Ciphers

The Substitution Cipher

Lecture notes of Alexander Wood
CSCI 360 Cryptography and Cryptanalysis
awood@jjay.cuny.edu

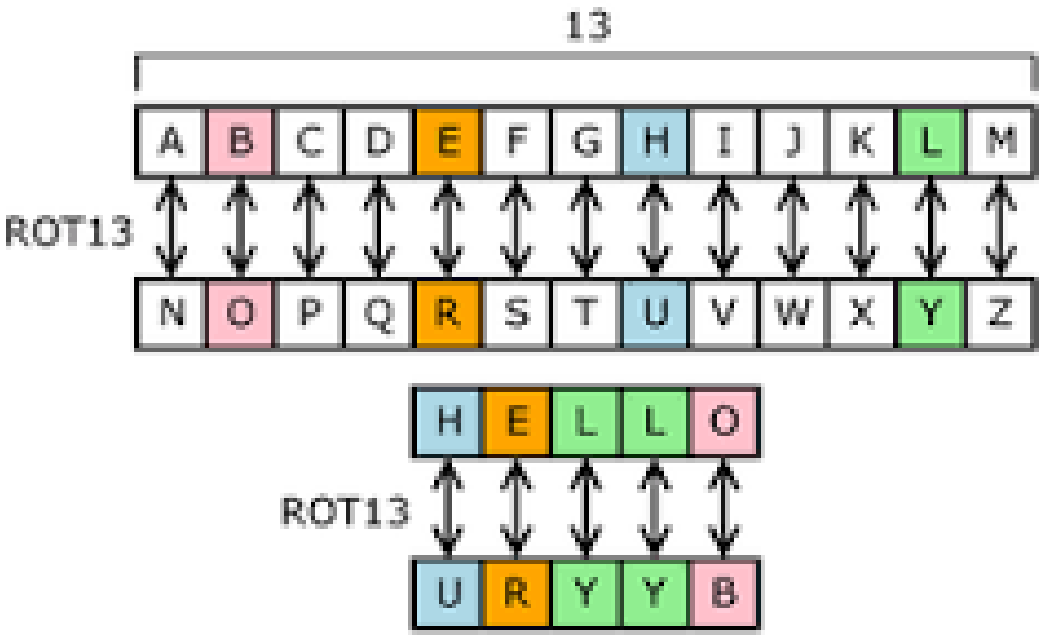
John Jay College of Criminal Justice

Shift Cipher

The first cryptosystem we will look at was called the **shift cipher**. This cipher disguised communication by shifting every letter exactly k letters to the right for some $0 \leq k \leq 25$.

Substitution Cipher

Next we look at the **substitution cipher**, which encrypts a plaintext by replacing each letter in the alphabet with a different letter in the alphabet.



Substitution Ciphers

There are many different ways which we could choose our key.

Exercise 1: Keys to Success

How many different possible keys are there?

Keys and Keyspaces

The size of the **keyspace** of the shift cipher is 26!, all possible permutations of the 26 letters in the English alphabet.

Coding Exercise 1: Key Generation

First, let's use Python generate a key for the substitution cipher via a call to the function `key_gen`.

Python Basics: The random module

Python has a module called `random` which imports some useful functions related to pseudo-random number generation. Import the module by including the following statement at the beginning of your code:

```
import random
```

Python: The Random Module

Access documentation for the random module here:
`https://docs.python.org/3/library/random.html`

Python: The Random Module

We will use the following function from the random module:

`random.`**`sample`**(*population*, *k*)

Return a *k* length list of unique elements chosen from the population sequence or set. Used for random sampling without replacement.

Returns a new list containing elements from the population while leaving the original population unchanged. The resulting list is in selection order so that all sub-slices will also be valid random samples. This allows raffle winners (the sample) to be partitioned into grand prize and second place winners (the subslices).

Disclaimer on the random module

The random module is not *cryptographically secure*, a notion which will be of great importance later in the course when we study more complex encryption methods. For now, the random module will work just fine.

Coding Exercise 2: Encrypt

Next write a function called `encrypt` which takes a plaintext and the key generated above as arguments and returns the ciphertext created by encrypting the plaintext under the given key using the substitution cipher.

Coding Exercise 3: Decrypt

Next write a function called `decrypt` which takes a ciphertext and the key generated above as arguments and returns the plaintext corresponding to the given ciphertext.

References

- The random module: `https://docs.python.org/3/library/random.html`