



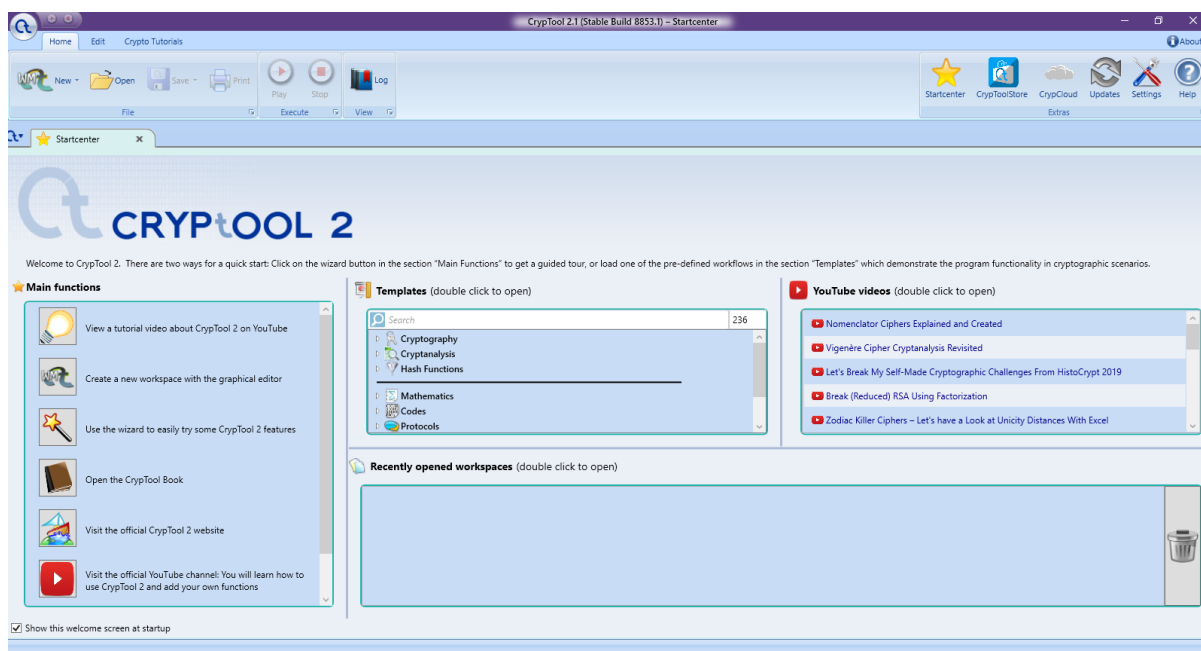
K. J. Somaiya College of Engineering, Mumbai-77

Internal Assessment 2

Implementation of the CrypTool 2.1

CrypTool is an open-source project that focuses on the free e-learning software CrypTool illustrating cryptographic and cryptanalytic concepts.

CrypTool implements more than 400 algorithms. Users can adjust these with own parameters. To introduce users to the field of cryptography, the organization created multiple graphical interface software containing an online documentation, analytic tools and algorithms. They contain most classical ciphers, as well as modern symmetric and asymmetric cryptography including RSA, ECC, digital signatures, hybrid encryption, homomorphic encryption, and Diffie–Hellman key exchange. Methods from the area of quantum cryptography (like BB84 key exchange protocol) and the area of post-quantum cryptography (like McEliece, WOTS, Merkle-Signature-Scheme, XMSS, XMSS_MT, and SPHINCS) are implemented. In addition to the algorithms, solvers (analyzers) are included, especially for classical ciphers. Other methods (for instance Huffman code, AES, Keccak, MSS) are visualized.



CrypTool 2 (CT2) consists of six main components: Startcenter, Wizard, Workspace Manager, Online Help, Templates, and CrypCloud

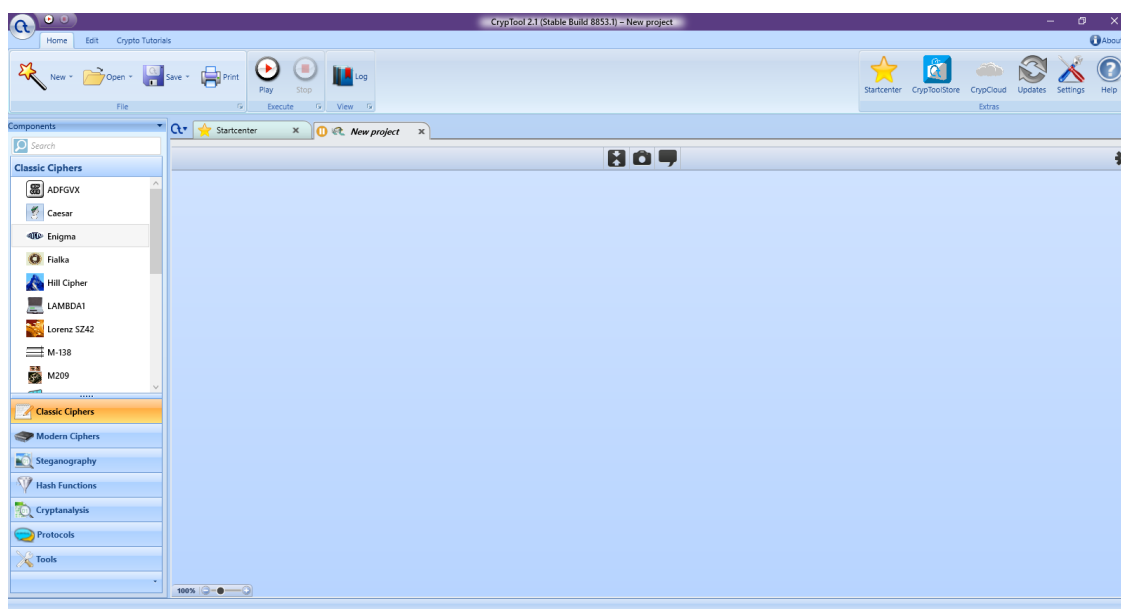


K. J. Somaiya College of Engineering, Mumbai-77

We will be performing the following:

- Caesar Encryption Algorithm
- Caesar Decryption Algorithm
- AES Encryption Algorithm
- AES Decryption Algorithm
- Exploring Watermark creator in CrypTool
- RSA Encryption Algorithm
- RSA Decryption Algorithm
- RSA With Big Numbers
- Wizard in CrypTool

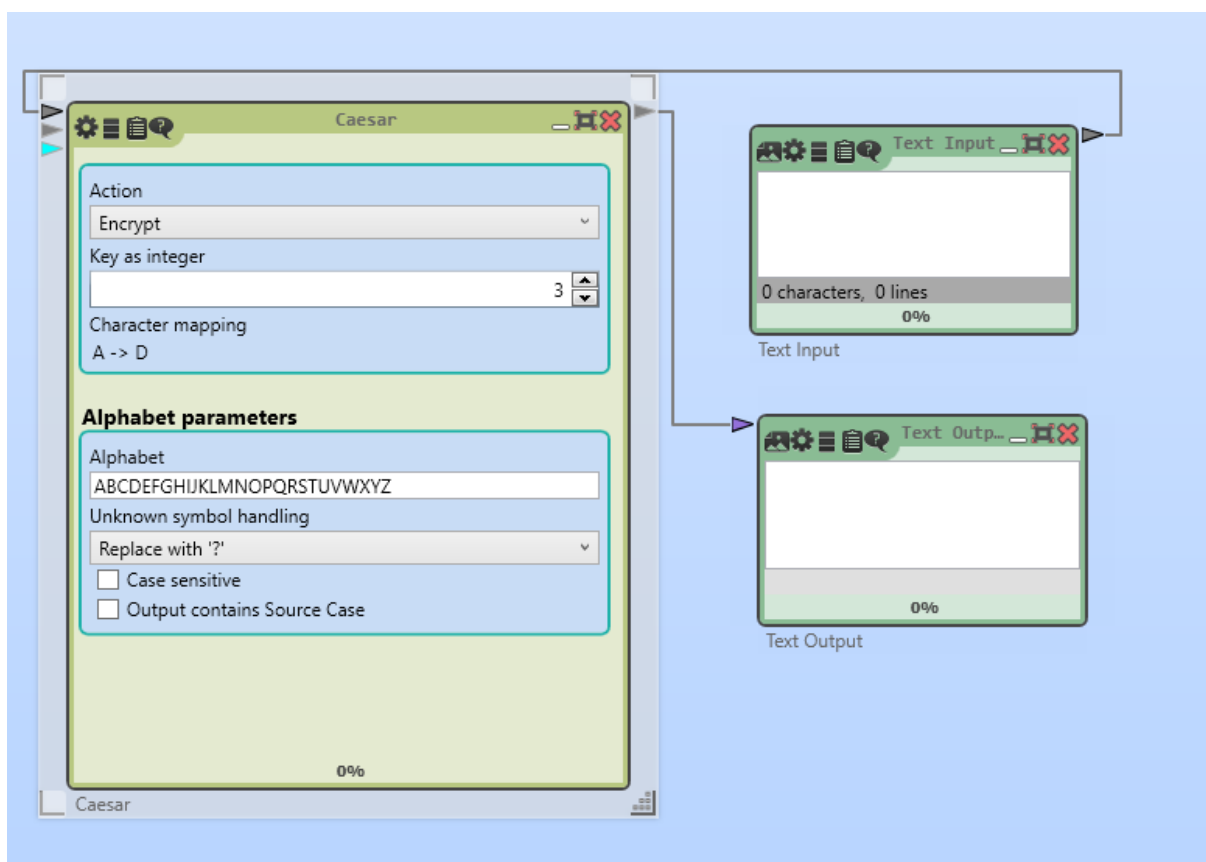
Creating New Project



Performing Caesar Cipher

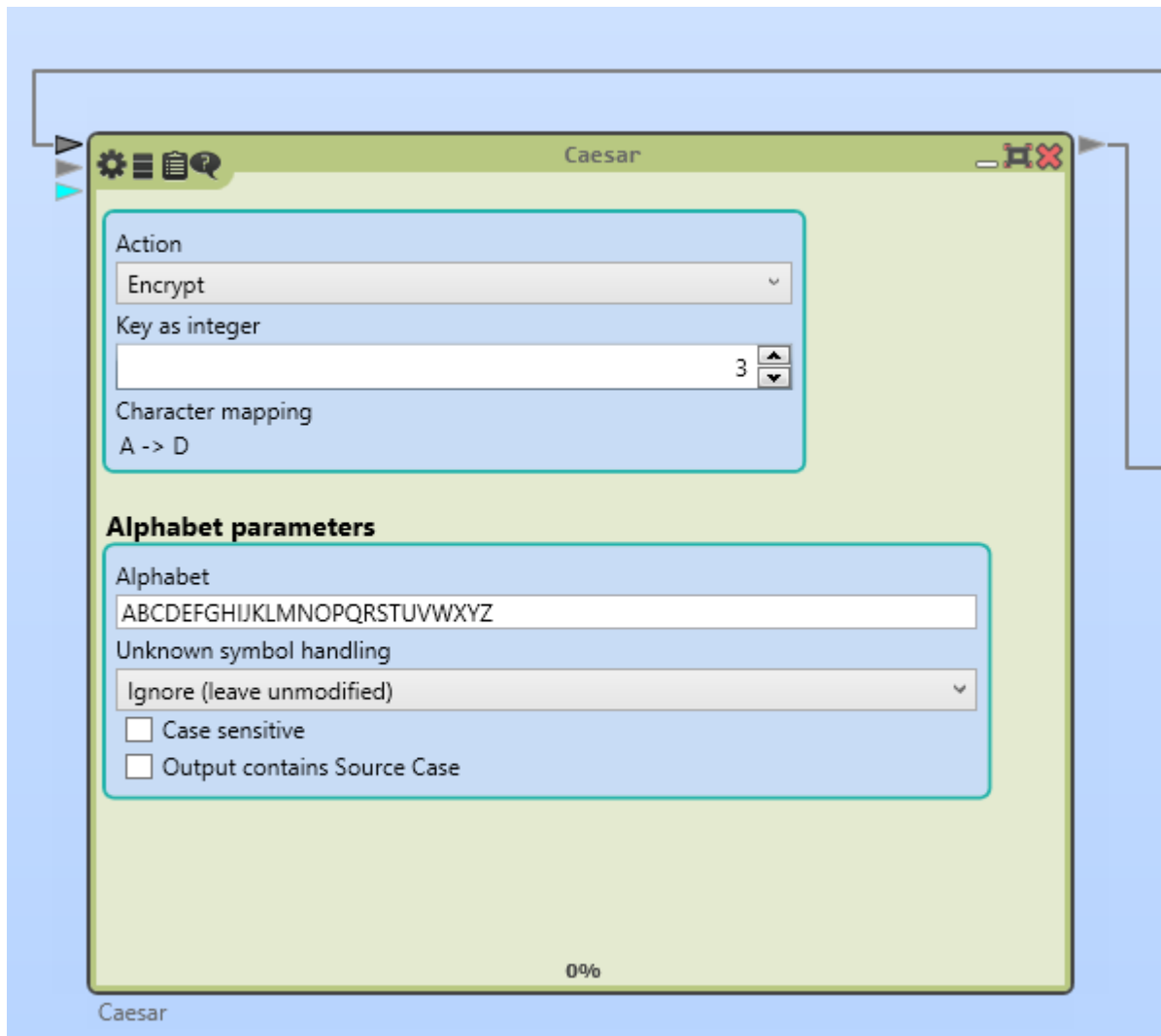
The Caesar Cipher technique is one of the earliest and simplest method of encryption technique. It's simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter some fixed number of positions down the alphabet. For example with a shift of 1, A would be replaced by B, B would become C, and so on. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.

Thus to cipher a given text we need an integer value, known as shift which indicates the number of position each letter of the text has been moved down. The encryption can be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, $A = 0, B = 1, \dots, Z = 25$.



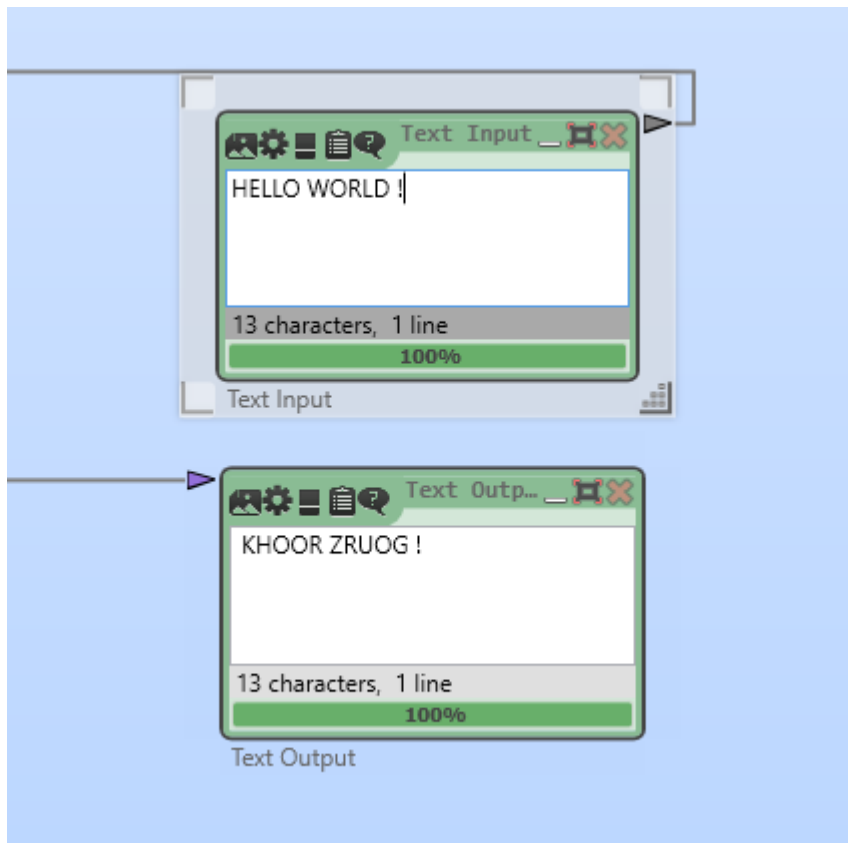
Here Caesar Window is present at the left which is set to encrypt and replace unknown symbols (other than alphabets) with question mark as provided in the options. The shift is set to 3. Following in the right are the input and output boxes.

Encryption with Caesar Algorithm



Here we will leave the unknown symbols ignored.

Below is the output for the same



Alphabet parameters

Alphabet
ABCDEFGHIJKLMNOPQRSTUVWXYZ

Unknown symbol handling
Replace with '?'

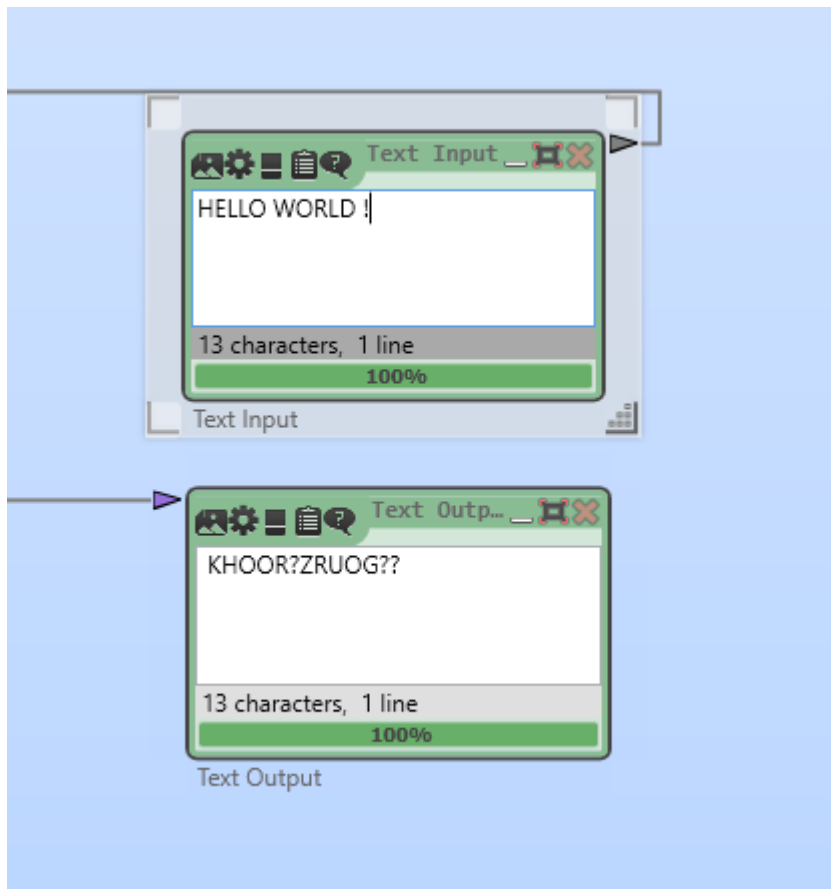
☐ Case sensitive

☐ Output contains Source Case

Here we will leave the unknown symbols replaced with “?” keeping same shift value.

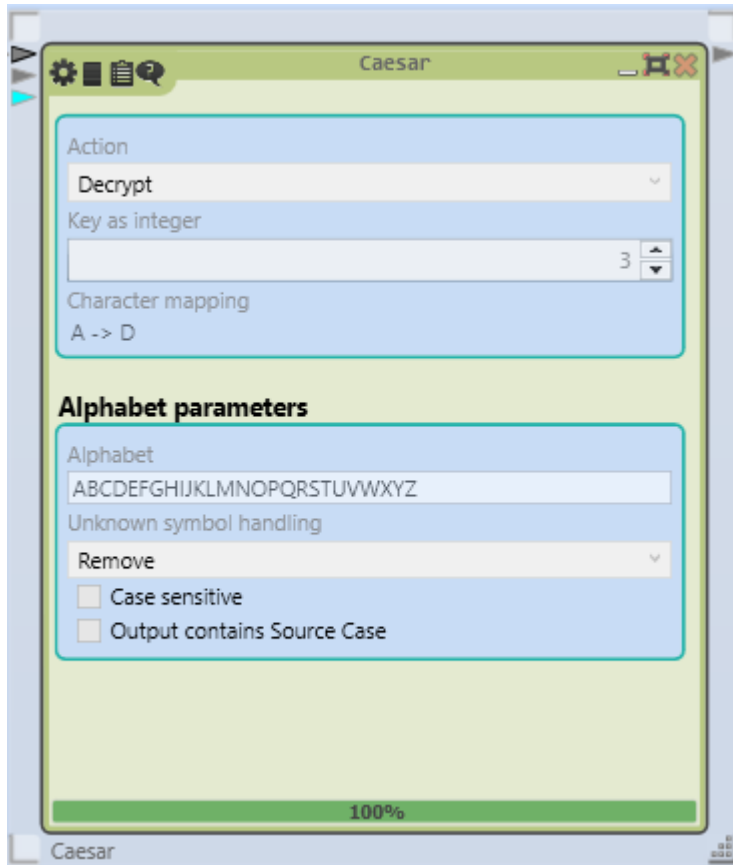
Below is the output for the same

K. J. Somaiya College of Engineering, Mumbai-77



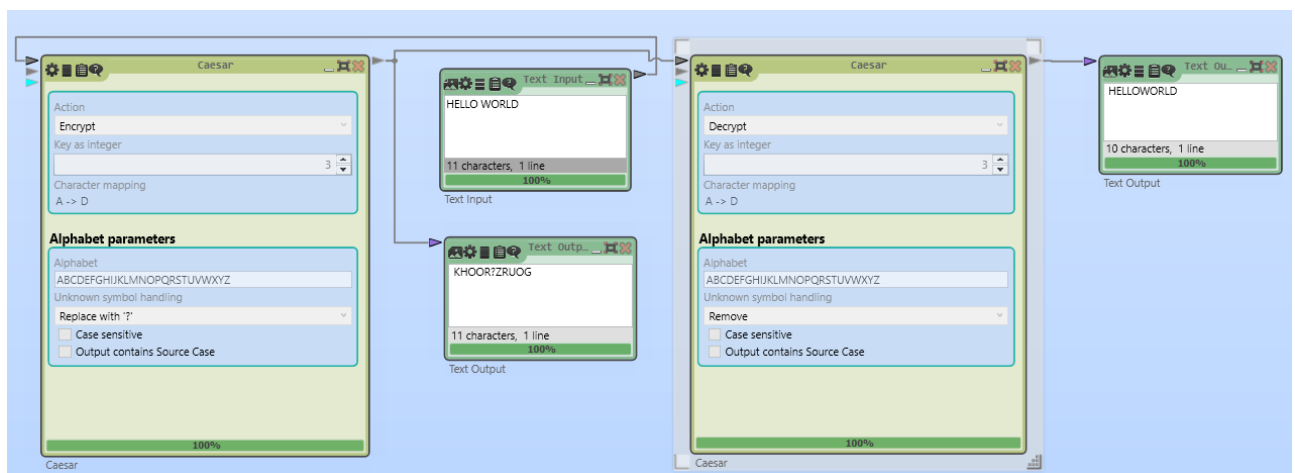
Decryption with Caesar Algorithm

Adding new Caesar Window to the project for Decryption



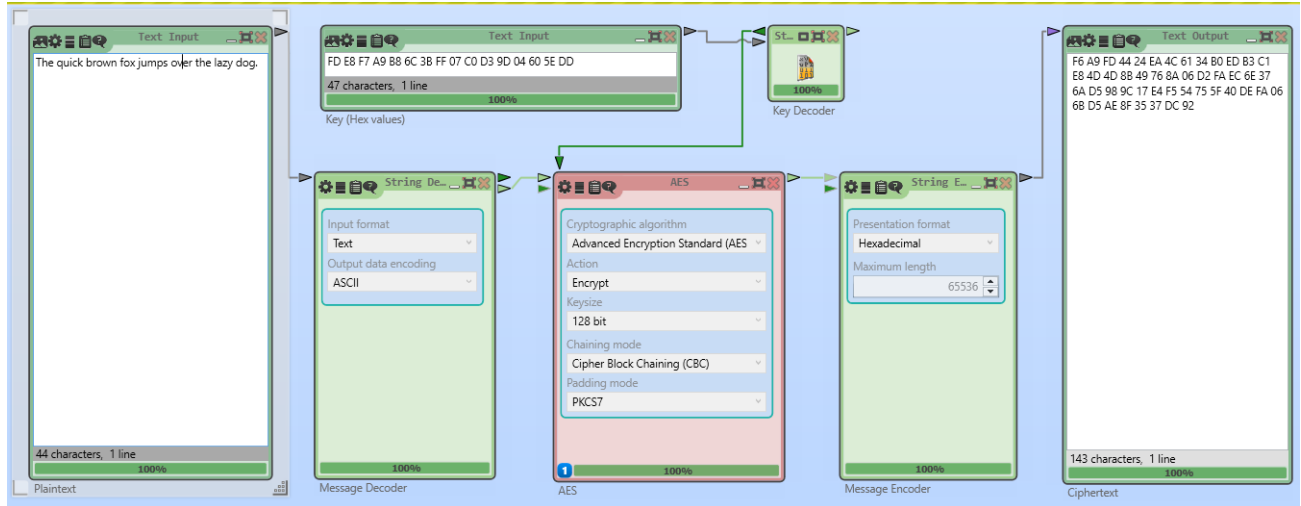
Here we set the action to Decrypt keeping the other default settings.

Hence after adding, the project window looks something like this:

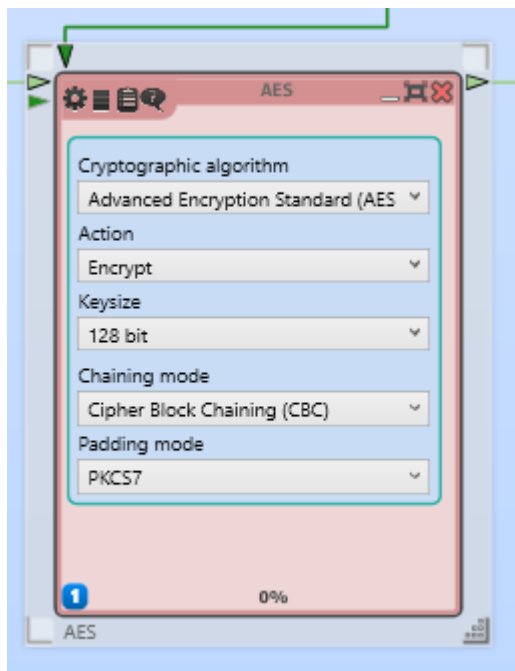


AES Algorithm

Encryption

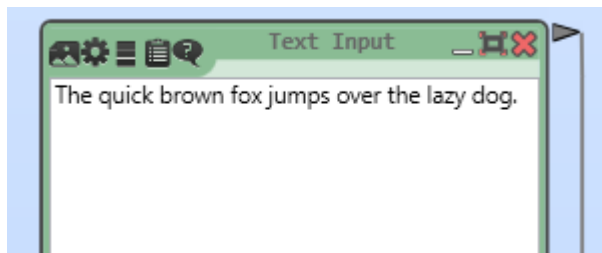


Here we have used the **AES** component to encrypt an arbitrary text entered in the **Plaintext** component on the left side. The resulting encrypted text is displayed in the **Ciphertext** component on the right side after hitting the Play button. The **AES** component works on binary values, i.e. bytes. Thus, the inputted text is first converted to bytes with the **Message Decoder** component. With the current settings, it is interpreted as ASCII. The resulting bytes are then encrypted with **AES**, yielding another sequence of bytes. These bytes are then simply printed as hexadecimal values with the help of the **Message Encoder** component.

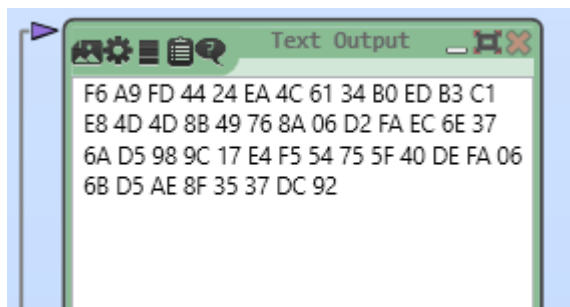


K. J. Somaiya College of Engineering, Mumbai-77

Here the input is

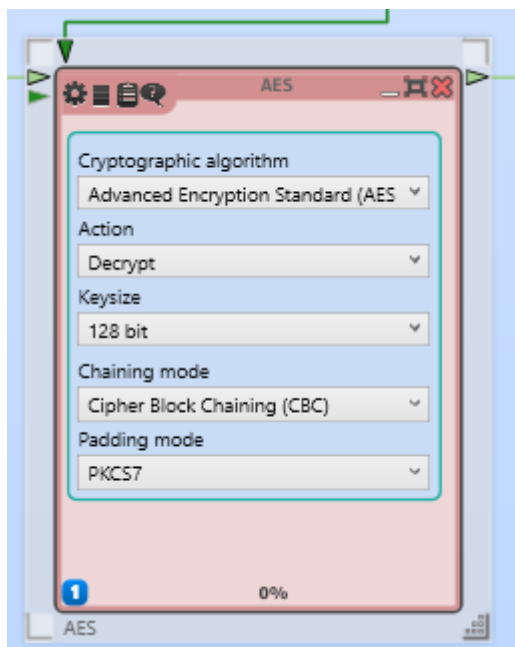


And receive the hexadecimal output as this



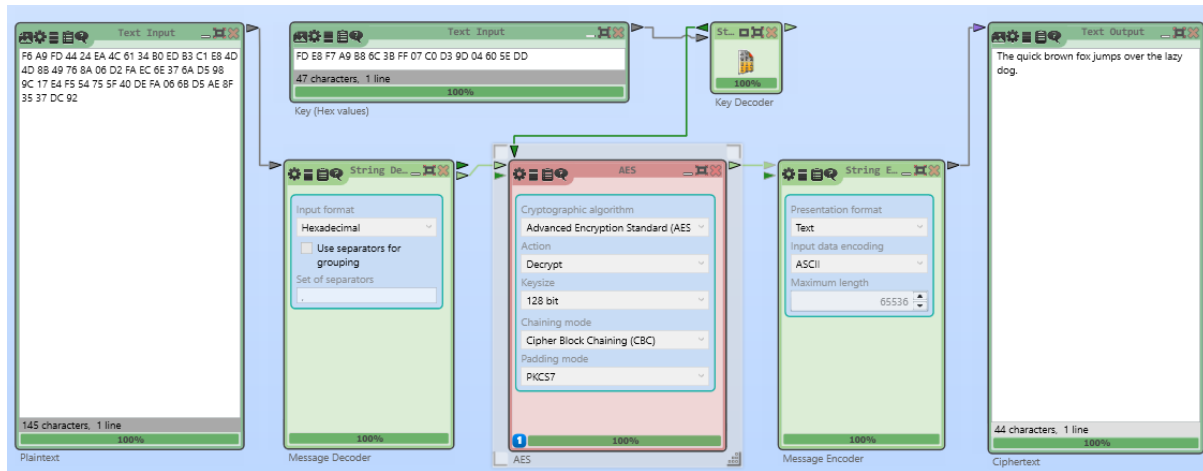
Decryption

Here we set the action to Decrypt and rest of the components are added same as we did for encryption

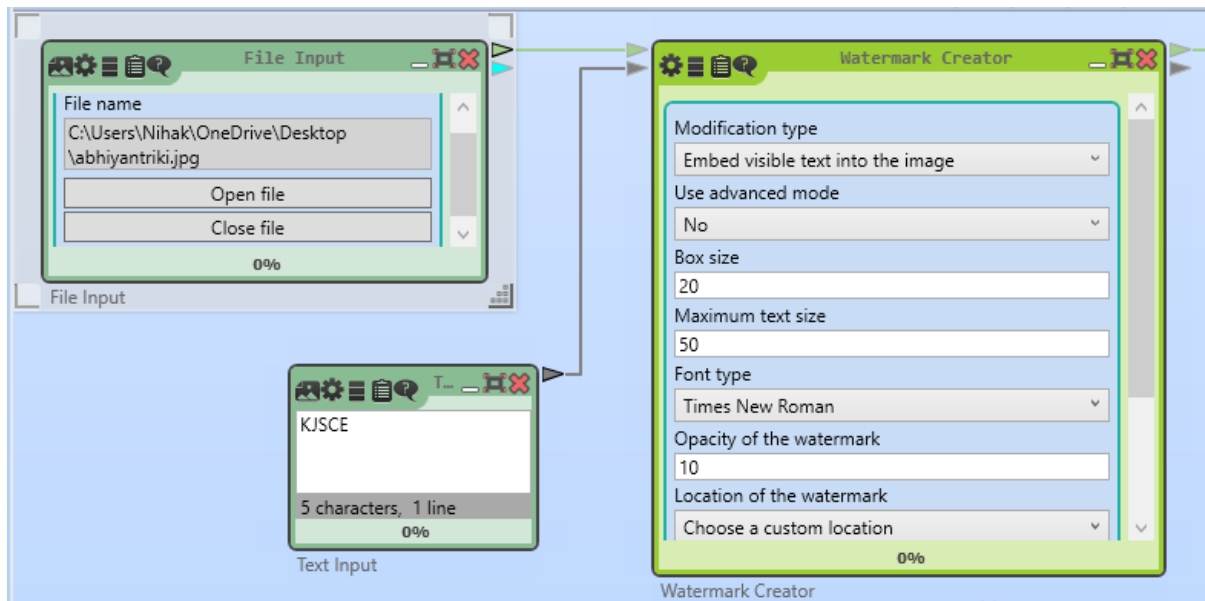


K. J. Somaiya College of Engineering, Mumbai-77

After adding decryption part to the project, the window looks as this:



WaterMark Creator in Cryptool

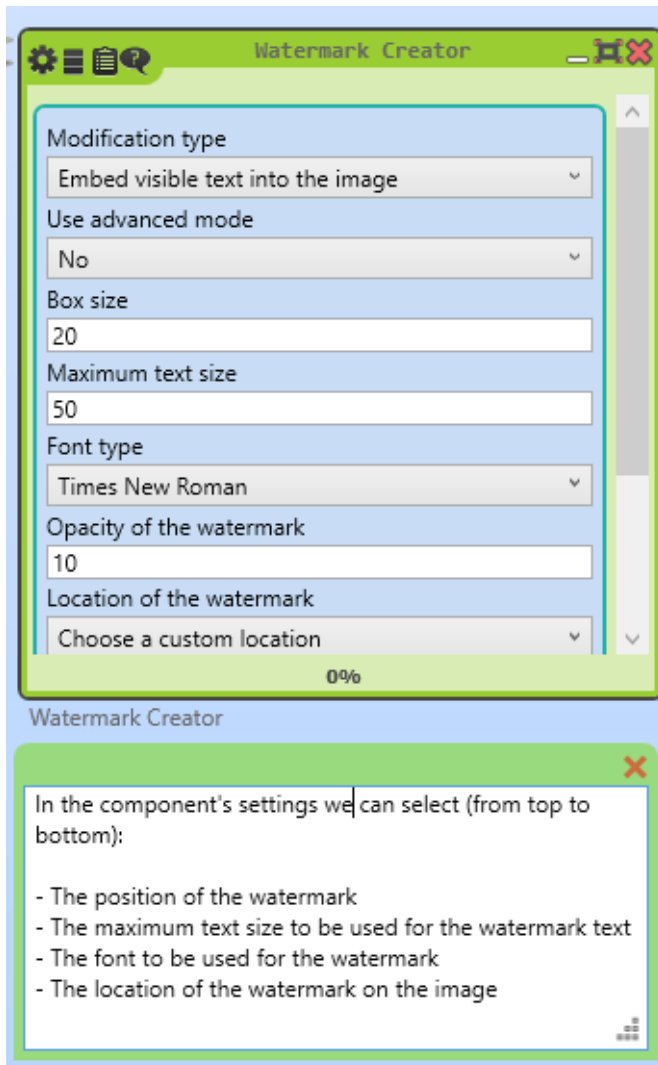


Here we show how to embed a visible text as a watermark into an image.

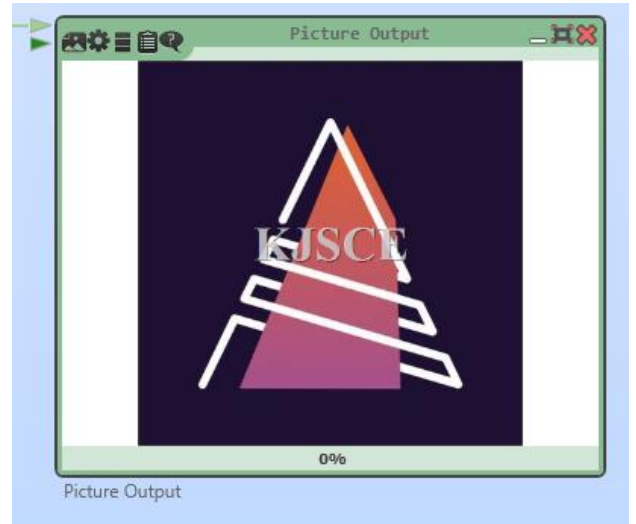
In the "File input" component we can choose the image to be used.

In the "Text input" component we can write a text that is added to the image as an watermark.

This text can be changed while the component is being executed.

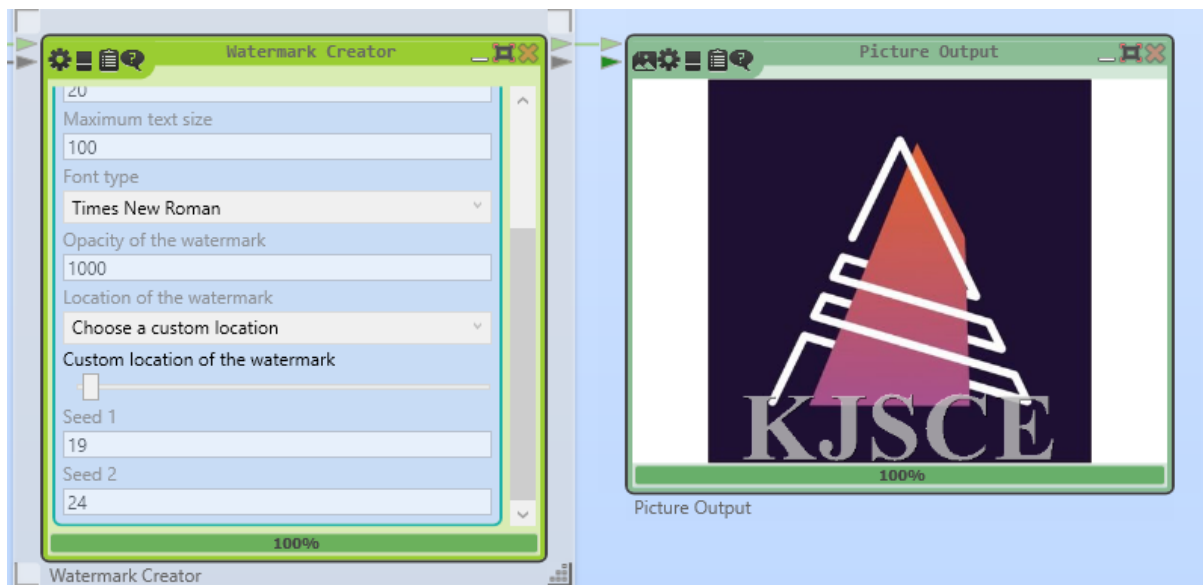


Output:



K. J. Somaiya College of Engineering, Mumbai-77

After changing size of watermark text and position:

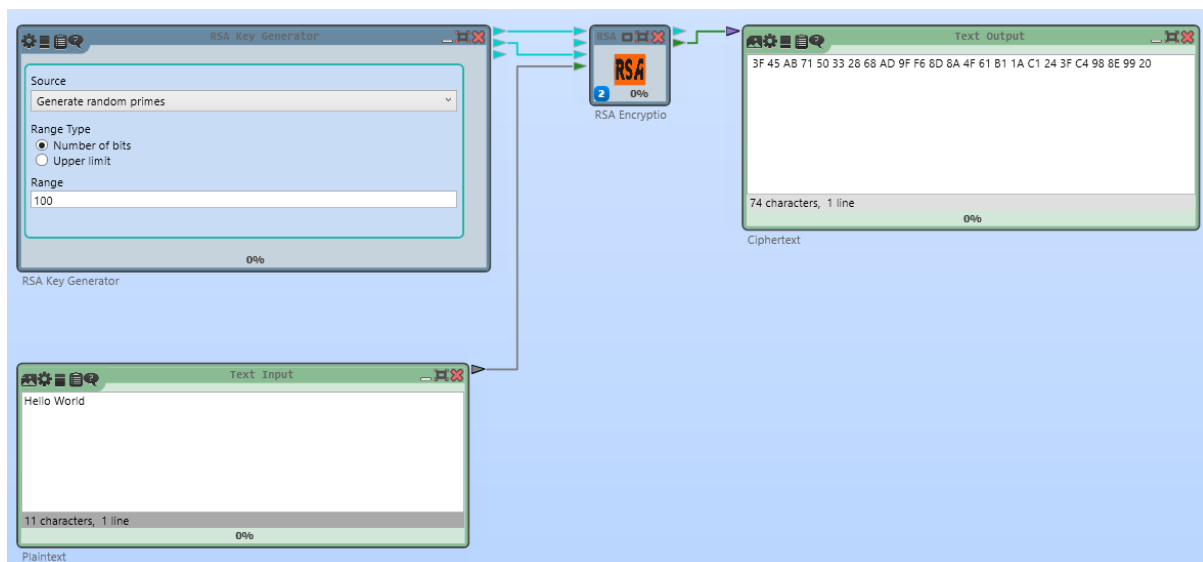


RSA Algorithm

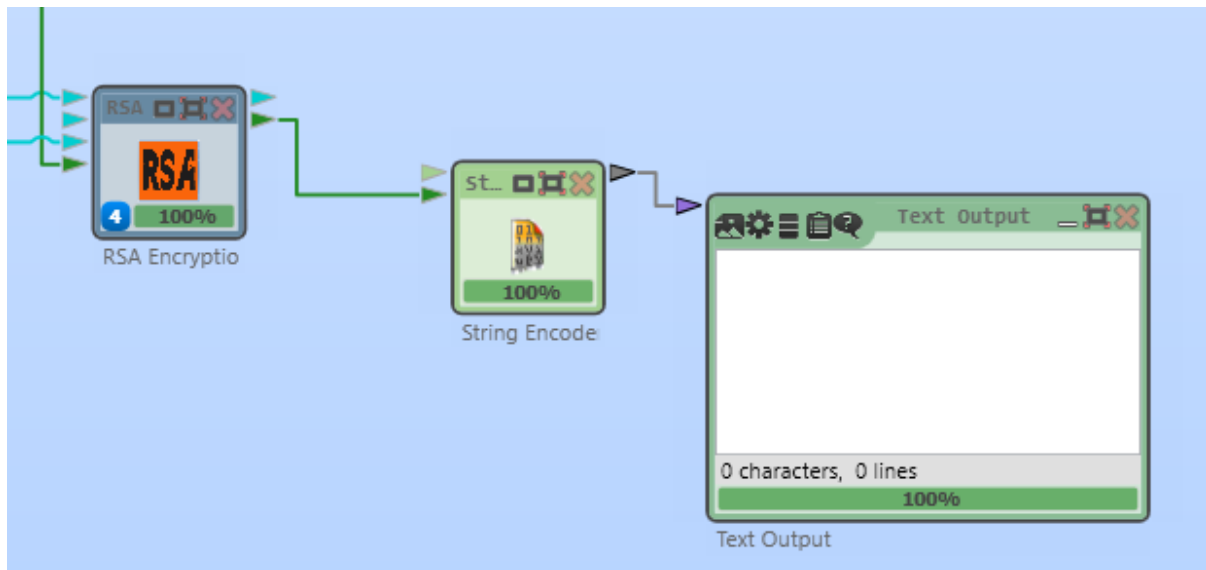
In cryptography, RSA (which stands for Rivest, Shamir and Adleman who first publicly described it) is an algorithm for public-key cryptography. It is the first algorithm known to be suitable for signing as well as encryption, and was one of the first great advances in public key cryptography. RSA is widely used in electronic commerce protocols, and is believed to be sufficiently secure given sufficiently long keys and the use of up-to-date implementations.

The RSA plugin has two ways to be used (you can use only one or both simultaneously): The user can use RSA to encrypt and decrypt numbers on the one hand or texts on the other hand. To encrypt/decrypt numbers, the user has to provide a number at the number input **m/c** of the plugin representing the message **m** or the ciphertext **c**. The resulting number is then provided at the number output connector **m/c** of the plugin. He also has to add a Number Input **N** representing the modulus **N** of the RSA algorithm. And the last input he has to add is a Number Input **e/d** which represents the public exponent or the private key depending on encryption or decryption. To encrypt/decrypt texts, the user has to add an input byte array and an output byte array. Also the direction (encrypt or decrypt) has to be set in the plugins options. He has also to add a Number Input **N** representing the **N** of the RSA algorithm. And the last input he has to add is a Number Input **e/d** which represents the public exponent or the private key depending on encryption or decryption.

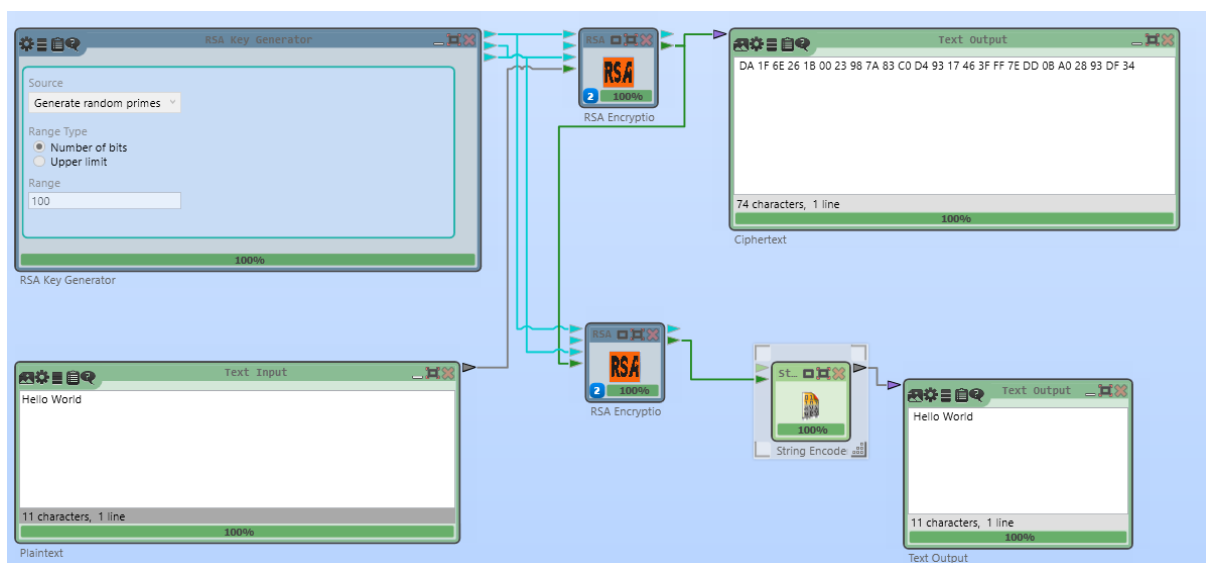
Encryption



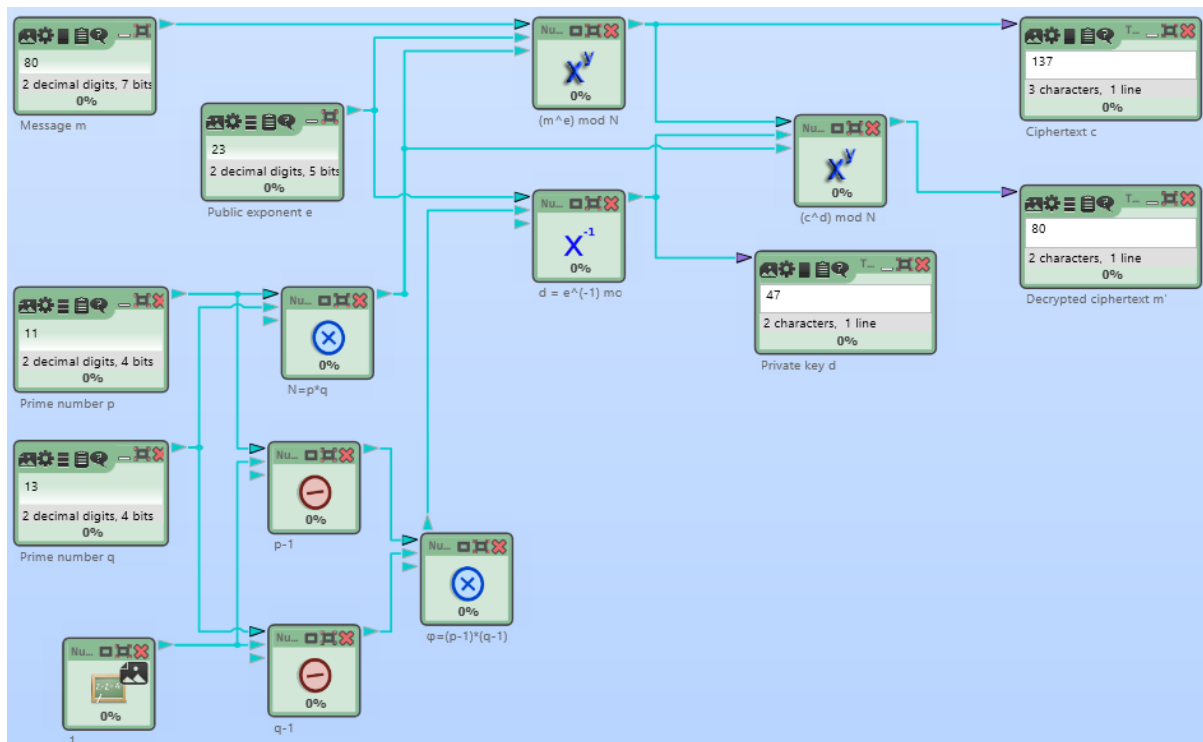
Creating **Decryption** components



Adding Decryption to the above components gives following results:



RSA With big numbers



Here we demonstrate RSA algorithm -- built up with simple mathematical components.

RSA uses a public exponent **e** and a private exponent **d**. The exponent **e** is part of the public key (**e**, **N**) which can be known to everyone. Messages encrypted with the public key **e** can only be decrypted using the private key **d**.

To encrypt a message **m**, the ciphertext **c** is calculated by $c = m^e \bmod N$ where **N** is the product of two randomly chosen prime numbers: $N = p * q$.

To decrypt a ciphertext **c**, the message **m** is calculated by $m = c^d \bmod N$.

d is the multiplicative inverse of **e mod phi(N)** where $\phi(N) = (p-1)(q-1)$.

To decrypt a stolen ciphertext, an attacker has to calculate **d**. As he only knows **N** and **e**, but not **p**, **q** or $\phi(N)$, he has to factorize **N** and do the key generation again. However, with big **N** (size = more than 1024 bits) this takes centuries.



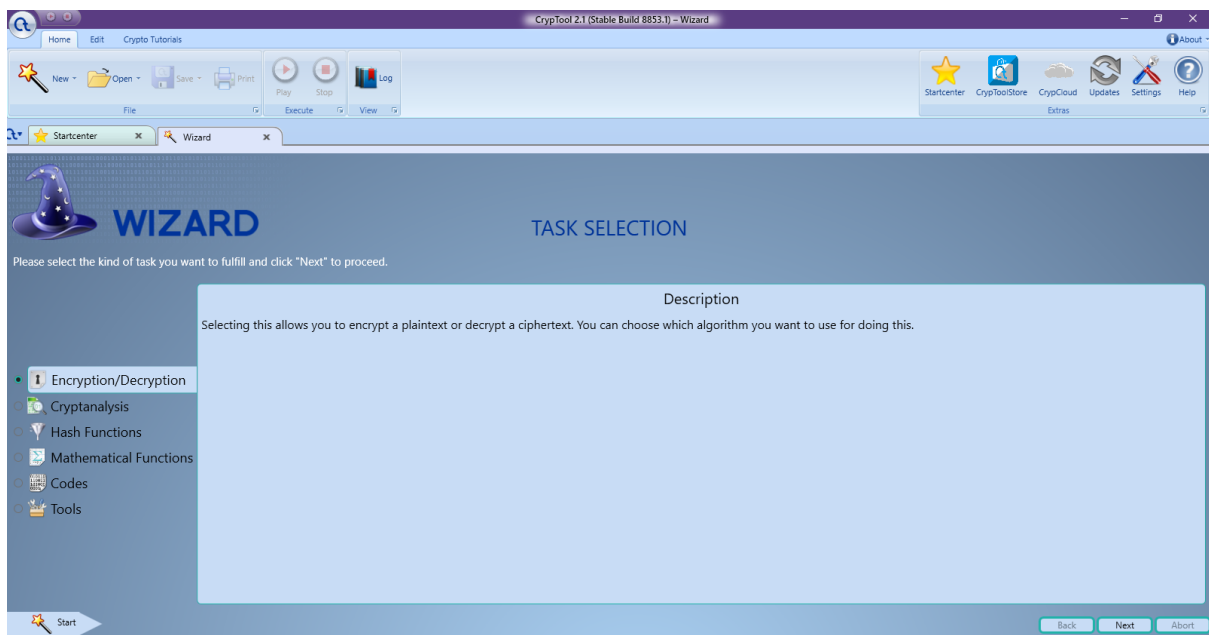
K. J. Somaiya College of Engineering, Mumbai-77

Wizard in CrypTool

The Wizard is intended for users not familiar with using the graphical programming language of the Workspace Manager and for beginners. It guides you through the different topics of cryptology until you “reach what you want to do”, e.g. encrypt something or break something. The Wizard can be started at two different places:

First, by clicking in the top ribbon bar on the new icon and selecting “Wizard”.

Secondly, it can be started using the Startcenter and clicking here on the “Magic wand” button



The Panel in the left suggests all the algorithms that we could choose to perform in the wizard. Bottom right corner has the next button to basically hit enter for the options chosen.

We will be demonstrating Caesar Encryption Algorithm in the wizard

K. J. Somaiya College of Engineering, Mumbai-77

 **WIZARD** **ALGORITHM SELECTION**



Select a classic encryption/decryption algorithm.


- ☒  Caesar
- ☐  Vigenère
- ☐  Substitution
- ☐  Enigma
- ☐  Playfair
- ☐  ADFGVX
- ☐  XOR
- ☐  Transposition
- ☐  Scytale

Description

This **encryption** algorithm, already used by the Roman general and dictator Caesar, substitutes single letters: The substitution rule is that the letters of the cleartext alphabet are shifted by a special value (key).

 Start  Encryption/Decr  Back  Next  Abort








 Startcenter  Wizard

 **WIZARD** **MESSAGE INPUT**

Here, you can input the message and the key to use.

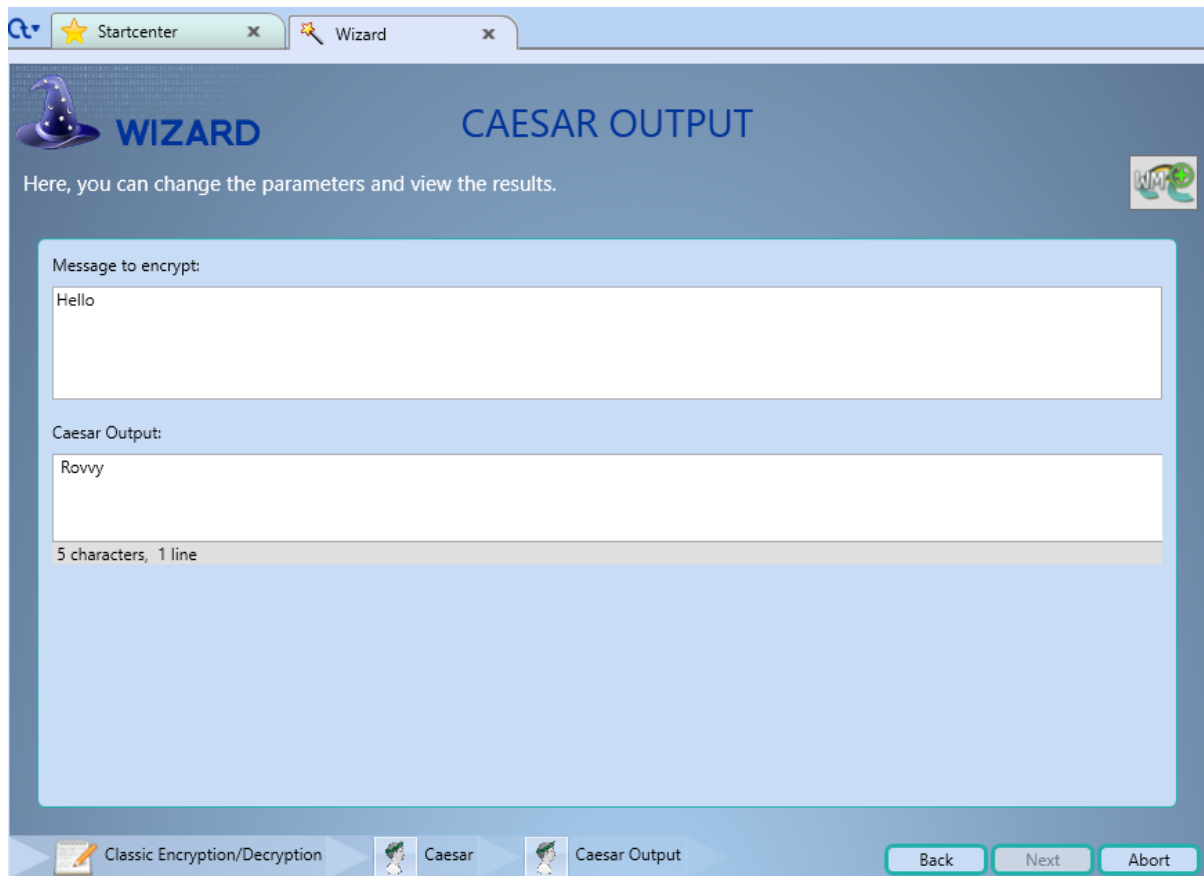
Encrypt or Decrypt:

Message to **encrypt**: Shift value (integer):

 Start  Encryption/Decryption  Classic Encryption/Decryption  Caesar  Back  Next  Abort



K. J. Somaiya College of Engineering, Mumbai-77



Team Members

- NIHA SHAIKH - 1921006
- HEENA KASALI - 1921003