MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORITORY

AI Working Paper 136 December 1976

CRYPTOLOGY AND DATA COMMUNICATIONS

by

Richard C. Waters

This paper is divided into two parts. The first part deals with cryptosystems and cryptanalysis. It surveys the basic information about cryptosystems and then addresses two specific questions. Are cryptosystems such as LUCIFER which are based on the ideas of Feistel and Shannon secure for all practical purposes? Is the proposed NBS standard cryptosystem secure for all practical purposes? This paper argues that the answer to the first question is "they might well be" and that the answer to the second is "no."

The second part of this paper considers how a cryptosystem can be used to provide security of data transmission in a computer environment. It discusses the two basic aspects of security: secrecy and authentication. It then describes and discusses a specific proposal by Kent of a set of protocols designed to provide security through encryption. Finally, an alternate proposal is given in order to explore some of the other design choices which could have been made.

Research reported herein was conducted at the Artificial Intelligence Laboratory, a Massachusetts Institute of Technology research program supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored by the office of Naval Research under contract N00014-75-C-0643.

I. CRYPTOSYSTEMS

The basic cryptographic situation and its associated terminology are as follows. Two people wish to communicate secretly. However, they realize that a third party (an enemy) will probably be able to overhear their communications. In order to prevent an enemy from understanding what he hears, they use a cryptosystem (cryptographic system).

When one person wishes to send a message to the other, he first uses the cryptosystem to encrypt the message. He then transmits the resulting cryptogram to the other person. This person uses the cryptosystem to decrypt the cryptogram to recreate the message. An enemy who intercepts a cryptogram is forced to attempt to cryptanalyze it. For a cryptosystem to be effective, cryptanalysis must be sufficiently difficult.

It is clear that an enemy must not have complete knowledge of the cryptosystem used. However, in general the legitimate sender and receiver must have complete knowledge of it. As a result, they must find some secure method of communicating the system to each other. Unfortunately, that method cannot be the system itself. To help alleviate this problem, cryptosystems are usually designed so that a small but vital part of the description of the encryption is segregated in a key. The intent is that an enemy who knows everything but the key will still not be able to cryptanalyze a cryptogram. It is a fundamental notion of cryptology that some information (the key) is transmitted by a physically secure method, and then used to secure the physically insecure transmission of other information at a later time.

It should be noted that the terms plaintext, encode, codetext, decode and plaintext, encipher, ciphertext, decipher are often used interchangeably with the terms message, encrypt, cryptogram, decrypt. However, they properly apply to two basic types of cryptosystems only: codes and ciphers. A code is a cryptosystem which deals with semantic units of a message. For example "bring more men" is replaced by "Q". A cipher is a cryptosystem which deals with arbitrary subunits. For example "B" is replaced by "Q". It is a point of confusion that ASCII and EBCDIC, which are prototypical ciphers, are referred to as codes.

An orthoginal destinction between two basic types of cryptosystems is the one between substitutions and transpositions. In a substitution the basic units are substituted for, but their order remains the same. In a transposition the basic units are reordered but not changed.

I.1 A PRECISE DEFINITION OF A CRYPTOSYSTEM

A cryptosystem S is composed of seven parts:

$$S = \langle M, C, K, KE, F, E, D \rangle$$

M is the set of possible messages m. C is the set of possible cryptograms c. K is the set of possible decryption keys k. KE is the set of possible encryption keys ke.

The relation $F{:}K \rightarrow KE$ maps decryption keys into corresponding encryption keys. In most cryptosystems, $K=KE$ and $F=I$ so that the encryption and decryption keys are the same.

The relation $E{:}KE \rightarrow (M \rightarrow C)$ maps encrypting keys ke into encrypting relations $e_{ke}{:}M \rightarrow C$. Each e_{ke} must be total and invertable, but need not be a deterministic function or onto. $D{:}K \rightarrow (C \rightarrow M)$ maps decrypting keys k into decrypting functions $d_k{:}C \rightarrow M$. Each d_k must be a deterministic function and onto but need not be total. E and D are related in that

$$ke = F(k) \supset D(k) = d_k = e_{ke}^{-1} = E(ke)^{-1}$$
$$m = D_{[k]}(E_{[F(k)]}(m)))$$

Often, the e_{ke} are 1 to 1 and onto in which case

$$S' = \langle M'=C,\ C'=M,\ K'=KE,\ KE'=K,\ F'=F^{-1},\ E'=D,\ D'=E\rangle$$

is also a cryptosystem.

In order to use a cryptosystem for communication, the reciever must know D and a decrypting key k. The sender must know E and the encoding key ke=F(k). The sender selects a message $m \in M$ and computs $c=E_{ke}(m)$. He transmits c to the receiver who can determine the message m by computing $d_k(c)$. Below, this process will be referred to as an encrypted communication event (ECE) which will be specified by a 4 tuple <S,m,c,k>. The use of a cryptosystem results in a stream of ECEs.

I.2 THE UNICITY DISTANCE

The unicity distance theorem [Shannon 49; Hellman 74; Hellman 75] is a basic cryptological result. Let M=C be a set of all strings over a finite alphabet. Let $R_\emptyset$ be the absolute rate of M in bits per character. Let R be the average true rate of M in bits per character.

$$R_\emptyset = \text{LOG(number of characters in the alphabet)}$$
$$R = \text{average over N of LOG((number of meaningful strings of length N)/N)}$$

(All logarithms are taken to the base 2 unless otherwise specified.) Let $R_d=R-R_\emptyset$ be the average redundancy of M in bits per character. For example in English $R_\emptyset=4.7$, R=1.6 and therefore $R_d=3.1$. Let Z be the set of every total 1 to 1 length preserving function from M to M. Here, a function is length preserving if it maps every string to a string of the same length.

Now consider the set of all cryptosystems S which can be constructed in the following way. Pick an M=C and Z as described above. Pick any subset Z' of Z. Pick sets K and Ke, a relation $F:K\rightarrow KE$ and a relation $E:K\rightarrow Z'$ which associates each element in Z' with a key in K. Consider that $H(K)=LOG(|Z'|)$ is the average information content of a key. Finally, pick a $D:K\rightarrow Z'^{-1}$ which maps keys into the appropriate inverses of elements of Z'.

Now consider the strings of length N. The functions e_k are total 1 to 1 and length preserving. Therefore, since M=C, they are onto. As a result each of the $2^{R_\emptyset N}$ strings is mapped to from $2^{H(K)}$ strings by the $2^{H(K)}$ different encryption functions. In other words, each cryptogram has $2^{H(K)}$ decryptions, but how many of these are meaningful? Averaged over all the possible subsets of Z, let $\#_{md}(N)$ be the expected number of meaningful decryptions of strings of length N.

$$\#_{md}(N) = 2^{H(K)}2^{R_\emptyset N}/2^{RN}$$
$$\#_{md}(N) = 2^{H(K)-NR_d} \qquad (1)$$

Let $N_\emptyset$, the unicity distance, be the N for which $\#_{nd}(N)=1$.

$$N_\emptyset = H(K)/R_d$$

Equation (1) can be interpreted as follows. Given a cryptogram of length N, if $N \ll N_\emptyset$ then it will have multiple meaningful decriptions with very high probability. As a result it will be impossible to cryptanalyze the cryptogram without knowledge of the key. There is no way to decide which of the meaningful decryptions is the correct one.

If $N \gg N_\emptyset$, then the cryptogram has no meaningful decryptions with a very high probability. If it is an intercepted communication then it can be expected to have one meaningful decryption. But, with very high probability there will be no other meaningful decryptions and hence no problem identifying the correct one. The exponential character of (1) causes an almost discontinuous change of behavior at $N_\emptyset$. This makes $N_\emptyset$ a good measure.

The above (which can be generalized to apply to essentially any kind of cryptosystem) points out two situations where cryptosystems are absolutely secure. First if H(K) is greater than the total content ($R_\emptyset N$) of all messages sent with the cryptosystem, then no enemy can ever collect enough data to reach the unicity distance. As a result he can never cryptanalyze the cryptograms. This system is used in practice even though the long keys (referred to as one time pads) are cumbersome to deal with.

Second, if $R_d=0$ then it is also impossible for an enemy to get enough cryptograms. However, it should be noted that an enemy can raise R_d to R_0 if he can obtain the cleartext message of a cryptogram. Due to the difficulty of perfect source coding and the dangers of a cleartext attack this approach is not used in practice.

I.3 PRACTICAL SECURITY

The greatest single lack in the field of cryptography is the derth of proofs of security. The above section says that a cryptogram is theoretically secure against a cryptanalytic attack utilizing ifinite computing power if the redundent information content of the cryptogram is less than the information content of the key used to encrypt it. This is a nice result. However, it is essentially the only theoretical result in the open literature which shows that a cryptosystem is secure in any sense. The literature is full of proposed systems and demonstrations of insecurity.

The unicity distance theorem clearly implies that only systems which utilize a one time pad are totally secure. However, all that is actually needed is practical security. Suppose a cryptosystem can be broken, but only by performing a computation consisting of a very large number of operations (say 10^{30}). If such a lower bound can be proved, then it is clear that the cryptosystem is secure for all practical purposes. One way to prove a bound of this type would be to prove that breaking the cryptosystem was equivalent to performing some computation which is known to be very difficult.

Shannon [Shannon 49] recognized the need for results of this type. Hellman [Hellman 74] discusses how such results might be precisely phrased. However, to this author's knowledge, no one has developed such a lower bound for any cryptosystem.

I.4 DESEDERATA FOR A CRYPTOSYSTEM

Throughout history, five basic features have emerged as fundamental requirements for a satisfactory cryptosystem.

1) The system should be practically secure.

2) H(K) should be small (ideally short enough to memorize).

3) The relations e_{ke} and d_k should be simple to compute (ideally only pencil and paper should be needed).

4) Error propogation should be small so that small errors in encryption, transmision, and decryption will not result in extreme garbling of the message.

5) The cryptogram should not be much longer than the message, in order to facilitate transmision.

A consensus has arizen that these five goals are incompatable. Unfortunately, in general, the situations where cryptography has been used have required features 2-5. As a consequence, in general, cryptographic systems have not been very secure.

If a cryptosystem is going to be used by a computer, then the situation is radically altered. This is particularly true of features 3 and 4. A computer based cryptosystem may be able to achieve practical security through the use of complex functions e_k which set up strong interdependencies over large areas of the cryptogram (see the discussion of LUCIFER section I.6).

I.5 CRYPTANALYTIC ATTACK AND DEFENSE

Suppose that an enemy has recorded a number of encrypted communication events (ECEs). That is to say he has the cryptograms. He wishes to discover the messages. How does he proceed? He first sorts the cryptograms into groups utilizing the same cryptosystem and key. He then tries to discover the system, if he does not already know it. Finally, he attempts to find the key. Once he knows the system and key, he can immediately decrypt all the cryptograms using that system and key.

This points out the fact that one of the most important defenses against cryptanalytic attack is compartmentalization. The fewer cryptograms created using the same system and key, the less information an enemy can gain from discovering the system-key pair and the less data he has on which to base an attack on the system-key pair. To provide compartmentalization, cryptosystems, and particularly keys, are changed as frequently as possible.

I.5.1 LEVELS OF DEFENSE

A hierarchy of defensive measures can be applied within a single compartment. These measures differ based on the amount of information the users of a cryptosystem can keep secret.

1) Steganography (eg. invisible ink, hiding micro dots in hollow nickels) is the attempt to prevent an enemy from even obtaining cryptograms.

2) A very important step is to try and prevent an enemy from obtaining any of the cleartext message corresponding to a cryptogram. If an enemy has no cleartext, he must pursue a ciphertext only attack. If he has cleartext he can use a cleartext attack. If he is able to obtain cryptograms corresponding to cleartext of his choice, he can use a chosen cleartext attack. These attacks are progressively more powerful (see the examples in section I.5.4). Most classical cryptosystems fall rapidly in the face of a cleartext attack. It should be noted that external circumstances often allow an enemy to deduce to some degree what the cleartext of a cryptogram must be.

3) Another important step is to keep the system S secret. Many classical cryptosystems can be broken by specialized cryptanalytic attacks once their nature is known.

4) The last line of defense in classical cryptosystems is the key. It must be kept secret.

As implied above, the last line of defense is ineffective in most classical cryptosystems. They can be rapidly broken with known system cleartext attacks. Computer based cryptosystems, such as LUCIFER (see section I.6), seriously attempt to be secure in the face of such attacks. However, many other computer based methods, such as binary shift register schemes and other pseudo-random number series generators, are very weak in the face of these attacks [Meyer & Tuchman 72].

I.5.2 IMPROVED SECURITY OF KEYS

Consider a cryptosystem strong enough that only the key need be kept secret. This reduces the problem of securely communicating the cryptosystem between users to the problem of securely communicating the key between users. Recently, two suggestions have been made which are designed to eliminate the need for security in the key communication process. These are public key distribution systems and public key systems.

These notions are based on the ideas of oneway functions and trapdoor functions. A oneway function F is a function whose inverse F^{-1} is significantly more difficult to compute than F (for example by the square). A trapdoor function is a function which is very difficult to compute unless a vital piece of information (a trapdoor) is available.

An effective cryptosystem has aspects of both a oneway function and a trapdoor function. Consider a known cleartext attack. Given the message and the key it is easy to compute the cryptogram. However, given the cryptogram and the message it should be very hard to compute the key. Consider a ciphertext only attack. It should be very difficult to cryptanalyze a cryptogram. However, if the key is known it is easy to decrypt it.

Pohlig and Hellman [Pohlig & Hellman 76] have suggested a possible oneway function $F(x) = \alpha^x$ MOD P where P is a prime of the form 2Q+1 (Q a prime). F(x) can be computed in O(LOG(P)) multiplications MOD P using space O(LOG(P)) bits. However, the best known algorithm for computing $F(y)^{-1} = \text{LOG}_\alpha y$ MOD P requires $O(Q^{1/2})$ operations and bits. For example if P is near 2^{400} then F requires time and space O(200) while F^{-1} requires time and space $O(2^{200})$ or about $O(10^{60})$. The only problem is that the lower bound on the complexity of F^{-1} has not been proven correct. In fact, no function has been proven to be a oneway function.

A public key distribution system is a method by which two people can give each other enough information so that they can both compute the same key, while not giving an enemy enough information to compute the key. Diffie and Hellman [Diffie & Hellman 76a; Diffie & Hellman 76c] suggest a public key distribution system based on α^x MOD P as follows. Let two people X and Y generate random numbers x and y. Person X computes α^x and tells this to person Y. Person Y computes α^y and tells it to person X. They both compute α^{xy} which they use as a key. An enemy can easily obtain α^x and α^y, but how is he to compute α^{xy} without computing a LOG_α MOD P, which is not practical to compute? Unfortunately, there is no proof that an enemy cannot avoid taking a LOG. Both of the legitimate people have a trapdoor in the function $H(\alpha^x,\alpha^y) = \alpha^{xy}$. For example person X knows x. As a result, he can compute α^{xy} easily as $(\alpha^y)^x$.

A public key system [Diffie & Hellman 76a] goes a step beyond a public key distribution system. Suppose a cryptosystem can be constructed so that the relation $F:K \rightarrow KE$ is a oneway function. For example if $ke = \alpha^k$ MOD P. In that case a potential reciever could make ke=F(k) public knowledge. Anyone could encrypt a message and send it to him, but no one could decrypt a cryptogram sent to him. If the receiver does not divulge his key then he can maintain secrecy. He does not have to trust the sender to keep anything secret. Unfortunately, no such cryptosystem has been developed as yet.

If either of the above ideas can be realized, then it will be possible to cryptographically secure information transmission without having to transmit any information by a physically secure method. This would be a fundamental change in cryptography. However, it should be stressed that oneway functions have not even been proved to exist. As a result, the above is only speculation. It is interesting to note that if it could be proven that oneway funcitons do not exist, then that would imply that practically secure cryptosystems also do not exist since a practically secure cryptosystem could be used as a oneway function.

I.5.3 METHODS OF ATTACK

Given a set of cryptograms produced by the same cryptosystem, how can a cryptanalist proceed? First he can use a brute force attack. If he knows the system he can simply try every key until he finds the correct one. This will work if he has ciphertext longer than the unicity distance. If he doesn't know the system, he can try all common systems one after the other. The way to make a cryptosystem practically secure against a brute force attack is to ensure that there are a very large number of keys, such as 10^{30} or 10^{60}. It should be stressed that the fact that a cryptosystem employs a large number of possible keys, and is therefore immune to brute force attack, in no way implies that the cryptosystem can withstand non brute force attack.

Assuming a brute force attack is not possible, a cryptanalist can use a variety of specific techniques to attempt to identify the system. In addition, he will bend every effort to obtain some cleartext. Cleartext can often be inferred from the context of the sending of the cryptogram. For example, cryptograms often mention major current events. Also, they often have stereotyped forms: headings, signatures, addresses, dates, and the like.

Having obtained as much information as he can, the cryptanalist then proceeds with specific methods which differ from system to system. The next section gives some examples of such methods. These methods fall into two basic catagories. First, many classic cryptosystems turn out not to be oneway at all, particularly in the face of a cleartext attack. Often, a straightforward procedure can be developed for breaking the system.

Second, procedures are used which yield partial information about the system or key based on information extracted from a cryptogram. This provides a beachhead for an attack. Information about the redundancy in a message is converted into information about the key. Another aspect of partial information is that with most cryptosystems there is a straightforward way to tell when one incorrect key is more correct than another incorrect key. If a key is mostly correct then the decrypted cryptogram will be mostly correct.

This idea of gaining partial information is the central notion of cryptanalysis. Cryptanalysts try to develop tests which will yield information about the key. For a given amount of computational effort, the best test is one which will elliminate half of the keys currently under consideration. The worst test is the brute force test, trying one key, which usually only elliminates that one key.

Before considering some examples of cryptanalytic attack, one radically different type of attack should be mentioned. This attack is traffic analysis. An enemy tries to gain vital information without cryptanalyzing cryptograms. Rather he uses knowledge of the time they were sent, where they originated and where they were sent to. In a military setting, this can allow an enemy to determine the location of units, and the chain of command. Further, he may be able to detect a coming offensive simply by the increase of message traffic, and the location of the offensive from the geographic distribution of the cryptograms.

I.5.4 EXAMPLES

This section discusses several specific cryptosystems in order to give the reader a feeling for how cryptanalysis actually proceeds, particularly through partial information attack. In general only the encryption process will be described in each section since the decryption process can easily be derived. All of the examples are described in terms of English text containing only A-Z. A corresponds to 0 and Z corresponds to 25. The first character of a string is taken to be character number 0. It should be noted that this section only mentions a small number of the cryptosystems which have been developed.

I.5.4.1 THE CAESAR CIPHER

In the Caesar cipher each character c_i of the cryptogram is derived from the corresponding character of the message as follows:

$c_i = (m_i + k) \text{ MOD } 26$ $\quad$ $k \in \{0\text{-}25\}$

IBM $\rightarrow$ HAL $\quad$ $k = 1$

There are only 26 keys. Therefore, the unicity distance is only LOG(26)/3.1 = 1.5 characters. A brute force attack is easy. Further, if any cleartext character is known, then $k = c_i - m_i$. This system obviously provides no security unless there is no cleartext and the enemy has no idea of the nature of the system.

1.5.4.2 THE SIMPLE SUBSTITUTION CIPHER

In a simple substitution each letter in the alphabet of the message is replaced by a different letter of the alphabet to form the cryptogram as follows.

$c_i = k_{m_i}$ $\quad$ k ϵ {permutations (ABCD ... YZ)}
BED → UVH $\quad$ k = QUFHVTAJ ... SD

There are $26! = 4*10^{26}$ keys. As a result the brute force appproach is not practical. The unicity distance is LOG(26!)/3.1 = 28.5 characters. Short messsages can actually be unbreakable. It should be noted that from the last example to this one, the number of keys went from 26 to 10^{26} but the unicity distance increased from only 1.5 to 28.5. In order to totally hide a reasonably lengthy message, the number of potential keys must be, not just large, but enormous.

Any segment of cleartext which contains each letter at least once will yield the entire key. More importantly, note that 1 cleartext letter will yield 1 letter of the key. As an even better example of a partial information attack consider the following.

English does not use each letter with equal frequency. In fact 70% of average running text is composed of the 9 letters E T A O N I S R and H, while only 7.5% of average running text is composed of the 9 letters W G B L J K Y Q and Z.

If a frequency distribution is taken of the letters in a simple substitution cryptogram, the distribution will have exactly the same shape as the frequency distribution of the message, though the identities of the letters will be different. If in the cryptogram V is most common, appearing 13% of the time, it cannot be concluded that V stands for E. However, it can be concluded with overwhelming probability that V stands for one of E T A or O, giving powerful partial information about the key. This can continue with analysis of the frequency of pairs of letters and beyond until only a few possible keys remain. At that time a brute force attack will yield the solution.

1.5.4.3 THE VIGNERE CIPHER

Here a key of length n is extended to the length of a message by concatination, and then added character for character to the message as in the Caesar cipher.

$c_i = (m_i + k_{(i \text{ MOD } n)}) \text{ MOD } 26$ $\quad$ k ϵ {(A-Z)^n}
AREAEXAM → TVWTXBSF $\quad$ k = TEST n = 4

There are $26^n = 10^{1.41n}$ keys. If n is as great as 20, brute force attack is not practical. Note, however, that if the key is chosen to be an English phrase, as in the example, then there are only $2^{RN} = 2^{1.6n} = 10^{.48n}$ possible keys. The unicity distance is LOG($10^{1.41}$)/3.1 = 1.5n characters in the strong case and only LOG($10^{.48n}$)/3.1 = .51n characters in the weak case. This shows how improper use of a cryptosystem can considerably weaken it. Here, a cryptanalist will get around 3 times as much mileage out of his data if the user uses only meaningful keys.

If cleartext is available, the Vignere cipher falls quickly. Differencing the cryptogram and text produces the key. Each letter of cleartext produces a letter of the key. If no cleartext is available, statistical tests can reveal n, the length of the key. Once n is known, the Vignere cipher can be broken by observing that the letters $\{c_i \mid i=j \bmod n\}$ for any j are encrypted with a Caesar cipher.

We saw above that a simple substitution alters the message but does not alter the frequency distribution. The Vignere substitution, changing as it does from character to character, does change the distribution. If the key contains enough randomly distributed letters, then the resulting cryptogram will have a flat frequency distribution.

However, consider the difference sequence d { $d_i = c_i - c_{i+j}$ for some j}. If j=n then d is independent of the key. This can be detected using the index of coincidence. If two random strings are differenced, the probability of a zero in any given position is 1/26=0.0385. If two sections of running English text are differenced, the probability of a zero in any given possition is 0.0667 due to the high proportion of some letters. If d is computed for a series of values of j, it will contain about 3.8% zeros except when j=0 MOD n, in which case it will contain about 6.6% zeros.

The index of coincidence is a very important tool of cryptanalysis because it applies in many more situations than the above. No matter what encrypting relations are selected by the characters of the key (as long as each relation is a function and the same relation is associated with each occurence of a character in the key) the index of coincidence can be used to detect when two segments of cryptogram were encoded using identical segments of key. Another interesting ramification comes from considering the probability of a sequence of several zeros in a difference string. This is unlikely to occur, but when it does, it corresponds with high probability to the same series of message characters encrypted with the same series of key characters.

Girsdansky [Girsdansky 72] describes an interesting attack on the Vignere cipher using suspected cleartext. Suppose that n has been established, and though no exact cleartext is known, an enemy expects that a particular phrase P occures somewhere in the message. If P is longer than n, it can be differenced, which produces a signature which must appear in d wherever P appears in m.

I.5.4.4 POLYALPHABETIC SUBSTITUTION

Suppose that each character of a Vignere like key is used to indicate a different simple substitution rather than a different Caesar cipher.

$c_i = {}^{(i \text{ MOD } n)}k\,m_i$ $\qquad {}_jk \in \{\text{permutaions of } (A \,..\, Z)\}\ 0 \le j < n$

AAA → UCU $\qquad {}_1k = \text{UQ} \,..\, \text{H} \quad {}_2k = \text{CT} \ldots \text{L} \quad n=2$

There are $26!^n = 10^{26.6n}$ keys. For any value of n brute force attack is not practical. The unicity distance is $\text{LOG}(10^{26.6n})/3.1 = 28.5n$ characters. This cipher, like the Vignere, obliterates the frequency distribution. However, the index of coincidence will still reveal n. The sets of characters $\{c_i \mid i=j \text{ MOD } n\}$ for each j can be attacked as simple substitutions to reveal the ${}_jk$s. Cleartext will make this task much easier. This cipher is quantitatively more difficult then the Vignere. Much more text is required for a solution.

I.5.4.5 PSEUDO-RANDOM KEYS

A long key is desired for security, but a short key is desired for practical use. Perhaps some of the advantages of a long key can be gotton from a short key k by using a function X to expand a short key into a long pseudo-key k'.

$k' = X(k)$

It should be noted that from the point of view of a brute force attack, and the unicity distance, such an expansion has no effect whatever. However, from the point of view of non brute force attacks, it can be helpfull by making partial information attacks more difficult.

The Vignere cipher uses one particular X, multiple concatination. It is clear that this is not a good X. It is not good because once an H(k) bit piece of k' is discovered, k can be computed and therefore the rest of k'. Most other Xs which have been proposed suffer from exactly the same problem [Meyer & Tuchman 72; Mellen 73].

For example consider pseudo-random number generators such as binary shift register schemes. They are often capable of generating a sequence of $O(2^{H(k)})$ pseudo-random numbers which can be combined into a k'. However if just 1 or 2 of these pseudo-random numbers is discovered, usually all the other numbers before and after them in the sequence can easily be calculated. Only an X which is designed to make this very difficult can provide a real increment to security.

The techniques described in the previous sections can be used to find segments of k'. If cleartext is available it is easy. The one real advantage of most pseudo-random keys comes when an enemy is forced to use a ciphertext only attack. Most ciphertext only methods require several segments of cryptogram encoded with the same segment of k'. The length of k' makes it difficult to obtain them.

I.5.4.6 SUBSTITUTIONS ON LARGE BLOCKS

Suppose that instead of substituting one character for another as the basic operation, blocks of n characters are substituted for each other. This is the same as simple substitution on a new alphabet that has 26^n characters. This implies that there are $26^n! > 10^{(26^n)}$ keys and the unicity distance is $\mathrm{LOG}(26^n!)/3.1 > 26^n = 10^{1.41n}$ characters. For values of n>8 this system offers theoretical security. However, with that many bits of key, much simpler systems also do. For values of n>3 it is not practical to handle the key or perform the encryption computation. This system has never been used for values of n over 2 or perhaps 3 in which case a cleartext attack is still powerful.

This cipher is mentioned here because of the effect it has on statistical analysis. No statistical analysis is possible except in blocks of characters which are a multiple of n. This makes this system somewhat powerful in the face of a ciphertext only attack even when n is only 2. The ciphers below retain a large block size in an attempt to foil statistical analysis while only working with a manageable subset of the possible substitutions.

I.5.4.7 TRANSPOSITON

A transposition on blocks of size n is one specific type of substitution on blocks of size n. The characters in the block are permuted but not changed.

$c_{jn+i} = m_{jn+k_i}$ $\quad$ $k \in \{\text{permutations of } (0 \ldots (n-1))\}$ $\quad i<n$

AREAEXAMS → EARXAESAM $\quad$ k = 312 n=3

There are n! keys (note $n! > 10^n$ if n > 25). The unicity distance is LOG(n!)/3.1 characters (which is > n characters if n > 25). If n>25 transposition is safe against brute force attack. Transformation leaves the single character frequency distribution unchanged but breaks up the distributions on other block sizes not =0 MOD n. However it is not strong against cleartext attack. Given the cleartext a cryptanalyst can see how the letters are rearranged except for problems with repeated characters. A chosen cleartext attack can be particularly useful in ellucidating particular details of a transposition. A particular character can be positioned in the messsage and tracked in the corresponding cryptogram.

Transpositions with a fairly long n are rather difficult to cryptanalyze with a ciphertext only attack. However, transposition keys are also hard to encrypt with, and as a result, usually only simple keys such as "write a block of 16 characters into a square by rows and then copy it out by columns" are used. This makes cryptanalysis easier. The main tool of ciphertext only transposition cryptanalysis is multiple anagramming. The analyst works on 2 or more cryptograms encrypted with the same transposition key. He guesses a part of the transposition in one cryptogram and then tests it in the others. This works because he can verify how a few of the characters move without having to know how the rest move (a partial information attack).

I.5.4.8 PRODUCT CIPHERS

What is desired is a complex cipher operating on large blocks. A transposition is a substitution on a large block size, but it is too simple. General substitutions on block sizes greater than 2 are impractical. Further, transpositions and substitutions are each closed under composition. The block size of the result is the least common multiple of the block sizes of the composands. The composition of two transformations is still too simple. The composition of two practical substitutions still operates on too small a block size. What about the composition of a transposition of block size m and a substitution of block size n?

In general, this will produce a substitution of block size LCM(m,n) which is not simply a transposition. Many ciphers have been designed where n=1 and the transposition acts on the same basic 1 character unit. In this case, it should be noted that the transpositions and substitutions commute. As a result, composing additional substitutions or transformations has no fundamental effect.

Consider an m character transformation following a 1 character substitution. There are $26!m! > 10^{MAX(26,m)}$ keys. The unicity distance is LOG(26!m!)/3.1 > MAX(26,m) characters. Brute force attack is impractical. Unfortunately, the resulting cipher is not complex enough. The two ciphers it is built up from can be attacked separately.

Single character statistics reach right through the transposition to give information about the substitution. The transposition can be directly attacked by using multiple anagramming with the appearance of normally shaped 2 and 3 letter frequency distributions as a test for success. This requires many cryptograms and/or a guess at m. Any cleartext provides simultaneous equations for elements of the two parts of the key greatly speeding the attack.

There is no question that this product cipher is considerably more complex than the ciphers above given the same size key. However it is also clear that it is not complex enough to provide practical security. In order to be able to improve the cipher by further compositions the transpositions must reach inside the substitution characters, ripping them apart. In the past, attempts to use such ciphers were not very succesful due to the complexity of applying them in practical situations.

I.6 LUCIFER

LUCIFER was designed by J. L. Smith [Smith 71; Smith, Notz & Ossek 71] closely following the ideas of H. Feistel [Feistel 70; Feistel 73; Feistel, Notz & Smith 75]. Feistel's basic idea, which follows the lines set down originally by Shannon [Shannon 49], is to create a cryptosystem CS by composing a large number of alternating substitutions S_i and transpositions T_i.

$$CS = S_1T_1S_2T_2 \cdots S_jT_j$$

Suppose the substitutions operate on blocks of n (for example 8) bits and the transpositions operate on blocks of m (for example 128) bits. CS is clearly one of the possible simple substitutions on m bit blocks and is not expressible as a substitution on smaller blocks for almost all choices of the T_is. Suppose that a key k of H(K) bits is used to select particular S_is and T_is. This creates a cryptosystem corresponding not to all of the possible substitutions on blocks of m bits, but to a subset of them.

From the point of view of a brute force attack this system is no better than any other system with $2^{H(K)}$ keys. However, it is hoped that the system will look effectively like simple substitution on m bit blocks to other types of cryptanalytic attack.

LUCIFER is a specific proposal of a cryptosystem of the above type. It operates on 128 bit blocks using a 128 bit key. It is a composition of 16 stages of substitution and permutation. All of the permutations are the same, and selected bits of the key act as a polyalphabetic cipher selecting one of two possible substitutions for bytes of the message at each stage.

The LUCIFER algorithm acts on only one half of the message at each stage. This is done so that decryption can be accomplished without having to invert the substitutions. After each stage, the two halves of the message are swapped so that the one which was not touched will get worked on next.

A basic algorithm R is iterated 16 times. It acts on three arguments m1 (the first half of the message) m2 (the second half) and key (the 128 bit key). At each stage R performs a substitution, XORs part of the key with the result, and then permutes that result. The key is permuted by a function KP at each stage so that new bits of it come into play. Each bit of the key is used once for substitution selection and is XORed into the message on 8 occasions. After the 16 rounds, the key is returned to its original form.

```
LUCIFER = R(SR)^15
          S swaps the two halves m1 and m2
          R leaves m1 unchanged,
             sets m2 = m2 xor MP(SEL2(key) xor SUB(P(m1,SEL1(key))))
             sets key = KP(key)   KP^16=I
        MP is a 64 bit permutation
     SEL2 selects 64 bits of the key
     SEL1 selects 8 bits of the key
          P swaps the halves of each of 8 bytes iff the corresponding key bit is 1
       SUB performs SUB0 on even half bytes and SUB1 on odd half bytes
             SUB0 and SUB1 are arbitrary nonlinear substitutions
             (P and SUB interact to produce one of two possible substitutions on each byte)
```

The system is designed so that decryption is accomplished by a minor change in KP. The specific functions SUB0, SUB1, SEL1, SEL2, P, MP, and KP were chosen to try and achieve three important properties.

1) Every bit of the cryptogram is a function of every bit of the message and the key. This has been proven to be true.

2) Given a particular valid message-key-cryptogram triple, holding one item fixed and changing 1 bit of either of the other two will cause each bit of the third to change with a probability of 1/2. This has been shown to be approximately true by simulation.

3) Given a message cryptogram pair there is no practical way to invert the function to get the key. No one has found a way, but no one has proven that there isn't a way.

There are $2^{128} = 3*10^{38}$ possible keys. This makes a brute force attack impractical. If English text was stored one character per byte the redundancy would be 8-1.6=6.4 bits per byte and the unicity distance would be 128/6.4 = 20 bytes.

Property (3) above prevents direct solution with a cleartext attack. The large block size makes frequency analysis impractical. Property (2) implies that an enemy will no longer be able to tell when a key he proposes is nearly correct. Suppose he has two key guesses, k_1 and k_2, corresponding to the actual key k. Let c be the cryptogram and m_1, m_2, and m be the decryptions of c under the three keys. Finally, suppose that k_1 differs from k by only 1 bit while k_2 differs from k by 64 bits.

Property (2) above predicts that m_1 and m_2 will both differ from m by 64 bits. There is no obvious way to tell which key is closer to the correct key. Note that property (2) also implies that 1 bit of error in transmission will cause the decryption to be totally garbled.

The fact that LUCIFER is immune to brute force attack is a neccessary but not sufficient condition for LUCIFER to be practically secure. It is believed that LUCIFER may well be practically secure because property (2) seems to indicate that there may not be any practical partial information attack on LUCIFER. However it must be stressed that, at least in the open literature, no one has proved property (2), or proved that it implies that no practical partial information attack is possible. The best that can be said so far is that no one has developed such an attack.

If LUCIFER is secure, it will revolutionize cryptography. Put on a chip, it could be put in a calculator like package and used anywhere. It would obsolete other methods of cryptography. It would also frustrate organizations such as the National Security Agency (NSA) which spends a great deal of effort cryptanalyzing intercepted communications. For centuries cryptanalysis as been one of the most effective and productive tools of espionage.

I.7 THE NBS STANDARD

The National Bureau of Standards (NBS) has proposed a standard cryptosystem of the type discussed above [NBS 75]. It differs from LUCIFER in that it uses a 56 bit key (64 bits with the parity bits ignored) and works on a 64 bit block. It also differs in a variety of small details though it still operates in basically the same manner consisting of 16 rounds of substitution, permutation, and swapping of halves. No study in the open literature comments on whether the small changes make the NBS system weaker or stronger than a 64 bit LUCIFER.

In any case, there is a glaring weakness in the NBS algorithm [Diffie & Hellman 76b]. There are only $2^{56} = 10^{17}$ keys! This is not large enough to resist a brute force attack. A special purpose machine which could try 10^6 keys in parallel in one microsecond could try every key in 10^5 seconds = 1 day. With current technology a chip which could try 1 key in 1 microsecond would only cost $10 in quantity. Allowing a factor of 2 in cost for control circuitry, the special purpose machine above would only cost $20,000,000 or $10,000 a day depreciated over 5 years. At this rate the average solution (1/2 a day) would only cost $5,000. Worse than that, if the cost of hardware continues to drop an order of magnitude every 5 years, such a solution would only cost $50 in 1986. Even the more costly current prices are well within the means of a large organization such as the NSA.

If the key were increased to 128 bits, then a brute force attack would become thermodynamically and quantummechanically impractical. Since this increase in key length would not cost very much money, there is no need to use a weak standard.

There is a deeper problem. Like LUCIFER, no one has publicly shown that the NBS algorithm can withstand non brute force attack. It is believed [Diffie & Hellman 76b p. 8] that IBM, NSA and NBS have convinced themselves that it can but none of their studies has been declassified. It seems foolhardy to accept a national standard that may have a trapdoor in it. In any case the NBS system simply does not have enough keys to resist brute force cleartext attack.

II. SECURITY OF DATA COMMUNICATION VIA CRYPTOGRAPHY

There are two aspects of security in a data communication link, secrecy and authentication. Secrecy deals with preventing an enemy from discovering what data is being transmitted. Authentication enables the receiver to detect any alteration of the data stream by an enemy.

II.1 SECRECY

A cryptosystem can be used to encrypt data transmissions. If the cryptosystem can resist cryptanalytic attack, and is properly used, then it will provide secrecy. (The issue of the strength of cryptosystems was discussed above.)

There are three general levels of secrecy recognized [Schmid 76]. The first, total secrecy, requires that an enemy not be able to detect that a particular user has ever been in communication with a given host computer. A spy requires total secrecy in his communications. Total secrecy involves phyisical precautions and will not be discussed here.

The second type of secrecy, traffic flow secrecy, allows an enemy to identify the users who communicate with a particular host computer. However, it requires that he not be able to tell at any given moment whether or not a cryptogram is being transmitted between a given user and the host. This type of secrecy is particularly important in military situations.

The third type of secrecy, message secrecy, requires that, though an enemy can identify the cryptograms being sent, he not be able to cryptanalyze them.

II.2 AUTHENTICATION

In order to authenticate a stream of cryptograms, a receiver must be able to show that each cryptogram was prepared by the expected sender (identity authentication). In addition, he must be able to show that each cryptogram he receives is in sequence (sequence authentication). A given cryptogram is in sequence if it arrives at the reciever immediatly after the cryptogram it was created immediatly after.

In order to authenticate a given cryptogram, there must be redundancy in the cryptogram. This stands in sharp contrast to the fact that redundancy in a message is a cryptanalytic weakness. The weakest attack an enemy can mount is to put a randomly generated cryptogram in the channel. In order to make the probability that this spurious cryptogram will be accepted by the reciever less than some P, it must be the case that only 1 out of P^{-1} cryptograms are acceptable. In other words, there must be -LOG(P) bits of redundancy in each legitimate cryptogram.

It is clear that P cannot be equal to zero while any communication is going on. As a result, "authentication" below will mean authentication at some level P where P is very small (10^{-10} or 10^{-30}).

It should be noted that even though secrecy has classically been considered the prime function of cryptography, authentication has always been important. The natural redundancy of the language of the message has been used for authentication. This redundancy is high enough that it is almost impossible for an enemy to generate a cryptogram which will not decrypt into gibberish unless he has broken the cryptosystem.

Unfortunatly, natural redundancy is not very useful in a computer application for two reasons. First, it is difficult for a computer to recognize this type of redundancy. Second, much of the data transmitted does not have high natural redundancy. As a result, explicit redundancy bits must be used.

In order to achieve authentication at level P with only -LOG(P) bits of redundacy, an enemy must not have any practical method available to him for generating acceptable cryptograms better than random generation. It is clear that the bits used for authentication must be so interrelated with the bits representing the message that it is impossible to change one without having to change the other. Further, this relationship must be sufficiently random looking so that an enemy has no practical method of deciding how to change one when he changes the other.

Two basic methods of authentication are used [Widmer 76]. The first method can be used whether or not the basic transmission to be authenticated is a cryptogram. In this method, an authenticater tag is attached to the transmission. This tag is composed of -LOG(P) bits of an encrypted form of the transmission.

The other basic method is applicable only when the transmission to be authenticated is a cryptogram. In this method a stereotyped tag is added to the basic message before it is encrypted. This method is only secure when each bit of the resulting cryptogram is a function of every bit of the message and tag.

As stated, both of the above methods only provide identity authentication. Sequence authentication can be provided by introducing some time varying aspect to the authentication scheme. This can be most easily done by making the encryption key or tag field a function of the time or of prior messages sent.

II.3 KENT'S PROTOCOLS

In order to provide security in a physically unsecure data channel, it is not sufficient merely to encrypt the data stream which would have been transmitted over a physically secure channel. In addition, protocols must be developed for initiating a session of encrypted data flow, authenticating the flow, and recovering from error situations. Many people have addressed aspects of this problem [Branstad 75; Feistel 73; Feistel, Notz, & Smith 75].

Kent [Kent 76] has proposed a complete set of protocols which are summarized below. He deals primarily with a situation where encryption is used to secure communication between a user at a remote terminal and a host computer. At each end of the physically unsecure link, he places a protection module (PM).

In his proposal, each PM is capable of block encryption (such as the NBS algorithm) and has three internal registers. One holds the current key, and the other two (R and T) hold counters which record the number of cryptograms recieved (R) and transmitted (T) since the time of the last key change. Each message a PM encrypts and sends has a field which identifies the message type and a tag field which, in general, contains the value of the PMs T counter and a bit identifying the PM. The level of authentication is determined by the number of bits in the tag field.

For ordinary data transmission, a PM waits untill it has a block full of data characters. It then packages them together with the value of the T counter, encrypts the result, and sends the cryptogram to the other PM while incrementing the T counter. Upon decrypting a data message, a PM compares the tag field with the value of its R counter. It then increments the R counter and accepts the message if and only if the tag field matched. A message will fail to match if it is garbled in transmission, illegitimate, or out of sequence.

Whenever a PM fails to authenticate a message it tries to resynchronize the counters in the PMs. A resynchronization protocol is also initiated from time to time in order to test for interruption of the data link. To attempt resynchronization one PM (PM1) will send a message to the other with R1 as its data field. If the other PM (PM2) can authenticate this message it sends a reply message specifying T2. An accompaning algorithm resets R1 and T2 to resynchronized values. This protocol will break down if T1 and R2 are not synchronized, or if the data channel is not open, because PM2 will not recieve an authenticateable message from PM1.

If PM1 does not recieve a reply from PM2 within a reasonable time, it initiates a key change protocol. In order to change the key, one PM generates a new key at random and sends it to the other encrypted under the current key with tag field zero. (If the new key is too long to fit in one cryptogram, then several cryptograms will have to be used.) Both PMs then switch to the new key and reset their counters to zero. The second PM then sends a message to the first acknowledging the key change. The tag field of zero allows the protocol to work even if all the counters are unsynchronized.

A secure data transmission session is initiated by a login protocol as follows. First, a user signals his desire to login at a terminal. Then the host identifies itself, and the user types in an identifier. These first three messages are in cleartext.

At this point, the user gives his personal encryption key to the PM at the terminal. The host looks up the users encryption key and gives it to the PM at the host. Both PMs then go into encryption mode, and the host PM initiates a key change protocol. The reply of the user's PM to the key change protocol shows the host PM, through knowledge of the randomly generated key, that it is not talking to a recording. The host then sends a message containing the data and time in order to authenticate itself to the user. Knowledge of the user's private key serves as the users password, and as identity authentication for both PMs.

II.4 ANALYSIS OF PROBLIMATICAL ASPECTS OF KENT'S PROPOSAL

This section focuses on the use of Kent's protocols to secure a data link, and asks four questions. How transparent is his system? How transparent is it possible for a system to be? How well does his system recover from error situations? Are there gaps in the security provided by his protocols?

II.4.1 TRANSPARENCY

There are two levels of transparency: logical and physical. A logically transparent data link security method can be added to a link without making any other change to the overall system. A physically transparent method can be added without there being any experiments which a user can run which will let him know whether or not the method is in use.

An encryption based data link security method cannot be physically transparent unless it includes a hardware modification which increases the baud rate of the link. This is because the authentication bits cause two kinds of time effects. First, the fact that these bits must be transmitted reduces the effective bandwidth of the link. Second, delay is introduced because authentication bits for a piece of information must be calculated after the piece of information is ready for transmission, and must be recieved before the piece of information can be authenticated and used.

There is a trade off between these two problems. Suppose that 10 bits of authentication are desired for each bit of data. The delay can be minimized by sending 10 authenticastion bits with every data bit. In this case, each bit is delayed 10 bits. However, 90% of the bandwidth of the channel is waisted. Effective bandwidth can be maximized, at the expence of delay, by having each authenticator bit apply to many data bits. For example suppose 90 bit blocks are created and sent with 10 bits of authentication. Only 10% of the bandwidth is waisted, but some data bits are delayed by 100 bits.

Logical transparency, on the other hand is achievable. Kent apparently did not have transparency as a major goal because his protocols fail to be logically transparent in two major ways. First, the way he handles blocking of characters for transmission requires some change in the external system to ensure that characters get transmitted when they need to be, even if a block is not full. For instance, when a user finishes typing a command his it must be transmited. This problem is further complicated by any echoing. Kent suggests using a local echoing mudule at the terminal to decide when to pad out blocks, and to do all ordinary echoing. This has the additional advantage of counteracting some of the physical delay caused by authentication, and of reducing the amount of data which must be transmitted from the host to the terminal. However a local echoer is a large change in the external system, and might not work very well with a host program, such as a real time editor, which does not do very much ordinary echoing.

Second, the user of Kent's protocols has to become directly involved with the encryption process through his use of the encryption key. The encryption key is actually only used as the user's password, and as part of a secure key distribution protocol (see below). If some other secure key distribution protocol can be found, then it is a moot question whether the user needs to have a password which is as cumbersome as an encryption key has to be. In any case, the introduction of Kent's protocols into a system requires a change in the password procedure and is therefore not logically transparent.

II.4.2 ERROR RECOVERY

Kent provides two protocols for achieving resynchronization of the authentication counters after an error. However, his protocols do not attempt to correct errors. This is left to the external system. The PMs should be able to use retransmission to correct natural errors and recover from enemy influence without having to call on the external system.

In addition, it seems reasonable that error correcting codes should be used to deal with natural errors if they are at all likely since resynchronization and retransmission are time consuming processes. (This is apparently Kent's intention though it is not stressed in his paper.)

II.4.3 SECURITY

There are two situations where Kent's protocols do not provide automatic sequence authentication. There is no automatic sequence authentication of the host in the login protocol, which means that there is no automantic verification that the host cryptograms are not a recording. The user is required to perform this authentication by reading the login message. The key change protocol also lacks sequence authentication since the tag field is predictably zero. As a result, there is no automatic detection of messages lost immediately prior to a key change.

Even assuming, as Kent does, that the block cryptosystem used can resist any cryptanalytic attack, compartmentalization is still advisable. Kent's key change protocol is a step in the direction of compartmentalization. However, it provides no compartmentalization in the forward direction. If an enemy has discovered the old key he can easily find out the new key since it is encrypted under the old key. In Kent's system true compartmentalization is provided by the change of users. This causes a switch to a new primary key. Which is used to securely distribute a new encryption key.

The PMs could achieve compartmentalization without reference to the primary key through the use of a secure key communication protocol. For example, they could refer to a one time pad. Alternately, they could use a public key distribution system (when and if one is developed) such as the proposal based on the oneway function α^X MOD P discussed in section I.5.2. Both methods provide excellent authentication.

An additional advantage of having the PMs handle the keys themselves is that it enhances key security. There need never be any knowledge of the keys outside the PMs. In Kent's proposol, the existance of the user's private key in a list in the host computer is a security weakness.

The host still has to be able to authenticate the identity of a user. This can be done by having the host remember a oneway function (when and if one is developed) of the user's password instead of the password itself [Evans 74 & Purdy 74]. When the user presents his password the host computes the oneway function of it for comparison, and does not have to remember anything which it has to hide. This solution could be applied in Kent's proposal as it stands so that the host wouldn't actually have to remember the value of the user's key between sessions. However, the login protocol would have to be changed in order to provide authentication of the host (which is currently provided through the host's knowledge of the key).

A completely separate issue is that of traffic flow security. If the data link is dedicated, then nothing is lost by transmitting continuously at full rate. All the gaps in the normal transmission can be filled with pseudo-random noise. If the resulting stream is random enough looking, then it will not be possible for an enemy to pick the real cryptograms out of the noise. The elimination of identifiable login protocols is also important. Kent chose not to restrict himself to dedicated data links, and therefore chose not to try and provide for traffic flow security.

II.5 AN ALTERNATE PROPOSAL

This section proposes an alternate set of protocols in order to explore some of the other design choices which could be made. The main goal of this alternate proposal is to produce a system which is as logically transparent as possible to the user and host. The only logical difference seen by them is that the data link is able to give an active indication if communication is blocked.

In this proposal, the two PMs communicate with each other via three protocols: normal data transmission, retransmission, and key change. An algorithm is used to convert a block cipher into a time varying stream cipher. (A somewhat similar method is described in [Branstad 75].) This stream cipher is used to continuously transmit real and null data in order to provide traffic flow security and detection of denial of service. The time varying nature of the cipher provides authentication. One major weakness in this proposal is the lack of any proof of the security of the stream cipher. The stream cipher is produced as specified in the following program:

```
         SEED' = block_cipher(SEED+CHAR,KEY)
           TAG = select1(SEED')
ENCRYPTED_CHAR = TAG || (CHAR xor select2(SEED))
          SEED = SEED'
```

The decryption process is similar. Select1 picks L bits from the seed for authentication. Select2 picks R_B bits of the seed to use for encrypting the character. It can be seen that the procedure generates a pseudo-random sequence of seeds (dependent upon the character stream, the initial seed and the key) which is used to encrypt the character stream.

To demonstrate that this is a powerful method, one would have to demonstrate the following properties. There should be no practical way to derive the key from a sequence of seed values. Further, there should be no practical way to derive the selected bits of the sequence of seeds without knowing the key. These two properties would guarantee secrecy.

It should be the case that once two sequences diverge, due to a difference in the characters encrypted, there should be no two streams of characters capable of resynchronization in a practical length of time. Further, since the tag bits are clearly identifiable, they should be so interrelated with the character that there is no practical method for changing either, or both, without invalidating the encrypted character. These two properties would guarantee authentication.

Consider an example of how the authentication process would work with this stream cipher. Suppose that the LUCIFER algorithm was being used, in which case both the seed and key would be 128 bit numbers. Further, suppose that the tag was 2 bits (L=2) and the characters 8 bits (R_0=8). Finally suppose that an illegitimate encrypted character is entered into the data channel.

This will cause the seed in the receiving PM to get out of synchronization with the seed in the sending PM. The illegitimate character itself will be accepted with a propability $2^{-L}=2^{-2}$. This is not very effective, however, consider the situation after N more charaters have been recieved. The sequence of seeds will be out of synchrony with these characters even if they are sent by the valid sender. As a result, the probability of all these characters being accepted is only $2^{-LN}=2^{-2N}$. This is limited by 2^{-128}, which is the random probability that the seed will get back in synchrony.

Suppose authentication at a level of 10^{-10} or roughly 2^{-32} is desired. In a block cypher this would require 32 bits in each block (1/4 of LUCIFER's block and 1/2 of NBS's block). In the example, this level of authentication exists for all the characters 16 characters old or older (at a cost of 1/5 of the bits sent). Combined with the fact that the stream cypher is continuously transmitting, this means that one has only to wait a moment before using a character if authentication is required.

The normal data transmission protocol goes as follows. The cleartext stream is precoded in order to allow two reserved null characters. The resulting characters are then encrypted and sent with the stream cipher system. Encrypted nulls are sent whenever there is no text character to send. Error correcting codes are used on the encrypted characters in order to correct most natural errors. The reciever can detect any alteration of the encrypted data stream within a reasonable time of its occurence. In addition, it can readily detect any interruption of data transmission.

In order to provide authentication, the receiving PM has to buffer up the last N characters before sending them on to the external system. This introduces the time delay which which is required by authentication. Deadlock cannot develop because even if the user stops typing, the sending PM continues to transmit nulls pushing the last data characters out of the authentication buffer.

Another important observation is that much of the time the authentication does not have to cause an N (in the example 16) character delay. This is because of the large number of nulls that the PMs are sending. This highly predictable behavior can be used for additional authentication because there is no practical method for an enemy to mimic a high density of nulls.

In the example, the probability of three nulls in row is only about 2^{-30} which provides as much authentication as 15 data characters. This greatly reduces the time delay when data transmission is sparse.

Whenever a receiver detects a problem it initiates a retransmission protocol. In this protocol the receiver requests that the sender retransmit the last N characters it sent. N is chosen to give the required level of security. In the example N is 16.

The sender will retransmit the characters, proceeded by an acknowledgement message. The receiver will compare them with the original characters it received, and correct its buffer as needed. In order to make this work, the sender must also remember at least the last N characters it has dealt with.

The retransmission message (and the response message which proceeds the retransmitted characters) are designed to work and resynchronize the seeds no matter how unsynchronized the seeds may be. To this end, the messages are recognized by a cleartext heading (for example 4 zero bytes). During the normal transmission mode, the senders utilize the two different nulls to insure that this header never appears. The cleartext heading allows an enemy to detect the messages. However, they carry no particular information for him since they occur at random intervals or in response to his actions.

The body of the retransmit message is block encoded under the current key, and contains the value of the seed at the point where retransmission should start. This seed is determined by the receiver by backing up its seed generator until a seed is reached which is authenticated at a high level as being one used by the sender at some time in the past. In the example this requires backing up over the last 16 characters received. The sender backs up over the characters it sent until it reaches the same seed. Note that this works even if the sender and receiver disagree on what the last several characters transmitted were.

A key change protocol is applied whenever retransmission fails to resynchronize the seeds. It is also applied from time to time in order to provide compartmentalization by switching to a new key and seeds. Any secure key distribution system can be used for this protocol. Login is completely independent of the protocols proposed here. It goes on in encrypted form as part of the normal data traffic.

BIBLIOGRAPHY

Branstad, D.K. (1975) "Guidelines for Iplementing & Using the NBS Encryption Standard" draft of Nov. 1975, to be published by NBS.

Diffie, W. & Hellman, M.E. (1976a) "Multiuser Cryptographic Techniques" AFIPS conf. Proc. v45, 1976, pp. 109-112.

Diffie, W. & Hellman, M.E. (1976b) "Cryptanalysis of the NBS Data Encryption Standard" submitted to IEEE spectrum, May 1976.

Diffie, W. & Hellman, M.E. (1976c) "New Directions in Cryptography" to appear in Nov. 1976 IEEE Transactions on Info. Theory.

Evans, A., Kantrowitz, W. & Weiss E. (1974) "A User Authentication Scheme not Requiring Secrecy in the Computer" CACM v17 #8, Aug 1974.

Feistel, H. (1970) "Cryptographic Coding for Data-Bank Privacy" IBM research report RC-2827, March 1970.

Feistel, H. (1973) "Cryptography and Computer Privacy" Scientific American, May 1973.

Feistel, H., Notz, W.A., & Smith, J.L. (1975) "Some Cryptographic Techniques for Machine-to-Machine Data Communications" Proc. IEEE v63 #11, Nov 1975.

Girsdansky, M.B. (1972) "Cryptology, the Computer, and Data Privacy" Computers and Automation, April 1972.

Hellman, M.E. (1974) "The Information Theoretic Approach to Cryptography" Stanford Univ Information Systems Lab publication, April 1974.

Hellman, M.E. (1975) "The Shannon Theory Approach to Cryptography" Submitted to IEEE Trans on Info. Theory, Octy 1975.

Kahn, D. (1967) The Code Breakers the Story of Secret Writing MacMillan co. N.Y., 1967.

Kent, S.T. (1976) "Encryption-Based Protection Protocols for Interactive User-Computer Communication" MIT/LCS/TR-162, May 1976.

Mellen, G.E. (1973) "Cryptology Computers and Common Sense" AFIPS Proc. v42, 1973.

Meyer, C.H. & Tuchman W.L. (1972) "Pseudo-Random Codes can be Cracked" Electronic Design #23, Nov. 9 1972, pp. 74-76.

NBS (1975) "Computer Data Protecting NBS Standard Proposed" Federal Register v40 #52, March 17, 1975, pp. 12067-12250.

Purdy, G.B. (1974) "A High Security Log-in Procedure" CACM v17 #8, Aug. 1974.

Pohlig, S.C. & Hellman, M.E. (1976) "An Improved Algorithm for Computing Logarithms Over GF(P) and Its Cryptographic Significance" Submitted to IEEE Transactions on Info. Theory, June 1976.

Schmid, P.E. (1976) "Review of Ciphering Methods to Achieve Communications Security in Digital Data Transmission" Proc. 1976 Int. Zurich Seminar on Digital Communications, March 1976.

Shannon, C.E. (1949) "Communication Theory of Secrecy Systems" BSTJ v28, Oct. 1949, pp. 656-715.

Smith, J.L. (1971) "The Design of LUCIFER, a Cryptographic Device for Data Communications" IBM Research Report RC-3326, April 1971.

Widmer, W.R. (1976) "Message Authentication, a Special Identification Requirment in Oneway Digital Data Transmission" Proc. 1976 Int. Zurich seminar on Digital Communications, March 1976.