SOME OPEN PROBLEMS IN CRYPTOGRAPHY

C. Leung

Department of Electrical Engineering
and Electronic Systems Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

In the past, cryptography has been mainly concerned with the problem of private communication between two parties. A number of ciphers exist which solve this problem more or less satisfactorily. One common factor behind these ciphers is the use of certain secret keys. With the advent of commercial data networks, there is a need for many pairs of users to communicate in privacy. The classical method of distributing secret keys (over a secure channel) to each user pair becomes very expensive and alternative means have to be explored. This paper describes a method which does not require prior exchange of secret keys for private communication over a public network. The cryptanalytic complexity of breaking this system is related to the complexity of solving a certain zero-one integer programming problem.

Keywords: Complexity; cryptography; key distribution; security

1. Introduction

The need for private communication arises whenever the distribution of information has to be restricted to certain groups. A wide variety of schemes for achieving privacy have been devised, ranging from invisible ink to very sophisticated ciphers. The problem addressed here concerns the use of ciphers for secure (private) transmission of messages over insecure (or public) channels.

Conventional ciphers allow private communication only among parties who have exchanged secret keys a priori. Unfortunately such cryptographic systems become impractical when a large number of users is involved. For example, in a network with 1000 users, there are nearly 500,000 pairs of users who may wish to communicate privately. The key distribution problem associated with this network becomes unmanageable and alternative ways of achieving privacy have to be explored. In a network environment, there is also a need for authenticating messages; i.e., the recipient of a message should be able to verify the identity of the sender.

In this paper, we will survey some cryptographic systems which have been recently proposed for use in data networks. Before doing so, we take a brief look at conventional cryptography and discuss the different levels of security that a cipher may possess.

2. Classical Cryptography

The classical study of cryptography is based on the system depicted in figure 1. The aim is to design easily implementable enciphering and deciphering algorithms which will allow the message m to be transmitted privately over the insecure channel to the legitimate receiver. We note that both the transmitter and the legitimate receiver know the key k which is kept secret from the cryptanalyst. $E_k(\cdot)$, $k \in [1,K]$, is a set of invertible transformations indexed by k. The transmitter does not send m directly over the insecure channel; instead an encrypted version $E_k(m)$ is sent. Since the legitimate receiver knows k, he can decipher m by operating with E_k^{-1} to obtain $E_k^{-1}(E_k(m)) = m$, the original message.

It is convenient to classify cryptographic systems into two broad categories. A system which can resist a cryptanalytic attack involving an unlimited amount of computation is said to be <u>theoretically</u> or <u>unconditionally</u> secure. On the other hand, a system which can be broken but at an inordinately high cost is referred to as being <u>computationally</u> secure.

Theoretically secure systems are based on the fact that there are multiple solutions to a cryptogram. An example is the one time pad or Vernam system. If our message is in English, we associate a distinct number (say from 0 to 31) with each letter of the alphabet and common symbols such as period, comma, space, etc. The key in this case is a completely random stream of numbers between 0 and 31. This random stream is then added (modulo 32) to the message to produce the cryptogram. If the cryptogram

KFRC KGO XF LDCBNS

is intercepted by a cryptanalyst, he will have no way to determine whether the message sent was

SELL ALL MY STOCKS
THEY ARE AT BILL'S, etc.

An excellent discussion of theoretically secure systems can be found in the classical work of Shannon [1]. Unfortunately the one-time pad requires one bit of key for each message bit and is impractical for many applications. For the most part, we will restrict our attention to computationally secure systems. Whether a system is to be considered computationally secure or not depends largely on the specific application. If the benefit which a potential cryptanalyst can derive from breaking a cipher is minimal, then a relatively small amount of computational effort will deter that person. Before we discuss how computationally secure privacy systems requiring no keys can be designed, we take a look at a computer security log-in problem.

3. A Secure Computer Log-in Procedure

The usual method used in a computer system for checking the authenticity of users involves the use of passwords. Each user is assigned an account number when he first joins the system. He is also allowed to choose a password which he keeps secret from other users. In order to verify the authenticity of users, the system stores a password directory consisting of all the users' account numbers and their corresponding passwords. Each time a user logs on, he is asked for his password. Only if

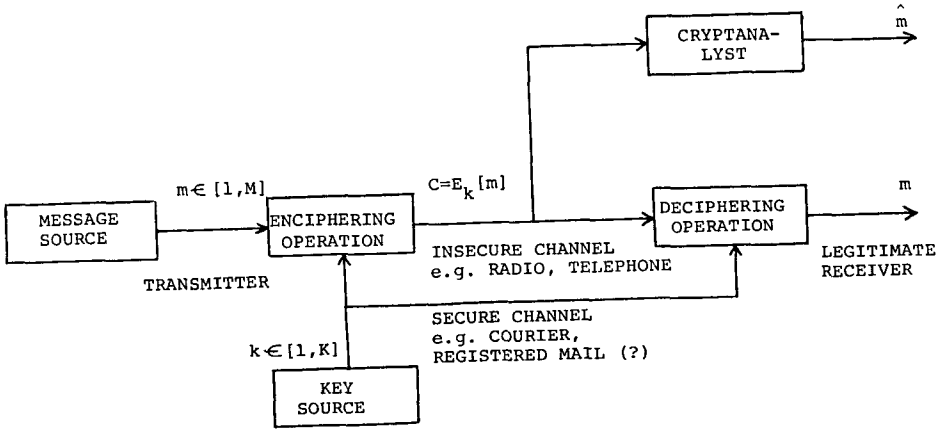

FIGURE 1
BLOCK DIAGRAM FOR A SECRECY SYSTEM

this matches the stored password is he allowed access to the account.

There are several problems associated with this log-in scheme. First, an intruder who steals the password directory can gain easy access to all the accounts in the system. Secondly, the system administrator can read everyone's password and there is no reason why he should be able to. What is required is a log-in procedure which can authenticate passwords without actually knowing them. Although this may at first appear to be inconsistent, it can quite easily be implemented using the notion of a one-way function. The original idea is due to R.M. Needham and various implementations have been suggested. [2,3,4]

A function f is said to be a <u>one-way function</u> if for all x in the domain of f and for any y in the range of f, it is very easy to compute f(x) but very difficult to compute any x such that f(x) = y. A one-way function f can be used to solve the log-in problem in the following way: When a user sets or resets his password PW, the computer automatically calculates f(PW) and stores f(PW), not PW, in the password directory. Subsequently, whenever the user enters a password PW' the computer forms f(PW') and compares it against the stored value f(PW). Only if f(PW') = f(PW) is the user authorized access to the system. Note that the password directory is not very useful to someone who steals it, since he will have to perform a lot of computation to invert f: in general f(PW) $\neq$ f(f(PW)).

One has to be somwhat cautious in formulating solutions to the computer log-in problem. Irrespective of the actual function which is used as the one-way function, the trial-and-error threat is always present: the cryptanalyst can try to break the system by computing f(x) for many x's in the domain of f. Therefore a necessary condition for a secure system is that the domain of f should be very large. To get an idea of the numbers involved, let us consider a typical present day computer system in which the password consists of three alphanumeric characters. Such a system can have $36^3 = 46656$ distinct passwords. If we assume that an intruder can try a million passwords per second, it takes less than 1/20 second to try all passwords. To get a reasonable level of security, about 10 or 15 characters have to be used for each password. Here, one has to face a serious human engineering problem since users do not easily remember random 15-character long passwords. A clever cryptanalyst can easily make use of redundancy introduced in the choice of passwords to break the system.

A possible one-way function can be obtained by considering a certain NP-complete problem known as the "knapsack problem" 5, p. 401 which can be stated as follows: Given an n-tuple of positive integers $\underline{a}$ and an integer y, is there a binary n-vector $\underline{b}$ such that $\underline{a}.\underline{b}^T = y$? A variation of this problem is to find $\underline{b}$, given that $\underline{b}$ exists. For convenience, we define $f(\underline{b}) \triangleq \underline{a}.\underline{b}^T$. Thus the problem is to invert f. We note in this context that inversion of f refers to finding any binary n-vector $\underline{b}'$ such that $f(\underline{b}') = f(\underline{b})$; it is not necessary to compute the exact original n-vector $\underline{b}$. It is therefore highly undesirable for f to be very degenerate in the sense that it maps a lot of distinct passwords into the same image. More will be said about the knapsack problem in the next section, where a method which does not require the use of secret keys for private communication over a public channel is discussed.

4. Achieving Privacy Without Secret Keys

The basic idea behind the scheme is the use of a technique known as the trap door method [6]. When designing a puzzle the designer might construct the puzzle in such a way that anyone possessing certain information can easily solve the puzzle, but this information is not easily obtained from a statement of the puzzle. This information is called a trap door. Several schemes using this technique have been proposed [7,8,9]. (see also [11]). The following addresses the implementation in [9].

Before describing the implementation, we indicate how a trap door system can be used to communicate privately over a public network. Consider a network with n users. Each user i constructs an easily computable encoding algorithm E_i using some trap door information which is kept secret from other network users. Because of his knowledge of the trap door, user i can construct an efficient decoding algorithm D_i for his encoding algorithm. D_i is kept secret by user i. If the encoding algorithm is properly designed, it is very difficult, if not impossible, for another user to compute user i's decoding algorithm without knowing the trap door.

E_i is placed in a public directory alongside user i's name, address, etc. Anyone wishing to send a message m to user i looks up E_i in the public directory and sends $E_i(m)$. User i can easily recover the intended message m by forming $D_i(E_i(m))$.

4.1 An Implementation

The implementation is based on the variation of the knapsack problem mentioned earlier. It is generally believed that inverting $f(.)$, where $f(\underline{b}) \triangleq \underline{a}.\underline{b}^T$ and $\underline{a}$ is an n-vector with components chosen uniformly and independently from the integers in $[1,2^n]$, is very hard, requiring $O(2^n)$ operations. However, there are certain classes of knapsack vectors $\underline{a}$ which admit very easy solutions.

In particular, we define a knapsack $\underline{a} = (a_1,\ldots,a_n)$ to be <u>simple</u> if $\underline{a}$ can be permuted to give

$$a_j > a_1 + \ldots + a_{j-1} \qquad (1)$$

for $j = 2,\ldots,n$. The motivation behind this definition is that any knapsack which satisfies eq (1) can be easily solved in O(n) steps, using the following algorithm.

Algorithm 1: Given a simple knapsack $\underline{a}$ which satisfies eq (1), and given an integer y, solve $\underline{a}\cdot\underline{b}^T = y$ for a binary n-vector $\underline{b}$.

```
i ← n
While i > 0 Do Begin
     If y < a_i  then  b_i ← 0 else b_i ← 1
     y ← y - a_i.b_i
     i ← i - 1
     End
Stop
```

It is not difficult to see that this algorithm computes the correct binary vector $\underline{b}$. From eq (1) it is clear that $y < a_n$ if and only if $b_n = 0$. This idea is used iteratively to compute all components of $\underline{b}$.

The idea behind the encoding algorithm is to construct a set of knapsacks such that combining them in a certain way produces a knapsack which is simple. The way in which they are combined is the trap door information which allows the designer to decode cryptograms sent to him.

In the following, by a <u>random mod K n-vector</u> $\underline{x}$, we will mean an n-vector each of whose components is chosen independently and uniformly from the integers mod K. Let us also denote by $\underline{a} \triangleq (a_1,\ldots,a_n)$ some simple knapsack of length n.

Each user A begins by choosing (s-1) random mod K n-vectors $\underline{x}_1,\ldots,\underline{x}_{s-1}$ and an n-vector $\underline{x}_s$ such that

$$\underline{x}_1 + \ldots + \underline{x}_s = \underline{a} \bmod K. \qquad (2)$$

A then chooses r additional random mod K n-vectors $\underline{x}_1',\ldots,\underline{x}_r'$. These (r+s) vectors are randomly arranged as the rows of an (r+s) x n matrix E_A. This E_A matrix represents the encoding algorithm for user A and is placed in a public directory. Anyone wishing to send a message (represented as a binary n-vector $\underline{m}$) to user A looks up E_A and computes and transmits

$$\underline{c}^T = E_A\cdot\underline{m}^T \pmod K \qquad (3)$$

The trap door information of user A is the set of the s rows of the E_A matrix which when added modulo K yields the easily solved knapsack $\underline{a}$. A uses this information (known only to him) to construct an easy decoding algorithm as follows. He adds up the components of $\underline{c}^T$ corresponding to the s rows $\underline{x}_1,\ldots,\underline{x}_s$ of E_A. The addition is done modulo K and the result is some integer y. Then as long as K is chosen such that

$$K > \sum_{i=1}^{n} a_i \ , \qquad (4)$$

the message $\underline{m}$ can be recovered by solving

$$\underline{a}\cdot\underline{m}^T = y. \qquad (5)$$

This is easily done since $\underline{a}$ is simple.

4.2 Cryptanalysis

There are two obvious approaches to cryptanalyzing the system described in section 4.1. In discussing these approaches it will be assumed that s is known. In practice s does not have to be made public since the encoding algorithm does not require it.

The most obvious form of cryptanalysis is to solve eq (3) directly for $\underline{m}$ without making use of the knowledge that $\underline{m}$ is binary. This would involve writing r+s-1 components of $\underline{m}$ as a function of the remaining n-(r+s-1) components. Since there are 2**(n-r-s) possibilities for these components, choosing n-r-s large makes searching all or most of the possibilities an impractical task.

A second approach is to combine (by addition mod K) components of $\underline{c}$ so that combining the corresponding rows of E_A yields a knapsack which is easy to solve. Since the components of every row of E_A but one are chosen as uniform random variables over the integers mod K, the result of adding (mod K) selected rows of E_A (other than the combination which yields $\underline{a}$) is an n-vector with each component chosen uniformly (randomly) over the integers mod K. If the cryptanalyst knows s, the number of rows which when added (mod K) give $\underline{a}$, then he has

$$\binom{r+s}{s} = \frac{(r+s)!}{r!s!} \qquad (6)$$

possible combinations of s rows and only one combination has a nonrandom result. If r = s, then eq (6) is approximately 2**(r+s). Choosing s large makes trying every combination of s rows an impractical task.

When adding together (mod K) a set of rows of E_A which is not the set which yields a, the result is a random knapsack mod K. The fraction of random knapsacks mod K which have a large simple subset is very small[10], so that it is unlikely that the random knapsack is easy to solve.

5. Discussion

An implementation of a general scheme for transmitting information securely over a public channel without the need for a prior exchange of secret keys has been proposed. In the design of his encoding algorithm, a user makes use of some secret information which enables him to construct a computationally simple decoding algorithm. It is believed that without this secret information a cryptanalyst would find it difficult to determine the message sent. Unfortunately, this fact is not yet proven. The security of the implementation presented here can only be judged on its success or failure to withstand concerted attacks.

It is important to note that the basic idea behind this implementation is not restricted to the knapsack problem. More generally, one can generate a collection of hard problems with the property that a certain subset when combined in a special manner yields an easily solved problem.

References

1. Shannon, C. E., Communication theory of secrecy systems, Bell System Tech. Journal, Vol. 28, pp. 657-715. (1949).

2. Wilkes, M. V., Time-Sharing Computer Systems, (Elsevier, 1972).

3. Purdy, G. B., A high security log-in procedure, Communications of the ACM, Vol. 17, pp. 442-445 (1974).

4. Evans, A., Kantrowitz, W., and Weiss, E. A user authentication system not requiring secrecy in the computer, Communications of the ACM, Vol. 17, pp. 437-442 (1974).

5. Aho, A. V., Hopcroft, J. E., and Ullman, J. D., The Design and Analysis of Computer Algorithms, (Addison-Wesley, 1976).

6. Diffie, W., and Hellman, M. E., New directions in cryptography, IEEE Trans. on Information Theory, Vol. IT-22 pp. 644-654 (1976)

7. Rivest, R. L., Shamir, A., and Adleman, L., A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM, Vol. 21, pp. 120-126 (1978).

8. Merkle, R. C., and Hellman, M. E., Hiding information and receipts in trap door knapsacks, submitted to IEEE Trans. on Information Theory.

9. Leung, C., and Vacon, G., A method for private communication over a public channel, submitted to IEEE Trans. on Information Theory.

10. Pohlig, S. C., Bounds on a class of easily solved knapsacks, submitted to IEEE Trans. on Information Theory.

11. Merkle, R. C., Secure communications over insecure channels, Communications of the ACM, Vol. 21, pp. 294-299 (1978).