Secret Sharing

Arianna Novaro

24.10.2014

Secret sharing

- Method for dividing a secret S into n pieces of information (*shares* or *shadows*) $s_1, s_2, \ldots, s_n$, where each s_i does not reveal anything about S.
- The shares can be distributed among a group of *participants* or kept by a single person (depending on the purpose of the computation).
- With k (or more) shares it is possible to easily recover the secret.
- All possible values of S are equally likely for someone with knowledge of only x shares (with $x < k$).

The (k, n)-threshold scheme

- A particular type of *access structure*: a description of which subsets of participants are allowed to recover the secret.
- *Monotone*: if a subset W of the set of participants is allowed to recover the secret, any superset of W is also allowed.
- n : total number of participants (shares)
- k : minimal number of participants (shares) needed to recover the secret.

Properties
(*) Possibility to add or remove any s_i without affecting the other shares, once k is fixed (floating number of participants);
(*) Possibility to change the shares without changing the secret (for security reasons);
(*) Possibility to get a hierarchical scheme giving a different number of shares to each participant, depending on his/her importance.

Protocol
Since any information can be expressed as a number (in our case: the secret S), polynomials can be used as a tool to achieve secret sharing with a (k, n)-threshold scheme.
In fact, any polynomial $q(x)$ of degree $k-1$ can be determined if k points $(x_1, y_1), \ldots, (x_k, y_k)$ in the two dimensional plane are given.
A polynomial $q(x)$ of degree $k-1$ is chosen:

$$q(x) = a_0 + a_1 x + \cdots + a_{k-1} x^{k-1} \tag{1}$$

We set the free term to be equal to the secret, i. e. $S = a_0$; thus, $S = q(0)$. We have that $s_i = q(i)$.
This means that pair(s) of the form $(i, q(i))$ will be given to each participant, and with k of them it will be possible to recover the secret.

Lagrange Interpolation method

$$q(x) = \sum_{j=1}^{k} y_j p_j(x) \tag{2}$$

where

$$p_j(x) = \prod_{i=1; i\neq j}^{k} \frac{(x - x_i)}{(x_j - x_i)} \tag{3}$$

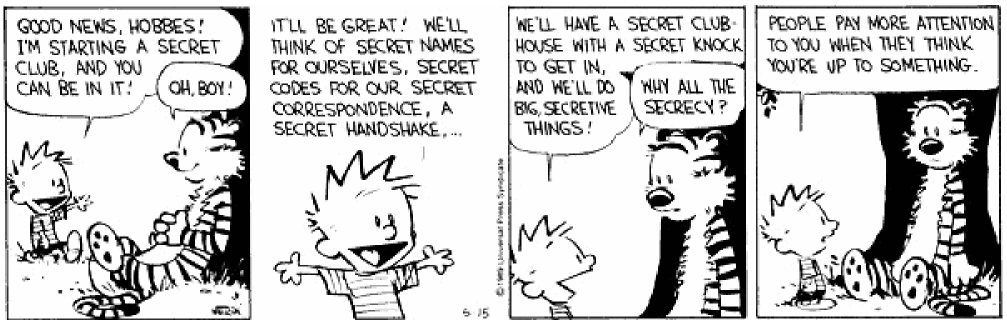
with $j = 1, \ldots, k$.

With any set of k points (x_i, y_i) it is possible to efficiently find the coefficients of $q(x)$ by interpolation and then to evaluate $S = q(0)$. Knowledge of only $k - 1$ points is not enough to find the curve represented by the polynomial.

Application of secret sharing

Multi-party computation: The main idea is to combine confidential information that comes from several sources (in order to store it or to do computations on it), even when the parties do not trust each other.
Parties (players) : $P_1, \ldots, P_n$ (where every P_i owns a secret input s_i).
The goal is to compute *securely* a function $f : (s_1, \ldots, s_n)$; that is:
- the correct value of y is output (correctness);
- y is the *only* new information that is released (privacy);

Example: **voting**.
During elections it is mandatory to have a secure protocol for adding the votes of the participants, without anyone discovering the votes of the others and in a way with which it is possible to detect whether someone has deviated from the protocol (*secure addition*).


GOOD NEWS, HOBBES! I'M STARTING A SECRET CLUB, AND YOU CAN BE IN IT!
OH, BOY!
IT'LL BE GREAT! WE'LL THINK OF SECRET NAMES FOR OURSELVES, SECRET CODES FOR OUR SECRET CORRESPONDENCE, A SECRET HANDSHAKE,...
WE'LL HAVE A SECRET CLUB-HOUSE WITH A SECRET KNOCK TO GET IN, AND WE'LL DO BIG, SECRETIVE THINGS!
WHY ALL THE SECRECY?
PEOPLE PAY MORE ATTENTION TO YOU WHEN THEY THINK YOU'RE UP TO SOMETHING.

References

- Cramer R., Damgaard I., Nielsen J. B., *Secure Multiparty Computation and Secret Sharing*, Draft: 7-18. (available at: http://daimi.au.dk/ ivan/MPCbook.pdf)
- Shamir A. (1979), "How to Share a Secret", *Communications of the ACM*, 22, 11: 612-613.