume first that message digests are not employed. Suppose, for example, Na < Nb, then user A computes

$$C = EBa(DA(M))$$

And sends C to user B. Then user B recovers M as expected by

$$M = EA(DB(EB(DA(M))))$$

For non-repudiation user B may retain DA(M). This implementation works because the range of DA is a subset of the domain of EB; that is, [0,na-1] is a subset of [0,nb-1]. In case na > nb, Kohnfelder [94] notes that user A can choose to transmit C'= DA(EB(M))

and user B can recover M as:

$$M = DB(Ea(Da(Eb(M))))$$

Since the range of Eb is a subset of the domain of Da, this will work. To arbitrate possible disputes, user B must save C' and M. Then to prove that user A sent M, the arbitrator can compute Ea(C') and Eb(M) and test for equivalence. In spite of this, a disadvantage of this solution is observed in [44] and [51]; that is, user A signs Eb(M) rather than M. Another way is to apply Ea to the stored value Da(M) and realize M. Kohnfelder [94], proposed in his protocol, that both C' and M must be stored, thereby doubling the storage requirements.

By using message digests, this problem is eliminated. For example, assume H is a hash function. Once again, to send M to user B, user A could create a new message M containing M and Da(H(M)) and send C = Eb(M') to user B. This provides both secrecy and authentication; also, C may be saved for non-repudiation. Additionally, M and Da(H(M)) are reblocked in forming M', hence the sizes of Na and Nb are of no consequence. The speed of RSA in hardware is about 1000 times slower than DES. While there are chips that perform 1024-bit RSA encryption, the

fastest VLSI hardware implementation for RSA with a 512-bit modulus has an output of 64 kilobits per second [30]. Additionally although slower, RSA has also been implemented in Smart cards by manufacturers.

At present, RSA is about 100 times slower than DES in software implementation. However, there is promise of improvement as the technology changes. More detailed discussion on software speeds of RSA are in [97]. While the security of RSA is based on the assumption that factoring large numbers is difficult, it is conceivable to expect that an easy factoring method would compromise RSA security. From time to time, people claim to have discovered easy ways to break RSA, but to date no such methods have worked. The attacks that work against the implementation of RSA are not attacks against the basic algorithm, rather, they are attacks against the protocol. It is important to remember that it is not enough to use RSA, the proper implementation of the protocols should be followed.

4.2.7 Digital Signature Algorithm (DSA) - A Controversial Encryption Standard

There have been continued efforts for years by the U.S. government security officials to develop a single, reliable encryption standard for protecting electronic messaging. Part of the arguments in favor of this move was that advances in computerized communications technology called for a unified method to improve the security of unclassified government information and sensitive public data. As the networked population grew, so did the demand for technologies capable of

conducting standard business practices, such as verifying signatures, via computer. The governments proposal has provoked controversy and is yet to be accepted as a standard in some quarters.

On August 30, 1991, the National Institute of Standards and Technology (NIST) introduced a newly developed Digital Signature Algorithm (DSA) as the public encryption standard. The DSA is for use in their Digital Signature Standard (DSS), It was constructed under the guiding influence of the National Security Agency (NSA). Let me explain here for clarity purposes that: DSA is the algorithm; the DSS is the standard. The algorithm is part of the standard and the standard employs the algorithm.

According to the Federal Register [128]:

"A Federal Information Processing Standard (FIPS) for Digital Signature Standard (DSS) is being proposed. The proposed standard specifies a public-key digital signature algorithm (DSA) appropriate for Federal digital signature applications. The proposed DSS uses a public key to verify to the recipient the integrity of data and identity of the sender of the data. The DSS can also be used by a third party to ascertain the authenticity of a signature and the data associated with it. this proposed standard adopts a public-key signature scheme that uses a pair of transformations to generate and verify a digital value called a signature."

"And .... this proposed FIPS is the result of evaluating a number of alternative digital signature techniques. In making the selection NIST has followed the mandate contained in section 2 of the Computer Security Act of 1987 that NIST develop standards to .... assure the cost-effective security and privacy of Federal information and, among technologies offering comparable protection, on selecting the option with the most desirable operating and use characteristics."

Among the factors that were considered during this process were the level of security provided, the ease of implementation in both hardware and software, the ease of export from the U.S., the applicability of patents, impact on national security and law enforcement and the level of efficiency in both the signing and verification functions. A number of techniques were deemed to provide appropriate protection for Federal systems; the technique selected has the following desirable characteristics:

- **NIST expects it to be available on a royalty-free basis.**
- **Broader use of this technique resulting from public availability should be an economic benefit to the government and the public.**

The technique selected provides for efficient implementation of the signature operations in smart card applications. In these applications the signing operations are performed in the computationally modest environment of the smart card while the verification process is implemented in a more computationally rich environment

such as a personal computer, a hardware cryptographic module, or a mainframe computer.

NIST's signature verification and coding scheme has met with devastating opposition. Unfortunately, it was more political than academic. By the end of the six-month open comment period, NIST had accumulated over 100 responses form some of the encryption specialists in industry, academia, and other federal, state, and local government agencies. Over 90 of the respondents were against the plan calling it "weak", "irrational" and "potentially dangerous." It was soon obvious that the proposed digital signature standard (DSS) had not assembled many fans.

Strong criticism was directed toward the secrecy in which this encryption code was developed and its surprising lack of detail. For example, RSA Data Security, Inc.,. attacked the "common modulus," which is suspected of being able to give the government the capability to forge signatures. One respondent wrote, "it's hard to evaluate this proposal when pieces of the puzzle are held back."

More inflammatory, however, was NIST's rejection of the existing data encryption Standard (DES), specifically the very popular RSA public key algorithm generally regarded as industry's de facto encryption standard. After years of testing and proven reliability, RSA is now used by the majority of software makers around the world, including IBM, Apple, Lotus, Sun, Microsoft, DEC, Novell, and Northern Telecom. Furthermore, it is also the encryption measure chosen by most

government agencies. As another DSS opponent noted: "It's tough to expect universal acceptance of a totally untested algorithm."

NIST officials further explain that the intent is to offer the new standard to the public on a royalty-free basis, so government and industry could benefit economically from its wide-ranging commercial use. RSA public key algorithm, on the other hand, has always been made available to government agencies on a royalty-free basis.

The following demonstrate the controversial elements surrounding the DSS proposal:

1. DSA is different from the de facto public key standard (RSA).

About two-thirds of the U.S. computer industry is already using RSA. A partial list of current RSA users include IBM, Microsoft, Digital, Apple, General Electric, Unisys, Novell, Motorola, Lotus, Sun, Northern Telecomm, among others. These companies are using industry-developed interoperable standards - the public key cryptographic standard (PKCS) [89].

2. DSA is not compatible with existing international standards.

International standards organizations such as ISO, CCITT, and SWIFT, as well as other organizations (such as Internet) have accepted RSA as a standard. DSS is not compatible with ISO 9796, the most widely accepted international digital signature standard. Adopting DSS would create a double standard, causing difficulties for U.S. industry that have to maintain both DSS (for domestic or U.S. government use) and RSA (for international use).

3. DSA has patent problems.

Users of the NIST proposal may be infringing one or more patents. Claus Schnorr [160] claims that DSS infringes his U.S. patent #4,995,082, and Public Key Partners (PKP) asserts that DSS infringes U.S. patents #2,200,770 and #4,218,582. NIST does not give a firm opinion on this matter, and has not made licensing arrangements with either Schnorr or PKP. This leaves potential users of the NIST proposal vulnerable.

4. DSA is too slow.

This is true especially for verification. The signing speed is slightly slower than RSA, while the verification speed is over 100 times slower than RSA. Here, we

assume RSA uses a small public exponent such as 3, as is common practice. This slow verification time is likely to introduce unacceptable performance delays in many applications. For example, almost all public key applications are based on the use of certificates to authenticate public keys, hence, in a typical application, the ratio of verifications performed to signatures performed is fairly large. In other applications, such as software virus detection, the ratio can be enormous. The claim by NIST that signing speed is more important than verification speed just does not hold for most application .

5. DSA carries the risk of "trap-doors" in the primes.

While the NIST proposal does not require users to use a common modulus, it encourages such a practice. This is a security weakness, since "breaking" that one modulus can compromise the security of all users simultaneously. Therefore, users who share a modulus are putting all their eggs in the same basket. The NIST proposal should warn users always from such a practice; that is, each user should choose their own modulus. There is a risk of "trap-doors" in the primes. Arjen Lenstra and Stuart Haber at Bellcore (in their letter to NIST) observed that for some primes the discrete logarithm problem is "easy" for the person who created the prime (thus, a "trap-door" prime). Moreover, it may be difficult for a user to recognize trap-door primes. Again, NIST should have warned users not to use primes created by other; that is, each user should choose their own modulus.

6. DSA selection process is flawed.

The process by which the DSS proposal was generated, and by which it is being evaluated, seems to be flawed. It is curious that NIST did not publish a call for proposals for a signature algorithm. It is worth mention that NIST did so when the DES was adopted (The DES arose from a proposal from IBM). DSS was created without any such public solicitation for proposals. This is especially curious given the much greater cryptographic expertise now available outside the government, and the much larger base of public key applications already fielded. The DSA algorithm was created by the NSA, and adopted by NIST as its proposal, without any input from the U.S. industry.

The close-door approach toward the development of DSA, together with NIST=s assertion regarding patents have created a confrontational, rather than a cooperative, situation between NIST and the U.S. industry.

7. The DSA key size is too short.

The proposed DSA is restricted to a 512-bit modulus or key size. It is generally accepted in the cryptographic research community that this is too short for a system such as DSS, which is based on discrete logarithms and which requires a long life. To quote from a recent paper written prior to the announcement of DSS [99], "even 512-bit prime (Key size) appears to offer only marginal security." While

there should be lower limits on the key size to ensure a reasonable level of security, there is no reason for an upper limit. If, in spite of this argument, NIST keeps an upper limit, it should be increased to at least 1,024 bits.

8. DSA does not include key exchange.

Public key cryptography provides two advantages over conventional cryptography:

- Key exchange: the ability of users to communicate in private without fear of being overheard, and without couriers, registered mail, or similar means for rearrangement of a secret key.
- Digital signatures: the ability to sign messages which are easily checked by anyone, yet which cannot be forged or modified, even by intended recipient.

The DSA addresses only the second of these two needs. It would have been easier for NIST to include a key exchange standard with the DSA, either by adopting the RSA system [151] for both operations or by specifying the Diffie-Hellman key exchange system [55] as the standard to be used with the current DSA. (The Diffie-Hellman system is the natural key exchange choice if the proposed DSA is used for digital signatures). The DSS is derived from the Diffie-Hellman system.

In response to some of the controversies surrounding the DSS proposal, on May 7, 1992, NIST director John W. Lyons and James Burrows, director of NIST's

Computer Systems Laboratory appeared before the Judiciary Committee of the House of Representatives to discuss information security.

The following are the responses to the comments and criticisms the agency has received:

1. The DSA is different from the Defacto public key standard (RSA).

In response, NIST claimed that the DSA is a new invention designed to meet a number of federal government criteria. It is proposed as a federal government standard for all unclassified applications, including Warner Amendment information. The DSA is being proposed as a candidate for American National Standard by a voluntary industry standards group. While not compatible with the leading de facto standard public key algorithm, the basic mathematical operations (exponentiation of large integers in a finite field) are identical and a large amount (60-80%) of the code needed to implement the two techniques is identical. The International Organization for Standardization (ISO), is developing two generic digital signature standards, one which utilizes a hashing algorithm and one which does not. The DSA utilizes a hashing technique for efficiency purposes and can be proposed as a candidate algorithm for that generic standard.

2. The DSA is not compatible with existing international standards

In response, NIST claimed that the DSA was designed to satisfy many diverse and often conflicting criteria. It was difficult to meet, and impossible to optimize all of the criteria simultaneously. NIST was aware that the DSA is more efficient in computing signatures than in verifying them. Generating keys and signing must be efficient in some applications where limited computing capability may exist. For example, many users in the near future will carry a smart card, similar in size to a credit card, but employing a tiny computer that will generate personal signature keys and be able to "sign" electronic transactions for the owner. Prototype smart cards are in design now which sign in a second and verify in three seconds.

3. The DSA has patent problems.

In response, NIST claimed that a major criterion for the invention and selection of the DSS by the government was to avoid patented technology that would result in payment of royalties for government, commercial, and private use. A patent has been applied to the DSA by the government with the intent of issuing licenses for this patent on a nonexclusive, royalty-free basis. Three patent holders: Diffie-Hellman, Merkle-Hellman, and Schnorr; claimed that the DSA infringes on their patents. This infringement claims are still being reviewed.

4. The DSA is too slow.

In response, NIST claimed to be aware that the DSA is very fast in generating the needed keys. They further claimed to be aware of current DSS hardware implementations that sign and verify in milliseconds. The ability of the DSA to achieve greater efficiency through precomputation of some variables and through optimization techniques that are being developed will make the DSA highly desirable in many applications.

5. The DSA carries the risk of "trap-doors" in the primes.

In response, NIST claimed that they intend to include in a revised proposal a prime generator process which, together with the variable-length modulus, will mitigate many arguments about the security provided by DSA.

6. The DSA Selection process is flawed.

In response, NIST claimed it acted within its normal standards development procedure and the provisions of the Computer Security Act of 1987, drawing on the expertise of the National Security Agency (NSA), in proposing the DSS. In the normal standards development process, NIST identifies the need for a standard, produces technical specifications of a standard using inputs from different sources,

and then solicits government and public comment on the proposal. After the comment period, the comments are analyzed, appropriate changes are made and a revised standard issued (or further comment is solicited if the revisions are substantial). This public process is being followed.

7. The DSA key size is too short.

In response, NIST claimed that what they are proposing provides excellent security. It provides orders of magnitude greater protection than is needed in the foreseeable future. NIST proposed 512-bit modulus for efficiency reasons. Based on comments received and subsequent government analysis, NIST is proposing a number of modulus lengths between 512 bits and 1,024 bits.

8. The DSA does not include key exchange.

In response, NIST claimed that the DSA was designed to be a digital signature algorithm. It is not for key distribution encryption. The DES is a much more efficient method of data encryption than available public key techniques in current implementation. A technique is needed for key distribution that is optimized to meet all federal cryptographic criteria.

Finally, on May 19, 1994 the standard was announced [132]. The release stated in part: ".....this standard is applicable to all Federal departments and

agencies for the protection of unclassified information.....this standard shall be used in designing and implementing public-key based signature schemes which Federal departments and agencies operate or which are operated for them under contract. Adoption and use of this standard is available to private and commercial organizations".

5.1 Introduction

Cryptographers have always had to contend with the problem of not only having to design a secure cryptosystem, but also having to deal with the issue of key management. In real life, key management is the hardest part of cryptography. Even if the cryptographic algorithm is computationally infeasible to break, by not adequately protecting the keys; the entire system is vulnerable to cryptanalysis. Cryptanalysts often attack both symmetric and asymmetric cryptosystems through the flaws in their key management. For example, the last few years have witnessed reported incidents like;

- The Walkers selling U.S. Navy encryption keys to the Soviet Union for years.
- The Marines who guarded the U.S. Embassy in Moscow allowed easy access for the Soviets into the U.S. encryption facilities.
- The CIA's director of counterintelligence and his wife were "brought" for less than S2 million to provide the Soviets with classified secrets.
- And recently a Veteran FBI agent - Earl Edwin Pitts - was charged with selling secrets to Moscow for more than S224,000.

These incidences are not uncommon. However, the important point to be made here is the fact that it is cheaper for the cryptanalyst to "buy" information

through flaws in people protecting the keys than building massive cracking machines and luring brilliant cryptanalysts or even find flaws in the cryptosystems itself. For this reason, the management of cryptographic keys is an important factor in data security.

Keys used for authentication and data integrity need to be managed with care. A key management scheme should be able to handle a multi-user, multi-terminal and multi-host situation and protect data both in the communication network and in storage.

Complications often arise in the distribution of keys and their storage. Hence, key management should include every aspect of the handling of keys from their generation to their eventual destruction. Many commercial products that claimed using DES for protection often forget about other aspects of protection besides encryption. For example, the Macintosh's DiskLock program (version 2.1), claims the security of DES encryption.

Basically, it encrypts files using DES. While the implementation of the DES algorithm is correct, DiskLock store the DES algorithm key with the encrypted file. The troublesome aspect of this is that all a cryptanalyst has to do in order to read the files encrypted with DiskLock's DES, is to recover the key from the encrypted files and decrypt the file. In this particular example, among others, using DES is not a factor because the implementation is totally flawed. In order to move a key

through a communication network, it must be enciphered with another key. For example, if we have a key ks which is used to encipher data and we need to transport this key through the network, we will use another key kt to encipher it as EKt(ks). There are advantages in using a data enciphering key like ks for only a short period and then establishing a new one through the network.

Because the key ks is used to encipher data for just one session, it is called a session key. When the session key is being moved out to a terminal, the key kt used to encipher it for transit is call a terminal key. A terminal key is used for a longer period than a session key and it may have to be stored at a host computer with a number of similar keys for different terminals. In order to minimize the amount of secure storage needed, all these can be enciphered under another key km which is called a master key. Hence, the storage of kt is in the form EKM(kt).

In this key hierarchy, the master key is at the top; that is, the master key protects the terminal keys, and they in turn protect the data-enciphering keys ks, which in turn protects the data. In their protected form, the lower keys and the enciphered data can be stored in ordinary memory or transmitted through a communication network. The whole system depends on the master key which therefore needs to be properly secured. Usually in real life, the master key is stored in a physically secure box.

5.2 Generating Secure Keys.

How secure an algorithm is depends on the key. Any means or algorithm that is provided for generating keys carries with it a danger that someone has arranged the algorithm to generate predictable, though apparently random, numbers. It is a common knowledge that if the key generation algorithm is weak, then the whole encryption process will be weak. Hence, unpredictability is the essential requirement of an ideal method of key generation. Manual key generation is not suitable for the large number of keys in the levels below the master key.

5.2.1 Random Bit Generators.

One method of generating good keys is to use random-bit strings generated by some automatic process. If the key is 56-bits, any 56-bit key can be the key. The key bits can be generated from either a reliable random source or a cryptographically secure pseudo-random source or a cryptographically secure pseudo-random number generator. Where these automatic processes are unavailable a classical method of tossing a coin or throwing a dice can be employed.

Although it is essential to use a good random-number generator for key generation, the use of good encryption algorithms and key management procedures is best. Some encryption algorithms have weak keys; that is, keys that are less

secure than the others. For example, DES has 16 weak keys out of 256, hence the odds of generating any of these keys are minimal. Experts in the field of cryptography have argued that a cryptanalyst would have no knowledge of the use of a weak key and consequently derives no advantage from their casual usage.

Key generation for public-key cryptosystems is more difficult, because the keys must have certain mathematic requirements; that is, they may have to be prime or be a quadratic residue, and so on. The essential requirement from a key management standpoint is the unpredictability of the random seeds for those generators. Most systems use random number generators that are a little more complex, using many different sources of randomness or unpredictability, for extra safety. One source of unpredictability is a secret number which is already in the system such as a top-level master key. Though this master key is well protected against discovery, it is used in the cryptographic operations of the system.

A pseudo-random number can employ these operations, which are not available to a cryptanalyst outside the computer system. For example, the seed values used in generating the random sequence can be obtained by cryptographic operations (using the master key) on a number representing the current date and time.

5.2.2 Poor Key Choices.

Often, people choose their own keys and in doing so wind up choosing poor ones. For example, they choose easy to remember works like their spouses's names for keys. Experts in the field of cryptography have advised that while it might not always be possible to generate a random key, if you have to generate an easy-to-remember key, it should be obscure because in case of brute-force attack, the crytanalyst only need to try the obvious keys first. This type of attack where the cryptanalyst does not try all possible keys in numerical order, but rather try the obvious keys first is known as a dictionary attack, because the attacker uses a dictionary of common keys. A dictionary attack is very powerful when used against a file of keys as opposed to when used against a single key.

5.2.3 Reduced Key spaces.

DES has a 56-bit key; that is, any 56-bit string can be the key, if properly implemented. Norton Discreet for MS-DOS (version 8.0 and earlier) only allows ASCII keys, covering the high-order bit of each byte to zero. The program also converts lowercase letters to uppercase (so the fifth bit of each byte is always the opposite of the sixth bit) and ignores the low-order bit of each byte, resulting in only 240 possible keys. These poor key generation procedures have made its DES ten thousand times easier to break than a proper implementation [159].

5.3 Transferring Keys.

One of the prevailing difficulties in using symmetric cryptographic algorithm to set up secure communication is how to properly exchange the key securely. Fortunately, public key cryptography solves the problem with minimum of rearrangement. The X9.17 standard [5] defines two kinds of keys: data keys and key-encryption keys. Data keys are used to encrypt message traffic while key-Encryption keys are used to encrypt other keys for distribution. The data keys are distributed more frequently, whereas the key-encrypting keys are not distributed in frequently, but also have to be distributed manually in a secured tamperproof device like a smart card. This multi-level key concept is used frequently in key distribution [11].

Splitting the key into several different parts has also been proposed as a possible solution to the key distribution problem. The split key is then sent over a different channel; that is, by mail, by trusted courier, by telephone and so on. Unless in extreme cases, an eavesdropper could not possibly recover all the split parts, hence, the eavesdropper would still have no idea what the key is. Once the sender (say Alice) and receiver (say Bob) or vice versa both have the key-encryption key, they can communicate over the same channel.

Key-encryption keys work very well in small networks because the pair of users is relatively small. However, in large networks it quickly became unmanageable. For example, in a 500-person network, nearly 250,000 key exchanges are required. One way to make operation much more efficient is to create a central key server (or servers) to handle key distribution in large networks, in which the protocols provides for secure key distribution.

5.4 Key Verification.

Key substitution has always been a serious concern in public-key cryptography. However, when public-key cryptography is used with digital signatures and trusted key distribution centers, it becomes much more difficult to substitute one key for another. Other means of verification are also possible; for example, voice recognition scheme has proven to be a reliable means of authentication especially for public-key cryptography. Sometimes, key verification can be done over the telephone where it is possible to hear actual voice; that is, if it is public key, it can be safely recited in public and if it is a secret key, a one-way hash function could be used to verify the key. This type of key verification is used both by the Pretty Good Privacy (PGP) [187], a freeware electronic-mail security program and by the AT&T Telephone Security Device (TSD) which is actually a clipper phone.

Errors introduced into keys during transmission often results in garbled keys and eventually undecryptable ciphertext. This is often a problem, and it has been strongly suggested by experts in the field that all keys should be transmitted with some form of error detection and correction bits. This way, errors introduced into keys during transmission can be easily detected and when necessary, the key can be retransmitted.

5.5 Using Keys.

Implementation of encryption in software in today's computing environment is terrible. In the past, microcomputers operate under the control of simple operating systems. Today, the operating systems have become complex and they perform multiple functions (multitasking). A few of the most common examples of such operating systems include Windows NT, UNIX, and Macintosh System 7. It is difficult, if not impossible, to follow in sequence the operation of these operating systems. For example, it is difficult to determine when the operating system will suspend the encryption application in progress, write everything to disk, and then depart to take care of some other urgent task. Later the operating system will return to encrypting whatever is being encrypted and everything will be alright.

However, the concern here is that when the operating system wrote the encryption application to disk, it also wrote the key along with it. Hence, having the key on the disk, unencrypted, until the computer writes that area of memory again

could compromise the entire system, especially, if an adversary attacks the system and decides to go over the hard drive with some precision. As a countermeasure, the encryption operation in a multi-tasking environment should be assigned high priority so that the process will not be interrupted once it starts.

On the other hand, hardware implementations are more safer. Many encryption devices are designed to erase the key if tampered with [159]. For example, the IBM PS/2 encryption card has an epoxy unit containing the DES chip, battery, and memory. Other communications applications, such as telephone encryptors, can use session keys. A session key is a key that is just used for one communication session; that is, say for a single telephone conversation, and then discarded. Because there is no reason to store the key after it has been used, it is unlikely that the key would be compromised.

Controlling key usage in certain applications might be desirable for different reasons. For example, some users need session keys only for encryption or only for decryption . The session keys might also be authorized for use on a certain machine or at a certain time. One method for handling these sort of restrictions is to attach a Control Vector (CV) to the key; the control vector specifies the uses and restrictions of that key [111]. This control vector is hashed and XORed with a master key; the result is used as an encryption key to encrypt the session key. The resultant encrypted session key is then stored with the control vector. To recover the session key, the control vector is XORed with the master key, and the result is used to

decrypt the encrypted session key. the advantages of this method are that the control Vector can be of arbitrary length and that it is always stored in the clear with the encrypted key. Hence, with a tamperproof hardware (e.g., Smart card), the user will not have direct access to the keys.

5.6 Storing and Updating Keys.

There are a lot of ways to change keys daily for whatever reason(s), but the easier solution to change or distribute keys daily is simply to generate a new key from the old key. This process whereby a new key is generated form the old key is called key updating. By using a one-way function, it is easy to actualize key updating process between two users. For example, since both users share the same key and they both operate on it using the same one way function, they will get the same result. From these results, they can take the bits they need to create the new key.

Furthermore, the security of the new key depends on the security of the old key; that is, so long as an eavesdropper does not have the old key, even if a ciphertext-only attack is mounted on the encrypted traffic using the new keys, the attack would not yield any meaningful results.

Besides the issue of changing keys daily, another issue of concern is the storage of keys. The more manageable solution to the management of keys that are

used in a single user environment to protect data stored in files is to avoid storing cryptographic keys in the system. Rather, a user is responsible for managing their own keys and entering them at the time of encryption or decryption. In Information Protection System (IPS) [95], for example, keys do not reside permanently in the system. A user can either directly enter the 64-bit key (56 key bits plus 8 parity bits) or enter the key as a longer character string. The system then generates a 64-bit key from the character string using a key-crunching technique.

In a similar way, a key could be recorded on magnet stripe cards, smart cards or plastic key with an embedded ROM chip called a ROM-key [(67), (71), (48)]. This hardware-implemented key could then be entered by a user into the system by inserting the physical token (Smart card) into a special card reader attached to the user's terminal. In order to establish a secure channel between their computer terminals and the host central computer in conventional systems, users may register private keys with the system. In some systems, the master keys are used for this purpose. The master key is usually stored in a file enciphered under the system's master key. Additionally, encryption keys cannot be protected with one-way functions, because, unlike passwords, it would be impossible to recover them.

On the other hand, a user is not required to register a private key with the system in order to establish a secure channel in public-key systems. This does not mean that the key does not require security. For example, if the key is used for

signatures, it must be protected from disclosure to prevent forgery. As matter of fact, it must be protected from deliberate disclosure by the user. For example, if the user could take away the private key or claim to have lost it, then it is possible for the user to deny sending or receiving any message [see (155), (104)].

To avoid this type of situation, Merkle [113] proposed that the user's signature key should not be disclosed to anyone, including the user too. Rather, a single copy of the user's private key should be kept in a dedicated microcomputer or sealed in a ROM. Additionally, this method could be made more secure by splitting the key into two halves, storing one half in a dedicated microcomputer and the other half in the ROM Key. The U.S. government's NSA-designed secure phone, STU-III, works this way. By compromising either the ROM key or the system does not compromise the cryptographic key because an adversary must have both parts. For example, misplacing the ROM key does not compromise the cryptographic key and vice versa for the terminal key.

Sometimes keys that are not so easy to remember are usually stored in encrypted form using a key-encryption key form. For example, an RSA private key could be encrypted with a DES key and stored on disk. To recover the RSA portion of the key, the user would have to type in the DES Key to a decryption program. Keys that regenerated with a cryptographically secure pseudo-random-sequence generator might be easier to regenerate from not-so-difficult to remember

passwords whenever needed. Conceptually an unencrypted key should never be displayed outside the encryption device.

5.7 Compromised and Backup Keys.

Ordinarily, all protocols and algorithms are considered secure only if the key (the private key in a public-key system) remains secret. If a key is lost, stolen, or otherwise compromised, then the security of the system it's expected to secure is also compromised. The damage incurred when a key is compromised is directly dependent upon the class of the key,. For example, if the compromised key was for a symmetric cryptosystem, the concerned user would need a new key and also hope that the actual compromise would be minimal. On the other hand, if the user's key was a private key, and probably on servers throughout the network, then the user has a different set of problem; that is, an eavesdropper can impersonate the user on the network, thereby gaining unauthorized access to encrypted mail, signing contracts or entering into one on behalf of the legitimate user and so on. In other words, this is a major compromise.

As a standard practice, whenever there is a compromise of a private key, all databases of public keys on the network should be notified of the particular private key that has been compromised. This is to prevent an attack against an unsuspecting user who might want to encrypt a message with the compromised key. In a situation where the Key Distribution Center (KDC) is managing the keys, the

user whose key has been compromised should notify the center. Otherwise, the user should notify all correspondents that messages received after the compromise of the key will be invalid. A corrective measure to this problem will be to append a time stamp to messages in order to determine which message is authentic and those which are not.

The purpose of encryption is to make file recovery impossible without the appropriate key. Should a user loose his or her key, and unless there is a backup key, the files are lost for good. However, one way around this problem is to use the key escrow [181]. Using the key escrow, the private key is split into a predetermined pieces and stored by different escrow agencies. Also, the pieces have the additional property that they can be individually verified to be correct, without reconstructing the private key. Hence, the loss of a piece would not necessarily compromise the entire system, since none of those pieces alone is the key. The concern with the escrow key management system is that the users has to trust the escrow agencies not to misuse their keys. A comparable but better solution is to use a secret-sharing protocol [162].

Another backup scheme [21] uses Smart cards for the temporary escrow of keys. A user can store the key to secure his or her computer system's hard drive in a smart card. This card can then be given to a friend to make use of the computer system while the owner is away on vacation. The rationale here is that while the friend can use the computer system, there is no way to learn the key stored in the

card. Furthermore, reciprocity is built into the system; that is, while the friend can verify that the key will provide access into the user's computer system, the user on the other hand can verify if the key has been used and how many times it was used.

In general, this scheme would not work in all cases. For example, on a secure telephone, the key should exist for the length of the call and no longer. For data storage, key escrow might be a good idea. Whatever the application, it is reasonable to keep backups of data-encryption keys in the same manner as we keep off-site backups of other data files.

5.8 Lifetime of Keys.

Our experiences in life has shown that means of personal identity like driver's licenses or passports should have expiration dates on them to deter fraud. By the same reasoning, encryption keys should also expire automatically at some predetermined point in time. There are several reason for this; for example,

- The longer a key is used, the more the loss if the key is compromised.
- The longer a key is used, the more the possibility of compromise.
- The longer a key is used, the more the temptation for an adversary to spend resources to attempt to break it.
- ciphertexts encrypted with the same key are vulnerable to cryptanalysis.

Depending on the application, different keys have different lifetimes. For example, where practical, session keys should be changed daily. Also, keys used for gigabit-per-second communications link might have to be changed more often than the key for 9600-band modern link. Another example is key-encryption keys that are replaced infrequently because they are used occasionally for key exchanges. Because of the limited frequency of usage, they generate a minimal amount of ciphertext for a cryptanalyst to work with. However, if a compromise does occur, the loss could be devastating; that is, all communications encrypted with all keys encrypted with the key-encryption key will be compromised.

Conversely, the encryption keys used to encrypt data files for storage cannot be change often. On the contrary, decrypting and re-encrypting them with a new key so often doesn't necessarily enhance security in any way, rather, it just provides the cryptanalyst with more information to work with. One possible solution would be to encrypt each file with a unique file key, and then encrypt all the file keys with a Key-Encryption-Key [159]. The down side to this solution is that losing this key would amount to losing all the individual file keys. Hence, it is advisable that the key-encryption key should be stored in a secure location, possibly in a safe somewhere.

Private keys for public-key cryptography applications have varying lifetimes, depending on the application. In some cases, private keys used for digital signatures and proofs of identity may last a lifetime, whereas in other instances, private keys

are discarded immediately after they satisfied their intended use. For example, private keys used for coin-flipping protocols can be discarded immediately after the protocol is completed. No matter the life expectancy of a key's security, it is still advisable to change the key periodically. The reason being that while new keys would be used to sign new documents, old keys are still good and still have to remain secret, should the user have reason to verify a signed document from that period. Furthermore, by using new keys to sign new documents, this reduces the number of signed documents a cryptanalyst would have to work with.

5.9 Destroying Keys.

When old keys are replaced with new ones, it is prudent that the old keys are safely destroyed. Old keys, if not properly destroyed, could provide an adversary with useful tools to read old messages encrypted with those keys [9].

Keys must be deleted or destroyed securely. Usually when a file is deleted on most computers, the file isn't really deleted. What's really deleted is an entry in the disk's index file, indicating to the machine the existence of the file. In today's computer software market, there are many file-recovery programs that recover data files after they have supposedly been deleted.

Additionally there is the issue of virtual memory to deal with. Simply put, virtual memory means that the computer can read and write memory to disk at any

time during processing. In order to effectively erase a file so that file-recovery software cannot read it, all of the file's bits on the disk need to be physically written over. The National Computer Security Center (NCSC) [1281] states:

"Overwriting is a process by which unclassified data is rewritten to storage locations that previously held sensitive data... To purge the ... storage media, the DoD requires overwriting with a pattern, then its complement, and finally with another pattern; e.g., overwrite first with 0011 0101, followed by 1100 1010, then 1001 0111. The number of times an overwrite must be accomplished depends on the storage media, sometimes on its sensitivity, and sometimes on different DoD component requirements. In any case, a purge is not complete until a final overwrite is made using unclassified data."

In situations where the key is in a hardware such as EEPROM or on a computer disk, the actual bits of the storage should be overwritten multiple times. If the key is in a hardware EPROM or PROM, the chip should be disintegrated into disposable tiny pieces. However, there is a potential problem because keys can be easily copied into multiple storage locations. For example, computers that perform their own memory management, are constantly swapping programs in and out of memory, thereby increasing the problem.

There is no guaranteed way to ensure successful key erasure or that one has taken place in the computer, especially if the computer's operating system controls the erasure process. To be on the safe side, not only would erasing files be in order,

but erasing the entire drives including all unused space on the hard disk would be appropriate.

5.10 Key Management in Conventional (one-key) Cryptosystems.

Regardless of whether a public key or conventional cryptosystem is used, users are required to obtain recipient users keys to communicate securely. Although key management is made easier with Public-Key cryptography, that is by no means saying that it has no problems of its own. For example, a chosen-plaintext attack can be particularly effective if there are relatively few encrypted messages. In most practical implementations public-key cryptography is used to secure and distribute session keys; those session keys are used with symmetric algorithms to secure message traffic [94]. This is sometimes called a hybrid cryptosystem.

Using public-key cryptography for key distribution solves an important key-management problem; that is, session keys are created when needed to encrypt communications and are destroyed when they are no longer needed. This reduces the risk of compromise. On the other hand, the private key is vulnerable to compromise, but it is at less risk because it is only used once per communication to encrypt a session key.

To actively address the issue of key management using conventional cryptography, the network should provide a Key Distribution Center (KDC). The

KDC is a trusted network resource that shares a key with each subscriber and provides additional keys to the subscribers as needed. When one user wants to communicate securely with another, the user obtains a session key from the KDC for use in that particular conversation. There is wide variation in key distribution protocols depending on the cost of messages, the accessibility of multiple simultaneous connections, whether the users have synchronized clocks, and whether the KDC has authority not only to facilitate, but also to allow or prohibit, communications [166].

In the following example, the importance of cryptographic authentication property is demonstrated. In a conventional context, any altered message will not pass the authenticity test, hence the introduction of the concept of a certificate. A certificate is a cryptographically authenticated message that contains a cryptographic key. Modern key management concepts exploits the virtues of this concept. For example, if two users - named A and B respectively - want to exchange communication, the following is a typical protocol to establish such communication steps;

- When user A wants to communicate with user B, user A calls the KDC and requests a key to communicate with user B.
- The KDC sends user A a pair of certificates. Each certificate contains a copy of the requested session key, one encrypted so that only user A can read it and the other so that only B can read it.

- When user A calls user B, user A presents the issued certificate as a means of identification to user B. Each of them decrypts the appropriate certificate under the key that user A shares with the KDC and hence, get access to the session key.
- User A and user B can now communicate securely using the session key.

In order to avoid going through this procedure on every call, user A and user B can save the certificates issued by KDC for late use. This process of storing keys for later use affords the subscriber the pleasure of not having to call the KDC every time a call needs to be made. Although the number of KDC calls is still proportional to the number of distinct pairs of users who want to communicate securely. The downside to this arrangement is the fact that since each user shares a key with the KDC, if the KDC is compromised, the users would also be compromised.

The use of public-key cryptograph can facilitate greater improvements both in terms of the economy and security. The certificate generated by the KDC serves as an instrument of introduction, introducing one user to another. In the protocol above, user A obtained a certificate from KDC which serves as instrument of introduction to user B only. On the contrary, if this scenario were for a network using public-key encryption, user A would require only a single certificate from the KDC that introduces the user to any other network user [94].

Upon analysis, we found that in a public-key network, each user has the public-key of the KDC and hence the ability to authenticate any message from the KDC, but lacks the capability to forge one. Whereas, in a convetional network, each user shares a secret key with the KDC and can only authenticate messages specifically meant for the user, that is, if for example user A has the key needed to authenticate a message intended for user B, then user A will also have the potential to create such a message but authentication of such message will fail.

In order to use a certificate obtained from the KDC prior to making any secure calls, user A and user B will communicate with each other using the following protocol:

- User A sends a certificate to user B.
- User B sends a certificate to user A.
- Each user checks the KDC's signature on the certificates they have received.
- Using the keys contained in the certificates, user A and user B can now communicate securely.

In this case, the keys issued by KDC are public-keys and any message(s) encrypted with these keys can only be decrypted by using the matching secret keys, to which the KDC has no access. Hence, the added security is that the KDC is not privileged to any information that would enable it to eavesdrop on the users.

Roger Needham and Michael Schroeder [134] conducted a definitive attack on the system by comparing conventional key distribution protocols with similar ones using public-keys. They concluded that conventional cryptography offers more efficiency than public-key cryptography. The study however failed to record that security was better using public-key protocol than using the conventional protocol.

In conventional cryptography, by corrupting the KDC, the network it serves is also corrupted. By corrupting the KDC, the intruder gains access to information sufficient enough for recovering the session keys used to encrypt messages (past and present).

The intruder faces a more difficult situation on a public-key network. Despite the fact that the KDC has been corrupted, the information supplied is insufficient to an adversary to read the information recorded by a passive eavesdropping. In order to gain access to this information, the intruder not only has to forge certificates but also fool other users into encrypting their messages with phony public-keys. It is worth noting that the KDC's secret keys are useful only for signing certificates containing user' public-keys, it does not provide capability for the intruder to decrypt any user traffic.

In order to eavesdrop on a call from user A to user B, the intruder with possession of the KDC's secret key must intercept the communication in which user B sends user A the certificate corresponding to user B's public-key. Then he

substitutes it for a public-key the intruder has fabricated and whose corresponding secret key is known only to the intruder. This will enable the intruder to decrypt any message that user A sends to user B. The danger in this arrangement is that if a misencrypted message gets to user B, and user B cannot decrypt such message, then user B would request a retransmission which thus alerts user A to the error and hence the suspicion of intrusion on the link.

To get around this problem, the intruder(s) must therefore intercept user A's messages, decrypt them, and re-encrypt them using user B's public-key so as maintain the illusion. If the intruder wants to decrypt user B's replies to user A, the same procedure, as before, will be executed with user B, that is, supplying user B with phony public keys for user A and translating all the messages user A sends to user B.

In real life, active wiretaps are not only detectable, the scenario illustrated above is not manageable because the number of intruders grows in proportion to the number of users to be intercepted, hence the whole process will be inefficient. Over large network, for example, radio broadcast network, message deletions are extremely difficult. This forces the intruder(s) to place their taps close to the targets and recreates the environment of conventional wiretapping, thereby denying the opponents those advantages of communication intelligence that make it attractive in the first place.

The use of a hybrid scheme lessens the gain in security because the intruder need not control the channel after the session key has been selected. This problem can be solved by occasionally using the public-keys to exchange new session keys [59]. Also, the problem of using compromised keys to read messages taken at an earlier date [9], which is actually common in conventional cryptography security scheme can be solved by using public key cryptography scheme. Furthermore, a more lasting solution can be produced by combining exponential key exchange and digital signatures. This solution is inherent in the operation of a secure telephone currently under development at Bell-Northern Research [(60), (137)] and intended for use on the Integrated Services Digital Network (ISDN).

Each ISDN secure phone has an operating secret key/public-key pair that has been negotiated with the network's key management facility. The public-key portion is included in a certificate signed by the key management facility along with such identifying information as its phone number and location. In the call setup process that follows, the phone uses this certificate to convey its public key to the phones.

- The telephones perform an exponential key exchange to generate session keys unique to the current phone call. These keys are then used to encrypt all subsequent transmissions in a conventional cryptosystem.
- After establishing an encrypted (unauthenticated) communication channel, the telephones start exchanging credentials. Each telephone sends the other its public-key certificate.

- **Each telephone then checks the signature on the certificate it has received and extracts the other telephone's public-key from it.**
- **The telephones now query each other to sign test messages and check the signatures on the responses using the public-key from the certificates. Once the call setup is complete, each phone displays for its user the identity of the phone with which it is in communication.**

Using exponential key exchange produce unique session keys that exist only inside the telephones and only for the duration of the call. Hence, it provides a security assurance lacking in conventional cryptography; for example, after the completion of a cal between two uncompromised ISDN secure telephones, and the destruction of the session keys, by compromising the stored long-terms keys in the telephones will not aid an intruder to decrypt the recording of the call. By using a combination of intercepted messages and long-term keying material, it is possible to acquire session keys, through the implementation of conventional key Management techniques. Whenever a long-term conventional keys are compromised, all communications, including those of earlier dates, encrypted in derived keys, are compromised too.

In the 1970's, Christopher Boyce, a code checker, who worked for the Central Intelligence Agency (CIA), copied keying material that was supposed to have been destroyed and sold it to the Russians [103]. Recently, Jerry Whitworth of the Communication center of the Alameda Naval Air Station [13] did the same

thing. If exponential key exchange had been used, it would have rendered any previously used keys worthless.

Message digest is another value-added component of modern public-key cryptography. However, signing an entire document encrypted with a single secret key implemented through a digital signature has two primary disadvantages; first, using the signature process to encrypt the message means that the recipient of the message has to retain the ciphertext for as long as the signed message is needed. The other disadvantage is that both the signature process, and the verification process will be slow because a public-key system is a slow system. In order to maximize the benefit of the encryption process at this time, the user has two alternatives; the first being that the cipher text has to be repeatedly decrypted, or in the alternative, save the message plaintext as well as cipher-text. In either case, the potential is there for compromise.

Donald Davies and Wyn Price [42], of the National Physical Laboratory in Teddington, England, were the first to propose a solution to this problem. They proposed using the digest of the message or building a cryptographically compressed form of the message and signing by encrypting the message with the secret key. Besides the advantage of allowing the signature to be circulated independently of the message, it also offers tremendous economy of scale.

Additionally, this is also true in protocol systems; especially in situations where a portion of the message required in the authentication process is not actually transmitted because both parties are aware of its existence. Until recently, public key cryptography, specifically key management, has not used certificates as a strategy. While most of the problems associated with public key cryptography have long been solved, critics still categorize the system as being weak when it comes to implementation on authentication.

Part III: Authentication Protocol

Chapter 6. An Introduction

Secrecy and integrity are two common requirements for secure communication. Secrecy specifies that a message can be read only by its intended recipients, while integrity specifies that every message is received exactly as it was sent, or a conflict is detected. A strong cryptosystem can provide a high level of assurance for secrecy and integrity. More precisely, an encrypted message provides no message regarding the original message, hence assuring secrecy; and an encrypted message, if tampered with, would not decrypt into a useful message, hence assuring integrity. Replay of old messages can be foiled by using time stamps or nonces ([53], [144]). Perfect random numbers are good nonces.

The purpose of secure protocol is to ensure that when the cryptosystem is applied, the level of security or authentication required by the system is actually attained. Protocols use encryption as a tool to achieve secrecy, authenticity, and integrity. The focus was on the protocol and the security properties to be achieved, and not on the encryption itself. Protocols, which are independent of any specific encryption algorithm, assume only the existence of a particular type of encryption

(conventional or public key), perhaps having a fairly general property (such as cummutativity).

Security ensures the authentication of users of the system to protect the integrity of the information stored in the system (both data and code), as well as the physical resources of the computer system.

The use of protocol in cryptosystem is to ensure that an intruder (human or program) is denied access. There are several reasons for denying access. The most apparent is the need to prevent deliberate breach of an access restriction by a user. The information stored in the system (both data and code), as well as the physical resources of the computer system need to be protected from unauthorized access, malicious destruction, alteration, and accidental introduction of inconsistency. An unprotected resource cannot defend itself against use or misuse by an unauthorized user. A protection-oriented system provides means to distinguish between authorized and unauthorized usage.

While the system administrator may have definite ideas about which user can use which resources, the administrator may not be able to determine a legitimate user from an intruder. Furthermore, users using electronic mail and electronic transfer of funds, as well as huge databases of sensitive information stored in computers with dial-up capabilities, also want their transactions protected. Obviously, a process should be allowed to access only those resources it has been

authorized to access. In addition, at any time, it should be able to access only those resources that it currently requires to complete its task. This requirement, commonly referred to as the need-to-know principle, is useful in limiting the amount of damage a faulty process can cause in the system. In general, a protection system must have the flexibility to enforce a variety of policies that can be declared to it.

The problems of access control are not limited to networks alone, they manifest in time-sharing systems as well and they are solved by having the kernel manage all the resources. Persistent attacks on computer systems, especially in a distributed system by intruders is the single motivating factor responsible for the continued development of new algorithms by cryptographers using complex mathematical processes to withstand attacks by intruder(s) with nearly unlimited resources available to them.

Computer networks are subject to two types of security threats: Passive and Active. Passive threats involve the order to profit from the information or to identify the source of such information. In passive threats, the primary aim is the interception of messages, without detection. Protection against this attack is provided by enciphering transformations. Active threats, on the other hand, involves attempts to alter, destroy, divert message data, or to pose as a network node. In most cases, the intruder may become an active participant on the network, obtaining information with legitimate sites and obtaining free network services. Encryption protects against message modification and injection of false message, by

making it impossible for an eavesdropper to create ciphertext that deciphers into meaningful plaintext.

Encryption alone does not protect against replay. In order to deal with these threats, network managers must control access to the system and to the data in it, as well as protect data in transit. User identification and passwords, data set classification and access protection, and data encryption, all help to secure the computer network from intrusion or protect data from unauthorized viewing or use.

Authentication is a simultaneous process of identification and verification. Identification is the process whereby an entity claims an identity, while verification is the process whereby that claim is validated. Hence, the correctness of an authentication relies primarily on the verification procedure used. We identify three main types of authentication in a computer network system: message content authentication; message origin authentication; and general identity authentication. In this dissertation, we confine our interest to general identity.

Modeling an authentication protocol using a smart card for authentication presents a wide range of problems in the real world. As science and technology keep advancing through the next century, the encryption process used for authentication will become more specialized within a particular domain. On the other hand, many of the most urgent problems we try to solve lie on the boundaries of different

domains. Solving such problems requires integration of different domains. Encryption is a key factor in the design of authentication protocols.

In their discussion of the role of authentication protocols in different levels of computer security, the National Research Council [143] states:

"If one waits until a threat is manifest through a successful attack, then significant damage can be done before an effective countermeasure can be developed and deployed. Therefore, countermeasure engineering must be based on speculation. Efforts may be expended in countering attacks that are never attempted. The need to speculate and to budget resources for countermeasures also implies a need to understand what it is that should be protected, and why; such understanding should drive the choice of a protection strategy."

In our model, we will first present the problem followed by problem analysis, then develop authentication protocols using smart card. Finally, we will show how to use this model for solving different authentication protocols with respect to the use of smart card.

Chapter 7.1 The Design

We take as our model general identity authentication, that is, verifying that a principal's (human or machine) identity is as claimed, to design authentication protocols using smart card. The model is simplified so it can be used as an example in the following discussion but the characteristics of mutual authentication is retained. Designing authentication protocol is a complex process. The task is decomposed into sub-processes: design of protocols to authenticate users, hosts, and processes. A successful design depends not only on the extensive in-depth knowledge that the designer possesses but also the individual's expertise for problem solving. In our model, all authentication procedures involve checking known information about claimed identity against information acquired from the claimant during the identity-verification procedure. Such checking can be based on the following three approaches [175].

- Proof-by-Knowledge. The claimant knows information regarding the claimed identity that can only be known or produced by a principal with that identity. For example, password knowledge is necessary to most login procedures. A proof of knowledge can be conducted by a direct demonstration, like typing in a password, or by an indirect demonstration, such as correctly computing replies to verifier challenges. From a security standpoint direct demonstration is not preferable, because a compromised verifier can record the submitted

knowledge and later impersonate the claimant by presenting the recorded knowledge. This is known as the replay attack.

On the other hand, indirect demonstration can be designed to induce high confidence in the verifier without leaving any clue to how the claimant's replies are computed. For example, Feige, Fiat, and Shamir [72] propose a zero-knowledge protocol for proof of identity. This protocol allows claimant A to prove to verifier B that A knows how to compute replies to challenges without revealing the replies.

- Proof-by-Possession. The claimant produces an item that can only be possessed by a principal with the claimed identity, for example, an ID card. The item has to be unforgeable and safely guarded.

- Proof-by-Property. The verifier directly measures certain claimant properties with such biometrics techniques such as fingerprint and retina print. The measured property has to be distinguishing, that is, unique among all possible principals.

- Proofs by knowledge and possession (and combination thereof) can be applied to all types of authentication needs in a secure computer network system, while proof of property is generally limited to the authentication of human users by a host equipped with specialized measuring instruments.

In an environment where both host and communication compromises can occur, mutual authentication, whereby both communicating principals verify each other's identity, rather than one-way authentication, whereby only one principal verifies the identity of the other principal, is usually required. In computer network environment, authentication is carried out using a protocol involving message exchanges. We refer to these protocols as authentication protocols.

Chapter 8.1 The Development

We present our design of an authentication protocol system using a smart card for mutual authentication. It is intended for use in a university computing environment to solve both academic and administrative computer usage problems. For cooperative problem solving, the card will work on self-service terminals to be located around the campus, thus allowing faculty, students, and administrators to access a central database on campus. For example, students can register on-line, ask for documents, consult their curricular, obtain information from their faculty or from their counselor, or for other uses around campus. Our system is also equipped with a biometrics reader designed to provide facial representation of the card holder whenever required.

The self-service terminals are interactive, thus enabling the students to request changes in their courses, if applicable, or documents. Where there is no need for document to be certified, they are obtained on-line from the terminal's built-in printer. Certificate requests, for example, will be transmitted to a net-worked computer which generates the documents automatically, so that the student may eventually collect it from the faculty's office.

Academic staff members will be provided with a different batch numbered cards with different authorization code that can be used, once properly authenticated, for recording students' examination results (in the university's

database and in the students' cards), as well as for requesting any kind of academic information. Additionally, registered students can be quickly identified and a determination could be made as to whether they are attending classes or taking the correct classes with proper prerequisites.

Administrative staff members will be provided with other batch numbered cards, different from the one issued to the academic staff. Among other possible applications, this card will be used to update database information relating to their specific job area. For example, financial aid and bursar could be tied together using the same access code for their operations, while security and buildings and grounds could be tied together for operational purposes, such as issuing new cards for new students.

Other uses and features that could be implemented in the card include, privilege access to restricted areas, use for lending and handling reserve books and journals from the library, photocopying machines, vending machines, laboratories, buying books from the school bookstore, car parks and parking meters, campus convenience store, post office, medical records, public telephones on campus, pay TV on campus, financial transactions using ATM machines on campus and for electronic purse functions.

Using smart card guarantees the confidentiality of user data (as long as the user does not compromise the card's safety). Most smart card design require an

external keyboard to enter sensitive data such as the user's PIN; hence, a corrupted terminal could secretly store this information without the user's knowledge.

To respond to this frightening reality, we present our design. In addition to storing the encrypted PIN in a secret location in the card's memory, there is a provision for storing a biometrics feature of the user as a compressed file containing a digitized picture of the cardholder in the same secret location in the card's memory. This means that whenever the card is presented for authentication, if required, the user's picture can be checked for visual verification by human operator on a color display unit attached to the main computer screen (see Appendix A). This is to discourage illegal use of the card.

We will further show that the smart card is not only secure but can also play an active role in a computer security system, storing secrets, and providing an easy opportunity to change algorithms without compromise to the system and without need to change the entire system. For example, in 1993, Core State Financial Corp., U.S.; launched an electronic purse system based on their existing network of ATMs, Money Access Service (MAC). MAC cards combine magnetic stripe and microchip, thus allowing both on-line and on-line transactions. The card could be used with any equipment that accepts coins and notes, such as public telephones, photocopying machines, laundry machines, and soft drink dispensers. Other participants eventually joined MAC thus making the card usable in many other automated devices.

Chapter 9.1 Schematic Description - The Smart Card

A schematic diagram of the electronic parts internal to the smart card is shown in Figure 10.1. Within the card, at the center, is a Central Processing Unit (CPU) [2] which is attached to the battery [4], which in turn is connected to a shock detector [6]; the shock detector will interrupt the flow of current if the card is tampered with, thereby blanking the Random Access Memory [14], whenever the shock detector [6] detects any abrupt electrical signal or unusual mechanical shock. The connection from the shock detector [6] to the card's metal contact/decoder [8] and back to the C.P.U. [2] completes the circuit. Also connected to the C.P.U. [2] are a non-volatile Read Only Memory [10] containing the card's operating system; a non-volatile Programmable Read Only [12], which stores both confidential information about the user, also stores the compressed file containing the digitized picture of the cardholder; a Random Access Memory [14] which stores, very temporarily, input data into the system; and a beeper alarm [16], which beeps if there is continuous attempt to gag the system. Appendix A shows our model of smart card reader (SCR) machine.

Chapter 10.1 Types of Authentication Networks

We identify the following major types of authentication networks in a computer network system: Host-to-Host, User-to-Host, and Process-to-Process.

Host-to-Host. Host-to-Host activities require cooperation between hosts. For example, individual hosts exchanges link information for updating their internal topology maps. In remote bootstrapping, a host, when re-initialized, must identify a trustworthy boot server to supply the information (for example, a copy of the operating system) required for correct initialization.

User-to-Host. A user gains access to computer network system by login to a host in the system. In an open-access environment where hosts are located across unsecured areas, a host can be compromised, necessitating mutual authentication between the user and host.

Process-to-Process. We identify two main subclasses:

- Process-to-process communication. Peer process must authenticate each other's identity before private communication can occur.

- **Client-to-server communication. A client's request for access will be granted only when the client's identity is confirmed. A client will give valuable information to a server only after verifying the server's identity.**

As shown later, these two classes of communication authentication relate closely and can be handled by similar protocols.

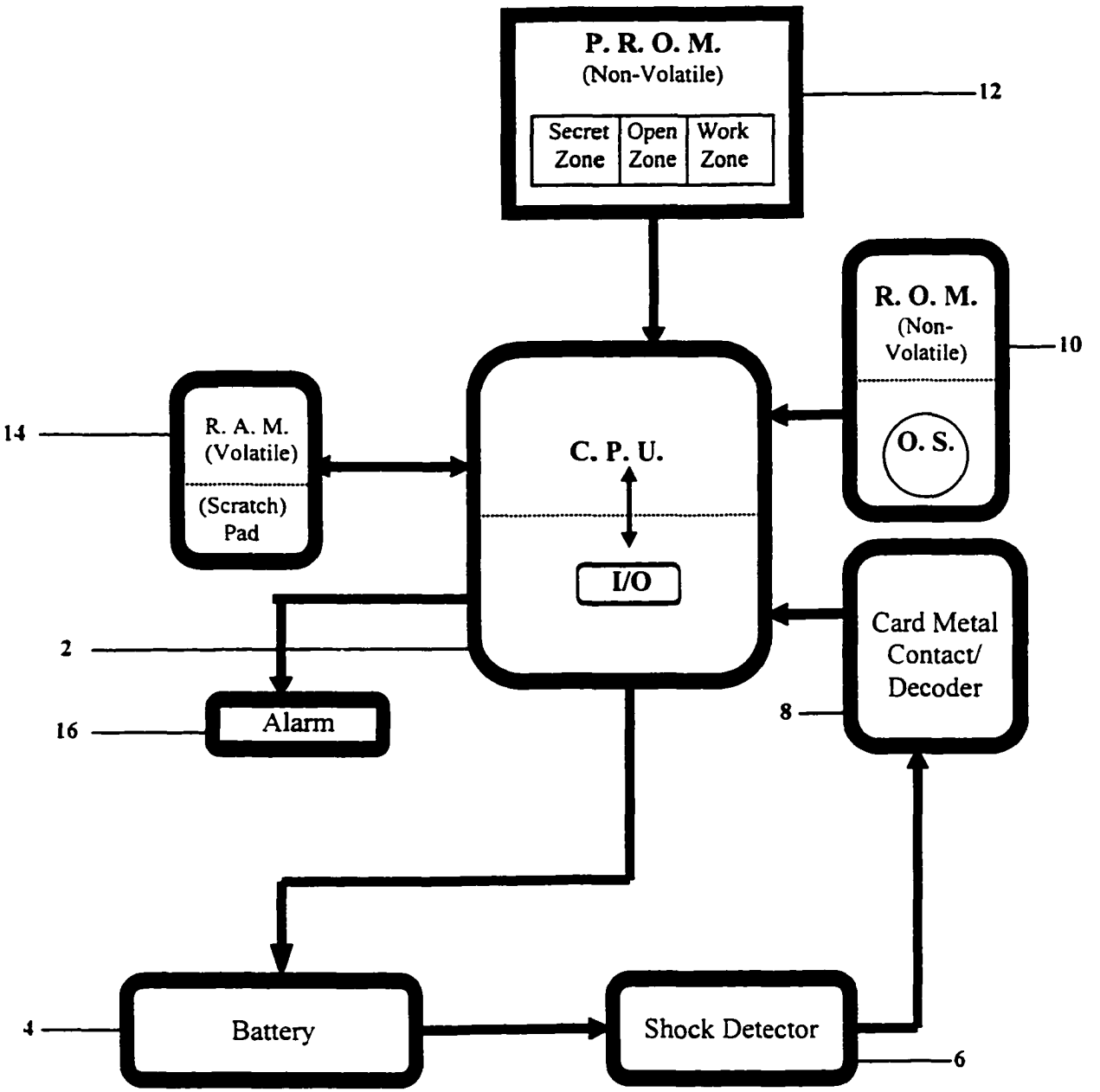


Figure 10.1 A schematic diagram of the electronic parts internal to the smart card

Chapter 11.1 Principles of Authentication Protocols

Authentication in computer networks is always carried out with protocols. A protocol is a defined sequence of communication and computation steps. A communication step transfers messages from one principal (sender) to another (receiver), while a computation step updates a principal's internal state. Although the goal of any authentication is to verify the claimed identity of a principal, specific success or failure states are highly protocol dependent. For example, the success of an authentication during the connection establishment phase of a communication protocol is usually indicated by the distribution of a fresh session key between two mutually authenticated peer processes. On the other hand, in a user login authentication, success usually results in the creation of a login process on behalf of the user.

We present protocols using the following format. A communication step where user A sends a message M to user B is represented as A --> B: M, and a computation step of A is written as A: _____, where "_____" is a specification of the computation step. For example, the typical login protocol between host H and user A is as shown below.

A → H: A

H → A: " Logon please"

A → H: p

H : compute y = f(p)

: retrieve user information (A, f(password$_A$)) from user database

: if y = f(password$_A$) accept; otherwise reject

In the above protocol, f is a one-way function; that is, given y, it is computationally infeasible to find an x such that f(x) = y.

Chapter 12.1 Protocol Examples

Next, we present the key ideas that underline authentication protocol design by presenting various protocol examples. Since authentication protocol paradigms directly use cryptosystems, their basic design principles also follow closely the type of cryptosystem used, that is, a cryptosystem may use both symmetric and asymmetric cryptosystems.

We identify five areas of secure network system design and the related authentication needs. The five areas are

- Host initialization. All process executions take place inside hosts. Some hosts (like workstations) also act as system-entry points by allowing user logins. The overall security of a computer network system is dependent on the security of each host. However, in an open network environment, not all hosts can be physically protected. Hence, resistance to compromise must be built into a host's software to ensure secure operation. This shows the importance of host software integrity. Loading a host that employs remote initialization with the correct host software is essential to its proper functioning. As a matter of fact, one method to compromise a public host is to reboot the host with incorrect initialization information. Authentication can be used to implement secure bootstrapping.

- **User logins. User identity is established at login, and all subsequent user activities are attributed to this identity. All access-control decisions and accounting functions are based on this identity. In essence, correct user identification is essential to the functioning of a secure system. Since any host in an open environment is susceptible to compromise, a user should not engage in any activity with a host without first ascertaining the host's identity. A mutual user-host authentication can achieve the required assurances.**

- **Peer communications. Computer network system can distribute a task over multiple hosts to achieve a higher throughput or more balanced utilization than centralized systems. Correctness of such a distributed task depends on whether peer processes participating in the task can correctly identify each other. Authentication can identify friend or foe.**

- **Client-server interactions. The client-server model provides an attractive example for constructing computer network systems. Servers are willing to provide service only to authorized clients, while clients are interested in dealing only with legitimate servers. Authentication can be used to verify a potential consumer-supplier relationship.**

- **Interdomain communication. Most computer network systems are not centrally owned or administered; for example, a campus-wide computer network system often interconnects individually administered departmental subsystems. Identifying principals across subsystems requires additional authentication process.**

In the malicious environments assumed in our threat models, some basic assumption about the system must be satisfied to achieve a reasonable level of security. We present a set of assumptions (for other possible assumptions see [1], and [114]). Figure 11.1 also depicts these assumptions.

- **Each host hardware W has a unique built-in immutable identity id_W and contains a tamper-proof cryptographic facility that encapsulates the public key k_W and the private key k_W^{-1} of W. The cryptographic facility can communicate with the host reference monitor via a secure channel. Each host that supports user logins also has a smart card reader that communicates with the host reference monitor via a secure channel. Lastly, the host reference monitor has a secure channel to the network interface.**

- **Each legitimate user U is issued a smart card C that has a unique built-in immutable identity id_C. Each smart card C performs encryption and decryption, and encapsulates its private key k_C^{-1}, the public key for the certificate authority k_A, and a pin number Pin_C for its legitimate holder. (The**

pin number is assigned by a card-issuing procedure.) The channel between the smart card and reader is secure. Each smart card has its own display, keypad, and clock.

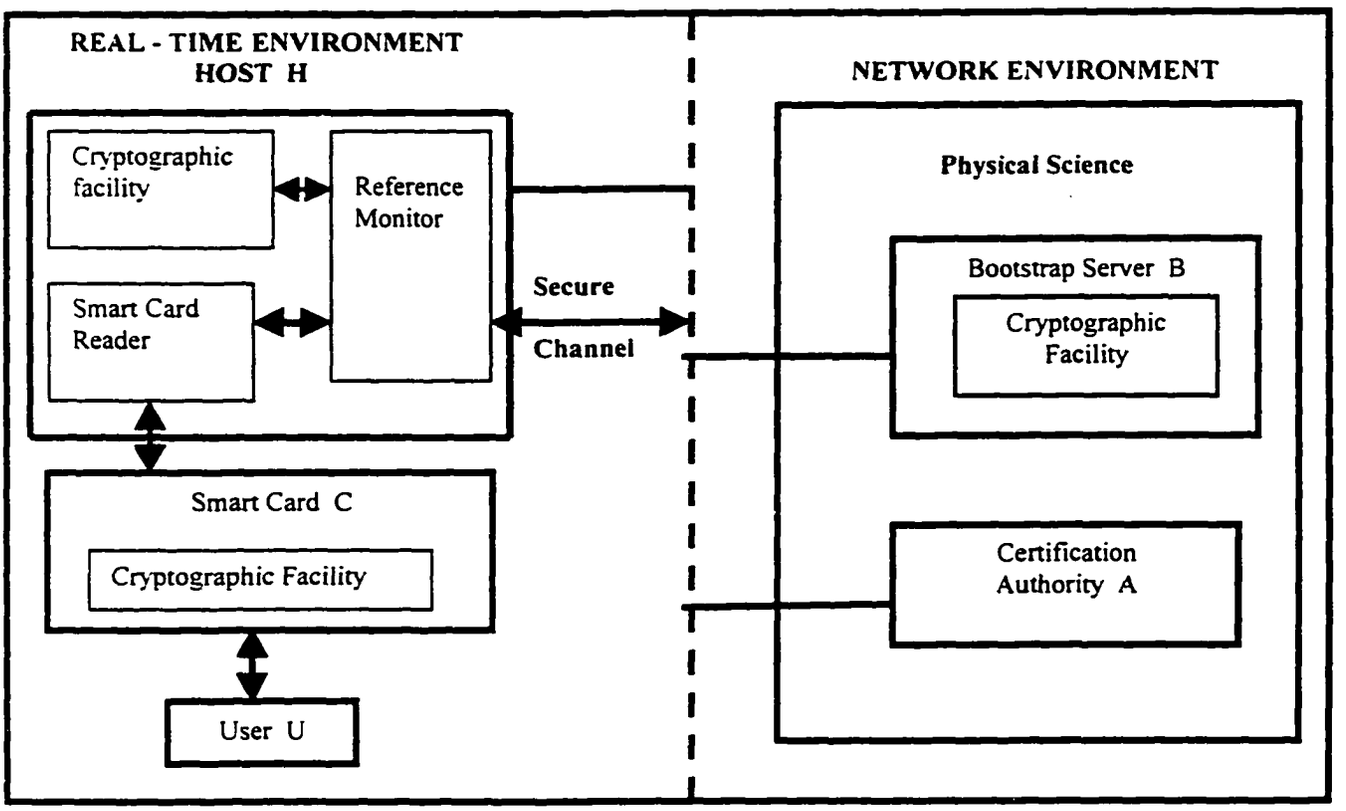


Figure 11.2 Distributed System Environment

- **A physically secure centralized bootstrap server B maintains a database of host information. More precisely, for each host H, it keeps a record (id_H, k_H, k_H^{-1}, id_W, k_W, k_W^{-1}) specifying the unique hardware W that can be initialized as H.**

All database records can be encrypted under a secret master key for added security. B has a public key k_B and a private key k_B^{-1}.

- A physically secure centralized certification authority A maintains a database on all principals. More precisely, for each user U, A keeps a record (U, id_C, k_C), binding U to its smart card C. For each host H, A keeps a record (id_H, id_W), specifying the hardware W that H can run on. Also, for each server S, A keeps a record of its public key certificate (S, k_S). The certification authority A has a public key k_A and a private key k_A^{-1}.

Each assumption is achievable with the smart card technology. Many vendors are now including specialized cryptographic facilities and smart card readers for hosts as options. The use of a smart card or other forms of computational aid is essential to realizing mutual authentication between a host and a user. Unaided human users cannot carry out the intensive computations required by an authentication protocol.

We assume the bootstrap server and certification authority are centralized. Decentralized servers/authorities can also be supported by adding authentication between them. Such authentication can be carried out in a hierarchial approach as suggested in the protocol standard CCITT X.509.

Chapter 13.1 Secure Bootstrap Protocols

- **Secure bootstrapping. The following protocol is initiated when host hardware attempts a remote initialization. This could occur after a voluntary shutdown, a system crash, or a malicious attack by an adversary who attempts to penetrate the host. The secure bootstrap protocol allows a re-initialized host to attain a "safe" state prior to resuming normal operation. In particular, a correctly loaded reference monitor is ready to assume control of the host in this state.**

(1) W → all: "boot," id_w, $\{n_w, id_w\}k_w$

(2) B : retrieve record $(id_H, k_H, k_H^{-1}, id_w, k_w, k_w^{-1})$

For W from database

: recover n_w from $\{n_w, id_w\}k_w$

by decrypting with k_w^{-1}

: generate a random key k

: compute $m = \{n_w, k_A, k_B, k\}k_w$

(3) B → W : m

(4) W : recover (n_w, k_A, k_B) from m by decrypting with k_w^{-1}

(5) W → B : $\{n_w, \text{"ready"}\}k$

(6) B → W : $\{n_w, n_B, id_H, \{k_H^{-1}\}k_w, OS\}k$

(7) W → B : $[\{n_B\}k_H^{-1}]k$

(8) B → W : $\{id_H, id_w, k_H, T\}k_B^{-1}$

(9) H : validate certificate $\{id_H, id_w, k_H, T\}k_B^{-1}$

by encrypting with k_B

Figure 13.1 Secure bootstrap protocol.

For example, suppose that the hosts and the bootstrap server B are on the same broadcast network, allowing the message in step 1 of Figure 13.1 to be received by B. In that protocol, n_W and n_B are nonces, k is a session key. OS is a copy of the operating system, and T is a time stamp.

In step 1, W announces its intention to reboot by broadcasting a boot request. Only B (which has knowledge of k_W^{-1}) can recover the nonce n_W. In step 2, B generates a fresh key k to be used for loading OS. In step 4, W ascertains that m is not a replay by checking the component n_W, since only B could have composed message m. Thus k_A, k_B, and k in m can be safely taken to be the public key of certification authority A, the public key of B, and the session key to be used for loading OS. At this point, B has been authenticated by W.

When the message "ready" encrypted with k is received in step 5, B is certain that the original boot request actually came from W, since only W can decrypt m to retrieve k. Hence, B and W have mutually authenticated each other.

Step 6 is the actual loading of OS and the transferring of host H's private key k_H^{-1}. OS includes its checksum, which should be recomputed by W to detect any OS tampering in transit. W acknowledges receipt of k_H^{-1} and OS by returning the nonce n_B signed with k_H^{-1}. B verifies that the correct n_B is returned. Then in step 8, a license signed by B affirming the binding of host id_H with public key k_H and hardware id_W is sent to W. After receiving the license, W officially "becomes" H, which retains this license as proof of successful bootstrapping and of its own identity. The time stamp field T within the license denotes its expiration date.

If its secrecy is not required, OS can be transferred unencrypted. However, the checksum of OS must be sent in encrypted form.

Chapter 14.1 User-Host Authentication Protocol

- **User-Host Authentication. This function occurs when user U logs onto host H and attempts to log in. The authentication requires smart card C. A successful authentication guarantees host H that U is the legitimate holder of card C and guarantees user U that H is a "safe" host to use. The host is safe if it holds a valid license (which may have been obtained through secure bootstrapping) and possesses knowledge of the private key k_H^{-1}.**

In most systems, the end result of a successful user authentication is the creation of a login process by the host's reference monitor on the user's behalf. The login process is a proxy for the user, and all requests generated by this process are taken as if they are directly made by the user. However, a remote host/server has no way of confirming such proxy status, except to trust the authentication capability and integrity of the local host. Such trust is unacceptable in a potentially malicious environment because a compromised host can simply claim the existence of user login processes to obtain unauthorized services.

This trust requirement can be alleviated if a user explicitly delegates authority to the login host ([114],[1]). The delegation is carried out by having the user's smart card sign a login certificate to the login host upon the successful termination of a user-host authentication protocol. The login certificate asserts the

host's proxy status with respect to the user and can be presented by the host in future authentication exchanges.

Because forgery can occur, the possession of a login certificate should not be taken as sufficient proof of delegation. The host also must demonstrate the knowledge of a private delegation key k_d^{-1} whose public component k_d is named in the certificate. Also, to reduce the potential impact of a host compromise, the login certificate is given only a finite lifetime by including an expiration time stamp.

We present such a user-host authentication protocol in Figure 14.1 We assume that the host holds a valid license $\{id_H, id_W, k_H, T\}k_B^{-1}$, as would be the case if the host has executed the secure bootstrap protocol. In the figure, n_c is a nonce, k_d is the public delegation key whose private counterpart k_d^{-1} is kept secret by the host, and T_c is a time stamp denoting the expiration date of the login certificate.

(1) C → H: id_c, n_c

(2) H → A: $id_c, \{id_H, id_w, k_H, T\}k_B^{-1}$

(3) A : check time stamp of certificate

: if time stamp expired, abort

(4) A → H: $\{id_H, id_w, k_H, T\}k_A^{-1}, \{U, id_c, k_c\}k_A^{-1}$

(5) H : generate new delegation key pair (K_d, K_d^{-1})

(6) H → C: $\{id_H, id_w, k_H, T\}k_A^{-1}, \{U, k_d, n_c\}k_A^{-1}$

(7) C → U: id_H, id_w

(8) U : verify if id_H/id_w is the desired host

: if not, abort

(9) U → C: Pin

(10) C : verify Pin =? Pin_c

: if not equal, abort

(11) C → H: $\{U, id_H, k_d, T_c\}k_c^{-1}$

(12) H : verify correct delegation by decrypting the login Certificate $\{U, id_H, k_d, T_c\}k_c^{-1}$ with k_c

Figure 14.1 User-host authentication protocol.

A user inserts his/her smart card into the host's card reader. The card's identity id_C and a nonce n_C are sent through the card reader to the host in step 1. In step 2, H requests user information associated with id_C from certification authority A. Since the license held by H was signed by B and hence is not decipherable by C, a key translation is requested by H in the same step. (Note that these licenses can be cached by H and need not be requested for every user authentication.) After receiving a reply from A in step 4, H knows both the legitimate holder U of the card C and the public key k_C associated with the smart card. Knowledge of U can be used to enforce the local discretionary control to provide service (or not), while k_C is needed to verify the authenticity of C. In step 5, H generates a new delegation key pair (k_d, k_d^{-1}). H keeps k_d^{-1} private, to be used as proof of a successful delegation from U to H In step 6, H returns the nonce n_C with the public delegation key k_d and a copy of its license to C. In step 7, C retrieves (id_H, id_W), the identity of H,

from the license by decrypting it with k_A. A check ensures that the time stamp in the license has not expired. The identity (id_H, id_W) is then displayed on the card's own screen. To proceed, the user enters the assigned pin number on the card's keypad. In step 10, the pin number is compared with the one stored in the card. If they are equal, C signs a login certificate binding the user U with the host id_H and the public delegation key k_d. This is sent to H in step 11. The host (and others) can verify the validity of the login certificate using k_C, the card's public key.

When user U logs out, the host erases its copy of the private delegation key k_d^{-1} to void the delegation from U. If H is compromised after the delegation, the effect of the login certificate is limited by its lifetime, T_C.

As a further alternative, we also propose the following User-host authentication protocol using Blum integers:

Step 1: User (U) inserts card (C) into host's card reader.

The card's identity id_c is sent through the card reader to the host.

Step 2: Hardware (H) requests user info-associated with id_c from certification authority A.

Step 3: After receiving a reply from A, H knows both the legitimate user (U) of the card (C) and the public key k_c associated with the smart card.

Note: k_c is needed to verify authenticity of C.

Step 4: H returns a copy of its license to C with the public delegation key k_d.

Step 5: C retrieves (id_H, id_W), the identity of H, from the license by decrypting it with k_A.

Step 6: U verifies the identity (id_H, id_W)

Using Blum integers to sign a login certificate in this protocol:

Step 7: User (U) enters the assigned pin number on the card reader's keypad and also generates a Blum integer, n, a random x relative prime to n, $x_0 = x^2 \bmod n$, and $x_1 = x_0^2 \bmod n$.

Step 8: U sends n and x_1 to H.

Step 9: The pin number and the generated Blum integer is compared with the one stored in the card.

Step 10: If equal, H accredits x_0, and C signs a login certification binding the user U with the host id_H and public delegation key k_d.

Step 11: U sends x to H along with the signed certificate.

Step 12: H checks that n is a Blum integer, and verifies that $x_0 = x^2 \bmod n$ and $x_1 = x_0^2 \bmod n$. If all checks out, host (H) and others can verify the validity of the login certificate using the card's public key k_c.

Figure 14.1b User-host authentication protocol using Blum integers

Note: (i) User U would have to execute some zero-knowledge protocol to convince H that n is a Blum integer.

(ii) It is important that n is a Blum integer. Otherwise, a hacker may be able to find an x'_0 such that $x'_0{}^2 \bmod n = x_0{}^2 \bmod n = x_0{}^1$, where x'_0 is also a quadratic residue. If x_0 and x'_0 are complements of each other, the security of the protocol would be compromised.

(iii) When user U logs out, H erases its copy of the private delegation key k_d^{-1} to void the delegation from U.

(iv) If H is compromised after the delegation, the effect of the login certificate is limited by its lifetime, T_c.

Chapter 15.1 Peer-Peer Authentication

- **Peer-Peer Authentication.** This type of mutual authentication and cryptographic parameters negotiation is performed in the connection-establishment phase of a secure connection-oriented protocol.

The protocol in Figure 15.1 mutually authenticates peers P and Q, and establishes a new session key for their future communication (n_P and n_Q are nonces, while k is a fresh session key).

(1) $P \rightarrow A$: P, Q

(2) $A \rightarrow P$: $\{Q,k_Q\}k_A^{-1}$

(3) $P \rightarrow Q$: $\{n_p, P\}k_Q$

(4) $Q \rightarrow A$: $Q, P, \{n_p\}k_A$

(5) $A \rightarrow Q$: $\{P, k_p\}k_A^{-1}, [\{n_p, k, Q\}\ k_A^{-1}]k_Q$

(6) $Q \rightarrow P$: $[\{n_p, k, Q\}\ k_A^{-1}, n_Q]\ k_p$

(7) $P \rightarrow Q$: $\{n_Q\}k$

Figure 15.1 Peer-peer authentication protocol.

In step 1, P informs A of its intention to establish a secure connection with Q. In step 2, A returns to P a copy of Q's public key certificate. In step 3, P informs Q of its desire to communicate, and sends a nonce n_P. In step 4, Q asks A for P's

public key certificate and requests a session key at the same time. In order for Q to subsequently prove to P that the session key k is actually from A (not Q's own creation), A sends a signed statement stating that the key k is actually from A (not Q's own creation), A sends a signed statement containing the key k, n_P, and Q's name. This basically says that k is a key generated by A on behalf of Q's request identified by n_P. The binding of k and n_P assures P that k is fresh. In step 6, A's signed copy of (n_P, k, Q) is relayed to P together with a nonce n_Q generated by Q. P's knowledge of the new session key k is indicated to Q by the receipt of n_Q in step 7.

Chapter 16.1 Client-Server Authentication

- **Client-Server Authentication. Since both clients and servers are implemented as processes, the basic protocol for peer-peer authentication can be applied here as well. However, several issues peculiar to client-server interactions need to be addressed.**

In a general-purpose computer network environment, new services (hence servers) are made available dynamically. Thus, instead of informing clients of every service available, most implementations use a service broker to keep track of and direct clients to appropriate providers. A client first contacts the service broker by using a purchase protocol that performs the necessary mutual authentication prior to the granting of a ticket. The client later uses the ticket to redeem services from the actual server by using a redemption protocol.

Authentication performed by the purchase protocol proceeds the same way as the protocol for peer-to-peer authentication, while in the redemption protocol authentication is based upon possession of a ticket and knowledge of some information recorded in the ticket. Such a ticket contains the names of the client and server, a key, and a time stamp to indicate lifetime (similar to a login certificate). A ticket can be used only between the specified client and server. A

prime example of this approach is the Kerberos authentication system, which we later discuss.

Another special issue of client-server authentication is proxy authentication. To satisfy a client's request, a server often needs to access other servers on behalf of the client. For example, a database server, upon accepting a query from a client, may need to access the file server to retrieve certain information on the client's behalf. A straightforward solution requires the file server to directly authenticate the client. However, this may not be feasible. In a long chain of service requests, the client may not be aware of a request made by a server in the chain and hence may not be in a position to perform the required authentication. An alternative is to extend the concept of delegation previously used in user-host authentication [82]. A client can forward a signed delegation certificate affirming the delegation of its rights to a server along with its service request. The server is allowed to delegate to another server by signing its own delegation certificate as well as by relaying the client's certificate. In general, for a service request involving a sequence of servers, delegation can be propagated to the final server through intermediate ones, forming a delegation chain.

Various refinements can extend the scheme. A form of restricted delegation can be carried out by explicitly specifying a set of rights and/or objects in a delegation certificate.

Chapter 17.1 Interdomain Authentication

- **Interdomain Authentication.** We have assumed a centralized certificate authority trusted by all principals. However, a realistic distributed system is often composed of subsystems independently administered by different authorities. We use the term domain to refer to such a subsystem. Each domain D maintains its own certification authority A_D that has jurisdiction over all principals within the domain. Intradomain authentication refers to an exchange between two principals belonging to the same domain, whereas interdomain authentication refers to an exchange that involves two principals belonging to different domains.

Using the previously described protocols, A_D is sufficient for all intradomain authentications for each domain D. However, a certification authority has no way of verifying a request from a remote principal, even if the request is certified by a remote certification authority. Hence, additional mechanisms are required for interdomain authentication.

To allow this type of authentication, two issues need to be addressed: naming and trust. Naming ensures that principals are uniquely identifiable across domains, so that each authentication request can be attributed to a unique principal. A global naming system spanning all domains can be used to provide

globally unique names to principals. A good example of this is the Domain Name System used on Internet.

Trust refers to the willingness of a local certification authority to accept a certification made by a remote authority regarding a remote principal. Such trust relationships must be explicitly established between domains, which can be achieved by

- sharing an interdomain key between certification authorities that are willing to trust each other,

- installing the public keys of all trusted remote authorities in a local certification authority's database, and

- introducing an interdomain authority for authenticating domain-level authorities.

A hierarchical organization corresponding to that of the naming system can generally be imposed on the certification authorities. In this case, an authentication exchange between two principals P and Q involves multiple certification authorities on a path in the hierarchical organization between P and Q [83]. The path is referred to as a certification path.

Chapter 18.1 Applications of Authentication Protocols to Kerberos

We study one authentication service: Kerberos. It addresses client-server authentication needs. Its services are generally available to an application program through a programming interface. Kerberos uses a symmetric cryptosystem.

- Kerberos. This system was designed for Project Athena at the Massachusetts Institute of Technology [184]. The project goal is to create an educational computing environment based on high-performance workstations, high-speed networking, and servers of various types. Researchers envisioned a large-scale (10,000 workstations to 1,000 servers) open-network computing environment in which individual workstations can be privately owned and operated. Therefore, a workstation cannot be trusted to identify its users correctly to network services. Kerberos is not a complete authentication framework required for secure distributed computing in general; it only addresses issues of client-server interactions.

We limit our discussion to the Kerberos authentication protocols and omit various administrative issues. The design is based on using a symmetric cryptosystem with trusted third-party authentication servers. It is a refinement of ideas presented in Needham and Schroeder [144]. The basic components include Kerberos authentication and ticket-granting servers (TGSs). A database contains

information on each principal. It stores a copy of each principal's key that is shared with Kerberos. For user U, its shared key k_U is computed from its password $password_U$; specifically, $k_U = f(password_U)$ for some one-way function f. Kerberos servers and TGSs read the database in the course of authentication.

Kerberos uses two main protocols. The credential-initialization protocol authenticates user logins and installs initial tickets at the login host. A client uses the client-server authentication protocol to request services from a server.

The credential-initialization protocol uses Kerberos servers. Let U be a user who attempts to log into host H, and f be the one-way function for computing k_U from U's password. The protocol is specified in Figure 18.1 (here, Kerberos refers to the server):

U → H	:	U
H → Kerberos:		U, TGS
Kerberos	:	retrieve k_U and k_{TGS} from database
	:	generate a new session key k
	:	create ticket-granting ticket
		$tick_{TGS} = \{U, TGS, k, T, L\}k_{TGS}$
Kerberos → H :		$\{TGS, k, T, L, tick_{TGS}\}k_U$
H → U	:	"Password?"
U → H	:	passwd
H	:	compute $l = f$(passwd)
	:	recover k, $tick_{TGS}$ by decrypting

$\{TGS, k, T, L, tick_{TGS}\}k_U$ with *l*

: if decryption fails, abort login

: otherwise retain $tick_{TGS}$ and k

: erase passwd from memory

Figure 18.1 Kerberos credential-initialization protocol.

If passwd is not the valid password of U, *l* would not be identical to k_U, and decryption in the last step would fail. (In practice, *f* may not be one-to-one. It suffices to require that given two distinct elements x and y, the probability of f(x) being equal to f(y) is negligible.) Upon successful authentication, the host obtains a new session key and a copy of the ticket-granting ticket.

$$tick_{TGS} = \{U, TGS, k, T, L\}_{KTGS}$$

where T is a time stamp and L is the ticket's lifetime. The ticket-granting ticket is used to request server tickets from a TGS; note that $tick_{TGS}$ is encrypted with k_{TGS}, the shared key between TGS and Kerberos.

Since a ticket is susceptible to interception or copying, it does not constitute sufficient proof of identity. Therefore, a principal presenting a ticket must also demonstrate knowledge of the session key k named in the ticket. An authenticator

(to be described) provides the demonstration. Figure 18.2 shows the protocol for client C to request network service from server S (T1 and T2 are time stamps).

(1) C → TGS : S, $tick_{TGS}$, {C, T1}k

(2) TGS : recover k from $tick_{TGS}$ by decrypting with k_{TGS}

: recover T1 from {C, T1}k by decrypting with k

: check timeliness of T1 with respect to local clock

: create server ticket ticks = $\{C, S, k', T', L'\}_{ks}$

(3) TGS → C: {S, k', T', L', ticks}k

(4) C : recover k', ticks by decrypting with k

(5) C → S : ticks, {C, T2}k'

(6) S : recover k' from ticks by decrypting with k_s

: recover T2 from {C, T2}k by decrypting with k'

: check timeliness of T2 with respect to local clock

(7) S → C : {T2 + 1}k'

Figure 18.2 Kerberos client-server authentication protocol.

In step 1, client C presents its ticket-granting $tick_{TGS}$ to TGS to request a ticket for server S. (Note that each client process associates with the unique user who created the process. It inherits the user ID and the ticket-granting ticket issued to the user during login.) C's knowledge of k is demonstrated using the authenticator $\{C,T1)_k$. In step 2, TGS decrypts $tick_{TGS}$, recovers k, and uses it to

verify the authenticator. If both step 2 decryptions are successful and T1 is timely, TGS creates a ticket $tick_S$ for server S and returns it to C. Holding $tick_S$, C repeats the authentication sequence with S. Thus, in step 5, C presents S with $tick_S$ and a new authenticator. In step 6, S performs verifications similar to those performed by TGS in step 2. Finally, step 7 assures C of the server's identity. Note that this protocol requires "loosely synchronized" local clocks for the verification of time stamps.

Kerberos can also be used for authentication across administrative or organizational domains. Each domain is called a realm. Each user belongs to a realm identified by a field in the user's ID. Services registered in a realm will accept only tickets issued by an authentication server for that realm.

An interrealm key supports cross-realm authentication. The TGS of one realm can be registered as a principal in another by using the shared interrealm key. A user can thus obtain a ticket-granting ticket for contacting a remote TGS from its local TGS. When the ticket-granting ticket is presented to the remote TGS, which uses the appropriate interrealm key to ascertain that the ticket was issued by the user's local TGS. An authentication path spanning multiple intermediate realms is possible.

Kerberos is an evolving system on its fifth version. Bellovin and Merritt [18] discuss limitations of previous versions, some of which have been remedied.

Chapter 19.1 Applications of Authentication Protocols to other Scenarios

After the model of the smart card problem is developed, the model can be used to simulate different scenarios of problem solving strategies. For example, the card can be used in a very wide range of applications that replace cash to make purchases or pay for services. Such areas of use would include; road toll payments, health care cards, phone cards, ID cards, access keys, electronic purses, GSM mobile phone SIMS, pay TV subscriber cards, groceries, dry cleaning, and much more.

Since data exchanges take place in the card's microchip memory, it is relatively quick to use. To use it to make a purchase, the users dip the card into the terminal at the store's cash register. The salesperson keys in the amount of purchase, the customer presses "yes" at terminal to confirm sale. The computer microchip in the card performs the computation and debits the customer's account accordingly.

Smart cards designed for specific purposes are more simpler than the multipurpose card. For example, visual readouts may not be necessary, rather, output could be indicated only by a brief intermittent buzzing alarm sound. More so, smart cards designed for use in highly secured area may be manufactured with

extra safeguards against tampering or fraud. For example, each such card might be given special features which will conclude a variable time-limit, which causes the device to blank its RAM memory automatically if it has not received expected input for too long a time or a special secret sequence of keys, which produces a particular response.

Chapter 20.1 Advantages of our design

A common type of fraud against credit cards and automatic banking machine cards involve "personal identification numbers (PIN)," or password modification. For example, when a magnetic-stripe card is used in an automatic banking machine, the machine asks the cardholder for a PIN. It then reads the stored correct PIN directly from the card and compares it with the given PIN. At some point during the transaction the correct PIN must be brought into the banking machine's working memory. Any "hacker" or intruder who has the resources to monitor that memory can learn the cardholder's PIN. In this manner, magnetic-stripe cards are entirely passive; that is, they are merely medium for storing information, and are vulnerable to many forms of fraud and abuse.

Smart cards however, are not susceptible to this kind of attack(s). Smart cards have two important properties that make them invulnerable [122]. First, a smart card has a non-volatile, programmable read-only memory; that is, a memory into which information can be placed after the card has been issued and will remember any such information even when it is not connected to a power source. Second, each smart card contains its own central processing unit (CPU) - essentially a small computer - which controls all the interactions between the memory and the various external devices that can read the card and enter data into it.

In our smart card authentication system design, through the central processing unit in the card's memory, the card itself can examine any supplied password and compare it with the correct password, stored in a secret location in the card's memory. For example, the card can engage the terminal in a sequence of questions and answers session to verify the validity of information stored on the card and the identity of the card-reading terminal without going on-line to centralized databases to check this information. A card with such an encryption algorithm might be able to convince a local terminal that its owner is legitimate and have enough resources to pay for a transaction without revealing the actual balance or account number.

The card never has to reveal the correct password to any outside system. In fact, even the company that issues the card does not need to know the correct password. Depending on the importance of the information involved, security might rely on a personal identification number (PIN) such as those used with automated teller machines, a mid-range encipherment system, such as Data Encryption Standard (DES), or a highly secure public-key scheme, such as Blum, Blum, and Shub (BBS) generator for key generation.

The smart card protects the user's privacy, and the contents of the transaction is not exposed to an eavesdropper either. For example, the central processing unit (CPU) in the card through its memory, can update data on the card

after each transaction. This ability makes it possible to authorize many transactions without reference to the central database. The card can, for instance, keep track of the balances remaining and authorize transactions within those balances. This feature protects against double or overspending.

Furthermore, the microchips are more resistant to tampering than magnetic stripes, which can be read and written on with the readily available equipment. For example, the central processing unit (CPU) and the memory architecture are designed in such a way that certain parts of the card's memory are physically or logically inaccessible to anyone but the card's issuing agent: the central processing unit will not obey an instruction from anyone else (user or "hacker") to read or change those part of the memory. As a matter of fact, the card can be programmed to self-destruct when an unauthorized party probes it [130]. This design enhancement can help to foil an "IP-spoofing" (Internet Protocol spoofing) attack.

Figure 20.1 shows the block schematics of the programmable, read-only memory (PROM), which gives the smart card much of its flexibility and utility; for example, the secret zone stores other information such as the card's serial number and a sequence of alphanumeric characters chosen by the card's issuer, that an be used to determine whether the card is legitimate or not. The open zone, might store the card holder's name, address, telephone number, and account number. The open zone can also be read by an card-reading machine, but it cannot be altered; the

card's central processing unit will not obey instructions to change any information in the open zone.

Whenever the card is used at the terminal to make a transaction, such information as the name and address of location, the session time (duration), and date is stored in a third zone of the memory, called the working zone. Information can be written there only under certain conditions (for example, when the card is in legitimate card-reading/writing machine) and can be read and written only with the card holder's permission. A cardholder can buy a separate card-reading machine, (Figure 20.2) which when connected to a home computer, a television set or a printer, displays a complete record of all transactions made with the card and hence provide an audit trail.

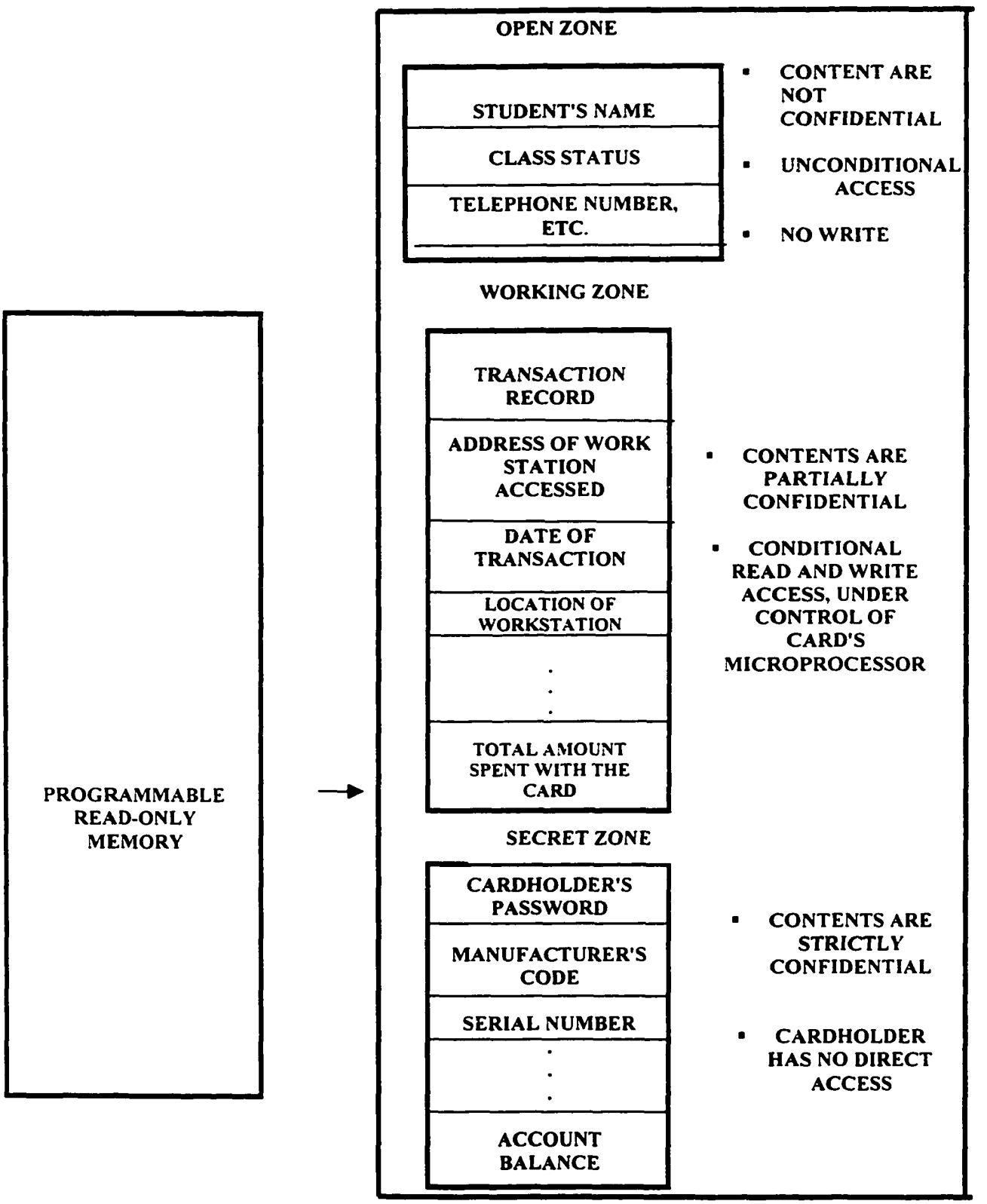


Figure 20.1 Schematic view of the smart card PROM.

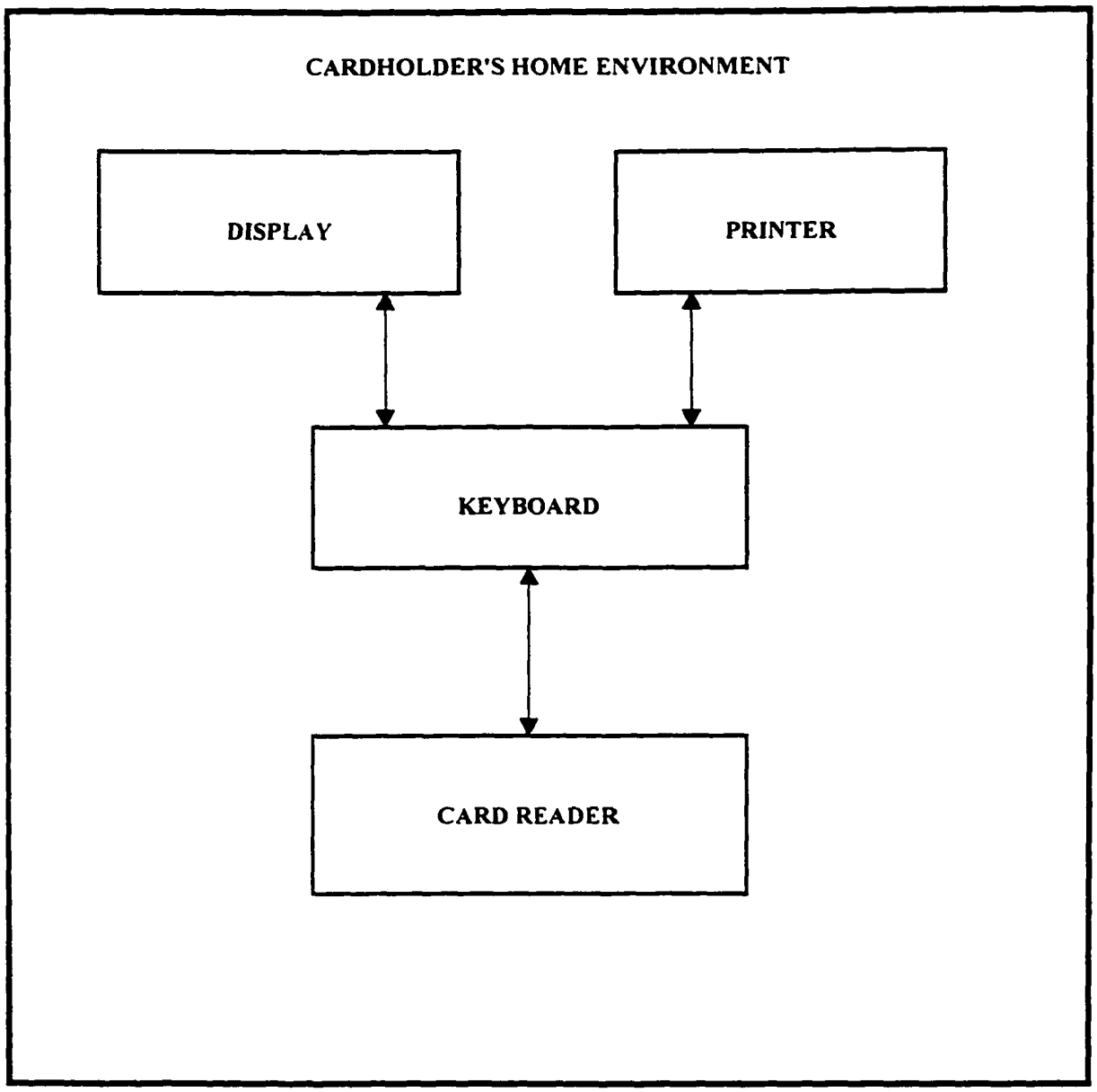


Figure 20.2 Operation of I/O device for home environment.

Chapter 21.1 Summary

The analysis of our smart card authentication protocol is based on six properties; resolution, leakage, privacy, security, simplicity, and remoteness. We have shown how the smart card allows information to be stored in the card rather than on a computer at some remote location. This feature is an added advantage for security and allows encryption techniques to be used in the card. Smart cards are more durable than traditional magnetic stripe cards because the microchip in the card cannot be affected by magnetic fields or scratches. Smart card is a unique product and is becoming the natural choice for many applications besides the classic financial cards.

Although its only a recent development in microprocessor technology, continued research and spawning of smart card applications is producing a snowballing effect. Smart cards are here to stay, and their impact on the everyday habits of the population is increasing. For example, in October, 1997, Citibank, Chase Manhattan, Visa, and MasterCard launched a six-month pilot program in New York City. The New York pilot program is part of a worldwide push to convince consumers to forego cash for smart cards. The system is already widely use in Canada.

Finally, we hope this smart card authentication protocol design would serve as a benchmark for further research relating to the new CUNY smart card.

Part IV

22.1 Conclusions and Open Areas for Future Research

Security vulnerabilities are ever present in computer systems, and are intrinsically impossible to avoid completely. Some of the main concerns of data security in banking and other commercial information systems are those of authentication. We have reviewed some problems concerning the security and performance of magnetic stripe cards. A tremendous advantage of our Smart card design is that the Smart card's built in security features are the best alternative for defeating these problems. Although magnetic stripe cards are inexpensive, even with added features such as holograms, many institutions are now using Smart cards.

The security provided by smart card surpassed that of magnetic stripe card. For example, as the number of card applications increases for magnetic stripe card, so does the number of cards in customers' wallet, whereas, a Smart card can be used as a multifunction card. The Smart card deals with security levels that are unattainable with conventional magnetic stripe cards. While many European countries have been using Smart cards for several years, the U.S. and Japan have been slow to react. For example, in U.S., the Smart card was tried on large scale, for the first time, at the 1996 Summer Olympics in Atlanta. Also, the U.S. Marine

Corp. is experimenting with some chip cards to replace the conventional "dog-tags". Today, however, there are many different Smart card projects throughout the U.S. government.

To respond to the increasing prevalent fraud with magnetic stripe cards, Smart cards are the solution. The Financial institutions could integrate Smart cards into their current systems through hybrid cards featuring magnetic stripe and microchip, as in the Message Authentication Code (MAC) network in the U.S. This would not only allow the use of current Automatic Teller Machines (ATM) and Electronic Point of sales (EPOSs), but would also complement these devices by offering additional security among other services and authenticating off-line users through secure PINs or biometric tests.

The financial institutions should make the Smart card services universal for regular clients and not for only VIPs as is the case in France. Because of the global nature of commerce in the world today, developing countries, with no magnetic stripe card tradition would have no problem implementing the new technology. Already, some European countries have also adopted financial Smart cards. While there will be no abrupt replacement of magnetic stripe cards by smart cards, gradual adaptation of smart cards into financial services is inevitable.

To implement smart cards in non-financial areas such as telephones, health, transport, and so on, should not pose a problem. For example, several

financial applications may be merged into the same card, together with prepaid services, thus leading to the concept of user's card, as opposed to an issuer's card. In this context, card issuers would simply become service providers because their influence would be restricted on the card's content at the user's pleasure.

For an applications requiring extensive memory, WORM (Write Once Read Many) or Erasable Magneto-Optical memory can be used. Other features such as bar codes or any other desired options, may also be included. In either situation, the stored information should be controlled through data encryption by the microprocessor.

Finally, Smart cards will be the natural choice for many applications besides the conventional financial cards in the very near future. Because their range seldom extends beyond 1 million cards, their applications are less noticeable. Currently, France leads the world in the use of Smart cards, with other European countries not far behind. Infrastructure and expertise barriers that have slowed the growth of Smart card adoption would soon become a thing of the past due to technology improvements and reduction of overhead prices. Smart cards are here to stay, and their impact on the everyday habits of the population is steadily increasing.

Appendix A: Implementation of the Smart Card Reader Machine

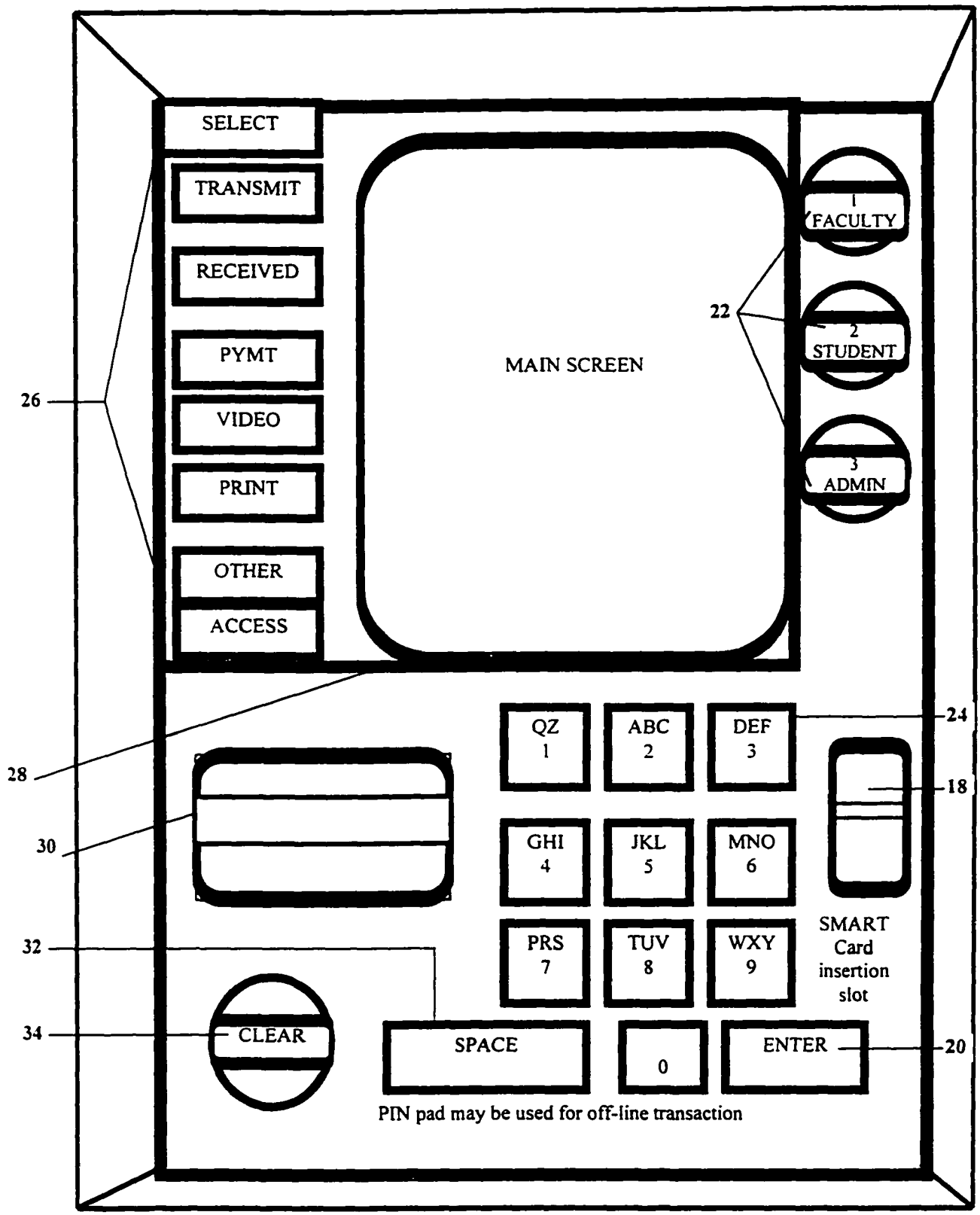


Fig. A.1 shows the proposed Smart Card Reader (SCR) machine.

Fig. A.1 shows the proposed Smart Card Reader (SCR) machine. At the top left are the Menu selection keys [26] which provide possible selection options to the user after being properly authenticated by the system. At the center, next to the Menu selection keys [26], is the display screen [28] which provides visual representation of the transaction process to the user. At the top right are the Status Mode selection keys [22] from which the user is required to select the appropriate Status Mode before initiating a transaction. Below the Menu selection keys [26] is the on-board printer [30] which will provide printed output for the user if one is requested. Next to the printer [30] is the alphanumeric key pad [24] from which the user can enter the PIN. On the right is the Smart Card insertion slot [18] through which the user inserts the card before any processing can take place. At the bottom left is the "CLEAR" key [34], followed by the "SPACE BAR" key [32], and the "ENTER" key [20] used to confirm any entry into the system.

General Mode of Operation of the Smart Card Reader (SCR)

In its preferred representation, the Smart Card Reader (Fig. A.1), henceforth referred to as "SCR" looks like a typical Automatic Teller Machine (ATM). However, unlike the ATM, the SCR can operate both in "On-line" and "Off-line" states. Furthermore, its use goes beyond the realms of "retail banking" as is the case with ATM. More importantly, it has the capability to decode the stored compressed file containing the digitized picture of the cardholder for further identification if needed.

When the user inserts a card and depresses the "ENTER" key, the SCR displays a menu of possible mode options from which the user is to select from. The user selects one of the modes, namely, "Faculty = 1", "Student = 2", or "Administrator = 3" by depressing the corresponding number key.

Once the mode number has been selected, the SCR checks for confirmation from the card, thereby ensuring that the encrypted mode status stored in the card matches the selected mode number by the user from the SCR. If there is a match, the session status mode and session option will be displayed at the bottom of the screen throughout the transaction session.

If there is no match for any reason, that is, if the selected mode number does not match with the stored status mode on the card itself, the card communicates a mismatch status to the SCR. The SCR then displays a rejection message to the user on its screen and requests another try from the user. Each time there is a rejection, the card keeps a record of the number of rejection in its secret memory. After three consecutive mismatches (rejections), the SCR displays on its screen the following messages: "MAXIMUM NUMBER OF TRIES, PLEASE CONTACT THE ACADEMIC COMPUTING CENTER."

Unless the user contacts the Administrative/Academic Computing Center, the card would automatically be temporarily put out of commission. This disabling process can last anywhere from a few minutes to several hours before the user is

allowed access to the system again. Optionally, if after three consecutive tries and rejections, a user persistently continues to try to gain unauthorized access, this action would trigger the beeper alarm circuit, and the alarm would go off.

If the correct mode has been selected, the card, through the SCR asks for additional information concerning the user. Options would include asking for the user PIN or in other instances, the visual (VIDEO) identification of the user by human operator(s). For the PIN option, the PIN is entered from the alphanumeric key pad on the SCR. For the VIDEO option, the video key on the SCR is depressed to reveal the compressed digitized picture of the user .

If there is no match between the selected mode and the user's PIN, the SCR displays: "NO SUCH PIN, PLEASE TRY AGAIN." Once again, the user is allowed a maximum of three (3) attempts, after which the card is temporarily disabled. If there is a match, the SCR will display a "Menu" of transaction options for the user to select from. After a selection has been made from the Menu list, the system further asks the user for the transaction mode in order to determine whether the transaction is going to be "On-line" or "Off-line." The mode chosen by the user will be displayed at the bottom of the screen throughout the transaction session, so that users will not be misled. In every case, the communication exchanges between the user and the system are secure from eavesdroppers on the line at any time.

When the transaction is completed, the card, through the SCR, queries the user in order to determine if a printed output will be required. If the user's response is "yes", a printed output is generated for the user. If the response is "no", the SCR displays the result(s) of the transaction on the screen.

Thereafter, the user is further asked whether that is the last transaction or not. If the user's response is "yes" indicating last transaction, the system will stop processing immediately. On the other hand, if the user indicates willingness to do another transaction by answering "no", the system will immediately return to the Menu list and display possible options from which the user is expected to make a choice and go through the transaction process again.

After every transaction, the Smart card's RAM memory is blanked as soon as the transaction is completed. The memory is also blanked when a mechanical shock is detected, when any abrupt electronic signal is received, and when the battery case is opened.

The flowchart is shown in Figures A.2 through A.5 (see end of text). Numbers in brackets refers to points in the flowchart. In Figure A.2, we present a Dbase program which encodes the flowchart of the smart card reader. The program starts when the card is inserted into the SCR's card slot and "ENTER" is depressed [36]; the SCR responds by displaying its function mode menu [38]. The SCR then waits for a mode to be selected by the user by depressing one of the

function mode keys (all other inputs are ignored). Depending on which key is depressed [40], the "Faculty" mode [42], the "Student" mode [44] or the "Administrative" mode [46] is set and indicated on the screen. The flow now proceeds through point A [48] to Figure A.3.

At the top of Fig. A.3 the program begins at point A [48] and checks the number of possible tries against the digit stored in variable N [50]. If N>3 the program STOPS [88]. If N<3, the card, through the SCR, asks the user to enter his/her PIN [52]. The digit stored in variable N [50] is incremented by a value of 1 [54]. Next, the card compares the PIN to "BLANK" [56]. If the result = "Yes", the program proceeds to "STOP" [88]. If the result = "No", the card, through the SCR compares "MODE" to "PIN" [58]. If MODE = PIN the flow now proceeds through point B [60] to Figure A.4.

If MODE is not equal to PIN, the digit stored in variable N = 3 [62]. If N = 3, the program displays "MAXIMUM NO. OF TRIES. PLEASE CALL THE COMPUTER CENTER" and sounds the beeper alarm [64], then proceeds to "STOP" [88]. If N is not equal to 3, the program displays "NO SUCH PIN, PLEASE TRY AGAIN" [66]. The flow returns to [50].

At the top of Figure A.4 the program begins at point B [60], the program display "SELECT OPTIONS FROM MENU LIST" [68]. Whatever selection is made from the menu list, the SCR asks [70] for session type. If session = "ON-line", the program displays a reminder on the screen indicating that the session is in "On-

line" mode [72] then proceeds to the unifying point above [76]. If session = "Off-line", the program displays a reminder on the screen indicating that the session is in "Off-line" mode [74], then proceeds to the unifying point above [76].

At [76], the "PRINT" flag is checked and the program asks for output type. If a printed output is requested [78], a hard copy is generated for the user. If no printed output is required [80], a soft copy is generated for the user. The flow now proceeds through point C [82] to Figure A.5.

At the top of Figure A.5, the program begins at point C [82] and checks [84] if this is the last transaction. If this is not the last transaction, the program proceeds to point D [86]. If it is the last transaction, the program proceeds to "STOP" [88].

This completes the description of the Smart card and SCR's flowchart. The following listing is for a working program. The program duplicates the algorithm of the flowchart. We conclude by taking advantage of the above listing to restate the fact that the magnetic stripe cards is not suitable to function as a Smart card. With the Smart card, information is stored on the card itself rather than on a computer somewhere. This is an added advantage for security because it allows encryption techniques to be used in the card. Smart cards are more durable than traditional magnetic stripe cards as the chip cannot be affected by magnetic field or scratches like the magnetic stripe can.

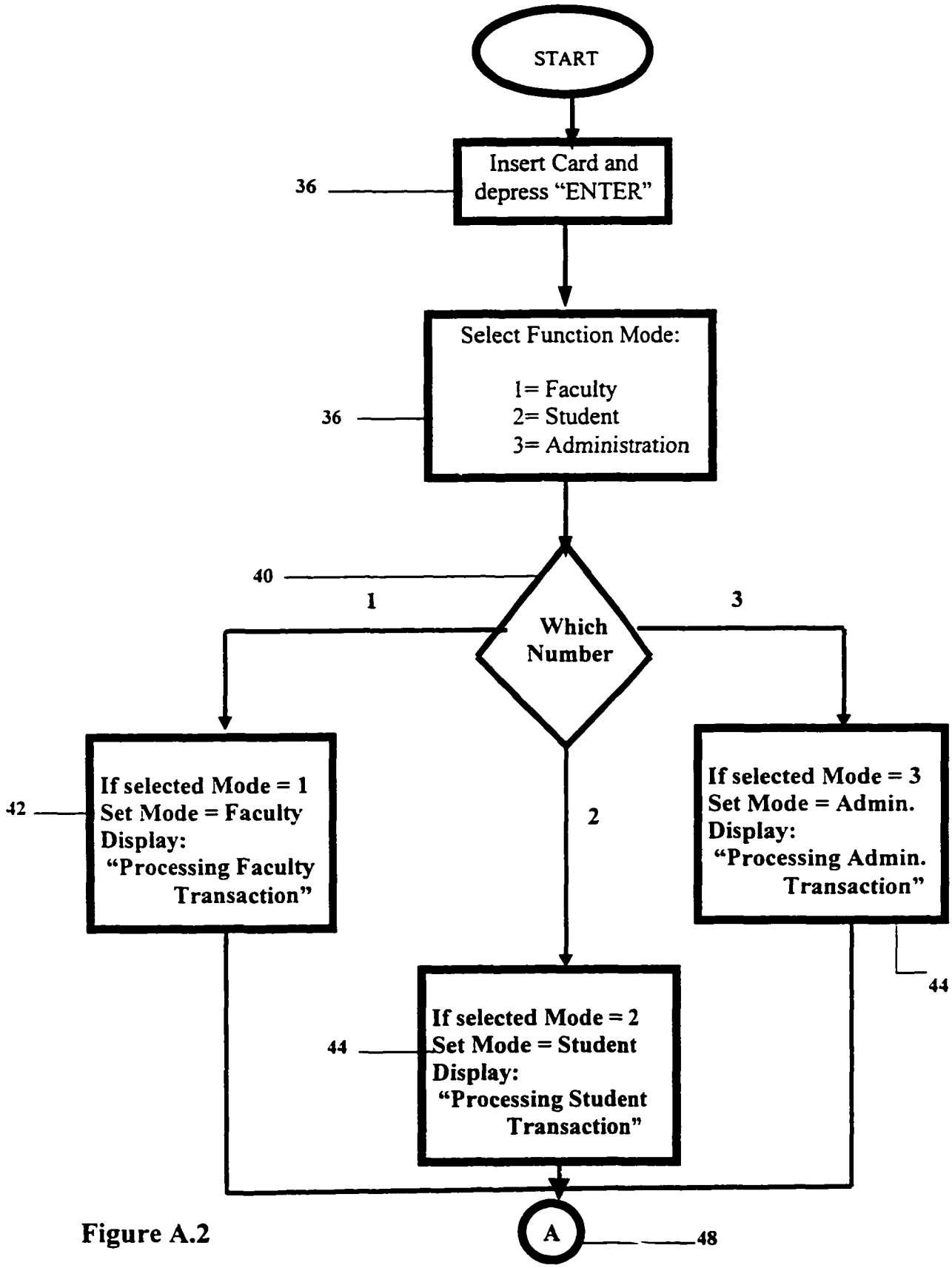


Figure A.2

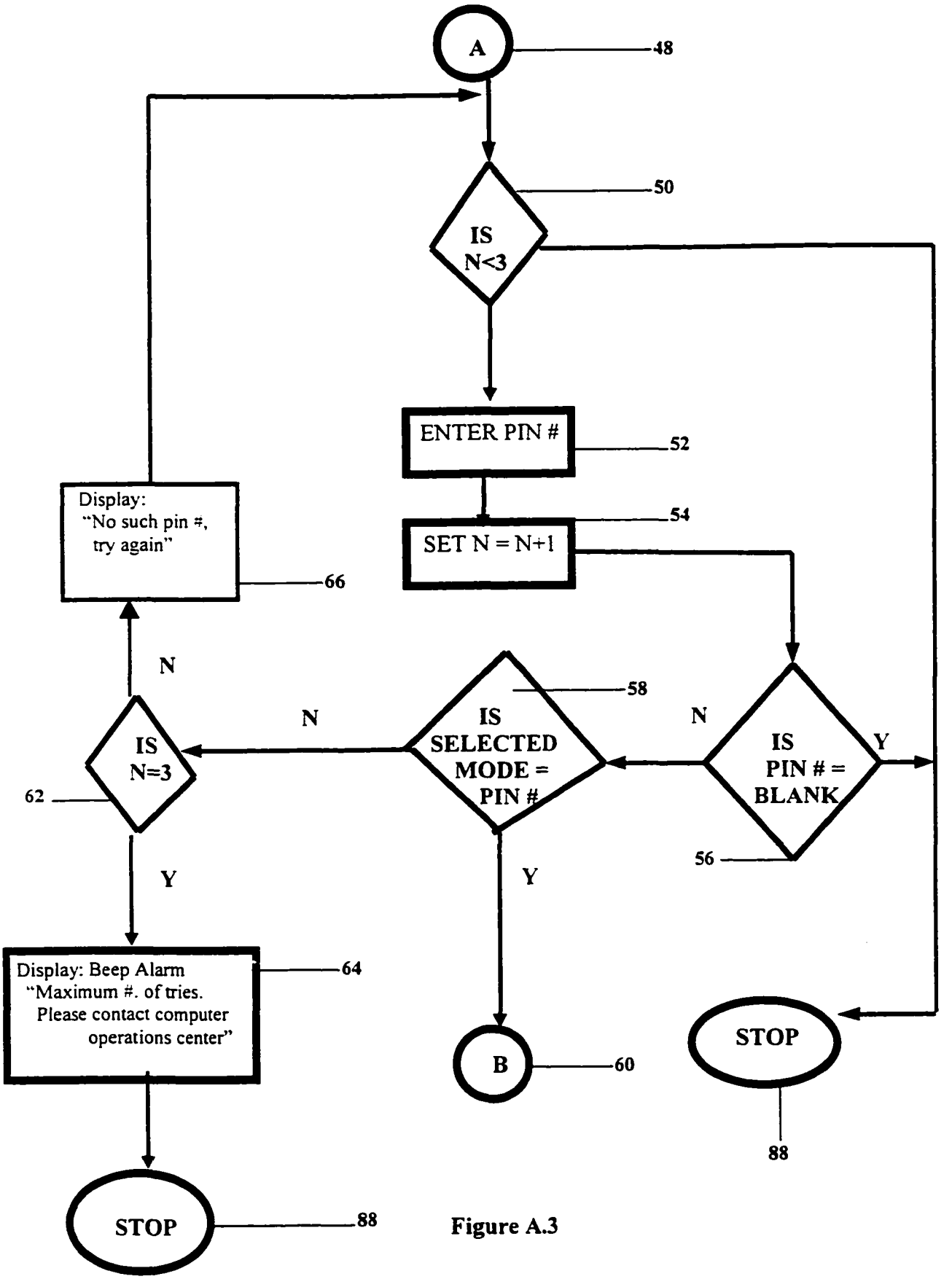


Figure A.3

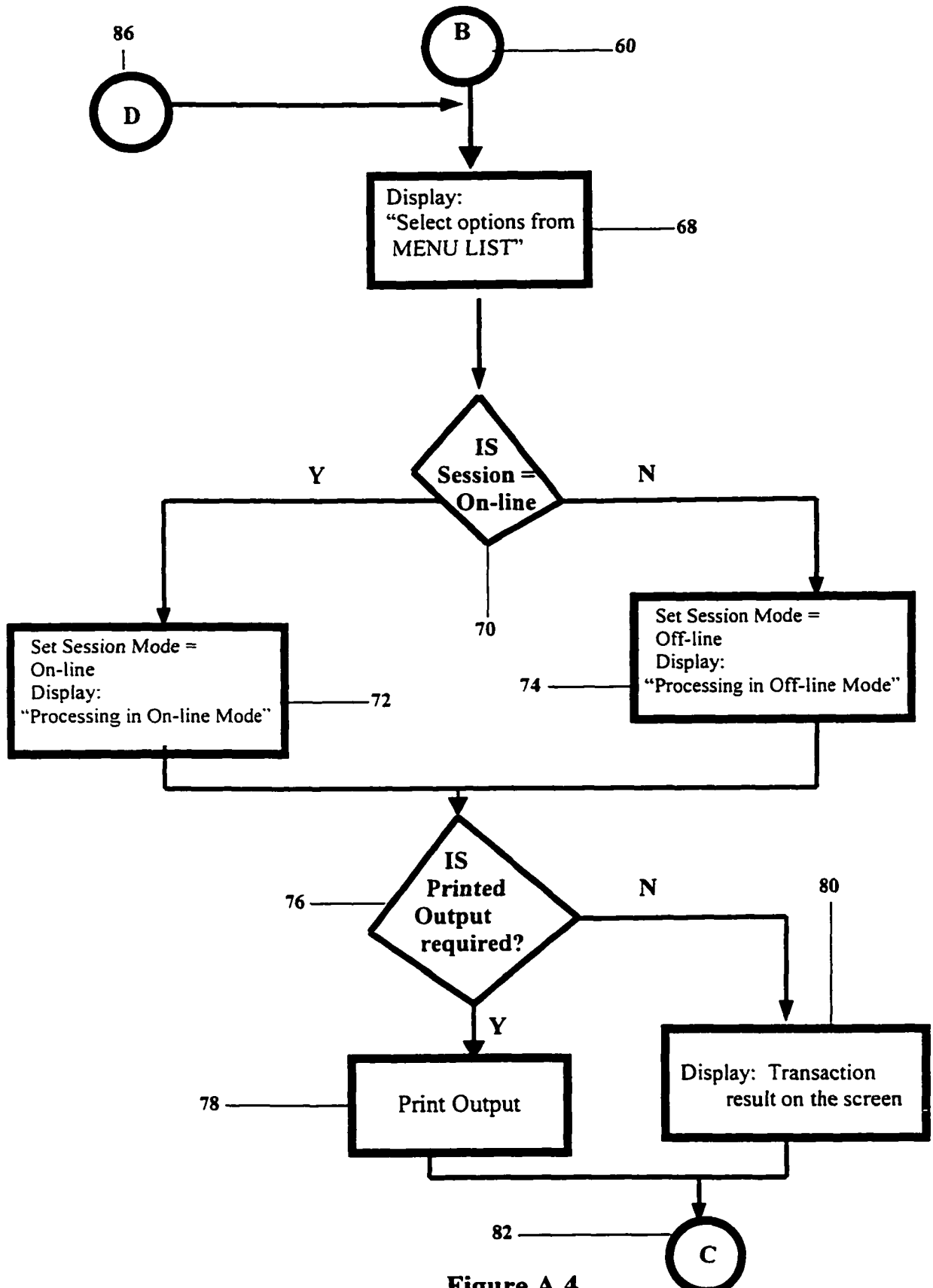


Figure A.4

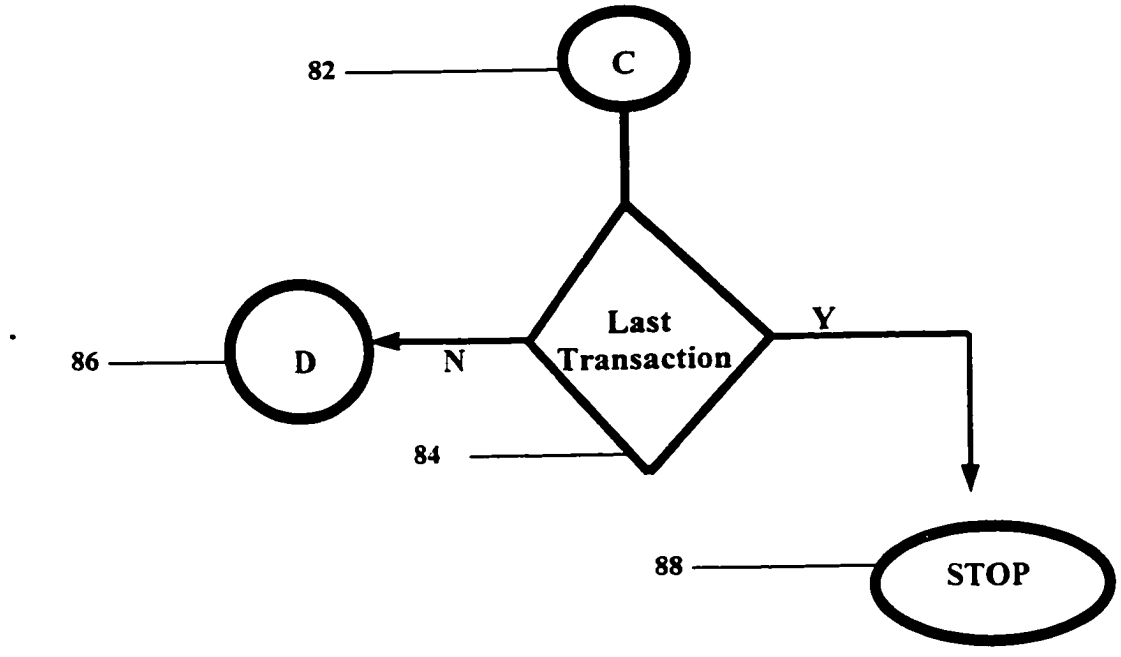


Figure A.5

SECURE SMART CARD LOADER

```
** FADSCARD.PRG ** This program produces the screens for data entry
** and processing of smart card transactions

close databases

set talk off
ctr= 0
pinno = space(6)
choice = 0

clear
start = ' '
@ 2, 21 say 'SMART CARD PROCESSING SCREEN'
@ 7, 22 say "Insert Card and Depress 'ENTER'";
   get start
   read

clear
@ 2, 21 say 'SMART CARD PROCESSING SCREEN'
@ 4, 21 say 'SELECT FUNCTION MODE:'
@ 5, 22 say '1 - Faculty'
@ 6, 22 say '2 - Student'
@ 7, 22 say '3 - Administrator'
@ 8, 22 get choice pict '@z 9' range 0,3
@ 8, 24 say '(0 or blank to Quit)'
read
*******************************
if choice = 0
 return
endif

if choice = 1
 mmode = 1
 clear
 @ 2, 21 say 'SMART CARD PROCESSING SCREEN'
 @ 4, 21 say 'Processing Faculty Transaction'
endif

if choice = 2
  mmode = 2
 clear
 @ 2, 21 say 'SMART CARD PROCESSING SCREEN'
 @ 4, 21 say 'Processing Student Transaction'
endif
```

```
if choice = 3
 mmode = 3
clear
@ 2, 21 say 'SMART CARD PROCESSING SCREEN'
@ 4, 21 say 'Processing Administrator Transaction'
endif

*******************************
************************************************************
asking = .T.
DO WHILE ASKING
*--- Ask for PIN number
@ 6, 15 say 'ENTER YOUR PIN NUMBER, (Blank to Cancel/Quit)'
@ 7, 20 get pinno pict '!!!!!!'
read

ctr = ctr + 1
*--- Exit on blank.  User presses the ENTER key for this.
   IF pinno = space(6)
     EXIT
   ENDIF
   *--- make sure pin number exists.
   USE PINFL
   SET INDEX TO PINNUMX
   REINDEX

   SEEK pinno

   *--- If pin number is not found (max 3 tries)
   IF .NOT. FOUND()

    @ 9, 15 say 'NO SUCH PIN #, TRY AGAIN (maximum: 3 tries)'
      If ctr = 3
       @ 9, 15
       @ 9, 15 say CHR(7) + 'MAXIMUM TRIES.  PLEASE'
       @ 11, 15 say 'CONTACT COMPUTER OPERATIONS CENTER'
       asking = .F.  && deactivate the loop
       EXIT        && exit the loop
      Endif

    notok = ' '
    @ 13, 15 say 'PRESS ANY KEY TO CONTINUE ';
         get notok
         read
```

```
      pinno = space(6) && clear pin number in memory
      *--- clear screen from row 6 downwards
      @ 6,0 CLEAR
      LOOP
ENDIF

*--- If pin number is found
IF FOUND()
 @ 6, 0 CLEAR
 @ 6, 21 say 'SELECT OPTIONS FROM MENU LIST'
 ok = 'N'
 @ 8, 21 say 'Is Session = On-line (Y/N)? ';
      get ok pict 'Y'
      read
      if ok = 'Y'
        do onlnpgm
      endif
      if ok = 'N'
        do offlnpgm
      endif

 YESNO = .F.
 SET CONFIRM ON
 @ 14, 17 say 'PRINTED OUTPUT (Y/N + enter key)? ';
       get YESNO PICT 'Y'
       READ
       if YESNO
         @ 6,  0 CLEAR
         mok = ' '
         @ 8,  17 say 'Printing Output ...'
         @ 9,  17 say 'Press any key to continue ';
               get mok
               read
       endif

       if .not. YESNO
          @ 6,  0 CLEAR
          mok = ' '
          @ 8,  17 say 'WAIT, Displaying ';
              + 'Transaction Result ...'
          @ 9,  17 say 'Press any key to continue ';
                get mok
                read
        endif

 YN = .F.
```

```
    @ 12, 17 say 'MORE TRANSACTIONS (Y/N + enter key)? ';
          get YN  PICT 'Y'
          READ
          if YN
            @ 6, 0 CLEAR
            pinno = space(6)
            LOOP
          endif

    @ 6,  0  CLEAR
    @ 12, 17 say 'END OF SESSION            GOOD BYE'
     asking = .F.
    ENDIF

ENDDO
*********************************************************
set confirm off
close databases
clear all

RETURN
```

BIBLIOGRAPHY

[1] M. Abadi et al., "Authentication and Delegation with Smart Cards," Tech. Report 67, System Research Center, Digital Equipment Corp., Palo Alto, Calif., Oct. 1990.

[2] M. Abdelguerfi, B.S. Kaliski, Jr., and W. Patterson, "Public-key Security Systems," in IEEE Micro, vol.16, no.3, June 1996, pp. 10-13.

[3] ACM SIGSOFT Software Engineering Notes, vol. 17 (3), July 1992.

[4] J. Adamek, Foundation of Coding: Theory and Applications of Error-Correcting Codes with an Introduction to Cryptography and Information Theory, John Wiley & Sons, N.Y., 1993.

[5] American Bar Association, Report on Computer Crimes. American Bar Association, Section on Criminal Justice. Task Force on Computer Crime, Washington, D.C., 1984.

[6] American National Standard for Information Systems (ANSI X3.106), B Data Encryption Algorithm - Modes of Operation. American National Standards Institute, 1983.

[7] American National Standard for Financial Institution Key Management (Wholesale). American Bankers Association, 1985.

[8] R.J., Anderson, Why Cryptosystems Fail. Comm. of the ACM, vol. 37, 11, Nov. 1994, pp. 32-40.

[9] "Approval of Federal Information Processing Standards Publication 186, Digital Signature Standard (DSS)," Federal Register, vol. 58, no.96, 19 May 1994, pp. 26208-26211.

[10] C. Asmuth and J. Blum, "A modular approach to key safeguarding," IEEE Trans. Inform. Theory, vol. IT-29, March 1983, pp. 208-210.

[11] S. Azzarone, "Safety PIN: can it keep card systems secure?". Bank Systems and Equipment, Nov. 1978.

[12] D. Balenson, "Automated Distribution of Cryptographic Keys Using the Financial Institution Key Management Standard," IEEE Comm. Magazine, vol. 23, no.9, Sept. 1985, pp. 41-46.

[13] S.K. Banerjee, "High Speed Implementation of DES," Computers and Security, vol.1, no.3, Nov. 1982, pp. 261-267.

[14] J. Barron, Breaking the Ring, Houghton Mifflin, Boston, 1987.

[15] R. Baskerville, "Information System security Design Methods". ACM Comput. Surv. 25, 4, Dec. 1993, pp. 375-414.

[16] S.M. Bellovin, and M. Merritt, "Encrypted Key Exchange: Password-based protocols secure against dictionary attacks". In Proc. IEEE Computer Society Symposium on research in Security and Privacy, Oakland, CA, May 1992, pp. 72-84.

[17] S.M. Bellovin, and M. Merritt, "An Attack on the Interlock Protocol when used for authentication". IEEE Trans. on Inform. Theory, vol. 40, 1, Jan. 1994, pp. 73-275.

[18] S.M. Bellovin and M. Merritt, "Limitations of thr Kerberos Authentication System," Proc. Winter Usenix Conf., Usenix Assoc., Berkeley, Calif., 1991, pp. 253-267.

[19] C.H. Bennett, G. Brassard, A.K. Ekert, Quantum Cryptography. Scient. American., Oct. 1992, pp. 50-57.

[20] T.A. Berson, and G.L. Barksdale, "KSOS - Development Methodology for a Secure Operating System," in Proc. NCC, vol. 48, AFIPS Press, Montvale, N.J., 1979.

[21] T. Beth, "Confidential Communication on the Internet," Scient. American, Dec. 1995, pp. 88-91.

[22] M. Blaze, " A Cryptographic File System for Unix," in Proc. of the First Conference on Computer and Comm. Security, Nov. 1993, pp. 9-16.

[23] M. Blaze, "Key Management in an Encrypting File System," Proceedings of the Summer '94 USENIX Conference, USENIX Association, 1994,pp. 27-35.

[24] B. BloomBecker, "Spectacular Computer crimes: What They Are and How They Cost American Businesses Half A Billion Dollars A Year". Dow-Jones-Irwin, Homewood, Illinois, 1990.

[25] L. Blum, M. Blum, and M. Shub, " A Simple Unpredictable Pseudo-Random Number Generator," SIAM Journal on Computing, vol. 15, no. 2, 1986, pp.364-383.

[26] C.L. Boltz, "Secrecy and security in one magnetic strip", Private Communication, Emidata/Malco, 1977.

[27] R.L. Brand, "Coping with the threat of Computer Security Incidents: A Primer from Prevention through Recovery", copyright Russell Brand, June, 1990.

[28] Brandstead, J. Gait, and S. Katske, " Report on the Workshop on Cryptography in support of Computer Security," NBSIR 77- 1291, National Bureau of Standards, Sept. 1976, pp. 21-22, and Sept. 1977.

[29] D.K. Branstad, Encryption Protection in Computer Comm. Systems. Proc. 4th Data Communications Symp., 1975, pp. (8-1) - (8-7).

[30] D.K. Branstad, Security of Computer communications. IEEE Comm. Soc. Mag., vol. 16(6), Nov. 1978, pp. 33-40.

[31] G. Brassard, "Modern Cryptology: A tutorial," in Lecture Notes in Computer Science, vol. 325. Springer-Verlag, 1988.

[32] E.F. Brickell, "Survey of Hardware Implementation of RSA," Advances in Cryptology - CRYPTO '89 Proceedings, Springer- Verlag, 1990, pp. 368-370.

[33] R. Broadbridge, and J. Mekota, "Secure Communications Processor Specification," ESD-TR-76-351, AD-A055164, Honeywell Information Systems, Mclean, Va., June 1976.

[34] M. Burrows, M. Abadi, and R.M. Needham, "A Logic of Authentication," ACM Trans. Computer Systems, Vol. 8, No. 1, Feb. 1990, pp. 18-36.

[35] C.M. Campbell, Design and Specification of Cryptographic Capabilities. IEEE Comm. Soc. Mag., vol. 16(6), Nov. 1978, pp. 15-19.

[36] D. Chaum, "Achieving Electronic Privacy," Scient. American, Aug. 1992, pp. 96-101.

[37] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," in Comm. of ACM, vol. 28, no. 10, OCT. 1985, pp. 1030-1044.

[38] D. Chaum, "The dining Cryptographers problem: Unconditional sender and recipient untraceability," in Journal of Cryptology, vol.1, no.1, 1988, pp.65-75.

[39] D. Chaum, "Privacy protected payments: Unconditional payer and/or payee untraceability," in Smart Card 2000: The future of IC Cards. Edited by David Chaum and Ingrid Schaumuller-Bichl, North-Holland, 1989.

[40] W.R. Cheswick, and S.M. Bellovin, Firewalls and Internet Security - Repelling the Willy Hacker. Addison-Wesley, Reading, Mass., 1994.

[41] Communications Privacy: "Federal Policy and Actions", General Office Report GAO/OSI-94-2, Nov. 1993.

[42] Computer World, Computer Crime in Japan. Computer World 17, 45, ID7-ID8, ID17-ID20. Nov. 7, 1983.

[43] J.A. Cooper, Computer and Communications Security: Strategies for the 1990's. McGraw-Hill, New York, 1989.

[44] G. Davida, R. DeMillo, and R. Lipton, Protecting Shared Cryptography Keys. Proc. IEEE Symp. Security and Privacy. April 1980, pp. 99-102.

[45] D.W. Davies and W.L. Price, "The Application of digital Signatures based on Public Key Cryptosystems," National Physical Laboratory Report DNACS 39/80, Dec. 1980.

[46] D.W. Davies, Tutorial: The Security of Data in Networks. IEEE Computer Society Press, Los Angeles, 1981.

[47] D.W. Davies, "Applying the RSA digital signature to electronic mail," Computer, vol.16, no.2, Feb. 1983, pp. 55-62.

[48] D.W. Davies and W.L. Price, Security for Computer Networks. John Wiley and Sons, Second Edition, 1989.

[49] D.W. Davies and W.L. Price, "Security for Computer Networks: An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer," John Wiley & Sons, 1994.

[50] R. DeMillo and M. Merritt, Protocols for Data Security, Computer, vol.16, no.2, Feb. 1983, pp. 39-59.

[51] D.E. Denning, "Secure Personal Computing in an Insecure Network," Comm. ACM, vol. 22(8), Aug. 1979, pp. 476-482.

[52] D.E. Denning and G. Sacco, "Timestamps in Key Distribution Protocols". Comm. ACM, Aug.1981, vol. 24, no.8, pp. 533-536.

[53] D.E. Denning, Cryptography and Data Security. Addison- Wesley, Reading, Mass., 1982.

[54] D.E. Denning, "Protecting public keys and signature keys," Computer, vol. 16, no.2, Feb. 1983, pp. 27-35.

[55] D.E. Denning, "To Tap or Not To Tap," Comm. ACM, vol. 36(3), March, 1993.

[56] "Department of Defense Trusted Computer System Evaluation Criteria," DoD 5200.28-STD, Dec. 1985.

[57] W. Diffie and M.E. Hellman, "New Direction in cryptography". IEEE Trans. on Infor. Theory, IT-22, 1976, pp. 472-492.

[58] W. Diffie and M.E. Hellman, "Multiuser Cryptographic Techniques," Proc. of AFIPS National Computer Conference, Nov. 1976, pp. 109-112.

[59] W. Diffie and M.E. Hellman, "New Directions in Cryptography". IEEE Trans. on Infor. Theory, vol.IT-11, Nov. 1976, pp. 644-654.

[60] W. Diffie and M.E. Hellman, "Exhaustive Cryptanalysis of the NBC Data Encryption Standard," Computer, vol.10, no.6, June 1977, pp. 74-84.

[61] W. Diffie and M.E. Hellman, "Privacy and Authentication: An Introduction to Cryptography," Proc. IEEE, Vol. 67, No.3, Mar. 1979, pp. 397-427.

[62] W. Diffie, "Cryptography Technology: Fifteen Year Forecast," BNR Inc., Jan. 1981.

[63] W. Diffie, "Securing the DoD Transmission Control Protocol," in Lecture Notes in Computer Science 218; Advances in Cryptology: Proc. Crypto '85, H.C. Williams, Ed., Santa Barbara, CA., Aug. 18-22, 1985, pp. 108-127. Berlin: Springer-Verlag, 1986.

[64] W. Diffie, L. Strawczynski, B. O'Higgins, and D. Steer, "An ISDN Secure Telephone Unit," in Proceedings, National Comm. Forum 1987, Rosemount, IL, Sept. 28-30, vol.41, book 1, pp. 473-477. Chicago: National Engineering Consortium, 1987.

[65] W. Diffie, "The First Ten Years of Public-Key Cryptography," Proceedings of the IEEE, vol. 76, no. 5, May 1988, pp. 560-577.

[66] R. Dixon, C. Marston, and P. Collies, A Report on the Joint CIMA and IIA Computer Fraud Survey. Computer, Sec. 11, 4, July 1992, pp. 307-313.

[67] D. Dolev and A.C. Yao, "On the Security of Public Key protocols," IEEE Trans. Information Theory, Vol. IT-30, No. 2, Mar. 1983, pp 198-208.

[68] W.F. Ehrsam, S.M. Matayas, C.H. Meyer, and W.L. Tuchman, "A Cryptographic key anagement scheme for implementing the Data Encryption Standard, "IBM Syst. J., vol 17, no.2, 1978, pp. 106-125.

[69] T. ElGamal, A Public-Key Cryptosystem and A Signature Scheme Based On Discrete Logarithms. IEEE Trans. Inf. Theory 31 (1985), pp. 469-472.

[70] C.H. Fancher, "Smart Cards," Scient. American, Aug. 1996, pp. 40-45.

[71] R. Fagin, M. Naor, and P. Winkler, "Comparing Information - without leaking it," in Comm. of CM, vol.39, no.5, May 1996, pp. 77-85.

[72] U. Feige, A. Fiat, and A. Shamir, "Zero-Knowledge Proofs of Identity," Proc. ACM Symp. Theory of Computing, ACM Press, New York, 1987, pp.210-217.

[73] H. Feistel, W.A. Notz, and J.L. Smith, "Some Cryptographic Techniques for Machine to Machine Data Communication," Proc. IEEE, vol. 63, 11, Nov. 1975, pp. 1545.

[74] A.M. Findlay, "A chip off the old security block," in Card Technology (Faulkner & Gray), vol.1, no.2, May/June 1996, pp. 52-60.

[75] M. Fite, P.Kratz, and A. Brebner, " Control and Security of Computer Information Systems", Computer Science Press, 1989.

[76] P. Fites, P. Johnston, and M. Kartz, The Computer Virus Crisis. Van Nostrand, Reinhold, New York, 1989.

[77] R. Flynn and A.S. Campasano, "Data Dependent Keys for a Selective Encryption Terminal," in Proc. NCC, vol.47, AFIPS Press, Montvale, N.J., 1978, pp. 1127-1129.

[78] W.F. Friedman, "The Index of Coincidence and its Applications in Cryptography," The Riverbank Publications, Aegean Park Press, Laguna, CA. 1979.

[79] P. Gannon, French Losses Rise Sharply. Computer Fraud Sec. Bull. 14, 12, Oct. 1992, pp. 3.

[80] G. Garon and R. Outerbridge, "DES Watch: An Examination of the Sufficiency of the Data Encryption Standard for Financial Institution Information Security in the 1990's," Cryptologia, vol.15, no.3, July, 1991, pp. 177-193.

[81] P. Gauthier, "Get Set! Smart cards are coming to America," in Portable Design, vol.1, no.6, May 1996, pp. 31-34.

[82] M. Gasser and E. McDermott, "An Architecture for Practical Delegation in a Distributed System," Proc. IEEE Symp. Research in Security and Privacy, IEEE CS Press, Los Alamitos, Calif., Order No. 2060, 1990, pp. 20-30.

[83] M. Gasser et al., "The Digital Disributed System Security Architecture," Proc.Nat'l Computer Security Conf., Nat'l Institute of Standards and Technology, 1989, pp. 305-319.

[84] B.D. Gold, R.R. Linde, R.J. Peeler, M. Schaefer, J.F. Scheid, and P.D. Ward, "A Security Retrofit of VM/370," in Proc. NCC, vol. 48, AFIPS Press, Montvale, N.J., 1979, pp.335-344.

[85] F.T. Grampp and R.H. Morris, Unix Operating System Security. AT&T Bell Lab. Tech. Journal, 63(8, Part 2): Oct. 1984, pp. 1649-1672.

[86] C.G. Gunther, "An Identity-Based Key-exchange protocol in Advances in Cryptology". Eurocrypt, 1989. New York: Springer-Verlag, 1990, pp.29-37.

[87] K. Hafner and J. Markoff, Cyberpunk: Outlaws and Hackers on the Computer Frontier. Simon and Schuster, New York, 1991.

[88] M.A. Harrison, "Theoretical Issues Concerning Protection in Operating Systems", in Advances in Computers, vol.24, M.C. Yovits (ed.), Academic Press, pp. 61-100.

[89] H. Highland, Random bits and bytes: Michelangelo - Part II. Computer, Sec. 11, 4, July, 1992, pp. 294-303.

[90] J. Hoffer and D. Straub, The 9 to 5 Underground: Are You Policing Computer Crimes? Sloan Managerial Rev. 30, 4, Summer, 1989, pp. 35-43.

[91] J. Hruska, Computer Viruses and Anti-Virus Warfare. Ellis Horwood, New York, 1990.

[92] Identification Cards - Physical Characteristics ISO 7810, International Organization for Standardization, 1988.

[93] R.R. Jueneman, "A high speed manipulation detection code," in Lecture - 214 -Notes in Computer Science 263; Advances in Cryptology: Proc. Crypto '86, A.M. Odlyzko, Ed., Santa Barbara, CA, Aug. 11-15, 1986, pp.327-346. Berlin: Springer-Verlag, 1987.

[94] R.R. Jueneman, S.M. Matyas, and C.H. Meyer, "Message authentication with manipulation detection codes," in Proc. 1983 IEEE Symp. Security

and Privacy, R. Blakley and D. Denning, Eds., Oakland, CA, April 1983, pp. 33-54. Los Angeles: IEEE Computer Society Press, 1983.

[95] D. Kahn, The Code-Breakers. MacMillan, New York, 1967.

[96] S.C. Kak, "Data Security in Computer Networks,"in IEEE Computer, Feb.1983, pp. 8-10.

[97] B.S. Kaliski Jr., An overview of the PKCS standard. Tech. Rep., RSA Data security, Inc., June, 1991.

[98] S.T. Kent, "Encryption-based protection protocols for Interactive user - Computer Communication". MIT/LCS/TR - 62, MIT Lab. for Computer Sciences. Cambridge, Mass., May, 1976.

[99] S.T. Kent, "Encryption-based protection for Interactive User-Computer Communication". Proceedings of the Fifth Data Comm. Symposium, Sept., 1977.

[100] D.V. Klein,"Failing the Cracker: A survey of, and improvements to, password security", in Proc. of the USENIX Unix Security Workshop, Porland, Oregon, Aug., 1990, pp. 5-14.

[101] L.M. Kohnfelder, "Towards a Practical Public Key Cryptosystem," bachelor's Thesis, MIT Department of Electrical Engineering, May 1978.

[102] L.M. Kohnfelder, "On the signature reblocking problem in Public-Key Cryptosystems," Comm. ACM, vol 21, no.2, Feb. 1978, pp. 179.

[103] A.G. Konheim, M.H. Mack, R.K. McNeill, B. Tuckerman, and G. Waldbaum, "The IPS Cryptographic Programs," IBM Syst. J. vol. 19(2), 1980, pp. 253- 283.

[104] E. Kranakis, Primality and Cryptograph, Wiler-Tuebner Series in Computer Science, 1986.

[105] J.B. Lacy, D.P. Mitchell, and W.M. Schell, "CryptoLib: Cryptography in Software," Unix Security Symposium IV Proc. USENIX Association, 1993, pp.1-17.

[106] G.M. Lancaster, "Code Comparison - An Inexpensive Approach to Security and Auditability," in Datapro Newsbriefs, March 1981.

[107] S.S. Lam, A.U. Shankar, and T.Y.C. Woo, "Applying a Theory of modules and Interfaces to Security Verification," Proc. IEEE Symp. Research in Security and Privacy, IEEE CS Press, Los Alamitos, Calif., Order No. 2168, 1991, pp. 136-154.

[108] B.A. LaMacchia and A.M. Odlyzko, Computation of Discrete Logarithms in Prime Fields: Designs, Codes, and Cryptography, vol.1, 1991, pp.47-62.

[109] B. Landreth, "Understanding Organizational Transformation Using a Dissipative Structure Model", Hum. Rel. vol.42, 10, 1989, pp. 899-916.

[110] A. Lempel, " Cryptography in Transition," ACM Comput. Surveys, vol. 11, no.4, Dec. 1979, pp. 285-303.

[111] P. Leong and C. Tham, "Unix password encryption considered insecure", in Proc. Winter USENIX Conference, Dallas, Texas, 1991.

[112] R. Lindsey, The falcon and the Snowman, New York: Simmon & Schuster, 1979.

[113] S.M. Lipton, and S.M. Matyas, "Making the Digital Signature Legal - and Safeguarded," Data Communications, Feb., 1978, pp. 41-52.

[114] J. Linn," Practical Authentication for Distributed Computing," Proc. IEEE Symp. Research in Security and Provacy, IEEE CS Press, Los Alamitos, Calif., Order No. 2060, 1990, pp.31-40.

[115] J. Loxton, D.S.P. Khoo, G.J. Bird, and J. Seberry, "A cubic residue code equivalent to factorization". Journal of Cryptology, vol.5, 1992, pp. 139-150.

[116] M. Luby and C. Rackoff, " A study of Password Security," in Journal of Cryptology, 1989, pp. 151-158.

[117] D. MacMillan, "Single chip encrypts data at 14 Mb/s, "Electronics, vol.54, no.12, June 16, 1981, pp. 161-165.

[118] E.J. McCauley and P.J. Drongowski, "KSOS - the Design of a Secure Operating System," in Proc. NCC, vol.48, AFIPS Press, Montvale, N.J., 1979, pp. 345-353.

[119] J.L. Massey, "Contemporary Cryptology: The Science of Information Integrity", G.J. Simmon, Ed, IEEE Press, New York, 1991, pp. 3-39. Also in proceedings of the IEEE 76, 5, May, 1988, pp. 533-549.

[120] S.M. Matyas, "Key handling with control vectors," IBM Systems Journal, vol.30, no.2, 1991, pp. 151-174.

[121] S.M. Matyas, A.V. Lee, and D.G. Abraham, "A key management scheme based on control vectors," IBM Systems Journal, vol.30, no.2, 1991, pp. 175- 191.

[122] R. McIvor, "Smart Cards," Scient. American, Nov. 1985.

[123] R.C. Merkle, "Protocol for Public Key Cryptosystems," Proc. 1980 Symp. on Security and Privacy", IEEE Computer Society, April 1980, pp. 122-133.

[124] R.C. Merkle, "Secret Authentication and Public Key Systems". Ann Arbor: UMI Research Press, 1982.

[125] R.C. Merkle, "Protocols for public key cryptosystems," in Secure Communications and Asymmetric Cryptosystems, G.J. Simmons, ed., Boulder, CO: Westview Press, 1982, pp. 73-104.

[126] J.K. Millen, "Security Kernel Validation Practice," Comm. ACM vol. 19, 5, May 1976, pp. 243-250.

[127] S. Micali, "Fair Public-Key Cryptosystem," Lab. for Computer Sci., MIT, Aug. 21, 1992.

[128] M.G. Morgan, "Viewpoint: The Institute," IEEE, Nov. 1992.

[129] R.H. Morrison and K. Thompson, "Unix password security: a case history", Comm. of the ACM, 22(11), Nov. 1979, pp. 594.

[130] W. Myers, "On Trial at the Summer Olympic Games: Smart Cards," IEEE Computer, July 1996, pp. 88-91.

[131] D. Naccache and D. M'Raihi, "Cryptographic Smart Cards," in IEEE Micro, vol. 16, no. 3, June 1996, pp. 14-24.

[132] R.N. Nagel and A. Rosenfeld, "Computer detection of freehand forgeries",IEEE Trans. on Computers, Sept. 1977, C-26, 9, 895.

[133] National Bureau of Standards, Guidelines on evaluation of techniques for automated personal identification, Federal Information Processing Standards Publication 48, 1977.

[134] National Bureau of Standards, Federal Information Processing Standards Publication 46: Data Encryption Standard, Washington, D.C.,: U.S. Dept. of Commerce, Jan. 15, 1977.

[135] National Bureau of Standards, Federal Information Processing Standards Publication 81: DES Modes of Operation, Washington, D.C.,: U.S. Dept. of Commerce, Dec. 2, 1980.

[136] National Bureau of Standards, Federal Information Processing Standards Publication 74: Guidelines for Implementing and Using the NBS Data Encryption Standard, Washington, D.C.: U.S. Dept. of commerce, April 1, 1981.

[137] National Bureau of Standards, NBS FIPS PUB.81, "DES Modes of Operation," U.S. Dept. of commerce, Dec. 1980.

[138] National Computer Security Center, "A Guide to Understanding Data Remembrance in Automated Information Systems," NCSC-TG-025 version 2, Sept. 1991.

[139] National Institute of Standards and Technology, Executive Guide to the Protection of Information Resources, Federal Information Processing Standards Publication 500-169, Oct. 1989.

[140] National Institute of Standards and Technology, Management Guide to the Protection of Information Resources, Federal Information Processing Standards Publication 500-170, Oct. 1989.

[141] National Institute of Standards and Technology, Computer User's Guide to the Protection of information Resources, Federal Information Processing Standards Publication 500-171, Oct. 1989.

[142] National Institute of Standards and Technology, NIST FIPS PUB 186, "Digital Signature Standard," U.S. Dept. of Commerce, May 1994.

[143] National Research Council, "Computers at Risk: Safe computing in the information age". National Academy Press, Washington, D.C., 1991.

[144] R.M. Needham and M. Schroeder, "Using encryption for authentication in large networks for computers", Comm. of ACM, 21(12), Dec. 1978, pp. 993-999.

[145] P.G. Neumann, "Computer Related Risks", ACM Press, New York, 1995.

[146] M.G. Noblett, "The Computer: High-Tech Instrument of Crime", FBI Law Enforcement Bulletin, June, 1993, pp. 7-9.

[147] B. O'Higgins, W. Diffie, L. Strawzynski, and R. de Hoog, "Encryption and ISDN - a atural fit," in Proceedings International Switching Symp., Phoenix, AZ, March 16-20, 1987, pp. 1.4.1-7.

[148] D. Parker, "Crime by Computer", Scribners & Sons, New York, 1976.

[149] W. Patterson, "Mathematical Cryptology for Computer Scientists and Mathematicians". Totowa, N.J.: Rowman and Littlefield, 1987.

[150] Peer Entity Authentication Mechanisms Using an n-bit Secret Key Algorithm, Draft Proposal 9798, International Standards Organization, (ISO/9798), 1988.

[151] Peer Entity Authentication Mechanism Using a Public Key Algorithm with a Two-way Handshake, Draft Proposal 9799, International Standards Organization, (ISO/9799), 1988.

[152] Peer Entity Authentication Mechanism Using a Public Key Algorithm with a Three-way Handshake, Draft Proposal 10117, International Standards Organization, (ISO/10117), 1988.

[153] Personal Identification Number Management and Security, X9.8-1982, American National Standards Institute, (ANSI/X9.8-1982), 1982.

[154] C.P. Pfleeger, "Security in Computing". Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[155] S.C. Pohlig and M.E. Hellman, An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. IEEE Trans. Inf. Theory IT-24, 1978, pp. 106-110.

[156] G.J. Popek and C.S. Kline, "Encryption protocol, public key algorithms and digital signatures in computer networks," in Foundations of Secure Computation, R.A. DeMillo, D.P. Dobkin, A.L. Jones, and R.J. Lipton, eds., New York: Academic Press, 1978, pp. 133-153.

[157] G.J. Popek and C.S. Kline, " Encryption and Secure Computer Networks,"ACM Compt. Surveys, vol.11, no.4, Dec. 1979, pp. 331-356.

[158] G.J. Popek, M. Kampe, C.S. Kline, A. Stroughton, M. Urban, and E. Walton, "UCLA Secure Unix," in Proc. NCC, vol.48, AFIPS Press, Montvale, N.J., 1979, pp. 355-364.

[159] "Proposed Federal Information Processing Standard for Digital Signature Standard (DSS)," Federal Register, vol. 56, no. 169, Aug. 1991, pp. 42980-42982.

[160] M. Rabin, "Digitalized Signature: Foundations of Secure Computation", R.A. DeMillo et al., eds., Academic Press, New York, 1978, pp. 155-168.

[161] R.L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystem", Comm. of ACM, vol. 21, 2, Feb. 1978, pp. 120-126.

[162] R.L. Rivest and A. Shamir, "How to expose an eavesdropper", Comm. of ACM, vol. 27, 4, April, 1984, pp. 393-395.

[163] R.L. Rivest, "Response to NIST's proposal,"Comm. ACM, vol.35(7), July 1992, pp. 41-47.

[164] A. Salomaa, "Cryptography", in Encyclopedia of Mathematics and its Applications, vol. 25: Computation and Automata, A. Salomaa, pp. 186-230. Cambridge, United Kingdom: Cambridge University Press, 1985.

[165] J. Saltzer, "On Digital Signature," Oper. System Rev., vol. 12, 2, April 1978, pp. 12-14.

[166] J. Seberry and J. Pieprzyk, "Cryptography: An Introduction to Computer Security", Prentice-Hall, 1989.

[167] J.I. Schiller, "Secure Distributed Computing", Scient. American, Nov. 1994, pp. 72-76.

[168] W.L. Schiller, "The Design and Specification of a Security Kernel for the PDP-11/45," ESD-TR-75-69, The MITRE Corp., Bedford, Mass., March 1975.

[169] B. Schneier, "Applied Cryptography: Protocols, Algorithms, and Source Code in C", John Wiley &Sons, New York, 1994.

[170] C.P.Schnorr, Efficient identification and signatures for smart cards. Advances in Cryptology: Proceedings CRYPTO '89, G. Brassard Ed., Lecture Notes in Computer Science No. 435, Springer-Verlag, N.Y., 1990, pp. 239-252.

[171] M.D. Schroeder, D.D. Clark, and J.H. Saltzer, "The MULTICS Kernel Design Project," Proc. 6th Symp.on Oper. Syst. Princ., ACM Oper. Syst. Rev. vol.11, 5, Nov.1977, pp.43-56.

[172] A. Shamir, "How to share a secret", Comm. of ACM, vol. 22, 11, Nov. 1979, pp. 612-613.

[173] C.E. Shannon, "A mathematical theory of communication", Bell System Tech. Journal, vol. 27(3,4), July, Oct., 1949, pp. 379-423, 623-656.

[174] C.E. Shannon, "Communication theory of secrecy systems", Bell System Tech. Journal, vol. 28, Oct. 1949, pp. 656-715.

[175] K. Shankar, "The Total Computer Security Problem," Computer, Vol. 10, No.6, June 1977, pp. 50-73.

[176] G.J. Simmons, "Symmetric and Asymetric Encryption," ACM Comput. Surveys, vol. 11, no. 4, Dec. 1979, pp.305-330.

[177] G.J. Simmons, ed., 1992a, "Contemporary Cryptology: The Science of information Integrity," IEEE Press, N.J., 1992.

[178] G.J. Simmons, "Cryptanalysis and Protocal failures", Comm. of ACM, vol. 37(11), Nov. 1994, pp. 56-65.

[179] M.E. Smid, "Integrating the data encryption standard into computer networks," IEEE Trans. Compt., vol. COM-29, no.6, June 1981, pp. 762-772.

[180] M.E. Smid and D.K. Branstad, "The Data Encryption Standard: Past and Future," Proc. IEEE, vol. 76, no.5, May 1988, pp. 550-559.

[181] A. Solarz, "Computer related embezzlement", Comput. Sec., vol.6, no.1, 1987, pp.49-53.

[182] E.H. Spafford, "The Internet Worm: Crisis and Aftermath," Comm. of ACM vol.32, no.6, June 1989, pp.678-687.

[183] W. Stallings, "Network and Internetwork Security: Principles and Practice", Prentice-Hall, N.J., 1995.

[184] J.G. Steiner, C. Neuman, and J.I. Schiller, "Kerberos: An Authentication Service for Open Network Systems, "Proc. Winter Usenix Conf., Usenix Assoc., Berkeley, Calif., 1988, pp. 191-202.

[185] B. Sterling, "The Hacker Crackdown: Law and disorder on the electronic frontier," Bantam,
New York, 1992.

[186] C. Stoll, "The Cuckoo's Egg: Tracking a Spy Through the Maze of computer Espionage", Doubleday, New York, 1989.

[187] U.S. Dept. of Commerce, National Bureau of Standards, "DES Modes of Operation," Federal Information Processing Standards Publication 81, Dec. 1980.

[188] J.J. Tardo and K. Alagappan, "SPX: Global Authentication Using Public Key Certificates," Proc. IEEE Symp. Research in Security and Privacy, IEEE CS Press, Los Alamitos, Calif., Order No. 2168, 1991, pp. 232-244.

[189] B. Violino, "Cover Story: Hackers", Information Week, June 1993, pp. 48-56.

[190] K.G. Walter, et al., Structured Specification of a Security kernel," Proc. Int. Conf. Reliable Software, ACM SIGPLAN Notices, vol. 10(6), June 1975, pp. 285-293.

[191] C. Weissman, "Security Controls in the ADEPT-50 Time-sharing System," in Proc. Fall Jt. Computer Conf., vol. 35, AFIPS Press, Montvale, N.J., 1969, pp. 119-133.

[192] D. Welsh, Codes and Cryptography, Oxford University Press, 1989.

[193] T. Whiteside, "Computer Capers: tales of Electronic Thievery, Embezzlement, and Fraud", Fitzhenry and Whiteside, Toronto, 1978.

[194] M.J. Wiener, "Efficient DES Key Search," TR-244, School of Computer Science, Carleton University, May 1994.

[195] M.V. Wilkes, "Time-Sharing Computer Systems," Elsevier, New York, 3rd edition, 1975.

[196] H.C. Williams, "An M^3 public-key encryption scheme," in Lecture Notes in Computer Science 218; Advances in Cryptology: Proc. of Crypto '85, H.C. Williams, ed. (Santa Barbara, CA., Aug. 18-22, 1985).: Springer-Verlag, Berlin, 1986, pp.358-368.

[197] J. Wilson, "Views as the Security Objects in a Multilevel Secure Relational Database Management System," 1988 IEEE Symposium on Security and Privacy, April 18-21, 1988.

[198] C. Wood, "Principles of Secure Information Systems Design," Comput. Sec., vol.9, no.1, Feb. 1990, pp. 13-24.

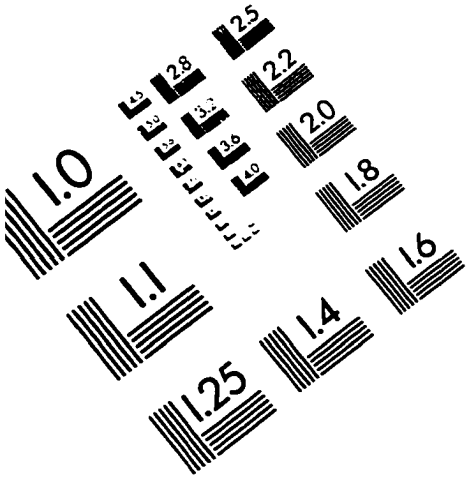
[199] C. Woodward and R. Bernstein, "All President's Men," Simon and Schuster, New York, 1974.

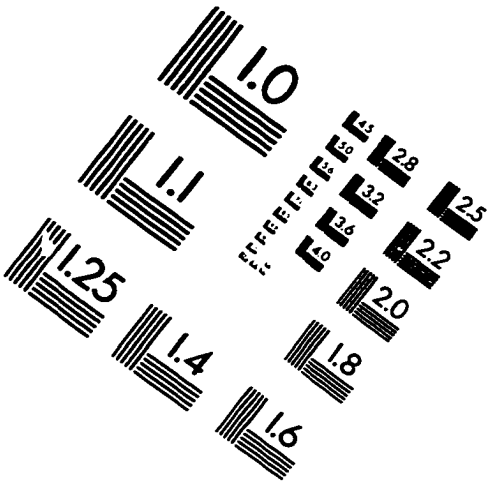
[200] P.R. Zimmermann, The Original PGP User's Guide, Boston: MIT Press, 1995.

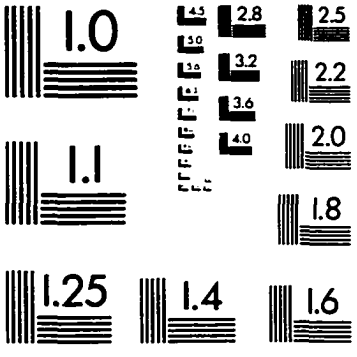
[201] E. Zipp, "More Security Tips for VMS System Managers," Digital Review, April 1985.

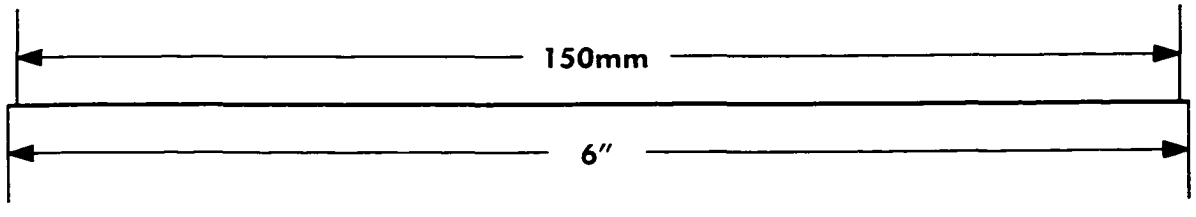
[202] J.J. Zoreda and J.M. Oton, "Smart Cards", Artech House, Norwood, MA.,1994.

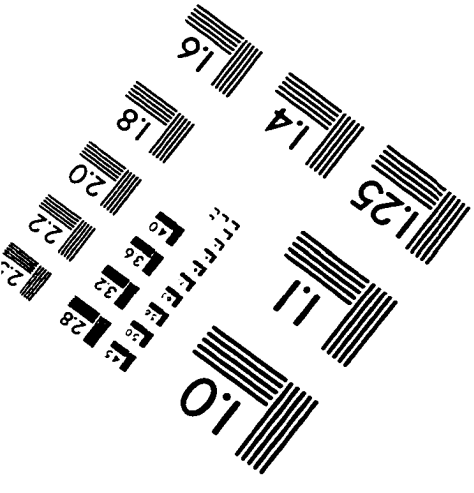
IMAGE EVALUATION TEST TARGET (QA-3)

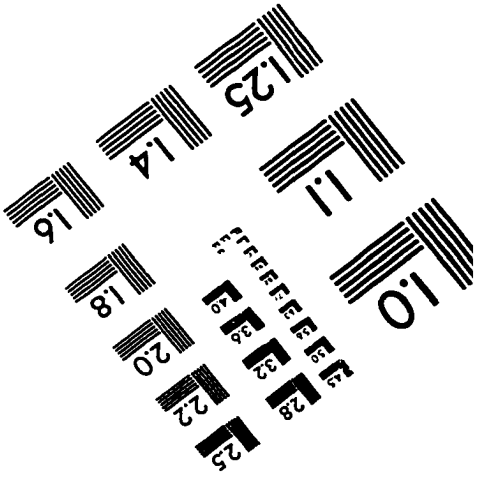












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989