Programming Techniques
R. Rivest Editor

How to Share a Secret

Adi Shamir
Massachusetts Institute of Technology

In this paper we show how to divide data D into n pieces in such a way that D is easily reconstructable from any k pieces, but even complete knowledge of $k-1$ pieces reveals absolutely no information about D. This technique enables the construction of robust key management schemes for cryptographic systems that can function securely and reliably even when misfortunes destroy half the pieces and security breaches expose all but one of the remaining pieces.

Key Words and Phrases: cryptography, key management, interpolation
CR Categories: 5:39, 5.6

1. Introduction

In [4], Liu considers the following problem:

> Eleven scientists are working on a secret project. They wish to lock up the documents in a cabinet so that the cabinet can be opened if and only if six or more of the scientists are present. What is the smallest number of locks needed? What is the smallest number of keys to the locks each scientist must carry?

It is not hard to show that the minimal solution uses 462 locks and 252 keys per scientist. These numbers are clearly impractical, and they become exponentially worse when the number of scientists increases.

In this paper we generalize the problem to one in which the secret is some data D (e.g., the safe combination) and in which nonmechanical solutions (which manipulate this data) are also allowed. Our goal is to divide D into n pieces $D_1, \ldots, D_n$ in such a way that:

(1) knowledge of any k or more D_i pieces makes D easily computable;
(2) knowledge of any $k-1$ or fewer D_i pieces leaves D completely undetermined (in the sense that all its possible values are equally likely).

Such a scheme is called a *(k, n) threshold scheme.*

Efficient threshold schemes can be very helpful in the management of cryptographic keys. In order to protect data we can encrypt it, but in order to protect the encryption key we need a different method (further encryptions change the problem rather than solve it). The most secure key management scheme keeps the key in a single, well-guarded location (a computer, a human brain, or a safe). This scheme is highly unreliable since a single misfortune (a computer breakdown, sudden death, or sabotage) can make the information inaccessible. An obvious solution is to store multiple copies of the key at different locations, but this increases the danger of security breaches (computer penetration, betrayal, or human errors). By using a (k, n) threshold scheme with $n = 2k-1$ we get a very robust key management scheme: We can recover the original key even when $\lfloor n/2 \rfloor = k-1$ of the n pieces are destroyed, but our opponents cannot reconstruct the key even when security breaches expose $\lfloor n/2 \rfloor = k-1$ of the remaining k pieces.

In other applications the tradeoff is not between secrecy and reliability, but between safety and convenience of use. Consider, for example, a company that digitally signs all its checks (see RSA [5]). If each executive is given a copy of the company's secret signature key, the system is convenient but easy to misuse. If the cooperation of all the company's executives is necessary in order to sign each check, the system is safe but inconvenient. The standard solution requires at least three signatures per check, and it is easy to implement with a $(3, n)$ threshold scheme. Each executive is given a small magnetic card with one D_i piece, and the company's signature generating device accepts any three of them in order to generate (and later destroy) a temporary copy of the actual signature key D. The device does not contain any secret information and thus it need not be protected against inspection. An unfaithful executive must have at least two accomplices in order to forge the company's signature in this scheme.

Threshold schemes are ideally suited to applications in which a group of mutually suspicious individuals with conflicting interests must cooperate. Ideally we would like the cooperation to be based on mutual consent, but the veto power this mechanism gives to each member can paralyze the activities of the group. By properly choosing the k and n parameters we can give any sufficiently large majority the authority to take some action while giving any sufficiently large minority the power to block it.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

Author's present address: A. Shamir, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139.

This research was supported by the Office of Naval Research under contract no. N00014-76-C-0366.

©1979 ACM 0001-0782/79/1100-0612 $00.75.

2. A Simple (k, n) Threshold Scheme

Our scheme is based on polynomial[1] interpolation: given k points in the 2-dimensional plane $(x_1, y_1), \ldots, (x_k, y_k)$ with distinct x_i's, there is one and only one polynomial $q(x)$ of degree $k-1$ such that $q(x_i) = y_i$ for all i. Without loss of generality, we can assume that the data D is (or can be made) a number. To divide it into pieces D_i, we pick a random $k-1$ degree polynomial $q(x) = a_0 + a_1 x + \ldots a_{k-1} x^{k-1}$ in which $a_0 = D$, and evaluate:

$$D_1 = q(1), \ldots, D_i = q(i), \ldots, D_n = q(n).$$

Given any subset of k of these D_i values (together with their identifying indices), we can find the coefficients of $q(x)$ by interpolation, and then evaluate $D = q(0)$. Knowledge of just $k-1$ of these values, on the other hand, does not suffice in order to calculate D.

To make this claim more precise, we use modular arithmetic instead of real arithmetic. The set of integers modulo a prime number p forms a field in which interpolation is possible. Given an integer valued data D, we pick a prime p which is bigger than both D and n. The coefficients $a_1, \ldots, a_{k-1}$ in $q(x)$ are randomly chosen from a uniform distribution over the integers in $[0, p)$, and the values $D_1, \ldots, D_n$ are computed modulo p.

Let us now assume that $k-1$ of these n pieces are revealed to an opponent. For each candidate value D' in $[0, p)$ he can construct one and only one polynomial $q'(x)$ of degree $k-1$ such that $q'(0) = D'$ and $q'(i) = D_i$ for the $k-1$ given arguments. By construction, these p possible polynomials are equally likely, and thus there is abolutely nothing the opponent can deduce about the real value of D.

Efficient $O(n \log^2 n)$ algorithms for polynomial evaluation and interpolation are discussed in [1] and [3], but even the straightforward quadratic algorithms are fast enough for practical key management schemes. If the number D is long, it is advisable to break it into shorter blocks of bits (which are handled separately) in order to avoid multiprecision arithmetic operations. The blocks cannot be arbitrarily short, since the smallest usable value of p is $n+1$ (there must be at least $n+1$ distinct arguments in $[0, p)$ to evaluate $q(x)$ at). However, this is not a severe limitation since sixteen bit modulus (which can be handled by a cheap sixteen bit arithmetic unit) suffices for applications with up to 64,000 D_i pieces.

Some of the useful properties of this (k, n) threshold scheme (when compared to the mechanical locks and keys solutions) are:

(1) The size of each piece does not exceed the size of the original data.
(2) When k is kept fixed, D_i pieces can be dynamically added or deleted (e.g., when executives join or leave the company) without affecting the other D_i pieces. (A piece is deleted only when a leaving executive makes it completely inaccessible, even to himself.)
(3) It is easy to change the D_i pieces without changing the original data D—all we need is a new polynomial $q(x)$ with the same free term. A frequent change of this type can greatly enhance security since the pieces exposed by security breaches cannot be accumulated unless all of them are values of the same edition of the $q(x)$ polynomial.
(4) By using tuples of polynomial values as D_i pieces, we can get a hierarchical scheme in which the number of pieces needed to determine D depends on their importance. For example, if we give the company's president three values of $q(x)$, each vice-president two values of $q(x)$, and each executive one value of $q(x)$, then a $(3, n)$ threshold scheme enables checks to be signed either by any three executives, or by any two executives one of whom is a vice-president, or by the president alone.

A different (and somewhat less efficient) threshold scheme was recently developed by G.R. Blakley [2].

Received April 1979; revised September 1979

References

1. Aho, A., Hopcroft, J., and Ullman, J. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Mass., 1974.
2. Blakley, G.R. Safeguarding cryptographic keys. Proc. AFIPS 1979 NCC, Vol. 48, Arlington, Va., June 1979, pp. 313-317.
3. Knuth, D. *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms.* Addison-Wesley, Reading, Mass., 1969.
4. Liu, C.L. *Introduction to Combinatorial Mathematics.* McGraw-Hill, New York, 1968.
5. Rivest, R., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM 21,* 2 (Feb. 1978), 120-126.

[1] The polynomials can be replaced by any other collection of functions which are easy to evaluate and to interpolate.