1. בחר קובץ כלשהו מהמחשב- חשב עבורו את SHA1, SHA256, SHA512 ואת MD5 (נקליד את החישוב הרצוי בterminal בתוספת המילה "sum" ואת שם הקובץ הרצוי. לדוגמא "md5sum example.docx"
(או ב https://md5calc.com/hash/md5)

א. מדוע המפתח הפרטי והציבורי מוצפנים דווקא באמצעות SHA256? מה ההבדל בין כל סוגי הפונקציות? מדוע משתמשים בripemd160 אח"כ?
גודל הפלט. התאמה של המפתח לכתובת ביטקוין.

ב. את המפתח הציבורי (12,16) – הפוך לכתובת ביטקוין חוקית עפ"י הפרוטוקול הנח שהגרסה היא 1.
0212
SHA256 →
b7ca93a0ea3d4b2d2330b5363aa251bf2c8bd184f3b4d8032b645064772ad214
RIPEMD160→
ca2a1fafa70958268232ad5e42093be4abb619c8
version+payload+(double SHA256 last 4 digits)→
1ca2a1fafa70958268232ad5e42093be4abb619c825f4
Base58Check Encode→
39MMS4Up9fAGpFj8p1W1vjMpFLeNtFd

2. באפליקציות ארנק ביטקוין (ובפרט בbitcoin core) מאוחסנים זוגות של מפתחות – ציבורי ופרטי. אמרנו שאפשר לגלות את המפתח הציבורי מהפרטי. אז מדוע שלא נאכסן **רק** את המפתחות הפרטיים בארנק?
על מנת לגשת למפתח פרטי כלשהו, נצטרך את המפתח הציבורי שלנו (הוא ידוע). כיוון שהדרך ההפוכה היא אינה אפשרית בזמן סביר, נצטרך לאחסן דווקא זוגות.

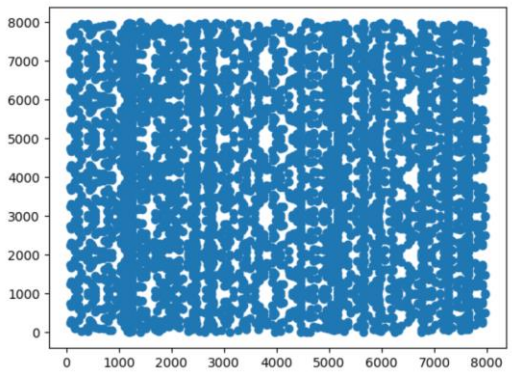
3. א. כתוב תוכנה שמציירת את העקומה האליפטית secp256k1 מודולו מספר קטן (p=17 למשל).

ב. G=(0x65d5b8bf9ab1801c9f168d4815994ad35f1dcb6ae6c7a1a303966b677b813b00,0xe6b865e529b8ecbf71cf966e900477d49ced5846d7662dd2dd11ccd55c0aff7f)
עבור p=17 ונקודה G על העקומה - מצא את המפתח הפרטי עבור מפתח ציבורי (12,16) – 475

ב. מאיזה מספר מודולו הפעולה נעשית מורכבת מדי?

```python
import matplotlib.pyplot as plt

Fp = 17


x = []
y = []

for i_x in range(Fp):
    for i_y in range(Fp):
        if ((i_x**3 + 7)-(i_y**2)) % Fp == 0:
            x.append(i_x)
            y.append(i_y)

plt.scatter(x, y)
plt.show()
```

עבור Fp=8000:




```python
from ecpy.curves import Curve, Point

SEARCH_LIMIT = 10000000
P = 1000  # modulo 17
publicKey = (12, 16)    # example for public key modulo 17


c = Curve.get_curve('secp256k1')    # using the "y^3 = x^2 +7" curve for generate key
p = Point(0x65d5b8bf9ab1801c9f168d4815994ad35f1dcb6ae6c7a1a303966b677b813b00,
          0xe6b865e529b8ecbf71cf966e900477d49ced5846d7662dd2dd11ccd55c0aff7f, c)    # point p is on the curve

privateKey = 0  # initial as null
for i in range(SEARCH_LIMIT):     # limit the search. means give up.
    currentPoint = i*p      # generate some public key - check if this is the our public key
    if (currentPoint.x % P) == publicKey[0] and (currentPoint.y % P) == publicKey[1]:
        privateKey = i      # private key found!
        break


if(privateKey):
    print("Private Key is: ", privateKey)
else:       # no private key found
    print("Private Key not found!!")
```

עבור p=100 המפתח הפרטי לנקודה (12, 16) הוא: 12647 (וזה לוקח בערך 10 שניות)
עבור p=1000 המפתח הפרטי לנקודה (12, 16) הוא: 612385 (וזה לקח 12 וחצי דקות!)