**Hack the Box**

LostKey (Cryptography)

Team Members

1. Ganesh Chowdavaram
2. Sinduja Bollikonda
3. Krishi Jyothirmai vadaparthi
4. Chetana Ramya Malampati

**Challenge Description**: Mustard Brightpants is an archaeologist who has been excavating ruins in Egypt for the past 25 years. In one of his discoveries, he found a sphere-shaped trinket that has a strange combination of letters and numbers printed around it. Alongside it was a scroll containing a riddle in a strange language, and a keypad. The sphere's contents might finally solve the mystery behind the downfall of the mythical city of Outlandis. Could you help Mr. Brightpants solve the riddle and find the correct key to unlock the sphere?

This Challenge has two files: One **python file encrypt.py** and another **text file output**.

After analyzing the **encrypt.py file** it’s clear that it used the elliptic curve cryptography and the encryption mechanism.

In the Output file, there are G(Base point), Gn(Encrypted Point), Cipher Text and IV(initialization vector). The output file is the one obtained after executing the encrypt.py file.

So, to get the flag, we need to reverse this and decrypt the Cipher text in the output file.

But we need the unknown value n which can be calculated using the coefficients of the elliptic curve. But to solve this we need a lot of computational power which normal python environment. So, used Sagemath which is designed to solve these types of computation problems.

General Form of Elliptic Curve in Weirstrass Form A black and white math equation

Description automatically generated

After comparing the values we get the values of a2, a3, a4, a6 which are the coefficients of the elliptic curve

a2 = 208913474430283759938044884583915265967

a3 = 3045783791

a4 = 177776968102066079765540960971192211603

a6 = 308081941914167831441899320643373035841

and the value of P which is already in the encrypt file.

**Code to get n value in Sagemath** A screenshot of a computer

Description automatically generated

**Python Code to Decrypt the Ciphertext**

A close-up of a computer screen

Description automatically generated

**Flag**

HTB{uns4f3\_3ll1pt1c\_curv3s\_l3d\_t0\_th3\_c0ll4ps3\_0f\_0u7l4nd1s}

**Code in Sagemath:**

**Open Sagemath** [**https://sagecell.sagemath.org/**](https://sagecell.sagemath.org/)

**Run the below code you will get the en value and use that value to decrypt the cipher text.**

**# Define the prime modulus of the field**

p = 101177610013690114367644862496650410682060315507552683976670417670408764432851

**# Create the finite field Z/pZ**

Zp = Zmod(p)

# Define the coefficients of the elliptic curve

a2 = 208913474430283759938044884583915265967

a3 = 3045783791

a4 = 177776968102066079765540960971192211603

a6 = 308081941914167831441899320643373035841

**# Calculated from the equation in the commented-out code**

**# Define the elliptic curve over Zp**

EC = EllipticCurve(Zp, [0, a2, a3, a4, a6])

**# Define the base point G on the elliptic curve**

G = EC(14374457579818477622328740718059855487576640954098578940171165283141210916477,

97329024367170116249091206808639646539802948165666798870051500045258465236698)

**# Define the point P (which is nG for some n)**

P = EC(32293793010624418281951109498609822259728115103695057808533313831446479788050,

12261320786387104409204182869939851233823411723368873908519712833126350182940)

# Factor the order of the elliptic curve to find prime factors

# (This is commented out but would be used to find factors for pe\_list)

# print(factor(EC.order()))

**# List of prime factors of the curve's order**

pe\_list = [9, 59, 14771, 27733, 620059697, 2915987653003935133321, 257255080924232005234239344602998871]

**# List of remainders when n is divided by elements in pe\_list**

emod\_list = [4, 27, 12977, 2568, 261975359]

**# Use the Chinese Remainder Theorem to combine the emod\_list into an approximation of n**

n\_approx = CRT(emod\_list, pe\_list[:5])

# Output the approximated value of n

print(n\_approx)

**Decrpt Cipher Text Code:**

**Run the below code in python and you will get the flag.**

from Crypto.Cipher import AES

from hashlib import sha1

def decrypt(key):

iv = 'baf9137b5bb8fa896ca84ce1a98b34e5'

c = 'df572f57ac514eeee9075bc0ff4d946a80cb16a6e8cd3e1bb686fabe543698dd8f62184060aecff758b29d92ed0e5a315579b47f6963260d5d52b7ba00ac47fd'

iv = bytes.fromhex(iv)

c = bytes.fromhex(c)

key = sha1(str(key).encode('ascii')).digest()[0:16]

cipher = AES.new(key, AES.MODE\_CBC, iv)

try:

m = cipher.decrypt(c)

if (b'HTB' in m):

print(m)

except:

pass

n = 134876030111980880301

en = 82438979720724695506

while (True):

decrypt(en)

en += n

**Result**:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated