Dec. 7, 1975 Ralph C. Merkle

SECURE COMMUNICATIONS OVER INSECURE CHANNELS

People have been communicating with each other for many millenia. Often they wish to communicate privately. Since the first use of Caesar's cipher, some two millenia ago, people have employed a number of ciphers and codes in attempts to keep their correspondence private. These have met with varying degrees of success, up until the modern age. With the advent of the modern digital computer, it has proven possible to create ciphers which are, in practical terms, unbreakable. (At least, if anyone has broken them, they are maintaing a discrete silenee.) Underlying this success has been a very definite paradigm, which makes very definite assumptions about the nature of the encryption process, and the conditions under/which private communications can take place. It is the purpose of this paper to consider this paradigm, and to explicitly state assumptions which are implicit in it. Once these assumptions have been made explicit, it is possible to question them. It so happens that one of the assumptions which is currently regarded as a necassary precondition for cryptographically secure communications, does not, in fact, hold. This is demonstrated by exhibiting a partial solution. I say the solution is partial only because there is no reason to believe that there is not a better solution. The partial solution, however, is logically complete in its own right, and appears /to have practical applications. The body of the paper will begin with an explanation of the paradigm currently in use, and will develop from there.

Communication takes place between two individuals. Secure communications takes place between two individuals, in the face of efforts by a third to learn what was communicated. (If communication takes place between more than two individuals, we shall simply regard that situation as the composite of several conversations between two individuals.) We shall, therefore, introduce three protagonists into our paradigm, A and B, the two communicants, and E, the enemy, who wishes to find out what A and B are communicating. A and B have available some method of encrypting messages for each other. It is a cardinal rule of cryptography that we must assume that the general method of encryption is known to the enemy. Thus, A, B, and E all know the general method for encrypting and decrypting messages. A and B, however, also know the key. This key is not known to E, and E is unable to deduce it, even with a fairly large number of messages. Thus, the security of the method relies on the fact that E does not know the key being used, and cannot deduce it. These observations are not knew, and have appeared many times before. The following observations, while obvious, are not usually stated explicitly.

According to the paradigm, A and B can communicate securely because they know the key, and E does not. How did this situation come about? Clearly, either A transmitted the key to B, or B transmitted the key to A, or some third party, that A and B both trust, transmitted the key to both of them. We shall ignore the third possibility, for it contributes nothing essentially new. The following logic could be extended to include this case without undue difficulty.

In view of this need to transmit the key from A to B, or vice-versa, we will extend the paradigm by creating a key-channel, over which key information is to be sent, and will distinguish it from the normal channel, over which the bulk of communications is to take place. While the normal channel need satisfy no particular property, save only the practical one that some of the messages do, indeed, get through, the key channel must satisfy a very restrictive set of properties. In particular, it would appear that the key channel must satisfy the following two properties:

1.) E is unable to modify or alter messages, and is unable to inject false messages into the key channel.

2.) E is unable to determine the content of any message sent over the key channel, i.e., E cannot intercept the messages.

If such a key channel is available, then A and B can communicate securely by sending the key over it, and then encrypting all further messages and sending them over the normal channel. The reason that the key channel is not used for normal communications is because of its expense. If the reader will pause a moment, and consider over what communication media it is safe to transmit a key, he will realize the severity of this restriction. Radio communications, in any form, are eliminated. Long distance telephone is eliminated, for almost all long distance calls are routed over microwave communication links. The local telephone can be tapped. Even word of mouth is unsafe, unless special precautions are taken to eliminate bugs. We might use the mails, if we could mail a small safe, or perhaps we could mail the information on an undeveloped piece of film. Safes can be broken, though, and someone might develop the film, and create a copy of it that was undeveloped. Perhaps the only safe method is to send a trusted courier, with an attache case chained to his wrist. This, of course, requires that you trust the courier. Whatever the method used, if E should manage to discover the key by "practical cryptanalysis", then A and B might very well continue in blissful ignorance of the fact.

The first assumption, that E cannot modify messages, appears to be a self-evident truism. If E could modify the messages, then E could imitate the actions of B while talking with A, and likewise E could imitate the actions of A while talking with B.

The second assumption, too, appears to be a self-evident truism. No matter what A transmits to B, B must respond to it in some fashion. However B responds, i.e., whatever program governs B's behavior, E can imitate it. Thus, it would appear that both assumptions are necassary prerequisites for secure communications.

It is the thesis of this paper that the second assumption is unnecessary, i.e., that secure communications can take place when E is able to read all messages sent on the key channel.

Before continuing with the solution, it would be wise to restate the problem in more precise terms. In particular, we shall assume that A and B are both computers. They are connected by two communications channels, a normal channel, and a key channel. The key channel satisfies assumption 1, but does not satisfy assumption 2. In fact, we assume that E can receive all messages sent over the key channel with no noise or distortion at all. E can read the messages at least as well as A and B. (Naturally, E can also read the normal channel

We also desire that E know the "method" being used. We shall simplify this requirement to the following: E knows <u>everything</u> known to A or B before we attempt to communicate securely. This can be visualized in the following way: a dump of the current states of A and B is made onto a magnetic tape. This tape is given to E. After the dump has been made, A and B can communicate only over the key channel and the normal channel. In addition, no other agency communicates with either A or B. Thus, the initial states of A and B are both known to E. However, E is unable to obtain further information about A and B, except what they transmit over the key channel and the normal channel.

The reader might wonder why E would wish to listen to further communications. After all, E knows everything there is to know about A and B. The answer is, of course, that for the sake of the problem, we have selected the most difficult set of circumstances imaginable. Also, even if E knows everything available to A and B now, A and B might wish to determine, by chance, which one of several already considered strategies they wish to follow. This situation arises in game theory, where the correct strategy is selected according to a predetermined weighting pattern.

To complete our description of the problem, we must consider what constitues a solution. If A and B eventually agree upon a key, and if the work required of E to determine the key is much higher than the work put in by either A or B to select the key, then we have a solution. Note that, in theory at least, E can determine the key used in most current methods simply by trying all possible keys and seeing which one produces a legible message. However, this means that E must put in an amount of work that is exponentially larger than the amount of work put in by A or B. As mentioned earlier, the current solution is "partial." It is partial in the sense that the amount of work required of E to determine the key will increase as the square of the amount of work put in by A and B to select the key. Clearly, it would be desirable to find a solution in which the amount of work put in by E increased exponentially as a function of the amount of work put in by A and B. This has not proven, (so far!) to be possible.

We now continue our attempt to solve the problem. E knows the current state of both A and B. If A and B are deterministic devices, then E can predict, given the inputs, i.e., the information on the normal channel and the key channel, all future states of A and B. The conclusion we reach is that neither A nor B is deterministic, both must be, in some sense, random. The most graphic way of illustrating this is by assuming that both A and B have a new input device: a geiger counter. From time to time, A and B will issue a command, LDFG, Load Accumulator From Geiger counter. While E will be able to simulate the activity of either A or B up to the time that this command is first issued, after that, E will not know the correct contents to place in the accumulator, and as the program takes further actions, based on the random information, E will become increasingly muddled about exactly what is going on.

We now assume that A (or B), selects a key and attempts to transmit it to the other computer. No matter what encoding technique is used, no matter what clever use is made of random information, if A tries to transmit the key to B, then B, a physically realizable device, must be able to determine what that key is. E can now simulate B's actions, and will be able to determine the key, even though B is behaving in a random fashion. The point is, that no matter <u>what</u> random information B uses, B must still be able to determine the key. Therefore, <u>any</u> choice of random information will result in B's determining what the key is. The conclusion we reach is that neither A nor B attempts to transmit the key to the other. They must both participate in a process of key selection. Intuitively, we can justify this decision by the following consideration. There is some random information known to A, and some random information known to B. A must guess at B's random information, and B must guess at A's random information. E however, must guess at the random information of both A and B. Thus, E has a harder problem than that of either A or B.

With the foregoing explanations as partial motivation, we will now consider the method of solution. A randomly generates some information. This information is then divided into N seperate pieces, and is encrypted into N seperate messages. The encryption used is deliberately weak, and each message, or "puzzle", deliberately provides enough redundant information so that it can, with some effort, be broken. Thus, A has generated a series of N puzzles, each of which requires a carefully calculated amount of effort to break. Once a puzzle has been broken, it contains two pieces of information, which can be summarized as "I am puzzle number ######", and "Use key ######". B, upon receiving these N seperate puzzles, discards all but one of them. This one puzzle, selected at random by B, can, with an expenditure of a fixed effort, be broken by B. B then finds himself in possession of two pieces of information. The puzzle number, and the key. Neither piece of information is available to E, unless E should chance to break exactly the same puzzle that B has selected. E however, does not know which puzzle B selected, for B selected the puzzle at random. Thus, E must break all, (or, on an average, $\frac{1}{2}$) of the puzzles that A sent to B before E, too, can acquire the information that is known to B.

Once B has the two pieces of information, it can then use the key which it found in the puzzle to encrypt messages over the normal channel, and can transmit the puzzle number, in the clear, over the key channel. The puzzle number will not convey any information to E, for the puzzles were numbered in some random fashion. However, A, which generated all the puzzles in the first place, has a record of what key value was associated with what puzzle number. Thus, A will also know what key to use to encrypt and decrypt future messages.

To summarize the above outline of the method: A creates a "menu" of N puzzles. Each puzzle contains, within itself, two pieces of information. Neither piece of information is readily available to anyone examining the puzzle. By devoting a carefully calibrated amount of effort to breaking the puzzle, it is possible to determine both these pieces of information. One piece of information is a puzzle number, which uniquely identifies the puzzle. The puzzle numbers were assigned by A at random. The other piece of information is a key. The key was also selected by A at random. A transmits these N puzzles to B over the key channel.

When B is presented with this menu of puzzles, he can select any one he pleases. B selects a puzzle at random. B then spends the amount of effort required to break the puzzle. B then transmits the puzzle id back to A over the key channel, and uses the key found in the puzzle as the key for further encrypted communications over the normal channel.

If E desires to determine the key which A and B are using, then E must solve all N puzzles. On an average, E will have to solve $\frac{1}{2}$N puzzles before reaching the puzzle that B solved. If each puzzle has been carefully constructed so that it requires N units of effort to break, then E must spend $\frac{1}{2}$N*N units of effort to determine ~~to determine~~ the key, i.e., E must spend $\frac{1}{2}N^2$ units of effort. B, on the other hand, need only spend N units of effort to break the one puzzle he selected. If we can mass produce puzzles at a cost of 1 unit apiece, then A need only spend N units of effort to manufacture the N puzzles. Thus, both A and B will only put in N units of effort.

The success of this scheme depends on our ability to mass produce puzzles which require a known amount of effort to break. This can easily be done by encrypting the two pieces of information with some key, (not to be confused with the key in the puzzle,) which is deliberately selected from a small key space. In particular, if we select a key from a key space with 2*N different values, then it is certain the puzzle can be broken by trying each one of these values in turn. To insure that we must proceed by this method, we will select a very strong encryption scheme. In particular, if we use the Lucifer encryption scheme, it is doubtful that any approach towards breaking the puzzle will be successful except that of exhaustive search among the possible keys. Thus, our approach to the construction of a puzzle is simply to encrypt the puzzle id and the associated puzzle key with a powerful encryption method, already proven to be effective, but to deliberately limit the number of possible keys to a particular number, namely, 2*N. In this fashion, we guarantee that the puzzle can be broken with N units of effort, on an average, and we rely on the power of the encryption method to assure us that it will not be broken ~~in~~ with much less effort. There is one point that must be brought out. In breaking an encrypted message, the cryptanalyst relies on the redundancy in the message. If we select both the puzzle id and the key at random, there will be no redundancy, and thus no way of breaking the puzzle. When we try a particular key with which to decrypt the puzzle, we will get a bunch of random bits, and will have no way of knowing if this bunch of random bits is the solution, or ~~wether~~ whether it's just a bunch of random bits. Thus, we must deliberately introduce redundancy into our puzzle, so that it can be broken. This can be done easily enough by encrypting, along with the puzzle id and the puzzle key, a constant piece of information that is known to A, B, and E. When we try to decrypt the puzzle with a particular key, then the recovery of this constant part can be taken as evidence that we have selected the right key, and thus have broken the puzzle. Contrariwise, the absence of the constant part in the decrypted puzzle indicates that we have used the wrong key, and should try again.

Having given an outline of the method, we shall now turn to a detailed look at its implementation. The following method has the virtue that it can be implemented in a very small amount of memory, and has, in fact, been implemented in _____ lines of Fortran. The small memory requirements of the method imply that it can be run on any minicomputer or micro-computer.

Before describing the method in detail, a few points of notation must be cleared up. F will be used to designate an encryption function. Note that F can be any encryption function that the reader feels is particularly powerful and effective. As mentioned earlier, I like to imagine that F is actually an implementation of Lucifer. F will accept an arbitrary number of arguements. The first arguement is the key, and the remaining arguements are the message to be encrypted. All of the data objects will be bit strings of arbitrary length. Thus, the key might be a 20 bit bit string, the message might consist of several bit strings of varying length. We imagine that the bit strings that make up the message are first concatenated into one long bit string, which is then encrypted using F. To illustrate, we might have the following call on F:

F(10011101101000l,001100001,0101001100111,00100)

The first bit string is to be used as the key, and the remaining three bit strings form the message. In addition, to make matters easier, we assume that $N=2^K$, i.e., that N is actually a power of 2, or that numbers which take on one of N possible values are actually bit strings of length K. In the course of the description, we shall designate variables as being of length K, or length 2K, or length 100, and so forth. This indicates the number of bits in the representation of the corresponding variable. When ~~[illegible]~~ a variable has length 100, this ~~is~~

means only that it is very large, and can take on any one of a large number of possible values. It is roughly equivalent to saying "many". (In computer science, instead of saying "1, 2, 3, many." we say "1, 2, 3, 4, 5, ...98, 99, many"). The reader should therefore not attempt to read deep value into the selection of the length 100. In general, the other values have been carefully selected, and where the reason for a particular selection is not immediately obvious, it should be made apparent in the ensuing text.

We shall also use the function, RAND. RAND(P) generates a random number between 1 and P, inclusive. Note that the normal random number generator on a computer is not suited for this. We require either truly random numbers, or at least pseudo random numbers using a very powerful method of generating the next number in the sequence, (such as an encryption function), and also with a large seed, i.e., a seed selected from any one of 2^L values, where L is very large, i.e., several hundred.

Finally, when we have finished the message, we will transmit it using the function, TRANSMIT(ARG). Using these conventions, we can write the algorithm for A, who is generating the puzzles, in the following fashion:

```
VAR  ID,KEY1, KEY2;  BIT STRING OF LENGTH K;
     CONSTANT: BIT STRING OF LENGTH K+20;
     K1,K2,K3: BITSTRING OF LENGTH 100;
BEGIN
     K1:=RAND($2^{100}$);
     K2:=RAND($2^{100}$);
     K3:=RAND($2^{100}$);
     CONSTANT:=RAND($2^{K+20}$)
```

```
      TRANSMIT(CONSTANT);
      FOR I:= 1 TO N DO
        BEGIN
          ID:=F(K1,I);
          KEY1:=F(K2,ID);
          KEY2:=F(K3,ID);
          TRANSMIT(F(RAND(2N),ID,KEY1,KEY2,CONSTANT));
        END;
END;
```

Assuming the same variable declarations for B, we can then write B's code. We will need two new primitives for B, RECEIVE(ARG) is a procedure which returns the value of the next puzzle in ARG. We will also need a primitive which will allow us to decompose a bit string into its pieces, i.e., to do field extraction. We will do this simply by allowing multiple variables on the left hand side of an assignment statement, thusly:

~~A,B:=F(SOMEKEY,A,B)~~
~~A,B:=F~~xxx

A,B:=F^{-1}(SOMEKEY,F(SOMEKEY,A,B))

All that this is intended to indicate is that the different fields in the decrypted result can be identified as distinct entities, and can be dealt with as such, in spite of the fact that the encrypted message should be regarded as a monolithic blob of bits. The code for B would then appear as follows:

```
VAR       DECLARATIONS FOR ID, KEY1, KEY2, AND CONSTANT AS BEFORE;
     SELECTEDPUZZLE:  BIT STRING OF LENGTH K;
     THEPUZZLE,CURRENTPUZZLE: BIT STRING OF LENGTH LONG ENOUGH;
     TEMPCONSTANT: BIT STRING OF LENGTH K+20;
```

```
BEGIN
      SELECTEDPUZZLE:=RAND(N);
      RECEIVE(CONSTANT);
      FOR I:= 1 TO N DO
        BEGIN
          RECEIVE(CURRENTPUZZLE);
          IF I=SELECTEDPUZZLE THEN THEPUZZLE:=CURRENTPUZZLE;
        END;
      FOR I:= 1 TO 2*N DO
        BEGIN
          ID,KEY1,KEY2,TEMPCONSTANT:=F^-1(I,THEPUZZLE);
          IF TEMPCONSTANT=CONSTANT THEN GOTO DONE;
        END;
          PRINT(" SHOULD NOT REACH THIS POINT.");
          PANIC;
DONE:TRANSMIT(ID);
      COMMENT KEY1 AND KEY2 CAN NOW BE CONCATENATED, AND USED
              TO ENCRYPT FURTHER TRANSMISSIONS;
END;
```

At the very end, A must receive the ID that B transmitted, and do something with it. The last actions that A must perform are as follows:

```
BEGIN
      RECEIVE(ID);
      KEY1:=F(K2,ID);
      KEY2:=F(K3,ID);
      COMMENT KEY1 AND KEY2 NOW HAVE THE SAME VALUE IN BOTH A
              AND B.  ALL A HAS TO DO IS CONCATENATE THEM, AND
              USE THE RESULT AS THE KEY WITH WHICH TO ENCRYPT
              FURTHER TRANSMISSIONS.
END;
```

The astute reader will have noticed that the puzzles now have four components. The ID and CONSTANT have been mentioned before, but we now have two KEY's, instead of one. The additional key was added for the following reason: KEY1 and KEY2 are both K bits long. If we only provided one K bit key, then there would be a total of N possible keys, $(N=2^K)$. This is bad, because E could then check all possible values of the key in time N. With two keys, E must check N^2 possible combinations of values, and thus, E gains nothing by this approach.

The reader will also notice that we are not selecting the values of the ID and of the two KEY values at random, but are, instead, computing them via the encryption function F, and some auxilliary keys, K1,K2, and K3. By letting ID:=F(K1,I), we guarantee that each ID will be distinct, for each ID is the encryption of a distinct value of I. E does not know the values of any of the auxiliary keys, K1, K2, or K3. Only A knows the value of these keys, and never transmits the value to B. By making the two key values, KEY1 and KEY2, depend on ID in a functional way, we are freed from the need for keeping them in a table. Given the ID, we can readily compute the two key values. The reader will notice that if E breaks a handful of puzzles, he will be in possession of several pairs of values, Y=F(SOMEKEY,X). We assume that the encryption function, F, is powerful enough to withstand this mistreatment. As was mentioned earlier, modern encryption schemes are indeed this powerful.

The only information available to E is the code executed by A and B, and the values actually transmitted over the key channel.

Thus, E is in possession of the ID that B transmitted to A, and is also in possession of the puzzles that A transmitted to B. While it is true that the ID uniquely identifies the puzzle which B selected, it is also true that the only way that E can determine the puzzle ID for any of the puzzles is by breaking the puzzle. Because of the way the ID is selected, each puzzle will be unique. The only point remaining is this: why is CONSTANT a bit string of length K+20? Clearly, we require some redundancy. How much, however, is enough? Examining the code for B reveals that the redundancy is used in one test, TEMPCONSTANT=CONSTANT. That is, when we decrypt a puzzle, using one of 2N possible keys, we wish to determine whether we have found the correct solution or not. If, by chance, we should decrypt a puzzle and get the correct CONSTANT, even though this is not the correct answer, then we have problems. We are examining a total of 2N possible decryptions. If we assume that our encryption function reduces its input to a good approximation of white noise, then we are, in essence, selecting 2N values for the CONSTANT at random. If any of these values should chance to equal the CONSTANT, then B will pick the wrong values of KEY1 and KEY2, as well as an erroneous ID. We must, therefore, make the space of possible CONSTANTS large enough that 2N samples looks like a drop in the bucket. We do this by making our constant space roughly 1,000,000 times bigger than 2N, i.e., 2^{K+20} is roughly equal to 1,000,000 N. Our chances of getting a false solution are thus about 1 in 1,000,000. If this is not enough, we can add 10 more bits, and reduce the chances to about 1 in 1,000,000,000.

We will now summarize the essential properties of the method described, ignoring the details of how it works. In essence, the method allows the use of channels satisfying assumption 1, and not satisfying assumption 2, for the transmission of key information. We need only guarantee that messages get through, unmodified, and we no longer require that they get through unread. If the two communicants, A and B, put in N units of effort, then the enemy, E, must put in $O(N^2)$ units of effort to determine the key. We now turn to the consideration of possible practical applications. In the course of the ensuing discussion, we shall consider various weaknesses in the ~~scheme~~ method as described, and consider techniques for overcoming them.

Upon first considering the matter, it would appear that we have purchased a less severe restriction on the nature of the key channel, at the cost of an increased bulk of information that must be transmitted over that channel to establish a key. To put it another way, if we wish to insure that E must spend at least $1,000,000 dollars to break current encryption methods, then we must transmit a small amount of information, perhaps a few hundred bits, over a key channel which E cannot read. To accomplish the same effect using the current method, we must transmit N puzzles, each/puzzle consisting of perhaps one hundred bits, over a key channel which E can read, but can't modify. We must select N large enough so that the cost of breaking $\frac{1}{2}N^2$ puzzles is greater than $1,000,000 dollars. This might require that we transmit a few megabytes of data.

Upon considering the matter more closely, ~~W~~ we realize that we need not transmit the bulk of the ~~pu~~ data over the key channel, but can instead transmit all the puzzles over the normal channel, and confirm their proper reception via the key channel. All we need do is to maintain a hash total of the transmitted puzzles, and then transmit this hash total over the key channel. If A and B both compute hash totals directly from the transmitted puzzles, then they can compare these hash totals over the key channel. If the hash totals match, then the puzzles were correctly received. If the hash totals do not match, then someone or something has tampered with the communications. The alert reader might notice certain logical difficulties with this scheme, but they will be dealt with in detail later on. ~~Wa~~

With this addition, we see that the capacity of the key channel need only be a few hundred bits in either case. We have, therefore, significantly reduced the restrictions on the key channel, at the cost of an increased bulk of transmitted information on the normal channel, as well as increased processor time to generate and break the puzzles. There are several interesting properties of this arrangement, that will be brought out in the following paragraphs.

First, we do not care if E knows the hash total. Therefore, we can maintain multiple copies of the hash total without a loss of security. We can maintain a permanent log of the hash total, and we can transmit the hash total by any means that is available to us. We can publish the hash total, and propogate so many copies of it, that E will be hard pressed to find them all, let alone alter them.

Second, we no longer have to transmit information over the key channel prior to using the key that we have agreed upon. Thus, it is possible to establish a key today, use it, and confirm, sometime next week, that no one tampered with our communications.

Third, we can detect a violation of assumption 1. If E does, in fact, falsify or alter messages, then what A transmitted, and what B received, will be different. There is no way that E can get around this fact. In effect, if E wishes to break the method by tampering with the messages actually sent, he must post a large sign, saying "I know and understand your current cryptographic set up, I have broken your current keys, and even now, I am listening to what you say." Needless to say, E might be reluctant to do this.

~~Fourth,~~

It should also be noted that if E attempts to determine what key will be selected by tampering with transmissions, then we can create the following interesting situation. For the key that we actually use to encrypt and decrypt messages, we can use the concatenation of the key that we select from the puzzle, and the hash total. Thus, if E tampered with transmissions, and by this method, determined the key being used, we can still force the key used by A, and the key used by B, to be different. If A transmits messages using one key, and B receives messages and attempts to decrypt them using a different key, then we have one of two possible situations. Either B gets complete garbage, or he gets messages that make sense. The only way he can get messages that make sense is if E is encrypting them with the correct key. The only way a message can get from A to B, is if E first decrypts it using the key that A knows, and then recrypts it using the key B knows. E is not only listening to messages being sent, he is an integral part of the communications system! The opportunities for detecting E's presence are correspondingly increased.

It was mentioned earlier that there are certain logical difficulties with maintaining the hash total. In particular, if all that we remember of the puzzles is the hash total, then we must insure that E cannot create a different series of puzzles which produces the same hash total. Normal hashing methods will not insure this. Simply adding the puzzles together is not enough, for E can then inject false puzzles into the channel, and then inject a final "puzzle" which was chosen to make the sum come out right. If E should do this, then he can fool A and B into thinking the transmission took place correctly, when in fact it didn't. We must, therefore, create a hash method ~~total~~ which has the interesting property that it is virtually impossible to find a message sequence that produces a given hash total. Those familiar with the concept of one-way encryption will see the analogy at once. For those who are unfamiliar with it, I shall give a brief summary of the problem, and how it can be solved.

As its name implies, one-way encryption may be viewed as a form of encryption. The object is to encrypt a message or arguement in such a fashion that it may never be decrypted with

any merely human amount of effort. We desire that no one can decrypt the message even though everyone is aware of the method used to encrypt it. Unlike normal encryption, we do <u>not</u> assume that there is a key, which is unknown to the enemy. The enemy knows everything about what occured, with the sole exception that he does not know what the message actually was. To put it another way, we wish to find a function, G, which is easy to compute, but for which the inverse, G^{-1}, is very hard to compute. A method for doing this is, in fact, given in the literature. It relies upon the following observation.
In a normal encryption scheme, E knows a great deal of message traffic, but does not know the key. Even though the key is, in all probability, uniquely specified by the message traffic, E will be unable to deduce what it is. In effect, the message traffic uniquely defines the key, and we may therefore regard the key as the result applying some function, G^{-1}, to the message traffic. On the other hand, given the key, and given the messages actually sent, we can easily compute the message traffic. We can select almost any sample of message traffic that we desire, and, for the sake of convenience, we might select the one piece of message traffic which is most convenient, the arguement of the one-way encryption function itself.
To summarize, if F is a powerful encryption function, then G(X)=F(X,X) is an effective one-way encryption function. It is hard to invert this function for precisely the same reason that it is hard to deduce the key of an encryption function, given the messages that were encrypted, and their encryption.

With this concept safely in hand, we see that we are simply attempting to extend the ~~concept~~ idea of one-way encryption to encompass the current situation. We wish to compute a one-way hash total. To do so, we will start by assuming that we have available a one-way encryption function, as described, and will then use this function, G, to aid us in computing the hash total.
In a normal hash total, all we do is keep a running sum of the numbers concerned. We shall adopt a similiar strategy for a one-way hash total, but will simply insert G at strategic points in the operation. Thus, we might code a normal hash total as follows:

```
BEGIN
     TOTAL:=0;
     FOR I:= 1 TO N DO
       BEGIN
         RECEIVE(MESSAGE);
         TOTAL:=TOTAL+MESSAGE;
       END;
END;
```

For our one-way hash total, we will proceed in a similiar fashion, but with a few small changes.

```
BEGIN
     TOTAL:=0;
     FOR I:= 1 TO N DO
       BEGIN
         RECEIVE(MESSAGE);
         TOTAL:=TOTAL+G(MESSAGE);
         TOTAL:=G(TOTAL);
       END;
END;
```

The reader might wish to try and break this method, to convince himself that it works.

There is one flaw yet remaining in the method of agreeing on a key. While it is true that E must, on the average, put in $\frac{1}{2}N^2$ units of effort, E might be lucky, and select the correct puzzle after only a few tries. A and B might wish to guarantee that E must put in a certain amount of effort before he can obtain a solution, and might be unwilling to gamble that E will probably have to put in $\frac{1}{2}N^2$ units of effort, but might luck out, and put in significantly less effort. This can be done, but only for a price. Let us examine how.

E might be lucky, and select the correct puzzle after only a few tries. There is no reason, however, for B to break only one puzzle. B could, in fact, have broken two puzzles. Before E can determine the key used, E must break the same two puzzles. While E might be lucky, and find one of the puzzles quickly, his chances of selecting both puzzles quickly is significantly less. In general, if B selects P puzzles to break, then the chance that E will happen to break all P puzzles, after breaking only $\overset{Q}{\cancel{X}}$ of the N puzzles transmitted, is $(\overset{Q}{\cancel{X}}/N)**P$. If E has broken $\overset{Q}{\cancel{X}}$ puzzles, then the chance that he has broken a particular puzzle is $\overset{Q}{\cancel{X}}/N$. E must break all P puzzles, however, and so the chance that E has broken all of them is just the product of the probability that E has broken each puzzle, i.e., $(\overset{Q}{\cancel{X}}/N)*(\overset{Q}{\cancel{X}}/N)*(\overset{Q}{\cancel{X}}/N)*......*(\overset{Q}{\cancel{X}}/N)$, where the factor $(\overset{Q}{\cancel{X}}/N)$ is repeated P times, i.e., $(\overset{Q}{\cancel{X}}/N)**P$. If P is 10, i.e., B breaks 10 puzzles, then the chance that E will have broken all ten puzzles after having broken $\frac{1}{4}$ of the N puzzles, is just $\frac{1}{4}**10$, or about 1/1,000,000.

As can be seen, we can reduce the chance that E lucks out to an arbitrarily small amount, but only at the price of increasing the amount of work that B puts in. The trade-off involved must be considered in the context of a particular application. If improved reliability is desired, however, then it can be obtained.

In summary of the discussion to the current point, we can say the following. The paradigm for cryptographically secure communications was examined. This paradigm was extended to include two seperate channels of communication: the key channel, and the normal channel. A method was described which reduced the (severe) restrictions on the key channel. Various difficulties with the method as initially described were considered, and overcome. The only remaining weakness in the method is the inherent one that it is $O(N^2)$, and not exponential. The weaker restrictions on the key channel open up the possibility of using more normal, i.e., cheaper, channels of communication with which to update the keys. In addition, violation of the weaker restriction on the key channel can be detected, and corrective action taken. Violation of the stronger restriction that the key channel must be unreadable, might go unnoticed. In the event that there is <u>no</u> channel available which satisfies the stronger restriction, but there is <u>is</u> a channel which satisfies the weaker restriction, then the current method provides an option which is otherwise unavailable. We now turn to some examples to illustrate the ideas developed so far, and to provide the reader with some ~~numerical examples of~~ feel for just how much safety can be provided by $O(N^2)$.

The first example we shall consider is the application of the current method to the security of a computer network, such as the ARPA net. In this situation, we have available a communications channel with a large enough capacity to handle a large N without excessive strain. The ARPA net, in particular, is connected by 50 kilobit lines. Also, in a network, we frequently have more than a single channel connecting two nodes of the network. This makes it more difficult to block message transmission. The ARPA net is, in fact, a two connected network. If E were to try and defeat the current method by altering message transmissions between nodes of the ARPA net, he would have to successfully block most transmissions along two 50 kilobit lines, and then inject his own. Once he succeeded in doing this, his actions could be easily detected.

To be specific, let us assume that we desire to change keys once a week, and that we are willing to tolerate a 1% overhead for this operation. Let us further assume that each puzzle is 100 bits long. With 50 kilobit transmission lines, 100 bits per puzzle implies we can send 500 puzzles per second. Restricting ourselves to 1% of this capacity, we get an average transmission rate of 5 puzzles per second. Continued over a period of one week, this is 5*60*60*24*7 or 3,024,000. For further computations, we shall approximate this by $3*10^6$.

~~We first note that $3*10^6$ can be represented with 22 bits. Examining the description of a puzzle reveals that it needs 4*~~K~~N+20 bits, or, in this case,~~ 108 ~~98 bits. Our approximation that a puzzle needs about 100 bits is therefore~~ almost ~~fully justified.~~

Given that there are $3*10^6$ puzzles, and that B is willing to devote 1% of his time to the task of breaking a single puzzle, then E must devote $3*10^6*.01$ weeks of computer time to break all N puzzles, or, on an average, $1.5*10^4$ weeks of computer time. If we assume that computer time is worth 10 dollars per hour, then E must spend $1.5*10^4*24*7*10$ or 25,200,000 dollars of computer time to break the link. The estimate of 10 dollars per hour is deliberately conservative, for there is no reason for E to use the same computer that we use. E might be able to get a bargain, or he might be willing to wait a few years, when the cost of computer time will be less. Also note that we are computing the mean cost to E to break the link. As pointed out earlier, the actual cost might differ significantly from the mean cost. If we assume, however, that E wishes to maximize his expected gains, then E will feel no temptation to break the link until he expects to learn information worth about 25 million dollars. Having broken the link, E will be in possession of one weeks worth of transmissions. I find it hard to imagine that E would seriously consider spending 25 million dollars on such a venture. E would be better advised to spend his money on bribes, blackmail, physical burglary of the nodes, and the like.

At this point, the reader might object that each node must communicate with several other nodes. We can avoid this problem if we assume that the cryptographically secure communications are routed through a sub-network which is in a star configuration. Each node establishes a secure link with the center of the star. The node which has been designated as the center of the star can then distribute keys to the other nodes in the network, as they find that they wish to communicate with each other. Physical security at the center of the star will have to be, of course, excellent.

The effort to implement this on the ARPA net would involve a modest coding effort, no additional hardware, and a 1% transmission and computation overhead. Please note that this 1% overhead could be distributed over a week in the most convenient manner, i.e., during time when the nodes would otherwise be idle. Once established, it would require little further attention from personnel. If the reader prefers a link-by-link encryption scheme over a star configuration, then each node will be connected to a few other nodes, and must distribute its 1% of effort over all of them. If we assume that each node is directly connected to between 3 and 4 other nodes, then, under the same assumptions as before, we can establish links that will require about 5 million dollars to break.

The method would also appear to have applications in a military setting. In particular, I invite the reader to consider the following situation. Two countries, X and Y, are at war. X manages to break some of Y's military ciphers, without Y's knowledge. X maintains tight security on its knowledge of Y's ciphers, but X must use the knowledge that it obtains from its cryptanalysis of Y's ciphers, and sometimes X's actions are suspiciously well timed. In particular, during the course of a major military campaign, X's actions are so well suited to

The method would also appear to have applications in a military setting. In the ARPA net example, it was shown that devoting 1% of the networks resources to establishing keys would result in links that would require some 25 million dollars worth of computer time to break. The method is $O(N^2)$, so increasing the amount of effort to 10%, i.e., an increase by a factor of 10 in the work put in by A and B, would increase the work put in by E by a factor of 100. In that example, the figure computed was 25 million dollars. Multiplying this by 100 gives 2.5 billion dollars. This figure is large enough that it is interesting even in military situations, where a great deal can rest on the secrecy of communications. The reader is advised to view the figure of 2.5 billion dollars with the following two considerations in mind. First, as was mentioned earlier, the mean cost would be 2.5 billion dollars. The actual cost, in a particular situation might be quite different from this figure. If assurance is sought that this figure will be close to the actual amount of effort put in by E, then several puzzles must be broken by B, increasing the amount of work he puts in. Also, it is possible to do a lot of hardware development for 2.5 billion dollars. E might choose to build the algorithm into hardware, and thus significantly increase both his speed and his cost effectiveness. The counter measure for this is to have A and B produce hardware realizations of the algorithm themselves. This, however, will increase its cost.

One of the significant advantages of the current method in a military setting would be the possibility of ~~updating~~ changing keys with less fuss, and less delay. The primitive methods that one is forced to use to change keys ~~that are always kept in the clear and hence must be~~ using current methods is perhaps best illustrated with a quote from <u>The Code-Breakers</u>: "A new edition... ~~which was~~ was to be placed in service April 1. But administrative confusion in the Navy libraries, which had custody of the codebooks, plus difficulties in physically distributing the books by destroyer and airplane to moving ships and widely dispersed installations, forced a postponement to May 1." A few pages later, we learn that "...the effective date of the new edition of the fleet cryptographic system, which had been postponed once from April 1 to May 1, had to be again set back another month, to June 1. Perhaps the very extent of the Japanese conquest defeated their distribution efforts." "Had Japan changed her main Naval code on May 1 as scheduled, she would have blacked out Allied cryptanalysts for at least several weeks—weeks that, as it turned out, were to be crucial to history." "Her failure to do so meant she was/masking ~~preparing~~ her Midway preparation messages behind a cryptographic smoke screen that American cryptanalysts had almost entirely blown away." The American victory at Midway was due, in large part, to American cryptanalytic success in breaking Japanese codes. This in turn relied on the fact that the Japanese failed to change their code on schedule, which was in turn caused, to a large extent, by physical difficulties in distribution. These difficulties were forced by the need to maintain absolute secrecy. If the need to maintain secrecy is relaxed, the problem of changing keys is significantly simplified.

The paper has so far dealt with the strengths and limitations of a particular method, which is $O(N^2)$. Most of the problems dealt with would become insignificant if a method that was $O(2^N)$ were possible. If an exponential method were possible, it would offer such significant advantages over current techniques that it would almost surely supplant them in short order. One of the most interesting features of the current work is that it suggests that such exponential methods are not impossible. The author has been able to find no mention in the literature of any ~~serious~~ attempts to solve this problem, and has found only one instance in which the problem was even defined with any clarity. This was in a science fiction story. The problem appears to offer enough leverage that it can be attacked, as witness the current solution, and an exponential solution would appear to offer significant practical advantages over current techniques. It would appear to merit serious consideration. The author will, in fact, make the following conjecture:

An exponential method is possible.

The reader is invited to consider the problem.