need to know how theses ciphers workneed to know why they are not safe

Information Systems Security
Exercices Series 3 : Historical Ciphers

October 13th, 2021

Reminders : Historical Ciphers

Cesar's cipher

Cesar's cipher works by rotating all letters in the text the same number of steps : for example, with Cesar favorite key k=3, we will replace each character in our alphabet by the character that's three steps later in the alphabet : if we work with only uppercase letters, A is replaced by D, B is replaced by E, C by F, and so on. And this works as a cycle : if we reach the end of the alphabet, we're cycling back to the beginning : with the same example, V becomes Y, W becomes Z, and then X becomes A, Y becomes B and Z becomes C.

The key is just a number, between 0 and N-1, where N is the length of the alphabet. This can of course be applied on way bigger alphabets (for example all the ASCII 8-bit), all we need is for characters to be ordered.

Monoalphabetical substitution

That is a more generical cipher, that assigns to each character of the alphabet a character from another alphabet (can be the same alphabet or a different one). A key is just a order on characters in the alphabet (can be seen as a string the same length as the alphabet, containing each character exactly once).

Let us suppose we have the initial alphabet that is all 26 uppercase letters for the following examples :

First example : The key "MONALPHBETICDFGJKQRSUVWXYZ" replaces each "A" with a "M", each "B" with a "O", each "C" with a "N", ...
In this case, the alphabet for ciphers is the same as the one for plaintexts. The key is a permutation of these characters. The word "HELLO" would be encrypted as "BLCCG".

Second example : The key ":y;kà@i?tsè)=ù*f(m!,l¡]w[p" replaces "A" with ":", "B" with "y", "C" with ";", and so on. The alphabet for ciphers is not the same one in this case. "HELLO" would be encrypted as "?à))*".

This can be applied to bigger alphabets as well, the only condition is that we need alphabets of the same size (to ensure a bijection). And we can see that Cesar is just a sub-case of this substitution (with the same alphabet for ciphertexts, and a key that is a rotation of the initial alphabet).

Vigenère's cipher

This is what we also call a polyalphabetical substitution, because a given characters can be encrypted by different characters bepending on the situation. Vigenere's cipher is a simple addition modulo N (where N is the length of the alphabet) between a key and the message. Each character is given a value between 0 and N-1.
With our usual example of uppercase letters, we have A=0, B=1, and so on to Z=25.
To obtain the encrypted character, we sum the character and the character at the same position in the key, modulo N (modulo 26 in our example. The key, which is just a chain of characters, is repeated as many times as needed to encrypt the whole text.
Note that we can also use that with different alphabets for the key and for ciphers.

Example : The message "BONJOUR" encrypted with the key "BAC" gives the ciphertext "COPKOWS" ($B + B = C$, $O + A = O$, $N + C = P$, then we repeat the key $J + B = K$, $O + A = O$, $U + C = W$ and so on).

The following figure shows this encryption as a table for the uppercase alphabet :

	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
A	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
B	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	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	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

Figure 1: Table de Vigenère

Exercice 1: Simple Encryptions and Decryptions

For this whole exercice, alphabet is the 26 uppercase letters.

1. Cesar : Encrypt the message "HELLOWORLD" with key 5.
2. Monoalphabetical : Encrypt the message "SUBSTITUTION" with the key "ILOVECRYPTGAHBDFJKMNQSUVWXZ".
3. Substitution monoalphabétique : Decrypt the message "EOHITKMTVM", knowing the key used for encryption is "AZERTYUIOPQSDFGHJKLMWXCVBN".
4. Vigenère : Encrypt the message "YOUCANTSEEME" with the key "CENA".
5. Vigenère : Decrypt the ciphertext "XIWARVETUYNOHMIMB", knowing the key used for encryption is "PIKACHU".

Exercice 2 : Breaking Monoalphabetical Ciphers

1. You just intercepted one of Cesar's encrypted messages, and you kow it has been encrypted by Cesar (and that the alphaet is only uppercase

letters). But you don't know the value of the key. Here is the ciphertext : "WIGYVMXC MW KYEVERXIIH AMXL XLMW GMTLIV". Find the original message (it is a real english sentence that has a meaning).

2. In most known languages, each letter appears in texts with different frequencies. Supposing the text is long enough, how would you use that to break Cesar's cipher instead of using brute force ?

3. If you know the language of the encrypted text, how would you use that and the previous principle to break the monoalphabetical substitution ?

4. Are Cesar and the monoalphabetical substitution good ciphers ?

Exercice 3 : Breaking Vigenere's Cipher

The goal of this exercice will be to use Python to decipher a text encrypted with Vigenere's cipher, without knowing anything about the key.

Vigenere's cipher is unconditionally secure if we respect two rules : using a key with the same length as the message, and never reusing that key (which is then a one-time pad). But in the case where the key is repeated multiple times, we can break vigenere without knowing anything but the language of the text (which is generally known from context, and even in case of doubt, we can just test with different languages).

First step : finding the key's length

To find the length of the key, we use the superposition technique : we will compare the text with a copy of itself where we cut the L first characters, for each possible value of the length L (generally from 1 to a good enough number). For each value, we will compute what we call the index of coincidence (we will use a simplified version here), which is simply the number of identical letters in this comparison (normalised by the total of comparisons). The key length that has the maximal index has a very high probability to be the length of the key used to encrypt the text (or a multiple of that length, which leads to the exact same decryption).
Let N be the length of the text, L the key length we try, A and B the two texts (A is the original, B is amputed of its first L characters), and a_i, b_i indicates the i-th character of respectively A and B (also meaning $b_i = a_{i+L}$). The index of coincidence is computed as :

$$\frac{\sum_{i=1}^{N-L}[a_i == b_i]}{N-L}$$

Where $[a_i == b_i]$ equal 1 if a_i and b_i are the same character, and 0 if not.

This works thanks to the fact that letters in known languages don't have the same probability of appearance : for example, "e" has the biggest probability of appearance in both english and french (with around 15%), whereas certain letters have a very low rate of appearance ("y", "z", "x", "k, "j", "w", "v", ...). That means when we have a multiple of the length of the key, we expect many more collisions.

Frequential analysis

If we know the length of the key, then we can easily decompose Vigenere as multiple Cesar ciphers, and then use frequential analysis to find each character of the key.

We start by dividing our cipher in k sub-texts, where k is the key's length. In the first sub-text, we put characters in position 1, k+1, 2k+1, ... from the ciphertext. All these character are the ones encrypted by the first character of the key.
And even though this sub-text has no reason to have a semantic meaning when deciphered, we can still break this "Cesar" case with a frequency analysis (as letters have different frequencies, we'll try to map the frequencies in our sub-text to the ones of the letters in the language).

So we will compare the frequencies of the characters in this sub-text, with the frequencies of the characters in the language. We'll compute the distance between these frequencies, and we'll compute this distance for every possible value of the key's first character (26 possibilities if we work only with letters). The one with the shortest distance has a very high probability to be the key's first character.

In our case, we'll use the simple euclidian distance :

If F_i are the knows mean frequencies in the language (i from 0 to 25, 0 for A, 25 for Z in our case), and M_i the frequencies in our sub-text, then for a given key first character value x, we compute the distance as :

$$Dist_x = \sqrt{\sum_{i=0}^{25}(F_i - M_{(i+x)mod26})^2}$$

We just have to compute this distance for each x (from 0 to 25 both included in our case), and then chose the x value that minimises the distance. This x is the value of the key's first character (if the distance is minimised for x=3, then we know that the first character of the key is D=3 (reminder : we start at A=0)).

By applying that to all k sub-texts (second one has characters 2, k+2, 2K+2, ... from the cipher, third one has 3, k+3, 2k+3, ... and so on), we can find one by one each character of the key.

The exercice itself : breaking Vigenère

You will find on moodle a Python file containing a ciphertext that has been encrypted by Vigenere's cipher.
You know that the original text was in english.
The file also contains the mean frequencies of letters in the english language, and the max possible length for the key : to simplify, you already know that the key's length is between 1 and 10. The alphabet in this case is only the 26 lowercase letters (nothing else, no spaces, ponctuation, uppercase letters or anything else).

Use the two principles seen before, index of coincidence and frequential analysis, to find the length of the key and then the key itself, and decipher this ciphertext.