Secret Sharing and Threshold Decryption

The goal of secret-sharing is to divide a secret S into n pieces $S_1, \ldots, S_n$ such that any $m+1$ pieces are sufficient to reconstruct S, but any m pieces give no information about S.

Secret-sharing has applications in online voting, digital cash and threshold decryption, which we will see later in this lecture. There are more direct applications: You may have some sensitive data in files at work that you encrypt using a one-time private key, e.g. about student grades. It may be important for other people at work to access this information in the case where you are unavailable (or worse, if you are available but have lost the key!). In this case, you could share the key S into e.g. $n = 5$ shares, and give them to a five people at work. If any (e.g.) $m + 1 = 3$ of the people decide to use their keys to access your data, they can do so even if you are not there. Sharing is useful in this case because if you gave the key to these people directly, any one of them could wrongfully use their keys to access your data. It may be very hard to detect this. Using secret-sharing makes dishonesty less likely and it also makes it hard for one person to cheat without being discovered. If $m + 1 = 3$, any person that wants to cheat needs two accomplices, and all three could be witnesses against the others.

Lets consider a very simple secret sharing scheme with $n = m + 1$. Let S be the secret to be shared. Assume $S < p$ for some prime p. Choose $n - 1$ random values $S_1, \ldots, S_{n-1}$ uniformly from $\mathbb{Z}_p$. Then set

$$S_n = S - \sum_{i=1}^{n-1} S_i (\mathrm{mod}\ \ p)$$

from which it follows that

$$S = \sum_{i=1}^{n} S_i (\mathrm{mod}\ \ p)$$

So we can recover S if we have all n shares by simple addition. What about if we have $n - 1$ shares? Clearly if we have $S_1, \ldots, S_{n-1}$ they give us no information about S.

Now suppose we have $n - 1$ values including S_n (which means all but one of $S_1, \ldots, S_{n-1}$). The first $n - 2$ of these values (which are randomly chosen) are independent. The last value S_n depends on these values, but also on the missing S_i. Each distinct value of S_i produces a distinct value of S_n. Therefore S_n has the uniform distribution, and is independent of the first $n - 2$ S_i's. So these $n - 1$ numbers appear to us as independent random numbers, and give no information about S.

Shamir Secret Sharing

More generally, we would like to split a secret S into n pieces so that $m + 1 \leq n$ pieces are sufficient to reconstruct it. Using $m < n$ makes the sharing tolerant of loss of a part of the key, or

non-cooperation by someone who has a share. It is still safe because we can set m to be large, e.g. $n/2$, so that more than half of the people with shares would have to collude to improperly decode the secret.

Assume $S < p$ for some prime $p > n$. Now choose $m \leq n$ values $a_1, \ldots, a_m$ independently and uniformly at random from $\mathbb{Z}_p^*$. Finally set $a_0 = S$, the secret. The coefficients a_i define a polynomial:

$$a(x) = a_m x^m + a_m x^{m-1} + \cdots + a_1 x + a_0$$

Now choose n distinct values $r_1, \ldots, r_n$ at random from $\mathbb{Z}_p^*$. You can do this by choosing independently and uniformly, but discarding repeats. You must have $p > n$ to get enough values. Then set:

$$S_i = (r_i, p(r_i)) \quad \text{for } i = 1, \ldots, n$$

to get the n shares. The shares consist of the values of the polynomial at specified values of $x = r_1, \ldots, r_n$. Polynomials have the following property:

Interpolation property Given $m + 1$ pairs (x_i, y_i) with x_i's all distinct, there is a unique polynomial $a(x)$ of degree m passing through all the points. This polynomial can be effectively computed from the pairs (x_i, y_i):

$$a(x) = \sum_{i=1}^{m+1} y_i L_i(x)$$

where $L_i(x)$ is the Lagrange polynomial:

$$L_i(x) = \frac{\prod_{j \neq i}(x - x_j)}{\prod_{j \neq i}(x_i - x_j)}$$

which has value 1 at x_i and zero at every other x_j. This method is called Lagrange interpolation and is very efficient.

The Lagrange interpolation formula involves only addition, multiplication, and division by non-zero values assuming the x_i are distinct. Therefore since the values (x_i, y_i) all lie in $\mathbb{Z}_p$, the coefficients of $a(x)$ are also elements of $\mathbb{Z}_p$. It follows from the interpolation property that:

Lemma: Any $m + 1$ shares suffice to determine the secret The interpolation property implies that we can uniquely reconstruct $a(x)$ from any $m + 1$ pairs (x_i, y_i). Once we have $a(x)$, we certainly know $a_0 = S$. In fact since we need only the constant term of $a(x)$, the Lagrange formula gives us

$$a_0 = a(0) = \sum_{i=1}^{m+1} y_i L_i(0)$$

Lemma: Any m shares give no information about the secret If we have m pairs (x_i, y_i), we can add one more which is $(0, v)$ for any $v \in \mathbb{Z}_p$. Then we would have $m + 1$ points which would uniquely define a polynomial $a(x)$. But notice that $a_0 = v$. Thus, we can choose any value for the secret S and still be consistent with the m values we are given. Not only that, but every such choice is equally likely (because $m + 1$ values define a unique polynomial with a unique choice for the random coefficients). Therefore m values give us no information about the secret S.

Threshold Decryption

As you might imagine, sharing keys like this can be a complicated and time-consuming process. There is the process of creating the shares, then making sure they do to the right people etc.. Once enough people decide to combine their shares, the secret will be reconstructed and isnt secret anymore. What if the secret being shared were a secret key d for a public-key crypto-system? Intuitively, you might expect the holders of the share to be able to decrypt messages collaboratively. Can this be done, and how difficult is it to do?

The answer is yes, for both RSA and El-Gamal encryption. And the method is surprisingly simple: users simple decode the message in the usual way using their piece of the shared decryption key. Then the partial decryptions from any $m+1$ users can be combined to recover the original message. Lagrange interpolation is used in an interesting way.

Suppose (N, d) is a secret RSA key and (N, e) is the corresponding public key. Assume that d has been shared using Shamir secret-sharing into n shares $d_1, \ldots, d_n$ such that any $m+1$ suffice to reconstruct d. The shares are distributed to n people.

Let $M = T^e (\mathrm{mod}\ \ N)$ be the encryption of a message T using RSA. We send M to all the people, and ask them to decrypt it using their share of d. If at least $m+1$ people cooperate, we obtain $m+1$ values:

$$v_i = M^{d_i} (\mathrm{mod}\ \ N)$$

Going back to the Lagrange formula, recall that the secret is a linear combination of the polynomial's values:

$$d = a_0 = a(0) = \sum_{i=1}^{m+1} y_i L_i(0)$$

which suggests that we apply these weights as exponents:

$$\prod_i v_i^{L_i(0)} = \prod_i M^{d_i L_i(0)} = M^{\sum_i d_i L_i(0)} = M^d \ (\mathrm{mod}\ \ N) = T$$

So we can recover the message T from M as long as at least $m+1$ share-holders participate. They do not expose their keys by participating, so we can repeat this process as often as we want. (Note that the values $L_i(0)$ depend on the particular set of x_i's which we get back).

A very similar idea works for El-Gamal encryption. Let S be a secret key which is shared among n people as $S_1, \ldots, S_n$. The corresponding public key is $(p, g, h = g^S (mod p))$. A message T would be encrypted as $(g^r, Th^r)(\mathrm{mod}\ \ p)$. How should each share-holder "decrypt" the message, and how would you combine those partial decryptions?