Secret Sharing - Liam Hardiman

What is Secret Sharing?

The goal of secret sharing is dividing a secret S into n pieces (or *shares*) so that no fewer than $k \leq n$ pieces are sufficient for reassembling S. This is called a (k, n)-threshold scheme.

Why is Secret Sharing?

As Shamir puts it, "Threshold schemes are ideally suited to applications in which a group of mutually suspicious individuals with conflicting interests must cooperate."

Some uses include flexibly enforcing consensus while granting veto power and allowing for packets to be sent over a network securely and efficiently.

How is Secret Sharing?

Shamir's Secret Sharing (1979)[1]

Shamir's approach is based on polynomial interpolation. Say we want to share a secret among n people so that no fewer than k of them can recover the secret. Choose a big prime p and say our secret is an element $S \in \mathbb{Z}/p\mathbb{Z}$. Choose random elements $a_1, \ldots, a_{k-1} \in \mathbb{Z}/p\mathbb{Z}$ and set

$$p(x) = S + a_1 x + \cdots + a_{k-1} x^{k-1}.$$

Issue to person i the share $D_i = (i, p(i))$, $1 \leq i \leq n$. If m people come together with their shares, $(i_j, p(i_j))$, $1 \leq j \leq m$ then they know that

$$\begin{array}{ccccccccccc} S & + & a_1 \cdot i_1 & + & a_2 \cdot (i_1)^2 & + & \cdots & + & a_{k-1}(i_1)^{k-1} & = & p(i_1) \\ S & + & a_2 \cdot i_2 & + & a_2 \cdot (i_2)^2 & + & \cdots & + & a_{k-1}(i_2)^{k-1} & = & p(i_2) \\ & & & & & & \vdots & & & & \\ S & + & a_1 \cdot i_m & + & a_2 \cdot (i_m)^2 & + & \cdots & + & a_{k-1}(i_m)^{k-1} & = & p(i_m) \end{array}$$

This represents a system of m equations in the k unknowns, S, a_1, $\ldots$, a_{k-1}. Elementary linear algebra tells us that this system has a unique solution if and only if $m \geq k$. That is, we need at least k shares in order to uniquely determine S.

Main Disadvantage of Shamir's Scheme

Each share is just as large as the secret. n shares means n-fold blowup in storage.

Improvements by Rabin and Krawczyk

Rabin - Information Dispersal (1989)[2]

We can split a file F into n pieces so that any $k \leq n$ pieces are sufficient for reconstructing F, with the added feature that each piece has size roughly $|F|/k$. With n shares, each of size roughly $|F|/k$ we get a blowup of around $\frac{n}{k}$, but this can be chosen to be close to 1. *Secrecy isn't the objective.*

[1] A. Shamir, "How to share a secret". Commun. ACM 22(11), 612613 (1979)

[2] M. O. Rabin, "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance". In: Journal of the ACM, vol. 36, iss. 2, 1989, pp. 335-348

Let's share a file F with 15 people so that any 8 of them can reassemble it. Say F is composed of 800 bytes, $F = b_1, \dots, b_{800}$, where each b_i is an integer $0 \le b_i \le 255$. Let p be a prime bigger than 255. Break the file into 8-byte blocks

$$F = (b_1, \dots, b_8), (b_9, \dots, b_{16}), \dots, (b_{793}, b_{794}, \dots, b_{800}) = f_1, f_2, \dots, f_{100}$$

where $f_1 = (b_1, \dots, b_8)$ is the first block, and so on. The idea is to compress each 8-byte block into a single number in such a way that any 8 compressed blocks can be used to recover the original block.

To create 15 shares, choose 15 vectors, a_j, $1 \le j \le 15$ in $(\mathbb{Z}/p\mathbb{Z})^8$ so that any subset of 8 different vectors is linearly independent (how this can be done is a non-obvious, but still elementary exercise in linear algebra). To compute the j-th share, we compress each block of F by calculating the dot product $F_{ji} := a_j \cdot f_i \pmod p$ for each $1 \le i \le 100$.

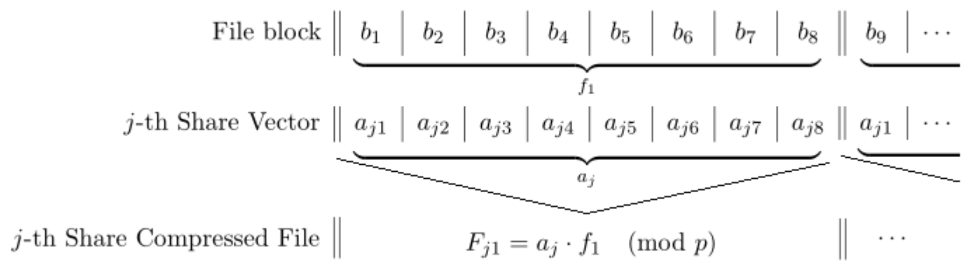

Figure 1: Compressing F into a share.

This creates a compressed file, $F_j = F_{j1}, \dots, F_{j(100)}$ consisting of 100 elements in $\mathbb{Z}/p\mathbb{Z}$. We let the pair $S_j = (a_j, F_j)$, $1 \le j \le 15$, be the j-th share of our original file.

Say we're given 8 shares, $S_j = (a_j, F_j)$, $1 \le j \le 8$. When we look at the first entry in each compressed file, $F_{j,1}$, we obtain this matrix equation

$$\begin{bmatrix} \text{—} & a_1 & \text{—} \\ & \vdots & \\ \text{—} & a_8 & \text{—} \end{bmatrix} \begin{bmatrix} b_1 \\ \vdots \\ b_8 \end{bmatrix} = \begin{bmatrix} F_{1,1} \\ \vdots \\ F_{8,1} \end{bmatrix}.$$

Since we have 8 shares, the matrix with rows a_j is actually square. Since the a_j were chosen to be linearly independent, we can invert that matrix to solve for $[b_1 \ \cdots \ b_8]^t$. We can use the same matrix to recover the other 99 blocks as well.

Krawczyk - Secret Sharing with Short Shares (1994)[3]

Combine the ideas of Shamir and Rabin. We want to set up a (k, n) threshold scheme with secret S. We start by encrypting S with some secure cipher using key K, $E = Enc(S, K)$. Using Rabin's information dispersal method, partition E into n shares, $E_1, \dots, E_n$ so that any k of them can rebuild E. Using Shamir's method, generate n shares of the key, $K_1, \dots, K_n$ so that any k of them can rebuild K. Send person j the pair (E_j, K_j).

Now when k people come together, they can reassemble E through Rabin's matrix inversion method and K through polynomial interpolation. The k participants can then decrypt E with K to obtain S.

While Shamir's method didn't shrink the size of the key shares, Rabin's method shrinks the size of the secret shares.

[3] H. Krawczyk, "Secret Sharing Made Short". In: Stinson D.R. (eds) Advances in Cryptology CRYPTO 93. CRYPTO 1993. Lecture Notes in Computer Science, vol 773.