# FIT1047 Tutorial 6

## Instructions

- The tasks are supposed to be done in groups.

## Task 0: Learn about *Monoalphabetic Substitution* via Cryptool Online

In this task you get to know Cryptool, a learning tool for cryptography. We use the online version at http://www.cryptool-online.org/. This version mainly supports classical ciphers up to after the second world war. Ciphers can directly be tested. However, as the interface is a bot clumsy for the task of guessing a key, we use a small tool in the Moodle FIT1047 Software section.

For learning about modern cryptography, the full version of the tool for Windows can be downloaded here: https://www.cryptool.org/

0.a Load Cryptool online at https://www.cryptool.org/en/cryptool-online

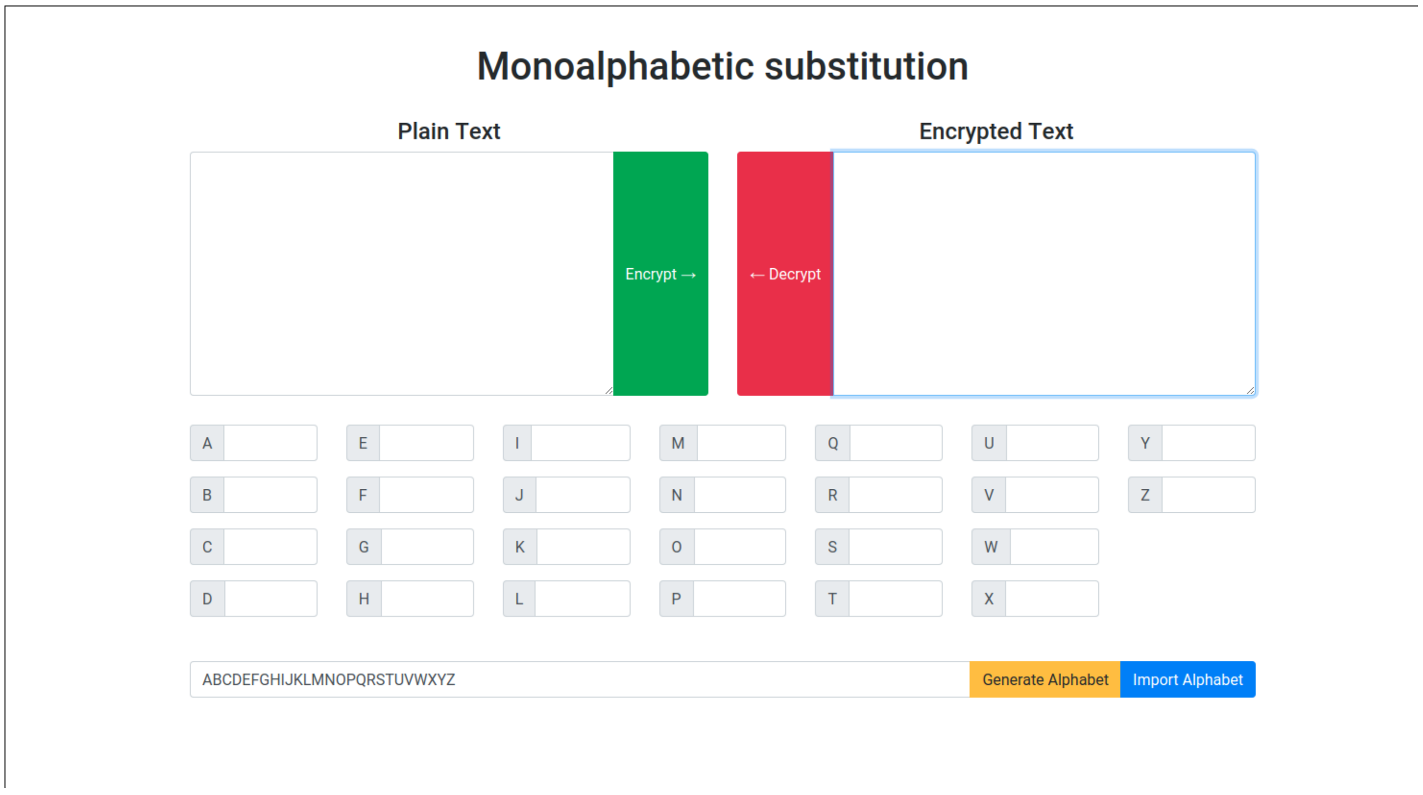
0.b Chose *Monoalphabetic Substitution Cipher* from the *Ciphers* Menu . Chose *Description* and read the text on the Monoalphabetic substitution cipher.

0.c Read the descriptions for the Caesar cipher and the Vigen'ere cipher. Why can the Vigen'ere cipher be considered more secure. Why is it still a very insecure cipher if the key-length is not long enough? Is there a variant of the Vigen'ere ciphere that is secure? (Hint: Read the *Security* sections).

## Task 1: Decryption exercise

The following ciphertext is derived from an English plaintext using a Monoalphabetic Substitution Cipher. It is not case sensitive, punctuation is unencrypted, blanks are not deleted. Use the tool under *Monoalphabetic Substitution Tool* in Moodle.

You should see the screen shown here:



A few hints before you start:

- Copy the ciphertext into the *Encrypted Text*.
- Note that while the encrypted text might contain lower case and symbols, the plaintext will be in upper case.
- Now start trying letters. E.g. if you think a ciphertext *g* should become a plaintext *P*, just write *g* into the box next to *P*.
- Next, you can click *Decrypt* and the plaintext in accordance to your guess of the key will appear in the *Plain Text* box.
- Caution: Clicking *Encrypt* will overwrite the text in the *Encrypted Text* box.

```
‘f9y2 x$m fnay mg q2 d9y u$s2q2k, g$$9,‘
jnqc gqkwyd nd wnjd,
‘f9nd qj d9y 6qsjd d9q2k x$m jnx d$ x$msjyw6?‘

‘f9nd qj 6$s rsyna6njd?‘ jnqc g$$9.
‘f9nd c$ x$m jnx, gqkwyd?‘

‘q jnx, q f$2cys f9nd qj k$q2k d$ 9nggy2 ybtqdq2k d$cnx?‘ jnqc gqkwyd.
g$$9 2$ccyc d9$mk9d6mwwx. ‘qd qj d9y jnuy d9q2k,‘ 9y jnqc.
```

Try to answer the following questions:

1.1 Which approach would you chose to start an analysis of this ciphertext?

1.2 What is the plaintext?

1.3 Which key is used?

## Task 2: RSA using pencil and paper

The following text provides a short example for RSA just using small numbers. It was written by Serge Matikov, but it is no longer available online. In principle, it can be done just using pencil and paper (for some steps a calculator is useful...)

2.1 Read the text. What is the server's secret key and what is the public key.

2.2 Calculate encryption and decryption with message P=13.

### What is the RSA Algorithm

OK, here is what we want to do: We have a *piece of data* that we want to somehow *scramble* so nobody can learn what this data is, and we want to send this data over unsecure lines to the recipient. Upon receipt of this scrambled data, the recipient must be able to *unscramble* this data to its original shape. The important thing here is that we want to do this *scrambling/unscrambling* process without requiring usage of any secret keys that both the sender and the recipient must posses in order to scramble and descramble the data. This is why the method we are going to discuss here is called *Public Key Cryptography*. There are several Public Key Cryptography algorithms in use today. The most popular is called RSA algorithm, and is named after the initials of its inventors: R for Rivest, S for Shamir, and A for Adelman. By the way, they were students when they invented this algorithm in 1977.

So here is the summary of operations. Please continue reading below for the detailed explanation of how this is achieved. Let's say that your WEB Browser has a piece of data, say number 14 (we'll call it a Plain message and label it as P=14). and it wants to encrypt this Plain message first and then send it to the Server.

Upon receipt of this encrypted message, the Server wants to decrypt it to its original value. Here is the summary of what transpires. Before any communication happens, the Server had calculated, in advance, its public ($n = 33$ and $e = 7$) and private ($d = 3$) keys.

Now, to initiate the transaction, the Browser sends this message to the server: Hey Server, please send me your public key. The Server obliges: Here it comes, it's $n = 33$, $e = 7$. After receiving the Server's public key, the Browser converts the Plain message $P = 14$ into the Encrypted message $E = 20$ and sends it to the Server.

The Server receives this encrypted message $E = 20$ and using its secret key $d = 3$ (and publicly known key $n = 33$) decrypts the $E = 20$ message into its original Plain message $P = 14$.

Now, let's look a bit more into the math behind all this.

## Section1. Generating Public and Private Keys

First, as we mentioned above, before any transmission happens, the Server had calculated its public and secret keys. Here is how.

1.1 pick two prime numbers, we'll pick $p = 3$ and $q = 11$

1.2 calculate $n = p * q = 3 * 11 = 33$

1.3 calculate $z = (p - 1) * (q - 1) = (3 - 1) * (11 - 1) = 20$

1.4 choose a prime number $e$, such that $e$ is co-prime to $z$, i.e, $z$ is not divisible by $e$. We have several choices for $e$: 7, 11, 13, 17, 19 (we cannot use 5, because 20 is divisible by 5). Let's pick $e = 7$ (smaller k, "less math").

1.5 So, the numbers $n = 33$ and $e = 7$ become the Server's public key.

1.6 Now, still done in advance of any transmission, the Server has to calculate it's secret key. Here is how.

1.7 $e * d = 1(mod\ z)$

1.8 $e * d = 1(mod\ 20)$

1.9 $(7 * d)/20 =?$ with the remainder of 1 (the "?" here means: "something, but don't wory about it"; we are only interested in the remainder). Since we selected (on purpose) to work with small numbers, we can easily conclude that 21 / 20 gives "something" with the remainder of 1. So, $7 * d = 21$, and $d = 3$. This is our secret key. We MUST NOT give this key away.

Now, after the Server has done the above preparatory calculations in advance, we can begin our message transmission from our Browser to the Server. First, the Browser requests from the Server, the Server's public key, which the Server obliges, i.e., it sends n=33 and e=7 back to the Browser. Now, we said that the Browser has a Plain message P=14, and it wants to encrypt it, before sending it to the Server.

**Section 2. Encrypting the message**

Here is the encryption math that Browser executes.

2.1 $P^e = E(mod\ n)$
P is the Plain message we want to encrypt
n and e are Server's public key (see Section 1)
E is our Encrypted message we want to generate

After plugging in the values, this equation is solved as follows:

2.2 $14^7 = E(mod\ 33)$ This equation in English says: raise 14 to the power of 7, divide this by 33, giving the remainder of E.

2.3 105413504/33 = 3194348.606 (well, I lied when I said that this is "Pencil and Paper" method only. You might want to use a calculator here).

2.4 $3194348 * 33 = 10541348$

2.5 $E = 105413504 - 10541348 = 20$

So, our Encrypted message is $E = 20$. This is now the value that the Browser is going to send to the Server. When the Server receives this message, it then proceeds to Decrypt it, as follows.

**Section 3. Decrypting the Message**
Here is the decryption math the Server executes to recover the original Plain text message which the Browser started with.

3.1 $E^d = P(mod\ n)$
E is the Encrypted message just received
d is the Server's secret key
P is the Plain message we are trying to recover
n is Server's public key (well part of; remember that Server's public key was calculated in Section 1 as consisting of two numbers: n=33 and e=7).

After plugging in the values:

3.2 $20^3 = P(mod\ 33)$

3.3 8000/33 =? with the remainder of P. So to calculate this remainder, we do:

3.4 $8000/33 = 242.424242...$

3.5 $242 * 33 = 7986$

3.6 $P = 8000 - 7986 = 14$, which is exactly the Plain text message that the Browser started with!

Well that's about it. While we did not discuss the theory behind the formulae involved I hope that you got at least a basic idea of how the public key cryptography using the RSA algorithm works. example of the RSA Algorithm

**Section 4. Cracking the Code**

The essential requirement of the Public Key Cryptography is that the public and secret keys are mathematically related, but this relationship must be made very hard to determine by an outsider.

As you saw in the preceding text, everything starts with p and q, from which we calculated n. The public key consists of two numbers: n and e, where e is calculated from z and z is calculated from p and q. The secret key d, was calculated from k and z and, as we just stated, e and z are calculated from p and q. It follows then, that d is also calculated from p and q,which proves that the public and private keys are mathematically related.

So, if an outsider wanted to find the secret key d, by only knowing n, he can do it by breaking down n into the two prime numbers that were used to produce it (remember that n = p * q). Now, here is the real crux of the bisquit: Decomposing a very large n into p and q is really difficult to do. It is easy with the small numbers that we have used in our demonstration, but try, for example decomposing p into p and q when p has several hundred digits.