Public key vs. conventional key encryption*

by CHARLES S. KLINE and GERALD J. POPEK

University of California
Los Angeles, California

INTRODUCTION
As distributed computer systems grow and their convenience attracts uses for which maintenance of privacy and security is important, the means by which encryption is integrated into these systems also becomes important. Encryption is the only practical way by which secure, private communication can be conducted while employing untrusted media to carry the transmission. The interest has spurred developments in the use of conventional encryption algorithms and there is even a federal standard algorithm for commercial use.4 In addition, an innovative approach to encryption, called public key algorithms, has recently been proposed as a way to address many of the key distribution and other problems which are present in conventional algorithm-based approaches.

Here we examine the general classes of functions desired of encryption algorithms as they are integrated into computer systems, and discuss the characteristics and properties desired in each case. We then look at the two general approaches—conventional algorithm-based, and public key-based. In every case we examine, we conclude that the two approaches are essentially equivalent. Neither approach has any particular advantage over the other. This conclusion is surprising, since one's intuition may suggest that public key algorithms are intrinsically superior because of their potential additional flexibility. Certainly, many public key proponents claim so. Yet, upon closer examination, the advantages of public key systems evaporate when one actually integrates them into a larger system.

In fact, we will see that the major unsolved problems are not concerned with key distribution, or the development of trusted software, but instead with the need for strong algorithms, whatever their form, and for reliable authentication methods: ways by which the human being can be effectively identified by the computer system.

PUBLIC KEY AND CONVENTIONAL ENCRYPTION ALGORITHMS—GENERAL CHARACTERISTICS

Encryption provides a method of storing data in a form which is unintelligible without the "key variable" used in the encryption. Basically, encryption can be thought of as a mathematical function

\[ E = F(D, K) \]

where \( D \) is the data to be encoded, \( K \) is the key variable, and \( E \) is the resulting enciphered text. For \( E \) to be a useful function, there must exist an \( F' \), the inverse of \( F \),

\[ D = F'(E, K) \]

which has the property that the original data can be recovered from the encrypted data if the value of the key variable originally used is known.

However, the use of \( F \) and \( F' \) is valuable only if it is difficult to recover \( D \) from \( E \) without knowledge of the corresponding key \( K \). A great deal of research has been done to develop algorithms which make it virtually impossible to do so, even given the availability of powerful computer tools.

The "strength" of an algorithm is traditionally evaluated using the following assumptions. First, the algorithm is known to all involved. Second, the analyst has available to him a significant quantity of matched encrypted data and corresponding cleartext. He may even have been able to cause messages of his choice to have been encrypted. His task is to deduce, given an additional, unmatched piece of encrypted text, the corresponding cleartext. All of the matched text can be assumed to be encrypted through the use of the same key variable which was used to encrypt the unmatched segment. In particular, therefore, the difficulty of deducing the key used in the encoding is directly related to the strength of the algorithm.

Recently, Diffie and Hellman1 proposed a variation of the conventional encryption methods that may in some cases have certain advantages over standard algorithms. In their class of algorithms, there exists

\[ E = F(D, K) , \]

as before, to encode the data and

\[ D = F'(E, K') \]

to recover the data. The major difference is that the key \( K' \) used to decrypt the data is not equal to, and cannot be easily derived from, the key \( K \) used to encode the data. Presumably there exists a pair generator which based on some input

---

* This research was supported by the Advanced Research Projects Agency of the Department of Defense under Contract MDA 903-77-0211.
information produces the matched keys $K$ and $K'$ with high strength (i.e., resistance to the derivation of $K'$ given $K$, $D$, and matched $E=F(D,K)$).

The value of such a public key encryption algorithm lies in some potential simplifications in initial key distribution, as well as for "digital signatures." The key $K$ used to encrypt the data is expected to be publicly known, and is referred to as the public key. The key $K'$ used to decrypt the data would be kept secret and is referred to as the private key.

**SYSTEM FUNCTIONALITY**

There are a number of privacy and security-related functions which are desired from a distributed computer system. Each of the classes of functions which have been suggested will be discussed, together with the properties desired of the function.

The assumptions made regarding the threats against which protection is desired include tapping of lines, introduction of spurious traffic and retransmission of previously-transmitted genuine traffic. It is assumed that malicious attacks are expected, and that there do not already exist secure, high bandwidth paths between those who wish to communicate in a private manner. This typically is the case when common carrier transmission media, including packet-switched services, are employed.

**Private communication**

This is the conventional use of encryption. In the distributed systems being envisioned, such as a large-scale computer network, it is viewed as desirable to separately protect each different conversation from every other. That is, the goal is to guarantee that each user/user connection is protected via encryption separately from all other connections. Therefore, each conversation needs its own key pair in conventional methods, and each user must have his own public/private key pair in the public key schemes. As a result, there can be a formidable key distribution problem. Any solution to this problem should not, in the view of many, depend on a single, central key distribution center, since the security of the entire network could depend on it, and the center contains high potential for abuse.

**Internal authentication**

In constructing a distributed system, it is not uncommon for the sites in the system to be connected using common carrier circuits, so that a potentially large amount of switching mechanism may be involved in the links. Hence, the sites may wish to exchange several messages each time that they recommence operation, to assure that they are each connected to the right sites. The usual method seen in computer systems today, where one member of the pair reveals some secret information to the other, is unacceptable in general if one is concerned about spoofing. The conventional form of this problem concerns authentication of the user to the system, via some form of a login protocol.

In networks, however, the problem is mutual—each "end" of the channel may wish to assure itself of the identity of the other end. Quick inspection of the class of methods used in centralized systems show that straightforward extensions are unacceptable. Suppose one required that each participant send a secret password to the other. Then the first member that sends the password is exposed. The other member may be an imposter, who has now received the necessary information to pose to others in the network as the first member. Whoever goes first potentially reveals his secret to a spoofer, who can then masquerade and collect other secrets. Obviously, extension to a series of exchanges of secret information will not solve the problem. It only makes necessary a several-step-posing procedure by the imposter. A different approach is in order.

**Datagrams**

In the private communication function, it is generally understood that the parties wishing to communicate are willing to pay some reasonable amount of overhead to get the private conversation established, so that a key distribution algorithm involving several messages would be quite suitable, for example. However, in the case of short messages, or datagrams, it is generally viewed as unreasonable for the actual transmission of the short message to require significant overhead, such as several preceding messages to set up the channel. On the other hand, some queuing delays at the sending or receiving site may well be acceptable if the number of overhead messages can be significantly reduced. Also, the datagram function is similar to mail, in that the receiver need not be active, or logged in, at the time the message is received.

**Digital signatures**

The goal here is to provide a way by which the author of a digitally-represented message can "sign" it in such a fashion that the "signature" has similar properties to the analog signature written in ink for the paper world. Without a suitable digital signature method, many have argued that the growth of distributed systems will be seriously inhibited, since large classes of applications would be precluded.

The properties required of a digital signature method include the following:

1. Unforgeability—It should only be possible for the author to create the signature for any given message.
2. Authenticity—There must be a straightforward way to conclusively demonstrate the validity of a signature in case of dispute, even long after authorship.
3. No repudiation—It must not be possible for the author of signed correspondence to subsequently disclaim authorship.
4. Low cost and high convenience—The simpler and lower-cost the method, the more likely it will be used.

Minimum trusted mechanism: Minimum central mechanism

In all of these functions, it is desirable that there be minimum trusted mechanism involved. This desire occurs because the more mechanism, the greater the opportunity for error, either by accident or by intention (perhaps by the developers, maintainers, etc.). One wishes to minimize the involvement of a central mechanism for analogous reasons. This fear of large, complex and central mechanisms is well justified, given the experience of the failure of large central operating systems and data management systems to provide a reasonable level of protection against penetration. All the Kernel-based approaches to software architectures have as their goal the minimization in size and complexity of central, trusted mechanism. Others are distrustful that a centralized, governmental (presumably) communication facility, or even a large common carrier can be trusted to assure privacy and other related characteristics. These general criteria are quite important to the safety and credibility of whatever system is eventually adopted. They also constrain the set of approaches that may be employed.

PRIVATE COMMUNICATION—KEY DISTRIBUTION

There are several requirements which any encryption protocol must satisfy. First, the encryption algorithm must be strong. Second, the keys to be used must be chosen and stored securely. Third, the keys must be communicated securely. Finally, authentication must be provided to separate this conversation from others which may use, or have used, the same key. Of these issues, the main problem is key distribution.

Conventional and public key systems solve these problems in different ways. In conventional systems, a new key is typically requested for each new communication. A key controller (which may or may not be centralized) chooses all keys, and performs any protection policy checks. The key controller only communicates the keys, establishing the communication channel if the protection checks succeed. The key controller communicates the keys chosen utilizing the same communications system as will be used for data transfer, but using previously arranged secret keys between the controller and the parties in the planned communication. However, it is straightforward to design the system so that the secret keys are stored in read only storage of the encryption units and never revealed. No authentication mechanism is needed to separate the new secure channel from prior ones, since the new keys chosen effectively form an authentication—no prior messages are useful.

Public key advocates claim that one of the advantages of public key algorithms over conventional algorithms lies in potentially simplified key distribution. Simply put, public key advocates argue that an automated “telephone book” of public keys can generally be made available, and therefore whenever user \( x \) wishes to communicate with user \( y \), \( x \) merely must look up \( y \)'s public key in the book, encrypt the message with that key, and send it to \( y \). Therefore, there is no key distribution problem at all. Further, no central authority is required initially to set up the channel between \( x \) and \( y \).

It is clear, however, that this viewpoint is incorrect—some form of a central authority is needed and the protocol involved is no simpler nor any more efficient than one based on conventional algorithms. First, the safety of the public key scheme depends critically on the correct public key being selected by the sender. If the key listed with a name in the “telephone book” is the wrong one, then there is no security. Furthermore, maintenance of the (by necessity machine-supported) book is non-trivial because keys will change; either because of the natural desire to replace a key which has been used for high amounts of data transmission, or because a key has been compromised through a variety of ways. There must be some source of carefully maintained “books” with the responsibility of carefully authenticating any changes and correctly sending out public keys (or entire copies of the book) upon request.

Needham and Schroeder\(^4\) exhibit protocols to provide the desired properties for public key systems, and show that there are equivalent protocols for conventional algorithms. The protocols are equivalent both in terms of numbers of messages required as well as in the mechanisms which must be trusted. In particular, the public key must be requested from the central authority (be it implemented in a centralized or distributed manner) and transmitted in a way which guarantees that the right key is received. Since the public key is reused, some authentication mechanism, such as a sequence number, is required to isolate this communication from others which may have used the same key. The communications required to retrieve the key and to establish the authentication mechanism make the public key distribution algorithm entirely equivalent to conventional algorithms.

Some public key advocates have suggested ways to avoid requesting the public key from the central authority for each communication. First, a cache of keys can be kept (a small local “telephone book”) and frequently used keys will be found there.

Second, a concept known as certificates has been suggested. A user can request that his public key be sent to him as a certificate. A certificate is a user/public key pair, together with some certifying information. For example, the user/public key pair may be stored as a signed message from the central authority. When the user wishes to communicate with other users, he sends the certificate to them. They each can check the validity of the certificate using the certifying information, and then retrieve the public key. Thus, the central authority is only needed once, when the initial certificate is requested.

Both the previous approaches have several problems. First, the mechanism used to store the cache of keys must be correct, since it will be relied upon. Second, the user of the certificate must decode it and check it (verify the sig-
nature) each time before using it, or must also have a secure and correct way of storing the key. Perhaps most important, as keys change the cache and old certificates become obsolete. This is essentially the capability revocation problem revisited. Either the keys must be verified (or re-requested) periodically, or a global search must be made whenever invalidating a key. Notice that even with the cache or certificates, an internal authentication mechanism is still required.

Public key systems also have the problem that it is more difficult to provide protection policy checks. In particular, conventional encryption mechanisms trivially allow protection policy issues to be merged with key distribution. If two users are not to communicate, then the key controller can refuse to distribute keys. However, public key systems imply the knowledge of the public keys. Methods to add protection checks to public key systems add an additional layer of mechanism.

**INTERNAL AUTHENTICATION**

There are a number of straightforward encryption-based authentication protocols which provide reliable mutual authentication without exposing either participant. The methods are robust in the face of all the network security threats mentioned earlier. The general principle involves the encryption of a rapidly changing unique value using a prearranged key. In the following we outline a simple authentication sequence between nodes A and B. At the end of the sequence, A has reliably identified itself to B. The analogous sequence is needed for B to identify itself to A. Typically, one expects to interleave the messages of both authentication sequences.

Assume that A uses a secret key, associated with itself, in the authentication sequence. The reliability of the authentication depends only on the security of that key. Assume that B holds A’s matching key (as well as the matching keys for all other hosts to which B might talk).

1. B sends $A$ in cleartext the current time of day as known to $B$.
2. $A$ encrypts that time of day using its authentication key and sends the resulting ciphertext to $B$.
3. $B$ decrypts $A$’s authentication message, using $A$’s matched key, and compares it with the time of day which $B$ had sent. If they match, then $B$ is satisfied that $A$ was the originator of the message. If the received time of day is not much older than the current time of day, $B$ is satisfied that the message has not been delayed and retransmitted.

This simple protocol does not expose either $A$ or $B$ if the encryption algorithm is strong, since it should not be possible for a cryptanalyst to be able to deduce the key from the encrypted time of day, even if he knew what the corresponding cleartext time of day was. Synchronized clocks in the network are not required. Further, since the authentication messages change rapidly, it is not possible to record an old message, retransmit it, and have it treated as valid by the recipient.

Authentication protocols such as these require the prior distribution of secret keys. If data security is the goal, no formal authentication protocol is actually required when all data transmissions are encrypted, since possession of the key serves as prima facie evidence that the participants are the appropriate ones, as well as providing the mechanism empowering the communication. Nevertheless, authentication protocols can give immediate assurance, and protect against the playback of previously recorded traffic.

**DATAGRAMS**

Datagrams are short messages from one user to another. These messages should be delivered with relatively low overhead if services such as electronic mail are to be practical. In addition, it is desired that buffering be performed at the recipient site. That is, the mail should be delivered as soon as possible to the recipient site, and stored there, even if the desired user is not logged in.

Assume that a user at one site wishes to send mail to a user at another site. Using conventional encryption algorithms, the first user would request a connection to the second user, and a new key would be chosen and distributed by the key controller to each end of the communication channel. That key is sent using the secret keys of the two users.

However, since the second user may not be signed on at the time, a daemon process is used to receive the mail and deliver it to the user’s “mailbox” file for his later inspection. It is desirable that the daemon process not need to access the cleartext form of the mail, for that would require the mail receiver mechanism to be trusted. This task can be accomplished by sending the mail to the daemon process in encrypted form and having the daemon put that encrypted data directly into the mailbox file. The user can decrypt it when he signs on to read his mail. In that way, the daemon only needs the ability to append to a user’s mailbox file.

In order for the user to know the new key used for this mail, however, the key distribution algorithm described earlier must be modified. Rather than send the key for this connection to both the sender and the receiver, the key controller sends the key twice to the sender, one copy encrypted with the sender’s secret key and one copy encrypted with the receiver’s. The sender can prepend the copy of the key encrypted in the receiver's secret key to the mail before transmission. When the recipient signs on, his own mail program will examine the mailbox file, find the key message encrypted with his secret key, decrypt it to obtain the key for that message, and then use that key to decrypt the remaining text.

In the case of public key encryption algorithms, the mail problem is somewhat simplified since the recipient knows...
what key to use in decryption (his private key). However, authentication is not possible since the recipient is not present when the message is received. Thus, it may be a replay of a previously sent message. This problem can be prevented in the conventional encryption algorithm case via various protocols with the key managers, for example, by timestamping the mail and having the recipient keep track of recently used mail keys.

Both mechanisms just outlined do guarantee that only the desired recipient of a message will be able to read it. However, as pointed out, they don't guarantee to the recipient the identity of the sender. This problem is essentially that of digital signatures, and is discussed in the next section.

DIGITAL SIGNATURES

The need for digital signatures has by now become apparent to many. At first, it appeared that public key methods would be superior to conventional ones for use in digital message signatures. The method, assuming a suitable public key algorithm, is for the sender to encode the mail by "decrypting" it with his private key and then send it. The receiver decodes the message by "encrypting" with the sender's public key. The usual view is that this procedure does not require a central authority, except to adjudicate an in deciphering the first message received from any given author (to get the corresponding public key, as mentioned). Second, the central authority must keep all old values of public keys in a reliable way to properly adjudicate conflicts over old signatures (consider the relevant lifetime of a signature on a real estate deed for example).

Further, and more serious, the unadorned public key signature protocol just described has an important flaw. The author of signed messages can effectively disavow and repudiate his signatures at any time, merely by causing his secret key to be made public, or "compromised." This fact has also been pointed out by Saltzer. When such an event occurs, either by accident or intention, all messages previously "signed" using the given private key are invalidated, since the only proof of validity has been destroyed. Because the private key is now known, anyone could have created any message claimed to have been sent by the given author. None of the signatures can be relied upon.

Hence the validity of a signature on a message is only as safe as the entire future history of protection of the private key. Further, the ability to remove the protection resides in precisely the individual (the author) who should not hold that right. That is, one important purpose of a signature is to indicate responsibility for the content of the accompanying message in a way that cannot be later disavowed.

The situation with respect to signatures using conventional algorithms initially appears slightly better. Rabin proposes a method of digital signatures based on any strong conventional algorithm. Like public key methods it too requires either a central authority or an explicit agreement between the two parties involved to get matters going.*** Similarly, an adjudicator is required for challenges. Rabin's method, however, uses a large number of keys, with keys not being re-used from message to message. As a result, if a few keys are compromised, other signatures based on other keys are still safe. However, that is not a real advantage over public key methods, since one could readily add a layer of protocol over the public key method to change keys for each message as Rabin does for conventional methods. One could even use a variant of Rabin's scheme itself with public keys, although it is easy to develop a simpler one.

However, all of the digital signature methods described or suggested above suffer from the problem of repudiation of signature via key compromise. Rabin's protocol or analogues to it merely limit the damage (or, equivalently, provide selectivity!). It appears that the problem is intrinsic to any approach in which the validity of an author's signature depends on secret information, which can potentially be revealed, either by the author or other interested parties. Surely improvement would be desirable.

A reliable digital signature method

A simple, obvious solution is to interpose some trusted interpretive layer between the author and his signature keys, whatever their form. For example, suppose the list of keys in Rabin's algorithm were not known to the author, but instead were contained in a secure Unit (hardware or software). Whenever the author wished to send a signed message, he merely submitted the message to the Unit, which selected the appropriate keys and then used the signature algorithm. Each author would have access to such a Unit.

The loading of each Unit requires some examination. In particular, the means which are used to select keys and insert them into each Unit must be correct if mail challenges are to be handled satisfactorily. That is, there must be some trusted Source of keys (and matching "standard message" in the Rabin protocol), and the key list for each author/recipient pair must be deliverable in a correct, secret way to the appropriate Units. We will call the collection of Units and the Source(s), together with their internal communication protocols, a Network Registry (NR). Such an NR appears required to solve the problems raised earlier. Note that some secure communication protocol among the components of the Network Registry is required. However, it can be very simple; low-level link style encryption would suffice.

For safety and efficiency, the NR functions presumably should be decomposed and distributed throughout the network. In particular, the failure or compromise of a local NR would then only have local consequences. One can even

---

*** In his paper, Rabin describes an initialization method which involves an explicit contract between each pair of parties that wish to communicate with digitally-signed messages. One can easily instead add a central authority to play this role, using suitable authentication protocols, thus obviating any need for two parties to make specific arrangements prior to exchanging signed correspondence.
Simplification of the proposed signature architecture—
Specialized digital signature protocols unnecessary

Once the necessity of a Network Registry is recognized, including a guaranteed authentication mechanism, it appears that simplifications in the mechanisms required for digital signatures can be made that seem to remove the need for specialized digital signature protocols. Instead, any of a collection of simple methods will suffice.

In particular, in order for the Network Registry to operate satisfactorily (including performing user authentication), it clearly must be distributed, and clearly must be able to communicate securely internally among the distributed components. Given that such facilities exist, then the following is an example of a simple implementation of digital signatures which does not require a specialized protocol or encryption algorithm:

1. The author authenticates with a local Network Registry component, creates a message, and hands the message to the NR together with the recipient identifier and an indication that a registered signature is desired.

2. A Network Registry (not necessarily the local component) computes a simple characteristic function of the message, author, recipient and current time, encrypts the result with a key known only to the Network Registry, and forwards the resulting "signature block" to the recipient. The NR only retains the encryption key employed.

3. The recipient, when the message is received, can ask the NR if the message was indeed signed by the claimed author by presenting the signature block and message. Subsequent challenges are handled in the same way.

This simple protocol involves little additional mechanism beyond that which was needed by the Network Registry anyway. It does require that the Network Registry be involved in every message signature and validation. However, recall that all of the unadorned signature methods reviewed earlier require involvement of some form of a Network Registry for at least the first message between any two parties. Public key protocols must check the "telephone book," and Rabin's method requires either a contract or a Network Registry. Furthermore, when one adds a more complete Network Registry on top of those other signature methods to correct their repudiation problem, all methods involve the NR for each message. Note that this protocol also does not require the NR to maintain any significant storage for signature blocks.

Performance and safety

Certain elementary precautions should be taken in the design of the Network Registry to avoid unnecessary internal message exchanges and to assure safety of the keys used to encrypt the signature blocks. Performance enhancements presumably would involve distributing the signature block calculation. Safety enhancements could include the use of different keys at each distributed site, replicating sites, and employing a signature block computation which requires the cooperation of multiple sites. Each of these facilities is straightforward to build and so they are not discussed further here.

It has been speculated that the task of constructing a Network Registry would be simpler using public key systems, since only the secret key of the Registry needs to be stored securely. However, using conventional encryption the Registry could encrypt all the private keys using a master key which belongs to the Registry. Thus, it is again the case that only the master key of the Registry needs long-term secure storage.

From the preceding discussions, we conclude that the digital signature algorithms proposed heretofore are unsatisfactory, and the improvements required to correct their inadequacies make the use of a specialized digital signature algorithm unnecessary.

We note here that the safety of signatures in this proposal also depends on the future history of protection of keys as before—in this case those held by the Network Registry. However, there are several crucial differences between this case and previous proposals. First, the authors of messages do not retain the ability to repudiate signatures at will. Second, the Network Registry can be structured so that failure or compromise of several of the components is necessary before signature validity is lost. In the previous methods, a single failure could lead to compromise.

USER AUTHENTICATION

While digital signatures are important, it is necessary to realize that there still must exist a guaranteed authentication mechanism by which an individual is authenticated to the NR (presumably directly the local Unit). Any reasonable communication system of course ultimately requires such a facility, for if one user can masquerade as another, all signature systems will fail. What is required is some reliable way to identify a user sitting at a terminal—some method stronger than the password schemes used today. Perhaps an unforgeable mechanism based on fingerprints or other personal characteristics will emerge.

Once the user has been correctly identified to the system, public key systems also must deal with the problem of retrieving the recipient's private key. That key must be securely stored, either in part in the user's head (not a very secure place), or somewhere else. Various forms of storage are possible, for example a simple card the user carries around with him, or in the system itself. However, the
storage must be in a form which is not useful to anyone else. For example, if the user loses the box containing his key, no one else should be able to use it, nor decipher its contents. This requirement means that the box itself can not be used as the sole user authentication mechanism. In addition, the key must be stored in the box in an unreadable form, presumably encrypted using some system key. The user must first authenticate himself to the system, have the system read the box, decode the key, and store it securely for use.

Once it is recognized that the system must be able to store keys securely, it becomes clear that the box just suggested can be dispensed with, except possibly as part of a user authentication mechanism. The system then would store users’ private keys. Once a user has authenticated himself, the system can retrieve the key. This approach avoids the problem of requiring the user carry around a card, and makes revocation/change of keys simpler.

Thus, there appears to be no advantage in the use of public key systems over conventional ones for user authentication or private key storage, since keys must be securely stored in either case. In fact, in conventional encryption, only the keys used for initial connection establishment with the key controllers require long-term storage. Others only remain as long as the connection is in use.

CONCLUSIONS

Based on the preceding discussions, we draw several conclusions. First, the debate over the relative advantages of public versus conventional key encryption algorithms is just not very important, at least for the class of applications discussed in this paper. In either approach, there must exist a similar amount of secure mechanism that must be trusted. Public key algorithms do not aid that problem to any significant degree. In any event, a strong algorithm is needed. Whether it is public key-based or a conventional one doesn’t matter much at all, compared to the overriding necessity that it be strong. If strong conventional algorithms are easier to develop, as has been speculated, research would be better devoted to that area rather than public key systems. Once suitable algorithms are available, the remaining weak link in the principles of secure, distributed systems lies with the requirement to accurately authenticate the user to the system.

Also, it seems that the digital signature methods which have been proposed, both public key- and conventional algorithm-based, do not adequately protect recipients of signed documents from repudiation of signatures by the author revealing the secret key(s) employed. The difficulty appears intrinsic to the approaches being taken. An alternative is available which overcomes this problem; however, that involves a small amount of trusted software.

The necessary underlying mechanism required to support improved digital signature methods, as well as other user-visible secure network communication protocols, is relatively well understood, and takes account of the important requirement that the amount of trusted mechanism involved be minimized for the sake of safety.

In more global terms, this discussion of network security has been intended to illustrate the current state of the art. Assuming a common carrier philosophy, then general principles by which secure, common carrier-based, point-to-point communication can be provided are reasonably well in hand. Of course, in any sophisticated implementation, there will surely be considerable careful engineering to be done.

However, this conclusion rests on one important assumption that is not universally valid. Either there exist secure operating systems to support the individual processes and the required encryption protocol facilities, or each machine operates as a single protection domain. A secure implementation of a Key Distribution Center or Registry is necessary in any case. Fortunately, reasonably secure operating systems are well on their way, so that this intrinsic dependency of network security on an appropriate operating system base should not seriously delay common carrier security.

REFERENCES
