CHALLENGE 2: ARITHMETIC OF ELLIPTIC CURVE CRYPTOGRAPHY

30/08/2019
European Cyber Security Challenge 2019
Bucharest, Romania

1. Initial Write-Up

Description:

Elliptic Curve Cryptography (ECC) is broadly used today in various applications, from small, constrained devices to large cloud servers. In the following tasks, you can use the open source software SAGE [1] (either at the server or download it in advance) and you need to calculate several values and properties of the curves. The goal is to understand ECC principles with a practical arithmetic example by calculate doubling/addition on a curve, finding tangent line, performing finite field arithmetic with the use of SAGE and with some tricks, that do not require SAGE, but some thinking and understanding of finite field arithmetic. Then we can see how an encryption protocol works on elliptic curves.

2. Challenge specifications

- Category: Crypto
- Difficulty : Medium
- Expected time to solve: 1h

3. Technical specifications

Recommended tool: Sage software

4. Questions and answers

<u>Question 1:</u>

Given the curve E:y^2 = x^3 + 2x + 3 on the finite field $F_{6277101735386680763835789423251}$ verify that P=(6101065374528763628460436663049 : 2223659447177254506928398297998 : 1) and Q=(3812161343866210535516881234 4 : 3123847732620019127314512268740 : 1) belong on the curve. What is the order of point P? Compute the point R=P+Q and the tangent line passing from point P.

Solution:

n = P.order() = 1569275433846670338504130648889

Sage code for verification:

```
>> Pr=Primes();

>> q= Pr.next(6277101735386680763835789423207); q

>> E=EllipticCurve(GF(q),[2,3]); E;

>> is_prime(q);

>> P=E(6101065374528763628460436663049,2223659447177254506928398297998);

>> P in E = True

>> n=P.order(); n

>> k=15;

>> Q=k*P; Q; Q in E = True

>> R=P+Q$;

>> Sum: R = (4656368481481548480065848122955 : 1037624607674094246105294702964 : 1)
```

For the tangent line, we need to calculate the derivative: 2yy' = 3x^2+2 => y' = \dfrac{3x^2+2}{2y} mod(q)

From Sage we calculate:

```
>> a =mod(3*(6101065374528763628460436663049)^2+2,q);

>>  a=2336871647132103542957107140397

>> y2=2*2223659447177254506928398297998;

>> y2=4447318894354509013856796595996

>> b=inverse_mod(y2,q);

>> b = 2091618248115153689711465597859

>> tangent=mod(a*b,q);

>> tangent =4525832513492997654950403808144
```

So the tangent line is: y= 4525832513492997654950403808144 * x

Question 2:

European Cyber Security Challenge 2019 challenge 2: Arithmetic of elliptic curve cryptography
Bucharest, Romania **30/08/2019**

ElGamal cryptosystem over elliptic curves of finite fields is a natural extension of the classical ElGamal over finite fields, based on the difficulty of solving the discrete logarithm problem over an elliptic curve (ECDLP). Such cryptosystems can be defined for encryption and signature schemes. One important point is how to define an optimal function F that maps messages to points in the field, is invertible and not hard to compute. Let p be an odd prime and F_q a finite field over which the elliptic curve E(F_q) is defined. The bijection [1,p-1] -> E(F_q*) is an optimal injecting encoding, as described in [2].

Let us assume that Bob and Alice have agreed on the following parameters: elliptic curve $E: y^2=x^3+2x+3$ over F_q, where q=673, a generator point P=(667:197:1) on the curve of order n=654.

- Suppose Bob has chosen a secret key x in [1,q-1] randomly and this is x=506. What is his public key?

Solution: Y = xP = (600:335:1)

- Alice used Bob's public key and encrypts for him the message m. They use the ElGamal scheme described by Koblitz in [3], with injecting encoding of messages m_2 -> m P = P_m (so the message m in binary format is transformed to integer and multiplied by P, to give the point on the curve P_m). On reception of ciphertext (C,D) = ((662:116:1), (70:68:1)) how can Bob decrypt the message? Use Sage or any other tool you need and show all the steps to decrypt the ciphertext.

  Solution: Bob can calculate C' by C' = xC = 506*(662:116:1) = (457:604:1), and retrieve the point P_m with P_m=D - C' = (k(xP)+P_m) - (x(kP)) = (70:68:1) - (457:604:1) = (400 : 192 : 1), which is equal to P_m. Then calculate the message m by brute-forcing for all the possible values i in the prime field, such that i*P = P_m => i = m = 27.

  Sage code:

```
>> is_prime(673);

>> q=673;

>> E=EllipticCurve(GF(q),[2,3]); E

>> P=E.gens();

>> P=E(667,197)

 >> n=P.order(); n=654

 >> car=E.cardinality();car

 car=n which shows that P is generator indeed

>> x=506;

Bob sends to Alice public key Y

>> Y=x*P; Y
>> k=37;

>> C=k*P;C; Cipher1 = C

>> Cc=k*Y; Cc;

>> m=11011; mm=Integer(str(m), base=2);mm

>> Pm=mm*P;Pm;

>> Cipher2=Cc+Pm;Cipher2 Cipher2=Cc + Pm
```

Alice sends to Bob (Cipher1,Cipher2)=(C,Cc+Pm) - Decryption by Bob:

```
>> b1 = x*C; b1; b1=(457:604:1)

>> b1==Cc

>> true

>> bPm=Cipher2-b1; bPm;

>> bPm ==Pm

>> true

>> for i in range(q-1):

>> if  i*P==Pm:

print i
```

- Describe a scenario where Bob cannot decrypt the message in polynomial time.
  Solution:
  When the finite field is of high prime order, where the prime q is hundreds of digits long, then decrypting the ciphertext is equivalent to solving the ECDLP, which is shown to be hard in polynomial time.

5. Attack Scenario

N/A

6. Installation instructions

No special installation instruction needed

7. Tools needed

Sage software

8. Artefacts Provided

N/A

9. Walkthrough (writeup)

N/A

10. References

[1] Injective Encodings to Elliptic Curves, Pierre-Alain Fouque, Antoine Joux and Mehdi Tibouchi, 2013, https://eprint.iacr.org/2013/373.pdf.

[2] Neal Koblitz, Elliptic Curve Cryptosystems, 1987, https://www.ams.org/journals/mcom/1987-48-177/S0025-5718-1987-0866109-5/S0025-5718-1987-0866109-5.pdf