M328K Homework 11

Joshua Dong

April 16, 2014

0.1 8.1.8

We count the word frequencies using Python. 'V' is the most common character. Listing all the rotations of the text, we also see that the rotation of +9 yeilds "THEVA LUEOF THEKE YISSE VENTE EN".

```
cipher = "KYVMR", "CLVFW", "KYVBV", "PZJJV", "MVEKV", "VE"
freq = {}
for c in ''.join(cipher):
    if c in freq.keys():
        freq[c] += 1
    else:
        freq[c] = 1
print sorted(freq.items(), key=lambda x: x[1])
def rot(w, i):
    return ''.join(chr(((ord(c)-ord('A')+i)%26) + ord('A')) for c
        in w)
for i in xrange(26):
    print ' '.join(rot(w, i) for w in cipher)
```

0.2 8.1.12

Using the same program as before, we produce this output:
('A', 1), ('E', 1), ('K', 1), ('J', 1), ('O', 1), ('V', 1), ('D', 2),
('G', 2), ('I', 2), ('N', 2), ('S', 2), ('U', 2), ('Z', 2), ('C', 3),
('X', 3), ('W', 4), ('P', 5), ('Y', 5), ('R', 6), ('M', 7)
We can now guess that 'M' corresponds to 'E' and 'R' corresponds to 'T'.
$c(ord(M) - d) \equiv ord(E) \pmod{26}$
$c(12 - d) \equiv 4 \pmod{26}$
$12c - cd \equiv 4 \pmod{26}$

$c(ord(R) - d) \equiv ord(T) \pmod{26}$
$c(17 - d) \equiv 19 \pmod{26}$
$17c - cd \equiv 19 \pmod{26}$

$5c \equiv 15 \pmod{26}$
$(-5)5c \equiv (-5)15 \pmod{26}$

$-25c \equiv -75 \pmod{26}$
$c \equiv 3 \pmod{26}$
$3(12 - d) \equiv 4 \pmod{26}$
$36 - 3d \equiv 4 \pmod{26}$
$-3d \equiv -6 \pmod{26}$
$d \equiv 2 \pmod{26}$

Now we use a similar program to decode the text:

```
def decode(C, c, d):
    return ''.join(chr( ( c*((ord(i)-ord('A')-d) )%26) + ord('A'))
        for i in C)
print decode(cipher, 3, 2)
```

This produces:
EVERYALCHEMISTOFANCIENTTIMESKNEWHOWTOTURNLEADINTOGOLD

0.3 8.1.8

We keep guessing the factorization for 2881 until we find that $2881 = (43)(67)$.

We can find $\varphi(2881) = \varphi(43)\varphi(67) = 42 * 66 = 2772$
Now we find the inverse of 5 modulo 2772.

```
for i in xrange(2772):
    if (5*i)%2772 == 1:
        print i
```

This returns 1109, the value of d.
It is now trivial to decode the cipher:

```
import re
ciphertext = '05041874034705152088235607360468'
def decode(C, d, n):
    output = ""
    for c_i in re.findall('..?.?.?', C):
        c_i = str((int(c_i)**d)%n).zfill(4)
        output += chr(int(c_i[:2])+ord('a'))
        output += chr(int(c_i[2:])+ord('a'))
    return output
print decode(ciphertext, 1109, 2881)
```

This produces the string 'eatchocolatecake'.

0.4 8.1.14

If n_1, n_2, n_3 are not coprime, then factoring them is trivial and thus the key is easily produced.
If n_1, n_2, n_3 are coprime, then we have a system of congruences:
$x_1 \equiv P^3 \pmod{n_1}$
$x_2 \equiv P^3 \pmod{n_2}$
$x_3 \equiv P^3 \pmod{n_3}$
By the Chinese Remainder Theorem, there must exist some $y \in \mathbb{Z}$ where $y \equiv P^3 \pmod{n_1 n_2 n_3}$
Since RSA requires that $P < n_i$, we know that $P^3 < n_1 n_2 n_3$.
$\therefore y = P^3$.
Thus deciphering is reduced to solving the set of linear congruences and then finding the cube root of the solution. This is significantly easier than factoring large numbers.
(Note: we can make the same argument for any e, meaning that having e ciphertexts with their corresponding n_i will lead to an easier computation to determine the plaintext)