The Mathematics of Cryptography

Joseph Allen

11/12/2014

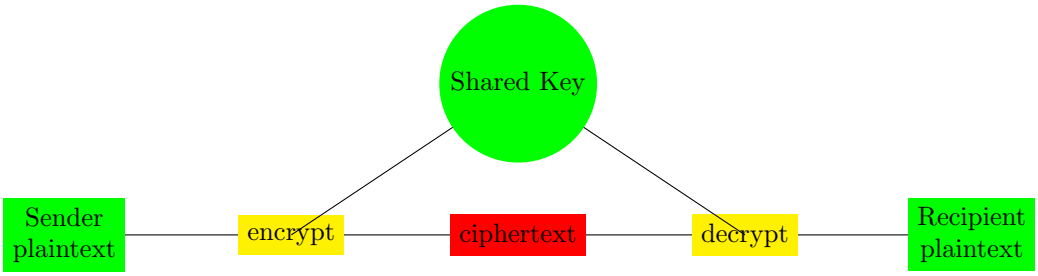
1 Introduction

Cryptography is the study of methods of secure communication between a given number of parties.

Mathematical theory and computer science are heavily involved in modern cryptography. In World War 2, the German enigma machine was used to encrypt messages, and in turn, efforts by British cryptanalysts eventually resulted in their decryption.

Cryptography relies on keys; a key is a piece of information which determines the result of a cryptographic algorithm. In encryption it allows the conversion of plaintext(the information being transmitted) into ciphertext(the encoded information), or vice versa for decryption.

2 Symmetric-Key Cryptography



Symmetric key cryptography uses the same algorithm for encryption and decryption.

Caesar's Cipher is one of the most simple, well known methods of encryption. It is a substitution cipher;units of plaintext are replaced with those of ciphertext to encrypt, and vice versa for encryption. Caesar's Cipher originally used a shift of three to the right on the alphabet, giving it a key of 3.

That is to say:

ABCDEFGHIJKLMNOPQRSTUVWXYZ $\Rightarrow$ DEFGHIJKLMNOPQRSTUVWXYZABC

This can also be represented with modular arithmetic as follows:

$$E_n(x) = (x + n) \bmod 26 \text{ for encryption}$$
$$D_n(x) = (x - n) \bmod 26 \text{ for decryption}$$

This can be coded in python:

```
key = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',
'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
```

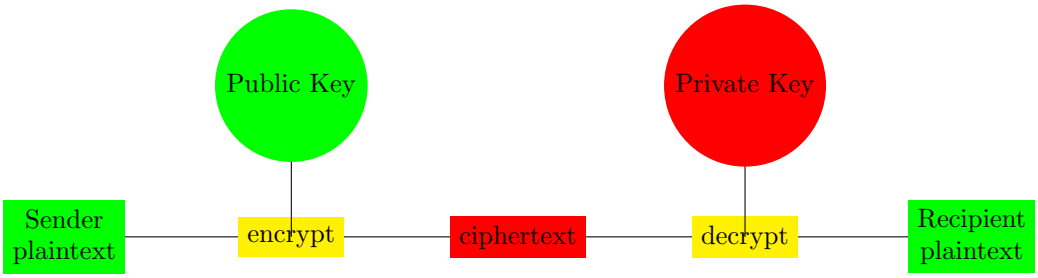
And the code for encryption:

```
def basicencrypt(s):
    """
    This function takes a string of plaintext and produces a string of ciphertext
    Input: String of plaintext
    Output: Encrypted string of ciphertext
    """
    c = "" #string of ciphercode
    f = 0
    for n in s:
        for k in range(len(key)):
            if n == key[k]:
                if k <= 22:
                    f = k + 3
                    c += key[f]
                elif k == 23:
                    c += key[0]
                elif k == 24:
                    c += key[1]
                elif k == 23:
                    c += key[2]
    return c
```

And the code for decryption:

```
def basicdecrypt(s):
    """
    This function takes a string of ciphertext and produces a string of plaintext
    Input: String of ciphertext
    Output: Encrypted string of plaintext
    """
    p = "" #string of plaintext
    f = 0
    for n in s:
        for k in range(len(key)):
            if n == key[k]:
                if k >= 3:
                    p += key[k-3]
                elif k == 0:
                    p += key[-3]
                elif k == 1:
                    p += key[-2]
                elif k == 2:
                    p += key[-1]
    return p
```

3 Asymmetric-Key Cryptography



In asymmetric-key cryptography, also known as public key cryptography, the usage of different keys has numerous benefits.Despite being mathematically related, it is incredibly difficult to compute the private key from the public one. This prevents the lapse in security should a key become compromised, and allows fewer keys between a greater number of people. Prime numbers are used as keys, as factoring large primes is a very time consuming task and therefore very difficult with higher bit encryptions. For example:

Two distinct prime numbers, x and y, form private keys.

Their product, xy, is a public key.

If $x = 101, y = 139$ then the product xy is 14039

The prime numbers used are smaller than usual, but the product is still time-consuming to factorise.

4 Cryptanalysis

Cryptanalysis is defined as finding and exploiting weaknesses within a cryptographic scheme, in order to gain access to its contents without requiring a key. The method used largely revovles around the encryption system. Asymmetric keys, due to their reliance on more complex mathematical problems, are more widely linked to greater mathematical research. Brute force attacks consist of systemetically iterating over every possible passcodes/keys until the correct one is found. The following is a piece of brute force code written to find a combination of lower case letters that make up a simple password.

```
def brute(c):
    """
    This function compiles a list of possible combinations of the number of letters
    """
    count = 0
    possibilities = []
    alphabet = 'abcdefghijklmnopqrstuvwxyz'
    for n in range(len(c)):
        a = [i for i in alphabet]
        for y in range(n):
            a = [x+i for i in alphabet for x in a]
    possibilities = possibilities + a
```

```
    for p in possibilities:
        count += 1
        if p == c:
            return count, c
```

5 References

www.wikipedia.org, www.cs.cornell.edu