Computers & Security, 12 (1993) 79–97

A survey of security issues in office computation and the application of secure computing models to office systems

R. B. Vaughn Jr.[1], H. Saiedian[2] and E. A. Unger[3]

[1] *U.S. Army Information Systems Software Center, Fort Belvoir, VA, USA*
[2] *Department of Computer Science, University of Nebraska, Omaha, NE, USA*
[3] *Department of Computing and Information Sciences, Kansas State University, Manhattan, KS, USA*

With the proliferation of office information systems (OIS) into almost every area of industry and government, it is important to design systems that offer a guarantee of privacy and security to their users. The same solutions and research pertaining to traditional data-processing environments cannot, in most cases, be applied directly to the OIS environment. Many OIS do not provide the hardware/software controls necessary to protect information from anyone who gains physical access to the system. Furthermore, users of an OIS cannot be expected to possess a clear understanding of the system, its operating characteristics, or even the implication of interconnecting component devices. This paper examines the typical OIS environment with a view toward the provision of a secure operating architecture. Important security problems faced by OIS are enumerated and explained. We argue that the OIS environment presents a different problem to solve in a security sense from when working with a traditional nondistributed system. Existing solutions found in large scale operating systems and networks cannot simply be scaled down and moved to an OIS, if for no other reason than the architecture's inability to support locks and multiple system states. An architecture of both software and hardware controls must be built for this new environment using concepts from large scale operating systems but recognizing the limitations and constraints of OIS. While we emphasize the security issues, we look at alternative technologies that can be combined to provide a solution.

Keywords: Office information systems, Privacy, Protection, Security models.

1. Introduction

Office automation (OA) has become a new and active area of research in computer science. The focus of this research is on the design methodologies, software tools, and system integration techniques that increase the productivity of office workers. An important instance of OA is called an office information system (OIS). This is a set of diverse and highly interactive components that attempts to perform the functions of an office by means of a computer-based system.

One of the oldest problems and perhaps the most persistent in computer science has to do with the development of secure systems. Concerns over the issue of security and privacy in computer systems have steadily increased within the computer science discipline since the mid 1960s. These concerns have often been expressed as a need for some assurance that their information is reasonably safeguarded from theft, destruction, unwanted modification, and unauthorized viewing. These same concerns are also relevant within the emerging discipline of OIS and must be addressed within the context of that environment. In general, such systems typically consist of multifarious heterogeneous components servicing both a localized area and a distant customer (or group of customers) interconnected over a network facility. The personnel using these systems may not be computer literate and will likely require a highly efficient, easy to use, user interface. They trust the system to safeguard their electronic documents and rely upon its ability to guarantee separation of users without appropriate authorization from information that is restricted in some sense. Herein lies the crux of the OIS security problem—the "guarantee" of security, taking into consideration the unique operating environment and constraints of an electronic office. It is only within recent years that some answers to this problem have begun to emerge and that these concerns have gained recognition as a separate discipline of serious intellectual challenge [15].

Researchers have pointed out that automated office systems are emerging as an interdisciplinary research area with a strong computer science component [12, 30, 11]. In fact, OIS can be viewed as an attempt to perform the functions of an office by means of a computer system. These systems can consist of a large number of clusters of a few computer devices, often separated geographically and managerially; or they may consist of simply a few computer devices interconnected within the confines of a single office. This forces us to look at the provision of security at two levels: both within the system existing at a single local site and within the network created when such sites are interconnected.

The control of information within a distributed OIS is best accomplished by viewing the system as a set of active subjects and passive objects. Subjects are entities that cause action to occur in a system. They are active by nature and include processes and users. Objects are passive entities which are acted upon by subjects. They include files, processes, memory segments, and disk space. Note that a process can be a subject and object. Within the OIS environment we are most concerned with users and processes as the active subjects and documents, files and messages as the passive objects.

The interconnected OIS itself is a somewhat different network from that normally found in the traditional computer network model. Here we may find each geographical location on the network consisting of a series of heterogeneous devices interconnected on a local area network (LAN). This LAN may then be connected to other LANs some distance away via a wide area network (WAN). Each OIS may have a requirement that part of its information holdings be kept physically or logically separate from network access or even a requirement that a particular information repository be accessible to an identified set of network participants. This separation will likely be accommodated by a combination of hardware controls, encryption, and trusted software.

2. Areas of security concern

In any computer based system, large or small, the security issues can be partitioned into those affecting hardware facilities, those most concerned with the software construction, those involving inference, and those of personnel. The hardware security issue is by far the most documented and understandable. Users of a system, computer literate or not, seem to easily grasp the need for physical protection of the hardware that stores and manipulates information resources. The requirement to prevent electronic emanation is readily understood and accepted in government intelligence applications and is often referred to as "Tempest" protection. These topics can be related by users to protection issues encountered in every discipline and accepted as necessary to insure property preservation and equipment availability [27].

The security problems requiring software control represent the antithesis to the above in terms of user understanding and acceptance. This is particularly true in applications like OIS where users are not well oriented in software and simply use the system to support their work in some other discipline. These users may readily accept the need to lock up the data disks when not in use, but fail to comprehend the significance of adding new untrusted software to their system.

The inference problem reflects a concern that classified information can be "inferred" by validly obtaining unclassified information from the system. This can be accomplished by manipulation of specially contrived queries into the system's database or files, or through browsing, or through simple aggregation of data.

Consideration of the personnel that use the system, service the system, and those that are serviced by the system play an important role in the establishment of a security environment and must not be overlooked.

2.1. Hardware security

This security issue can be viewed as three distinct (yet supportive) categories of protection: physical, equipment shielding, and cryptographic. Each area has been quite well stated in the literature and although the issues presented must be considered in designing an overall secure architecture, they are not seen as current areas of research for the purposes of this paper.

2.1.1. Physical security

Physical security is certainly the best-understood protection measure and the most readily accepted. It encompasses such solutions as guards, walls, locks, key entry systems, uninterrupted power supplies, back-up or archival files, fire protection systems, disaster recovery procedures, etc. These measures and others are well documented in almost any computer security text written during the past 10–15 years. The need for these protections is real regardless of the system size, function, or architecture. These protection measures are commonly implemented through the organization's standard operating procedures, rules, and informal policies.

2.1.2. Equipment shielding

This protection category focuses on the electronic characteristics of the hardware and can be typically viewed as both the prevention of certain information carrying signals from emanation beyond a defined geographical area, and the prevention of damaging electronic signals from entering a defined protection space.

The first of these problems (electromagnetic eavesdropping, EME) is the property of electronic waves emanating from their source (e.g., a computer, coaxial cable, copper wire). These waves can be received at various distances with special equipment and then interpreted. The solution to this problem is one of prevention by shielding those devices that are known to emanate. Optic fiber technology offers significant assistance in this regard since it is based on data transmission by light-wave modulation and therefore cannot emanate. It is important to note that this problem attracts no significant interest outside the classified areas of the federal government.

The second shielding problem identified has an opposite focus—that of preventing damaging electronic waves from entering the computing and transmission environments. The major issue addressed by this category is shielding against electromagnetic pulse (EMP) which may be generated by a high-altitude nuclear explosion. Technical solutions exist which involve equipment shielding and use of various filtering techniques. This paper will not further discuss this particular problem as its applicability to the office information systems environment is seemingly minimal in terms of need or interest in providing this type of protection. The issue is included only for the sake of completeness.

2.1.3. *Cryptographic techniques*

The interconnection of systems implies the transmission of data across various communication media. It is logical to assume that at least a part of this information transfer involves private information and should be protected while in transit. This protection is afforded by encrypting the data at the sending host location and reliably decrypting the same information upon its arrival at the receiving host site.

Standards exist for encryption and the cipher algorithm can be implemented in a single hardware chip. This is seen as a major advantage for office automation systems. A possible encryption algorithm for OIS use is the Data Encryption Standard (DES), which was developed by IBM and adopted by the US Government in January 1977 as the official encryption standard for unclassified information. Its major advantages are speed, low cost, and its implementation in a single chip [29]. Cryptographic techniques may also be employed to provide the very useful functions of electronic signatures and repudiation (e.g., the inability to deny having sent an electronic message).

2.2. Software security

The controls necessary in the provision of OIS software security are subject to constraints not normally imposed on hardware security mechanisms. First, the controls must be somewhat transparent to the user. This means that the system should not be intolerably slow; it should not require users to attain system expertise that greatly exceeds that required to run the system being protected; and the overall reliability of the operating software should not be degraded due to involvement of the control mechanisms. Secondly, the controls themselves must be considered part of the system's trusted computing base. This trust can be achieved through application of both informal and formal verification techniques on the control software or the use of trusted, evaluated products. It must then be accorded a privileged operation mode above any other in the system lest the trusted software be subverted by untrusted code. This status might be accomplished through the use of a security kernel buried within the operating system which runs in a protection level higher than the supervisory software. The decisions involved in the design of the kernel and what software to include are certainly non-trivial tasks and the subject of current industrial research.

Through these software controls a defense is established against a number of classical security problems. The majority of these problems are discussed within this section and include such issues as authentication of users and terminals; attempts to deny service to valid users; undesired aggregation of data within a system; user browsing with the intent to gain access to information not meant for public dissemination; the unauthorized copying of data or files that are meant to be protected from such action; the intentional transfer of information through covert channels to subjects not cleared to receive such data; the prevention of Trojan horse attack which involves hidden functions within useful software that surreptitiously exploit the legitimate authorizations of the invoking process, or other forms of malicious code attack; and lastly, the issue of multilevel security where we allow users of various security classifications access to the system and then trust the system to segregate the users and data [19, 20, 23].

The above problems were selected from available literature because they represent active areas of research in large scale computer systems and networks, but each is a problem in its own right within the related discipline of office automation. Solutions that work for traditional data-processing environments may or may not be suitable within the OIS environment.

2.2.1. Distributed authentication

In an office environment, any individual who can successfully impersonate someone else will be able to acquire all the capability which that someone was authorized. Authentication deals with the verification of claimed identity. According to Israel and Linden [16], authentication can be considered as the foundation on which access controls, audit measures, and most security and accounting controls can be built.

In a manual system we have little difficulty authenticating the person we are communicating with. We generally know who is authorized to receive various kinds of information and who is allowed access to specific files. If we are directed to take some action we authenticate the direction authority by sight, signature, or voice, and have some reasonable assurance that we are responding appropriately. If the direction we receive has legal or monetary implications, we normally retain some proof of our having received an order to carry out a specific action (e.g., withdrawal of funds from a bank account or purchase of stock). In an automated system we must provide for the same assurances. This requirement is quite difficult in itself and is made even more difficult when the system is distributed such as it might be in an OIS environment. Two major problems exist. First we must provide for the proper identification of the person desiring to use the system, and second we must build into the system some highly reliable, trustworthy mechanism that guarantees the identity of the subject with whom we are conversing (authentication). It is important to note that we may wish to identify both user and location in the office automation context.

The traditional manner of solving the identification problem is to employ the use of login ID and passwords. Normally it is necessary for the system to store internally the user ID for each authorized user as well as some representation of the password, security clearance, and perhaps access authorizations that are associated with each user. Many implementations require encryption of the password database as a minimum level of protection in addition to imposing various access controls on the database itself. This is particularly important within an open network system where passwords could potentially be transmitted in clear text. Furthermore, one must be concerned with the ability of a hostile third party to intercept the communication between the user and the authentication server. Successful interception might then provide for the capability of recording and replaying the legitimate user's login session, or perhaps allow the intruder to impersonate the legitimate service being requested, or even permit harmful, passive analysis of traffic over the established communication paths. Given the fact that cryptography solutions are now reasonably economical, consideration should be given to protecting the entire session. Safeguards can be implemented for each of these concerns but require some additional overhead on the system and a substantial use of encryption. Several scenarios involving distributed authentication offer potential solutions as described below.

2.2.1.1. Application of cipher keys

One technique involves the establishment of a distributed authentication service which resides in the system as trusted software. This authentication service maintains the passwords of each authorized system user and their respective cipher keys. If we can guarantee that each node in the system has its own unique cipher key which is known only to the trusted authentication service, then communication between any two nodes must be mediated by the authentication service. This service would become very inefficient if it was required to participate fully in every communication session, so a more palatable approach is as follows. If node A of our system desires to establish a session with some node B, it must first authenticate itself to the system (probably via password). Once this level of authentication is satisfied, node A can then access the authentication service and request establishment of communication between itself and node B. Once at the authentication service level, the potential exists to enforce both discretionary and nondiscretionary access controls if desired by the system administrator. Regardless, the authentication service can then allow (or disallow) communication between A and B by providing (or denying) a common cipher key for use by both A and B. This key's usefulness is limited temporarily to avoid its unrestricted use. It is not necessary for the authentication server to communicate directly with B and, in fact, the system is more secure if this does not take place. Instead, the authenticating service will provide a set of "credentials" for presentation to B. These credentials will be encrypted with the cipher known only to B and will contain, as a minimum, A's identification, a time-stamp, and the cipher key used in communicating with A. When B receives the credentials, it can decrypt them and use the provided key to respond to A's initiation. A workable communication path has now been established with level three authentication in that full encryption now exists in both directions. This system is somewhat robust in that the authentication service is distributed, and if one server is unavailable for any reason another may be used, although some degradation in service might be experienced. If this technique is attempted in an office system environment, one quickly finds a major disadvantage in that not all devices in such a system can support the encryption/decryption requirements of this technique. In such a case it appears that a system of various authentication methods may have to be employed.

2.2.1.2. *"Big Brother" technique*

A related technique introduces the concept of a single, trusted "Big Brother" (BB) authority within the system that plays a more active role in communication between subjects [24]. In such an implementation we would expect each network participant to select their own secret key and securely provide it to BB. If subject A wishes to send a message to subject B, it would first notify BB of the intended communication and request an appropriate secret session key, $K(s)$. This key would then be returned to subject A in two copies: one encrypted with A's secret key $K(a)$ and one encrypted with B's secret key $K(b)$. Subject A could then elect to send $K(s)$, encrypted with $K(b)$, directly to B with instructions for B to decrypt it and establish a session. This method can then be expanded to provide a trusted signature service between users by following a protocol detailed in [29]. This technique can be quite cumbersome in a large network, but is potentially workable in a medium or small system with relatively light traffic. Two disadvantages are inherent in this technique. First, the BB server is a central facility and its loss or compromise means that all secure authentication between users must terminate. Secondly, the success of such a system is predicated upon the rather weak assumption that a secret key is known only by its owner and the BB server. We can reasonably assume that a protection environment can be established for BB, but if each user is in control of his own secret key security and if any one user allows a compromise, the system can be subverted. (In today's environment there may not be any requirement for a user to be in control of a key if the key is installed on chips which cannot be easily probed.) The BB server must also be concerned with retaining copies of all old keys that it has used during its lifetime and audit trail information reflecting key change activity and service history.

2.2.1.3. *Use of digital signatures*

A third technique often used is that of digital signature using public key cryptography [29, 24, 8]. This technique can be attributed to the work of Diffie and Hellman who developed it in 1976. They proposed the use of an encryption algorithm, E, and a separate decryption algorithm, D, where E and D are chosen such that the derivation of D given E would be impossible in a practical sense. For a proper digital signature system, four requirements must exist:

(1) $D(E(P))=P$ (where P represents an arbitrary message);

(2) $E(D(P))=P$;

(3) it is effectively impossible to deduce D from E;

(4) E cannot be broken by a chosen plain text attack.

Using this technique, all participants in the system can publish their public key E to all others and uniquely retain their private key, D. Given such an environment, users can securely communicate across the network by using each other's public keys or can digitally sign messages by using a combination of public and private keys.

Authentication of users, devices, and senders is a critical requirement in any secure system and must be incorporated in interconnected office information systems that process sensitive information or actions based on distant requests that involve legal or monetary transactions. There are many possible approaches to authenticating users in a small office information system; however, as the system grows larger and more diverse, authentication will be more difficult and may become a large administrative burden. In any case, a useful system is expected to efficiently send a "signed" communication to one or more addresses such that the following two characteristics are present:

- All addresses can reliably verify the identity of the sender.

- The sender cannot deny that he sent the message.

2.2.2. Denial of service

The DoD Trusted Computer System Evaluation Criteria [10] fails to address this problem adequately, although this may be an extremely important requirement for many interconnected systems, whether they are traditional networks or OIS. Imagine the financial disaster for a major airline company if an attack results in denial of use of its reservation database, or the impact on the credibility of a stock brokerage company if customers' buy and sell transactions fail to reach their intended source in a timely fashion or even at all.

The major problem in protection against denial of service attacks is to determine that one is actually occurring. Two types of attacks may occur—active or passive. Active attacks are quickly noticeable by the network involved and include such blatant activities as electronically jamming the communication path, physically capturing a node within the system, cutting transmission medium, or causing a power loss. These deliberate types of attack are not seen as a major threat outside of a warfare environment and are of little concern within an office information system.

The more difficult type of attack to contend with is the passive one. Here it is not obvious that the attack is occurring and it may take some indefinite amount of time for users to exercise proper countermeasures. The most common and effective method for conducting this type of disruption is for the intruder to monitor the communication path and to occasionally alter messages so that they are received in an unintelligible format or not at all. (The altering of a message is an integrity problem, but if the alteration renders the data less useful to the recipient it may be considered a form of denial of service.) Numerous techniques exist to perform this altering process and range from altering all messages (quickly detectable) to altering only selected messages (difficult to detect). When combined with traffic analysis procedures, a third party can wait undetected until messages of a certain type or addressed to a specific party are encountered, and deny their delivery.

Countermeasures can be built into a network to provide various degrees of protection against such attacks. The two most effective measures are first to ensure a periodic message exchange between peers on a system to confirm that an open channel exists, and second, implementation of redundant transmission of packets through multiple network paths. Other methods include various forms of cryptography and use of an intrusion-resistant communication medium such as fiber optic technology. The true protection against denial of service attacks lies in the system's ability to effect early detection.

In an office information systems environment this threat must be taken into account as a security concern with protection measures oriented toward passive attacks.

2.2.3. Unauthorized copying

In a normal office environment sensitive documents are often protected from copying by locking them in safes, file cabinets or some similar device, and then by controlling access to the documents. In some cases copied documents are numbered to reflect which copy they are of a set of copies and all such copies are periodically accounted for. Other physical protections may exist to further foil attempts at unauthorized copying. When we move this paper analogy to an automated environment we often lose all our protection, and in fact copies can be made much easier and certainly much faster. Thousands of copies of a document can be electronically created and routed to distant destinations with only a few key strokes.

A literature search has revealed specific countermeasures to this problem; however, it must be addressed in an OIS environment. Unauthorized copying can be reasonably safeguarded against by combining audit system controls and user-initiated file encryption where a user can uniquely encrypt a file to prevent its use by another user. Obviously problems exist with such an approach since copying is occasionally desired; therefore, we must somehow deal with the problem by controlling the number of copies existing in the system at the same time and the prevention of forwarding a copy received from a distant source if that forwarding would violate either discretionary or non-discretionary rules in the system. It would appear that such a protection system could be constructed within a trusted monitor process. Denning [7] advocates the use of a "copy flag" which, when present, authorizes the user to forward (or copy) the document and if absent the user can only read the document. All such countermeasures will add to the overhead incurred by the system and this overhead must be kept to an absolute minimum in order to achieve user acceptance of the final solution.

2.2.4. Confinement problem

This problem may not be totally solvable in either the standard operating system or the OIS environments. Lampson [18] outlined what he called the "confinement problem" which, briefly stated, is the prevention of a program from retaining and relaying classified information to a process other than the one that invoked the program itself.

The use of flow controls can help somewhat in checking information flow at a low level, but they alone are insufficient to act as a complete countermeasure. Lampson described three separate channels of possible information flow:

(1) *Legitimate channels*: these are the channels of information flow one normally associates with process to process communication. It could be, for example, the printed output from a program or the parameters of a procedure.

(2) *Storage channels*: these channels are utilized when information is passed through shared files, global variables, or temporary files. Data are stored somewhere in the system by one process and accessed by another process.

(3) *Covert channels*: sometimes referred to as "timing channels," these paths of information flow were never intended for information transfer. They can be used surreptitiously to pass information to another subject without detection. An example might be using the frequency of system calls to "tap out" a message to a third party or setting a global parameter to a 1 or 0 at various time intervals to code a message in binary. It is possible to vary line spacing in a written report or to vary paging frequency to pass information.

Legitimate channels are reasonably secured by existing techniques. Storage channels are somewhat more difficult to secure but through the use of low level flow controls they too can be safeguarded.

The real difficulty in solving the confinement problem lies with the detection and prevention of covert channels. Cost-effective methods of closing all such channels most likely do not exist. The question of whether or not the confinement problem is a worthy issue in OIS security remains to be answered.

2.2.5. Trojan horses and other malicious code

The definition of a Trojan horse in a computer security system environment seems to take as many forms as there are writings on the subject. For the purpose of this paper, we will use the DoD version [10], which defines a Trojan horse as a computer program with an apparent or actual useful function that contains additional (hidden) functions that surreptitiously exploit the legitimate authorizations of the invoking process to the detriment of security. It appears that this kind of threat can take four distinct forms that we must concern ourselves with [20]:

(1) *Surreptitious copying*: the copying of sensitive information from a file to another location where a second party can access it at a later time without the approval or knowledge of the owner.

(2) *Automatic bypass*: a hidden, on-demand ability to bypass selected security codes without the system's knowledge (e.g., by bypassing the system's authentication procedures).

(3) *Operating system masquerade*: the intercepting and acquiring of user requests meant for the operating system and the return of a valid operating system response to the user from the Trojan horse. This technique can be used to steal a password or gain access to a protected file.

(4) *Malicious destruction*: the unauthorized destruction of data rendering it unavailable for use (either temporarily or permanently).

Various writings have appeared on this subject, with most agreeing that the total prevention of Trojan horse attacks is not cost effective. A better approach to defend against this threat is to detect an attack or reduce the potential damage to an absolute minimum.

A few years ago this type of attack was only considered in large-scale, multi-user systems. Office automation mostly consisted of word processors, and perhaps a telex device, and as such was not really susceptible to this problem. The rapid decline in hardware costs coupled with advances in OIS software and its utilization in an MS-DOS environment have created an environment that is extremely vulnerable to the Trojan horse attack.

True protection from such attack might be achieved if all software were to be formally verified, but this becomes expensive very quickly. Additionally, in an OIS environment software is generally purchased from an outside (untrusted) vendor, it changes frequently with new releases and new packages, and it normally contains a capability for local, user modification. It would be necessary to reverify the system (or at least the new software) upon every change. This procedure is of course not practical. In most cases a vendor would be unwilling even to consider such constraints. If such verification were not accomplished, a new release of existing software could potentially contain a particularly dangerous form of Trojan horse known as a virus program, in which the Trojan horse embeds a copy of itself in other programs, thereby infecting the formerly clean system. The detailed issues related to *computer viruses* are beyond the scope of this paper. Our readers are encouraged to see Peter Denning's timely new work entitled *Computers Under Attack: Intruders, Worms, and Viruses* [9] for a comprehensive discussion of viruses. Denning's book collects some of the most informative, provocative, and frightening reports on the vulnerability of computer systems to harmful, if not, catastrophic, attacks. The articles collected in this book are a pointed warning that our computer systems are already under attack, that the privacy and integrity of information in our personal, business, office, and research activities are seriously threatened and that the security of free societies is on the line. When one considers the current interconnected architectures found within the OIS environments, the magnitude of this problem is quickly seen. The dangers of an Internet-like "worm" attack should also be noted here [25]. (A worm is a program that runs independently and consumes the resources of its host from within in order to maintain itself. A worm can propagate a complete working version of itself onto other computers.)

Solutions to many specific types of attack do exist and these may be of some value in designing a protection system within an OIS. Early encryption of data and files with a key known only to the user may lessen the risk of an agent gleaning useful information after illegally accessing a file. Denning's work [7] with a lattice model of information flow within a system seems very likely to offer detection mechanisms. The automatic bypass problems might be approached by having the trusted portion of the system verify that an application has in fact not bypassed any critical stages of system entry. A way of accomplishing this might be to attach a particular bit stream to the user's password as it passes through various entry modules and then to check this bit stream when the subject attempts access to sensitive objects. This technique is sometimes referred to as entry transaction coding. The reference monitor abstraction again offers some protection from a Trojan horse attack in blocking attempts to write information from one security level to a lesser one. (One should not overlook what may be the most effective protection mechanism against malicious code attacks in an OIS environment—that of procedural control coupled with education.)

2.2.6. Multilevel security

A multilevel secure system is a class of system containing information with different sensitivites that simultaneously permits access by users with different security clearances and needs-to-know, but prevents users from obtaining access to information for which they lack authorization. The key concept here is that the system itself can be trusted to separate users from data they should not have access to. Many systems touted as being secure are in fact a class of system known as "system level high", where it is assumed that everyone having access to the system has a security classification at least as high as the highest level of information stored on the machine. In many applications this sort of an assumption is valid, but it seems risky within an OIS environment, especially if the OIS is interconnected with other systems. The logical staffing around an OIS would include clerical personnel, intermediate supervisors, and strategic management. One would expect these users to have various levels of security clearance and trustfulness, and not each of them should be able to access or see all information existing on the system. The multilevel secure system concept now being researched by industry for large-scale general-purpose or communication systems seems an ideal approach for OIS if it can be designed for such environments. We considered the design of such a protection environment by approaching security at two levels: the individual node level and the network level [31].

2.3. Information acquisition control

An experienced system user can, in some cases, circumvent existing security controls to extract information that would normally not be provided due to existing policies. This may be done in a variety of ways, but in general these techniques can be partitioned into areas: aggregation of information, and browsing. In each case the user obtains unprotected information from the system and uses that information to discover classified or protected information that should not have been released.

2.3.1. Aggregation problem

In a manual system one must guard against the searching through of all available files and correlation of information found, because often the aggregation of data may require a higher security clearance or protection than do the individual data elements themselves. In an automated system this problem is much more difficult to control due to the speed of information retrieval and the privacy afforded the individual querying the system.

In an OIS environment we should protect against this type of problem and give it the same level of interest as we give the subject of security itself. Certainly, a strong audit trail and reporting system will be a major factor within an OIS, but its detection of aggregation efforts will be difficult for all but the most skilled observer to locate. It appears at this point that we must expect some reliance on a trusted system administrator in any aggregation countermeasures.

2.3.2. Browsing problem

In an non-automated environment it is likely that an individual doing excessive browsing through numerous files would be noticed and questioned. Unfortunately, within a computer based system this same activity can go on without notice for lengthy periods. This is especially true in systems that have a transaction rate measured in hundreds per second. Audit trails offer some protection from this activity but only after the fact, and will usually require some sort of expert system audit reduction tool to assist in locating security-relevant events of interest. The appropriate counter-measure to the browsing problem would appear to be the establishment of strict, reliable "need-to-know" controls within the OIS. In standard secure operating system design, an access matrix or capability list is sometimes used to provide for need-to-know authority for subjects desiring access to objects in the system. A form of this control seems potentially usable in the OIS environment.

It seems reasonable that a complete countermeasure to this problem will involve three distinct approaches. First, the aforementioned expert system assisted audit trail will be essential. Assuming the organization has a trusted individual that screens machine-generated audit reports, it is likely that the browser will be detected early and investigated. Browsing is typically an ongoing activity over several days since the browser normally doesn't know exactly what to look for and therefore attempts many queries into multiple files. This kind of activity is quickly detectable in audited systems. Second, the software management system (i.e., operating system) most probably will need to arbitrate access by subjects to objects through the use of discretionary access controls implemented by either an access control list or a capability list. A third approach appears to be more oriented toward the physical security area and would involve the logical positioning of access terminals throughout the organization in such a manner that excessive browsing would be most likely noticed by fellow workers. Alternatively, consideration may be given to limiting a physical terminal's access to a particular set of files through hardware/firmware controls. Used together, these three measures appear to be adequate to safeguard reasonably against the browsing problem, which remains unsolved on the whole today.

2.3.3. Personnel

For the sake of completeness we add this final security concern. The thoughts presented here are our own and present no particular reference. We must be concerned about three categories of individuals: those that use the system, those that service the system, and finally those that are serviced by the system. (We also believe, from an intuitive basis, that the ordering of the three groups above represents an ordering from greatest to lowest threat in most cases.) These three categories of individuals can be referred to as *users, maintenance,* and *clients* respectively:

(1) *Users*: users of an office information system can be categorized as clerical/secretarial, intermediate management/supervisory, or strategic management. It seems logical to assume that they each have various degrees of trustworthiness and have a need to access information of different sensitivity levels. The degrees of trustworthiness problem can be alleviated to some extent by proper screening of personnel employed by the enterprise, but this is often an expensive and ineffective solution. More often, we might expect to see a hierarchy of security clearances within an organization and the assignment of a particular clearance to an individual. One would expect to begin at the low end of the hierarchy and progress upwards as trust is

gained in the organization. A multilevel secure system can accommodate such a hierarchy and guarantee separation of users from data they are not cleared to see.

(2) *Maintenance*: those personnel servicing the system represent a potential threat that is often overlooked. Office information systems typically are serviced by organizations outside the enterprise. Service personnel may have free access to internal information-holding hardware components (e.g., memory chips or hard disk drive) or to the software itself (e.g., operating system or application programs). Internal administrative operating procedures are the best defense against this threat.

(3) *Clients*: those personnel serviced by the system represent a threat in terms of denial actions and impersonation. The system must have mechanisms (in an interconnected environment) to guarantee the identity of the distant party and later to prove certain communications were actually received from that party. These subjects were discussed at length in section 2.2.1 of this paper.

3. The office information system environment

One does not fully automate an office. There are simply too many tasks and interactions inherent in such an environment to automate. Automation best applies to static and deterministic processes and these usually can be categorized within an office as functions such as filing, mailing (or routing documents/messages), word processing, or scheduling (e.g., calendars). If these functions can be automated, one can greatly increase the efficiency of information flow and retrieval. It is in this context that we shall try to create a secure environment. The security lies in the storage, access and flow of electronic information—not in the security of the office as a whole.

What is an office information system and why is it any different from the standard computing environment so familiar to us? Many might agree that an OIS is an attempt to perform the functions of an office by means of a computer system. This task is far from simple and, as stated in [12, 28], solutions to a large number of difficult problems must be obtained before such systems can reach their full potential. Four such difficult problem areas that must be examined are: the complexities of distributed systems that are implemented within the OIS [12, 13]; the integration of knowledge-based systems into OIS; the provision of good user interfaces [21]; and the security of the working environment. If the last problem—security—is not solved, the effectiveness of a good OIS may be severely degraded and even the future usefulness or acceptance of such systems may be in jeopardy. OIS are different from standard computing environments in many ways and thus measures to secure the systems become complicated. A data-processing system is used to implement algorithms with a single locus of control. There may be little human interaction. An OIS, on the other hand, is generally highly interactive, with some number of autonomous tasks executing in parallel. Some would view OIS as a distributed operating system with a highly refined user interface and database facility. The environment of an OIS poses special problems to those attempting to secure it. We have chosen six examples to illustrate this point. Each is discussed below.

The first example is that of personnel. The people using an OIS use it as a tool to perform some other main task. They have no (or at the most, very little) computer background and probably would not be afforded lengthy training sessions on such topics. They fully expect the system to support the office and would not accept a system that required an office to support it. They demand a highly efficient user interface that is easy to use physically and easy to grasp conceptually. If these two requirements are met, the system provides the correct perception and stands a high probability of acceptance. The system is expected to conform to the paper analogy that has been in the office place for a few hundred years. System designers and analysts sometimes find this difficult to work with, but the successful systems have been those that provide a familiar environment to the user (e.g., use of icons and desktops). The office personnel supported by OIS are likely to fall into three general categories: clerical/secretarial; middle management/supervisory; and strategic management. Obviously, the needs and expectations of these three groups will vary considerably, but of more importance is the security problem they pose. They are quite likely to have different levels of trust within the organization and different levels of access to various kinds of information. Users expect the system to provide the necessary access controls to safeguard their information just as they would if they locked up paper documents in a safe each night. This problem is further exacerbated if users choose to join a network of any kind.

The typical growth of an OIS presents additional problems. Nolan suggests that an organization moves through four stages of growth as its information systems mature: initiation, expansion, formalization, and maturity [33]. (The Nolan model has been revised several times to a six-stage version. Our usage is based on the original definition.) It is stages two and three that cause problems for us. In a typical data processing environment we might expect to see formalization as stage two, followed by a careful and deliberate expansion of the system. Within OIS we tend to see a stage of rapid expansion, with little prior planning. This rapid growth seems spontaneous in some cases and is characterized by the introduction of heterogeneous equipments. Stage three seems always to come after the growth and occurs as it becomes apparent to management that the expansion is lacking in cohesiveness. If one adds a security structure to the OIS, this expansion must and will be slowed to conform with security requirements because, if we assume Nolan's stage theory to be correct, a security structure and policy will not allow expansion without first achieving formalization.

Software used to implement an OIS is most likely purchased from a private vendor. Unlike traditional data processing operations, there is little or no application programming and there may not be any in-house capability to perform software maintenance. A great reliance is placed on off-site vendor support. Given these constraints, our security architecture becomes more difficult to implement. We cannot expect verification of all application packages and thus we must be concerned about the possibility of a malicious code. Additionally, we must allow the rapid integration of new software packages without stringent testing. Integration becomes easier as open systems environments become standard.

System support typically comes from an external source. Often the manufacturer of a turnkey system will offer this type of total support, but in other cases a private contractor may be asked to provide it. Security problems surface when a maintainer runs private software (perhaps diagnostic) on the system, or copies stored information, or perhaps removes a hardware component for repair (memory board or fixed disk). These security challenges could be quite serious within an OIS, but must be dealt with by a combination of office operating procedures, security clearance for maintenance technicians, and use of trusted personnel. This research will not focus on these issues since they are more in the realm of physical security.

During rapid growth of computing environments, it is common to find a desire to interconnect the OIS to other OIS or computing systems or to a network interconnecting such systems. This is beneficial and should not be discouraged. The challenge comes in providing a guarantee that each separate OIS can control what information is made available to outsiders. Although this problem is far from solved, it would seem prudent to establish a distributed reference monitor to mediate access between interconnected systems [13, 14, 26, 32]. Interconnection of OIS also gives rise to a second concern—that of types of systems that we should allow connection to and types that must be disallowed in order to maintain the security level of the system and perhaps the network as a whole. It is imperative that one does not degrade the system's level of trust because of the connection itself. This problem is somewhat akin to the "weakest link in the chain" problem. When interconnection takes place one has to place some reliance on the security capabilities of the distant system and trust it to exercise the same rigid controls that we expect.

Lastly, authentication is a special problem within OIS. Recalling that the OIS is working from a paper analogy, it is reasonable to assume that there are definite requirements for evidence of delivery when forwarding various documents and there are legal requirements to prove that electronic signatures are in fact genuine.

4. Design of the secure office information systems

An existing system is not normally modified to make it secure. The accepted way of acquiring a reasonable assurance that the system is secure is to design and build it according to some agreed-upon model. If the model can be proved secure and if the system can be proved to follow the model, then we have increased our confidence in the ability of the system to safeguard the information entrusted to it. There are three general categories of model for use in secure systems: the access matrix model, the Bell and LaPadula model, and flow control models [19].

4.1. Access matrix model

This is one of the more popular models and is actually a protection system based on an operating system concept of an access matrix where access to objects by subjects is regulated by table look-up. The model is defined as a three tuple (S, O, X) where:

(1) S represents a set of subjects which are active entities in the system such as users, processes, and domains.

(2) O represents a set of objects which are passive entities such as files, programs or memory segments.

(3) X is an access matrix in which the rows represent subjects and the columns represent objects. An entry at point $X(S, O)$ represents the access right that S has to O.

The access rights present in the matrix may be, for example, read, write, append (allows one to add data to the end of an object but not to overwrite), search (for directory entries), and execute. During any instance in time the subject executing is associated with a particular set of access rights. This set is known as its "domain" or the protection environment in which the subject is executing. Obviously the domain is not static since access rights can be issued dynamically, so it may change during execution.

Use of an access matrix implies the use of a monitor to mediate access of subjects to objects during execution. A key point here is that the access matrix and monitor concept seems suited to the OIS environment due to its granularity of object control. In OIS applications we would appear to be concerned more about the protection of files, documents, and other whole entities than we would for actual program information flow at a fine granularity. (It is important to note that the access matrix as described so far is conceptual. Implementation of this concept requires some changes to avoid using large amounts of storage to contain a matrix that is sparse for the most part. One can easily imagine the large number of subjects and objects present in a system and the dynamics of their interaction. A true matrix implementation would be much too difficult to manipulate and far too large to occupy limited storage space. In practice the access matrix is implemented in one of two ways—either string the matrix row-wise, where for each subject a list is maintained of objects it has access to, or column-wise, where each object has an associated list of subjects that have access to it. The row-wise implementation is referred to as a "capability list" and the column-wise approach is known as an "access control list". Like all implementation choices, there are advantages and disadvantages associated with either technique, but regardless of the representation chosen the greatest advantage is the underlying simplicity and understandability of the access matrix in general.)

A popular concept used in many secure system designs is that of the reference monitor which was first introduced in [17]. The reference monitor is an abstract concept in which the monitor exists in trusted code and has as its function the complete mediation of all accesses to objects by subjects. Typically it is viewed as interacting not only with sets of objects and subjects, but also with a database of rules that represent the non-discretionary access controls enforced upon the system.

The access matrix associated with a particular system would exist in the reference monitor code to be referenced for every subject request arriving at the monitor. The monitor's decision process for authorizing access would be based on the current projection state of the matrix and the request would be approved or denied accordingly. Requests by subjects to change the protection request would be seen as discretionary access actions and would be checked by the monitor through its access to the non-discretionary access rules database. If the requested action is in compliance with the non-discretionary rules it is allowed, and if not, it is denied.

Using the access matrix model within a secure system generally implies that the matrix be located within a trusted perimeter of software. We refer to this software as the security kernel. The notion of a security kernel is based upon the work done by Lampson [17] in describing a reference monitor whose purpose was to mediate access of subjects to objects within the system. Three principles of a successful security kernel were specified [1]:

- *Completeness*: all accesses by subjects to objects must be mediated by the kernel.

- *Isolation*: the security kernel must be reasonably protected from tampering.

- *Verifiability*: correspondence must be shown between the actual security policy to be enforced and the implementation of the kernel. This is perhaps the most important characteristic.

4.2. The Bell and LaPadula model

The model chosen by many recent secure systems is the Bell and LaPadula (BLP) model which was developed by the Mitre corporation in 1973–1974 [4, 3, 2, 5]. It is a formal state transition model of computer security policy that describes a set of access control rules. In this formal model, the entities in a computer system are divided into abstract sets of subjects and objects. A system state is defined to be secure if the only permitted access modes of subjects to objects are in accordance with a specific security policy. In order to determine whether or not a specific access mode is allowed, the clearance of a subject is compared to the classification of the object and a determination is made as to whether the subject is authorized for the specific access mode. Four modes of access are permitted in the matrix: *read only*, *append*, *execute*, and *read–write*. The model asserts the principle that no operation may change the classification of an active object (tranquillity principle). Two properties, as shown below, are enforced that ensure the system cannot transition to a non-secure state. These properties are specified in terms of a function F, where F_s give the clearance associated with a subject, F_o gives the classification belonging to each object, and F_c gives the current security level for a subject:

- *The simple security property*: a state satisfies the SS property if, for each element of B that has access mode *READ*, the clearance of the subject dominates the classification of the object. That is, (s, o, x) satisfies the SS property relative to F if $x = \mathit{read}$ and $F_s(s)$ dominates $F_o(o)$.

- *The *-property (star property)*: sometimes referred to as the "confinement property", this set of rules is designed to counter the Trojan horse problem as discussed in section 2.2.5. Three rules describe this property:

(1) A subject may not have *WRITE* access to an object unless the classification of the object is greater than or equal to the current security level of the subject. That is, (s, o, x) is OK if $x = write$ and $F_c(s) \leq F_o(o)$.

(2) A subject may not have *READ/WRITE* access to an object unless the object's classification equals the current security level of the subject. That is, (s, o, x) is OK if $x = read\text{-}write$ and $F_c(s) = F_o(o)$.

(3) A subject may not have *READ* access to an object unless the object's classification is less than or equal to the current security level of the subject. That is, (s, o, x) is OK if $x = read$ and $F_c(s) \geq F_o(o)$.

These two properties are sometimes simplified into two easy to remember rules: "no write down" and "no read up."

Because the model is extremely restrictive, the concept of a "trusted process" was introduced. This is a process (or subject) that is trusted not to compromise security and, as such, it is allowed to violate the two properties just stated. Obviously, such a process is needed to handle such important functions as reclassification of information, sanitation procedures, and downgrading.

Within an OIS, the BLP model might potentially form a good basis for a start in designing a security model. It has the advantage of working at the object level, it is easy to understand, involves a kernelized approach, and allows limited use of trusted processes. Disadvantages include the potential for overuse of the trusted process and the severe restrictions imposed by the model which could often disallow otherwise secure operations.

4.3. The information flow model

This particular model is largely attributed to Dorothy Denning [6, 8, 7]. Its need arises from the fact that access controls, such as those exhibited in the Bell and LaPadula model, regulate access to objects by subjects but do not concern themselves with the flow of information once access is granted. This makes access control models somewhat vulnerable to covert channel attack (in particular, storage channel attack). Information flow models, on the other hand, focus on the individual operations that transfer information within the system. This focus is at a much greater granularity than that provided by the access control models. The flow control models are sometimes mistaken as a replacement for access control models or are assumed to be a new attempt to solve an old problem, but in reality they were developed as an extension to access control and are intended to be used in conjunction with such models.

The model was first introduced by D. Denning [6] although references to this concept first appeared three years earlier. As taken from the above reference the information flow model is defined as a five tuple: $FM = (N, P, SC, \oplus, \Rightarrow)$ where:

- $N = \{a, b, \ldots\}$ is a set of objects representing information receptacles. The elements of N determine the level of detail that one wishes to exercise flow control and can represent files, segments, program variables, or even bits.

- $P = \{p, q, \ldots\}$ is a set of processes which are described as active agents responsible for all information flow.

- $SC = \{A, B, \ldots\}$ is a set of security classes which correspond to disjoint classes of information. The definition of these classes is left to the user and adds great flexibility to the model in terms of describing different security needs. Each object a is bound to a security class denoted by a which specifies the security class associated with the information stored in a. Within the model, two types of binding may occur: static or dynamic. Static binding implies that the security class of an object is constant and might be used to bind a class to a user or perhaps a process. Dynamic binding implies that the class of an object may vary with its contents.

- The $\oplus$ is a binary class-combining operator which exhibits the associative and commutative properties. It is used to specify the resulting class of information generated by a binary operation on two classes (e.g., the class of the result of any binary function on objects a and b is $a \oplus b$).

- A flow relation $\Rightarrow$ is defined on pairs of security classes. The notation $A \Rightarrow B$ means that information is permitted to flow from class A to class B.

Using the formal definition described above, a secure model can be specified in terms of flow relations. The security requirements of the model can then be simply stated by showing that a sequence of operations cannot give rise to a flow that violates the $\Rightarrow$ relation.

Under certain assumptions which are justified in [6], the set of security classes SC, the flow relations $\Rightarrow$, and the class-combining operator $\oplus$ can form a mathematical structure known as a *lattice*. A lattice is a structure consisting of a finite partially ordered set together with least upper and greatest lower bound operators on the set. This lattice turns out to be an extremely good choice to model many security environments, and in particular the government and Department of Defense (DoD) security systems [6, 19, 22].

The lattice model described by Denning is a four tuple, $LM = (SC, \Rightarrow, \oplus, \otimes)$, where:

(1) SC is again a finite set of security classes.

(2) The $\Rightarrow$ is a binary relation which includes a partial ordering on the security classes in SC.

(3) The $\oplus$ is an associative and commutative binary operator on SC, denoting the least upper bound of security classes.

(4) The $\otimes$ is an associative and commutative binary operator of SC denoting the greatest lower bound of security classes.

From the above formal statement, it can be seen that the entire finite set SC must have a greatest lower bound, say *LOW*, and a least higher bound, say *HIGH*, such that $LOW \Rightarrow A$ and $A \Rightarrow HIGH$ for all A in SC. This model can be used to specify allowable flow relations and the system can be designed to prevent explicit or implicit flow violations. It appears that with the OIS environment it will not be possible to enforce security with access controls alone if we desire to handle different levels of classified information simultaneously. Flow controls are important and add greatly to the overall security posture of the computing environment. They also offer protection against a Trojan horse attack and the establishment of storage channels which would be difficult or impossible to detect using access controls alone.

5. Summary and conclusions

We have attempted to show in this paper that the provision of secure computing in an OIS environment is different from that found in traditional systems. We identified several arguments to support this conclusion in section 3. Historically, research in this area has supported an approach to building secure systems by first identifying a security model and then building the system to conform to that model. We see the same approach being used in the OIS arena and suggest that a modified version of the BLP model would be a good starting point, as discussed in section 4.

The BLP model seems the best suited for use within office automation systems but may have to be modified to view security at the local level as well as the network level. This may entail the use of a reference monitor locally and a distributed reference monitor for the network level.

We would further assume that modification of the BLP model would be facilitated if we could identify a unit of information to control. In a standard computer application this unit (object) is normally a file, an area of memory, a variable, etc., but an OIS would be expected to largely process documents and forms. We propose to view information control of documents at the paragraph level known as the *basic information unit* (BIU). For forms, we would propose classification at form level itself based on what fields were filled in at an instant in time. A model can then check for secure information processing by ensuring that no combination of BIUs results in a violation of the model rules base.

Since we can expect the interconnection of OIS over a network we believe it is important to view security at two levels (local and network) and we propose that these two levels may be solvable in entirely different ways. Finally, we point out that to our best knowledge no federally recognized secure office information systems exist today although the market appears ready for such systems and the need is well documented.

Acknowledgment

The authors are grateful to anonymous referees for their constructive comments which substantially improved the quality of this paper.

References

[1] S. R. Ames, M. Gasser and R. R. Schell, Security kernel design and implementation: An introduction, *Computer*, 16 (7) (1983) 15–21.

[2] D. E. Bell, *Secure Computer Systems: A Refinement of the Mathematical Model*, ESD-TR-73-278, Vol. 3, ESD/AFSD, Hanscom AFB, Bedford, MA, April 1973 (MTR-2547, Vol. 3, Mitre Corp., Bedford, MA).

[3] D. E. Bell and L. J. LaPadula, *Secure Computer Systems: A Mathematical Model*, ESD-TR-73-278, Vol. 2, ESD/AFSD, Hanscom AFB, Bedford, MA, November 1973 (MTR-2547, Vol. 2, Mitre Corp., Bedford, MA).

[4] D. E. Bell and L. J. LaPadula, *Secure Computer Systems: Mathematical Foundations*, ESD-TR-73-278, Vol. 1, ESD/AFSD, Hanscom AFB, Bedford, MA, November 1973 (MTR-2547, Vol. 1, Mitre Corp., Bedford, MA).

[5] D. E. Bell and L. J. LaPadula, *Secure Computer Systems: Mathematical Foundations and Model*, M74-244, Mitre Corp., Bedford, MA, October 1974.

[6] D. E. Denning, A lattice model of secure information flow, *Commun. ACM*, 19 (5) (1976) 236–243.

[7] D. E. Denning, *Cryptography and Data Security*, Addison-Wesley, Reading, MA, 1982.

[8] D. E. Denning, Data security, *Computing Surveys*, 11 (3) (1979) 227–249.

[9] P. J. Denning (ed.), *Computers under Attack: Intruders, Worms, and Viruses*. Addison-Wesley, Reading, MA, 1990.

[10] Department of Defense Computer Security Center, Fort George G. Meade, MD 20755, *Department of Defense Trusted Computer System Evaluation Criteria*, August 1983.

[11] C. Ellis and N. Naffah, *Design of Office Information Systems*, Springer-Verlag, New York, 1987.

[12] C. Ellis and G. Nutt, Office information systems and computer science, *ACM Computing Surveys*, 12 (1) (1980) 27–60.

[13] D. Estrin, Non-discretionary controls for inter-organization networks. *Proc. IEEE Symposium on Security and Privacy*, pp. 56–61, 1985.

[14] D. Estrin, Controls for interorganization networks, *IEEE Trans. Software Eng.*, SE-13 (2) (1987) 249-261.

[15] V. D. Gligor and D. J. Baildy, Guest editor's note, *IEEE Trans. Software Eng.*, SE-13 (2) (1987) 125–127.

[16] J. E. Israel and T. A. Linden, Authentication in office system internetworks, *ACM Trans. Office Information Systems*, 1 (3) (1983) 193–210.

[17] B. W. Lampson, Protection, *Proc. Fifth Princeton Symposium on Information, Sciences and Systems*, pp. 437–443, March 1971.

[18] B. W. Lampson, A note on the confinement problem, *Commun. ACM*, 16 (10) (1973) 613–615.

[19] C. E. Landwehr, Formal models for computer security, *Computing Surveys*, 13 (3) (1987) 247–278.

[20] Y. Lapid, N. Ahituv and L. Neumann, Approaches to handling 'Trojan horse' threats, *Comput. Secur.*, 5 (3) (1986) 251–255.

[21] A. Lee and F. Lochovsky, User interface design. In D. Tsichritzis (ed.), *Office Automation*, Springer-Verlag, Berlin, 1985.

[22] J. McLean, The specification and modeling of computer security, *IEEE Software*, 23 (1) (1990) 9–16.

[23] W. H. Murray, Security in office automation and electronic document distribution, *Proc. IEEE 1984 Computer Society Conference on Office Automation*, 17–19 December, 1984.

[24] R. M. Needham and M. D. Schroeder, Using encryption for authentication in large networks of computers, *Commun. ACM*, 21 (1978) 993–999.

[25] J. K. Reynolds, The helminthiasis of the Internet, *Computer Networks and ISDN Systems*, 22 (5) (1991) 347–361.

[26] J. M. Rushby and B. Randall, A distributed secure system, *Computer*, 16 (7) (1983) 55–67.

[27] L. S. Rutledge and L. J. Hoffman, A survey of issues in computer network security, *Comput. Secur.*, 5 (4) (1986) 296–308.

[28] H. Saiedian, An object-based approach to the specification of office entities. In *Computing in the 1990's (Lecture Notes in Computer Science)*, Vol. 507, pp. 256–263, Springer-Verlag, Berlin, 1991.

[29] A. S. Tanenbaum, *Computer Networks*, Prentice-Hall, Englewood Cliffs, NJ, 2nd edn., 1988.

[30] D. Tsichritzis (ed.), *Office Automation*. Springer-Verlag, Berlin, 1985.
[31] R. B. Vaughn, *A Security Architecture for Office Automation Systems*, PhD thesis, Kansas State University, 1988.
[32] S. T. Walker, Network security overview, *IEEE Symposium on Security and Privacy*, pp. 62–76, 1985.
[33] M. D. Zisman, Office automation: revolution or evolution? *Sloan Management Review*, pp. 1–16, Spring 1978.

Col. Rayford B. Vaughn Jr. (US Army) received his PhD in Computer Science from Kansas State University in 1988. He is currently the Commander of the US Army Information Systems Software Center at Fort Belvoir, Virginia, and has previously served with the National Computer Security Center and as the Assistant Department of the Army Information Manager, the Pentagon. During 1990–1991 academic year, he served a one year appointment as a Visiting Professor of Computer Science at the US Naval Academy, Annapolis, Maryland. Col. Vaughn is a member of the Armed Forces Communications and Electronics Association. His research area includes security of office automation systems.

Hossein Saiedian is currently an Assistant Professor of Computer Science at the University of Nebraska at Omaha. He received his PhD in Computer and Information Sciences from Kansas State University in 1989. Dr Saiedian has numerous journal and proceedings publications in both computer science and information systems areas. His research areas include office automation, message passing semantics, and formal methods. Dr Saiedian is a member of the ACM and IEEE Computer Society.

Elizabeth A. Unger is Professor of Computing and Information Sciences at Kansas State University. Her areas of research interest include database systems and the use of the object-oriented approach to produce sound systems that enforce integrity and security. The security of database management systems is her primary research focus, with emphasis on inferential security and integrity. Prior to earning a doctorate in 1978 she had a career in the management of computing service centers.