CCITT/ISO Standards for Secure Message Handling

CHRISTOPHER MITCHELL, MICHAEL WALKER, AND DAVID RUSH

Abstract—**This paper examines some aspects of the security services in the CCITT Draft 1988 X.400 Recommendations for Message Handling Systems. A brief introduction to relevant cryptographic concepts is followed by a description of how some of the security services may be provided. Key management, as defined within the CCITT Draft X.500 Recommendations for Directory Services, and as referenced by the X.400 Recommendations, is also described. The paper concludes with a discussion of certain limitations of the current X.400 security architecture.**

I. Introduction

THE first version of the X.400 series of recommendations was approved by the VIIth Plenary Assembly of the CCITT in 1984, and published as part of the *Red Books* in 1985. They describe the overall architecture, as well as the operational requirements of various components which may be used to construct a distributed store-and-forward message handling system. X.400 was one of the first standardized applications conformant with the OSI model [1].

During the 1985–1988 study period, the X.400 series of recommendations has been developed to remove deficiencies highlighted by the various implementations, and to add further features. The revised draft recommendations now include support for various security services based, for the most part, on the use of asymmetric, or public key, cryptography. These draft recommendations were accepted at the final meeting of the CCITT Study Group VII in Geneva in March 1988 and were expected to be approved by the CCITT Plenary Assembly in October 1988.

During the same study period, the X.500 series of recommendations was developed. These describe another OSI application—Directory Services, and include mechanisms which may be used to manage the cryptographic keys used to provide the security features in the X.400 recommendations.

The revision of the X.400 recommendations and the development of the X.500 recommendations have been done as a collaboration between the CCITT and ISO. As a result, there exist two series of draft international standards very closely resembling the X.400 and X.500 recommendations. These are ISO DIS 10021 (Parts 1–8) and ISO DIS 9594 (Parts 1–8), respectively.

The purpose of this paper is to discuss the secure messaging extensions to the recommendations, with two main objectives in doing so. The first objective is to provide tutorial information about the security features in these recommendations. The key management and message security mechanisms are complex, the use of many of them is by no means obvious, and producing a secure messaging system based on the recommendations is a nontrivial task. One reason for this is that standardization is not primarily directed to making an implementor's task easy, but rather providing a framework within which different implementations may interwork.

The second objective of this paper is to point out potential shortcomings in the protocols, and areas where improvements can be made. The inclusion of security services into the X.400 recommendations occurred late in the study period. This has meant that a full analysis of the implications of security was not done, leading to a number of limitations of the architecture. This is especially apparent in the area of the message store, another new feature in X.400. In order that the protocols can be improved in the 1992 versions of the X.400 and X.500 recommendations, work needs to start now on designing better security elements. It is intended that this paper should contribute to that process.

The paper is divided into four main sections; these cover

- a brief overview of the cryptographic concepts used in the X.400 and X.500 recommendations,
- a description of the X.509 [2] authentication framework, which may be used by X.400 implementations for key management,
- an overview of the security services in X.400, with a description of how certain of them may be provided, and
- an indication of certain limitations of the security architecture.

II. Cryptographic Concepts and Constructs

The security mechanisms described in the X.400 and X.500 series of recommendations are largely based on asymmetric (public key) cryptography. The purpose of this section is to first give a brief tutorial overview of public key cryptography, tailored to provide a basis for an understanding of its use within the CCITT X.400 and X.500 recommendations. We then define, and briefly discuss, certificates and tokens. These are signed data struc-

Manuscript received August 1, 1988; revised January 15, 1989. This work was performed under the auspices of the LOCATOR Project, a part of the Alvey Mobile Information Systems Large Scale Demonstrator Project.

C. Mitchell is with Hewlett-Packard Ltd., Stoke Gifford, Bristol BS12 6QZ, England.

M. Walker and D. Rush are with Racal Research Ltd., Reading RG2 OSB, England.

IEEE Log Number 8927177.

tures which are basic to the provision of all the X.400 and X.500 security services discussed in subsequent sections.

A. Public Key Cryptography

Public key cryptography differs from conventional (symmetric key) cryptography in that decryption involves the use of a key which is different from that used for encryption. A user generates a corresponding pair of keys, keeps the decrypting key secret, and makes the encrypting key freely available. It is usual to refer to these keys as the secret key and the public key, respectively, and we adopt this convention throughout the paper. The characteristic feature of a public key cryptographic system is that it is computationally infeasible to compute the deciphering function from knowledge of the enciphering function, even though these are inverses of each other. In particular, a user may reveal the public key of a pair without exposing the secret key. Thus, user B wishing to send confidential information to user A may encrypt it under A's public key knowing that the encrypted data can only be deciphered by A using the corresponding secret key.

While confidentiality is provided by the above mechanism, authentication is not. Authentication can be provided using digital signatures. In X.400 and X.509, public key systems are used to provide digital signatures based on the following precept: if a block of ciphertext can be decrypted successfully using a given public key, then it must have been generated using the associated secret key; if only one user knows that secret key, then he must have generated the ciphertext. Thus, a signed data block consists of that data block and a copy of the data block encrypted under the secret key of the signatory. To check a signature, a recipient decrypts the encrypted portion of a signed data block, under the public key of the originator, and compares the result to the unencrypted portion. If the two agree, and provided the data have adequate redundancy, then the recipient can be confident that the data were indeed signed using the corresponding secret key. The data need to be redundant, and the recipient needs to be able to recognize this redundancy because otherwise, a signature may be forged by simply choosing a random value for the signed data block and setting the (clear) data block to be this value decrypted under the purported originator's public key.

One important feature of the digital signature described above is that, since it can only be computed with the originator's secret key, the originator cannot deny having sent the signed data. This fundamental property of digital signatures is the basis for all the nonrepudiation services in the X.400 recommendations.

If the data to be signed are too large to be encrypted in one operation, the signature procedure described above must be modified. The most widely advocated modification is to use a hashing function to first compress the data and then encrypt only these compressed data to provide a signature. Thus, a hashing function generates a digest of a message which can be encrypted in a single operation and automatically introduces redundancy into the message. The signature is checked by generating a copy of the digest by hashing the original message, and comparing the result to the result of deciphering the signature using the public key of the signatory. In order for this technique to be robust against cryptographic attacks, the hashing function must be such that it is not practical to generate two messages which compress to the same digest [3]. In the rest of this paper, we shall adopt the notation used in X.509 and let $X\{I\}$ represent data block I signed by user X. Thus, $X\{I\}$ consists of the data block I and the digest of I encrypted under the secret key of X.

In order to make use of the security features in the X.400 recommendations, it is necessary for users to generate key pairs in the public key system, and make the public keys available to all other users. In principle, different users may use distinct systems for different services. In practice, however, this apparent flexibility may not be possible because there are few known public key systems which can be used to provide digital signatures and confidentiality in the way prescribed above, and because the originator and recipient must both make use of a common system. One scheme which does meet all the requirements is the RSA scheme, and it is likely that a number of implementations will be based upon this. For details of this scheme, the reader is referred to [4]. A suitable hashing function which may be used in conjunction with RSA is defined in X.509 Annex D.

B. Certificates

From the brief discussion above, it should be clear that if a public key scheme is to be used to provide confidentiality or digital signatures, then it is absolutely essential that a user can be confident that a public key of another user really does belong to that user, and not to an impersonator. Thus, there is a requirement for a means to enable one user to validate the association between another user's name and his public key. This requirement is met within the X.400 recommendations by using a signed data structure called a certificate, the precise form of which is defined in X.509.

A certificate for a user's public key is generated by an off-line entity called a certification authority, which must be trusted by the user. The role of certification authorities and the management of certificates are described in more detail in the next section. Here we restrict ourselves to defining the precise form of a certificate, and the way in which it is used. We assume the certificate is generated by certification authority CA for a user with name A. The certificate then has the following form:

$$\text{CA}\ \langle\langle A\rangle\rangle = \text{CA}\ \{\text{sgnAlg, CA, } A, Ap, T^A\}.$$

Here sgnAlg identifies the algorithm used by the certification authority to generate the signature on the certificate (i.e., the asymmetric cipher and hashing function), Ap is A's public key, and T^A indicates the period of validity of the certificate.

The certificate is checked by hashing the contents of the certificate, and comparing the result to the result of de-

crypting the signature with the public key of the certification authority. If the two results are equal, then the user may assume that the certificate was generated by the specified certification authority. The user checking the certificate can therefore infer that the certification authority believes the association between the user name and the public key. Thus, if the user checking the certificate trusts the certification authority which issued the certificate, then he may confidently use the public key which it certifies.

It should be observed that if a user chooses to employ a number of keys, perhaps to support different services, then he will need a separate certificate for each of his keys.

C. Tokens

A token is a signed data structure which is used to pass security-related data from the originator of a message to a specific recipient. The X.400 recommendations have been written to allow tokens to be constructed using symmetric or asymmetric cryptographic techniques. In the 1988 series of recommendations, only a token based on asymmetric techniques has been defined, although it is expected that a token using symmetric techniques will be defined in subsequent versions. The recommendations also allow the use of user-defined tokens. Throughout this paper, we shall only consider the currently defined asymmetric construct.

Like a certificate, a token consists of a series of data fields with a digital signature appended. It is, however, always specific to a single communication, i.e., when required, a token is generated by a message originator for transmission to a single recipient. The precise form of a token, as specified in X.411 [5], is as follows:

$$A\{\text{sgnAlg}, t^A, B, \text{sgnData}, \text{encAlg}, Bp[\text{encData}]\}.$$

Here A is the name of the originator of the token, sgnAlg identifies the algorithm used by the originator to compute the signature, t^A is a time stamp, B is the name of the intended recipient, Bp[encData] is the data encData encrypted under B's public encryption key Bp, encAlg identifies the asymmetric algorithm used to perform this encryption, and sgnData and encData are collections of security-related parameters. The contents of sgnData and encData depend on the security services being provided. For example, sgnData may consist of an integrity check, which is used by the recipient to confirm the integrity of a received message content, whereas encData may consist of a secret key used to encrypt the message content.

In order to pave the way for an understanding of how various security services are provided, it is instructive to consider what cryptographic protection a token provides. First, the intended recipient of a token is assured that only he can derive the content of the encData from the token. This is because only the intended recipient has the secret key which corresponds to the public key used to encrypt encData. Second, by checking the signature on the token using the originator's public key, the recipient can confirm the integrity of the data in the token. Thus, in particular, the recipient can confirm that the contents of sgnData have not been deliberately or accidentally corrupted during transmission of the token. Third, since the signature on the token is computed with the originator's secret key, the recipient is provided with unforgeable evidence that the originator constructed the token.

In summary, the token provides the recipient with unforgeable evidence of the identity of its originator, assurance of the integrity of the information it conveys, and assurance that encData is not readable by any other party. However, it is important to note that the token does not provide the recipient with any assurance that the originator knows the content of the encData field. This limitation is discussed in more detail in Section V of the paper.

The X.400 series of recommendations defines two types of token: the bind token, and the message token. Bind tokens are used for peer entity authentication between entities attempting to generate an association. They will not be discussed in this paper. Message tokens are used to provide end-to-end security services, i.e., UA–UA services. They are generated on a per-recipient basis, and transferred in the envelope of a message. Throughout this paper, we shall only be concerned with message tokens, and henceforth, the term token is understood to mean message token.

III. The X.509 Directory Authentication Framework

X.509 (ISO DIS 9594-8) describes, among other things, a way in which a user can obtain a validated copy of the public key for another user, using certificates stored in the Directory.

Each user appoints a certification authority (CA) to generate a certificate for his public key. This certificate is passed to the user and stored in the directory under the user's entry. The CA also makes available a copy of its own public key to the user in such a way that the user can be confident in the authenticity of the key. The user is then able to employ this key to check certificates issued to other users subscribing to the same CA.

Within a small environment, it is possible for all users to subscribe to the same CA. Thus, all users will know the public key for that authority, and can use this to check the certificates for all other users. In a large environment, however, it may not be feasible, nor indeed desirable, for all users to subscribe to the same CA. In such a situation, it is necessary to provide means whereby users may obtain validated copies of the public keys of CA's other than their own. This is accomplished in the X.509 recommendations by using certain sequences of cross certificates.

A cross certificate is a certificate generated by one CA for the public key of another CA. Like user certificates, these are held in the Directory. To explain how cross certificates are used, we introduce the concept of a certification path. A certification path from CA_0 to CA_n is a sequence of cross certificates which takes the following form:

$$CA_0\langle\langle CA_1\rangle\rangle, CA_1\langle\langle CA_2\rangle\rangle, \cdots, CA_{n-1}\langle\langle CA_n\rangle\rangle.$$

content to other than the intended recipient(s). Content confidentiality is provided on a per-message basis.

Content Integrity: This allows the recipient(s) of a message to verify that the content of the message has not been modified. Content integrity is provided on a per-recipient basis.

Message Flow Confidentiality: This allows the originator of a message to protect information which may be derived from the observation of message flow.

Message Sequence Integrity: This allows the recipient(s) of a message to verify that the sequence of messages from the originator to that recipient has not been modified.

Nonrepudiation of Origin: This allows the originator of a message to provide irrevocable proof of the origin of the message to the message recipient. Nonrepudiation of origin is provided on a per-recipient basis.

Nonrepudiation of Delivery: This allows the originator of a message to obtain from the recipient(s) irrevocable proof that the message was delivered to that recipient. Nonrepudiation of delivery is provided on a per-recipient basis.

Nonrepudiation of Submission: This allows the originator of a message to obtain from the MTA to which the message was submitted irrevocable proof that the message was submitted to that MTA for delivery to the specified recipient(s). Nonrepudiation of submission is provided on a per-message basis.

Message Security Labeling: This allows the originator of a message to associate with the message an indication of the sensitivity of the message, called a security label, which may be used by the MHS to control the way the message is handled. Message security labeling is provided on a per-message basis.

Giving a detailed description of how all security services may be provided is beyond the scope of this paper. Instead, we have chosen to describe three groups of services. These services have been selected so as to illustrate the salient features of the mechanisms used to provide the full range of services, and to demonstrate that, in some cases, there is a choice of mechanisms to provide a particular service.

A. Content Confidentiality

Content Confidentiality allows the originator of a message to protect the content of the message from disclosure to other than the intended recipient(s). It is achieved by encrypting the content of the message. The encryption algorithm used is not specified in the recommendations, and may be either symmetric or asymmetric. Indeed, the originator of a message is, in theory, at liberty to choose an encryption algorithm for that particular message. In practice, however, he must be sure that the recipient(s) of the message have the capability to perform the chosen algorithm, and he must, of course, identify the algorithm to the recipient(s). The algorithm is identified to the recipients using the content-confidentiality-algorithm-identifier. This parameter may either be a defined field within the message envelope or may be a subfield of sgnData within the token.

If the content is encrypted using a symmetric algorithm, then the cipher key used must be shared between the originator and the recipient. This may be achieved by some means outside the scope of the recommendations (e.g., by prior agreement) or the key may be sent with the message in the token. If the message is to be sent to more than one recipient, the encryption key can be included in the encData field of the token for each of these recipients. The encData field is encrypted under the public key of the recipient, who can extract the cipher key from the token using his secret key, and then decrypt the message content using the algorithm specified by the content-confidentiality-algorithm-identifier.

If an asymmetric algorithm is chosen to provide content confidentiality, then the content is encrypted using the public key of the recipient, and the token is not required for key transfer. The disadvantage of this method is that a different encrypted content has to be provided for each recipient of the message.

B. Message Origin Authentication and Content Integrity

The Message Origin Authentication service allows the originator of a message to provide a means by which its origin may be verified. The Content Integrity service allows the originator of a message to provide a means by which the recipients of the message can verify that the message content has not been modified.

It is to be expected that the two services will normally be provided together. In fact, from the point of view of a recipient, it does not make much sense to provide one without the other. Message Origin Authentication on its own allows a user to identify the originator of a message, but does not allow him to check that the contents have not been modified. Indeed, the complete content could have been removed and replaced by another. Content Integrity on its own allows the recipient of a message to verify that the message content has not been modified, but cannot identify the originator of the message. As we now describe, within the X.400 series of recommendations there is a number of mechanisms which may be used to provide these services.

The first means of providing these services involves the use of a parameter called the message-origin-authentication-check (MOAC). This parameter is sent in the message envelope and consists of a digest encrypted under the originator's secret key. The digest is computed as a hashed version of the message content, together with a number of message-related variables; these variables include an algorithm identifier (which specifies the means used to compute the MOAC) and a security label. If the message content is encrypted, then the MOAC is computed using the encrypted version of the content; this is done so that all parties may check the integrity and authenticity of the message, even if they do not have the means to decrypt it.

It should be clear that the MOAC can be used to pro-

vide both Message Origin Authentication and Content Integrity to the intended recipients as well as all intermediate MTA's, as long as they have access to a trusted version of the originator's public key. However, the MOAC cannot be used to provide a nonrepudiation of origin service if the message content is encrypted since it only assures the irrevocability of the encrypted content and not the plaintext. This is because it is conceivable that the originator, while not being able to deny having sent the encrypted content, could claim that the plaintext equivalent is something different from what was originally intended. This is one reason why it is always considered "bad practice" to sign encrypted data; as we state elsewhere, as a general rule it is always better to authenticate first and encrypt second.

The second means of providing these services involves a completely distinct parameter called the content integrity check (CIC). This CIC may be computed by a variety of means, and includes both a check value and an algorithm identifier indicating the means by which it was computed. The check value within the CIC is computed as a function of the (unencrypted) message content. The CIC may either be sent as a message envelope parameter or within either the signed-data or encrypted-data fields of the message token(s). If a secret key is used to compute the CIC, then this may be included within the encrypted-data field of the message token(s).

If the CIC is sent as a message envelope parameter, then it is not protected against change by an unauthorized interceptor. Hence, in such a case, it must be computed using a secret key known only to the originator and the recipient [perhaps sent within the message token(s)]. Otherwise, the Message Origin Authentication service will not be provided, and as we have already pointed out, Content Integrity without Origin Authentication is a fairly meaningless service. However, the situation is actually worse than this; even if the CIC is key dependent, severe problems arise when a message is sent to more than one recipient. The X.400 protocols only allow for a single value for the CIC, and the key used to compute it must therefore be made available to all recipients. As discussed elsewhere [7], [8], this enables one recipient to defraud another. It would therefore seem that, unless a message is to be sent to a single recipient, this method of operation will not provide the desired service.

We therefore suggest that the CIC should always be sent within the message token(s). In such a case, there seems little to be gained by making it key dependent; the desired services are just as easily obtained if the CIC is computed using a publicly available hashing function (as long as this hashing function has the properties required for use as part of a digital signature scheme).

One great advantage with using this second method to provide the authentication and integrity services is that this same method can be used to simultaneously provide a Nonrepudiation of Origin service. A disadvantage is that, if the message content is encrypted, then the integrity service is provided only to the intended recipient(s) of the message, and not to any intermediate MTA's; this is because the CIC is computed using the unencrypted content, which will be unavailable to any parties other than the intended recipient(s).

C. Proof of Delivery and Nonrepudiation of Delivery

These services enable the originator of a message to obtain evidence that the message was delivered to its intended recipients. Nonrepudiation of Delivery is a stronger version of this service in which the recipient cannot subsequently repudiate the event. These services are provided by the originator sending a request for proof of delivery and the recipient returning a digital signature, called the proof-of-delivery-check. This is based on a number of fields in the delivered message, one of which is the clear message content. Of course, a recipient can decide whether or not to return a proof-of-delivery-check, so nonreceipt of this check by the originator does not mean nondelivery.

The recommendations do not specify the algorithm to be used to calculate the proof-of-delivery-check, and suggest that either symmetric or asymmetric algorithms may be used. If a symmetric algorithm is used, then only the Proof of Delivery service may be provided, and not the stronger Nonrepudiation of Delivery. If an asymmetric algorithm is used, the recipient may use his secret key to generate the proof-of-delivery-check, allowing the Nonrepudiation of Delivery service to be provided.

The proof-of-delivery-check is returned to the message originator in the delivery report. This report is generated by the message transfer system, and used to inform the message originator about delivery or nondelivery of the message. However, it is the MTS User to which the message is delivered who must generate the proof-of-delivery check. Thus, where a user agent is connected directly to the message transfer agent, it is this user agent which generates the check, while if a message store is situated between the message transfer agent and the user agent, it is this message store which generates the check. This means that for a user with a message store, it is the message store which must be assigned a key to generate the check. If asymmetric cryptography is to be used, this means that the message store must hold its own secret key, and make the corresponding public key available to other users. Alternatively, the user may entrust the message store with his secret key, although for many implementations, such an option would be unacceptable from a security point of view.

V. Limitations of the Current Architecture

Although the security features in the X.400 recommendations can be used to provide a wide range of security services, the user should be aware of a number of limitations.

A. Encrypted Data in the Token

The token involves signing encrypted data, which is generally accepted as bad practice; see, for example,

Davies and Price [9, Sect. 9.3]. This is because, as has been pointed out by Burrows and Needham [10], if the security operations are performed in this order, the recipient will only be able to guarantee the authenticity of the encrypted data. To illustrate this limitation, we need to first consider a "new" security service which, although not normally part of an authentication/integrity service, is intimately related to it. For want of a better name, we call the new service "Authorship." This service, if provided for a message sent from user A to user B, will guarantee to B that the message is known to A. Clearly, such a service is not useful unless provided in conjunction with integrity, authentication, and confidentiality.

As an example of a use for such a service, consider the Patent Offices of the future, to which claims are submitted via electronic mail (or some similar electronic communication medium). Individuals submitting patent claims to this office would, presumably, want to provide confidentiality, integrity, and authentication services for the patent claim message. The recipient of a claim would also, more than anything else, want to ensure that the patent claim comes from whom it claims to come from since he will ultimately be assigning ownership of the patent to the claimed originator.

Such a service can fail if provided using a token as defined in the X.400 series of recommendations. Suppose that a malicious third party, wishing to claim "authorship" of the message, intercepts it in transit. He now constructs a new message, containing the same encrypted content as the old message, but accompanied by a new token. This new message is then sent to the original recipient by this third party. The new token contains the same encrypted encData field as the old token and the same content integrity check in the sgnData field. The fundamental difference is that it is now signed by the third party. The recipient of this new message will believe that the message comes not from its true originator, but from the third party. Therefore, the "authorship" service is not provided, although message confidentiality, integrity, and origin authentication are.

The above example does make various assumptions as to how the security services are implemented, and there are implementations which overcome the limitation. However, it would be better if the token were constructed in such a way that the problem did not arise. For instance, defining the token so that encryption is performed after the signature is calculated removes the limitation, albeit at the cost of some increase in processing.

B. Proof of Delivery

The Proof of Delivery and corresponding Nonrepudiation of Delivery services are subject to a number of restrictions which may make them unworkable in practice when a message store is used. The most fundamental of these restrictions is the following.

As mentioned in Section IV of this paper, when a message store is used, the end point of the delivery service is the message store, and it is this entity which is required to compute any proof-of-delivery check. A problem arises if the content is encrypted because the check must be calculated on the clear text. To do this, the message store must gain access to the key required to decrypt the content. But this key is either a symmetric key shared exclusively by the recipient and the originator, or the secret key of the recipient, or an encrypted symmetric key which requires the secret key of the recipient to decrypt it. In every case, it is unlikely that the message store will be able to gain access to the key.

C. Digital Signatures

Although the recommendations do not explicitly specify the public key algorithm which is used to create digital signatures, the way in which the signatures are required to be constructed does limit the choice of algorithm. The method of construction requires that both public and secret keys can be used for encryption, and there are few well-established public key systems which enjoy this property, RSA being one. There is no need to specify so precisely how digital signatures are produced, and the fact that this is done is certainly a shortcoming of the recommendations.

D. Interaction with Other X.400 Services

Some of the security services in X.400 are incompatible with some of the other MHS services. Indeed, any service which modifies the message content invalidates any content integrity checks, and cannot be applied if the message content is encrypted. For example, Content Conversion by an MTA cannot be applied to an encrypted content unless that MTA has access to the key required to decrypt it.

VI. Concluding Remarks

The X.400 Recommendations are one of the first series of recommendations for which a security architecture has been defined. Its adoption for the provision of security services for users of X.400 message handling systems will lead to a more complete understanding of the issues involved in the implementation of large-scale security architectures, and is almost certain to be influential in the development of future standards and recommendations for security services.

References

[1] ISO 7498-1984, "Information processing systems—Open systems interconnection—Basic reference model."

[2] C. C. I. T. T., "Draft Recommendation X.509. The directory-authentication framework," Version 7, Gloucester, England, Nov. 1987.

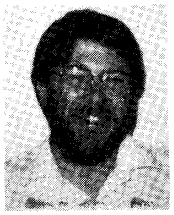
[3] I. B. Damgard, "Collision free hash functions and public key signature schemes," in *Proc. Eurocrypt '87*. Berlin: Springer-Verlag, 1988, pp. 203–216.

[4] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, pp. 120–126, 1978.

[5] C. C. I. T. T., "Draft Recommendation X.411. Message handling systems: Message transfer system: Abstract service definition and procedures," Version 5, Gloucester, England, Nov. 1987.

[6] C. C. I. T. T., "Draft Recommendation X.400. Message handling

systems: System and service overview," Version 4, Gloucester, England, Nov. 1987.
[7] C. J. Mitchell, "Multi-destination secure electronic mail," *Comput. J.*, vol. 32, pp. 13–15, 1989.
[8] C. J. Mitchell and M. Walker, "Solutions to the multidestination secure electronic mail problem," *Comput. and Security*, vol. 7, pp. 483–488, 1988.
[9] D. W. Davies and W. L. Price, *Security for Computer Networks*. New York: Wiley, 1984.
[10] R. M. Needham, private communication.



Christopher Mitchell received the B.Sc. degree in 1975 and the Ph.D. degree in 1979, both from Westfield College, London University, London, England.

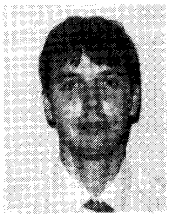
He is Project Manager for System Security at the Information Systems Centre of Hewlett-Packard Laboratories, where he has worked since 1985. He has worked on various aspects of information security since 1979, and his research interests include cryptology and combinatorial mathematics.

Dr. Mitchell is a member of the British Computer Society, the Institution of Electrical Engineers, and the London Mathematical Society, and is a Fellow of the Institute of Mathematics and Its Applications.

Michael Walker received the B.Sc. degree in 1969 and the Ph.D. degree in 1973, both in mathematics, from Royal Holloway College, University of London, London, England.

He is Head of Mathematics at Racal Research Ltd., where he is responsible for much of the company's work in cryptography, data security, and coding for error control. Before joining Racal in 1983, he was a Lecturer in Mathematics at the University of Tübingen, where he undertook research in finite geometries, groups, and combinatorics.

Dr. Walker is a member of the London Mathematical Society and a Fellow of the Institute of Mathematics and Its Applications.



David Rush received the B.A. degree in physics from Oxford University, Oxford, England, in 1979, and the M.A. degree in 1987.

After graduation he joined Racal Research, where he has been involved with many areas of data communications. He is currently a Principal Engineer, working on protocols for message handling and data security.