Essay 15

Cryptography

Marshall D. Abrams and Harold J. Podell

This essay discusses cryptographic protection of information confidentiality and integrity as that information passes from one point in space-time to another. More recent uses of cryptography, such as authentication and nonrepudiation are also discussed.

The essay begins with an introduction of these ideas, including some basic examples, then proceeds to the definition of a cryptographic system, making the distinction between conventional key or symmetric key schemes and public key or asymmetric key schemes. We present some classical examples beginning with Julius Caesar. Both substitution and permutation ciphers are included, as well as a word about their weaknesses. The Data Encryption Standard (DES) serves as an example of a product cipher whose strength derives simply from repeated applications of both permutations and substitutions.

The essay then turns to public key schemes or systems. A public key system can be used by anyone to encrypt a message for a given recipient but only that recipient can decrypt it. Although there are many proposed in the open literature and three have been widely implemented, we focus on the most popular system, RSA. RSA (Rivest, Shamir, and Adleman) is a widely used public key system whose strength lies in the difficulty of factoring certain large numbers.

A discussion of public key management is followed by an introduction to public key and conventional key management issues. We also discuss authentication and integrity issues that are associated with conventional key systems. In addition, link and end-to-end encryption are described and contrasted. The essay's final topic is the integration of computer and communications security.

What is encryption?

Encryption is a fundamental tool for the protection of sensitive information. Its historical purpose is privacy (preventing disclosure or confidentiality in communications. Encryption is a way of talking to someone while other people are listening, but such that the other people cannot understand what you are saying. It can also be used to protect data in storage as well as to detect active attacks, such as message or file modification.

We refer to encryption as a tool because it is a means for achieving an end; it is not an end in itself. Cryptography, hidden writing, is a method for transforming the representation (appearance) of information without changing its information content. Plaintext (cleartext) is one representation of the information expressed in natural language, intelligible to all. Ciphertext is a different representation, designed to conceal the information from unauthorized persons. Encryption (or encipherment) is the transformation from cleartext to ciphertext. Decryption (or decipherment) is the reverse transformation.

History. Since the time of Julius Caesar and even before, people have protected the privacy of their communications by cryptography. Things are still that way, and yet everything is quite different. People continue to use cryptography, though far more sophisticated than Caesar's, to protect their vital information as it passes through possibly hostile environments. Rather than crossing a few hills on its way to Rome, their data is moving from one point in the space-time continuum to another. Messages and documents created at one place are delivered at a later time at some distant place. When transmission of messages and documents is by electronic means, delivery is at essentially the same time but at a different place. A file created on a computer can be recovered at the same place but at a later time or, if it is copied onto a diskette, at some other place and at some later time.

Historically, cryptography has been used chiefly in communications. Its application in data retrieval is a far more recent occurrence. We shall tend to use the language of communications in describing cryptographic mechanisms, but the reader should keep the other examples in mind as well. The physical security and/or the access control mechanisms, whether they are on communications links, on network nodes and switches, on mainframes, file servers, and PCs, or on diskettes in transit, may not be sufficient to assure the confidentiality and the integrity of the data that passes through them. Cryptographic mechanisms are available that go far in establishing assurance in all these environments.

The word *cryptography* and the associated word *cryptology* have very similar etymological origins. They are derived from the Greek words *kriptos*, which means "hidden"; *graphos*, which translates to "writing"; and

logos, which is "word" or "speech." In current usage, however, they have slightly different meanings. Cryptography is the science of hiding information. Encryption, sometimes called encipherment, is the act of concealing the meaning of a message. Decryption or decipherment is the inverse process of returning it to its original form. Any other, unauthorized method of recovering the original message is known as cryptanalysis or "breaking" the message. Cryptanalysis is the combination of science, art, and luck used to break messages or entire systems. The word cryptology nowadays refers to the study of both cryptography and cryptanalysis. When designing a strong cryptographic system, it is necessary to consider all possible attacks. In this essay, however, we discuss cryptography only. We include only such references to cryptanalysis that aid the reader in better understanding the strength of a particular cryptosystem.

Acknowledgments. In developing the perspectives for the history, types of attacks, encryption function standardization, and related topics for this essay, review assistance was provided by Shimshon Berkovits and H. William Neugent. Their comments and insights have been useful in balancing the presentation of cryptographic issues. If there are any omissions or misinterpretations in this essay, they are the authors' responsibility.

What is a cryptosystem?

A historical example. As a starting point for our description of what is cryptography, let us return to Julius Caesar. His scheme can encrypt any sequence of characters from the Roman or any other alphabet. His technique requires rotating the alphabet three positions to the right. Thus, each letter of the message is replaced by the one that occurs three places later in the alphabet. To decrypt, rotate the alphabet three positions to the left; that is, replace every letter in the encrypted message by the one that occurs three places to its left in the alphabet.

This is the basis for a class of ciphers known as Caesar ciphers. There is no great significance attached to the number three. Rotate the alphabet right k places to encrypt and k places left to decrypt. It is only necessary that both the sender and the receiver know the value of k. The k is called the key. For the single pair of encryption and decryption algorithms used in Caesar ciphers, different values of the key k will have different effects. The key can be changed once a month, once a day, or even for each message. There are even cipher systems that change the value of k for each character in the message. The sequence values for k can be randomly chosen, in which case the entire sequence is the key. The sequence can be generated by a pseudorandom number generator. If we incorporate the generator into the encryption and decryption algorithms,

the generator's seed value becomes the key. Alternatively, the sequence can be derived from some preselected text, such as the jth line of the ith page of this book or the jth line of the ith column of today's *New York Times*. In this case, the key is the pair i, j and the name of the document to which they refer.

Any system of substituting an element of some ciphertext alphabet for each character in the plaintext alphabet yields an encryption algorithm. The key is the actual correspondence between the characters of the ciphertext alphabet and those of the plaintext alphabet. Actually, there is a slight difference between the encryption and decryption keys. The encryption key tells what cipher character to use in place of each plaintext character, much like an English-French dictionary indicates what French word to use in place of each English word. The decryption key indicates which plaintext character replaces each cipher character. That corresponds to a French-English dictionary. These two keys are not the same, but it is not difficult to derive one from the other.

These cipher systems are collectively called substitution ciphers. Given a long enough random key sequence or a pseudorandom number generator with a long enough cycle before it repeats its output sequence, such systems can encrypt long streams of plaintext characters. When used that way, they are examples of stream ciphers; they treat the plaintext as simply a long stream of characters.

Block ciphers have a different characteristic. Block ciphers subdivide the plaintext message into blocks of some fixed size. Each block is then encrypted as a whole. The simplest and oldest example of a block cipher is a permutation cipher. It shuffles the characters in a block. In fact, it shuffles each block in exactly the same way. One way is to break the plaintext into blocks of size $m \times n$. Write each block in m rows of n characters. Now read the characters by columns in some preselected order. To decrypt, write the ciphertext characters in columns in the same order and read the plaintext row by row. The key, which must be known to both sender and receiver, consists of the numbers m and n and the sequence of the columns. For the general permutation cipher, the encryption key is the size of the block and the permutation. The decryption key is the size of the block and the inverse permutation.

Product ciphers. Some very powerful encryption algorithms called product ciphers have been produced by using combinations of substitutions and permutations. In his information theory approach to cryptography, Claude Shannon spoke of two concepts for hiding information: "confusion" and "diffusion." Substitutions create confusion and permutations introduce diffusion.

For example, the Russian spy master Rudolf Abel used a cipher that followed a substitution cipher with two permutations. The cipher replaced the most frequently used letters of the Russian alphabet by single

digits and all others by pairs of digits. It was done in such a way that there was no ambiguity on decryption how to divide the sequence of digits into single and double digit letters. The sequence of digits produced by the substitution was shuffled using a rectangular-array permutation cipher, as we have described.

The result was modified again by another rectangular-array permutation cipher. The dimensions of the second array were different from the first. The second cipher also featured triangular perturbations of the array. A letter written in this cipher was instrumental in the conviction of Abel. However, the cipher itself was so strong that it was never broken. Its workings were described to the authorities by Abel's assistant Reino Hayhanen when he defected.

The Data Encryption Standard (DES), about which we speak further on, is another example of a product cipher. We generally include product ciphers in a category referred to as conventional or symmetric key cryptography, because the sender and receiver share the same secret key.

A formal definition. Encryption functions take at least two inputs. The first is the plaintext, and the other is an encryption key that is sometimes referred to as keying material. It is useful to think of the algorithm as the way the tumbler action in a lock seals access to the information. The data is protected when the safe is locked by someone holding the key. In reality, the encryption key is information that affects the functioning of a given encryption transformation or algorithm, just as different tumbler settings affect the action of a single lock.

Similarly, decryption has two inputs also. They are the ciphertext and a decryption key. Again, think of the decryption algorithm as the way the tumblers work to open a physical lock. That lock cannot be opened without the key that corresponds to the tumbler settings. That key must correspond in some way to the encryption key. In fact, we are used to having a single key to lock and to unlock a door. But, when talking of cryptosystems, there can be a subtle difference.

Let us look at the Caesar cipher one more time. Let encryption be described as rotation of the alphabet E_k steps to the right, where E_k is the encryption key. Decryption can be described in one of two distinct (but related) ways. With decryption stated as rotation of D_k steps to the left, then $D_k = E_k$. But with decryption defined in the same way as encryption (and there is some benefit in having both algorithms the same), $D_k = -E_k$. The decryption key, while obviously tied to the encryption key, is nonetheless not identical to it. This possibility that the two associated keys are different leads to some interesting cryptosystems, as we shall see.

A generalized representation of the encryption and decryption processes is illustrated in Figure 1.

Let

A = Alice or the sender
B = Bob or the receiver
M = Plaintext message or message
C = Ciphertext
E_k = Encryption key
D_k = Decryption key
E = Encryption function or transformation
D = Decryption function or transformation

Then

$$C = E(E_k, M)$$

One way of reading this notation is as follows: The ciphertext (C) is produced by operating on the plaintext (M) with an encryption algorithm (E), using the encryption key (E_k). This notation is a variation of algebraic notation, where the parentheses indicate the operational relationships. For example, $C = E(E_k, M)$ uses parentheses to show that the encryption algorithm (E) is operating on the plaintext message (M) with a specific key (E_k).

For the cryptosystem to be of practical use, we must have

$$M = D(D_k, C) = D(D_k, E(E_k, M))$$

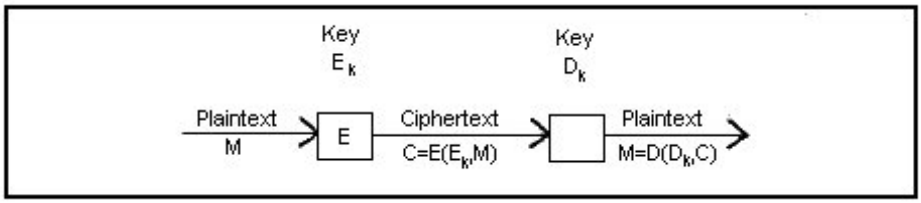

Figure 1. Generalized representation of encryption and decryption processes.

Conventional and public key systems. If it is easy to compute the decryption key D_k from the encryption key E_k, as is the case in all classical substitution and permutation examples, then both keys must be protected. Anyone who has access to either key can unlock the information protected by them. As introduced in our discussion of product ciphers, these cryptosystems are called symmetric key or conventional systems.

Unintuitive as it may seem at first reading, there are schemes in which it is computationally infeasible to derive the decryption key from the encryption key. Such cryptosystems are called asymmetric, and we present

the most popular example, RSA (Rivest, Shamir, and Adleman), later in this essay. Asymmetric systems have a useful property. One can make the encryption key (E_k) public without fear of disclosing the decryption key (D_k). Then anyone can encrypt a message, but only the single holder of the decryption key can decrypt it. For this reason, asymmetric cryptosystems are also called public key systems. The published key is known as the public key, while the other is the private key. For certain public key digital signature systems, encryption and decryption are inverse functions. For these systems, it makes no difference which is performed first. However, this symmetry does not apply to other public key digital signature systems such as ElGamal, the associated Schnorr algorithm, and the proposed US Federal Digital Signature Standard (DSS). We use the notation of D_A for Alice's private key and E_A for her public key.

If the encryption and decryption functions of a public key cryptosystem commute, that is M (message) = $E(E_A,\ D(D_A,\ M))$, even though decrypting first seems to make no sense, we have another useful characteristic. Alice, who is the holder of her private key (D_A), can send information that is modified by applying her decryption algorithm using her private key. If the recipient, Bob, knows her corresponding public key (E_A), applying the encryption function to the modified information will give him assurance of the identity of Alice. In essence, Alice has "signed" the information by first using her secret or private key, which she alone possesses. This is an example of a digital signature, of which we speak again below. Figure 2 illustrates the process.

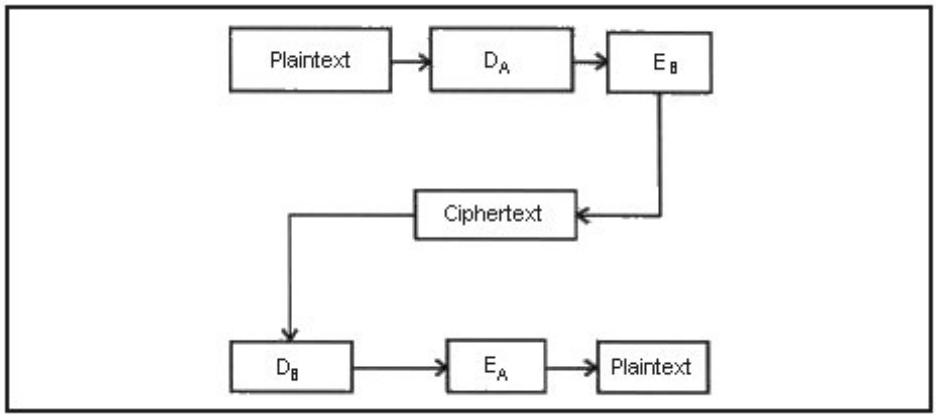

Figure 2. Public key cryptosystem.

The example in Figure 2 shows the plaintext or message (M) being signed by Alice with her secret or private key (D_A). After the plaintext is signed, it is now encrypted or "sealed" with Bob's public key (E_B). Only Bob can "open" or decrypt the ciphertext because he is the only entity in

the network to possess the secret or private key (D_B) that corresponds to his public key used to "seal" the message (E_B). Once he has decrypted the message, he or anyone else possessing Alice's public key can verify her digital signature.

Encryption function confidentiality. The functions *E* (encryption) and *D* (decryption) may be kept secret or published, even as standards. The choice involves questions of work factor, open or closed network architecture, and user community. The cryptanalyst has a harder job in breaking the system if the functions *E* (encryption) and *D* (decryption) are kept secret. This is the approach taken for protecting national security related data. However, even in this highly sensitive arena, it is not the reliance on the confidentiality of the functions that protects the information. After all, there are too many known cases in which such information has been leaked or sold to "the enemy." It is the confidentiality of the decryption keys and the fact that they are changed on a regular basis that are the ultimate protection of the information.

Maintaining *E* (encryption) and *D* (decryption) as secret involves procedural and physical protection. If an intruder acquires a cryptographic device, secrets may be broken by reverse engineering. Physical protection can include denial of access to the cryptographic device and automatic destruction of the keys if unauthorized access is attempted. Advances in very large scale integration (VLSI) make it possible to implement the cryptographic function on a single chip that is highly resistant to reverse engineering, even to the extent of self-destruction or zeroization of the keys. Such chips can be put into service with considerably less physical protection than prior technology. In the final analysis, however, reverse engineering does not help recover the keys. If, at a minimum, the key registers zeroize on an intrusion attempt, the information they protect is still safe.

Types of attacks

Attacks and protection. Passive attacks consist of observation of information passing on a connection or residing in a file; release of message or file content is the fundamental compromise. Active attacks include modification, delay, reordering, duplication, and synthesis. Active attacks, resulting in message-stream modification (MSM) or file modification, offer three threats:

- Authenticity attack. Doubt of source and delivery to intended destination of a message; doubt of origin of the file or message.
- Integrity attack. Modifies information content.

- Ordering attack. Changes sequence of information arrival at destination; changes order of records in file.

Communication protocols and computer operating systems generally offer minimal protection against these threats, unless they are specifically to support secure communications. Masquerading, or spurious initiation, is an attack in which an intruder attempts to establish a communications session by falsifying his or her identity. Encryption is the fundamental tool for countering these attacks. Release of message or file content and traffic analysis can be prevented; MSM (message-stream modification), file modification, and masquerading can be detected.

There are several types of attacks that can be mounted against any cryptosystem. Some attacks attempt to recover the plaintext that corresponds to some stolen ciphertext or to discover the key in which one or more cryptograms are enciphered. Others seek to exploit weaknesses in the system so that plaintexts or keys can readily be recovered no matter what keys are used and how frequently they are changed. A cryptosystem designer must be wary of all these attacks.

Ciphertext only. The most difficult form of attack against a system is the ciphertext-only attack, which requires that someone has captured a segment of ciphertext. With no other information, except possibly a guess at the cryptogram's context, he or she attempts to determine the corresponding plaintext and, if possible, the key that was used. Many classical cryptosystems are vulnerable to ciphertext-only attacks which, given sufficient ciphertext, can be examined for evidence of the statistical properties inherent in the underlying plaintext language. Abel's product cipher, which combines one substitution and two permutations, successfully prevents these statistical characteristics from filtering through to the encrypted message.

It is always possible to begin a naive ciphertext-only attack. An intruder can expect that, if he or she begins exhaustively trying every possible decryption key from the space of all such keys, he or she will eventually try the correct one. Of course, if the key space is sufficiently large, that eventuality may not occur before the encrypted information no longer has any value, before the intruder losses interest, or before he or she dies of old age. Furthermore, there may be several different plaintexts that encrypt under different keys to the stolen ciphertext. The intruder has no way of knowing if an apparent decryption to some message that makes sense in the given context is, in fact, the correct decryption.

Alternatively, if the correct plaintext has no recognizable properties, the intruder cannot differentiate it from all the other trial decryptions he or she obtains. This situation occurs when the correct plaintext does not consist predominantly of real words or even of printable characters, but

appears to be a random bit string such as the middle of a compressed ASCII file or some other cryptographic key.

These observations lead to several fundamental principles of system design. First and foremost, the key space must be large. The easier it is to recognize a correct decryption among all other possible decryptions the larger the key space should be. In the best of all worlds, the key space is so huge that an intruder would not even consider this attack, and the plaintext is so random that he or she could not recognize successful decryption if he or she stumbled onto it.

A known plaintext-ciphertext pair. Sometimes, through a lucky guess or other good fortune, an intruder has the plaintext that corresponds to a segment of ciphertext. He or she then tries to discover what key was used in the hope that other data is encrypted in the same key. This situation is similar to the 1799 discovery in Rosetta, Egypt, of the Rosetta stone. This basalt tablet has an inscription in Greek, Egyptian hieroglyphic, and Demotic. The stone provided known plaintext-ciphertext pairs that led to the decipherment of hieroglyphics.

A chosen plaintext-ciphertext pair. If an intruder can somehow obtain the ciphertext associated with one or more plaintexts possessing some special characteristics or the plaintext corresponding to ciphertexts with certain specific patterns, his or her chances of discovering the key may be enhanced. Consequently, the chosen plaintext-ciphertext attack has the potential to be more dangerous than either the ciphertext-only or the known plaintext-ciphertext attack.

Encryption function standardization

Interoperability, the ability for independently manufactured systems and subsystems to work together, is a major driving force for standardization. Market share competition is another driving force. Encryption standards can be used to protect information from intruders, yet permit mutually suspicious parties, such as competitive banks engaged in electronic funds transfer, to work with each other. The Data Encryption Standard (DES) is a well-know symmetric key encryption standard. The CCITT X.509 Secure Directory Service is a standard that includes the use of public key cryptography for certificates, which use a digital signature process. DES can be used for an encryption algorithm to provide for confidentiality in conjunction with a system based on CCITT X.509. For example, the Internet Privacy Enhanced Mail (PEM), which is discussed in Essay 17, uses two algorithms, RSA and DES, and a variation of CCITT X.509.

CCITT X.509 is one of several CCITT standards that pertain to secure international networking. For example, CCITT X.400 pertains to Message

Handling Services and does not assume the directory service. CCITT X.500 defines the use of certificates for Directory Service, and X.509 defines Secure Directory Service.

In addition to the standardization of encryption functions, there are international requirements for the registration of cryptographic algorithms. For example, the organizations that use nonpublic algorithms for secret messages may wish to identify these algorithms by neutral identifiers. Certain evolving protocols could be used to support this type of communication need. The Secure Protocol (SP) 4 at the Transport Layer is such a protocol, and it is being considered by ISO (International Organization for Standardization). ISO is also working to facilitate the registration of cryptographic algorithms.

The data encryption standard

Background. The United States National Institute of Standards and Technology (NIST, formerly National Bureau of Standards) established the Data Encryption Standard (DES) in 1977 as the federal standard encryption algorithm, following a public solicitation for suggested algorithms. The Data Encryption Algorithm (DEA), the algorithm in DES, was derived from a design submitted by IBM. DES is an example of conventional cryptography, because the sender (Alice) and the receiver (Bob) share the same secret key. The standards are:

- Federal Information Processing Standards *(FIPS)*. Data Encryption Standard (DES), Publication (FIPS PUB) 46-1 (recertified until 1992, under review for recertification for another five years) and ISO standard IS 8372.
- *American National Standards Institute (ANSI)*. Data Encryption Algorithm (DEA), X3.92-1981, and Model of Operation of the DEA, X3.106-1983.

DES is designated for non-national-security applications such as electronic funds transfer (EFT). In the late 1970s, several cryptographic authorities commented that DES may become inadequate in 10 years. However, DES has been reaffirmed over the years by NIST.

Recent information, however, has added new knowledge to the DES story. For example, the *New York Times* reported that DES is much stronger than people had thought. Adi Shamir and E. Biham had found an attack on DES that was initially reported as breaking DES, but that actually is only a "slight improvement over laboriously trying every key." Shamir said that DES is "the strongest possible code of its kind." He said that his attack method "devastates similar codes," while only slightly denting DES.

DES technology. The Data Encryption Standard (DES) is formed as a product of substitutions and permutations. It is a block cipher using a 64-bit block. The key consists of a 56-bit block, padded by eight parity bits, one for each byte. DES encryption begins with a 64-bit permutation. It ends with the inverse of that permutation. In between are 16 rounds of confusion and diffusion. The message block is split into two 32-bit halves. The old right half becomes the new left half. The right half is also replaced by using a number of small substitution ciphers. First, it is combined with 32 bits selected from the key and permuted. Then each group of four bits is replaced by a different four bits. For each group of four, there are four different substitution ciphers to be used. The choice for each group is determined by the first and last bit of the group, each combined with a different specified bit of the key. Decryption begins with the same initial permutation and ends with its inverse. In between, decryption goes through exactly the same rounds as encryption, with only one minor modification. The key bits are used in the reverse order.

Although each step in the Data Encryption Standard is a simple substitution, permutation, or exclusive OR operation, the total result is so complicated that an attempt to express a single ciphertext bit as a logical combination of the 64-bit block of plaintext and 56-bit key resulted in a computer printout that was several inches thick.

The strength of DES. The DES algorithm is well publicized and has withstood intensive attempts of many people the world over, who have tried and are trying to break it. Even though none of these efforts has yet succeeded, considerable insight into the inner workings of this and similar algorithms has been developed. At the time of writing, NIST has reaffirmed DES in hardware and certified software implementation of DES.

From the beginning, a major criticism of the DES has been the fact that each key has only 56 bits. That makes a key space of only 2^{56} or about 7.2×10^{16} different keys. The first attack ever suggested against DES was an exhaustive, known plaintext-ciphertext search. It exploited the size of the key space as well as the relation between EXCLUSIVE OR and bitwise complementation. Through a clever trade of time and memory, it searched for the key that encrypted a stolen DES cryptogram. At the time, it was estimated that within 10 years a special-purpose device could be built to do all the needed encryptions in a reasonable time and at a reasonable cost. No such machine has been announced, but it becomes more feasible with every improvement of microchip efficiency and price.

As mentioned, recent attacks on DES by Biham and Shamir [BIHA90] have shed new light on the inherent strength of DES. Their analysis is a variation on the chosen plaintext theme. Their approach, which they call Differential Cryptanalysis, collects many different plaintexts and their ciphertexts. It catalogs differences in the plaintexts and collects statistics

on the differences in the corresponding ciphertexts. Then, given a known plaintext-ciphertext pair, they find the most likely key used in encrypting that pair. The known plaintext-ciphertext pair is conceptually similar to the pairs on the Rosetta stone. They have gradually developed their attack that now it threatens a full 16-round DES. However, the time currently required to complete a successful attack is, at this writing, no better than exhaustive search.

The volume of data that must first be collected and the time needed to complete an actual attack against a single DES key do not yet seem to justify the death knell that has been sounded for the standard in recent newspaper articles. Currently, Differential Cryptanalysis is an attack only against Electronic Code Book, the simplest mode of use included in the standard. It is possible that similar approaches are possible and will be developed for the other three modes (described below). It is also likely that the authors of this attack will push its development further in the hope of making it a meaningful threat.

Modes of operation

The Data Encryption Standard includes a set of standard modes of operation [NIST80]. These and one or two others are appropriate for use with any block cipher — that is, with any encryption algorithm that acts on a fixed-size plaintext block. Each mode possesses different characteristics that are important in different situations. We describe them briefly.

Electronic Code Book (ECB). The Electronic Code Book mode involves simple block encryption of a message or a file. The process is illustrated in Figure 3. The data is broken into blocks of a standard size. Each block in turn is the input to the encryption algorithm. The output blocks comprise the ciphertext message or file. For decryption, that message or file is again divided into blocks, and each block is decrypted individually. The resulting output blocks are concatenated to reconstruct the plaintext message or file. If an error occurs in a single ciphertext block, the decryption of only that block will be corrupted.

Electronic Code Book has a major disadvantage. A single block that appears several times in the plaintext stream will be encrypted in the same way each time. Suppose the plaintext is a file of sensitive information in a database system. If each field in some record forms a single block, an intruder can browse cryptographically. While he or she may not know what the individual entries in each field are, he or she can identify which records have the same value in any specific field. Any side information about the meaning of a single encrypted field entry may give him or her similar knowledge about many other similarly encrypted entries.

The remaining modes of operation use the context of each plaintext block to modify how it is encrypted. Therefore, ECB is generally used for low-volume operations, such as encrypting master keys for transmission.

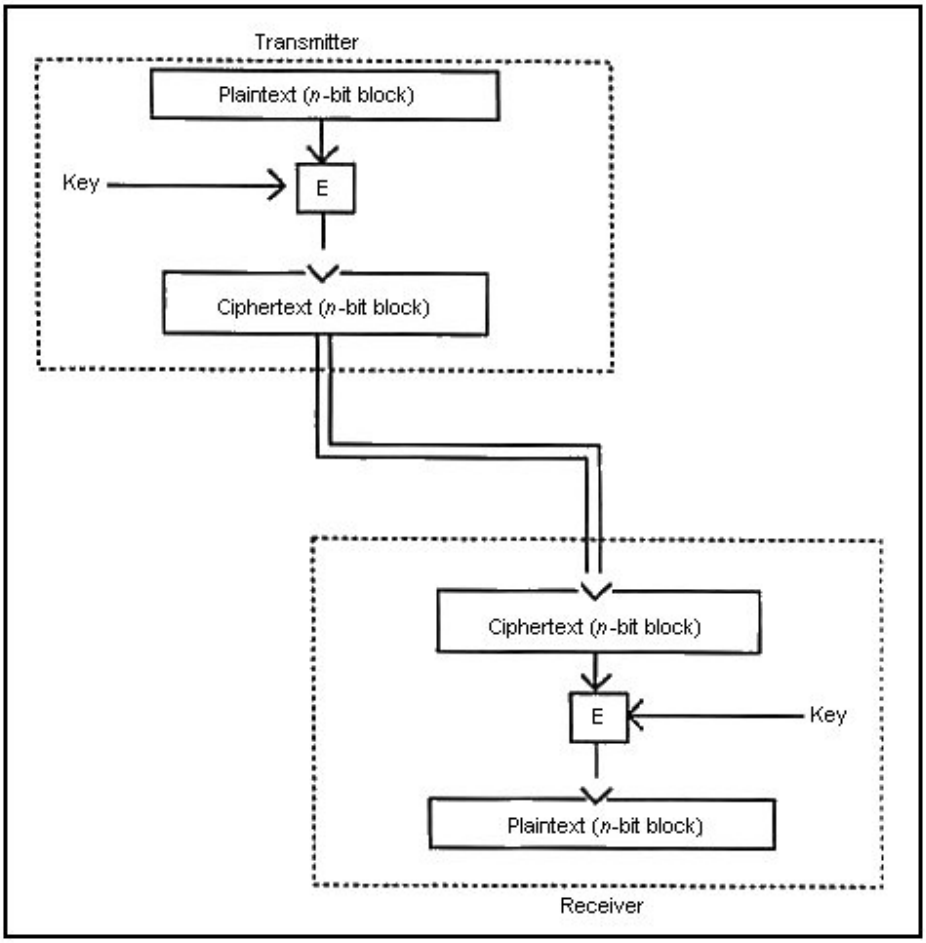

Figure 3. Electronic Code Book mode of DES.

Cipher Block Chaining (CBC). Cipher Block Chaining (CBC) is one way to change the encryption of plaintext blocks that repeat. CBC involves the EXCLUSIVE OR (XOR) of every plaintext block with the preceding ciphertext block. The first plaintext block must be treated differently. It is XORed with a publicly known initialization vector (IV) or with a secret initialization vector that is distributed with the key. For each block, the result of the XOR is the input to the encryption algorithm. The output of

that algorithm becomes the next block in the ciphertext message or file. It is also XORed with the next plaintext block before that block is input to the algorithm. Figure 4 shows the process.

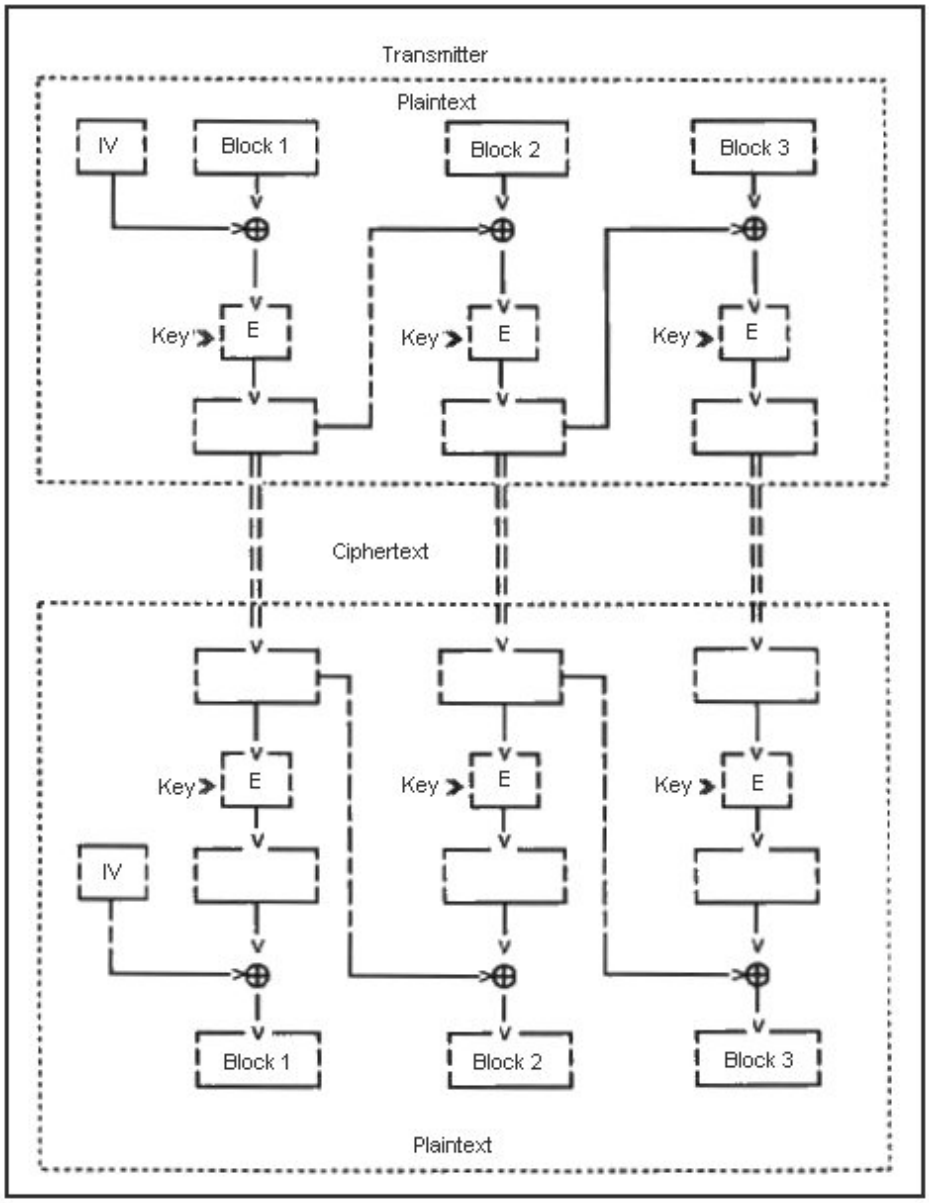

Figure 4. Cipher Block Chaining mode of DES.

The first ciphertext block is passed through the decryption algorithm, and the output is XORed with the initialization vector. The result is the first plaintext block. Thereafter, each ciphertext block is passed through the decryption algorithm. The output block is XORed with the preceding ciphertext block. The result is the next plaintext block. If a single ciphertext block contains an error, neither the corresponding plaintext block nor the next one will be recovered correctly. However, even in the face of errors, as soon as two ciphertext blocks are error free, the decryption is again successful. Such a scheme is called self-synchronizing.

It is apparent that Cipher Block Chaining does solve the cryptographic browsing problem. Records with the same plaintext value in a particular field will not be identifiable because each value will be encrypted using the presumably different ciphertext in the preceding field. Anyone with authorized read or write access to that field in those records can still decrypt correctly. He or she needs only the encrypted value in the preceding field. However, if he or she changes the value in that field, the entire file must be re-encrypted from that point on.

Cipher Block Chaining has another attraction. It can be readily used to create a message or file digest. Encrypt the data using this mode and save only the last ciphertext block as the digest. Then append the digest to the message or file. Anyone who reads it can, if he or she knows the correct key, recompute the digest. If it matches the one that came with the unencrypted message or file, he or she knows that, with very high probability, the data was not changed. He or she also knows that the message or file originated with the only other person who holds the same key. Thus, he or she has both message authentication and origin authentication. We shall see other cryptographic techniques that yield similar assurances.

Output and cipher feedback modes (OFB and CFB). Output and cipher feedback modes (OFB and CFB) can be illustrated by the US Department of Defense (DoD) Key Auto-Key or KAK and Ciphertext Auto-Key or CTAK, respectively. Before introducing these examples, we introduce applicable issues pertaining to stream ciphers, length of keys, and initialization vectors (IVs).

Stream ciphers all have the property that they attempt to integrate context into the encryption. The key is a long bit stream that is to be combined, through an XOR or some other operation, with the plaintext stream. What position a particular data segment takes in the plaintext stream determines with which segment of the key stream it will be combined. If the data segment repeats itself, its different occurrences will most likely be encrypted with different key stream segments. They will be encrypted differently.

The key can be either a long, completely random bit stream that must be delivered to both the encrypting and the decrypting stations, or a

pseudorandom bit stream that is generated as needed. In the latter case, the authors prefer to reserve the word "key" for the pseudorandom seed and to refer to the pseudorandom bit stream that is generated as the "key stream." It should be noted that, if a pseudorandom generator is used, it must be cryptographically strong. That means it must not be possible predict the rest of the key stream even if some keys are discovered or guessed, as it might occur in a known plaintext-ciphertext attack.

One way to create cryptographically strong, pseudorandom bit streams is to use a block cipher like DES. Some fixed number of bits from the output block are added to the key stream on each iteration. The input to the block encryption algorithm is a shift register or a counter. In the latter case, the register is loaded with an initial value and incremented once after each encryption. The decryptor must start his or her counter at the same initial value. As long as he or she stays in synchronization, he or she will decrypt correctly.

If the input is a shift register, it must be loaded with an initialization vector. After each iteration, the register contents are shifted the same number of bits that are added to the key stream from the block encryptor output. The same number of bits are shifted in to fill the empty space in the register. They can be the same bits taken from block encryptor output. In that case, we have Output Feed Back (OFB) mode, which is used in the US DoD Key Auto-Key (KAK). Alternatively, they can be the last bits encrypted. This is Cipher Feed Back (CFB) mode, which is known in DoD as Ciphertext Auto-Key (CTAK). At the decryptor, exactly the same procedure is followed with the block algorithm used to encrypt. Now the key stream is combined with the ciphertext stream to recover the plaintext stream, and the ciphertext bits must be saved for use as feedback.

The reader is encouraged to consider what happens if errors occur in the ciphertext stream. Both OFB and CFB are self-synchronizing. There are, however, situations in which this property is undesirable. If it is most important to flag where a ciphertext stream has been tampered with, it is better to feed back plaintext bits. Then errors introduced into the ciphertext stream cause errors in the plaintext, which are shifted into the input register of the block encryptor. This causes an erroneous output which, in turn, yields an incorrect decryption. The wrong plaintext bits are again fed into the shift register and all decryption is incorrect from the point of the ciphertext error on. This is a very strong indication that the ciphertext has been modified, either accidentally or maliciously.

Perfect confidentiality

Perfect confidentiality can be achieved with a completely random key stream. For such an encryption mechanism, our distinction between key and key stream disappears. The key stream is the key. It must be as long as the message it is to encrypt. Although a courier with a large magnetic

tape containing the random bit stream forms a communication channel with a large capacity, this encryption scheme seems somewhat unwieldy. Nonetheless, it does have a very important characteristic to recommend it for use in certain situations.

Because the keys are completely random, it is possible to find a candidate key stream that decrypts a given intercepted ciphertext message into any plaintext message of the same length. A cryptanalyst has no way of determining which is the right key and which is the right plaintext. Thus, there is one key that decrypts IPOOEHWLRCR as ILOVEMOTHER; another that yields IHATEMOTHER; a third that produces ATTACKATTWO; and one more that generates DONOTATTACK. The cryptanalyst has no way of determining which is the correct decryption. Stated another way, all decryptions are equally likely. In general, it is impossible for anyone who captures the ciphertext stream to determine statistically that one plaintext stream is more likely than any other. Even if he or she can guess a likely word in the plaintext, he or she cannot determine where to place it or what the remainder of the message might be. Perfect confidentiality occurs because no amount of analysis, and not even an exhaustive search were he or she to try it, will help the intruder guess the plaintext. This cryptosystem is unbreakable.

A stream cipher with a completely random key stream is called a one-time pad. It derives its name from the keypad that its users once employed and from the fact that any use of a key stream more than once can be disastrous. If the key stream is reused, the difference between the two ciphertext streams is the same as the difference between the two plaintext streams, the key stream canceling. Now an analyst who looks at the differences between the statistically most common letters in the alphabet will yield a breaking of both plaintexts. The one-time pad is the only kind of cryptosystem that exhibits perfect confidentiality. As such, it is often used for the most important of diplomatic correspondences. For everyday transmissions of lower priority between the many users of a communications network or the many files to be protected on sensitive databases, something less demanding in key handling is required. A block cipher, such as DES, in one of the feedback modes or a key stream with some other cryptographically strong pseudorandom bit stream generator is an approximation to a stream cipher with a completely random key stream.

All strive to achieve some form of computational security. This means that, given the computing resources available to a prospective intruder, it is very unlikely that he or she will be able to break a single cryptogram. In evaluating the computational security of a system, we must examine the computational time and resources required for each possible attack compared with legitimate decryption. This is the work factor associated with each attack.

Some estimate of the intruder's computing power and technology is also necessary. An intruder can compare his or her capability for attack with the potential value of the sensitive data he or she is trying to steal. That value can be measured in dollars, in time, or in intelligence. Unfortunately, the last metric is somewhat difficult to quantify. If, given the size of the work factor, the cost of the computational power needed to mount each attack exceeds the value of the information we are protecting, our system is computationally secure.

Public key cryptography[1]

Public key two-key cryptosystems may be considered to be supplementary to conventional cryptography, such as DES. Diffie and Hellman first envisioned a cryptosystem in which decryption keys cannot be derived from the corresponding encryption keys. Three public key systems that have been widely implemented are RSA (Rivest, Shamir and Adleman), ElGamal, and the Diffie-Hellman key exchange system. We use the RSA system as the main example for this discussion, because it is the most widely adopted by industry and the international standards community.

As mentioned, the important difference is that public key cryptography uses matched pairs of keys. For example, Alice has one for encryption (E_A) and one for decryption (D_A). The encryption key is called the public key and the decryption key is called the private key. One entity is responsible for each matched pair. The strength of the public key process is twofold. First, the public key (E_k) can be electronically published in a network directory for wide access. Second, anyone (for example, Alice) in a network system can send a secret message to the holder of the private key (for example, Bob) by using the public encryption key of the recipient (E_B).

Public key cryptography can provide secure key management or key exchange functions to transmit secret conventional keys to the receiver (Bob) or perform an equivalent operation. This process supports message privacy because the secret key, such a DES key, is transmitted with the protection of a public key algorithm. Message privacy is achieved using the conventional secret key to encrypt one or more messages between the sender (Alice) and the receiver (Bob). We discuss these and related issues in the subsequent sections.

RSA uses a pair of parameters consisting of a public exponent and an arithmetic modulus. Briefly, the plaintext M (message) is represented as a sequence of bits by using some encoding scheme. The sequence is then divided into blocks X of the largest length that can be interpreted as the

[1]The presentation on public key cryptography is adapted, in part, from [NECH91].

binary expansion of a number less than the modulus n. Encryption then produces numbers Y of the same binary length. The relationships are as follows:

n	=	Arithmetic modulus
e	=	Public exponent
d	=	Secret exponent
Y	=	X_e mod n $(0 < X < n)$
X	=	Y_d mod n $(0 < Y < n)$
X, Y	=	Data blocks that are arithmetically less than the modulus

The modulus n is chosen to be the product of two sufficiently large prime numbers p and q: $n = p \times q$. The value of n and e together form the public key; d and the two prime numbers — p and q — constitute the private key.

The exponents are chosen so that

$$e \times d = 1 \bmod (p - 1)(q - 1)$$

A key length of between 512 to 1,024 bits is generally recommended for RSA, as compared with 56 bits for DES (plus 8 parity bits). For approximation purposes, we can say that the strength of the RSA using a key length of 512 bits is generally comparable to a key length of 56 bits for DES. One reason for the general comparability of such different key lengths is that the computational processes differ substantially. An attack or cryptanalysis against RSA is considered, in part, to be a function of the difficulty to factor large numbers. Therefore, RSA is generally associated with large keys. The strength of DES is considered, in part, to be a function of the number of computational rounds in its algorithm (16 rounds). These 16 rounds, when coupled with a key length of 56 bits, are claimed to provide adequate resistance to attack.

Message confidentiality and authenticity. Message confidentiality can be supported by the transformations of public key systems that have the relationship $D(E(M)) = M$. The notation $D(E(M)) = M$ refers to the decryption of the ciphertext $C = E(M)$, which yields the plaintext message M. The ciphertext is created by $E(M)$ or encrypting the plaintext M. We designate the sender A as Alice and the receiver B as Bob.

For example, if Alice (A) wishes to send a secure or private message M to Bob (B), then Alice must have access to E_B (Bob's public key). We denote the common encryption algorithm using Bob's public key as E_B and the common decryption algorithm with his private key as D_B. The notation for this discussion of public key cryptography uses subscripts to refer to the sender (Alice) and the receiver (Bob) rather than keys. For example, we say that Alice encrypts the message M with Bob's public key (E_B).

In other words, Alice encrypts M (the message) by creating ciphertext $C = EB(M)$ and sends C to Bob. Bob reverses the process when he receives C by using his private transformation D_B (Bob's private key) for decryption. This process requires that Bob computes $DB(C) = DB(EB(M)) = M$. We also generally refer to this process as Bob uses his private key (D_B) to "read" the encrypted message or ciphertext C.

If Alice's transmission is intercepted, the attacker or intruder cannot decrypt C (the ciphertext) since Bob's D_B (Bob's private key) is only known by Bob. This process provides for confidentiality. We assume that any entity in the network can access E_B (Bob's public key), because Bob has no means of identifying the sender. Also, Alice's transmission could have been changed. Therefore, authenticity and integrity are not assured in this example. However, authenticity and integrity can be provided.

Authentication of the sender (Alice) and integrity of the message (M) can readily be satisfied by using certain public key processes. The mathematical transformations in a public key system can be achieved in a variety of ways. In general, where Alice wishes to send an authenticated message M to Bob, he is able to verify that the message was sent by Alice and was not changed. Alice could use D_A (Alice's private key) to compute S (signature or signed text) = $D_A(M)$ and send S to Bob. We generally refer to this process as Alice signing her message. The signed message is also referred to as a digital signature. Bob can use E_A (Alice's public key) to find $E_A(S) = E_A(D_A(M)) = M$. Assuming M (message) is valid plaintext, Bob can verify that S was actually sent by Alice, and was not changed in transit. Verification follows from the one-way nature of E_A (Alice's public key). If a cryptanalyst or an intruder could start with a message M, he or she could find S' such that $E_A(S') = M$. The implication is that the intruder can invert or reverse E_A. However, inversion is not computationally feasible in this public key process.

Verifying the sender's (Alice's) identity could be difficult if M (the message) or any portion of M is a random string. For example, it may be difficult for Bob to determine that S is authentic and unchanged based only on review of $E_A(S)$.

In practice, a slightly more complex procedure is generally used. Variable-length long messages are uniquely reduced to fixed-length representations by an auxiliary public hash function or algorithm H. Therefore, Alice is actually "signing" ($H(M)$). This process yields a digital signature $S = D_A(H(M))$. Alice sends her digital signature S, which is unique to a given message M, to Bob along with M. If Alice encrypts her message M and digital signature $S = D_A(H(M))$ with Bob's public key (E_B), we can say the result is a digital envelope.

Bob can compute $H(M)$ directly when he receives a digital envelope. First, he "opens" the envelope by decrypting it with his private key (D_B). Second, $H(M)$ is found by using Alice's public key to operate on her signature S — that is, $E_A(D_A(H(M))) = H(M)$. Third, $H(M)$ may be checked

against $E_A(S)$ to ensure authenticity and integrity of M. The ability of a cryptanalyst or intruder to find a valid S' (digital signature′) for a given M (message) would violate the one-way nature of E. The hash function or algorithm (H) must also be one-way. A strong hash function has the property that it is computationally infeasible to find a message (M) which hashes to the same digest as a given message (M') with $H(M) = H(M')$. A security risk is that if Bob could find M' with $H(M') = H(M)$, then Bob could claim that Alice sent M'. A judge receiving M', $H(M)$ and S would reach a false conclusion.

Sending C (ciphertext) or S (digital signature) as shown above ensures authenticity and confidentiality. Confidentiality was provided because only Bob could "open" the digital envelope containing M and S. Bob used his private key (D_B) to open it.

If no digital envelope were used, M (the message) and S (the digital signature) would be transmitted in the clear. An attacker or intruder who intercepts C (ciphertext) = $S = D_A$ (Alice's private key) (M) may have access to E_A (Alice's public key) and could therefore compute M (message) = $E_A(C)$. Therefore, confidentiality of M is denied.

International electronic commerce may require communication systems that provide confidentiality, authenticity, and integrity. However, in some cases it is possible to use the same public key system for these security services simultaneously. For example, RSA supports digital signature and confidentiality. In the authenticity/integrity-related process, D (decryption) is applied to M (message) or $H(M)$. This contrasts with applying E (encryption) to M (message) for confidentiality. If the same public key system is to be used in both cases, then $D(E(M)) = M$ and $E(D(M)) = M$ must both hold; that is, D (decryption) and E (encryption) are inverse functions. A requirement is that the plaintext space (the domain of E) must be the same as the ciphertext space (the domain of D).

In practice, there are no generally available systems versatile enough for the last usage without modification. There is only one major public key system (RSA) that satisfies $E(D(M)) = D(E(M)) = M$ (message). The absence of a common domain between two users creates a technical problem in using such a system for confidentiality and authenticity.

Figure 5 illustrates a method of achieving confidentiality and authenticity in a public key process. The message M is placed in a digital envelope which is sealed with Bob's public key (E_B).

The public key process in Figure 5 is a simplified version of a process for confidentiality and authenticity. Certain issues, such as the question of domains, are not considered in the figure. The illustrated public key system complies with a hash function H. This system works with any encryption and any signature. They need not be related. However, the verification for the DSS is slightly different.

Applicability and limitations. Public key algorithms are computationally intensive. Therefore, confidentiality of *M* (the message) can be achieved only for short *M*s. The resulting slow encryption process may be referred to as a low-bandwidth secure transmission. In contrast, conventional key algorithms, such as DES, are must faster for encryption. Therefore, conventional key algorithms can produce wide-bandwidth secure transmissions.

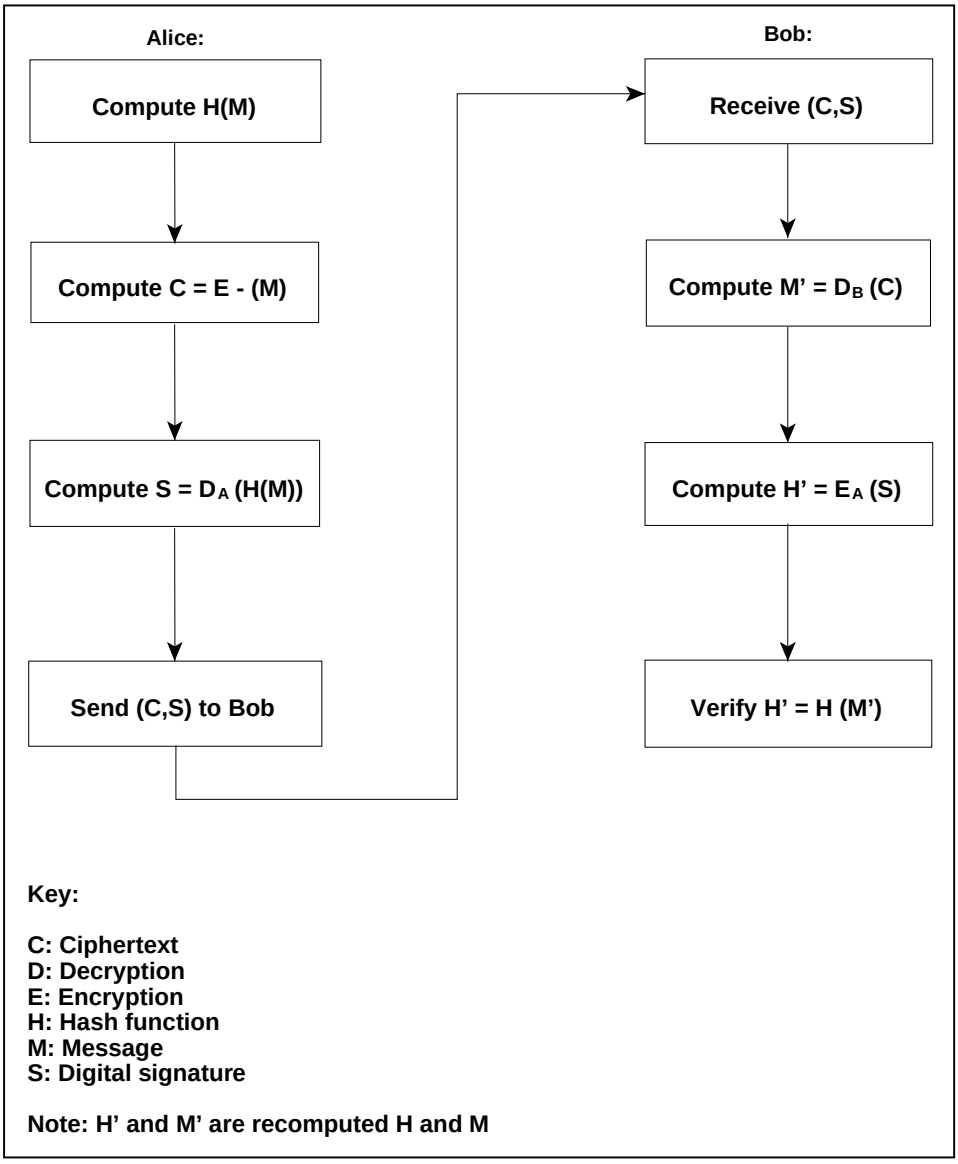

Figure 5. Using a public key process for confidentiality and authenticity.

Chip and algorithm breakthroughs will most likely continue to occur. Therefore, we do not rule out certain near-term and long-term uses of public key algorithms for message privacy. However, bulk encryption remains the domain of conventional cryptographic systems. These systems use fast encryption techniques such as permutations and substitutions.

The international electronic commerce process uses public key for two major applications:

- Secure distribution of secret conventional keys, such as DES keys, for bulk encryption.
- Digital signatures (*S*s).

In electronic commerce, there is a need for confidentiality of conventional decryption keys and public key private keys. There is also a need for integrity of encryption keys, symmetric or asymmetric. For example, if Alice can trick Bob into believing that the encryption key she sent him (for which she has the corresponding decryption key) is that of the president of the XYZ Corporation, then she can read any secret that Bob is sending him. This case includes any conventional key system used by Bob to send the president of XYZ encrypted data.

Digital signature

Authentication, nonrepudiation, and integrity checks can be supported with a digital signature. A digital signature is similar to a written signature, however, it is stronger. For example, detection will result from any attempt to change the message content or to forge the signature. We note that a Message Authentication Code (MAC), as defined in ANSI X 9.9, provides integrity protection against alteration, but does not provide nonrepudiation because of the sharing of the conventional secret DES key. (Another term for a MAC is a manipulation detection code, or MDC.)

A digital signature must be a function of the entire document. Changing even a single bit should produce a different signature. A signed message cannot be changed without detection.

Public key digital signatures. The use of public key digital signatures and supporting hash functions can provide both authentication and verification of message integrity. Hash functions, which have been briefly introduced, will be discussed further. They can also serve as cryptographic checksums used for validating the contents of a message. Public key schemes supporting authentication permit generation of digital signatures algorithmically from the same key repeatedly, although the actual signatures are different. Digital signatures are a function of the message and a long-term key. Therefore, key material can be reused many times before replace-

ment. Hash functions also reduce the impact of the computationally intensive nature of public key algorithms.

Public key digital signatures are generally preferred for electronic commerce because

1. private keys can be used repeatedly for generating digital signatures algorithmically, and
2. nonrepudiation of the sender (Alice) is inherently a part of the system design.

Therefore, public key implementation of digital signatures is effective and versatile.

Nonrepudiation. Nonrepudiation is the system capability that prevents a sender (Alice) from denying that she has sent a message. The integrity of nonrepudiation is a function of the degree of security maintained for the sender's (Alice's) private key (D_A) [NEED78, POPE79]. For example, Alice could repudiate or deny sending a message if D_A is compromised. Depending on the applicable legislation, Alice may still be held liable for messages signed before the compromise was reported to a central authority. Certain administrative approaches have been proposed for incorporation into protocols. Most of these involve use of some form of arbitrator [DEMI83]. However, certain disputes may require litigation, because nonrepudiation is a critical business issue.

One method of supporting nonrepudiation is to use a central authority. For example, the receiver of a message (Bob) sends a copy to the central authority. The central authority can verify sender's (Alice's) signature. This verification provides assurance that there is no report that Alice's private key (D_A) was compromised at the time of sending. In this case, Alice would have to rapidly report the compromise of her private key. We must also consider the impact of the increased workload of the central authority on the throughput of the network.

An alternate approach is to use time stamps [DENN81, MERK82]. Although a network of automated arbitrators may still be required, the system overhead is modest because the arbitrators only have time stamp messages. A receiver (Bob) may check the validity of the sender's (Alice's) private key by checking with a central authority. Bob has a degree of assurance of nonrepudiation if the received message is time stamped before the validity check. He still has to determine if a compromise is discovered and reported later.

Legal requirements for nonrepudiation may include a requirement that the sender (Alice) is responsible for signing until a compromise of her private key is reported to the central authority. Implementation of this approach could require an on-line central authority and real-time validity checks and time stamps. In addition to peak load concentrations that

may occur at the central authority, certain requirements for a network-wide clock should be considered. A network-wide clock has other security vulnerabilities, such as vulnerability to forgery of time stamps [BOOT81].

If users, such as Alice, are permitted to change their private keys, a central authority should archive past keys to assist in resolving disputes. Each industry should have a set of legal and administrative safeguards to maintain continuity of operations in the event of a compromise or change of keys. For example, credit card systems have effective legal and administrative provisions for cases of lost or stolen credit cards.

Hash functions. Hash functions or algorithms (H) have been introduced as a method of producing a fixed-length representation of a variable-length message M. As mentioned, public key algorithms are generally computationally intensive and compute more slowly than conventional algorithms. Therefore, it is usually not desirable to apply a digital signature directly to a long message. Since we also want to sign the entire message, we need an algorithm to reduce the size of the message. Hash functions or algorithms meet this need for computation of digital signatures to supplement public key techniques. For example, MD (Message Digest) 4, from R. Rivest, produces a 128-bit representation or message digest of a variable-length message. RSA is used to encrypt this message digest with sender's (Alice's) private key (D_A). This becomes $S = D_A(H(M))$. Other hash functions that can be used include MD 5, from R. Rivest, which essentially adds an additional computational round to MD 4.

The encrypted message digest is a digital signature that can be attached to the message for secure transmission in a digital envelope (in this case, containing the digital signature and the message M). As mentioned, a digital envelope is sealed by the public key E_B of the receiver, Bob.

The receiver (Bob) may validate the signature on $H(M)$ and then apply the public function H (hash function) directly to M (message) and verify that it matched the received signed version of $H(M)$. Authenticity and integrity of M are validated simultaneously. Only integrity would be assured if $H(M)$ were unsigned.

Hash functions should produce unique message digests. However, it is theoretically possible that two distinct messages could be reduced to an identical same message digest and cause a collision. Collisions cannot be avoided completely because there are generally more potential messages than the number of possible message digests. In practice, the probability of collisions should be very low. For hash functions with random or near random output, the probability of collisions is a function of the size of the message digest and the number of bit sequences that are meaningful messages.

In public key cryptography, the minimum requirements for a hash function include the ability to adequately support the authentication process. For example, if we have a message M and a message digest MD, it must

not be computationally feasible to find another message M' that also reduces to MD. Therefore, forgery can be avoided because appending the signed MD to M' would not verify as a valid signature.

Public key digital signature sequence. A public key digital signature process is briefly highlighted:

1. Compute a unique fixed-length message digest MD from the message M.
2. Use Alice's private key (D_A) to form the signature as encrypted hash, that is, $D_A(H(M)) = S$.
3. Attach Alice's signature S to her message M.
4. Seal in a digital envelope M and S with Bob's public key (E_B) for authenticity and confidentiality.
5. Bob opens the digital envelope on receipt using his private key (D_B).

Confidentiality is provided with the digital envelope, because only Bob can open the digital envelope with his private key (D_B). He validates Alice's signature S by computing $H(M) = E_A(S)$. As mentioned, Alice's public key (D_A) is a trapdoor one-way function. Therefore, an intruder should not be able to determine S' such that $H(M') = E_A(S')$ for a given forged message M′. As a result of this situation, Alice's signature cannot be forged. Also, if Alice attempts to repudiate the message sent to Bob above, Bob may present M (message) and S (digital signature) to a judge. The judge can use Alice's public key (E_A) to compute $H(M) = E_A(S)$. If Alice's private key has been kept private, then only Alice could have sent S. This is nonrepudiation.

To provide for nonrepudiation, Bob can use his private key to open DE (digital envelope) = M, $D_A(H(M)$. A judge can use E_A (Alice's public key) to operate on $D_A(H(M)$ and compare the results to $H(M)$.

Digital signatures and certificate-based systems. Electronic commerce requires sender authentication, data integrity, and nonrepudiation. These three security services are achieved with the use of digital signatures in distributed open systems. Certificate-based public key systems provide effective implementation.

For example, the Internet uses certificates to make public keys available to authorized entities. These issues are discussed in Essay 17 on Privacy Enhanced Mail (PEM). For example, PEM uses RSA and certificates derived from CCITT Recommendation X.509 for Secure Directory [CCIT88c]. Using RSA in X.509, Bob's (the receiver's) public key is cryptographically sealed (wrapped) in a certificate, along with other identification information. A trusted third party, called a Certification Authority (CA) in X.509, uses its private key (DCA) to seal the certificate. The use of PEM and

X.509 with DSS may not be exactly the same as for RSA. The PEM protocols may need to be extended to facilitate multiple algorithms.

Since X.509 provides for multiple CAs, a certification authority hierarchy or tree can be constructed. Authorized network entities (users) have the applicable CA's public key to decrypt or unseal the receiver's (Bob's) certificate in a directory. It may be necessary to repeat the process for nested certificates. The result is the receiver's (Bob's) public key, which can be used to send encrypted messages to Bob that only he can decrypt with his private key. Certificate-based key management is another way of describing this process. This process supports a zero knowledge technique that is being standardized as DIS 9979.

Public key management

In public key systems, the key management problem is inherently simple and relatively low risk (compared with conventional key management, for example, ANSI X9.17). For instance, the key information to be exchanged between users, or between a user and a central authority, is public. Also, a physical mail system might be satisfactory to communicate with the central authority, if redundant information is sent via an insecure (electronic) channel.

Management of public keys. We have briefly introduced the need for Alice and Bob to exchange their public keys. One reason is that public keys do not need privacy in storage or transit. For example, public keys can be managed by an on-line or off-line directory service, or they can also be exchanged directly by users.

Integrity has also been introduced. For example, if Alice thinks that the intruder's public key (E_I) is really Bob's public key (E_B), then Alice could possibly encrypt using E_I. The result would be that I could decrypt using D_I. Integrity should also be considered, because any error in transmission of a public key could eliminate its usefulness. Therefore, error detection is desirable.

A central authority, such as the Certification Authority (CA) that we introduced, is generally required for electronic commerce. However, there are situations where the CA may not have to be on-line. For example, Alice could retain Bob's public key for future use.

Use of certificate-based key management. We introduced certificate-based key management as a way of providing authenticity and integrity in the distribution of public keys [KOHN78]. A certificate-based system requires a central issuing authority CA (Certification Authority in CCITT X.509). For example, Alice will generally follow some form of identification and authentication procedure in registering with the CA. In addition, registration can be handled by a tree-structured system. In this case, the CA

provides certificates to local CAs. The local CAs can register users at lower levels of the hierarchy.

In the general case, Alice receives a certificate signed by the CA (Certification Authority) and containing E_A (Alice's public key). The CA prepares a message M containing E_A, identification information for Alice, a validity period, and so on. Her certificate is computed by the CA as $CERT_A = D_{CA}(M)$. A certificate is a public document that contains E_A and authenticates it. The authentication occurs because the CA signs $CERT_A$. As we have mentioned, certificates can be distributed by the CA or by users. Our discussion of certificate validity can also be considered as a generalization of time stamping.

There are exceptions to the utility of time stamping. For example, a certificate may be compromised or withdrawn before its expiration date. Therefore, if certificates are retained by users (rather than being requested each time from the CA), the CA must periodically publish an invalidated certificate list.

Public key and conventional key management issues

We need public key management for confidentiality of private keys and integrity of public keys. In addition, we need secure delivery for conventional secret keys to assure confidentiality and integrity. In either case, if we have a hierarchy of keys with the confidentiality and/or integrity of each key guaranteed by some key one level up, we need the secure delivery of the key at the highest level in some secure channel. Public key standards that provide for these considerations include CCITT X.509 and the Internet Privacy Enhanced Mail (PEM).

Secure delivery of certain keys, such as public keys, may involve delivery in a nonelectronic channel at the highest level of trust. For example, some public keys may be delivered in person or by trusted courier to a Certification Authority (CA).

The applicable standards involve, in part, using public key systems for secure and authenticated exchange of verified identities and data-encrypting keys between two parties. Data-encrypting keys are secret shared keys connected with a conventional cryptographic system that may be used for bulk data encryption. The public key approach permits users to establish common keys for use with a system such as DES.

Conventional key systems often use a central authority for assistance in the key management and exchange processes. Use of a public key system permits users to establish a common secret key without the riskof a third party having the secret key. In other words, a public key system has a lower risk than a conventional key cryptosystem for key management and exchange. Therefore, international standards to support the evolving open distributed processing systems include public key management concepts.

Public key cryptography can be used to distribute conventional secret keys securely and effectively. The overhead is modest because keys are essentially short fixed-length messages. Also, digital signatures are generally applied only to outputs of hash functions, which are also the equivalent of short fixed-length messages. Therefore the bandwidth limitation of a public key cryptosystem is not a major factor for these applications.

Authentication, integrity, and key management issues for conventional key systems

We focus in this section on issues pertaining to MAC for authentication and integrity (X9.9) and a related standard for conventional key management (X9.17). Digital signature is discussed in the following section.

Certain financial systems use conventional cryptography to provide for authentication and integrity of financial messages. In this case, encryption is performed and used to generate a MAC, which is appended to the cleartext for transmission. The receiver (Bob) calculates the MAC and compares the calculated and received MAC. A match ensures that the sender (Alice) possessed the proper conventional encryption key and that the message was undamaged. The limitation of the MAC process is that Alice and Bob share the same secret key.

Historically, MACs have been used to provide message authentication in financial systems. The message remains in cleartext, which may be required in certain international banking communities. One of the difficulties of using MACs has been the complexity of conventional key management. However, a standard has evolved to assist in key management for well-defined communities.

We briefly introduce two representative ANSI (American National Institute of Standards Institute) standards for wholesale banking that have also been adopted internationally:

- ANSI X9.9-1982, 1986: Financial Institution Message Authentication (Wholesale).
- ANSI X9.17-1985, 1991 (Extension): Financial Institution Key Management (Wholesale).

It is important to note that in the interbank (wholesale) electronic funds transfer environment the primary goal is authentication rather than privacy. Privacy can be provided only by use of an additional key.

Message Authentication Code (MAC): Standard ANSI X9.9-1982, 1986. The Message Authentication Code (MAC) (ANSI X9.9), not to be confused with Mandatory Access Control (MAC), is a cryptographic checksum appended to a message. It seals the message against modifi-

cation. All fields such as time, date, sources, and so on included in the checksum are rendered unalterable. Either the entire message or selected fields are processed through the algorithm using the Cipher Block Chaining Mode (CBC). As mentioned, the last block is the only output of the process that is used in the MAC. MAC requires a key management protocol, such as ANSI Standard X9.17.

Financial Institution Key Management: Standard ANSI X9.17-1985, 1991 (Extension). There are three environments in ANSI X9.17 for conventional key establishment:

- *Point-to-point environment.* Two parties share a master key, and the master key is used for distribution of working keys.
- Key Distribution Center *(KDC) environment.* Master keys are generated by a Key Distribution Center and are shared between each entity and the centralized server.
- Key Translation Center *environment.* One entity originates the working key. (This is a minor variation on Key Distribution Center.)

Two entities can share in the key management process in a point-to-point environment. Each entity has the same master key that is used to distribute working keys for individual messages. Working keys are generally changed periodically, depending on the risk associated with the application. For example, a high-risk environment could require a new working key for every transaction or every day.

The full implementation of this standard involves the second option, namely, the Key Distribution Center (KDC) environment. A trusted entity in the network is designated to perform the KDC functions for a defined community of users. Each entity in the user community has to establish a trusted relationship with the KDC, which has a duplicate of each of the user's master keys.

A Key Translation Center is used when one of the entities wishes to perform some of the KDC functions. This entity originates the working keys for the user community.

Risk and cost of conventional key management. The concentration of risk is a security disadvantage of conventional key management. Risk concentration may be considered a function of the need to have the secret keys for a network community concentrated in one node. Also, the cost or overhead of conventional key management is relatively high because of the need for the KDC to share all master keys. Substantial complexities may occur if a large number of KDCs wish to join together in ad hoc relationships to support international electronic commerce.

For example, if Alice and Bob wish to communicate securely, they must first securely establish a common key. As mentioned, one possibility is to

employ a third party such as a courier. Historically, couriers have been used; however, electronic commerce requires electronic key management.

The most common approach for Alice and Bob to use in conventional key management would be to obtain a common key from a central issuing authority or a key distribution center [BRAN75]. The higher risk occurs because the key distribution center is at risk to attack from an intruder. Unfortunately, a single security breach by an intruder would compromise the entire system. For example, the intruder could passively eavesdrop without detection.

The higher overhead of a key distribution center occurs, in part, because of the bottleneck effect. Since each pair of users needing a key must access a key distribution center at least once, the volume of activity would increase rapidly. If the number of users is n, then the number of pairs of users wishing to communicate could potentially be as high as $n(n-1)/2$. In addition, each time a new secret key is needed, at least two communications are required for the user pair and the key distribution center. Furthermore, network availability could become a function of the key distribution system. Questions should also be asked concerning the capability for maintaining effective access control for the system containing the secret keys. Examples of systems that provide this type of access control are security-enforcing or trusted systems.

Other aspects of conventional key management are not unique to conventional cryptography. For example, life-cycle management is required over the life of conventional keys, which can include the need for archiving, for example, five to 30 years for business purposes. Life-cycle management procedures include distribution, storage, and destruction. Key maintenance is also required. For example, some keys may be lost or compromised. In addition, employee changes may make it necessary to cancel some keys and issue others.

Manual key distribution must occur at least once for conventional cryptographic key management, after which automated distribution can occur. Master keys or key-encrypting conventional keys (KEKs) are the manually distributed keys. These keys are used only to encrypt other conventional keys called "working keys." Other terms for "working key" include "data-encrypting keys" (DEKs).

An introduction to encryption in networks

We briefly discuss some network aspects of encryption. Our purpose is to introduce some of the common terms and concepts for link and network encryption. However, we do not address the encryption issues associated with communication protocols and internetworking. For some of these issues, see Essays 17 and 18.

Relating encryption to data network communications. The increased application of communication technology in international electronic commerce has accelerated the need for security in data network communications. These communications support global interconnectivity and distributed operations, thereby introducing security risks. New developments in communication protocols offer promise of providing solutions to reduce certain security risks. A protocol specification details the control functions that may be performed, the formats and control codes used to communicate those functions, and the procedures that the two entities must follow. We introduce some of the basic issues that are useful when evaluating security services that can be satisfied with encryption mechanisms.

Link encryption. The most straightforward application of encryption is to the communications link. Information is not processed as it passes on a link. There are no packet switches, no gateways or other intermediate systems. All of the information can be encrypted to prevent release of message contents. Traffic analysis can also be prevented by padding (adding null or blank characters so that all messages are the same length). Padding entails no additional cost if dedicated links are used; the converse is true on shared links. Link encryption provides protection only on the communications link. Information in an intermediate node reverts to plaintext. Protection of this plaintext involves physical protection of the node hardware and trust of the node software. Naturally, there are costs associated with physical protection as well as operation of the encrypted links, mostly in key management and distribution.

Link encryption is the oldest and most common form of encryption in computer networks. In a packet-switching network, link encryption can be used to encrypt the communication links between keys such as hosts and switches.

A simple view of data communications is to consider the system as composed of two pieces of equipment closely collocated. The communication path is protected and error-free, and possesses unlimited bandwidth. Equipment must be added to approximate this ideal in the real world.

Link encryption illustrated. Figure 6 shows a schematic representation of data circuit-terminating equipment (DCE) adapting a physical circuit to carry data communications. Figure 7 adds encryption equipment.

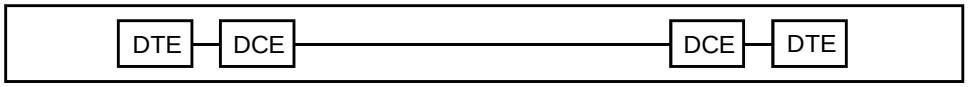

Figure 6. Data circuit without encryption (DTE: data terminal equipment; DCE: data circuit-terminating equipment).

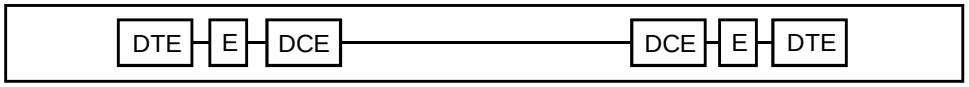

Figure 7. Data circuit with encryption (E: data encryption equipment).

Link encryption for point-to-point circuits. Link encryption is appropriate for point-to-point circuits. In addition, it can be easily placed in the OSI context. For example, the entire bit stream is encrypted when link encryption is present at the Physical Layer, layer 1. Encryption at the Data Link Layer, layer 2, results in some fields in plaintext and others encrypted.

End-to-end encryption. End-to-end encryptionencryption (E3 or E^3) is different from link encryption in that we no longer have to expose information in cleartext in packet switches — that is, at each node. The reason for this difference is that E3 refers to encryption above the Data Link Layer. Simple link encryption is inadequate when applied to ISO layered protocols for wide area networks (WANs) in layers 3 to 7, because commercial WANs generally do not provide link encryption capabilities among the switches.

When discussing E3 with respect to the ISO seven-layer model, we usually refer to encryption by layer. For example, encryption in layer 3 or 4 could be called Network or Transport encryption, respectively. Certain protocols that are being considered by ISO use more specific designations, such as encryption at the top of layer 3 (SP3, Secure Protocol 3) or the bottom of layer 4 (SP4).

Encryption must be generalized to protect the protocol data units (PDUs) at a given layer. Extending encryption into higher protocol layers increases the number of entities protected, at the cost of interfacing and the overhead associated with additional hardware and/or software. Higher layer encryption and the accompanying protocols can be intrusive. However, substantial hardware and software advances are being made. Therefore, there is a gradual international trend to higher layer encryption and encryption in commercial application software packages.

File encryption for storage protection. File encryption is the encryption of a file in a computer system and/or a distributed processing system. It gives protection in case someone breaks through electronic system defenses and accesses the file. File encryption also enables us to put the file on a floppy disk and mail it without any special protection. In other words, file encryption substitutes for physical protection. The main problem with file encryption is losing the key. Losing the key in file en-

cryption is like losing all our data when our hard disk crashes, except that, with file encryption, our backup copies probably are lost as well.

A process that uses encryption to cryptographically "sign" or "seal" software before distribution has been introduced as digital signature. The digital signature is used to verify the integrity of the software in operation.

Integration of computer and communications security

In the past, computer security was used inside computers, and communications security was used outside on the transmission lines. Today this boundary is disappearing as file encryption, digital signatures, message integrity, E3 or E^3, password encryption, and other such applications are incorporated into computer systems. This change can strengthen functions such as identification, authentication, and access control. However, the integration of the two disciplines will require the interface of two cultures as two sets of rules are combined. This integration is complicated by the development of internetworking, which is bringing many technologies together, such as wired and wireless communications.

This interface of the two disciplines — computer and communications security — may require answers to systems questions. For example, should this integration of security-enforcing or trust technology in computer security (COMPUSEC) and cryptography in communications security (COMSEC) require that the information system provide both sets of security attributes — computer and communications security, that is, information security (INFOSEC)? Even as we bring the two disciplines together, we still may need well-defined interfaces and boundaries for reasons of modularity, certification, and international electronic commerce.