17. Dijkstra, E.W. After many a sobering experience. E.W. Dijkstra note EWD500.
18. Fenichel, R.R., and Yochelson, J.C. A LISP garbage-collector for virtual-memory computer systems. *Comm. ACM 12*, 11 (Nov. 1969), 611–612.
19. Greenblatt, R. LISP Machine Progress Report memo 444, A.I. Lab., M.I.T., Cambridge, Mass., Aug. 1977.
20. Greenblatt, R. Private communication, Feb. 1977.
21. Hoare, C.A.R. Optimization of store size for garbage collection. *Inform. Processing Letters 2* (1974), 165–166.
22. Knuth, D.E. *The Art of Computer Programming, Vol. I: Fundamental Algorithms.* Addison-Wesley, Reading, Mass. 1968.
23. Lamport, L. Garbage collection with multiple processes: An exercise in parallelism. CA-7602-2511, Mass. Computer Associates, Wakefield, Mass., Feb. 1976.
24. Lamport, L. On-the-fly garbage collection: Once more with rigor. CA-7508-1611, Mass. Computer Associates, Wakefield, Mass., Aug. 1975.
25. McCarthy, J., et al. LISP 1.5 Programmer's Manual. M.I.T. Press, Cambridge, Mass., 1965.
26. Minsky, M.L. A LISP garbage collector algorithm using serial secondary storage. Memo 58, M.I.T. A.I. Lab., M.I.T., Cambridge, Mass., Oct. 1963.
27. Moon, D.A. MACLISP Reference Manual. Project MAC, M.I.T., Cambridge, Mass., December 1975.
28. Muller, K.G. On the feasibility of concurrent garbage collection. Ph.D. Th., Tech. Hogeschool Delft, The Netherlands, March 1976 (in English).
29. Schonhage, A. Real-time simulation of multidimensional Turing machines by storage modification machines. TM-37, Project MAC, M.I.T., Cambridge, Mass., Dec. 1973.
30. Steele, G.L. Jr. Multiprocessing compactifying garbage collection. *Comm. ACM 18*, 9 (Sept. 1975), 495–508.
31. Steele, G.L. Jr. Private communication, March 1977.
32. Teitelman, W., et al. INTERLISP Reference Manual. Xerox Palo Alto Res. Ctr., Palo Alto, Calif., 1974.
33. Wadler, P.L. Analysis of an algorithm for real-time garbage collection. *Comm. ACM 19*, 9 (Sept. 1976), 491–500.
34. Weizenbaum, J. Symmetric list processor. *Comm. ACM 6*, 9 (Sept. 1963), 524–544.

Programming Techniques
S. L. Graham, R. L. Rivest Editors

Secure Communications Over Insecure Channels

Ralph C. Merkle
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

According to traditional conceptions of cryptographic security, it is necessary to transmit a key, by secret means, before encrypted messages can be sent securely. This paper shows that it is possible to select a key over open communications channels in such a fashion that communications security can be maintained. A method is described which forces any enemy to expend an amount of work which increases as the square of the work required of the two communicants to select the key. The method provides a logically new kind of protection against the passive eavesdropper. It suggests that further research on this topic will be highly rewarding, both in a theoretical and a practical sense.

Key Words and Phrases: security, cryptography, cryptology, communications security, wiretap, computer network security, passive eavesdropping, key distribution, public key cryptosystem
CR Categories: 3.56, 3.81

Author's address: Dept. of Electrical Engineering, Stanford University, Stanford CA, 94305.

Introduction

People have been communicating with each other for many millennia. They often wish to communicate secretly. Since the first use of Caesar's cipher, some two thousand years ago, people have employed a number of ciphers and codes in attempts to keep their correspondence secret. These have met with varying degrees of success until the modern age. The modern digital computer has made it possible to create ciphers which are, in practical terms, unbreakable.[1] (At least, if anyone has broken them, they are maintaining a discreet silence.) 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 secret communications can take place. It is the purpose of this paper to consider this paradigm, and to question the assumptions which underlie it. One assumption (that we must transmit a key, by secret means, prior to an attempt to communicate securely) which has traditionally been regarded as a necessary precondition for cryptographically secure communications is not, in fact, necessary. This is demonstrated by exhibiting a solution which allows two communicants to select a key publicly, but in such a fashion that no one else can easily determine it.

The body of the paper begins with an explanation of the traditional paradigm and then develops a new paradigm, which differs significantly from the traditional one. We then argue that the new paradigm is consistent with secret and secure communications. Finally, the implications of the new paradigm are explored in more detail, with the aid of some examples.

Review

Secure communication takes place between two individuals when they communicate privately in spite of efforts by a third person to learn what is being communicated. We shall, therefore, introduce three protagonists into our paradigm, X and Y, the two communicants, and Z, the third person, who wishes to find out what X and Y are communicating. X and Y have available some method of encrypting and decrypting messages that they send to each other. X, Y, and Z all know the general method of encryption. X and Y also have available a normal communications channel, over which they send the bulk of their messages. To allow X and Y to be secure, we also assume that they know a key, which is unknown to Z. The general method uses this key as a parameter, and will perform a particular transformation on messages for a particular key. Because Z does not know this key, he cannot perform the particular transformation, and thus cannot encrypt or decrypt messages.

X and Y must both know what the key is, and must insure that Z does not know what it is. In the traditional paradigm for cryptography, this situation comes about by the transmission of the key from X to Y over some special and secure communications channel, which we shall refer to as the key channel. Z cannot intercept messages sent on this channel, and the key is therefore safe.

The reason that the key channel is not used for normal communications is because of its expense and inconvenience. Radio and telephone cannot be used, as both are vulnerable to passive eavesdropping. Registered mail might be acceptable for moderate security. Word of mouth is better, but listening devices might compromise it. Perhaps the only safe method is to send a trusted courier, with an attache case chained to his wrist. This requires that you trust the courier. Whatever the method used, if Z should manage to discover the key by "practical cryptanalysis," then X and Y might very well continue in blissful ignorance of the fact.

In view of the central position that the key channel will occupy in this paper, it would be wise to state, somewhat more clearly, the conditions which it must satisfy. 1) All attempts by Z to modify or alter messages on the key channel, or to inject false or spurious messages, can be detected. 2) Z is unable to determine the content of any message sent over the key channel, i.e., Z cannot intercept the messages.

The paradigm, as stated so far, is not new, and has appeared previously in the literature (see, for example, Shannon [6]). We will now make a modification to the traditional paradigm, as given, which has not previously been considered.

The New Approach

We modify the traditional paradigm by dropping the second restriction on the key channel: that is to say, we no longer demand that Z be unable to determine what is sent on the key channel. Even stronger, we assume that Z has perfect knowledge of everything that is sent over this channel.[2]

It is the thesis of this paper that secure communications between X and Y can still take place, even under the highly restrictive conditions we have described.

The reader should clearly understand that no key lurks in the background. There is no method by which X and Y can communicate other than the normal channel and the key channel. They have made no secret preparations prior to the time that they wish to communicate securely.

We must carefully consider what constitutes a solu-

[1] As mentioned by Kahn, in [3], p. 711, "In one case, I.D.A. [Institute for Defense Analyses] cryptanalysts were given 1,000,000 letters of error-free text in a top military cryptosystem. They put in the equivalent of six man years on it—and finally gave up in defeat."

[2] A somewhat different approach is taken in [7]. In this paper, the author considers the problem of security in the face of a wiretap, and where no previous preparations have been made. However, he makes the critical assumption that the enemy has inferior reception of the messages being transmitted. By taking advantage of this inferior reception, the enemy can be completely confused.

tion. If X and Y eventually agree upon a key, and if the work required of Z to determine the key is much higher than the work put in by either X or Y to select the key, then we have a solution. Note that, in theory at least, Z can determine the key used in most methods simply by trying all possible keys and seeing which one produces a legible message. However, this means that Z must put in an amount of work that is exponentially larger than the amount of work put in by X or Y. The current solution is not exponential. The amount of work required of Z to determine the key will increase as the square of the amount of work put in by X and Y to select the key. Clearly, it would be desirable to find a solution in which the amount of work put in by Z increases exponentially as a function of the amount of work put in by X and Y. We see no reason why such exponential methods should not exist.

The Method

The method used is based on a single concept: that of a "puzzle." We define a puzzle as a cryptogram which is meant to be broken. To solve the puzzle, we must cryptanalyze the cryptogram. Having done this, we learn the information that was "enpuzzled," the plaintext of the cryptogram. Just as we can encrypt plaintext to produce a cryptogram, so we can enpuzzle information to produce a puzzle. A puzzle, though, is meant to be solved, while ideally, a cryptogram cannot be cryptanalyzed. To solve a puzzle, all one need do is put in the required amount of effort.

To sharpen our definition, we will consider the following method of creating puzzles. First, select a strong encryption function. We are not interested in the details of how this encryption function works: our only interest is that it does work. The reader can select any encryption function that he feels is particularly strong and effective. A concrete example might be the Lucifer encryption function [2], which is currently felt to be quite strong.

After selecting an encryption function, we create our puzzle by encrypting some piece of information with that function. Of course, if our encryption function is really good, our puzzle is unsolvable, which is not what we want. To avoid this problem, we artificially restrict the size of the key space used with the encryption function. If the key is normally 128 bits, we might use only 30 bits. While searching through 2↑128 possible keys seems completely infeasible, searching through 2↑30 is tedious, but quite possible. We can control the difficulty of solving a puzzle, simply by changing the restriction on the size of the key space used. To make the puzzle harder to solve, we might select a 40 bit key, while to make it easier, we might select a 20 bit key.

The puzzles we create by this method are precisely as difficult to break as we desire. We rely on the strength of the underlying encryption function to insure that our puzzle can only be solved by exhaustive search through the key space, and we adjust the size of the key space to control the difficulty of solving the puzzle.

There is still one more point that must be brought out. In cryptanalyzing an encrypted message, the cryptanalyst relies on the redundancy in the message. If the information we enpuzzle is random, there will be no redundancy, and thus no way of solving the puzzle. We must deliberately introduce redundancy into our puzzle, so that it can be solved. This can be done easily enough by encrypting, along with the information, a constant that is known to X, Y, and Z. 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 solved 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.

With the concept of "puzzle" in hand, we can proceed. We let X and Y agree upon the value of N which they wish to use. X then generates N puzzles, and transmits these N puzzles to Y over the key channel. X chooses the size of the key space so that each puzzle requires $O(N)$ efforts to break. (That is, X selects a key space of size $C*N$, for a constant, C.) Each puzzle contains, within itself, two pieces of information. Neither piece of information is readily available to anyone examining the puzzle. By devoting $O(N)$ effort to solving the puzzle, it is possible to determine both these pieces of information. One piece of information is a puzzle ID, which uniquely identifies each of the N puzzles. The ID's were assigned by X at random. The other piece of information in the puzzle is a puzzle key, i.e, one of the possible keys to be used in subsequent encrypted communications. To distinguish the puzzle keys, one for each puzzle, from the keys randomly selected from the restricted key space to create the puzzles, we will call the former "puzzle keys," and the latter, "random keys." Thus, N puzzle keys are enpuzzled, and in the process of enpuzzling each puzzle key, a random key is used. (The puzzle key is also selected by X at random.)

When Y is presented with this menu of N puzzles, he selects a puzzle at random, and then spends the amount of effort required to solve the puzzle. Y then transmits the ID back to X over the key channel, and uses the puzzle key found in the puzzle as the key for further encrypted communications over the normal channel.

At this point, we summarize who knows what. X, Y, and Z all know the N puzzles. They also know the ID, because Y transmitted the ID over the key channel. Y knows the corresponding puzzle key, because Y solved the correct puzzle. X knows the corresponding puzzle key, because X knows which puzzle key is associated with the ID that Y sent. Z knows only the ID, but does not know the puzzle key. Z cannot know which puzzle contains the puzzle key that Y selected, and which X and Y are using, even though he knows the ID. To determine which puzzle is the correct one, he must break puzzles at random until he encounters the correct one.

If Z desires to determine the key which X and Y are using, then, on an average, Z will have to solve $(\frac{1}{2})\ N$ puzzles before reaching the puzzle that Y solved. Each puzzle has been constructed so that it requires $O(N)$ effort to break, so Z must spend, on an average, $O(N\uparrow 2)$ effort to determine the key. Y, on the other hand, need only spend $O(N)$ effort to break the one puzzle he selected, while X need only spend $O(N)$ effort to manufacture the N puzzles. Thus, both X and Y will only put in $O(N)$ effort.

Having given an outline of the method, we shall now turn to a detailed look at its implementation. Before proceeding, 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 the reader feels is particularly powerful and effective. F will accept an arbitrary number of arguments. The first argument is the key, and remaining arguments are the message to be encrypted. All of the data objects will be bit strings of arbitrary 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(1001010110,011110100001,
01000000101101011,0010111)

The first bit string is to be used as the key, and the remaining three bit strings form the message.

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 pseudorandom numbers generated by a very powerful pseudorandom number generator. Of course, such a pseudorandom number generator will have to be initialized with a truly random seed.

When we have finished making the puzzle, we will transmit it using the function, *TRANSMIT*(*ARG*).

To summarize:

- N Total number of puzzles.
- C Arbitrary constant. The random key is selected from a key space of size $C*N$.
- F A strong encryption function. Its inverse is called "*FINVERSE*"

In the algorithm presented, we generate neither the ID's nor the puzzle keys at random. The ID's are generated by encrypting the numbers 1 through N. With a good encryption function, this can be viewed as a method of generating pseudorandom numbers. The puzzle keys are generated by encrypting the ID's. Again, this can be viewed as a good pseudorandom number generator. It has the additional property that the puzzle key can be quickly and easily generated from the ID. Two auxiliary keys, K1 and K2, are used in these two encryption processes, and provide the truly random "seed" for these somewhat unorthodox pseudorandom number generators.

Using these conventions, we can write the algorithm for X, who is generating the puzzles, in the following fashion:

```
var  ID, KEY, CONSTANT, RANDOMKEY,
  PUZZLE, K1, K2:bit string;

begin
  K1:=RAND(LARGE);
  K2:=RAND(LARGE);
  CONSTANT:=RAND(LARGE);
  TRANSMIT(CONSTANT);
  for I:=1 to N do
    begin
      ID:=F(K1,I);
      KEY:=F(K2,ID);
      RANDOMKEY:=RAND(C*N);
      PUZZLE:=F(RANDOMKEY, ID, KEY, CONSTANT);
        TRANSMIT(PUZZLE);
    end;
end;
```

We can now write Y's code. We will need a new primitive for Y: *RECEIVE*(*ARG*) is a procedure which returns the value of the next puzzle in *ARG*. We also need to clarify some notation. If we encrypt some arguments with F, we wish to be able to decrypt those arguments. If we say:

```
CIPHERTEXT:=F(SOMEKEY,A,B,C);
```

we want to be able to invert this by saying:

```
A,B,C:=FINVERSE(SOMEKEY,CIPHERTEXT);
```

The meaning of this should be obvious, in spite of the fact that we have three variables, A, B, and C on the left hand side of the assignment statement. With these additional conventions, the code for Y would then appear as follows:

```
var  ID, KEY, CONSTANT, SELECTEDPUZZLEID,
       THEPUZZLE, CURRENTPUZZLE,
       TEMPCONSTANT:bit string;

begin
  SELECTEDPUZZLEID:=RAND(N);
  RECEIVE(CONSTANT);
  for I := 1 to N do
    begin
      RECEIVE(CURRENTPUZZLE);
      if I=SELECTEDPUZZLEID then THE PUZZLE:=
          CURRENTPUZZLE;
    end;
    comment  The computation to find the randomkey used by A
follows;
  for I := 1 to C*N do
    begin
      ID,KEY,TEMPCONSTANT:=FINVERSE
        (I,THEPUZZLE);
    if TEMPCONSTANT=CONSTANT then
      goto DONE;
    end;
  print("should not reach this point.");
  panic;
DONE:   TRANSMIT(ID);
end;
```

At the very end, X must receive the ID that Y

transmitted, and deduce the key. The last actions that X must perform are as follows:

```
begin
  RECEIVE(ID);
  KEY:=F(K2,ID);
  comment KEY now has the same value in both X and Y. All they
    have to do is use KEY as the key with which to encrypt further
    transmissions.
end;
```

The only information available to Z is the code executed by X and Y, and the values actually transmitted over the key channel. Thus, Z is in possession of N, the CONSTANT, the ID that Y transmitted to X, and also the puzzles that X transmitted to Y. All other variables are known either exclusively by X, or exclusively by Y.

In summary: 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 are unmodified, and we no longer require that they be unread. If the two communicants, X and Y, put in $O(N)$ effort, then the third person, Z, must put in $O(N\uparrow 2)$ effort to determine the key. We now turn to the consideration of various implications of this work.

Some Implications

There is no reason to assume an exponential method is impossible. We will discuss the implications of the new paradigm, but will not limit ourselves to the $O(N\uparrow 2)$ method presented. To attain realistic levels of security using the $O(N\uparrow 2)$ method would require a large value for N, which would be costly. An exponential method would eliminate this cost, and so be more attractive. The existence of such an exponential method will be implicit in the following discussion.

We will assume that whatever method we are using can, like the $O(N\uparrow 2)$ method, be cast into the following form: first X transmits some information, then Y makes a return transmission, after which the key is known to X and Y. We will refer to the information that X transmits to Y as the first transmission, and the information that Y transmits to X as the return transmission. In the $O(N\uparrow 2)$ method presented, the constant and the N puzzles would be the first transmission, while the ID would be the return transmission.

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

Second, we can detect a violation of assumption 1. If Z does, in fact, falsify or alter a message, then what X transmitted, and what Y received, will be different. There is no way that Z can get around this fact. In effect, if Z 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 setup, I have broken your current keys, and even now, I am listening to what you say." Needless to say, Z might be reluctant to do this.

What, in practical terms, are the implications we can foresee from these relaxations of the security required on a key channel? We can see these better by means of some examples. We first consider the key distribution problem. Using the traditional method, codebooks are generated and distributed under tight security. Strong controls are needed to account for all copies and to prevent duplication of the codebook. Loss of a codebook is a disasterous breach of security. What effect would the current ideas have on this picture? In essence, they imply that codebooks can be lost without compromising security. We can give codebooks to the enemy, and still leave him in no better position than he was in before. The general simplification of security procedures that this would allow would be a significant advantage. Even more important would be the increase in actual security.

A key distribution system based on the current ideas might proceed as follows. First, each unit or command that wished to be in the codebook would generate its own first transmission. These would all be sent to a central site, where the names and first transmissions of all involved communicants would be entered into the codebook. The codebook would then be distributed. In essence, we are simply specifying the nature of the communication channel between X and Y. It is not a direct communication channel, but is somewhat roundabout. X publishes his first transmission in the codebook, along with his name. The return transmission from Y to X can now take place over normal communication channels. Y is assured that he is talking to X, because Y looked up X's first transmission in the codebook. At this point X and Y have established a common key, but X does not know that he is talking to Y. Anyone could have sent the return transmission, claiming they were Y. To avoid this, X and Y repeat the process of selecting a key, but X now looks up Y in the codebook, and sends a return transmisison to Y, based on Y's first transmission. The return transmission will be meaningful only to Y, because the return transmission is based on Y's first transmission. X knows Y's first transmission came from Y, because it is entered in the codebook. If X and Y now use both keys, then they are assured they are talking to each other, and no one else. To summarize: using only a codebook, which is assumed to be correct, but which is not assumed to be secret, X and Y have established an authenticated, secure communications channel. They have done so quickly and easily. The key need be used for only a short period of time (a single conversation), and can then be changed with equal ease.

The new paradigm also has implications for network security. In a computer network, with many users with diverse needs, security is difficult to maintain. If the

codebook in the previous example were compiled at the same time and by the same people who normally compile the directory of network users, the additional effort required would be minimal. Those network users interested in security would submit a first transmission to be included next to their entry in the network directory. They would also make sure that their copy of the network directory was correct. Those users not interested in security would ignore the security procedures. Diverse needs, ranging from no security, to very tight security, could then be met on the same network.

As a final example, consider the following situation. Assume two forces, Us and Them, are fighting. They are winning, because they have broken our codes and ciphers. We only find out about this when we discover that they attack exactly where we are weakest, retreat just before our attacks, and generally seem to know too much too quickly. Our forces are in the field, fully deployed, with no chance for distributing new keys in accordance with the traditional paradigm. Under the traditional paradigm, we are lost. Using the new paradigm, we can easily change all our keys, and re-establish security. The difference is dramatic.

We summarize the discussion to the current point. The traditional paradigm for cryptographically secure communications was examined. A new paradigm was proposed, and a method of key distribution was described which is consistent with the new paradigm. The only weakness in the method is that it is $O(N\uparrow 2)$, and not exponential. The weaker restrictions on the key channel demanded by the new paradigm open up the possibility of using more normal, i.e., cheaper, channels of communication with which to update 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, and result in catastrophic loss of security. This possibility is eliminated with the new paradigm. In the event that there is no channel available which satisfies the stronger restriction, but there is a channel which satisfies the weaker restriction, then the current method provides an option which is otherwise unavailable.

Conclusion

This paper has thus far dealt with a method which is $O(N\uparrow 2)$. If an exponential method were possible, it would offer such significant advantages over traditional techniques that it would almost surely supplant them in short order. The problem appears to offer enough leverage that it can be attacked, as witness the current solution, and an exponential solution would offer significant practical advantages over traditional techniques. The problem merits serious consideration. The author will make the following conjecture: An exponential method is possible. The reader is invited to consider the problem.

Addenda

Further progress has been made since the original submittal of this paper. Diffie and Hellman have published a paper [1] which gives a broad overview of the new class of cryptographic problems. It should be mentioned that the "puzzle" approach used in this paper can easily be recast as a public key cryptosystem, in the terminology of [1]. The author and Martin Hellman [4] have devised an exponential method (public key cryptosystem). Rivest, Shamir, and Adleman [5] have also proposed a public key cryptosystem. Further research is in progress, and the results have been extremely encouraging.

Acknowledgments. The author wishes to thank the following people for their kind contributions of time, thought, facilities, references, and encouragement: (in alphabetical order) B. Englar, R. S. Fabry, S. L. Graham, and F. Olken.

Received August 1975; revised September 1977

References

1. Diffie, W., and Hellman, M. New directions in cryptography. *IEEE Trans. on Inform. IT-22*, 6 (Nov. 1976), 644–654.
2. Feistel, H. Cryptography and computer privacy. *Sci. Amer. 228*, 5 (May 1973), 15–23.
3. Kahn, D. *The Codebreakers.* MacMillan, New York, 1976.
4. Merkle, R., and Hellman, M. Hiding information and receipts in trap door knapsacks. To appear, IEEE Trans. on Inform.
5. Rivest, R.L., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM 21*, 2 (Feb. 1978), 120–126.
6. Shannon, C.E. Communication theory of secrecy systems. *Bell Syst. Tech. J. 28* (1949), 654–715.
7. Wyner, A.D. The wire tap channel. *Bell Syst. Tech. J. 54*, 8 (Oct. 1975), 1355–1387.