EDP ANALYZER

© 1978 by Canning Publications, Inc.

DECEMBER, 1978
VOL. 16, NO. 12

DATA ENCRYPTION: IS IT FOR YOU?

As we discussed last month, we foresee some major changes coming to the data processing environment in the next few years. The main sources will be the new office systems and the on-rushing micro-processors. One aspect of the new environment will be the growing use of data encryption (which turns information into 'gibberish' and can later recover it). There has been a lot of discussion in the trade press about encryption, plus a debate about the new U.S. federal data encryption standard (DES). Many companies might adopt the attitude that (a) "we will not need to encrypt our data," and (b) "even if we do have to encrypt it, we surely would not use the DES." There is a fairly good chance they would be making a mistake on both counts—and particularly the second. Here is why.

If there is such a thing as a 'typical' business data processing environment, it probably has the following characteristics. The organization has many data files, mostly organized by application and stored on magnetic tape. Some are stored on disk units, for fast access; in other cases, databases have been set up for serving multiple applications. These files contain business records, dealing with customers, suppliers, employees, products, work in process, finances, and so on.

While the organization does not want such data to be disclosed to outsiders, it generally would not be seriously hurt if some of the data were disclosed—or so it is believed. Financial data is guarded more closely than the other types mentioned.

Much the same situation applies to any data communications that this organization performs. Most data transmitted is for simple business transactions. Anything of a particularly sensitive nature is sent by other means, such as by courier or by registered mail.

For such an environment, is today's public debate on encryption methods really meaningful? Should such organizations be criticized if they adopt a 'couldn't care less' attitude about encryption?

Let's take a look at some reasons for encrypting.

Why encrypt?

For one thing, encryption is probably the only effective method available today for preventing the unauthorized disclosure of computer-readable data. If an organization is concerned about security in general, and data security in particular, then encryption should be considered. Other methods for controlling access to data, such as the use of passwords, provide little protection against an intelligent penetration effort.

But why should an organization be concerned about data security in the first place? One reason is that the business environment appears to be becoming more malignant. "There is nothing wrong about stealing from big organizations; after all, they are stealing from us" is a not-uncommon attitude among some elements of society. Social reformers may want to gain access to business files in order to analyze them and find embarrassing facts that can be used to pressure the organization into some desired action. Competitive organizations may find it helpful to 'browse' through the organization's files by means of remote terminals. Even competition among managers within the organization, each seeking promotion, may lead them to look for embarrassing data items in other managers' files.

Note that it is less and less realistic to assume that these 'penetrators' (people who deliberately gain unauthorized access to the data) will not have the technical skills needed for making effective attacks. Knowledge of computer technology is spreading throughout our society.

Another reason for data security is the obvious one—some organizations store and transmit data of great value, and such data must be protected. Funds transfer transactions, messages about new oil deposit discoveries, and trade secret data stored in computer files are examples.

Then there is a new requirement that is emerging—the need to protect information applying to persons, as covered by the new privacy laws. The unauthorized disclosure of such information can involve both civil and criminal penalties.

In short, organizations that have been accustomed to operating in a more or less benign environment may well find that the environment is changing. Encryption may not be so irrelevant as first assumed.

Then, too, as we discussed last month, we foresee a new generation of computer systems arriving in the next few years. Advanced security features, including data encryption, are sure to be main sales points for these systems. So computer users will be hearing *much* more about encryption in the months ahead.

At the same time, there are good reasons for not charging ahead with the use of encryption too rapidly.

Why be cautious?

The use of encryption will raise a whole host of new operating procedures. One problem area is that of 'key management,' which we will discuss in more detail below. Suffice it to say now that key management includes the generation of keys, transporting them to remote sites, protecting them against disclosure, and accurately entering them into the encryption devices. Mistakes in any of these steps can lead to severe difficulties.

What are keys? They are groups of digits (say, eight digits) that play the same role as a metal key does for locking and unlocking a door lock. Handling encryption keys can be even more difficult than handling physical ones.

Then the organization must make a choice of encryption methods, and this is not simple. (Of course, it *is* simple for U.S. federal agencies; they *must* use the new federal DES for unclassified data that is to be encrypted.) In general, there is no good way for an organization, using its own resources, to determine the strength of an encryption method. The size of the key space (the number of possible different keys) is, of itself, not an adequate way for determining strength.

From the point of view of society, apparently no one yet has a good idea of the social consequences of encryption. We have asked a good number of people about this point, and have done a literature search on it, to little avail. For instance, the wide spread use of encryption might thwart the sub-poena process. As far as we can tell, no one has really addressed the question of the possible side effects of encryption, and there are sure to be some.

Warranted or not, good for society or not, the use of encryption for business records will grow. Encryption is almost at the threshhold of being 'a thing to do.' However, the decision to use encryption is one that should be taken very seriously, as we hope will be made clear in this report.

Let us now briefly summarize some of the basic concepts of encryption.

Some basic concepts

Three articles in the *IBM Systems Journal* (Reference 1) provide good summaries of basic concepts in their introductions. Also, one of the sessions at the 1978 National Computer Conference (Reference 2b) included points of interest, on which we draw.

A cryptographic system seeks to turn human or machine intelligible data into 'gibberish' by means of either a code or a cipher, and in such a way that the original text can be recovered easily only by a possessor of the code or the cipher key. A *code* substitutes a code word or phrase for a word or phrase in the original plaintext. So coding and decoding involve a dictionary-type lookup. The use of codes is not relevant to our discussion. A *cipher* system operates on characters rather than on words or phrases—by permuting the sequence of the characters, or by substituting one character for another, or by a combination of these processes. Through the use of the key, it seeks to change the original stream of characters (plaintext) into pseudo-random noise (ciphertext). The methods we will be discussing are cipher systems.

Cipher systems for data processing sub-divide into two general types. One is block cipher systems where a block of, say, eight characters is encrypted and later decrypted; such systems always deal with full blocks of characters. The other type is stream systems, where a string of characters is encrypted one character at a time. Our concern here is with block cipher systems.

There are variations of block cipher systems, such as block chaining and cipher feedback systems. In general terms, an encrypted block—in addition to being stored or transmitted—is also fed back into the device to help encrypt the following block. This introduces 'noise' that can later be removed. We will discuss the importance of this below.

Another variation of importance in cipher systems is that of secret algorithm systems versus public algorithm systems. The algorithm is the procedure by which the plaintext is transformed into the ciphertext by the use of the key. One might think that the secret algorithms would be more secure. We side with the experts who say that the public algorithms, which have withstood attack by many interested parties, are more trustworthy. One cannot rely on a 'secret' algorithm staying secret.

Now consider the people who are not authorized to have the cipher keys but who attempt to unscramble the ciphertext—the penetrators. What a skilled penetrator hopes for is a short-cut solution to this unscrambling, by exploiting some weakness of the algorithm. Finding a short-cut for a truly strong algorithm is essentially an act of genius; one person might see a short-cut where hundreds of other skilled penetrators might not. For a 'garden variety' algorithm, finding a short-cut solution is much more routine, we are told. It is not possible for the lay user to know in advance if a short-cut will be found or how much effort it will require in its use. This is another advantage of public algorithm systems; any weaknesses that lead to short-cut solutions become publicized upon detection.

If a short-cut solution cannot be found, the other alternative is the exhaustive search. The penetrator must have a piece of ciphertext that he knows matches a piece of plaintext. Further, it must be assumed that he knows the algorithm (even if the user attempts to keep the algorithm secret). The penetrator then tries every possible key for transforming the plaintext to the ciphertext, or vice versa, until the right key is found.

It should be noted that strong cipher systems are designed under the assumption, as described by Walter Tuchman in Reference 2b, that the 'bad guys' (a) know everything about the algorithm, (b) have an accomplice who can insert any amount of chosen plaintext into the messages, and (c) have large scale, very fast computers available for performing the analysis. In fact, the 'bad guys' can be employees of the organization, with access to many system details. The cipher system must provide protection even under these circumstances.

Finally, the concept of work factor must be mentioned. This is the amount of work, measured in dollars or time or such, for a penetrator to achieve a solution and find the key. Designers of encryption systems seek to make the work factor very high.

These, then, are some of the basic concepts involved in encryption systems. Let us now consider the U.S. federal data encryption standard. It is a publicly known algorithm, and the debate on its strength is very informative.

The data encryption standard

What is now known as the federal data encryption standard (DES) was originally developed by IBM. It was submitted to the U.S. National

Bureau of Standards in 1974, in response to an NBS solicitation. NBS was seeking a government-wide standard for the encryption of unclassified government data. IBM's response was selected and DES was adopted as a federal standard in mid-1977.

DES is a block product cipher; that is, it uses both permutation and key-dependent substitution. The key consists of eight 8-bit bytes, of which every eighth bit may be used for parity purposes; of the 64 bits, 56 are used as the key. (This is the heart of most of the controversy about DES, as we will describe.)

Plaintext is inserted in blocks of 64 bits; if the text does not fill a block, padding characters may be used. In its first step, DES permutes these 64 bits; the 58th bit is moved to position 1, the 50th bit to position 2, and so on. Then begins a series of 16 rather complex iterations, using the key. The permuted 64-bit block is divided in two, a left 32 bits and a right 32 bits, which we will call L(0) and R(0). At the end of the first iteration, R(0) is just moved over without change to become L(1). But L(0) is EXCLUSIVE-ORed to a function of R(0) and key 1. Does this begin to sound sticky? We'll be brief.

What is this function? First, R(0) is expanded from 32 to 48 bits by means of a table. The table says: put bit 1 into both positions 2 and 48; put bits 2 and 3 into positions 3 and 4; put bit 4 into positions 5 and 7; and so on. So, 32 bits are expanded to 48. Then 48 of the 56 bits of the key are selected in a tricky fashion (not, of course, 48 bits in sequence), to give key 1. Then these two 48 bit numbers are exclusive-or added, giving a 48 bit number. This 48 bit number is entered into a set of eight 'S-boxes;' we mention this name because these are a part of the controversy. Each S-box accepts six bits in and delivers four bits out, as specified by eight tables. Every output number can come from one of four input numbers. The result is that the 48 bit number is now reduced to 32 bits again.

Why this funny expansion and contraction? We gather that it is so the penetrator cannot work backwards through DES, when trying to decrypt. The penetrator would know the permutation scheme because it is fully described in Reference 3a. But when the penetrator reaches an S-box, he will not know which one of the four possible outputs were used in any given case.

Anyway, this now-32-bit number is exclusive-ored with L(0) to give R(1). And the process is repeated with L(1) and R(1) as inputs—and so on for the rest of 16 iterations.

If it sounds like DES produces 'gibberish,' that is just what happens. It produces pseudo-random noise as output. If the same type of process is repeated for decryption, *using the identical key*, the original plaintext is obtained. But if there is a difference *of even one bit* between the encryption key and the decryption key, gibberish comes out, as discussed by Meyer in Reference 2a.

So that is a brief summary of DES. There is no way, surely, for non-experts in cryptography to know how strong DES is. So what is all the debate about? Let's look at that question next.

The debate on DES

We are trying to cover a lot of material in this one section and will not really do it justice. If you are interested in the debate, we urge you to read the references. (We thank those cited authors who reviewed our draft and offered comments.)

We see three main arguments being raised against DES. These are: IBM and the government are hiding things, the DES key is too short, and the new 'public key' systems are better than DES.

Keeping things secret

The U. S. Senate Select Committee on Intelligence considered it necessary to investigate a number of allegations about DES. Included among the allegations was the charge that the National Security Agency (NSA), under the guise of testing the algorithm, tampered with it in order to weaken it and create a 'trapdoor' which only NSA could tap. (A charge reported elsewhere was that NSA forced IBM to keep the design of the S-boxes secret, so that this might be the 'trapdoor.') Another charge was the NSA forced IBM to reduce the key size from the original proposal of 64 bits to the current 56 bits; perhaps this was a part of a 'trapdoor.'

Reference 4 is the unclassified summary of the Select Committee's findings. The committee had access to reports of two workshops (References 3c and 3d) that considered some of the charges.

It also had access to NSA and IBM personnel, as well as people at universities who are working in the subject area.

In short, the committee's report says Yes, NSA did convince IBM that a reduced (56-bit) key size would be sufficient. And NSA did assist indirectly in the development of the S-box structures. But the NSA certified that the final algorithm was, to the best of their knowledge, free of any statistical or mathematical weaknesses. Further, NSA certified, and IBM agreed, that NSA did not tamper with the design of the algorithm (other than what was just mentioned). IBM invented and designed the algorithm, made all pertinent decisions, and concurred that the 56-bit key size was more than adequate for the intended commercial applications for which DES is intended.

Further, we have been told that IBM never did propose a 64-bit key for DES, and that NSA's assistance in the S-box design was very, very indirect; the design was IBM's.

Reference 3d brings the picture into somewhat better focus; it is the report of a workshop organized by NBS to consider some of the charges levelled against the DES. Walter Tuchman of IBM reported that when IBM tried to get an export license for the algorithm, it was learned that the company had inadvertently used classified design principles. The structure of the S-boxes had been selected by IBM, not NSA, to give a stronger algorithm. It was an accident, not collusion, said Tuchman, that led IBM to use design principles for the S-boxes that the government now asks them to keep secret.

Martin Hellman of Stanford University, a leading challenger of DES, makes the point in Reference 3d that if DES is supposed to be a publicly known algorithm, then *all* aspects of it should be made public. The S-box structure is crucial to the algorithm's strength. How can an algorithm be certified as secure if a crucial part of it is kept secret, he asks.

The response made to this point is that the contents of the S-boxes *are* public knowledge; they have been published with the algorithm. It is only *why* these structures were chosen that is not disclosed. Further, if anyone can show that the algorithm can be strengthened by using other S-box structures, such a person is invited to do so.

That leads to the next charge: the DES key is too short.

Is the key too short?

This question is where most of the debate has centered. Most of the attacks on DES have argued that it should have a 64- to 128-bit key length. And NSA has been charged as the cause of the 56-bit key length.

For instance, Hellman is quoted in Reference 3d as saying, "...it is known that NSA systematically blocks the use of algorithms using longer (than 64 bit) keys." And at the 1978 National Computer Conference session on encryption, one attendee reported that his company tried to get an export license for an encryption device that used a 128 bit key. Upon the advice of NSA, he said, the government would not grant the license until the key length had been reduced to 64 bits

But the point that the backers of DES make is: the 56-bit length is *more than adequate*. So here is the debate: is it adequate or isn't it?

Hellman (in References 2a, 2b, and 3c) has led the attack on the key length. He argues that he has found a short-cut solution. But DES adherents consider his approaches as 'well-known' and adequately provided for with DES.

A 56-bit key has almost (as Hellman says) 100,000 million million different combinations—10 with 16 zeroes behind it. If one test takes only one microsecond (given a key, encipher a block of known plaintext and compare it with its matching ciphertext), it would take about 2,000 years to test all possible keys. So say the adherents.

But that isn't the way to look at the problem, says Hellman. Perform these tests in parallel, with many units. If this is not economically feasible today, it soon will be—to have enough fast units in parallel so as to test all possible combinations in one day.

So NBS convened a workshop of experts (Reference 3c) to consider this question. The workshop attendees extrapolated the probable characteristics of large scale integration (LSI) chips out to the early 1980s. They considered several alternative architectures for a 'key extraction machine' for making exhaustive searches of keys. Adding some special hardware to one of today's fastest computers (a CDC STAR) would allow it to generate *one solution in somewhere between 23* *and 91 years*. If a special machine were built with, say, one million DES chips—which might be accomplished by 1990—it might generate one solution in one day. But the cost of such a machine would be between \$50 and \$70 million, and the likelihood of success was estimated at only 10% to 20%.

Further, if such a machine *were* built, its existence probably could not be kept secret; a one-time order for that many DES chips would be noticed. When the existence of the machine was detected, the users could quickly start using multiple encryption, obsoleting the machine.

Hellman's next attack, given at the 1978 National Computer Conference (and found in Reference 2b, not in 2a), took another approach. "Assume I have an accomplice who can plant plaintext of my choosing—say, a block of eight blanks that I can be sure will occur frequently—in the message stream," says Hellman. "I will now be able to get blocks of ciphertext that correspond to the chosen plaintext. To break the system, I will build a large set of reduced tables in which the entries are (essentially) key and ciphertext, for the plaintext of my choosing. The creation of these tables will take several years of computing but I am under no pressure to get them done by a certain time. To find a key, I have to do a table look-up in the reduced table; this is a time/memory trade-off. If I put together a machine with 10,000 DES chips working in parallel (and we are skipping over some of the other details of the machine), I can perform 100 DES solutions *a day*, at an estimated cost of between \$1 and \$100 per solution." He arrives at this cost figure by estimating the cost for parts of the machine at \$4.2 million (today's parts at today's prices).

But—and there is a but—Hellman says, this system will not work with block chaining methods, where each encrypted block is fed back to help encrypt the following block. Hellman needs blocks of ciphertext that are matched to his plaintext and have been encrypted using only the key.

Tuchman says that this assumption of Hellman's is on very shaky ground. Almost *any* form of good 'pre-whitening'—such as good block chaining or cipher feedback—will inject an element of noise into the ciphertext and thwart the table lookup type of attack. Further, says Tuchman, all cryptographic products should include good pre-whitening as a matter of course.

Other supporters of DES point out that Hellman's time/memory trade-off is simply the well-known 'code book' type of attack—which was evaluated when the algorithm was under consideration as a federal standard. To thwart this type of attack, they say, one can simply introduce 'noise' into the message (Tuchman's pre-whitening) so that no two encrypted blocks are the same.

So, if Hellman could get his assumptions fulfilled—somehow being able to match plaintext and ciphertext for a selected sample of plaintext, not faced with the use of block chaining, and having a huge, fast deciphering machine—then DES might be broken by a table look-up method. DES needs a greater safety margin, he says. But this is a special case of penetration, it seems to us. That is, an accomplice must be planted within the target organization, the organization must not use pre-whitening or double encryption, and the deciphering machine must be built, made to work, and its existence must be kept secret. Only then would it work. But note: if all cryptographic products include good pre-whitening, this type of attack would be thwarted.

At the workshop reported in Reference 3d, the consensus of the experts was that the 56-bit key is 'adequate.' But we gather from the tone of the report that most of the attendees were unhappy that IBM and NSA had agreed on the use of a 56-bit key. A 64-bit key would provide a desirable amount of added protection some 8 to 10 years in the future, when a key extraction machine may be more feasible. Note that each added bit doubles the exhaustive search work factor.

We have been told that, for a strong algorithm such as DES, for which short-cut solutions are highly unlikely, a 56-bit key gives sufficient protection today. If and when exhaustive search for this key length becomes feasible, multiple encryption can be used with DES, thus extending its life indefinitely.

Interestingly, Hellman has also helped initiate the interest in 'public key' systems, which adherents think will play an important role in cryptography.

Public key systems

The concepts of public key encryption systems began with two papers, one by W. Diffie and M. Hellman (in the *IEEE Transactions on Information Theory*, November 1976) and the other by R. Merkle (in the *Communications of the ACM*, April 1978). These papers have stimulated much of today's interest in cryptography.

Diffie and Hellman proposed the concept of a 'trapdoor one-way' encipherment function. This is a function that allows relatively easy encipherment of a message with enciphering key E but involves *very* difficult deciphering without a second deciphering key D. Given a message M, the 'normal' ciphertext C of that message is C = E(M). Normal deciphering results in the message being retrieved: D(C) = D(E(M)) = M. The E key is thus used for *sending* a message and is of no value for deciphering the message.

In conventional systems, one key is used for both encryption and decryption. In a public key system, one key is used for encryption and another for decryption. Each participant in a public key system would have his own pair of E and D keys.

Next, let the participant's E key be publicly known. In fact, set up a public file that stores the E keys of all participants and makes them available upon request. But each participant keeps his D key secret.

If party A wishes to send an encrypted message to party B, he gets party B's E key from the public file, encrypts the message with it, and transmits the message to party B. Party B, and only party B, can decrypt the message with the secret D key, because it can take eons of time on today's fastest computer for anyone else to try to decrypt it. If this sounds like magic, we will indicate what the mathematical foundations of the method are that seem to insure this.

Rivest, Shamir, and Adelman (Reference 5), in a much quoted paper, have made some interesting extensions to the concept. Select D and E so that they can be used in either sequence, say the authors. That is, let D(M) = C and E(D(M)) = M.

Next, they say, let the encryption key be a *pair* of very large random numbers, E and N, and the decryption key be D and N. Let N be the product of two very large prime numbers (divisible only by themselves and by 1), P and Q. Each of these should be in the order of 100 decimal digits in length, making N 200 decimal digits long. To get P and Q, generate large random numbers and test them for primeness (which is relatively easy to do); it will take about 115 tries before a prime is found, they say. Although N will be publicly disclosed, no one knows how to factor a number of this size into its factors, P and Q; it would take millions of years on today's fastest computers, using the best-known factoring algorithms.

Once the participant has selected his N, he then determines his D to be "a large, random integer which is relatively prime to (P-1) x (Q-1)." This is number theory; for more details see Reference 5. Finally, the participant determines his E as the 'multiplicative inverse' of D. By using this means of determining E and D, they can be applied in either order, which leads to an interesting use.

To use the method, convert the message to a series of decimal digits. One way to do this is to code the alphabetic characters as two digits: A is 01, B is 02, etc. Next, raise this number to the power E (in essence, multiply it by itself E times), subtracting out the number N (the modulus) as often as possible, and keeping only the remainder. This remainder is the ciphertext message C. There are some short-cut methods for doing this and it can be done in a few seconds on today's high speed computers, say the authors.

To decipher C, just raise it to the D power, convert the resulting digits back into alphabetic characters, and presto, you have your message.

The 'interesting use' of this method is that of digital signatures, almost equivalent to handwritten signatures on contracts. For this use, the sender first uses his D key to encrypt message M, giving ciphertext C(1). Then the sender goes to the public file for the E key of the recipient, and enciphers C(1) with it, giving C(2), which is sent to the recipient. The message has thus been encrypted twice.

The recipient first uses his own D key to decipher C(2) to C(1). He then uses the senders public E key to recover M from C(1), in number form. Because the sender's E key did in fact produce the message, the recipient is sure that it came from the sender and from no one else. This 'signature' is strong enough to convince a judge, they say, as long as the sender's D key has not been compromised. (Of course, the judge also would probably want expert cryptographic testimony on the strength of the algorithm, we have been advised.)

Some arguments against. One of the main disadvantages of public key systems is that they are quite a bit slower in operation (measured in the number of digits enciphered per second) than, say, today's DES chips. While means to speed up the process may be found, still it is likely that public key encryption will always be much slower than DES.

Kohnfelder (Reference 6) points up some other problems with public key systems. A reliable third party is needed for operating the public key file, he says, otherwise a penetrator could impersonate the public file function. If a participant's D key is compromised, the validity of all his messages, past and present, is questionable, particularly if it is not known when the compromise occurred. Having to continually reference the public key file would be a nuisance, so Kohnfelder suggests the use of official 'certificates' of E and N numbers to avoid the need for frequent references to the public file. Also, the public key systems seem more appropriate for data transmission than for data storage.

One possible attack of public key system is to re-encrypt the ciphertext with the public key until plaintext results. We have seen some examples of such an attack, using very short keys. Rivest (Reference 7) has analyzed the problem and concludes that for the size keys he is recommending, the likelihood of getting plaintext by this method is vanishingly small.

The value of the debate

The debate of DES has, in our opinion, played a very important role. It has validated the algorithm like no other algorithm in the public sector has been validated. And, we think, DES has stood up very well under the close scrutiny being given it.

A 64- to 128-bit key length would be much more readily accepted, we gather, by the people in the public sector who are working on the design of cryptographic systems. But even with 56-bit keys, it is clearly impractical to make an exhaustive search for even one DES key on one of today's fastest computers. And there seems to be reasonable counter-measures that a user can take to avoid the time/memory trade-off type of attack, as described by Hellman at the 1978 NCC.

We are inclined to say "Hooray for Martin Hellman" for his continuing efforts to show up the weaknesses of DES, although we suspect that IBM and NBS are not happy with all the negative publicity about DES that he has caused. But he has made a lot of other people interested in the problem. If DES can continue to stand up under this kind of scrutiny, its strength will become more and more apparent.

What about the public key systems? Are they better than DES? They are still so new that it is hard to answer this question. Since breaking the cipher by means of exhaustive search is a function of key length, and since the public key systems use much longer keys, they have a much higher exhaustive search work factor. From the standpoint of possible short-cut solutions, two points should be made. One, mathematicians have been trying to find a way to factor large numbers for over 100 years; if they find such a method, public key systems will be in trouble. And two, cryptanalysts may find a short-cut that does not involve factoring large numbers; see Reference 9 for a discussion.

Also, the public key systems are not yet ready for field operation; the whole concept of the public key file has to be developed, for instance. Further, these public key systems should be validated by making concerted efforts to break them. Such efforts require a lot of time and money, which so far have not been expended.

As we see it, DES has so far stood up very well indeed under the debate. Further, even the adherents of the public key system see it as complementing DES rather than competing with it.

A final point about the relative strength of DES. Some secret algorithms are being sold on the basis of their long key lengths (over 100 bits). But key length by itself is not enough. The algorithm may have some weaknesses that a skilled penetrator can exploit. And as long as the algorithm remains secret. the user will not know if such weaknesses exist. That is the big advantage of a public algorithm such as DES. Because of the scrutiny to which it is subjected, it provides a degree of trust that no secret algorithm can give.

Other considerations

When installing an encryption system, the user must consider other factors in addition to the strength of the algorithm and the speed at which data can be encrypted. We single out two areas for a brief discussion here: (a) the need for secondary processing, and (b) the question of key management.

Secondary processing

The U.S. National Bureau of Standards is preparing a set of guidelines for using DES, for the guidance of federal agencies (Reference 3b). In it, NBS has pointed out a number of auxiliary processing operations that must be done in support of the encryption process.

The 'encryption device' at each site would include the algorithm chip plus a micro-processor that controls the use of that chip, mounted on a printed circuit board and suitably enclosed. The device should be physically protected, perhaps by being kept in a locked 'safe' designed for the purpose.

Here are some of the secondary processing functions that NBS believes must be performed—some in the above-mentioned micro-processor and others in the host processor. The secondary processing function must monitor the power for the DES device. It must manage the control and status lines to the DES device, signalling if a parity error has been detected in the key being entered or preventing input if the device is busy. Input and output data lines should feed through the secondary processing, and it should separate the 'to be encrypted' data from the 'not to be encrypted' data and from control data. Input data to be encrypted must be formatted into 64-bit blocks. Output from the encryption process must be delivered to the proper destination (channel for storage, front-end processor for transmission, etc.).

In most instances, the address information that specifies the recipient will be transmitted in the clear, so this must be handled as 'not to be encrypted' data. The secondary processing must also be on the lookout for bit patterns in the ciphertext that are the same as system control characters; when detected, appropriate action must be taken (such as sending *two* such bit patterns in sequence as an indication of text, not control). If pre-whitening is to be done, the secondary processing would do this. If blocks have fewer than 64 bits of data in them, the secondary processing must pad them out to the 64-bits—and, at the receiving end, strip off the pre-whitening noise and the padding.

Kohnfelder (Reference 6) points out that messages should be made unique, to avoid possible duplication and to prevent the possibility of a penetrator recording messages, modifying them, and re-transmitting them later. This can be done by the use of message sequence numbers and/or the use of date-time stamps.

Of course, the secondary processing might well have to perform the 'handshaking' operations with the secondary processing at the other end. And if any physical tampering is done to the 'safe'; or if environmental changes occur (heat, power), the secondary processing should detect these, and trigger an alarm.

For an operational encryption system, then, there is need for a secondary processing capability that is intimately tied to the encryption device.

And then there is the problem of key management.

Key management

Key management includes the generation, distribution, and protection of keys. The function is needed both for conventional systems as well as for public key systems, although the overhead may be less for the latter.

We came across two good discussions of this problem: three papers in the *IBM Systems Journal* (Reference 1) and a paper by Everton (Reference 8). These two discussions are not equivalent; Everton approaches the handling of keys rather differently from the IBM authors. Since his is a bit easier to describe, and since it sets the stage well, we will begin with it.

Everton's hierarchy of keys. The functions of key management are to provide keys where they are needed and to keep them secret, says Everton. And to keep them secret, whenever a key cannot be physically protected, it should be encrypted under a higher order key.

To accomplish this protection, Everton sees a hierarchy of keys. At the top of the hierarchy are the *master keys*. They are stored in the encryption devices and are used for protecting the next lower level of keys—the *sub-master keys*. The sub-master keys are stored at the nodes of the network (that is, at the hosts, terminals, and network processors) and are used for protecting the next lower level of keys—the *session keys*. These session keys are the data-protecting keys and are almost transient in nature. They are active only as long as the data they have encrypted remains in encrypted form. For data communications, this means that the session keys will be active only for the duration of the communications session.

The master and sub-master keys are to be generated by a security officer, using a computer temporarily dedicated to that task. They should be pseudo-random numbers. The masters are left in plaintext form, since there are no higher level keys for encrypting them. The master keys are then used for encrypting the sub-master keys. Each encryption device in the network will have its own master key, says Everton, and each node will have the encrypted sub-master keys for itself and for the nodes with which it communicates.

These keys—masters and encrypted sub-masters—must then be transported to the nodes by a secure means, ***probably not via the network***. The key file for each node must be delivered to a representative of the security officer at that node, and then stored in the node's security data file. The node's master key is entered into the encryption device in plaintext form by this representative (who also keeps his copy in a very secure place). If power fails or if the device is tampered with, the key should be destroyed, accomplished by storing it in a volatile storage, says Everton. (This means that the representative of the security officer must be available for re-entering the master key. It also may argue for standby battery power for the storage of the master key; even this power would have to be disabled if the device is moved or opened.)

Session keys are produced by software at key generation points (there may be one or several such points in the network) and transmitted to the nodes at the ends of the session paths. To produce a key, a key-length pseudo-random number is generated and is designated to be encrypted under the local sub-master key. Inside the device, the key is decrypted to plaintext form; no way is available for getting the plaintext key out of the device. For transmission to the other nodes, the key is re-encrypted using the sub-master keys of the destination nodes.

This approach of Everton's is essentially transparent to the end users of the system. That is, they never have to bother with entering keys. Only the representative of the security officer, at each node, handles keys, and only under the tightest security.

The IBM approach. The approach recommended by the IBM authors (Reference 1) has a number of points of similarity with Everton's—but also a number of significant differences. Their basic structure uses one *host master key* at the host computer which serves as the apex of the network; however, they do provide for the case of multiple hosts. This thinking is in line with IBM's preference for centrally controlled networks. Further, all secondary (key-encrypting) keys and session (data-encrypting) keys are generated at the host computer "for reasons of economy" and then transported or transmitted to the outlying sites.

The host's master key is stored within the host's encryption device in plaintext form (since there is no higher key). If external power goes off, the key would not be lost because the device would be supplied by battery power. At the same time, the master key must be protected from unauthorized users, by using appropriate software or hardware safeguards.

Each of the terminals served by the host computer requires its own *terminal master key*, stored in the terminal's encryption device in plaintext form. The host also has an encrypted copy of each terminal's master key. These terminal master keys provide the means for distributing session keys.

Session keys. Assume that a terminal requests a communications session with the host, for which encryption will be used. The host, in replying to this request, generates a 64-bit pseudo-random number. This random number is considered to be the *encrypted* session key, since it was generated outside of the encryption device. When it is moved inside the device, the device (a) decrypts it to plaintext, using the host master key, and stores it for local use, and then (b) re-encrypts it, using the *terminal* master key, for transmission to the terminal. Upon receipt of the encrypted key, the terminal's encryption device decrypts it to plaintext, using the terminal's master key. Both the host's device and the terminal's device now have the same key internally, in plaintext form.

How is the terminal master key encrypted at the host? By using the host's master key? No, say the authors; this would provide a way for a penetrator to get at plaintext session keys. (See the papers for the reason this is so.) Instead, create a *variant master key*, derived within the encryption device by, say, inverting specified bits of the host master key. With DES, even a single bit change in a key will have a drastic effect on encryption or decryption. So all of the terminal master keys that are stored in the host's key file are encrypted with this variant master key. For more details, see Reference 2a.

File keys. File keys are similar to communication session keys, and are used to protect stored data. Session keys are active only for the duration of the communications session. File keys have a longer life cycle; they must be active for as long as the data encrypted by them remains in the files.

This longer life cycle of the file keys poses a problem. How should they be encrypted, for storage in the key file? Encrypting them with the host master key is not the answer, say the authors, because the host master key may have to be changed. So they propose the use of *secondary key-encrypting keys*, similar in concept to the terminal master keys mentioned above. These are protected by encrypting them with a second variant of the master key. Thus, two variants are used, one for data transmission and one for data storage, so that the compromise of one does not lead to the compromise of the other. Also, if the master has to be changed, only the secondary keys have to be re-encrypted, not all of the file keys.

Multiple host computers. Additional problems arise when the network includes two or more host computers, each with its associated group of terminals. Means are needed for establishing both session keys and file keys that allow interaction between these hosts and their terminals, but without disclosing all of one host's keys to the other. For communications, the hosts must share special keys (comparable in concept to the terminal master keys) that can be used only for sending session keys from one host's domain to the other. File keys must be designed so as to handle three cases: 1) to encrypt and recover the records only within the local host domain; 2) to allow either host domain to encrypt and recover the records in either domain; and 3) to allow the records to be encrypted in one domain but recovered only from the other domain.

Key distribution. The general rule (with one main exception) is: keys outside of an encryption device must always be in encrypted form. The one exception is that the host master and terminal master keys will be created and transported in plaintext form. They have to be entered into the encryption devices in the clear since there are no higher order keys with which to encrypt and decrypt them. Like all private keys, they must be handled in utmost secrecy.

As the discussion has indicated, session keys can be distributed via the network, since they are encrypted by using the terminal master keys. Therefore, the system is transparent to the end users, who are relieved of any responsibility for handling keys.

On the other hand, the key-protecting keys need a higher level of protection for their distribution. Perhaps the most secure means of distribution is the use of a trustworthy courier. Other, not as secure, methods include the use of registered mail or the use of person-to-person telephone calls. Security can be improved by transmitting the keys as bit patterns over two or more independent paths and then combining them at the destination by exclusive-or addition to produce the keys.

The authors also discuss problems of key generation, key entry into the encryption devices, and a number of other topics that we will not try to cover here. But we hope that we have conveyed some of the considerations involved in key management.

As this discussion has pointed out, the concepts of secondary processing and key management are still under development. Yes, there is work going on in these areas—but a set of agreed-upon 'best practices' has not yet emerged.

Therefore, new users of encryption methods should expect to have to work out a lot of these problems to fit their specific situations. This is sure to make the installation of an encryption method more difficult and costly.

Conclusions

Interest in encryption is growing and most likely will accelerate. Privacy legislation affecting the private sector, as well as regulations on trans-border data flows, may force the use of encryption.

If you *are* considering the use of encryption, we suggest that you move as slowly as possible. Start out by encrypting only the minimum amount of data needed to meet the problem at hand. Start using encryption for data transmission rather than for data storage. There is lots of learning to be done in the use of encryption. As with any powerful tool, it can hurt just as much as it can help.

Encryption is a complex, highly sophisticated subject. Buying a 'super-value encryptor' at a local store, or by mail order, probably would cause more problems than it would solve. For instance, do not just look at key length (or, as expressed in sales literature, "trillions and trillions of different possible keys") as the measure of how good the encryption system is. A 128-bit key system, say, that uses an algorithm with mathematical weaknesses, can be far less secure than a 56-bit key system with a strong algorithm.

In our opinion, we laymen are just not in a position to judge how strong a *secret* algorithm really is; the analysis is too complex. Turning to one or more cryptographic experts for their opinion on a secret algorithm is a better approach—but not a lot better, we think. Breaking a cipher can be almost like an act of genius; one person can see a mathematical flaw in the algorithm that a hundred other experts might not see.

So who do you trust, when it comes to selecting an encryption method, if you cannot trust the opinions of a few cryptographic experts? The answer that appeals to us is: select a publicly known algorithm that has stood up under intense scrutiny. The DES algorithm and the public key systems are two examples, although the latter have not yet received the scrutiny that the former has.

The DES is a good, strong algorithm; even its critics and the supporters of public key systems say that. Their main concern seems to be key length, and not so much about the key being inadequate for use today as for the future (if? when?) huge, expensive key extraction machines might be built. But, in the future, key length can be increased, if needbe, by the use of multiple encryption. This makes a penetrator's work factor *much* greater. At the same time, the continued increase in speed of the encryption chips will compensate for multiple encryptions.

Public key systems have a number of attractive features, and it seems likely that they will be adopted in some form or other, both for national security data as well as for commercial and personal data. At the present time, we see these systems as being complementary to DES rather than competitive with it.

In closing, let us say that we sincerely hope that more attention will be given to the question of how widespread use of encryption will impact society. Encryption is a powerful tool, with the ability to hide information. Like all powerful tools, it probably has some as-yet-unforeseen side effects. It would be nice to know more about those side effects before computer users become too deeply committed to encryption.

REFERENCES

1. *IBM Systems Journal* (Armonk, N.Y. 10504), Vol. 17, No. 2, 1978; three papers by IBM authors on cryptography: key management (by W.F. Ehrsam, S. M. Matyas, C. H. Meyer, and W. L. Tuchman); key generation and installation (by S. M. Matyas and C. H. Meyer); and architecture for data security (by R. E. Lennon); p. 106-150; price $1.75. (Meyer and Matyas have written a new book on encryption, *Cryptography: A new dimension in computer data security*, just being published by John Wiley and Sons.)
2. *Proceedings of 1978 National Computer Conference*, AFIPS Press (210 Summit Avenue, Montvale, N.J. 07645), price $60:
   a) Three papers on encryption, p. 1119-1134. The papers by C. H. Meyer (IBM) and Martin Hellman (Stanford University) were not actually presented at the conference; instead, substitute papers were presented; see next entry.
   b) For a cassette recording of this conference session, including the paper actually presented by Walter Tuchman (IBM) and the one by Martin Hellman, write On-the-Spot Duplicators, Inc., 7309 Fort

Hunt Road, Alexandria, Virginia 22307; ask for session T40; price $5.50 plus billing and shipping.

3. Publications of the U.S. National Bureau of Standards. They have been prepared by the Institute of Computer Science and Technology, A-247 Tech Building, National Bureau of Standards, Washington, D. C. 20234: a) *Data encryption standard*, FIPS Pub 46, January 1977; b) *Guidelines for implementing and using the NBS data encryption standard*, (in preparation); c) "Report of the workshop on estimation of significant advances in computer technology," NBSIR 76-1189, December 1976; d) "Report of the workshop on cryptography in support of computer security," NBSIR 77-1291, September 1977.
4. "Unclassified staff report summary: involvement of NSA in the development of the data encryption standard," by Select Committee on Intelligence, U.S. Senate, April 1978. Order from U.S. Government Printing Office, Washington, D.C. 20401.
5. Rivest, R.L., A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM* (1133 Avenue of the Americas, New York, N.Y. 10036); February 1978; p. 120-126; price $5.
6. Kohnfelder, L.M., "Toward a practical public-key cryptosystem," thesis for Department of Electrical Engineering, Massachusetts Institute of Technology (Cambridge, Mass. 02138); May 1978.
7. Rivest, R. L., "Remarks on a proposed cryptoanalytic attack of the 'M.I.T. public-key cryptosystem,'" *Cryptologia* (Albion College, Albion, Mich. 49224); January 1978, p. 62-65; price $16 per year.
8. Everton, J.K. "A hierarchical basis for encryption key management in a computer communications network," *Trends and applications 1978: distributed processing* (IEEE Computer Society, 5855 Naples Plaza, Long Beach, Calif. 90803); Cat. No. 170; May 1978, p.25-32; price $12.
9. Tuchman, W.L. and C.H. Meyer, "Efficacy of DES in data processing," *Proceedings of Compcon 78 Fall* (IEEE Computer Society, address above), price $20.
10. For more information on Martin Hellman's work on encryption, write Hellman Associates, 730 Alvarado Court, Stanford, Calif. 94305.

Next month, we will continue our discussion of the new computer environment for the 1980s that we foresee. We will discuss the concepts of and user experiences with two software development methodologies which, we think, represent the direction in which software development will go.

SUBJECTS COVERED BY EDP ANALYZER IN PRIOR YEARS

1975 (Volume 13)
Number
1. Progress Toward International Data Networks
2. Soon: Public Packet Switched Networks
3. The Internal Auditor and the Computer
4. Improvements in Man/Machine Interfacing
5. "Are We Doing the Right Things?"
6. "Are We Doing Things Right?"
7. "Do We Have the Right Resources?"
8. The Benefits of Standard Practices
9. Progress Toward Easier Programming
10. The New Interactive Search Systems
11. The Debate on Information Privacy: Part 1
12. The Debate on Information Privacy: Part 2

1976 (Volume 14)
Number
1. Planning for Multi-national Data Processing
2. Staff Training on the Multi-national Scene
3. Professionalism: Coming or Not?
4. Integrity and Security of Personal Data
5. APL and Decision Support Systems
6. Distributed Data Systems
7. Network Structures for Distributed Systems
8. Bringing Women into Computing Management
9. Project Management Systems
10. Distributed Systems and the End User
11. Recovery in Data Base Systems
12. Toward the Better Management of Data

1977 (Volume 15)
Number
1. The Arrival of Common Systems
2. Word Processing: Part 1
3. Word Processing: Part 2
4. Computer Message Systems
5. Computer Services for Small Sites
6. The Importance of EDP Audit and Control
7. Getting the Requirements Right
8. Managing Staff Retention and Turnover
9. Making Use of Remote Computing Services
10. The Impact of Corporate EFT
11. Using Some New Programming Techniques
12. Progress in Project Management

1978 (Volume 16)
Number
1. Installing a Data Dictionary
2. Progress in Software Engineering: Part 1
3. Progress in Software Engineering: Part 2
4. The Debate on Trans-border Data Flows
5. Planning for DBMS Conversions
6. "Personal" Computers in Business
7. Planning to Use Public Packet Networks
8. The Challenges of Distributed Systems
9. The Automated Office: Part 1
10. The Automated Office: Part 2
11. Get Ready for Major Changes
12. Data Encryption: Is It for You?

(List of subjects prior to 1975 sent upon request)

PRICE SCHEDULE

The annual subscription price for EDP ANALYZER is $48. The two year price is $88 and the three year price is $120; postpaid surface delivery to the U.S., Canada, and Mexico. (Optional air mail delivery to Canada and Mexico available at extra cost.)

Subscriptions to other countries are: One year $60, two years, $112, and three years $156. These prices include AIR MAIL postage. All prices in U.S. dollars.

Attractive binders for holding 12 issues of EDP ANALYZER are available at $6.25. Californians please add 38¢ sales tax.

Because of the continuing demand for back issues, all previous reports are available. Price: $6 each (for U.S., Canada, and Mexico), and $7 elsewhere; includes air mail postage.

Reduced rates are in effect for multiple subscriptions and for multiple copies of back issues. Please write for rates.

Subscription agency orders limited to single copy, one-, two-, and three-year subscriptions only.

Send your order and check to:
EDP ANALYZER
Subscription Office
925 Anza Avenue
Vista, California 92083
Phone: (714) 724-3233

Send editorial correspondence to:
EDP ANALYZER
Editorial Office
925 Anza Avenue
Vista, California 92083
Phone: (714) 724-5900

Name____

Company ____________________________________

Address _____________________________________

City, State, ZIP Code___________________________

____