Shamir's Secret Sharing

- https://arnaucube.com
- https://github.com/arnaucube
- https://twitter.com/arnaucube

2019-07-05

Intro

- I'm not an expert on the field, neither a mathematician. Just an engineer with interest for cryptography
- Short talk (15 min), with the objective to make a practical introduction to the Shamir's Secret Sharing algorithm
- Is not a talk about mathematical demostrations, is a talk with the objective to get the basic notions to be able to do a practical implementation of the algorithm
- After the talk, we will do a practical workshop to implement the concepts. We can offer support for Go, Rust, Python and Nodejs (you can choose any other language, but we will not be able to help)

- Cryptographic algorithm
- Created by Adi Shamir, in 1979
  - also known by the RSA cryptosystem
    - explained in few months ago in a similar talk: https://github.com/arnaucube/slides/rsa

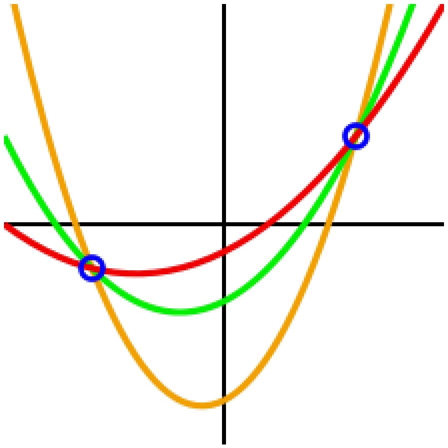
What's this about?

- imagine having a password that you want to share with 5 persons, in a way that they need to join their parts to get the original password
- take the password, split it in 5 parts, and give one part to each one
- when they need to recover it, they just need to get together, put all the pieces and recover the password (the `secret`)
- this, has the problem that if a person looses its part, the secret will not be recovered anymore.. luckly we have a solution here:

- Shamir's Secret Sharing:
  - from a secret to be shared, we generate 5 parts, but we can specify a number of parts that are needed to recover the secret
  - so for example, we generate 5 parts, where we will need only 3 of that 5 parts to recover the secret, and the order doesn't matter
  - we have the ability to define the thresholds of M parts to be created, and N parts to be able the recover

- 2 points are sufficient to define a line
- 3 points are sufficient to define a parabola
- 4 points are sufficient to define a cubic curve
- K points are suficient to define a polynomial of degree $k - 1$

We can create infinity of polynomials of degree 2, that goes through 2 points, but with 3 points, we can define a polynomial of degree 2 unique.



Naming

- `s`: secret
- `m`: number of parts to be created
- `n`: number of minimum parts necessary to recover the secret
- `p`: random prime number, the Finite Field will be over that value

Secret generation

- we want that are necessary n parts of m to recover s
  - where $n < m$
- need to create a polynomial of degree $n-1$
  $f(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3 + ... + +\alpha_{n-1} x^{n-1}$
- where α_0 is the secret s
- α_i are random values that build the polynomial
  *where α_0 is the secret to share, and α_i are the random values inside the $FiniteField$

$$f(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3 + ... + +\alpha_{n-1} x^{n-1}$$

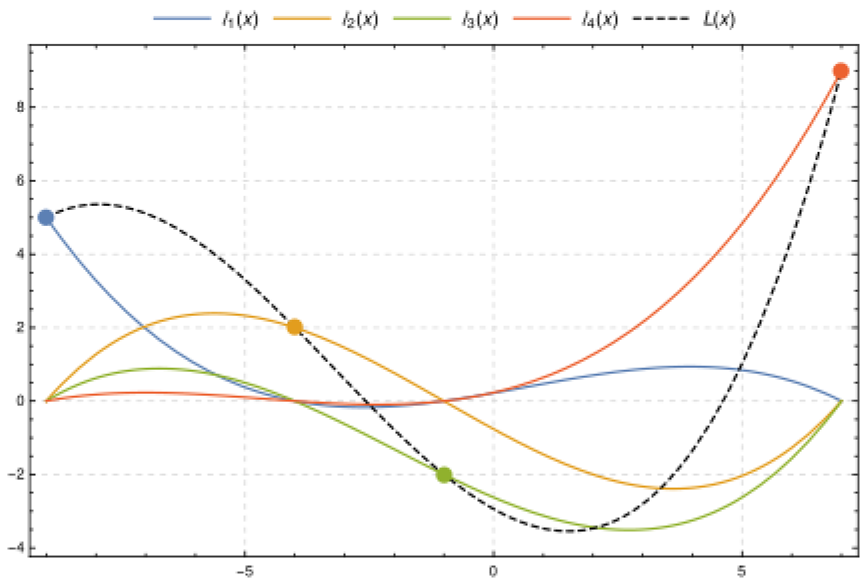
- the packets that we will generate are $P = (x, f(x))$
  - where x is each one of the values between 1 and m
    - $P_1 = (1, f(1))$
    - $P_2 = (2, f(2))$
    - $P_3 = (3, f(3))$
    - ...
    - $P_m = (m, f(m))$

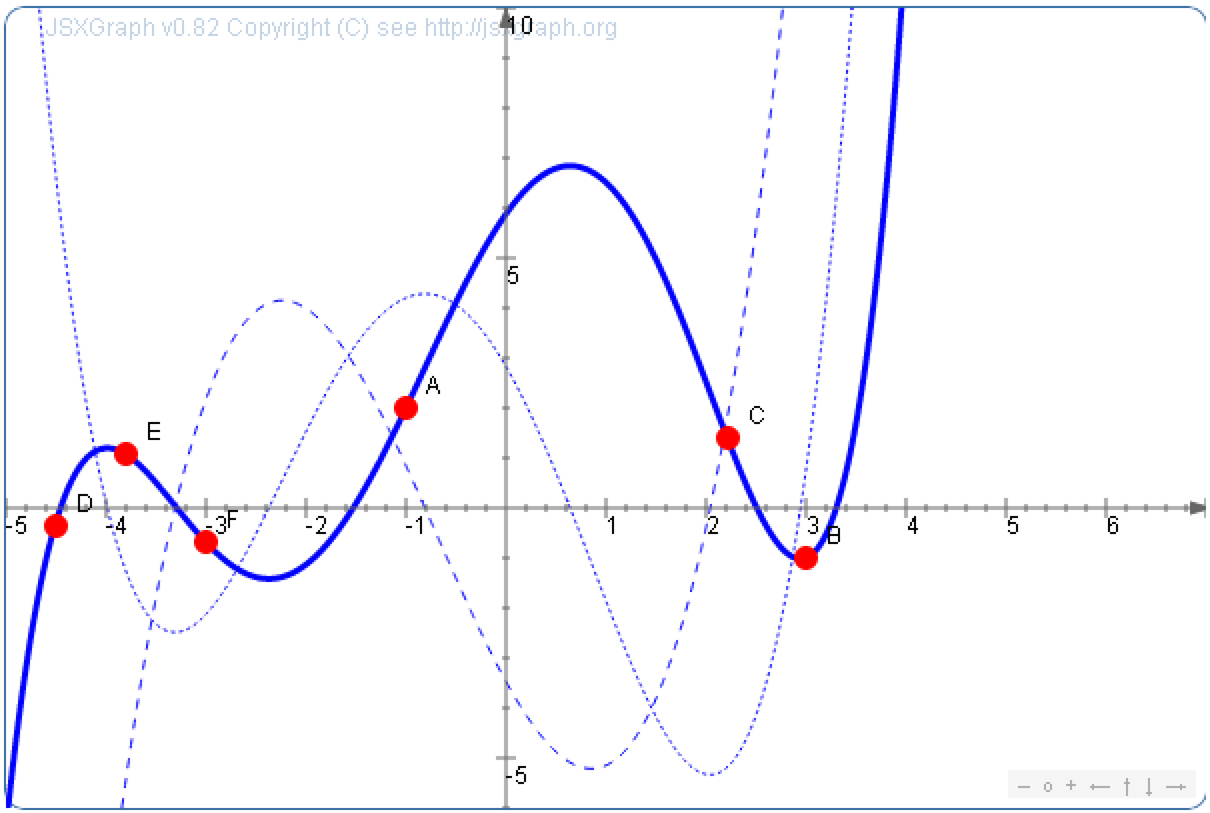
Secret recovery

- in order to recover the secret s, we will need a minimum of n points of the polynomial
  - the order doesn't matter
- with that n parts, we do Lagrange Interpolation/Polynomial Interpolation

Polynomial Interpolation / Lagrange Interpolation

- for a group of points, we can find the smallest degree polynomial that goees through all that points
  - this polynomial is unique for each group of points





JSXGraph v0.82 Copyright (C) see http://jsxgraph.org
10
5
-5
-5
-4
-3
-2
-1
1
2
3
4
5
6
A
B
C
D
E
F

$$L(x) = \sum_{j=0}^{n} y_j l_j(x)$$

$$\ell_j(x) := \prod_{\substack{0 \le m \le k \\ m \ne j}} \frac{x - x_m}{x_j - x_m} = \frac{(x - x_0)}{(x_j - x_0)} \cdots \frac{(x - x_{j-1})}{(x_j - x_{j-1})} \frac{(x - x_{j+1})}{(x_j - x_{j+1})} \cdots \frac{(x - x_k)}{(x_j - x_k)},$$

Wikipedia example

*example over real numbers, in the practical world, we use the algorithm in the Finite Field over p

(more details: https://en.wikipedia.org/wiki/Shamir's_Secret_Sharing#Problem)

- $s = 1234$
- $m = 6$
- $n = 3$
- $f(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2$
  - $\alpha_0 = s = 1234$
  - $\alpha_1 = 166$ *(random)*
  - $\alpha_2 = 94$ *(random)*
- $f(x) = 1234 + 166x + 94x^2$

- $f(x) = 1234 + 166x + 94x^2$
- we calculate the points $P = (x, f(x))$
  - where x is each one of the values between 1 and m
    - $P_1 = (1, f(1)) = (1, 1494)$
    - $P_2 = (2, f(2)) = (2, 1942)$
    - $P_3 = (3, f(3)) = (3, 2578)$
    - $P_4 = (4, f(4)) = (4, 3402)$
    - $P_5 = (5, f(5)) = (5, 4414)$
    - $P_6 = (6, f(6)) = (6, 5614)$

- to recover the secret, let's imagine that we take the packets 2, 4, 5
  - $(x_0, y_0) = (2, 1942)$
  - $(x_0, y_0) = (4, 3402)$
  - $(x_0, y_0) = (5, 4414)$

- let's calculate the Lagrange Interpolation
  - $\ell_0 = \frac{x - x_1}{x_0 - x_1} \cdot \frac{x - x_2}{x_0 - x_2} = \frac{x-4}{2-4} \cdot \frac{x-5}{2-5} = \frac{1}{6}x^2 - \frac{3}{2}x + \frac{10}{3}$
  - $\ell_1 = \frac{x - x_0}{x_1 - x_0} \cdot \frac{x - x_2}{x_1 - x_2} = \frac{x-2}{4-2} \cdot \frac{x-5}{4-5} = -\frac{1}{2}x^2 + \frac{7}{2}x - 5$
  - $\ell_2 = \frac{x - x_0}{x_2 - x_0} \cdot \frac{x - x_1}{x_2 - x_1} = \frac{x-2}{5-2} \cdot \frac{x-4}{5-4} = \frac{1}{3}x^2 - 2x + \frac{8}{3}$
  - $$\begin{aligned} f(x) &= \sum_{j=0}^{2} y_j \cdot \ell_j(x) \\ &= y_0\ell_0 + y_1\ell_1 + y_2\ell_2 \\ &= 1942\left(\frac{1}{6}x^2 - \frac{3}{2}x + \frac{10}{3}\right) + 3402\left(-\frac{1}{2}x^2 + \frac{7}{2}x - 5\right) + 4414\left(\frac{1}{3}x^2 - 2x + \frac{8}{3}\right) \\ &= 1234 + 166x + 94x^2 \end{aligned}$$
- obtaining $f(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2$, where α_0 is the secret s recovered
  - where we eavluate the polynomial at $f(0)$, obtaining $\alpha_0 = s$
- *we are not going into details now, but if you want in the practical workshop we can analyze the 'mathematical' part of all of this

And now... practical implementation

- full night long
- big ints are your friends
- $L(x) = \sum_{j=0}^{n} y_j l_j(x)$

$$\ell_j(x) := \prod_{\substack{0\le m\le k \\ m\ne j}} \frac{x-x_m}{x_j-x_m} = \frac{(x-x_0)}{(x_j-x_0)} \cdots \frac{(x-x_{j-1})}{(x_j-x_{j-1})} \frac{(x-x_{j+1})}{(x_j-x_{j+1})} \cdots \frac{(x-x_k)}{(x_j-x_k)},$$

About

- https://arnaucube.com
- https://github.com/arnaucube
- https://twitter.com/arnaucube

2019-07-05