MESSAGE AUTHENTICATION WITH MANIPULATION DETECTION CODES

R. R. Jueneman*, S. M. Matyas†, and C. H. Meyer†

* Computer Sciences Corp., Falls Church, Virginia
† IBM Corp., Kingston, New York

Abstract

In many applications of cryptography, assuring the authenticity of communications is as important as protecting their secrecy. A well known and secure method of providing message authentication is to compute a Message Authentication Code (MAC) by encrypting the message. If only one key is used to both encrypt and authenticate a message, however, the system is subject to several forms of cryptographic attack. Techniques have also been sought for combining secrecy and authentication in only one encryption pass, using a Manipulation Detection Code generated by noncryptographic means. Previous investigations have shown that a proposed MDC technique involving block-by-block Exclusive-ORing is not secure when used with the Cipher Block Chaining (CBC) mode of operation of the Data Encryption Standard (DES). It is shown here that the Cipher Feedback (CFB) mode of operation exhibits similar weaknesses. A linear addition modulo 2^{64} MDC is analyzed, including discussion of several novel attack scenarios. A Quadratic Congruential Manipulation Detection Code is proposed to avoid the problems of previous schemes.

Introduction

As cryptography is being used with increasing frequency to protect both communications and file storage, it is important to have a clear understanding of exactly what information is being protected, and against what threats. If the protection objectives or requirements are not clearly understood, together with any threats that are excluded from the protection objectives, then the user risks a possible compromise. The following objectives are proposed as a check-list for a secure communications system, in order to set the stage for the discussion that follows. Not all of them are necessarily relevant to every application, of course; and we will ignore the military's concern for Transmission Security, including the possibility of the detection of transmission, the detection of compromising emanations, and the possibility of traffic analysis.

General Protection Objectives For Communications Security

1. **To prevent disclosure of plaintext** to any person or process not possessing the appropriate cryptographic key and/or appropriate unique identification such as a password, Personal Identification Number, or process identifier.

2. **To prevent release of information by the sender,** either accidentally or deliberately, by deceitful or faulty (Trojan Horse) mechanisms operating via nominally secure media or transmission paths; including the transmission of plaintext (or ciphertext, using a collaborator's key) in such a way as to cause it to be ignored or rejected by a legitimate recipient without causing alarm, e.g., by sending it to a non-existent addressee, or by deliberately inserting false parity.

3. **To permit the receiver to detect any modification of a message whatsoever,** including insertion, deletion, transposition, or modification of the contents.

4. **To permit the receiver to detect any modification of the sequence of messages,** either in a session or on a recorded file; including the insertion, deletion, or rearrangement of messages. And further, to prevent the undetectable deletion or loss of message(s) at the end of the session or data file, or at some forced, timely checkpoint.

5. **To permit verification of message origin and destination.** If the same key used for traffic from A to B is used for B to A traffic, messages from A might be delivered back to A, as though they had come from B. Valid messages from C to A might be copied and sent as though they had come from B.

6. **To permit the verification of message timeliness.** In a telecommunications session environment this implies that the entire session or sequences of messages is current, and not a replay of some previous (perhaps valid) session. In the absence of a bidirectional session, the individual message or datagram must at least be timely, i.e. with an authenticated timestamp

CH1882-0/83/0000/0033$01.00

that is within some delta-t of the current date/time at the receiver.

7. **To permit the sender to detect a fraudulent acknowledgement of message receipt or non-receipt** by someone other than the message recipient or his implied agent, i.e., someone with knowledge of the cryptographic variables. That is, the opponent must be prevented from returning fraudulent acknowledgements to the sender while preventing or withholding the recipient's acknowledgements.

8. **To extend the above protections to include the case where any modification of the message, message sequence, or message acknowledgement must be detected, even in the absence of message secrecy**, i.e., when the plaintext may be known to or even originated by the opponent — whom, it is assumed, does not possess the secret cryptographic key(s).

9. **To extend the above protections on a pair-wise basis to multi-party colloquies** taking place via multi-drop line, packet network, or satellite broadcast circuit, etc. (Verification of true message synchronization as between multiple senders may require precise clock synchronization or certain transmission designs such as a loop, and is not addressed by this objective.)

10. **To prevent fraudulent disavowal and/or forgery of a signed message (digital signature)**, and to permit both sender and receiver to verify their claims to the satisfaction of an independent referee. The process of notarization and/or claim verification should not compromise the secrecy of the information to the referee or any notary, nor should it compromise the digital signature scheme to either the recipient or the referee.

It is not the intent of this paper to propose or discuss the procedures and protocols necessary to satisfy these objectives. Rather, we only wish to point out that a requirement for message authentication is implied in more than half of these objectives, and then proceed to discuss how authentication of arbitrary fields of data can be provided.

Message Authentication

Message authentication is a procedure which, when established between two or more communicants, allows each party to verify that received messages are genuine. Communicants can be people, devices, or processing functions. Typically, a communicant acts as both a sender and receiver of messages, although it is possible for a communicant to participate only as a sender or receiver. It is not necessary for the communication to take place in real-time - messages may be received on a delayed basis, as in an electronic mail system.

Message authentication between communicants must usually satisfy objectives 3 through 7 listed above, plus objectives 8, 9, and/or 10 in certain cases.

In other words, message authentication permits the receiver to validate a message's **origin and destination, contents, timeliness, and sequence** relative to other messages flowing between the communicants. Any attempt by the opponent to trick the receiver or sender into accepting a spurious message or acknowledgement is called **spoofing.**

Although message authentication permits the receiver to verify that messages are genuine, it does not always permit all of these properties (origin and destination, contents, timeliness, and sequence) to be proven to or verified by a third party. (If both the sender and receiver share the same secret information used in the authentication procedure, the sender could later claim that a message was manufactured by the receiver; or vice versa the receiver could claim that the message was never received, at least in the alleged form.) Digital signatures, which permit a receiver to prove to a third party that messages are genuine (including acknowledgment messages, thereby protecting the sender) are not treated in this paper.

In simple terms, message authentication as defined here can be characterized as the following problem:

Given: A received message, Mrec, decrypted if and as necessary,

Decide: Whether the received message should be accepted as being identical to the message actually originating with the sender, Msent.

As a practical matter, we shall see that we can only decide that question to within some tolerable margin of error.

In this paper, we do not deal with the question of whether a given plaintext message was authorized. Passwords, magnetic credit cards, fingerprints, palm geometry, voice recognition, signature acceleration profiles, retinal scans, or whatever other techniques may be feasible must be used to confirm or vouch for the user's identity (say to an automated teller machine, or an electronically controlled door lock), and it is left up to the programming in those machines, suitably protected through physical measures, to ensure that the proper account balances are checked, admittance criteria are valid, etc., before money is disbursed or the door is opened. By message authentication we mean to imply that a wiretapper will not be able to bypass these physical controls by imitating valid control signals sent between the ATM or the door opener, for example, and the physically remote device that may act as the intelligent controller for those devices.

In addition, the checks on message origin and destination, timeliness, and sequence which are part of the definition of message authentication are not specifically addressed in this paper. We are here

concerned only with authenticating the message bits as received, in order to confirm that they did indeed originate in that form.

Two basic approaches for achieving message authentication will be discussed. In the first approach, a quantity defined a Message Authentication Code (MAC) is generated, which is an explicit, strong cryptographic function of both the data to be authenticated and a secret key. In the second approach, a different quantity, called a Manipulation Detection Code (MDC), is generated as a function of only the data to be checked. Since the MDC generating function is assumed to be publicly known, the necessary secret component is introduced by encrypting both the data and the MDC together. In either case, the MAC or MDC will be used in a manner similar to the traditional data processing checksum, to validate the data over which it was ostensibly computed.

The first approach can implement authentication either with or without secrecy (i.e., encrypted or unencrypted data can be sent in conjunction with a MAC). The second approach is assumed to be used for those implementations where secrecy as well as authentication is required. (Variations, not discussed here, are possible. For example, only the MDC, not the data to be authenticated, could be encrypted. But this is really a weak variation of the MAC approach where the secret component operates only on the MDC, not on the total data.)

More precisely, for the techniques discussed here, we can distinguish between the following applications:

MAC Approach:

Data (either plaintext or ciphertext), Dsent, is created together with an attached MACsent of k bits, which is a function of Dsent and a secret key. Msent represents either plaintext or ciphertext with MACsent appended.

Data (plaintext), Dsent, is created together with an attached MACsent of k bits, which is a function of Dsent and a secret key. Msent represents the encryption of Dsent||MACsent, where || denotes concatenation.

MDC Approach:

Data (plaintext), Dsent, is created together with an attached MDCsent of k bits, which is a function of Dsent. The resulting message is then encrypted to achieve secrecy.

If the receiver uses the same procedure as the sender to calculate either an appropriate Message Authentication Code (MAC of reference) or an appropriate Manipulation Detection Code (MDC of reference) then it follows that if the received data is equal to the transmitted data, then the MAC of reference=MACrec=MACsent. Hence, the following probabilistic statements can be made (Prob stands for probability):

Prob (MAC of reference = MACrec = MACsent given that Mrec = Msent) = 1

On the other hand, if the received data Mrec is not equal to the transmitted data Msent, then one can only state that:

Prob (MAC of reference is not equal to MACrec not equal to MACsent given that Mrec is not equal to Msent) = $1 - 2^{-k}$, since the correct MAC could have been generated by accident, with probability 2^{-k}. Similar statements apply for the MDC.

Using a MAC or MDC, the basic problem of message authentication reformulated in probabilistic terms can now be stated as follows:

Given: A sequence of n genuine plaintext bits X = x(1), x(2), .., x(n) and a corresponding sequence of genuine ciphertext bits Y= y(1), y(2), ..., y(n).

Compute: A sequence of k genuine authentication code bits A=a(1), a(2), ..., a(k), where A=MDC if A is a function only of the genuine plaintext bits X, and X||A is encrypted with a secret key; or A=MAC if either A is a function of the genuine plaintext bits X and a secret key (authentication without secrecy), or A is a function of the genuine ciphertext and a secret key (authentication with secrecy). (|| denotes concatenation.)

Such that: **The probability that a spurious received message Mrec** (where Mrec equals the encrypted X*||MDC*, or the encrypted X*||MAC*, or the unencrypted X*||MAC*, or else the unencrypted Y*||MAC*) **could be created** (of practical length and using a computationally feasible procedure) (where Mrec is not equal to the message sent, Msent) **such that the received or recovered authentication code bits** (MAC* or MDC*) **are equal to the authentication code bits of reference** (MAC* of reference calculated on Y* or X* or recovered X*, or MDC* of reference calculated on recovered X*) **is** $= 2^{-k}$.

Further, the chance that any individual bit of the original message, Msent, could have been altered without detection upon receipt (Mrec not equal to Msent) shall computationally indistinguishable from a uniformly distributed random variable that is independent of the message, Msent. That is, the authentication algorithm should not knowingly be weaker with respect to certain parts or bits of the message than others.

Assuming: An adversary knows the authentication algorithm F, and the encryption and decryption algorithms E and D but not the encryption/decryption key K, where

$Y = E_K(X)$ and $X = D_K(Y)$, nor the key K_A used in the authentication algorithm, if any.

However, the adversary can observe or choose any X and observe the resulting $Y = E_K(X)$, and can modify the ciphertext to produce Y* at will. He can also observe the resulting $X^* = E_K(Y^*)$ at will at the receiving end. In addition, he can choose any X or Y and observe the result of the authentication algorithm $MAC = F_A(X)$, or $MAC = F_A(Y)$, or the encrypted product of the MDC operation, $Y = E_K(X||MDC)$. (Depending upon the encryption algorithm and mode of operation, the encrypted result of the MAC or the MDC may not be directly observable, i.e. $E_K(X||MDC)$ may not equal $E_K(X)||E_K(MDC)$).

We require: That the adversary cannot construct a single spurious Y*, X*, or A*, even using some multiple previous $E_K(X_i)$, such that $A^* = A$ unless $X^* = X_i$, except by random chance with probability $Pr = 2^{-k}$, or else through exhaustive procedures whose computational complexity is of order 2^k. (How large k should be is a question that has been much debated. We would suggest that k be at least 31 for financial transactions, in order that the expected value of Prob (lucky guess) * $value-of-success be insignificant. For cryptographically sensitive operations k should be at least 56, in order that it not be weaker than the DES.)

Note: If the adversary were to cause to be constructed N random, valid Y's or X's, and also constructed N random and spurious X*'s or Y*'s with an associated A*, then by virtue of the so-called "Birthday Problem" the probability that some A* = some $F_p(X)$ (or some A* = some $F_c(Y)$ would become approximately 50% as N approaches the square root of 2^k. (Although the spurious X* that gave rise to the associated A* may very well fail internal consistency tests such as time and date stamps or sequence numbers, our goal is to be independent of such internal context sensitive tests.)

Note that with the MAC or MDC approach, the number of redundant bits are generally less than the number of message bits to be checked. In that case, the MDC or MAC is generally a many-to-one function of the data to be checked. For that reason, with only k bits available for checking purposes, there is always the chance that the correct MAC or MDC was generated by a lucky accident even if changes took place in the message.

We are assuming that the calculation of the MAC or MDC is such that the result is computationally indistinguishable from a uniformly distributed random variable, independent of the message contents over which it was calculated. Note that this is not a requirement, merely an assumption, for if the messages were known in advance there are methods of constructing hashing codes which contain no duplicates[1]. Normally, however, the best that can be done with an authentication procedure is to reduce the probability of making an error (accepting Mrec as equal to Msent when in reality it is not equal) to 2^{-k}, averaged over all messages.

To achieve this best case it must be assumed that the cryptographic algorithm that underlies either the MAC authentication procedure or the encryption of the plaintext plus MDC is sufficiently strong, i.e., the secret key or keys involved in the authentication procedure cannot be obtained by cryptanalytic methods.

Another crucial factor is the amount and kind of information an adversary has available to attack the authentication procedure. In order to be as general as possible, we will consider the following cases:

1. The opponent is an outsider and thus can only obtain information outside the system node(s). This restricts him to plaintext and MAC, ciphertext and MAC, or ciphertext only in the case of the MAC approach; or ciphertext only in the case of the MDC approach. However, it must be assumed that there is some non-zero possibility that some plaintext will be eventually be compromised or deliberately disclosed (perhaps by the user, who carelessly discards his banking transaction receipt, or as in the case of a press release that is confidential until a certain date) and that matching plaintext/ciphertext could be obtained thereby.

2. The opponent is an insider or at least a user, and can exercise the cryptographic system or cause it to be exercised. In that case, he may have copious quantities of plaintext and corresponding ciphertext available, selected as desired. (Since the keys are normally managed by the system in a presumably secure manner it is still assumed that the system users do not know the keys themselves.)

3. If a unique session key is used for every pair of communicants and kept secret from the communicants themselves, it might seem that the ability to obtain plaintext/ciphertext pairs would be quite limited; but there is the possibility that the messages are compromised by being stereotyped, or perhaps by deliberate disclosure such as a commercial announcement. In addition, there is the possibility that a legitimate but dishonest user will try to subvert his own message. For example, he could entice node A to perform a transaction of say $100,000. In the outgoing message, on the

other hand, the amount is changed to $100, resulting in a debit to his account of only that amount.

To evaluate different authentication techniques, the following assumptions could thus be made:

1. Opponent does not have plaintext/ciphertext pairs.

2. Opponent has a set of plaintext/ciphertext pairs available (selected, if desired); however, he does not know the plaintext he wants to modify for the case where authentication and secrecy is implemented.

3. Opponent has a set of plaintext/ciphertext pairs available (selected, if desired) and he also knows the plaintext he wants to subvert.

It is always assumed that the authentication algorithm (to generate a MAC or an MDC) is known in addition to the encryption and decryption algorithm. Although the authentication key used by the sender with either the MAC or the MDC approach must be secret (which applies also for a public key algorithm), it must be assumed in the most general case that an opponent has selected plaintext and corresponding ciphertext available, and vice versa.

Summarizing, we mean that the adversary can, in the most general case, observe previous traffic and even cause any desired message(s) to be sent, including known selected plaintext. He can also add, modify, or delete the transmitted or stored message bits at will, but he cannot cause the receiver to accept some spurious received message Mrec* as valid, except by random chance (i.e., with probability 2^{-k}).

Message Authentication Codes

In order to transmit information through a hostile environment and detect any modification upon receipt, it is necessary to convey some additional information, or redundancy. In its simplest form, this redundancy may just be a parity check per ASCII character, or it may be a Cyclic Redundancy Check or other block check computed over the entire message block. In some cases, notably for magnetic storage media and for satellite data links, the redundant information may take the form of a Forward Error Correcting code, so that the information need not be retransmitted in order to correct an error. In these examples, however, although the environment may be hostile from the standpoint of noise or other random errors, it is not assumed to be malevolent. If the hostile environment includes the possible presence of an opponent who is presumed to be capable of recording, altering, and retransmitting the message, then most of the existing body of knowledge of coding theory is of no avail, and cryptographic methods must be employed. By involving a secret quantity in the redundant information (either directly or indirectly) we hope to prevent the intruder from modifying the message without detection, except by a very lucky accident.

The degree of complexity required in the generation of the redundant information presumably depends greatly on whether the information to be authenticated is also encrypted for secrecy. If the message itself is to be sent in the clear, then it is obvious that cryptographic methods involving a secret quantity or key must be used to generate the authentication information. On the other hand, if the message is to be encrypted, then perhaps some weaker, and hopefully easier-to-calculate quantity may be used to provide the necessary authentication.

As discussed before, if the redundant information is computed solely as a function of the plaintext, the result will be termed a **Manipulation Detection Code**, or **MDC.** If the redundant information is computed from the plaintext using cryptographic methods and a secret key, the result is defined as a **Message Authentication Code**, or **MAC.** [Although most of this paper presumes the use of the Data Encryption Standard (DES) or similar block cipher employing traditional bidirectional secret keys, this definition would also apply in the case of a Public Key Algorithm. In that case, although the sender would use the public key of the receiver to encrypt the information, and the receiver would use his secret key to decrypt it for secrecy, it will be necessary for the sender to decrypt the information with his secret key, and for the receiver to encrypt it with the public key of the sender in order to assure its authenticity. In addition, some defined field, k-bits in length, must be transmitted and checked upon receipt in order to assure the authentication of otherwise context-free information.]

Simultaneous Secrecy and Authentication Using Two Encryption Keys.

Although there might be a number of suitable generating algorithms to compute a MAC, the best known and "official" scheme is documented in the DES Modes of Operation publication, FIPS PUB 81.[2] Two different methods are provided, both of which provide equivalent security as far as is known. The first involves the use of the Cipher Block Chaining (CBC) mode to encrypt the plaintext, which must be padded (e.g. with zero bits) if necessary to make it a multiple of 64 bits in length. The MAC consists of the last 64 bits of ciphertext (or a k-bit subset of it), the rest of which is discarded. The alternative scheme uses the Cipher Feedback (CFB) mode of operation to encrypt the plaintext. After the final byte is encrypted the DES engine is operated one more time, and k bits are selected from the high order end of the 64-bit output block. (See Meyer and Matyas[3] for a more detailed discussion of MAC generation.) **Because of the well established forward error propagating properties of both Cipher Block Chaining and the Cipher Feedback modes of operation,**

the change of even so much as a single bit in the clear plaintext would cause an unpredictable change to every bit in the MAC with a probability of 50% for each bit. If a k-bit subset of the 64-bit MAC is chosen and transmitted along with the associated message to be authenticated, and if that portion is recomputed on the received message at the destination, then there is only a probability $Pr = 2^{-k}$ that the received MAC matches the recomputed MAC of reference in the event that the transmitted message, Msent, has been tampered with (i.e., Mrec is not equal to Msent). This probability can be made as small as desired by choosing k sufficiently large.

In many cases it is desired to both encipher and authenticate messages. As far as is known, **if two different keys are used, one for encipherment for secrecy and one for the generation of the Message Authentication Code (MAC), then a strong procedure is obtained and the order of the operations is unimportant.**

Single Key Encryption-Authentication.

If for some reason only one key were to be used for both encipherment for secrecy and for authentication using a Message Authentication Code, then various attacks against the authentication procedure have been shown to be feasible. Furthermore, the attacks are different depending on the order of the operations. **At this time, it is still an open question whether a satisfactorily strong encryption-authentication scheme can be obtained which uses only one key.**

Consider the case where the initializing vectors (IVs) for both operations are the same, and the MAC is computed first and appended to the message, using CBC mode. To express this a little more rigorously, let message $M = X_1, X_2, ..., X_t$ be encrypted with key K and initializing vector IV to produce ciphertext $Y_1, Y_2, ..., Y_t$, where the last block of ciphertext Y_t is defined as the 64-bit MAC. The augmented message $M' = M||MAC$ (M concatenated with MAC) is now encrypted using the same key and the same IV, using CBC mode, to produce $Y_1, Y_2, ..., Y_{t+1}$.

Now suppose that the ciphertext blocks $Y_1, Y_2, ..., Y_{t-1}$ are changed arbitrarily to $Y_1^*, Y_2^*, ..., Y_{t-1}^*$. On decipherment with key K and initializing vector IV, this causes spurious plaintext blocks $X^* = X_1^*, X_2^*, ..., X_{t-1}^*, X_t^*$ to be recovered. However, due to the self-synchronizing property of the CBC mode, it causes the original MAC of reference to be recovered as well. Consequently, when the MAC is recalculated (i.e., when $X_1^*, X_2^*, ..., X_{t-1}^*, X_t^*$ is again encrypted with K and IV), the ciphertext $Y_1^*, Y_2^*, ..., Y_{t-1}^*, Y_t$ is again produced, where Y_t = MAC. Thus the MAC generated on X* is equal to the recovered MAC of reference, and the described alteration to the ciphertext escapes detection. If an adversary changes ciphertext block Y_t to Y_t^* (and perhaps other previous blocks of ciphertext) then the recovered MAC of reference equals Y_t^*, and thus is different from the original MAC. But even in that case, the recalculated MAC and the recovered MAC of reference are both equal to Y_t^* and the authentication check again succeeds. (Although the modification of the ciphertext will certainly cause a string of error bits to appear in the decrypted plaintext, and it might seem that the MAC calculation would catch such an error, we can see that the error bits do not propagate to the last block of ciphertext during MAC generation, so the alteration goes undetected.)

This problem requires further analysis in order to ascertain exactly what the various attack scenarios might be. For example, does the above attack (or an extension) work in the case of a MAC formed by Cipher Feedback? Also, the authors had originally speculated that the use of two different IV's, one for encryption and one for authentication would suffice to defeat attacks against the one key approach, regardless of the order of the operations, but it now appears that this is not always the case. Dr. Don Coppersmith of IBM Research has shown that the following attack scenario exists:[4]

Consider the case where the initializing vectors for encryption and authentication are IV1 and IV2, respectively, IV1 is not equal to IV2, IV2 is nonsecret, and the MAC is generated on the ciphertext using CBC mode. (It should be noted that Federal Standard 1027 requires that CBC mode initializing vectors be kept secret for encryption, but does not specify how IV's are to be protected for authentication. The current ANSI Banking Standard X9.9-1982 calls for an authentication IV of 0.)

Since messages are assumed to be encrypted routinely for secrecy using the same key as is used for MAC generation, it is reasonable to assume that an adversary can obtain a string of plaintext and matching ciphertext. (In effect, since encryption and authentication use the same key, plaintext-ciphertext pairs obtained via message encryption can be used to attack the authentication scheme.) Let the previously known blocks of plaintext and ciphertext be denoted A_i and B_i, respectively, where $E_K(A_i) = B_i$ for i = 1, 2, ..., n. (From a previous message, B_i might be ciphertext block Y_j and A_i might be plaintext block X_j Exclusive-ORed with ciphertext block Y_{j-1}, i.e., $B_i = Y_j$ and $A_i = X_j \oplus Y_{j-1}$.) (We will use the symbol "$\oplus$" to denote the block-wise Exclusive-OR operation, for typographical convenience.)

Using the pairs (A_i, B_i), it is now possible to construct a string of ciphertext blocks, $Y_1^*, Y_2^*, ..., Y_t^*$ and MAC* such that the authentication check will succeed. The recovered plaintext itself will be "garbage", and therefore further consistency checks on the text could cause the authentication check to

fail. However, we stated as an objective that the authentication check should work correctly in a context-free environment, and therefore we must reject this single-key approach if the problem cannot be overcome.

To see how such a ciphertext could be constructed, suppose that an adversary knows just one block of plaintext (P) and corresponding ciphertext (C), and that the IV used for authentication, IV2, is equal to zero. The ciphertext

$$P, P \oplus C, P \oplus C, \ldots, P \oplus C, C;$$

where the last block C is the MAC, will pass the authentication check. To see this, note that when P, P ⊕ C, P ⊕ C, ..., P ⊕ C is encrypted, the string of ciphertext C, C, ..., C is produced (i.e., the last block of ciphertext is C, which is equal to the MAC at the end of the received message.)

Simple modulo 2 additions are all that is required to construct a spurious ciphertext that passes the MAC check. Thus an adversary monitoring a communication session could very rapidly construct such spurious ciphertexts and inject them into the communication path. Message headers or stereotyped error messages sent during a session could be used to deduce one or more blocks of plain and matching ciphertext.

Even if IV2 were secret, transmitted messages could be modified using a variation on the above attack. For example, if $Y_1, Y_2, \ldots, Y_t$, MAC denotes the encrypted message and MAC, then the modified message $Y_1, Y_2, \ldots, Y_t$, MAC ⊕ P, P ⊕ C, P ⊕ C, ..., P ⊕ C, C will also pass the MAC check.

A more general attack involving a table of known plaintext and matching ciphertext is now given. Consider the case where t=5 (i.e., the message contains five blocks). The problem then is to define $Y_1^*, Y_2^*, \ldots, Y_t^*$ and MAC* such that the authentication check succeeds. To do this we work backwards starting from MAC*. Let MAC* be any block of known ciphertext from our previous interception, say B_{21}. Thus, MAC* = B_{21}. Now define $Y_5^* = Q_5 \oplus A_{21}$, where Q_5 can be any block of ciphertext in the known table (say B_7). Thus, $Y_5^* = B_7 \oplus A_{21}$. Now define $Y_4^* = Q_4 \oplus A_7$, where Q_4 is any block of known ciphertext from the table, (say B_{13}). Thus, $Y_4^* = B_{13} \oplus A_7$. Proceeding in the same manner, let $Y_3^* = B_2 \oplus A_{13}$ and $Y_2^* = B_9 \oplus A_2$. The final block of ciphertext Y_1 is $Q_1 \oplus A_9$, where Q_1 is automatically set equal to IV2 (the initializing vector used for MAC generation, assumed to be non-secret.) Thus $Y_1 = IV2 \oplus A_9$. It is fairly obvious from the construction that the final block of doubly encrypted ciphertext (produced by encrypting Y_1^*, $Y_2^*, \ldots, Y_5^*$ with K and IV2) is just B_{21}, which is equal to MAC*. Thus, the authentication check succeeds. On the other hand, it should be obvious that decrypting $Y_1^*, Y_2^*, \ldots, Y_5^*$ with K and IV1 will produce a spurious recovered plaintext.

This latter attack by Coppersmith may or may not work against a MAC that is computed using CFB, but the authors feel that if an attack succeeds against one mode, some variation of that attack is very likely to succeed against other modes. As a result, **we would recommend that any attempt to minimize the key management problem by using only one key for both message encryption for secrecy and for a Message Authentication Code should be reviewed very carefully in light of the findings in this area.**

Comparisons with Public Key Cryptography.

Finally, it is worth noting the similarity of these results with those in the area of Public Key Cryptography (PKC). Heretofore it has been assumed that one of the major differences between PKC and conventional cryptosystems was that if both secrecy and authentication was desired in a PKC, then the message had to be encrypted twice and decrypted twice (decrypting using the sender's private key for authentication, to be encrypted using the sender's public key; and then using the receiver's public key for encryption for secrecy, to be decrypted using the receivers private key.) Now, surprisingly, we see that it may well be necessary to encrypt messages twice using two different keys even when using the Data Encryption Standard algorithm, if we want to ensure both the secrecy and the authenticity of the message through context-free electronic means.

Although all of these difficulties may make us rather pessimistic of our chances of finding some acceptable alternative, the fact remains that in some computing/communication systems it may be difficult to apply both authentication and secrecy-protecting encryption using two different keys. In many cases it may be felt that in general the threat of an active, message modification attack is not as severe as that of message interception, leading to a system where all outgoing traffic may be encrypted in bulk at a rather low level in the transmission architecture (e.g. at the data link level; or perhaps even lower than that, at the physical link or common carrier transmission system level). Only the particularly sensitive applications would require authentication, often on an end to end basis, and therefore before the encryption step. Since it would be desirable that the link encryption device not have to be cognizant of the formats and protocols of the message traffic and therefore unaware of any requests for authentication, it follows that either hardware or software encryption would be required within the general purpose computer.

Because the problems of MAC generation, key distribution, the number of keys, and the order of operation dependencies have not yet been studied as exhaustively as we might like, there is the perhaps vain hope that some of these conditions could somehow be relaxed. In particular, all of the attacks discussed above featured the use of the same

encryption algorithm for both the secrecy and authentication operations. Various thoughts for improvement come to mind, such as mixing CBC and CFB for encryption and authentication, authenticating backwards but encrypting forwards, or using some form of block chaining that would provide error propagation during both encryption and decryption.

Nevertheless, there is at least the problem that two separate encryption operations on the same text may be inherently undesirable from the standpoint of efficiency, hardware requirements, and key management, even if both were integrated within one monolithic unit. For that reason, it would be highly desirable if some scheme could be derived which would provide adequate security against message modification attacks, yet would not require two separate encryption operations when both secrecy and authentication are required. Such forms of authentication codes, called Manipulation Detection Codes to highlight the fact that they may potentially be weaker than Message Authentication Codes, will now be discussed.

Manipulation Detection Codes

A Manipulation Detection Code initially proposed by the National Bureau of Standards[5] found its way into a number of publications (notably a draft of Federal Standard 1026[6]) and books[7], and enjoyed considerable appeal due to its simplicity. Unfortunately, this particular code was found to have several flaws[8]. (It must be noted that all mention of message authentication, key exchange protocols, etc., has been removed from the most recent version of FS1026[9] which is now circulating for public comment. The error has also been corrected in the third printing of Reference 7.)

The technique originally proposed involved the computation of a Manipulation Detection Code as:

$$MDC = X_1 \oplus X_2 \oplus \ldots \oplus X_n \tag{1}$$

where $\oplus$ represents the block-by-block Exclusive-OR (modulo 2 addition) of k-bit blocks (where k = 16 typically). Such an MDC technique is now known to be quite insecure, because even if the received MDC were equal to the recalculated MDC based upon the received (deciphered) plaintext, there is a possibility that the ciphertext of the message was manipulated in such a way as to not be detectable. This will now be shown, first in the case of Cipher Block Chaining, then for Cipher Feedback mode.

MDC Based On Modulo 2 Addition In The Case Of Cipher Block Chaining

In the Cipher Block Chaining (CBC) mode of operation the successive 64-bit blocks of ciphertext are defined as:

$$Y_i = E_K(X_i \oplus Y_{i-1}) \tag{2}$$

where Y_0 = IV is the initializing vector and $\oplus$ indicates bit-by-bit modulo 2 addition or the equivalent Exclusive-OR operation. In this case, the plaintext can be recovered with the operation:

$$X_i = Y_{i-1} \oplus D_K(Y_i) \tag{3}$$

The Manipulation Detection Code (MDC) for this case is defined as

$$MDC = X_{n+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_n$$

resulting in a message of $X = X_1, X_2, \ldots X_n, X_{n+1}$. (4)

To check for the authenticity of the received encrypted message

$$Y = Y_1, Y_2, \ldots, Y_n, Y_{n+1}$$

we first decipher the message Y to obtain the possibly spurious plaintext

$$Xrec = Xrec_1, Xrec_2, \ldots, Xrec_n, Xrec_{n+1} \tag{5}$$

Next the MDC for the received plaintext is recomputed, i.e.,

$$MDCrec = Xrec_1 \oplus Xrec_2 \oplus \ldots \oplus Xrec_n$$

and is tested for equality with the received MDC, i.e., $Xrec_{n+1}$. If both are identical, the message is presumably authentic; otherwise it is rejected. An equivalent test is to check if the summation of all X, modulo 2, is equal to zero or not.

To show that this procedure does not detect certain data manipulation, X_i is expressed in terms of the Y_i according to Equation 3 to provide the expression for

the sum, S, of all X as follows:

$$S = X_1 \oplus X_2 \oplus \ldots \oplus X_n \oplus X_{n+1}$$
$$= (IV \oplus D_K(Y_1)) \oplus (Y_1 \oplus D_K(Y_2))$$
$$\oplus \ldots \oplus (Y_{i-1} \oplus D_K(Y_i))$$
$$\oplus \ldots \oplus (Y_n \oplus D_K(Y_{n+1}))$$

(6)

Since the terms in Equation 6 can be summed modulo 2 in any order, it follows at once that the MDC would not change if the ciphertext blocks were permuted, with the exception of the last block, Y_{n+1}. Thus the proposed modulo 2 addition or block-wise Exclusive-OR technique for computing the MDC would not detect various kinds of rearrangement of the text occurring on 64 bit boundaries. Furthermore, if additional data, Y*, is inserted such that

$$Y^*_j \oplus D_K(Y^*_j) = 0 \text{ for } j = 1, 2 \ldots, n;$$

(7)

then such an insertion can not be detected either. The simplest way of satisfying Equation 7 is by inserting the spurious data Y* into two different places in the message (on 64-bit boundaries, and prior to the last block).

Thus the block-wise Exclusive-OR (modulo 2 addition) technique for computing a Manipulation Detection Code (MDC) fails to detect the transposition or pair-wise insertion of spurious blocks of ciphertext when used with Cipher Block Chaining, and therefore must be rejected.

MDC Based On Modulo 2 Addition In The Case Of Cipher Feedback Mode

In the Cipher Feedback (CFB) mode of operation, ciphertext is fed back into the algorithm to generate a cryptographic bit stream, R. In particular, if k bits are fed back the operation is defined as k-bit cipher feedback. To obtain the ciphertext, Y, the plaintext, X is added modulo 2 to R, k bits at a time., i.e.,

$$Y_i = X_i \oplus R_i$$

(8)

The functional dependency of R_i on the previous ciphertext depends on how many bits (k) are fed back each iteration. Commonly used values of k are k=1 and k=8, but to simplify the analysis the case k=32 will be investigated first. The conclusions drawn for k=32 also apply for k ranging from 1 to the block size of the DES algorithm, i.e., k=64. In all cases it will be concluded that the proposed modulo 2 addition method of MDC generation is not sufficiently secure in the case of Cipher Feedback mode.

For k=32 the relationship between plaintext and ciphertext is as follows:

$$Y_i = X_i \oplus E_K(Y_{i-2}, Y_{i-1}); \quad i = 2, 3, \ldots,$$

(9)

where $Y_0 = IV_1$, the first 32 bits of the initializing vector, $Y_1 = IV_2$, the second 32 bits of the 64-bit IV, and the rest of the Y's are successive blocks of ciphertext of 32 bits each. The encryption operator has two arguments, corresponding to the left and right halves of the 64-bit input to the DES algorithm.

The plaintext can be recovered from the ciphertext with the following operations:

$$X_i = Y_i \oplus E_K(Y_{i-2}, Y_{i-1}); \; i = 2, 3, \ldots, n$$

(10)

Defining, again,

$$MDC = X_1 \oplus X_2 \oplus \ldots \oplus X_n = X_{n+1};$$

(11)

and for convenience,

$$S = X_1 \oplus X_2 \oplus \ldots \oplus X_n \oplus X_{n+1};$$

(12)

it follows that S equals 0 for the plaintext message to be transmitted.

Expressing S in terms of Y with the aid of Equation 10 results in

$$S = (Y_1 \oplus E_K(IV_1, IV_2)) \oplus (Y_2 \oplus E_K(IV_2, Y_1)) \oplus$$
$$(Y_3 \oplus E_K(Y_1, Y_2)) \oplus \ldots \oplus$$
$$(Y_i \oplus E_K(Y_{i-2}, Y_{i-1})) \oplus \ldots \oplus$$
$$(Y_{n+1} \oplus E_K(Y_{n-1}, Y_n))$$

(13)

We now notice that unlike the case of Cipher Block Chaining, rearrangement of the transmitted ciphertext blocks can now be detected, because the blocks are chained due to the term $E_K(Y_{i-1}, Y_i)$. Thus the simple permutation attack used against CBC no longer works.

However, insertion of additional blocks can be achieved such that Equation 13 does not change. For example, let Y_3 be replaced by the concatenation of Y^*_3, Y_2, Y^*_3, Y_2, Y_3; where Y^*_3 is an arbitrary quantity.

The correct term $Y_3 \oplus E_K(Y_2, Y_3)$ in S is then replaced by the terms

$$Y^*_3 \oplus E_K(Y_2, Y^*_3) \oplus Y_2 \oplus E_K(Y^*_3, Y_2) \oplus$$
$$Y^*_3 \oplus E_K(Y_2, Y^*_3) \oplus Y_2 \oplus E_K(Y^*_3, Y_2) \oplus$$
$$Y_3 \oplus E_K(Y_2, Y_3)$$

which are equal, due to cancellation, to

$$Y_3 \oplus E_K(Y_2, Y_3)$$

and thus detection of these insertions is not possible.

This analysis has shown that for 32-bit Cipher Feedback mode, although the transposition attack that worked against CBC mode is detected by the block-wise Exclusive-OR (modulo 2 addition) MDC technique, it is possible to insert arbitrary blocks of data into the message without detection. The authors (Meyer and Matyas) have also extended this analysis to cover the case of multiple insertion attacks, attacks against 8-bit cipher feedback, and arbitrary values of k, with equivalent results.

In general, it can be concluded that j blocks of ciphertext can be inserted preceding Y_i provided that (1) the arguments of $E_K(\ )$ in Equation 13 do not contain components of the initializing vector; and (2) the last ciphertext block, Y_{n+1}, remains unchanged.

However, it should be pointed out that it would be essentially impossible to insert a particular desired quantity into the message without knowing the key, because the decryption of the inserted block is unpredictable (except possibly for the known plaintext attack of Coppersmith to be described later). Nonetheless, insertion of random bits into the decrypted plaintext may be sufficient from the attacker's point of view, for without detailed syntactical or semantic analysis the modification may escape notice. For example, most people would feel that exchanging a random number distributed uniformly between zero and the maximum field size for the amount-of-money-deposited field in a banking transaction would almost always be profitable.

Thus, although Cipher Feedback mode affords improved protection against transposition attacks when used with the block-wise Exclusive-OR (modulo 2 addition) MDC technique, it fails to protect against the insertion of bits into the ciphertext, causing random data to appear in the decrypted plaintext. **The proposed Exclusive-OR MDC technique must therefore be rejected for the Cipher Feedback case as well.**

MDC Based On Modulo 2 Addition In The Case Of Output Feedback Mode

In the DES Output Feedback (OFB) mode of operation (also called Key Auto-Key) the 64 bit cryptographic bit stream, R, which is generated by the algorithm is fed back directly into the input of DES in order to generate additional bits, i.e.,

$$R_1 = E_K(IV)$$
$$R_i = E_K(R_{i-1}) \ ; \quad i = 2, 3, \ldots, n$$

where IV, the initializing vector, is 64 bits long.

To obtain the ciphertext, Y, the plaintext is Exclusive-ORed with the cryptographic bit stream, R, k bits at a time, i.e.,

$$Y_i = X_i \oplus R_i \ ; \quad i = 1, 2, \ldots, n$$

where X_i, Y_i, and R_i represent blocks of bits of length k, assumed to be equal to the block size of the algorithm for efficiency (k=64). (It should be noted that FIPS PUB 81 includes a variation on this traditional method, whereby only k bits of R are fed back each time, and only k bits of the plaintext are Exclusive-ORed with the keystream (the k bit subset of R), to produce k bits of ciphertext each DES cycle. However, use of OFB with feedback of other than k=64 bits has been shown to cause the DES keystream to repeat after unacceptably short cycle lengths, on the order of 2 billion iterations or $1.37*10^{11}$ bits[10], thereby giving rise to a possible transmission-in-depth compromise[11]. For that reason, use of other than k=64 bit feedback has been rejected by ISO, and will not be discussed further.)

The plaintext is recovered from the ciphertext by repeating the encipherment operation, since the Exclusive-OR (modulo 2 addition) is self-canceling upon repetition:

$$X_1 = Y_1 \oplus E_K(IV)$$
$$X_i = Y_i \oplus R_i \ ; \quad i = 2, 3, \ldots, n \qquad (14)$$

Once again defining

$$MDC = X_1 \oplus X_2 \oplus \ldots \oplus X_n = X_{n+1};$$

and

$$S = X_1 \oplus X_2 \oplus \ldots \oplus X_n \oplus X_{n+1}$$

it follows that S equals 0 for the plaintext message to be transmitted.

Expressing S in terms of Y with the aid of Equation 14 results in

$$S = Y_1 \oplus Y_{n+1} \oplus E_K(IV) \oplus E_K(R_n) \oplus$$
$$SUM[(Y_i \oplus E_K(R_{i-1})); \ i = 2, 3, \ldots, n] \qquad (15)$$

Equation 15 indicates that any modification of the data resulting from the modulo 2 summation

$$SUM\ [Y_i \text{ for } i = 2, 3, \ldots, n] = \text{constant}$$

will not be detected.

Since this value can be calculated from intercepted data, there are numerous ways to modify the Y's. The transposition attack discussed in conjunction with CBC works easily, as well as much more general manipulations of the data than are possible with either CBC or CFB, because only the Y_i terms have to be considered. For example, if Y_i and Y_j are replaced by $Y_i \oplus T$ and $Y_j \oplus T$, respectively, where T is any arbitrary bit pattern, the modification will not be detected by this MDC technique, whereas in the other modes of operation (CBC and CFB) encrypted Y terms also come into play, introducing restrictions on the possible attacks by requiring cancellation terms.

With respect to the transposition and/or bit modification attacks, Output Feedback mode is the weakest of all from the standpoint of acceptability of the proposed block-wise Exclusive-OR (modulo 2 addition) MDC scheme.

On the other hand, OFB is immune to the insertion attack, since inserting additional Y's will be detected because unpredictable terms $E_K(Y_i)$ would then be generated. These terms cannot be manipulated externally since their generation does not involve ciphertext, only the cryptographic bit stream produced internally. (Although these terms show up during the decryption operation, the terms are in fact $E_K(\ )$, because OFB uses the encryption mode of DES during both the encryption and decryption operations.)

To summarize our conclusions so far, we have the following vulnerabilities:

Vulnerabilities of Exclusive-OR MDC to Unknown Plaintext Attacks:

DES Mode:	Type of attack, vulnerability:
CBC	Permutation = **yes**; Substitution = **no**; Insertion = **yes**
CFB	Permutation = **no***; Substitution = **no**; Insertion = **yes**
OFB	Permutation = **yes**; Substitution = **yes**; Insertion = **no**

*If k=64 then CFB is vulnerable to a permutation attack, since CFB is very similar to CBC in this case.

Architectural Considerations Concerning Manipulation Detection Codes

The objectives of generating and using an MDC are two-fold. The MDC must be of sufficient power to detect all manipulation or corruption of the transmitted information, but it should also be simple enough to have only a slight impact upon performance. If the performance of the MDC were not important (and if the architectural disadvantages of the use of two keys and potentially two crypto-algorithms were of no concern), one might as well use a purely cryptographic approach that directly involve the use of a secret key, i.e., a MAC, which is thought to be sufficiently complex in its use of forward error propagation as to avoid the above difficulties. Where secrecy of the data is required one would in that case encrypt first, and then generate a Message Authentication Code (MAC) on the encrypted data, or else generate a MAC first and then encrypt. This approach thus requires two multiple-iteration encryption steps with, so far at least, two different keys and different initializing vectors. The advantage of this method is its cryptographic strength, the disadvantages being the presumed length of time required to perform the second set of encryption operations, the additional complexities of key management of two different (sets of) keys, and certain architectural considerations which will now be discussed.

One of the questions which is currently being addressed, both by various standards organizations and by major manufacturers of computers and/or communication equipment, is just where in a large system the cryptographic protection can best be provided. Most of the discussion to date has focussed on the use of cryptography to protect the secrecy of communications, with little effort being devoted as yet to considerations of authentication and its impact on communications architecture. Even so, there are significant differences in approach that have been recommended, ranging from the integration of encryption at the very highest level or layer (the application or end user level) of the system, in order to provide end-to-end protection of communications together with incorporation of compartmented or Need-To-Know protection of the information; down to the implementation of high speed bulk encryption at the physical link level, in order to gain the economies of scale that can be obtained by encrypting voice, data, and perhaps image traffic en masse (together with whatever additional benefits may accrue from the protection against traffic analysis that such a system may permit.)

One of the most powerful concepts to have emerged recently is that of a layered communications architecture as exemplified by the ISO Open Systems Interconnection Reference Model, wherein the various functions of a communications system are clearly separated into different layers with defined interfaces

between those layers. The advantage of such an approach lies in the apparent ease with which different physical implementations of the particular layers can be specified and presumably changed, without causing all of the rest of the system to be perturbed.

As we will see, however, acceptance of these architectural principles will have an impact upon the choice of a Manipulation Detection Code, or alternatively a Message Authentication Code.

At this time, there is no clear winner insofar as the preferred DES mode of operation is concerned (CBC, CFB, or OFB). All of the modes have their adherents and champions, and each can marshall good arguments based upon the particular application.

For high speed encryption of data (either for data communication or for file encryption on various types of media), Cipher Block Chaining is the leading contender because it encrypts 64 bits of data with one operation, and can therefore be used at data rates approaching 10 Mbps with existing high speed chips. It also provides for self-synchronization after 64 bits, which is important for rapid error recovery in the event of the loss of bit synchronization. (This self-synchronization is, however, the root cause of the difficulties with the MDC codes, both for CBC and CFB.) In addition, CBC mode has certain advantages for end-to-end encryption and file encryption in the area of key management (assuming the generation of session keys or files keys from master keys), since under these conditions an implicit IV of zero can be used. The use of CFB would require additional overhead, and would require special precautions against accidentally restarting from the same IV, for that would lead to a transmission-in-depth and a possible compromise.

For medium speed communications, especially link encryption of low-speed, unbuffered asynchronous terminals (or smart terminals that emulate dumb ones), either 1-bit or 8-bit Cipher Feedback mode is probably preferable from the standpoint of transparency, standardization, and the imposition of a minimal delay on the data. The use of CFB for file encryption is generally awkward, unless the entire database is encrypted serially (like a tape.)

For applications where the error-propagation properties of CBC or CFB might be intolerable, such as for the bulk encryption of voice, facsimile, and television, and where the resynchronization problem can be handled easily (for example in a Time-Division Multiple Access (TDMA) satellite system), Output Feedback mode is clearly superior to the other modes.

In addition to these considerations, real-life compromises must often be made in the design and implementation of security systems. For example, it might be decided that the threat of microwave or satellite monitoring is sufficiently real as to require the use of bulk encryption at the physical link level on all circuits using those media as a minimum, regardless of the sensitivity level of the traffic being handled. In addition, sensitive applications may require the use of approved, Federal Standard 1027[12] compliant encryption units on an end-to-end basis to assure adequate secrecy, while other applications may require authentication without secrecy, or both authentication and secrecy.

In general, the complexities of system design are such that authentication and secrecy may often have to be implemented in two different architectural layers, using different hardware devices or perhaps a combination of hardware and software. In particular, because the need for authentication is not as common as the need for secrecy (except perhaps in the banking industry, where the situation is reversed), it is possible in many cases that if authentication is to be provided at all it will have to be implemented in software at the End User or perhaps the Presentation Services or Transmission Services level, completely independent of the method of encryption chosen for secrecy.

The preferred solution to this problem of separation of function between authentication and secrecy would be to use a separately generated Message Authentication Code, but because the authentication code must in general be generated first (in the application layer of code, typically, for end-to-end applications) it will be both necessary and desirable to use two independent and unique crypto keys. However, the availability of appropriate hardware, cost considerations, key management difficulties, etc. may rule out this approach and require a software solution. Although the question of the relative speed and efficiency of a hardware generated Message Authentication Code versus some simpler Manipulation Detection Code will require detailed timings in order to resolve, there is no question that running DES in software to generate a MAC may be unacceptably slow for certain applications.

(There is a very limited amount of data regarding encryption performance that is presently available. Without trying to define all of the various circumstances, operating system characteristics, etc., a rough rule of thumb for large main-frame computers with external (channel accessed) encryption equipment would be that the time required to encrypt any reasonable length message is essentially fixed and dominated by the operating system overhead required to manage keys, etc.; whereas an in-line software implementation may actually be faster on the same machine for a small number of DES blocks. For small messages (64 bytes using CBC), the time is approximately the same (600-700 microseconds on an IBM 3033) for either hardware or software implementations, but the software execution time grows linearly with the message length. The timings and cross-over points for microprocessor applications (say a reasonably fast 8088 Fortran implementation versus the inexpensive Western Digital chip) are not yet known to the authors, but the basic hardware execution time for the chip is around 28 microseconds per encryption and the software overhead should be minimal; whereas executing DES in 8088 software should be significantly slower than on a 3033. Implementing all of the necessary checking and key management to be comparable to the main-frame implementation would slow down the micro significantly, of course.)

For all of the above reasons, we can establish a set of criteria for a desired MDC or MAC. The reader should bear in mind that the conclusions obtained as a result of the subsequent analysis depend on the stated criteria, and might be subject to change if the criteria were to be changed:

1. The method of authentication should satisfy the constraints of the mathematical statement of the message authentication objectives outlined at the beginning of the paper.

2. The method of authentication or manipulation detection used should be independent of the method used to protect the secrecy of the message. In particular, it should be possible to authenticate the message with complete confidence, independent of the encryption algorithm or mode of operation used for secrecy, i.e., no matter whether CBC, CFB, or OFB is used. The implementation of authentication/manipulation detection should not have to be changed if the encryption mode changes.

3. It would be desirable if authentication/manipulation detection could be implemented in application software with reasonable efficiency, and not require dedicated hardware.

4. Conversely, a solution that would be economical to implement in hardware would also be desirable.

Using these desiderata, we can evaluate several possible MDC algorithms to replace the block-wise Exclusive-OR function previously suggested.

A k-Bit Linear Addition MDC

An MDC function which would be almost as simple as the block-wise Exclusive-OR (modulo 2 addition) would be linear addition modulo 2^k, as first suggested by Meyer and Matyas following the discovery of the difficulty with the Exclusive-OR MDC. (See also Meijer and Akl[13].) We will denote this operation with the conventional + sign, but it must be remembered that there is no carry past the left-most (k-th) bit. To make the initial discussion easier we will assume k=64 for DES. We will also assume that the plaintext is padded with zero bits, if necessary, to have a length which is a multiple of k. The different modes of operation are discussed separately below.

MDC Based on Linear Addition with Cipher Block Chaining

Since in this case the block size is equal to 64, Equation 1 can be modified in a straight-forward way as follows:

$$MDC = X_1 + X_2 + \ldots + X_n, \text{ and} \tag{16}$$

$$S = X_1 + X_2 + \ldots + X_n + X_{n+1} \tag{17}$$

Similarly, in terms of the Y's, Equation 6 is transformed to

$$\begin{aligned} S = {} & (IV \oplus D_K(Y_1)) + (Y_1 \oplus D_K(Y_2)) \\ & + \ldots + (Y_{i-1} \oplus D_K(Y_i)) \\ & + \ldots + (Y_n \oplus D_K(Y_{n+1})). \end{aligned} \tag{18}$$

Since $\oplus$ and + are now different operations, the terms in Equation 18 do not cancel and thus it cannot be simplified to the form of Equation 6. As a result, we can see that linear addition modulo k is sufficient to defeat the transposition attack against Cipher Block Chaining. Similarly, the data insertion attack is also defeated.

However, it should be noted that this MDC technique does have the peculiar property that prefixing 2^i (i = 1, 2, ..., k) repetitions of the initializing vector to the front of the ciphertext, or inserting 2^i blocks of ciphertext C_j between ciphertext blocks C_j and C_{j+1} will result in an altered ciphertext with a probability of escaping detection (P), that is higher than obtainable strictly by random chance (i.e., 2^{-k}). In particular, if 2^k blocks are added to the ciphertext in one of the ways just outlined, then P = 1; if 2^{k-1} blocks are prefixed to the ciphertext or 2^{k-1} blocks of ciphertext are inserted as described, then P = 1/2, and so forth, or in general $P = 1/2^{k-i}$. This seemingly strange result is due to the fact that the linear addition is performed ignoring the high order carry. Thus, if a block of plaintext that results from the decryption of C_j has a low order one bit, 2^k iterations later that one bit will be shifted out. Since there is a 50% probability that the block had a 0 in the low order bit, the probability is 1/2 that 2^{k-1} iterations would have sufficed, etc.

Unfortunately, this possibility violates one of our desiderata, namely that the chance of any single bit being modified without detection should be a

uniformly distributed random variable with an average value of 2^{-k}, independent of the message content. This problem could be fixed by using linear addition with a wrap-around carry into the units bit in the event of overflow, or by using a value of k that is considerably larger than 64 so that no carry would ever occur, such as k=128. (A value of k=80 would be sufficient for most messages, but would become marginal for messages larger than 500K bytes.)

MDC Based upon Linear Addition with Cipher Feedback.

Assuming for simplicity of the discussion that 32 bit cipher feedback is used, and that the MDC is generated using modulo 2^{64} addition, Equation 13 can be rewritten as follows, noting that the two 32 bit blocks are now concatenated to perform the modulo 2^{64} operation:

$$\begin{aligned} S = {} & (Y_1 \oplus E_K(IV_1,IV_2)) \,||\, (Y_2 \oplus E_K(IV_2,Y_1)) + \\ & (Y_3 \oplus E_K(Y_1,Y_2)) \,||\, (Y_4 \oplus E_K(Y_2,Y_3)) + \ldots + \\ & (Y_i \oplus E_K(Y_{i-2},Y_{i-1})) \,||\, (Y_{i+1} \oplus E_K(Y_{i-1},Y_i)) + \ldots + \\ & (Y_n \oplus E_K(Y_{n-2},Y_{n-1})) \,||\, (Y_{n+1} \oplus E_K(Y_{n-1},Y_n)) \end{aligned} \tag{19}$$

Although it appears that Equation 19 mixes the Exclusive-OR with linear addition in a manner that would be very difficult to untangle, a rigorous analysis has not yet been completed. We cannot say with a certainty, therefore, whether this approach to an MDC is sufficiently secure in the case of CFB. In any case, as the next section points out, the linear addition technique is not totally satisfactory from an architectural standpoint, because it does not withstand bit manipulation attacks with Output Feedback mode, and is subject to known plaintext attacks as well.

MDC Based on Linear Addition with Output Feedback Mode.

Assuming the use of 64-bit Output Feedback and linear addition modulo 2^{64} for simplicity, Equation 15 can be rewritten as follows:

$$\begin{aligned} S = {} & (Y_1 \oplus E_K(IV)) + (Y_2 \oplus E_K(R_1)) + \ldots + \\ & (Y_n \oplus E_K(R_{n-1})) + (Y_{n+1} \oplus E_K(R_n)); \end{aligned} \tag{20}$$

Although the error terms of Equation 20 look quite complex, there are several attacks that would be possible. First, although the particular message may have been enciphered for secrecy, the plaintext might nonetheless be known because the originator of the message is trying to perpetrate a fraud (e.g. in the case of a banking transaction). In that case the original MDC can easily be calculated from the known plaintext, and the final block of keystream computed from the calculated MDC and the observed ciphertext:

$$R_{n+1} = MDC \oplus Y_{n+1}$$

(The remainder of the keystream is easily computed from the known plaintext and intercepted ciphertext.)

Then, knowing the keystream output, any modification whatsoever can be made and the spurious MDC altered to suit. With only two 64-bit blocks of known plaintext available, an attack is still possible. Let $B = b_1, b_2, \ldots, b_{64}$ and $C = c_1, c_2, \ldots, c_{64}$ denote two blocks of known plaintext in a transmitted message. All one needs to do is find bits b_i and c_i (for i = 1, 2, ..., 64) such that b_i is not equal to c_i, and invert the corresponding ciphertext bits (changing 0 to 1 and vice versa.) The error term, or net contribution to the MDC, is therefore guaranteed to be 0. More generally, any number of bits in the ciphertext corresponding to B and C can be inverted so long as the conditions stated above are met.

If the adversary intercepts the ciphertext but does not know the underlying plaintext, a variation on the attack is possible (provided the opponent knows the MDC is calculated using modulo 2^{64} addition.) Let Y and Y* be two 64-bit sequences in the intercepted ciphertext such that the distance between them is a multiple of 64. Y and Y* may or may not fall on block boundaries. Let $B = b_1, b_2, \ldots, b_{64}$ and $C = c_1, c_2, \ldots, c_{64}$ denote the unknown plaintext corresponding to $Y = y_1, y_2, \ldots, y_{64}$ and $Y^* = y_1^*, y_2^*, \ldots, y_{64}^*$. If bits y_i and y_i^* are inverted, we are guaranteed that on decipherment, plaintext bits b_i and c_i are also inverted. Now if $b_i = 0$ and $c_i = 1$, or $b_i = 1$ and $c_i = 0$, we are additionally guaranteed that the modification will be self-cancelling and go undetected with respect to an MDC calculated using modulo 2^{64} addition.

If B and C consist of random bits, then the modification will be undetected with probability 1/2. If two pairs of random bits are inverted, the modification will go undetected with probability 1/4, with three pairs the probability is 1/8, etc.

In reality the plaintext bits will not be randomly distributed, but will vary as a function of the text

and the coding scheme used (ASCII or whatever), so the probabilities 1/2, 1/4, 1/8 are only a first order approximation. However, regardless of what the actual probabilities are for the underlying message space, it is clear that such an attack is possible and that the probability of success is dependent on the number of inverted ciphertext bits and the distribution of 0 and 1 bits in the plaintext messages.

It is also interesting that if the 64-bit block boundaries are known, changes can be made to the ciphertext that are guaranteed to escape detection. All that one needs to do is invert the high order bits in any two blocks of ciphertext. Since there is no carry out of the high order bit positions, the inverted bits in the recovered plaintext blocks will result in a net contribution of 0 to the MDC.

In summary, a Manipulation Detection Technique based upon the use of linear addition modulo 2^k would detect all transposition and most insertion attacks in the case of Cipher Block Chaining and Cipher Feedback, but the technique is not sound in the case of Output Feedback. Further, we shall see in the next section that it does not stand up under a number of known plaintext attacks, at least when used with a common key across multiple messages. It therefore does not meet our criteria for the absolute detection of any modification. **Therefore, on the basis of the desiderata this technique should be rejected, at least in its present form.**

Other Attacks Against Manipulation Detection Codes

In addition to the above attacks there are other more subtle attacks that will succeed against a broad general class of Manipulation Detection Codes (including for example the linear addition MDC technique with Cipher Block Chaining), if the opponent has some knowledge of the plaintext.

The first one, suggested by Miles Smid, requires that the opponent has knowledge of the plaintext to be modified and the capability to encrypt in real time with the unknown secret key.

The attack by Smid assumes that a legitimate user of the system attacks his own messages. One might argue that such a scenario is not meaningful, since a user who can exercise the cryptographic function could generate any authenticated message he desires. There are, however situations where users may not have that freedom, in which case the above scenario is a valid concern.

Consider, for example, an environment where users of a terminal or an originating host (with an installed secret key) can input data to the terminal that in turn is formatted into a message, encrypted, and then sent to a receiving host for processing. Further suppose that the terminal (or originating host) examines all formatted messages prior to encrypting and sending them to the host to ensure that they pass certain system-defined consistency checks. A message that fails a consistency check is automatically rejected by the terminal or originating host. (An Automated Teller Machine (ATM) would be a good example of this kind of environment.)

In such an environment, the set of all possible messages can be divided into two subsets - those that will pass the consistency checks (good messages) and those that will not (bad messages). Suppose now that the objective of an adversary (whom we will assume is a user of the terminal or his accomplice) is to devise a set of good messages whose ciphertext is to be intercepted (and possibly jammed from reaching the host system) such that by "cutting and splicing" it is possible to form a bad message that can then be injected back into the communications path and be authenticated properly by the host system. (We assume that the system-defined consistency checks performed at the terminal or originating host, e.g. identity checks or account balance checks, are not duplicated at the receiving host.)

Clearly, a good authentication procedure would not allow such an attack, even though the envisioned protocol is admittedly hypothetical. It is shown next that the MDC approach based on modulo 2^{64} addition yields to this attack.

Let $X = X_1, X_2, \ldots, X_{n+1}$ and $Y = Y_1, Y_2, \ldots, Y_{n+1}$ denote the plaintext and ciphertext blocks, respectively, where X_{n+1} = MDC. We assume that Y is intercepted and that MDC can be calculated from the known plaintext $X_1, X_2, \ldots X_n$. From the relation, valid for CBC,

$$X_i = Y_{i-1} \oplus D_K(Y_i); \qquad \text{for } i = 1, 2, \ldots, n+1$$

It follows that the effect of permuting two ciphertext blocks by exchanging Y_i with Y_j can be evaluated as follows:

$$X_i^* = Y_{i-1} \oplus D_K(Y_j)$$
$$X_j^* = Y_{j-1} \oplus D_K(Y_i)$$
$$X_{i+1}^* = Y_j \oplus D_K(Y_{i+1})$$
$$X_{j+1}^* = Y_i \oplus D_k(Y_{j+1})$$

(Note that a change in Y_i affects X_i as well as X_{i+1} upon decryption, which accounts for the i+1st and j+1st terms in addition to the ith and jth terms.)

Using the four plaintext blocks X_i, X_{i+1}, X_j, and X_{j+1}, the quantities $D_K(Y_i)$, $D_K(Y_{i+1})$, $D_K(Y_j)$, and $D_K(Y_{j+1})$ can be evaluated. This in turn allows the computation of X_i^*, X_{i+1}^*, X_j^*, and X_{j+1}^* and thus the effect on the MDC can be calculated, i.e.,

$$\text{MDCnew} = \text{MDCold} + \text{Error} \pmod{2^{64}}$$

where

$$\text{Error} = (X_i^* - X_i) + (X_j^* - X_j) + (X_{i+1}^* - X_{i+1}) + (X_{j+1}^* - X_{j+1})$$

As a next step the system is asked to encrypt MDCnew $\oplus$ IV $\oplus$ Y_n as the first block of a message,

i.e., a selected plaintext attack must be accomplished. The first block of ciphertext, which is intercepted by the adversary, is just $E_K(\text{MDCnew} \oplus Y_n)$.

Now if the original ciphertext Y_1, Y_2, ..., Y_n, Y_{n+1} is modified by interchanging blocks Y_i and Y_j, and block Y_{n+1} is replaced by block $E_K(\text{MDCnew} \oplus Y_n)$ (i.e., the first block of ciphertext created by the chosen plaintext attack), then the MDC check will still be passed. Although the attacker has never sent that particular message before, he has been able to splice together a message from pieces of others that will pass the authentication test.

Variations on the above attack involving added or deleted blocks of ciphertext, or transposed blocks are also possible. **The underlying idea in all the attacks is that the error term can be evaluated and a new MDC calculated.** Once this is done a chosen-plaintext attack is used to obtain the ciphertext corresponding to the new MDC. Finally, a modified ciphertext is constructed that will be guaranteed to pass the MDC check.

A different approach (defined here as the under-determined knapsack method), has been suggested by Don Coppersmith. He leaves the original message blocks including the MDC unmodified, but inserts additional blocks into the message in such a way that the error term is equal to zero.

In Coppersmith's attack, one block of the particular message being attacked, and approximately 13 random known plaintext/ ciphertext pairs encrypted using the same key as the message to be attacked will be required. With such a set of plaintext/ciphertext pairs, enough combinations of plaintext are available that there is an excellent chance of finding a set of "garbage" blocks that would contribute an error term of zero.

The basic idea is to use a divide-and-conquer approach, splitting the problem of finding some set or permutation of known plaintext that will contribute a zero error term into two parts, then use a "birthday problem" approach to reduce the number of trials needed to an order of magnitude of $2^{64/2} = 2^{32}$, or a few billion operations, and then generate random permutations until those conditions are met. This is analogous to the classic birthday problem[14], with a modification. Given two independent samples of random k-bit variables, each of size M, the probability that some k-bit quantity appears in both samples is approaches 63% when $m = 2^{k/2}$, and increases rapidly beyond that point.

To obtain the required number of permutations, one needs 2t+1 blocks in CBC mode such that $t! > 2^{32}$, or $t \geq 13$ (84 bytes). If T blocks of chosen or intelligible text are required to be inserted, then T+2t+1 blocks will be required. (The calculation takes into account one additional block of known data, which accounts for the 1 in the t+1 and T+2t+1 terms, to provide a bridge between the two sets of plaintext-ciphertext pairs of 13 blocks each.) Although 27 blocks are required in total, only 13 unique blocks of known plaintext-ciphertext pairs are required, since the same 13 blocks can be permuted in different ways before and after the known message block.

Knowledge of 13 ciphertext blocks, Y_i, and 13 corresponding plaintext blocks, X_i, would enable one to obtain 13! (6.23 billion) permutations. For each of these permutations, the error contribution to the MDC is calculated. Since the actual 64-bit block (IV or ciphertext) that must be Exclusive-ORed to the first block of inserted plaintext is not known at this time, a constant value such as zero is assumed in calculating the MDC contribution. (If it were required to add intelligible text at some point in the message, as well as the required "garbage", the intelligible text would be appended to the end of each permutation of the 13 blocks. The resulting contribution is then calculated on each of these so-formed strings of text. Thus the text inserted into the message would consist of garbage words followed by intelligible text followed by garbage words.) The table of 13! values must be modified on-the-fly once the actual input value is known (the initializing vector if the added data preceded the message, otherwise the intercepted ciphertext) in order to be effective.

A similar evaluation of the MDC contribution is also done with a second set of 13 known plaintext-ciphertext blocks. Since it is not known at this time what the actual plaintext block is, to which the last block of inserted ciphertext must be Exclusive-ORed, a constant input parameter is again assumed, e.g., zero. Modification of these values must again be performed on-the-fly, using one given plaintext block in the intercepted message.

The pre-computed table values can thus be modified only when the actual encrypted message and at least one known block of plaintext within the message have been obtained. Subsequently, a search is made to find a value in one set that matches a value in the other set, which means that the contribution to the MDC of the inserted data is zero.

In addition to the requirement to know at least one block of plaintext in the intercepted message and to be able to perform these on-the-fly computations (consisting of about 2^{33} or 8 billion table modifications and about 2^{32} trials to match two 64-bit patterns in two tables of 2^{32} entries each, assuming k = 64), a transmission delay of the intercepted data must be tolerable in order to perform the calculation to evaluate the data to be inserted. For this reason, the attack may not be feasible in the case of interactive sessions, but might well succeed in the case of an attack against a stored, encrypted file. There is also the garbage part of the data to be added which would usually be detectable by other means, but our desiderata ruled out relying on the presence of such context sensitive checks on general principles.

It should be noticed that this attack would work against any form of MDC, **so long as the error term is computable given the known plaintext.** Note that it is not necessary to be able to compute the MDC itself. It suffices to be able to compute the error term contributed to the MDC by the added blocks, so that that term can be canceled by the second set of blocks. (Variations of the attack would allow deletion and transposition as well.)

It might be argued that these attacks are far from straightforward, that detection of the modification is likely to occur by other means, and that the necessary plaintext-ciphertext combinations are not likely to be available in most applications. Nonetheless, the most interesting applications of authentication techniques occur in the military (presumably), and in commercial transactions such as banking, where the stakes may be very high indeed. In particular, as Meyer and Matyas have discussed[15], as networks of banks and bank holding companies come into existence to implement regional and even national Electronic Funds Transfer systems, the banks may not even trust each other, nor their various employees, and certainly not their ATM customers, all of whom might have access to known plaintext/ciphertext pairs. In particular, although the use of session key techniques generally would limit the use of a given key to two correspondents, those two correspondents may be relatively "dumb" application programs, not the human end-user, and furthermore, the sessions might have quite a long duration - perhaps even for weeks at a time. In this environment, sessions are chained together between peer host machines, and messages may be passed on from one session to another until they arrive at their destination(s). As this very large and complex network is put together, it is unlikely that every institution will implement authentication right from the start, nor is it likely that all of the various key management problems can be worked out so that end-to-end encryption and authentication will be possible. There is therefore a possibility that one of the above attacks might be effective against such a system.

(It is conceivable that these attacks could be thwarted by using a unique key per message to be authenticated, thereby presumably eliminating the possibility of collecting the 13 known plaintext-ciphertext pairs, but the key management difficulties such a scheme would entail would have to be thought through very carefully, including how the key changes themselves were to be authenticated.)

A Quadratic Congruential Manipulation Detection Code

If we consider the reasons why the previously proposed MDC techniques have fallen short, they amount to these:

1. None of the standard or "approved" DES Modes of Operation provide for infinite error propagation ("garble extension") **during decryption**, although they do so during encryption. More elaborate DES modes have been suggested (e.g., by Meyer and Matyas) involving cross-linking of the plaintext and ciphertext during each iteration. However, the introduction of garble extension would necessitate giving up the self-synchronization-after-error properties of CBC and CFB, one of the most important reasons for using those modes.

2. Neither the Exclusive-OR nor the linear addition modulo 2^k techniques introduce any positional dependency or non-linearity in the generation of the MDC. Just as in the case of a good crypto-algorithm, it appears important to introduce as much error-extension with non-linear methods as quickly as possible, and to involve all of the bits of the message in a uniform manner.

3. Finally, it is necessary to find some method which prevents the computation of the MDC error term or the MDC itself due to the modification of an intercepted message, even for the case where the opponent may have complete knowledge of the plaintext.

A potentially promising MDC technique previously described by one of the authors[16] has been designed to avoid these difficulties, and will be elaborated upon here. Called a Quadratic Congruential Manipulation Detection Code (QCMDC), it is defined in equation form as follows.

Let the message be composed of n m-bit blocks X_1, X_2, ..., X_n, optionally including a secret seed, S, in block X_1, plus the MDC in block X_{n+1}. Let there be a prime modulus N, and an accumulator Z which is initialized to some secret value C; where N, C, and S are known at the sending and receiving ends, but where C (and optionally S) are not known outside of the operating system or application program, i.e., they are hidden from both the human end-user and any other adversary. Then we define:

$$Z_0 = C$$

$$Z_i = (Z_{i-1} + X_i)^2 \text{ modulo } N \quad (i = 1, 2, ..., n)$$

$$X_{n+1} = MDC = Z_n \qquad (21)$$

This QCMDC technique has the following properties, and is based on several techniques used in the generation of software hashing techniques and pseudo-random-number generators:

1. Instead of a power-of-two modulus, the intermediate results during the MDC calculation are constrained within a fixed word length by taking the result modulo a very large prime number, N, hence the term congruential. The modulus $N = 2^{31} - 1$ (a Mersenne prime), resulting in k=31, would appear to be

particularly attractive, at least if the resulting probability of erroneously accepting a fraudulent message is acceptable. Otherwise extended precision arithmetic would usually be required to divide the intermediate result of the squaring operation proposed next. **Note that if cryptographically sensitive operations are being authenticated (such as down-line loading of keys), a full k=64 bit MAC or MDC should be used.**

2. In order to destroy the commutative and associative properties of the previous MDC algorithms by introducing a non-linearity into the calculation, the M blocks of plaintext, each m bits long, will each be treated as an unsigned binary integer, added to an accumulator, and then the accumulator squared before the result is taken modulo the prime number. (Values of m = 7, 8, or 16 might be convenient, with m = 16 perhaps the most reasonable for implementation on the new 16-bit microprocessors). This quadratic component introduces a strong positional sensitivity into the MDC calculation, so that "ABC..." can not be confused with "ACB...", nor with "...ABC".

3. To ensure that insertion of data will be detected in the case where the opponent has knowledge of the plaintext, the accumulator must be initialized with the secret value, C. If C were known, a system user could attack his own message and calculate a new MDC as will now be discussed.

In order to ensure that the adversary cannot modify the MDC in the case of Output Feedback mode, it is necessary to insert some secret quantity into the calculation of the MDC. Otherwise, in the case of known plaintext the adversary could easily compute the MDC, Exclusive-OR the computed value with the corresponding ciphertext bits to determine the keystream, R, and then Exclusive-OR in any new, spurious MDC that he wished (assuming that the synchronization between plaintext and ciphertext can be determined - a non-trivial assumption).

In order to avoid this problem, it would at first appear that it would be sufficient merely to prefix the secret seed, S, to the beginning of the message. Encrypting the message would thereby assure the secrecy of S, and generating S by the system would hide it from the user. The MDC, which would then depend upon the secret seed as well as the plaintext, would also be encrypted and hidden from the user. The seed that would be transmitted at the beginning of each message could be truly random, not just pseudo-random. Since it would be sent encrypted at the beginning of every message, checked as part of the MDC generation, and then discarded, it would not even need to be deterministic; and therefore unlike a secret key, it would not have to be transmitted to the communicants before the session is established. This scheme was in fact proposed in the original paper by Jueneman[16]. The receiver in this case would have to be able to distinguish between the first block of a message (the seed), the last block of a message (the MDC), and the rest of the message (the plaintext).

Attacks Against A Secret-Seed Only QCMDC

However, two scenarios have been discovered whereby a user of the system could theoretically attack his own message if this scheme were used. The first is a variation of Coppersmith's under-determined knapsack attack, which proceeds as follows:

The attacker begins by intercepting and discarding (not transmitting) both the first block or blocks of ciphertext (containing the encrypted value of S) and the last block of ciphertext (containing the encrypted secret MDC). Then, using the under-determined knapsack attack, he calculates two strings of ciphertext of 14 and 13 blocks each (denoted S = S_1, S_2, ..., S_{14} and T = T_1, T_2, ..., T_{13}) that can be inserted into the intercepted ciphertext Y_2, Y_3, ..., Y_n (e.g., T and S are inserted before block Y_i), such that the MDC calculated on the plaintext recovered from the modified ciphertext Y_2, ..., S, T, Y_i, ..., Y_{n-1} equals the MDC recovered from Y_n (i.e., X_n, which is known.) The variation from Coppersmith's attack on the modulo 2^{64} MDC technique comes from the fact that the entire plaintext message is required to be known in this case.

Let us assume that a 64 bit MDC is being used. In order for the birthday problem to apply, we select trial values for the inserted strings S and T such that the resulting plaintext can be calculated. This can be done by using the plaintext-ciphertext pairs available in the message to be attacked, or obtained from another message encrypted under the same key. To have a reasonable chance of success, we must generate two sample sets, each of size $2^{64/2}$. For each of these 6.3 billion or so trial permutations of S, we calculate the recovered plaintext and work from the left (using squaring operations per Equation 21) to calculate all of the possible values of the intermediate accumulator that would result after the plaintext recovered from ciphertext block S_{14} (fixed in the problem) has been added to the accumulator and squared (modulo N), saving the values in a table on disk or tape. (Both the 64 bit accumulator and a 32 bit permutation index would have to be stored for each value in the table, amounting to $96*2^{32}$ bits or 51.5 gigabytes, or about 286 reels of standard 2400' high-density, 6250 bpi tape.)

For each of the 6.3 billion or so trial permutations of T, we could make use of the recovered plaintext and work backwards from the right, starting from (X_{n-1}), by taking square roots to calculate all of the possible starting accumulator values associated with with the permutations of T. The two tables of accumulator values (572 reels) would then be compared through a lengthy sort/merge process, to see if any value in the table of values produced from the permutations of S would match any other value in the table of the T permutations. If even a single such match is found between the two sets of accumulator

values (one calculated forward and the other calculated backward), then a set of permutations of S and T has been found which can be inserted without detection; otherwise the procedure continues until such a match is found. (We could be generous and use somewhat more than 13 blocks, in order to decrease the probability of not finding a match, or else be prepared to choose a new set of 13 blocks and run the calculations again in case of failure. The probability of not finding a match decreases rapidly as the number of samples exceeds the square root of 2^{64}, so 14! permutations (87 billion) would be sufficient in almost all cases, but this would involve doing 14 times more work and require 14 times more storage than would be required on the average.).

(Although square roots are being taken in the evaluation of the T permutation values and therefore two possible values are produced at each step, either root is equally valid, and it is not necessary to evaluate the entire set of all possible starting values. Any one starting value that produces a match with the S values is sufficient.)

It should be observed that the seed recovered by the receiver from ciphertext block Y_2 (which he thinks is Y_1) is known to the opponent. Although the opponent is prevented from determining the value of a system generated seed, he can cause the receiver to accept a seed known to the opponent. The spurious seed recovered from ciphertext block Y_2 is just $X_1 \oplus Y_1 \oplus IV$, where each of the individual components X_1, Y_1, and IV are known to the opponent.

It is worth noting that with a 64-bit MDC this attack borders on the very fringes of feasibility due to the storage required. If the size of the accumulator and the final result were increased significantly, say to 80 bits, the storage required would increase 8-fold, to 4576 reels of tape; and increasing the MDC to 128 bits would put the problem completely out of reach by requiring 128 bits for the accumulator value plus 64 bits for the permutation index, times 2^{64} values, or $4.4*10^{20}$ bytes of storage for each table, or a staggering 5 quadrillion reels. On the other hand, if the MDC were reduced in size to only 31 or 32 bits, only 64K permutations would be required for S and T, requiring only $2*(32+16)*2^{16}$ bits or 768,432 bytes of storage, and the entire problem could then be handled easily within the internal memory of a modern microcomputer.

In order to defeat this attack, therefore, we can do one of two things. We can either increase the size of the MDC to 80 or more bits, which would require multiple precision arithmetic (160 bits for the intermediate product) for the calculations, or we can introduce a secret quantity, C, into the accumulator before we start the process. If C is unknown to the user, and say 32 bits long, the attacker has only a probability of 2^{-32} of producing a spurious MDC that will be accepted by the receiver, no matter how much work he does.

Although we might like to avoid introducing an externally communicated secret quantity (because of the additional key management required), and therefore might consider extending the size of the MDC sufficiently to defeat the Coppersmith attack, there is an additional attack that must be considered that is much less complicated but nonetheless highly interesting.

This attack, which will be referred to as the delayed transmission OFB attack, could be effective against QCMDC when using the OFB mode of operation if only a secret seed were transmitted as the first block of the message and the initial value of the accumulator were known, and if once again the entire plaintext were known to the opponent (who is attacking his own message). Also, it is again necessary to assume that the opponent is able to determine the precise bit in the transmitted ciphertext that corresponds to the beginning of the plaintext message, i.e., that he knows the synchronization between plaintext and ciphertext.

The opponent would begin by causing a lengthy known plaintext message to be sent, but would intercept the ciphertext and prevent it from reaching the receiver. (Jamming the transmission would not be sufficient. The opponent must have previously installed an active tap on the line, and then introduced some delay. Then the "good" ciphertext would simply be deleted from the outgoing line, to be replaced with a spurious transmission in real time. Obviously this attack would not be possible with all transmission media.)

Next, the opponent would use the known plaintext to recover the cryptographic keystream, R, that is output from the DES engine, by Exclusive-ORing the plaintext against the ciphertext starting with Y_2, the block following the unknown seed. Then, knowing the cryptographic keystream, the opponent would send a spurious end of frame sequence (assuming a protocol such as SDLC or ADCCP) with an erroneous block check, followed by a valid, properly encrypted flag sequence (produced by Exclusive-ORing the flag bits with the known keystream, R). The original, unknown seed is therefore made to look like a noise burst and would be rejected by the data link protocol, unbeknownst to the end-to-end authentication mechanism that is being assumed here.

Then, again using the known keystream, the opponent would fabricate any desired seed, message, and fraudulent MDC of his choosing (so long as it was shorter than the original message), and would output it in proper synchronization with the original traffic stream, all in real time. This message would be accepted by the receiver as perfectly proper (assuming the attacker somehow determined the proper Nr, Ns frame numbers for the message).

Finally, the tail-end of the original message, containing the valid MDC (and extending to the old valid block check, plus the flag sequence and the start of the next message), would simply be replaced with zeroes or random garbage, simulating a line error. The receiver's data link control protocol would therefore reject those bits, up to the next valid flag and the start of the next message, and the normal

error correction protocol would cause the message(s) following the modified one to be retransmitted by the originator. Since the valid message was replaced with a spurious one with an apparently correct frame number and block check, the spurious message would be accepted, and only the subsequent messages will be retransmitted. But by then everything would be back to normal, and the fraud would pass undetected.

We therefore conclude that it is necessary to ensure that any opponent who can attack his own messages cannot cause the receiver to accept a erroneous MDC. Because of the delayed transmission OFB attack, merely extending the length of the MDC to increase the work required is not sufficient. The use of a secret, externally communicated (or derived) initial value of the accumulator, C, is necessary to defeat these two attacks, in keeping with our previous desiderata.

Having concluded that a secret initial value for the accumulator is necessary, at least if we wish to protect against a user who may attack his own messages, could we then dispense with the secret seed transmitted with each message? The answer depends on whether the initial value for the accumulator is a constant for multiple messages. If it is, then there might be ways (not yet found by the authors) whereby it could be compromised. If not, then the initial value should be perfectly secure since it is only used once, and only shows up indirectly in the calculation of the secret MDC.

So far, at least, the authors have not been able to come up with an attack that would compromise the secret initial value of the accumulator, although there is some concern that such an attack might be possible given a constant C. As a result, we would recommend a temporizing position: If the initial value of the accumulator, C, is constant from message to message, if there is no other source of randomness or time variant quantity in the message (such as a sequence number or date/time stamp), and if context-free authentication is required; or if the additional transmission overhead is of no great concern but the best possible protection is desired (short of using the two-key MAC approach), then **the use of the secret seed S in addition to the secret initial value C is recommended on the basis of general prudence.** It will at least ensure that if an accidental compromise should occur, then two out of the three secret quantities (the C, S, or the MDC) would have to be compromised before the authentication could be subverted, even by working backward attacks. Otherwise, it appears that secret seed may not be entirely necessary.

In order to avoid having to securely transmit the value of C, it would be sufficient to initialize the accumulator with a variant of the key used for the encryption. However, because less security is likely to be associated with the protection of C and the MDC, it is important that this variant be computationally independent of the key itself, lest the key be compromised somehow. For example, the encryption of zero under the current key, $E_K(0)$, would be satisfactory, but the use of K itself, or some non-cryptographic permutation or inversion of the bits of K would not be considered safe. As stated above, S could be purely random, or it could be the encrypted value of a sequence number or date/time stamp (which would appear doubly encrypted in the ciphertext.)

Implementation Considerations

It may be worth mentioning the kinds of environments where the Quadratic Congruential MDC technique might best be applied. Certainly one case would be where standard link encryption equipment has already been installed for the sake of secrecy, and it is now desired to add authentication between two cooperating application programs (or between a host application program and a "smart" terminal such as a personal computer) at the least possible cost. A software modification could be made to the existing programs to calculate the MDC, without requiring a new (and so far unavailable) link encryption unit that would be capable of computing and adding a MAC, and without requiring any hardware modification to the computer(s). The secret value of C could be installed in the two application programs directly, or one program might pick a random number and send it to the other via the encrypted link, and then the second program could return it for verification.

If an end-to-end encryption capability (either hardware or software) were to already exist within the computer, however, it would require detailed timing studies to pick the best method, comparing the use of the QCMDC technique to the use of two encryption passes to produce a MAC. The speed of current DES chips ranges from 28 microseconds to about 6 microseconds to encrypt 64 bits. As an example of the trade-off decisions, the 8087 coprocessor chip available as an option for the IBM Personal Computer can perform a double precision multiply instruction in about 26 microseconds, while the native processor 8088 would require almost 2100 microseconds for the same task. Without the 8087 coprocessor, therefore, two hardware DES passes to calculate the MAC and then encrypt would certainly be the faster technique; but with the 8087 the times might be roughly competitive, especially when the operating system overhead required to access the DES chip is included. On the other hand, if the 8087 chip is available but the DES chip is not, then the QCMDC technique would be significantly faster than computing DES in software for the authentication function.

The case of main-frame computers is similar to that of the coprocessor, except that while the main-frame arithmetic operations are highly optimized, the tedious bit-manipulation operations necessary to implement DES in software may not be concomitantly faster. Although the fastest DES chips would presumably be used, the system software required to access a DES chip would add substantially to the overhead, depending on how many bytes were to be encrypted at one time. Therefore, unless the DES function were implemented as a direct CPU instruction, without requiring any operating system overhead to speak of, the QCMDC technique would probably be faster and more cost effective. At this time it seems unlikely that the QCMDC technique would be used for a hardware implementation, except perhaps using a dedicated 8086/8087 microprocessor combination or a fast signal processing chip capable

of multiple precision operations including a divide.

Although the QCMDC technique appears to be quite strong, in view of the rather surprising number of attacks that have been devised against other MDC and MAC techniques since CRYPTO82 in August of 1982, **the authors would suggest that caution be exercised before using the Quadratic Congruential Manipulation Detection Code (or any other MAC or MDC technique for that matter) until it has received an exacting scrutiny by other professionals in the field.** Only the two independent key method of Message Authentication Code calculation can reasonably be considered to be beyond reproach at this time.

Summary And Conclusions

A list of general protection goals for a secure computing/communications system has been proposed, to set forth the various requirements for message authentication. Although the paper has not dealt with the subject of the protocols, sequence numbers, key management, etc. required to implement such a system, we have discussed the problem of authentication of a general message in and of itself.

Message Authentication Codes (MACs) that are generated with two cryptographic processes involving two different sets of cryptographic keys are currently considered to be the most secure means of message authentication, but they have the various drawbacks of inefficiency, difficulties of key management, and separation of function and interfaces across multiple layers of a communications architecture such as the ISO Open System Interconnection Reference Model. In addition, several attacks were presented which indicate that the use of a single key for both message encipherment and authentication is highly suspect. An interesting parallel to the use of two keys for secrecy and authentication in Public Key Cryptography was also presented.

Manipulation Detection Codes were defined to be a class of functions which do not involve the use of a strong cryptographic function in and of themselves, and thus do not require a secret key. However, they do depend upon the existence of encryption for message secrecy, for the protection of their function of validating the authenticity of the message as received and decrypted.

Architectural considerations were presented which illustrate the desirability of a common approach to message authentication and/or manipulation detection codes which would be universal in nature, and would not have to be changed if the underlying encryption mechanism were to change.

Analysis has shown that a previously proposed Manipulation Detection Code based on the use of a block-wise Exclusive-OR operation (modulo 2 addition) is not sound, and can be easily compromised by transposition and data insertion attacks.

A linear addition modulo 2^k MDC technique was evaluated and was also recommended for rejection, because it provides insufficient protection for use with Output Feedback mode. With OFB certain bit changes either go undetected or are detected only with low probability. The technique is computationally efficient, however, and may therefore be acceptable for use with CBC and CFB if this architectural incompatibility is acceptable, if there is no possibility that the plaintext associated with the intercepted encrypted data is known, and if these limitations are thoroughly understood. The technique has been shown to be vulnerable to the compensating error attack of Miles Smid, the under-determined knapsack attack of Don Coppersmith, and the delayed transmission with modification attack against OFB in the case where the adversary is attacking his own message, and therefore has knowledge of the plaintext.

A Quadratic Congruential Manipulation Detection Code (QCMDC) has been proposed, which was designed to avoid the difficulties of the previous MDC techniques. To provide the required strength in the case of Output Feedback and to defeat the under-determined knapsack attack it was necessary to introduce a secret, random component that is not accessible even to the legitimate user. To achieve the required degree of security, which even protects against the insider who tries to modify his own message (to a format or content not allowed by the system), the secret quantity must be treated in many respects as a secret cryptographic key. As a consequence, the MDC takes on some of the characteristics of a MAC.

In effect, we were not successful in generating the redundancy required for authentication using only nonsecret quantities. In fact, the under-determined knapsack attack of Coppersmith and the delayed transmission OFB attack both indicate that the search for an MDC technique that is both independent of the encryption method and strictly a complicated function of the plaintext itself is almost surely doomed to fail in the case of known plaintext.

On the other hand, the QCMDC technique appears to be adequately strong, and offers possible speed and implementation advantages. It is an interesting example of the substitution of one strong and one weak cryptographic procedure for the two strong procedures required by the two-key MAC approach. The QCMDC technique may be particularly useful because it can be implemented in application software, either in general purpose computers or microprocessors, independent of the encryption method and the operating system.

Although the QCMDC technique has so far withstood a number of novel and powerful attack scenarios, prudence suggests that still more extensive analysis would be desirable, both from the standpoint of algorithm behavior (a la Knuth) and cryptographically, before it can be fully accepted. In the meantime it is recommended that the full two-key approach to Message Authentication Code generation be utilized for any critical applications, at least until a sound, single key end-to-end encryption scheme that provides garble extension during decryption can be devised.

Finally, the "birthday problem", or under-determined knapsack attack has been shown to be particularly powerful. Because the attack reduces the size of the search effort to the square root of the time required for a serial exhaustive search, i.e., $2^{k/2}$, this fact must be kept in mind when choosing the size of the MAC or MDC. If an erroneous guess on the part of the attacker would result in a message or command being rejected, with the alarms being sounded and appropriate action taken to find the intruder, then a 31 or 32 bit MAC or MDC should be sufficient. **But if it is not possible to take effective action to cause the source of the spoofing to cease, or if cryptographically sensitive operations such as down-line key loading are being authenticated, then a full 64-bit MAC or MDC is recommended.**

Acknowledgements:

This paper would not have been possible without the contributions and insight of many other people. The authors would particularly like to acknowledge their many fruitful conversations and correspondence with Dr. Dennis Branstad and Miles Smid of the National Bureau of Standards, Jonathan Oseas of the IBM Crypto Competency Center, Dr. Donald Davies of the National Physical Laboratories in Great Britain, Dr. Gustavus Simmons and Judy Moore of Sandia National Labs, Dr. Selim Akl of Queen's University, Kingston, Ontario; and Drs. G. R. Blakley and Dorothy Denning, the IEEE Symposium Program Co-Chairpersons, for their support and encouragement. Particular credit must go to Dr. Don Coppersmith of IBM Research, who analyzed several approaches and provided us with his under-determined knapsack attack - a powerful tool for analyzing authentication schemes. Robert Jueneman would also like to recognize the support provided for part of this effort by Satellite Business Systems, his former employer.

References

1. Jaeschke, G., "Reciprocal hashing: a method for generating minimal perfect hashing functions." **Comm. ACM, 24,** 12 (Dec. 1981), 829-833.

2. Federal Information Processing Standards Publication FIPS PUB 81, "DES Modes Of Operation," National Bureau of Standards, 2 December 1980.

3. Meyer, C. H., and Matyas, S. M., **Cryptography: a new dimension in computer data security,** John Wiley & Sons, Inc. New York, NY, 1982, pp 536-537.

4. Coppersmith, D., Private communication to C. H. Meyer and S. M. Matyas, May 2, 1983.

5. Branstad, D., National Bureau of Standards. Private communication to R. Jueneman, March 1983.

6. Proposed Federal Standard 1026, "Telecommunications: Interoperability and security requirements for the use of Data Encryption Standard in the physical and data link layers of data communications." National Communications System, Washington, D.C. Draft of June 1, 1981.

7. Meyer and Matyas, op. cit., pp 361-363.

8. Jueneman, R. R., "Analysis of certain aspects of Output Feedback Mode," **Advances in Cryptology - The Proceedings of CRYPTO82,** Plenum Publishing Corp, New York, N.Y., 1983, pp 99-127.

9. Federal Standard 1026, "Telecommunications: Interoperability and security requirements for the use of Data Encryption Standard in the physical layer of data communications." National Comm. System, Office of Technology and Standards, Wash., D.C. 20305. Pre-publication copy circa Feb. 1983.

10. Davies, D. W. and Parkin, G. I. P., "The average cycle size of the key stream in Output Feedback encipherment." Extended abstract in **Advances in Cryptology - The Proceedings of CRYPTO82,** Plenum Publishing Corp., New York, NY, 1983, pp 97-98.

11. Jueneman, R. R., op cit., p. 107.

12. Federal Standard 1027. "General Security Requirements for Equipment Using the Data Encryption Standard." Issued by General Services Administration. April 14, 1982.

13. Meijer, H. and Akl, S., "Remarks on a Digital Signature Scheme," **Cryptologia,** Vol. 7, No. 2 (April 1983), pp 183-186.

14. Feller, W., **An Introduction of Probability Theory,** Wiley, New York, 1950, Section 2.3.

15. Meyer and Matyas, op cit., pp 429-606.

16. Jueneman, R. R., op cit., p 106.