Shamir's Secret Sharing

From Wikipedia, the free encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Català
Čeština
Deutsch
Español
Français
Magyar
Русский
Edit links

This article **may be too technical for most readers to understand**. Please help improve this article to make it understandable to non-experts, without removing the technical details. The talk page may contain suggestions. *(March 2014)*

Shamir's Secret Sharing is an algorithm in cryptography created by Adi Shamir. It is a form of secret sharing, where a secret is divided into parts, giving each participant its own unique part, where some of the parts or all of them are needed in order to reconstruct the secret.

Counting on all participants to combine together the secret might be impractical, and therefore sometimes the *threshold scheme* is used where any k of the parts are sufficient to reconstruct the original secret.

Mathematical definition [edit]

The goal is to divide secret S (e.g., a safe combination) into n pieces of data $S_1, \ldots, S_n$ in such a way that:

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

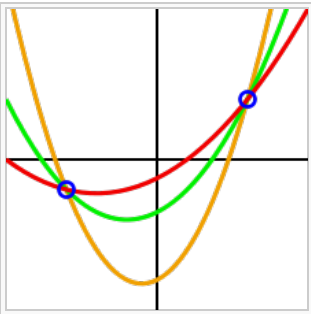
This scheme is called (k, n) threshold scheme. If $k = n$ then all participants are required to reconstruct the secret.

Shamir's secret-sharing scheme [edit]

The essential idea of Adi Shamir's threshold scheme is that 2 points are sufficient to define a line, 3 points are sufficient to define a parabola, 4 points to define a cubic curve and so forth. That is, it takes k points to define a polynomial of degree $k-1$.

Suppose we want to use a (k, n) threshold scheme to share our secret S, without loss of generality assumed to be an element in a finite field F of size P where $0 < k \leq n < P; S < P$ and P is a prime number.

Choose at random $k-1$ positive integers $a_1, \cdots, a_{k-1}$ with $a_i < P$, and let $a_0 = S$. Build the polynomial $f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{k-1} x^{k-1}$. Let us construct any n points out of it, for instance set $i = 1, \cdots, n$ to retrieve $(i, f(i))$. Every participant is given a point (an integer input to the polynomial, and the corresponding integer output). Given any subset of k of these pairs, we can find the coefficients of the polynomial using interpolation. The secret is the constant term a_0.



One can draw an infinite number of polynomials of degree 2 through 2 points. 3 points are required to define a unique polynomial of degree 2. This image is for illustration purposes only—Shamir's scheme uses polynomials over a finite field, not representable on a 2-dimensional plane.

Usage [edit]

Example [edit]

The following example illustrates the basic idea. Note, however, that calculations in the example are done using integer arithmetic rather than using finite field arithmetic. Therefore the example below does not provide perfect secrecy and is not a true example of Shamir's scheme. So we'll explain this problem and show the right way to implement it (using finite field arithmetic).

Preparation [edit]

Suppose that our secret is 1234 $(S = 1234)$.

We wish to divide the secret into 6 parts $(n = 6)$, where any subset of 3 parts $(k = 3)$ is sufficient to reconstruct the secret. At

random we obtain two ($k-1$) numbers: 166 and 94.

$$(a_0 = 1234; a_1 = 166; a_2 = 94)$$

Our polynomial to produce secret shares (points) is therefore:

$$f(x) = 1234 + 166x + 94x^2$$

We construct 6 points $D_{x-1} = (x, f(x))$ from the polynomial:

$$D_0 = (1, 1494); D_1 = (2, 1942); D_2 = (3, 2578); D_3 = (4, 3402); D_4 = (5, 4414); D_5 = (6, 5614)$$

We give each participant a different single point (both x and $f(x)$). Because we use D_{x-1} instead of D_x the points start from $(1, f(1))$ and not $(0, f(0))$. This is necessary because if one would have $(0, f(0))$ he would also know the secret ($S = f(0)$)

Reconstruction [edit]

In order to reconstruct the secret any 3 points will be enough.

Let us consider $(x_0, y_0) = (2, 1942); (x_1, y_1) = (4, 3402); (x_2, y_2) = (5, 4414)$

We will compute Lagrange basis polynomials:

$$\ell_0 = \frac{x - x_1}{x_0 - x_1} \cdot \frac{x - x_2}{x_0 - x_2} = \frac{x-4}{2-4} \cdot \frac{x-5}{2-5} = \frac{1}{6}x^2 - \frac{3}{2}x + \frac{10}{3}$$

$$\ell_1 = \frac{x - x_0}{x_1 - x_0} \cdot \frac{x - x_2}{x_1 - x_2} = \frac{x-2}{4-2} \cdot \frac{x-5}{4-5} = -\frac{1}{2}x^2 + \frac{7}{2}x - 5$$

$$\ell_2 = \frac{x - x_0}{x_2 - x_0} \cdot \frac{x - x_1}{x_2 - x_1} = \frac{x-2}{5-2} \cdot \frac{x-4}{5-4} = \frac{1}{3}x^2 - 2x + \frac{8}{3}$$

Therefore

$$f(x) = \sum_{j=0}^{2} y_j \cdot \ell_j(x)$$

$$= 1234 + 166x + 94x^2$$

Recall that the secret is the free coefficient, which means that $S = 1234$, and we are done.

Problem [edit]

Although this method works fine, there is a security problem: Eve wins a lot of information about S with every D_i that she finds.

Suppose that she finds the 2 points $D_0 = (1, 1494)$ and $D_1 = (2, 1942)$, she still doesn't have $k = 3$ points so in theory she shouldn't have won anymore info about S. But she combines the info from the 2 points with the public info:
$n = 6, k = 3, f(x) = a_0 + a_1 x + \cdots + a_{k-1}x^{k-1}, a_0 = S, a_i \in \mathbb{N}$ and she :

i. fills the $f(x)$-formula with S and the value of k : $f(x) = S + a_1 x + \ldots + a_{3-1}x^{3-1} \Rightarrow f(x) = S + a_1 x + a_2 x^2$
ii. fills (i) with the values of D_0's x and $f(x)$: $1494 = S + a_1 1 + a_2 1^2 \Rightarrow 1494 = S + a_1 + a_2$
iii. fills (i) with the values of D_1's x and $f(x)$: $1942 = S + a_1 2 + a_2 2^2 \Rightarrow 1942 = S + 2a_1 + 4a_2$
iv. does (iii)-(ii): $(1942 - 1494) = (S - S) + (2a_1 - a_1) + (4a_2 - a_2) \Rightarrow 448 = a_1 + 3a_2$ and rewrites this as $a_1 = 448 - 3a_2$
v. knows that $a_2 \in \mathbb{N}$ so she starts replacing a_2 in (iv) with 0, 1, 2, 3, ... to find all possible values for a_1:
   - $a_2 = 0 \rightarrow a_1 = 448 - 3 \times 0 = 448$
   - $a_2 = 1 \rightarrow a_1 = 448 - 3 \times 1 = 445$
   - $a_2 = 2 \rightarrow a_1 = 448 - 3 \times 2 = 442$
   - ...
   - $a_2 = 148 \rightarrow a_1 = 448 - 3 \times 148 = 4$
   - $a_2 = 149 \rightarrow a_1 = 448 - 3 \times 149 = 1$

   After $a_2 = 149$ she stops because she reasons that if she continues she would get negative values for a_1 (which is impossible because $a_1 \in \mathbb{N}$), she can now conclude $a_2 \in [0, 1, \ldots, 148, 149]$
vi. replaces a_1 by (iv) in (ii): $1494 = S + (448 - 3a_2) + a_2 \Rightarrow S = 1046 + 2a_2$
vii. replaces in (vi) a_2 by the values found in (v) so she gets $S \in [1046 + 2 \times 0, 1046 + 2 \times 1, \ldots, 1046 + 2 \times 148, 1046 + 2 \times 149]$ which leads her to the information: $S \in [1046, 1048, \ldots, 1342, 1344]$. She now only has 150 numbers to guess from instead of an infinite number of natural numbers.

Solution [edit]

This problem can be fixed by using finite field arithmetic in a field of size $p \in \mathbb{P} : p > S, p > n$.

This is in practice only a small change, it just means that we should choose a prime p that is bigger than the number of participants and every a_i (including $a_0 = S$) and we have to calculate the points as $(x, f(x) \pmod p)$ instead of $(x, f(x))$.

Since everyone who receives a point also has to know the value of p so it may be considered to be publicly known. Therefore, one should select a value for p that is neither too low nor too high.

Low values of p are risky because Eve knows $p > S \Rightarrow S \in [0, 1, \ldots, p-2, p-1]$, so the lower one sets p, the lower the number of possible values Eve has to guess from to get S.

High values of p are risky because Eve knows that the chance for $f(x) \pmod p = f(x)$ increases with a higher p, and she can use the procedure from the original problem to guess S (although now, instead of being sure of the 150 possible values, they merely have an increased chance of being valid compared to the other natural numbers).

For this example we choose $p = 1613$, so our polynomial becomes $f(x) = 1234 + 166x + 94x^2 \mod 1613$ which gives the points: $(1, 1494); (2, 329); (3, 965); (4, 176); (5, 1188); (6, 775)$

This time Eve doesn't win any info when she finds a D_x (until she has k points).

Suppose again Eve again finds $D_0 = (1, 1494)$ and $D_1 = (2, 329)$, this time the public info is:
$n = 6, k = 3, p = 1613, f(x) = a_0 + a_1 x + \cdots + a_{k-1} x^{k-1} \mod p, a_0 = S, a_i \in \mathbb{N}$ so she:

i. fills the $f(x)$-formula with S and the value of k and p:
$f(x) = S + a_1 x + \cdots + a_{3-1} x^{3-1} \mod 1613 \Rightarrow f(x) = S + a_1 x + a_2 x^2 - 1613 m_x : m_x \in \mathbb{N}$

ii. fills (i) with the values of D_0's x and
$f(x) : 1494 = S + a_1 1 + a_2 1^2 - 1613 m_1 \Rightarrow 1494 = S + a_1 + a_2 - 1613 m_1$

iii. fills (i) with the values of D_1's x and
$f(x) : 1942 = S + a_1 2 + a_2 2^2 - 1613 m_2 \Rightarrow 1942 = S + 2a_1 + 4a_2 - 1613 m_2$

iv. does (iii)-(ii):
$(1942 - 1494) = (S - S) + (2a_1 - a_1) + (4a_2 - a_2) + (1613 m_1 - 1613 m_2) \Rightarrow 448 = a_1 + 3a_2 + 1613(m_1 - m_2)$
and rewrites this as $a_1 = 448 - 3a_2 - 1613(m_1 - m_2)$

v. knows that $a_2 \in \mathbb{N}$ so she starts replacing a_2 in (iv) with 0, 1, 2, 3, ... to find all possible values for a_1:

- $a_2 = 0 \rightarrow a_1 = 448 - 3 \times 0 - 1613(m_1 - m_2) = 448 - 1613(m_1 - m_2)$
- $a_2 = 1 \rightarrow a_1 = 448 - 3 \times 1 - 1613(m_1 - m_2) = 445 - 1613(m_1 - m_2)$
- $a_2 = 2 \rightarrow a_1 = 448 - 3 \times 2 - 1613(m_1 - m_2) = 442 - 1613(m_1 - m_2)$
- ...

This time she can't stop because $(m_1 - m_2)$ could be any integer (even negative if $m_2 > m_1$) so there are an infinite amount of possible values for a_1. She knows that $[448, 445, 442, ...]$ always decreases by 3 so if 1613 was divisible by 3 she could conclude $a_1 \in [1, 4, 7, ...]$ but because it's prime she can't even conclude that and so she didn't win any information.

Javascript example [edit]

```
var prime = 257;

/* Split number into the shares */
function split(number, available, needed) {
    var coef = [number, 166, 94], x, exp, c, accum, shares = [];
    /* Normally, we use the line:
     * for(c = 1, coef[0] = number; c < needed; c++) coef[c] = Math.floor(Math.random() * (prime  -
1));
     * where (prime - 1) is the maximum allowable value.
     * However, to follow this example, we hardcode the values:
     * coef = [number, 166, 94];
     * For production, replace the hardcoded value with the random loop
     * For each share that is requested to be available, run through the formula plugging the
corresponding coefficient
     * The result is f(x), where x is the byte we are sharing (in the example, 1234)
     */
    for(x = 1; x <= available; x++) {
        /* coef = [1234, 166, 94] which is 1234x^0 + 166x^1 + 94x^2 */
        for(exp = 1, accum = coef[0]; exp < needed; exp++) accum = (accum + (coef[exp] * (Math.pow(x,
exp) % prime) % prime)) % prime;
        /* Store values as (1, 132), (2, 66), (3, 188), (4, 241), (5, 225) (6, 140) */
        shares[x - 1] = [x, accum];
    }
    return shares;
}

/* Gives the decomposition of the gcd of a and b.  Returns [x,y,z] such that x = gcd(a,b) and y*a + z*b
= x */
function gcdD(a,b) {
    if (b == 0) return [a, 1, 0];
    else {
        var n = Math.floor(a/b), c = a % b, r = gcdD(b,c);
        return [r[0], r[2], r[1]-r[2]*n];
    }
}

/* Gives the multiplicative inverse of k mod prime.  In other words (k * modInverse(k)) % prime = 1 for
all prime > k >= 1  */
function modInverse(k) {
    k = k % prime;
    var r = (k < 0) ? -gcdD(prime,-k)[2] : gcdD(prime,k)[2];
    return (prime + r) % prime;
}

/* Join the shares into a number */
function join(shares) {
    var accum, count, formula, startposition, nextposition, value, numerator, denominator;
    for(formula = accum = 0; formula < shares.length; formula++) {
        /* Multiply the numerator across the top and denominators across the bottom to do Lagrange's
interpolation
         * Result is x0(2), x1(4), x2(5) -> -4*-5 and (2-4=-2)(2-5=-3), etc for l0, l1, l2...
         */
        for(count = 0, numerator = denominator = 1; count < shares.length; count++) {
            if(formula == count) continue; // If not the same value
            startposition = shares[formula][0];
            nextposition = shares[count][0];
            numerator = (numerator * -nextposition) % prime;
            denominator = (denominator * (startposition - nextposition)) % prime;
        }
        value = shares[formula][1];
        accum = (prime + accum + (value * numerator * modInverse(denominator))) % prime;
    }
    return accum;
}

var sh = split(129, 6, 3) /* split the secret value 129 into 6 components - at least 3 of which will be
needed to figure out the secret value */
var newshares = [sh[1], sh[3], sh[4]]; /* pick any selection of 3 shared keys from sh */

alert(join(newshares));
```

Properties [edit]

Some of the useful properties of Shamir's (k, n) threshold scheme are:

1. **Secure**: Information theoretic security.
2. **Minimal**: The size of each piece does not exceed the size of the original data.
3. **Extensible**: When k is kept fixed, D_i pieces can be dynamically added or deleted without affecting the other pieces.
4. **Dynamic**: Security can be easily enhanced without changing the secret, but by changing the polynomial occasionally (keeping the same free term) and constructing new shares to the participants.

5. **Flexible**: In organizations where hierarchy is important, we can supply each participant different number of pieces according to their importance inside the organization. For instance, the president can unlock the safe alone, whereas 3 secretaries are required together to unlock it.

See also [edit]

- Secret sharing
- Lagrange polynomial
- Homomorphic secret sharing - A simplistic decentralized voting protocol.
- Two-man rule
- Partial Password

References [edit]

- Shamir, Adi (1979), "How to share a secret", *Communications of the ACM* **22** (11): 612–613, doi:10.1145/359168.359176.
- Liu, C. L. (1968), *Introduction to Combinatorial Mathematics*, New York: McGraw-Hill.
- Dawson, E.; Donovan, D. (1994), "The breadth of Shamir's secret-sharing scheme", *Computers & Security* **13**: 69–78, doi:10.1016/0167-4048(94)90097-3.
- Knuth, D. E. (1997), *The Art of Computer Programming*, II: Seminumerical Algorithms (3rd ed.), Addison-Wesley, p. 505.

External links [edit]

- A proper Javascript implementation of Shamir's secret sharing scheme with open source (MIT) license
- ssss: An open source (GPL) implementation of Shamir's Scheme with online demo
- An open source (GPL) perl implementation of Shamir's Secret Sharing
- Secret Sharp: An open source (GPL) implementation of Shamir's Scheme for windows
- Christophe David's web based implementation of Shamir's scheme 'How to share a Secret'
- Shamir's Secret Sharing in Java : An open source (LGPL) implementation of Shamir's scheme in Java
- An open source implementation of the Shamir's Secret Sharing as open Web application, augmented by additional security features
- libgfshare: a secret sharing library in GF(2**8), opensource (MIT)
- Web implementation of Shamir's method
- Java library implementation of multiple secret sharing methods, opensource(LGPLv2)

Categories: Secret sharing | Information-theoretically secure algorithms