The Python scripts utilized to accomplish the tasks are shown at the end of this document.

You intercept the following ciphertext generated using the following RSA public-key: pk={e,n}={23,20413}

Determine the prime numbers p and q:

I determined the following values for p and q, using the find_factors function in my rsa_encode.py script: $p = 137$, and $q = 149$.

Determine Euler's totient function $\phi(n)$:

I calculated $\phi(n)$ as follows:

$$\phi(n) = (p-1) \cdot (q-1)$$
$$\phi(n) = (137-1) \cdot (149-1)$$
$$\phi(n) = 20,128$$

Determine the private-key={d,n}:

To compute d, the following relationship must hold:

$$d \cdot e \equiv 1 \mod \phi(n) \tag{1}$$

I used the eea.py script with $e = 23$, and $\phi(n) = 20,128$. This produced $d = 13127$.

Compute the plaintext for EACH of the following ciphertext :

{236, 2743, 7983, 5919, 20213, 5520, 19563, 17083, 17083, 19326, 5919, 17258, 5919, 17215, 19563, 20213, 4940, 496}

The plaintext is shown below:

```
Plaintext: [65, 110, 100, 32, 115, 116, 105, 108, 108, 44, 32, 73, 32, 114, 105, 115, 101, 46]
```

Determine the ENGLISH plaintext:

The text output is shown below:

```
English plaintext:  And still, I rise.
```

Python Scripts:

```python
def xgcd(a, b):
    """return (g, x, y) such that a*x + b*y = g = gcd(a, b)"""
    x0, x1, y0, y1 = 0, 1, 1, 0
    while a != 0:
        q, b, a = b // a, a, b % a
        y0, y1 = y1, y0 - q * y1
        x0, x1 = x1, x0 - q * x1
    return b, x0, y0

def mulinv(a, b):
    """return x such that (x * a) % b == 1"""
    g, x, _ = xgcd(a, b)
    if g == 1:
        return x % b

def main():
  e, n, p, q = 23, 20413, 137, 149
  phi = (p-1)*(q-1)
  print("d = {}".format( mulinv(e, phi) ) )

if __name__ == "__main__":
  main()
```

eea.py

```
import math

def find_factors(a):
  factors = []
  for p in range(2,a-1):
    if (a % p) == 0:
      factors.append(p)
      factors.append( int( a/p ) ) # Append q
      break

  if len(factors) == 0:
    print("{} is prime".format(a))
  else:
    print("p = {0}, q = {1}".format(factors[0], factors[1]))
  return factors


def rsa_decode(a, p, q):
  n = p * q
  d = 13127
  # plaintext = ciphertext ^ d mod n

  output = (a ** d) % n
  return output


def mulinv(a, b):
    """return x such that (x * a) % b == 1"""
    g, x, _ = xgcd(a, b)
    if g == 1:
        return x % b

def main():
  inputs = [236, 2743, 7983, 5919, 20213, 5520, 19563, 17083, 17083, 19326, 5919,
   17258, 5919, 17215, 19563, 20213, 4940, 496]
  n = 20413
  e = 23
  p,q = find_factors(20413)
  print( "d = {}".format( mulinv(e, (p-1)*(q-1) ) ) )
  outputs = ""
  plaintext = []
  for input in inputs:
    pt = rsa_decode(input, p, q)
    plaintext.append( pt )
    outputs += str( chr( pt ) )

  print("Plaintext: " + str(plaintext) )
  print("Output: " + outputs)


if __name__ == "__main__":
  main()
```

rsa_decode.py