CSP Project

Abishek Agarwal and Laldinpuia Hmar

May 2024

Toy Example

We first initialzie an elliptic curve defined over a prime p

Let

$$y^2 = x^3 + ax + bmod3$$

where a = 1, b = 1 and p =3

Then, lets say our population size = 2 ,

When we initialize 2 points on the curve called Chromosomes

We randomly choose x1(from 0 to 2) = 0,

$$y^2 = 0 + 0 + 1mod3 = 1$$

Checking if y^2 is quadratic residue

$$1^{2/2}mod3 = 1$$

Then,

$$y = 1^{4/4}mod3 = 1$$

Population = [Chromosome1 = (0,1), Chromosome2 = (0,2)] where chrosmose2 = (0,p-1)

Again randomly x2 = 0

Then, similarly as before we will get y = 1

So,

Population = [Chromosome1 = (0,1), Chromosome2 = (0,2), Chromosome3 = (0,1), Chromosome4 = (0,2)]

Now, let assume a target point (1,0) which is on the curve, lets calculate fitness using distance formula

For (0,1), Distance = $\sqrt{((1-0)^2 + (0-1)^2)} = \sqrt{2}$

Fitness = 1/ 1 + 1.414 = 1 / 2.414 = 0.414

For (0,2), Distance = $\sqrt{(0-1)^2 + (2-0)^2} = \sqrt{5}$

Fitness = 1/ 1 + $\sqrt{5}$ = 1/3.2360 = 0.309

Now, we take tournament size as 2

Then we randomly select Chromosome1 and Chromosome2,

Then best fitness = Chromosome 1

Again, we select chromosome 3 and chromosome 4,

Then best fitness = Chromosome 3

So, Selected Parents = [Chromosome1 = (0,1), Chromosome3 = (0,1)]

Now, we perform crossover among the selected parents to get offspring,

Here we are taking a 50% probability of choosing from either both parents x and y. For example, if $x_1 = (2,4)$ and $x_2 = (3,5)$, their offspring can be $x = (2,5)$ and $y = (3,4)$. However, in our case, as both Chromosome1 and Chromosome2 are similar, their offspring will also be similar.

Now, Offspring1 = (0,1) and offspring2 = (0,1)

Nextly, we perform mutation and we randomly select mutation rate = 0.1

And for each x and y, we select a random value and if that random value is less than the mutation rate we choose new value for that x or y

So, mutation of offspring1

for x = random 0.2

As 0.1 0.2, so mutatedX = 0

For y = random 0.05

As 0.05 < 0.1, then

we check whether that is on the curve

$$y^2 = (0^3 + 0 + 1) mod 3 = 1$$

So, then, mutatedY = 1mod 3 = 1

So, mutatedOffspring1 = (0,1)

Again, for offspring 2

For x = random 0.8

As 0.8 > 0.1

mutatedX = 0

for y = random 0.3

As 0.1 ¡ 0.3, then mutatedY = 1

So, mutatedY = (0,1)

Now, population = [(0,1),(0,1)]
As we have assume 2 number of generations , the above will be used for parents again for next generation

After Crossover: (0,1) and (0,1)

and after mutation, we again have

Mutation: offspring: [(0,1), (0,1)]

So, Population: [(0,1), (0,1), (0,1), (0,1)]

Now we calculate fitness with the target function again.

For (0,1) : Fitness = 0.414

We will choose (0,1) as the best key pair and also (0,1) is in the curve's parameter.

```
----- KEYGEN -----
Best key pair found: Chromosome(x=0, y=1)
Keygen Duration: 0:00:00
```

However, choosing the key pair greatly depends on choosng crossover, mutation rate and the random number choosen during mutation and can vary

So, by combining the strengths of ECC's security properties and GA's optimization capabilities, it offers advantages with randomness of the generated key pairs.

Now, using similar parameters with normal ECC, lets compare we have

if we select k = 1

Then k(0,1) = 1(0,1) = (0,1)

Normal ECC security relies deeply on the DLP hardness. While our proposed algorihtm also uses ECC for initiazation and for the points to be on the curve, the genetic algorthim continously introduces lots of randomnesss inside the curve parameter which makes it hard. So, the key search space becomes larger and attack time becomes larger.

Now, we use this for encrpytion

lets say we have plaintext = 'dp'

If we convert to binary = 01100100 01110000

and padded binary text = 011001000111000000000000

Also, key = 0001

And padded key = 000000000000000000000001

Using crossover(padded binary) at the second point(We take first 2 elements from one binary and remaining from another, if it is odd, the the last one remains same)

crossover = ['01110000', '01100100', '00000000']

Mutation by flipping the first and last bits

Mutated data = ['11110001', '11100101', '10000001']

If we try to convert this binary to plaintext also, its already unreadable, so to finally encrpyt it, we xor it with the key

111100011110010110000001 = ñå?
So,

Encrpyted data =

$$\begin{array}{ccc} 11110001 & 11100101 & 10000001 \\ \oplus & & \\ 00000000 & 00000000 & 00000001 \\ \hline 11110001 & 11100101 & 10000000 \end{array}$$

```
----- USAGE ----
Please enter a string: dp
Enter crossover points (comma-separated): 2
Crossover Points:  [2]
----- ENCRYPTION -----
Plaintext in binary:  0110010001110000
Padded length 011001000111000000000000
key 0001
padkey 000000000000000000000001
Segmented Data:  ['01100100', '01110000', '00000000']
Crossover Data:  ['01110000', '01100100', '00000000']
Mutated Data:  ['11110001', '11100101', '10000001']
Encrypted Data:  111100011110010110000000
```

Decrpyted data =

$$\begin{array}{ccc} 11110001 & 11100101 & 10000000 \\ \oplus & & \\ 00000000 & 00000000 & 00000001 \\ \hline 11110001 & 11100101 & 10000001 \end{array}$$

Reversemutation = 01110000 01100100 00000000

Reversecrossover = 01100100 01110000 00000000

Plaintext = dp

```
----- DECRYPTION -----
Decrypted Data:  111100011110010110000001
Mutated Decrypted Data:  ['01110000', '01100100', '00000000']
Crossover Decrypted Data:  ['01100100', '01110000', '00000000']
Binary Decrypted Data:  0110010001110000
Decrypted Text:  dp
```

This process of encryption involves lot of randomness in the process and is hard to attack, and therefore its efficiency is better than aes or des and attack time is much higher. (https://thesai.org/Downloads/Volume9No6/Paper_51-Implication_of_Genetic_Algorithm.pdf)