Workshop – HistoCrypt 2019

Breaking Homophonic Substitution Ciphers with CrypTool 2

Nils Kopal

2019-06-26

Introduction

In the last year's workshop, we focused on working with breaking classical ciphers in CT2 in general. In this year's workshop, you will learn how to work with CT2 to break homophonic ciphers. You will learn (or repeat) the basic handling of the graphical programming language. After that, we will break monoalphabetic substitutions, polyalphabetic substitutions, and homophonic substitutions. Finally, we have a challenge part where you can break different ciphers on your own.

Structure of this workshop

The workshop is structured into different chapters in which we will show you how to use CT2:

1. Basics of Cryptology

Cryptology is the science comprised of secret writing (**cryptography**) and recovering of the secret texts without the knowledge of the secret keys (**cryptanalysis**).

Cryptographic algorithms are designed by a **cryptographer** and cryptanalysis is performed by a **cryptanalyst**.

A **cipher** is a special cryptographic algorithm used for encryption and decryption. For encryption, the input of a cipher is a **plaintext** and a (secret) **key** and the output is a **ciphertext**. For decryption the input is a ciphertext and a key and the output is the revealed plaintext. The type of the key is based on the type of the cipher and can consist of letters, numbers, machine settings, and so on.

```
Cipher(plaintext, key) = ciphertext
Cipher(ciphertext, key) = plaintext
```

In classical ciphers the key for encryption and for decryption is the same. All possible keys of a cipher define the **keyspace** of a cipher. With some ciphers, for example the Caesar cipher, it is possible to automatically test each key, since the keyspace of the cipher is very small (Caesar has 26 possible keys). But many classical ciphers have so many possible keys, that searching through the complete keyspace is impractical. In such cases, often **heuristics** can be used to break a cipher.

Breaking a ciphertext means, to reveal the plaintext without being in possession of the used key.

The used letters or symbols of plaintext and ciphertext are defined by **alphabets**. With some ciphers the alphabets are the same, with some they differ. Thus, we have a **plaintext alphabet** and a **ciphertext alphabet**.

Example: The Caesar Cipher

The Caesar cipher just shifts each letter in the alphabet according to a key (shift value).

Plaintext alphabet:	`ABCDEFGHIJKLMNOPQRSTUVWXYZ`
Key:	1 (i.e. shift alphabet by 1)
Ciphertext alphabet:	`BCDEFGHIJKLMNOPQRSTUVWXYZA`
Plaintext:	`HELLOWORLD`
Ciphertext:	`IFMMPXPSME`

a) Attacks on Ciphers

We differentiate between various attack types, depending on the knowledge of the attacker.

The **ciphertext-only attack** reveals the plaintext and/or the secret key. The cryptanalyst is here only in possession of the ciphertext. This is the strongest and most difficult attack on a cipher.

The **known-plaintext** attack reveals the key. Which then can be used to break other ciphertexts encrypted with the same key. Here, the cryptanalyst is in possession of the plaintext and the according ciphertext. If the cryptanalyst is only in possession of parts of the plaintext, we call that a **partially known-plaintext** attack.

b) Statistics

Based on language models and text statistics, it is often possible to break classical ciphers – even by hand. The letter frequency can be used, for instance, to identify which plaintext letter is replaced by which ciphertext letter. For example, the letter 'E' is the most frequent letter in English texts. Thus, if in a given ciphertext the letter 'X' is the most frequent letter (and we have a monoalphabetic substitution cipher – we will describe this later in detail) it may be the 'E' in the plaintext.

c) Substitution Ciphers

Substitution ciphers replace letters of the plaintext with other letters (or number, symbols, etc.). If the same letter is always replaced with the same ciphertext letter, the cipher is a **monoalphabetic substitution cipher**. If the same letter is replaced with more than one letter, the cipher is a **homophonic substitution**. In both cases, we have only one plaintext and one ciphertext alphabet. If the alphabet is exchanged after encrypting a letter, i.e. we have different ciphertext alphabets, we have a **polyalphabetic substitution**.

Cipher type	Number of plaintext symbols	Number of ciphertext symbols
Monoalphabetic Substitution	26	26
Homophone Substitution	26	> 26
Polyalphabetic Substitution	26	26; but different alphabets

Examples:

1. Monoalphabetic Substitution: The Caesar Cipher

The Caesar cipher just shifts each letter in the alphabet according to a key (shift value).

Plaintext alphabet: `ABCDEFGHIJKLMNOPQRSTUVWXYZ`

Key: 1 (i.e. shift alphabet by 1)

Ciphertext alphabet: `BCDEFGHIJKLMNOPQRSTUVWXYZA`

Plaintext: `HELLOWORLD`

Ciphertext: `IFMMPXPSME`

2. Homophonic Substitution

The homophonic cipher replaces each plaintext letter using different ciphertext symbols. Here, for example, a ciphertext letter consists of two-digit numbers from 01 to 99.

Plaintext alphabet: `ABCDEFGHIJKLMNOPQRSTUVWXYZ`

Key: `A = {01 or 02 or 06}, B = {03 or 04}, C = {05}, ...`

Plaintext: `HELLOWORLDHOWAREYOU`

Ciphertext: `15,09,23,24,29,45,30,35,23,07,16,29,46,01,36,10,49,30,41`

3. Polyalphabetic Substitution: The Vigenère Cipher

The Vigenère cipher uses different shifted ciphertext alphabets based on a keyword.

Plaintext alphabet: `ABCDEFGHIJKLMNOPQRSTUVWXYZ`

Ciphertext alphabets:
```
ABCDEFGHIJKLMNOPQRSTUVWXYZ  26 different shifted alphabets
BCDEFGHIJKLMNOPQRSTUVWXYZA
CDEFGHIJKLMNOPQRSTUVWXYZAB
DEFGHIJKLMNOPQRSTUVWXYZABC
EFGHIJKLMNOPQRSTUVWXYZABCD
…
```

Key: `SECRET`

Plaintext: `HELLOWORLDHOWAREYOU`

Ciphertext: `ZINCSPGVNULHOETVCHM`

d) Transposition Ciphers

Transposition ciphers do not replace letters with other letters. Instead, the position of the letters in the plaintext is changed. Thus, plaintext and ciphertext alphabet are the same. That means, that the text frequency of a ciphertext is exactly the same as its corresponding plaintext.

Example:

Transposition Cipher: The Columnar Transposition Cipher

With the classical columnar transposition cipher the plaintext is first copied, row by row, into a rectangular grid with a fixed number of columns. Then the individual columns are permuted according to a keyword. The final ciphertext is created by reading the text from the columns.

Plaintext alphabet: `ABCDEFGHIJKLMNOPQRSTUVWXYZ`

Ciphertext alphabet: `ABCDEFGHIJKLMNOPQRSTUVWXYZ`

Key: `SECRET`

Plaintext: `HELLOWORLDHOWAREYOU`

Ciphertext: `LLRERAOHYLDEHOWUWOO`

2. Introduction to the CrypTool 2 Application

CrypTool 2 (CT2) consists of six main components:

Startcenter,

Wizard,

Workspace Manager,

Online Help,

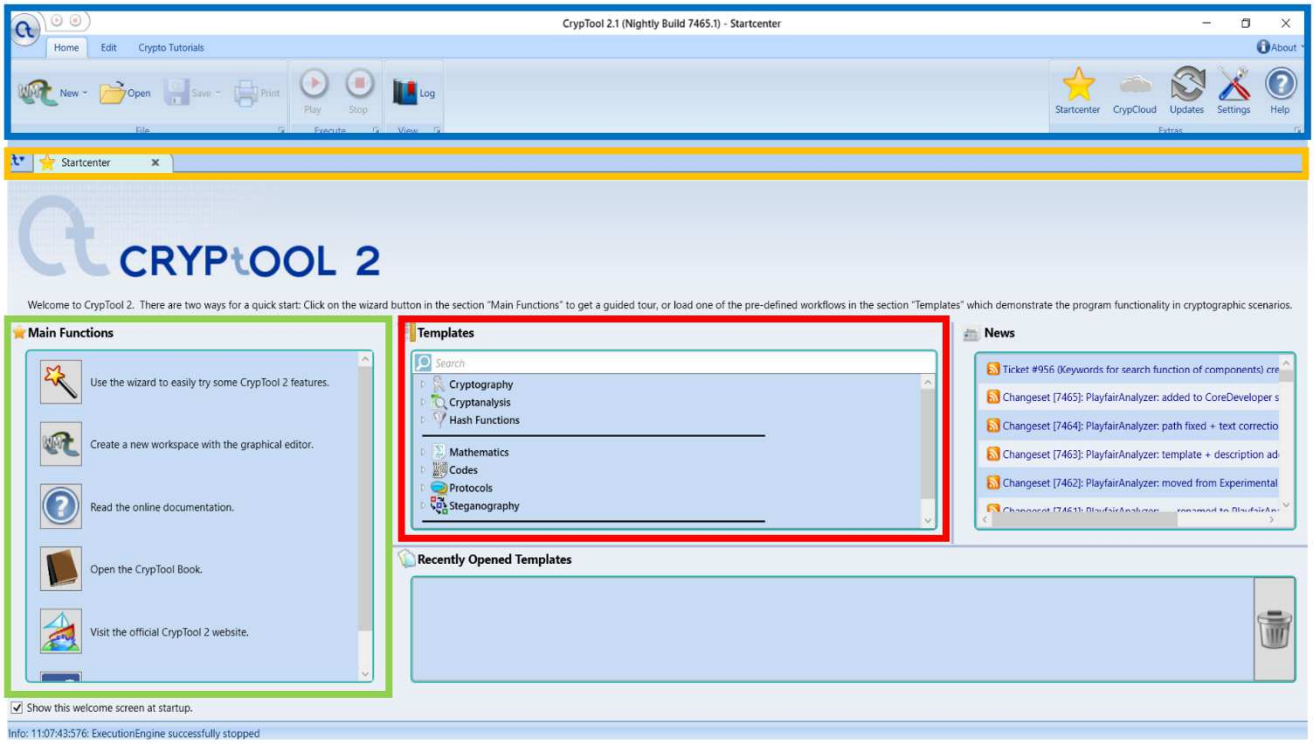
Templates,

and **CrypCloud**,

In this workshop we present the **Startcenter**, the **Wizard** and the **Workspace Manager** in detail.

a) Startcenter

Every time you start the CT2 application, you will first see the **Startcenter**.



CT2 and the Startcenter consists of different areas that we marked with different colors in the above image.

The blue marked area (“ribbon bar”) on the top of the image allows to either create new workspaces or open and save existing “CrypTool 2 workspaces” (shown later). Additionally, it allows to always go back to the Startcenter (yellow star icon), go to the CT2 settings (hammer and screwdriver icon), start the CrypCloud (cloud icon), open the online help (question mark icon) and start or stop the currently opened workspace (play and stop icons).

The yellow marked area contains a list of all open "tabs". A tab is a kind of window containing the Startcenter, workspaces, etc. Tabs can be closed, if not needed anymore using the X-icon of each tab. An arbitrary number of tabs can be opened but its amount is limited by the memory of the computer.

The green marked area of the Startcenter contains buttons to open all other components like the Wizard (magic wand), the Workspace Manager (2nd icon in the list), the online help (question mark icon), etc. Each button has a self-explaining text on its right side.

The red marked area of the Startcenter contains a list of all "templates" (more than 200) which we deliver with CT2. A template contains a specific cipher or cryptanalytic scenario using the graphical programming language of CT2 and is ready to use. The list of templates of the Startcenter can be filtered using keywords that can be entered in the search field.

Below the red marked area, you can find "Recently Opened Templates" showing a list of templates you opened in the past.

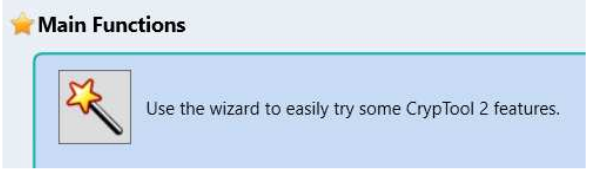
Finally, on the right side of the Startcenter you will see some "news", showing the last changes we did on CT2 with respect to its source code.

b) Wizard

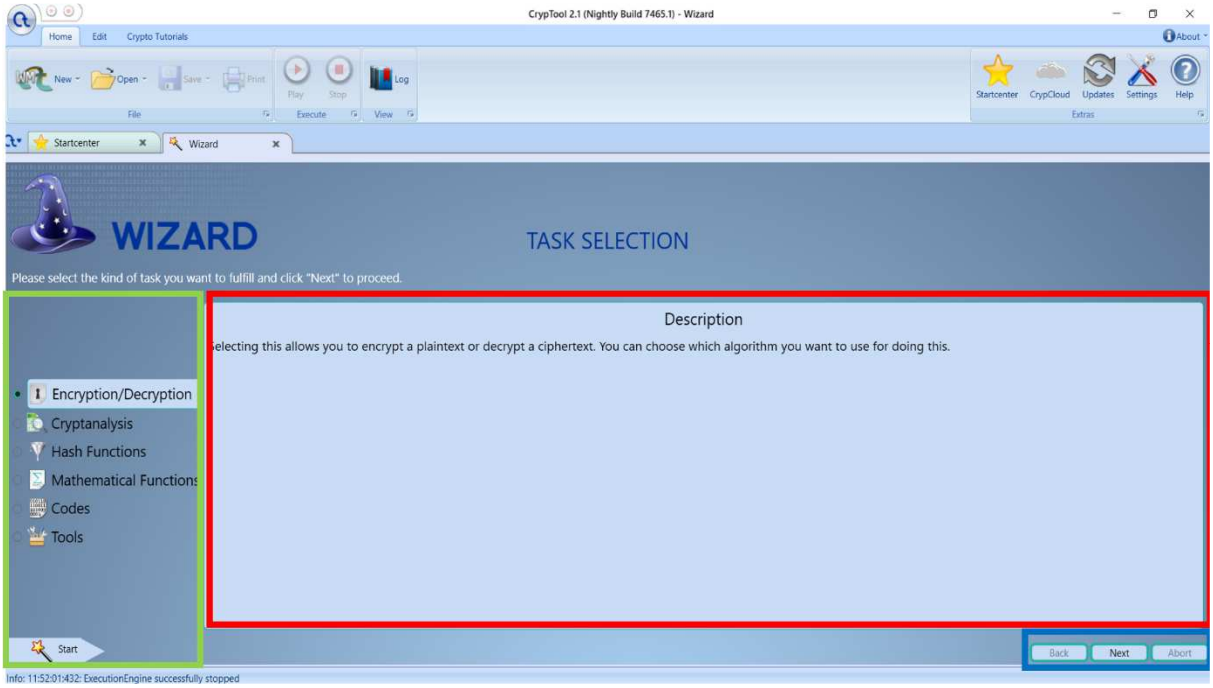
The Wizard is intended for users not familiar with using the graphical programming language of the Workspace Manager and for beginners. It guides you through the different topics of cryptology until you "reach what you want to do", e.g. encrypt something or break something.

The Wizard can be started at two different places. First, it can be started 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.

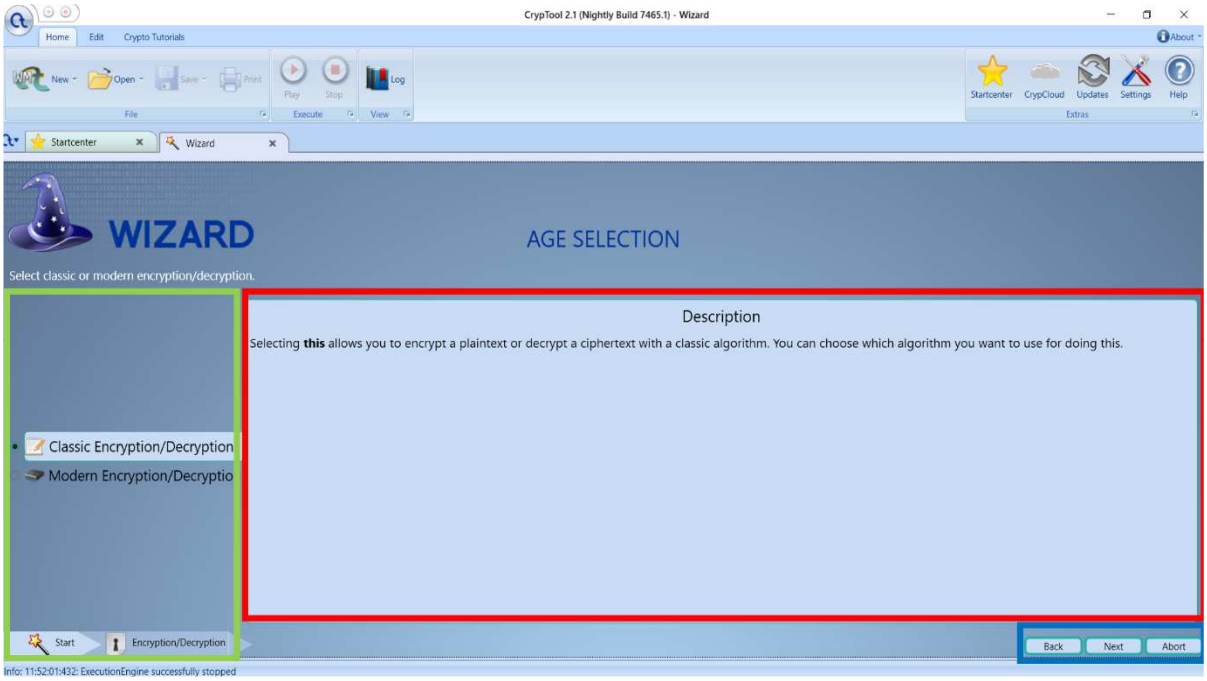


The Wizard consists of three main areas (here marked green, blue, and red).



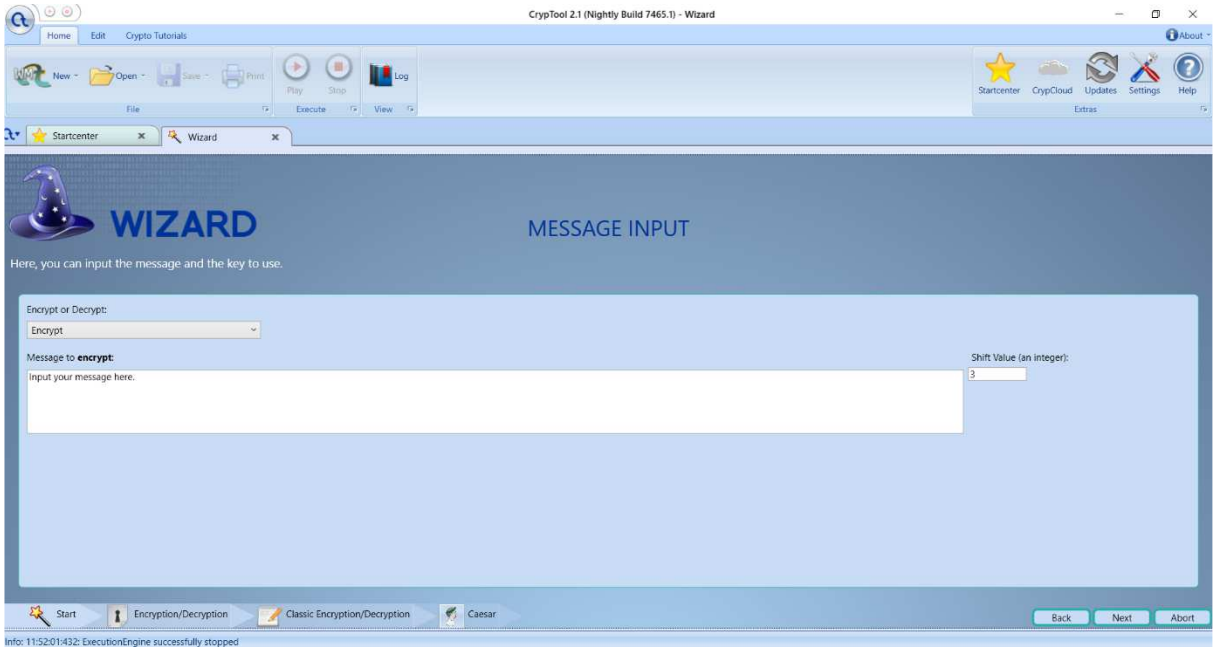
In the green marked area, you can “select what you want to do”.

For example, you want to encrypt a text using the Caesar cipher. Then, first select “Encryption/Decryption” and click on “Next” in the blue marked area. Instead of clicking on “Next” you may also double-click in the green area. Then, in the red marked area, the next page will appear.

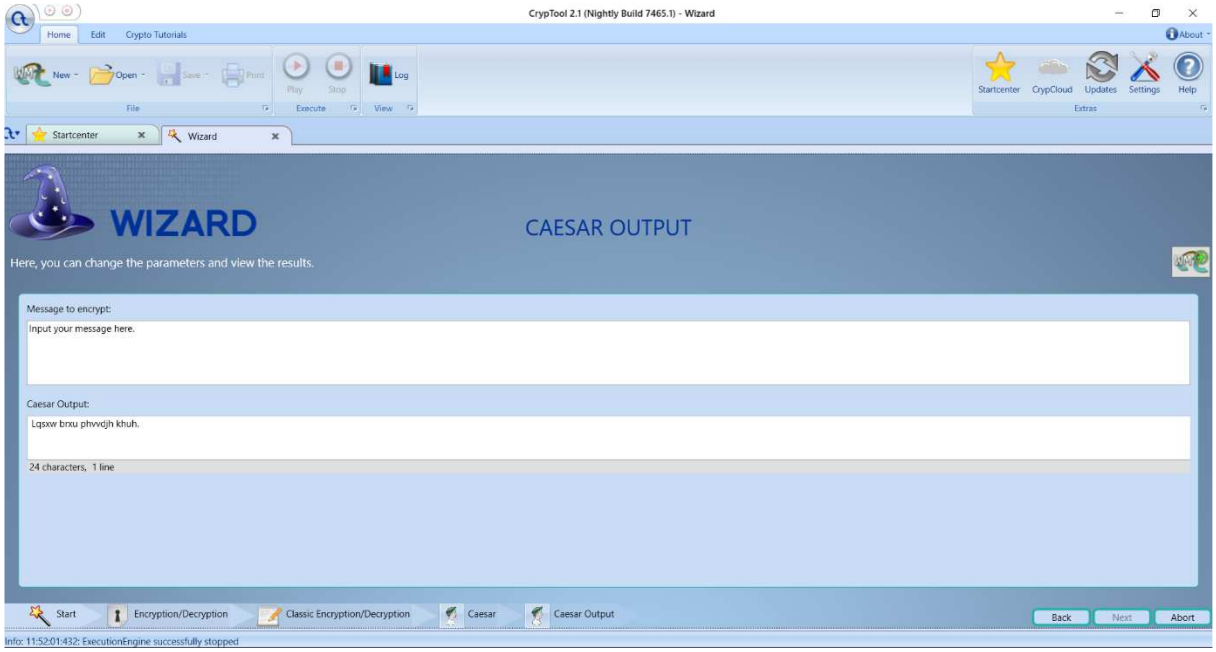


Then, the green area is updated with new options. The red area always contains some informational text based on the selections.

You repeat this step until you reach the “Caesar” cipher.



Here, you can enter the key and the text you want to encrypt. On the last time you click “Next” you will get the encrypted text.



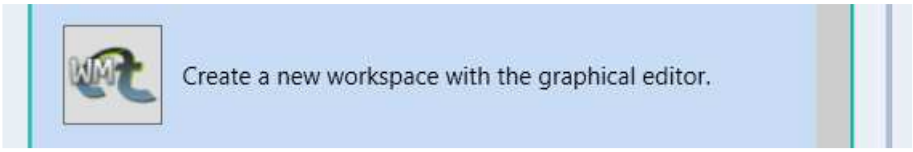
In each final step in the Wizard, you may click on the Workspace Manager icon on the top right side of the Wizard to open a template in the Workspace Manager corresponding to the cipher or cryptanalytic method you selected and currently use.

C) Workspace Manager

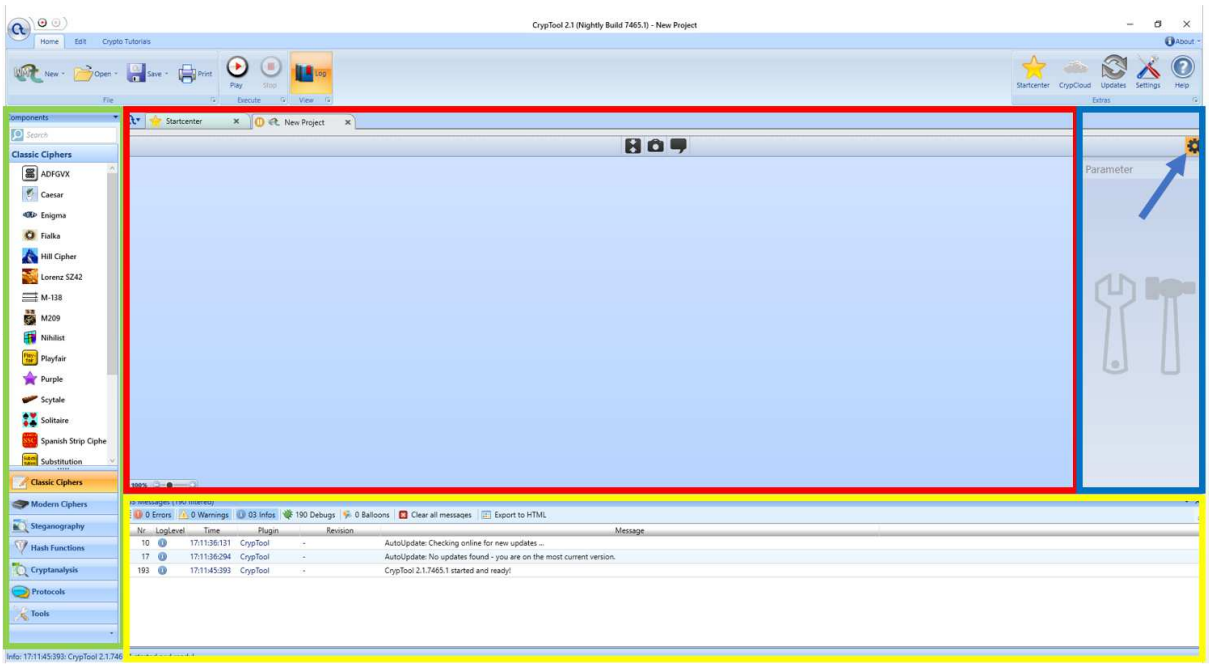
The **Workspace Manager** implements the graphical programming language of CT2. It allows to create arbitrary cascades of ciphers and cryptanalytic methods.

The Workspace Manager can be started at two different places. First, it can be started by clicking in the top ribbon bar on the new icon and selecting “Workspace”.

Secondly, it can be started using the Startcenter and clicking here on the “Workspace Manager” button.



A newly opened workspace of the Workspace Manager looks like this.



The red marked area is the actual workspace. It is used to create a visual program.

The green marked area contains the list of components (components = cryptographic methods implemented in CT2). Each component can be put onto the workspace. To do so, just drag a component from the left side onto the workspace in the middle and drop it.

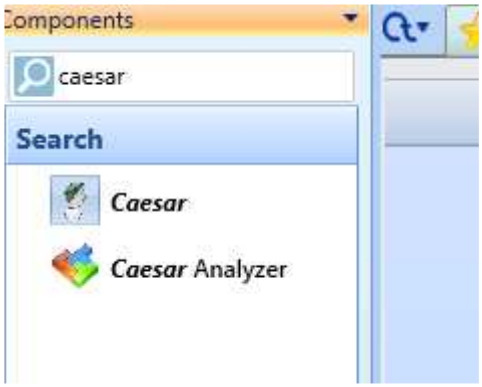
The yellow marked area is a logging window which contains messages generated by the components during the execution.

The blue marked area on the right side is the settings bar for the selected components. If a component is selected you can change its internal parameters here. The settings bar can be closed and opened with the gear-wheel button in the upper right corner (marked with a blue arrow in the picture above).

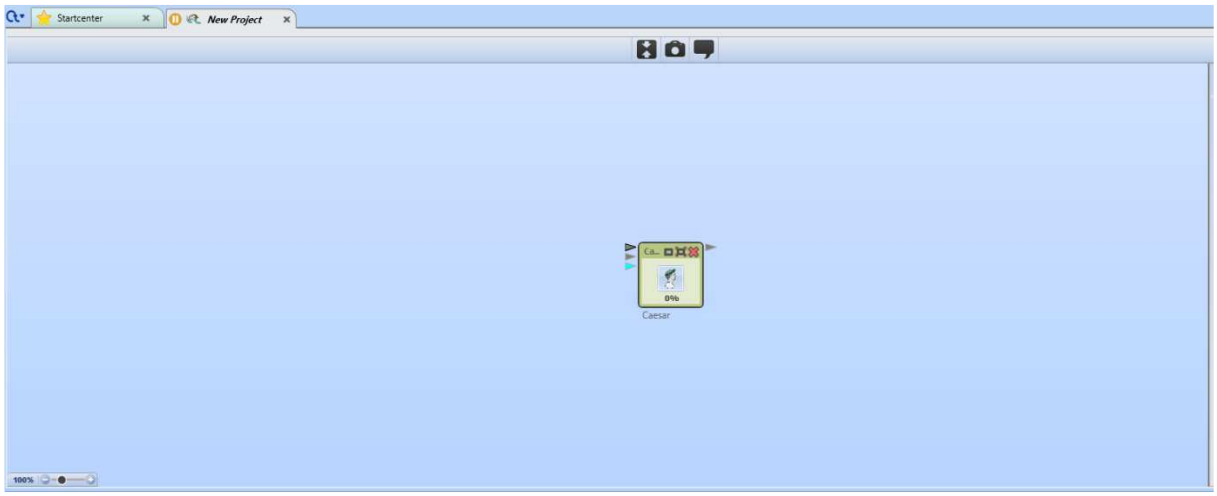
Example Build of a Caesar Cipher

Now we show you how to build a workspace for a Caesar cipher from scratch with CT2. To do so, open the Workspace Manager as shown above.

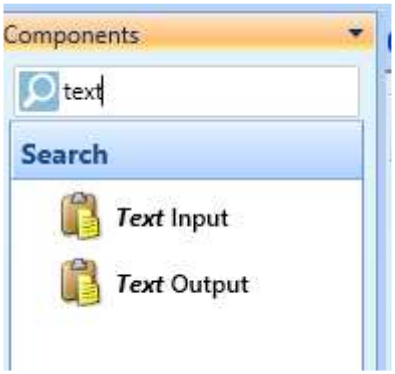
Then, go to the list of components on the left side. Here, enter “caesar” in the search field (it is not case-sensitive).



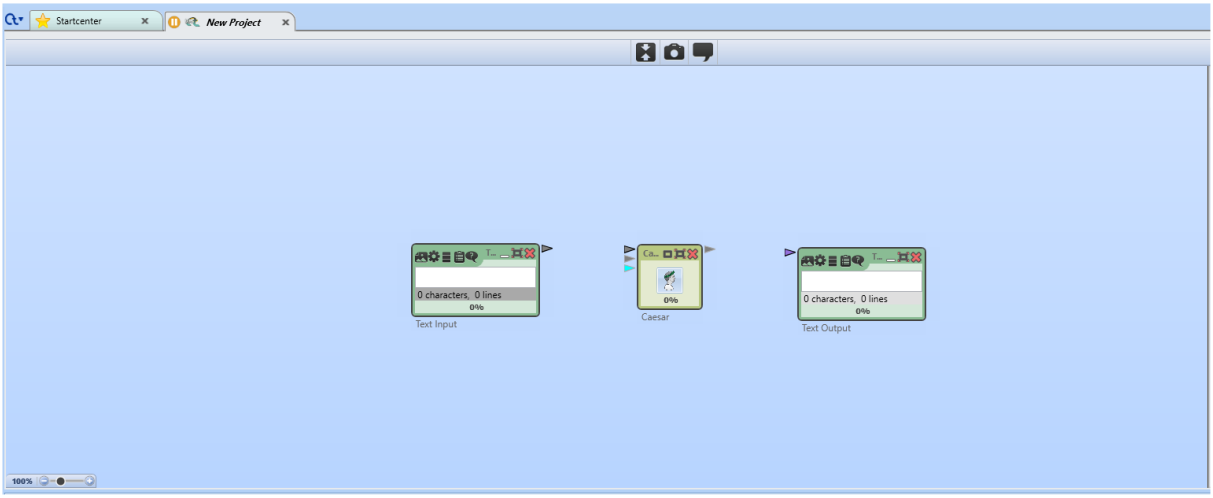
Now, use the left mouse button to drag the “Caesar” component and put it onto the middle of the workspace.



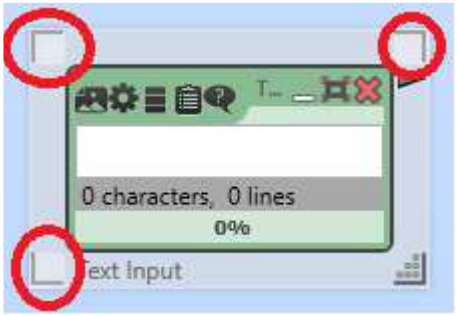
After that, use the components list again to search for “text”.



Now, drag&drop a “Text Input” component to the left of the Caesar component and a “Text Output” component to the right of the Caesar component.



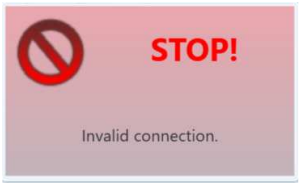
If you want to move them you can always drag a component. A minimized one can be dragged at each position within the icon (like the Caesar component in the picture). If it is not minimized but maximized, like the “Text Input” and “Text Output”, select the component by clicking on it. Then you can move the component using the upper gray corners or the lower left gray corner (marked red in the next picture).



To establish a workflow connect “Text Input” and “Text Output” with the Caesar component. For connections between components CT2 offers connectors. Connectors are small colored rectangles on the left or right side of a component. You can drag&drop a line between output and input connectors. The color of a connector shows it’s data type. For example, a number connector is blue (), a text connector is gray (), and so on. As a rule of thumb: You can always connect connectors of the same color without any problems. If you want to connect connectors with different colors, you may need

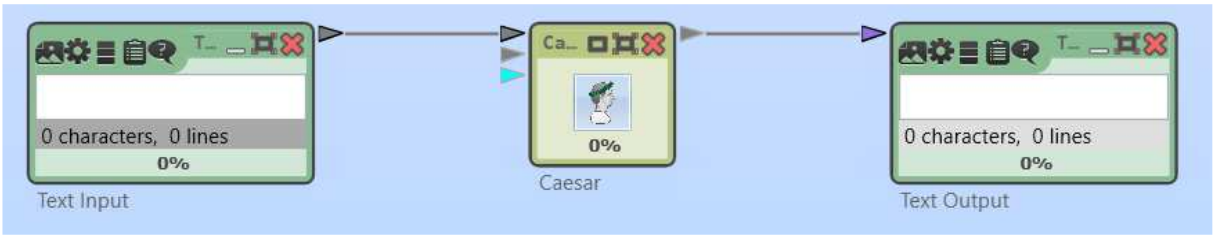
converter components. Some data types can be implicitly converted. CT2 will show a hint if this happens.

If a connection is not possible CT2 shows an error.



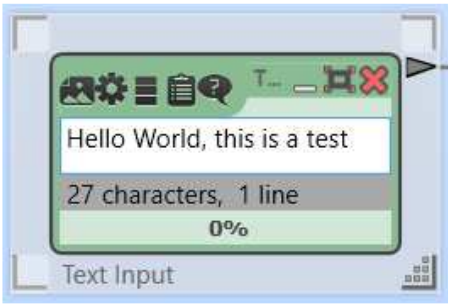
If a connection is valid without any problems CT2 shows a green text.

Now, connect the Caesar component, the “Text Input” component, and the “Text Output” component as shown in the next picture.



Now, you have built your first graphical program.

Click on the text field of the “Text Input” component and enter some text.



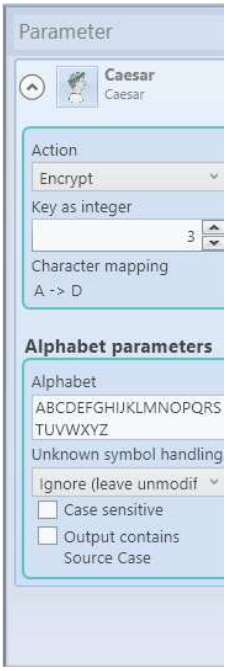
Finally, click on the “Play” button in the top ribbon bar.

Now, CT2 executes your graphical program. The output should look like this.

Try to type into the “Text Input” while the graphical program is being executed. CT2 will update your ciphertext in the “Text Output” component at once.

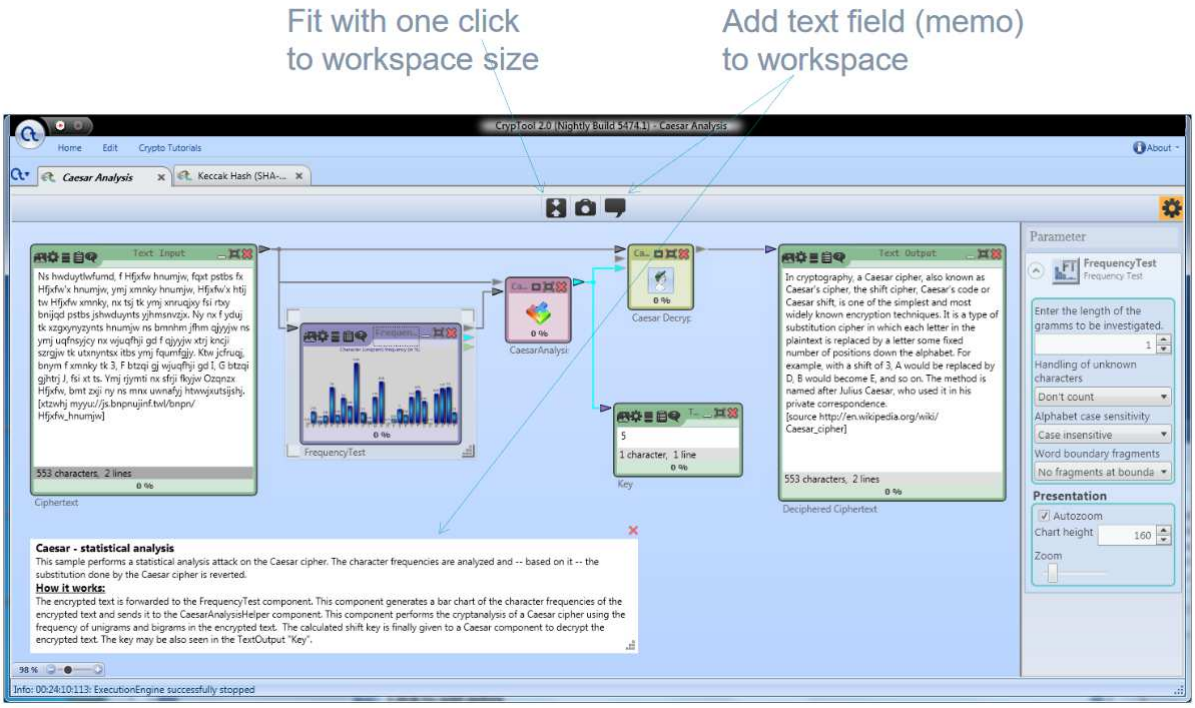
To change your graphical program, you have to stop it using the “Stop” button in the top ribbon bar.

If you want to change the key or other settings of the Caesar cipher, select it and use the toolbar on the right side of the workspace.



Here, with the Caesar component, you can change the key (shift number), the alphabet, etc.

You can adapt the zoom level of the workspace using the buttons in the top middle of the Workspace Manager.



Further hint for easy handling: Quickly adapt the CT2 GUI with the keyboard using F11 and F12 by fading-in or fading-out parts outside the actual workspace.

Each workspace can be stored as a file with the extension “cwm” (via the “Save” icon under the “Home” menu at the top of the CT2 main windows). All templates are also workspaces – predefined and delivered with CT2. So they are also stored in cwm files (see the directory “Templates” below the CT2 directory in your installation). Their specialty is that they are available in 2 languages at once.

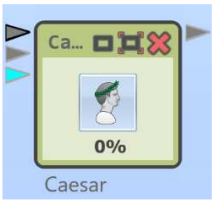
3. Substitution Ciphers

CrypTool 2 (CT2) contains different classic substitution ciphers. We will have a closer look at the following ones:

- **Caesar** cipher
- **Monoalphabetic substitution** cipher
- **Vigenère** cipher

To use the ciphers and their corresponding analysis methods, go to the Startcenter and use the template list to search for appropriate templates. You could also use the Wizard. To copy a text, mark it using the mouse and press "control key + C". Then, in CT2 you can enter the text by pasting it (pressing "control key + V") into the text input component.

a) Caesar Cipher



Task 1: Decrypt the following ciphertext using the Caesar cipher built in CT2:

```
Va fvkgl OP, Pnrfne fbhtug ryrpgvba nf pbafhy sbe svsglavar OP, nybat jvgu gjb bgure pnaqvqngrf. Gur ryrpgvba jnf fbeqvq – rira Pngb, jvgu uvf erchgngvba sbe vapbeehcgvovyvgl, vf fnvq gb unir erfbegrq gb oevorel va snibhe bs bar bs Pnrfne'f bccbaragf. Pnrfne jba, nybat jvgu pbafreingvir Znephf Ovohyhf
```

Key: 13

Hint: Open the template "Caesar Cipher" or use the Wizard.

Task 2: Encrypt the following plaintext using the Caesar cipher built in CT2:

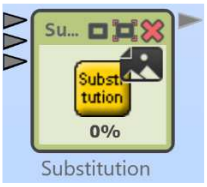
```
Gaius Julius Caesar was born into a patrician family, the gens Julia, which claimed descent from Iulus, son of the legendary Trojan prince Aeneas, supposedly the son of the goddess Venus.
```

Key: 10

Task 3: Break the following ciphertext using the template "Caesar Analysis using character frequencies":

```
Gur nffnffvangvba bs Whyvhf Pnrfne jnf gur erfhyg bs n pbafcvenpl ol znal Ebzna frangbef yrq ol Tnvhf Pnffvhf Ybatvahf, Qrpvzhf Whavhf Oehghf Nyovahf, naq Znephf Whavhf Oehghf. Gurl fgnoorq Pnrfne (gjraglguerr gvzrf) gb qrngu va n ybpngvba nqwnprag gb gur Gurnger bs Cbzcrl ba gur Vqrf bs Znepu svsgrra Znepu sbheglsbhe OP.
```

b) Monoalphabetic Substitution Cipher



Task 4: Decrypt the following ciphertext using the template "Substitution Cipher using a password":

```
rN YJBLGMUJaLTB, a YSLTWJ (MJ YBLTWJ) SI aN aPUMJSGTO VMJ LWJVMJOSNU
WNYJBLGSMN MJ XWYJBLGSMN—a IWJSWI MV DWPP-XWVSNWX IGWLI GTaG YaN ZW
VMPPMDWX aI a LJMYWXFJW. zN aPGWJNaGSEW, PWII YMOOMN GWJO SI
WNYSLTWJOWNG. kM WNYSLTWJ MJ WNYMXW SI GM YMNEWJG SNVMJOaGSMN SNGM
YSLTWJ MJ YMXW. rN YMOOMN LaJPaNYW, "YSLTWJ" SI IBNMNBOMFI DSGT "YMXW",
aI GTWB aJW ZMGT a IWG MV IGWLI GTaG WNYJBLG a OWIIaUW; TMDWEWJ, GTW
YMNYWLGI aJW XSIGSNYG SN YJBLGMUJaLTB, WILWYSaPPB YPaIISYaP
YJBLGMUJaLTB.
```

Key: password = Hidden, offset = 10

Hint: You have to change the setting "Action" of the "Encrypt" substitution component from "Encrypt" to "Decrypt". We change the given template, as this template was written to do encryption first. So just ignore the component with the subtitle Decrypt.

Task 5: Encrypt the following plaintext using the template "Substitution Cipher using a password":

```
Codes generally substitute different length strings of characters in
the output, while ciphers generally substitute the same number of
characters as are input.
```

Key: password = secret, offset = 8

Task 6: Break the following ciphertext using the template "Monoalphabetic Substitution Analyzer":

```
JRU GOLF "XWNRUL" WP BOLQUL JWQUK QUZPJ "CULO" ZPF RZF JRU KZQU
OLWAWP: QWFFSU BLUPXR ZK XWBLU ZPF QUFWUHZS SZJWP ZK XWBLZ, BLOQ JRU
ZLZYX KWBL = CULO (KUU CULO - UJDQOSOAD). "XWNRUL" GZK SZJUL IKUF
BOL ZPD FUXWQZS FWAWJ, UHUP ZPD PIQYUL. JRULU ZLU QZPD JRUOLWUK
ZYOIJ ROG JRU GOLF "XWNRUL" QZD RZHU XOQU JO QUZP "UPXOFWPA".
```

c) Polyalphabetic Cipher - Vigenère Cipher

Task 7: Decrypt the following ciphertext using the "Vigenère Cipher" template:

```
OPK QRXYSY WL IAGICKBOSA OESRV GW GLV ZDOKRRVV GDXNIE ARW HQYEGXIMWCZIQ
XF FGIOWR HV ZDOKRRVV MI BNI AMEIOMKRGL TIIBAVL EEH RIY MA JRGO NOVFX
UINKXMOIU FT OOSIEE FVBZMFXR FZTREFS ZR CQY FBSB PV KOJEE UIG AOKASII
BQUZNR SEOBOWGE SIGTGWB
```

Key: VIGENERE

Hint: You have to change the setting "Action" of the upper Vigenère component from "Encrypt" to "Decrypt".

Task 8: Encrypt the following plaintext using the "Vigenère Cipher" template:

```
VIGENERE CREATED A DIFFERENT, STRONGER AUTOKEY CIPHER IN FIFTEEN
EIGHTY SIX
```

Key: BELLASO

Task 9: Break the following ciphertext using the "Vigenère Analysis" template:

```
TSF ECUMTBX JMGHPS FP EMQV QHXKIDUE REJGZIP EIOFOH WHCTBTLR JIIUC
JXDGV LHW RN PBVCR XQTNHPGHLCIKBK EQEOII. AWCIIMQ WATK E DIIFH
REXJIQLX KO POGIRXV I BLWJARF.
```

4. Homophonic Substitution Ciphers

CrypTool 2 (CT2) contains different templates to create homophonic substitution ciphers:

> **"Homophone Substitution Cipher and Nomenclature – Encryption" and "Homophone Substitution Cipher and Nomenclature – Decryption"**

CT2 also contains a template for the cryptanalysis of homophonic substitution ciphers:

> **"Zodiac-408 Analysis" AND "Homophonic Substitution Analysis"**

Task 10: Decrypt the following ciphertext using the template "Homophone Substitution Cipher and Nomenclature – Decryption":

```
05 35 99 21 06 47 23 25 88 05 51 52 22 33 43 51 99 52 37 88 01 36 17
48 21 06 49 22 99 51 03 21 88 27 02 19 20 01 18 41 24 52 26 99 38 19
88 20 47 22 45 42 21 35 17 25 99 05 36 06 23 26 50 02 49 88 05 51 52
06 18 15 50 99 37 35 88 49 41 11 50 51 01 52 42 51 02 38 36 99 17 01
44 04 22 48 49 88 31 05 50 99 52 37 88 28 02 49 07 41 01 50 21 99 43
24 06 02 35 51 22 29 52 88 23 21 51 52 22 47 99 19 48 21 46 42 22 36
18 01 21 49 88 12 25 99 03 38 34 37 44 04 38 35 26 77
```

Key:

```
[ ];[99|88]
[.];[77]
[I];[01|02]
[H];[03|04]
[A];[05|06]
[G];[07|08]
[J];[09|10]
[B];[11|12]
[Z];[13|14]
[K];[15|16]
[C];[17|18]
[F];[19|20]
[E];[21|22]
[L];[23|24]
[Y];[25|26]
[D];[27|28]
[X];[29|30]
[W];[31|32]
[M];[33|34]
[N];[35|36]
[O];[37|38]
[V];[39|40]
[U];[41|42]
[P];[43|44]
[Q];[45|46]
[R];[47|48]
[S];[49|50]
[T];[51|52]
```

Hint: Copy the key into the "Nomenclature" TextInput of the template. Since we only substituted single letters, actually, this is no nomenclature. But it also works since a homophonic cipher can be seen as a subset of a nomenclature.

Task 11: Encrypt the following plaintext using the template "Homophone Substitution Cipher and Nomenclature – Encryption":

```
THE BEALE CIPHERS ARE ANOTHER EXAMPLE OF A HOMOPHONIC CIPHER. THIS IS
A STORY OF BURIED TREASURE THAT WAS DESCRIBED BY USE OF A CIPHERED
TEXT THAT WAS KEYED TO THE DECLARATION OF INDEPENDENCE.
```

Key:

```
[ ];[99|88]
[.];[77]
[E];[01|02]
[D];[03|04]
[F];[05|06]
[C];[07|08]
[G];[09|10]
[B];[11|12]
[H];[13|14]
[I];[15|16]
[A];[17|18]
[X];[19|20]
[Y];[21|22]
[J];[23|24]
[Z];[25|26]
[K];[27|28]
[V];[29|30]
[W];[31|32]
[L];[33|34]
[T];[35|36]
[U];[37|38]
[N];[39|40]
[M];[41|42]
[R];[43|44]
[S];[45|46]
[Q];[47|48]
[P];[49|50]
[O];[51|52]
```

Hint: Copy the key into the "Nomenclature" TextInput of the template. Since we only substituted single letters, actually, this is no nomenclature. But it also works since a homophonic cipher can be seen as a subset of a nomenclature.

Task 12: Break the following ciphertext using the template “Homophonic Substitution Analysis”:

```
39 03 51 11 49 52 29 30 04 15 29 51 99 45 43 09 10 52 88 33 35 13 30
51 99 52 09 16 36 88 07 37 35 19 38 99 51 15 88 40 03 36 27 52 37 38
04 99 50 51 29 30 03 16 29 52 88 46 44 10 09 51 99 34 35 14 30 52 88
20 51 15 99 52 37 88 04 29 51 10 03 52 38 99 16 47 01 36 09 51 13 88
33 35 10 21 12 52 30 02 99 51 37 45 88 34 09 52 22 19 14 04 39 01 29
99 20 02 36 88 10 03 28 43 46 99 04 38 88 37 51 33 09 44 15 99 52 30
88 29 01 43 99 30 03 11 44 88 35 41 99 29 02 43 88 16 48 04 44 38 30
03 42 04 47 99 13 43 27 36 10 17 29 03 35 37 88 51 38 45 99 14 44 41
36 13 12 52 30 04 35 37 77 88 99 40 03 51 11 49 52 29 30 04 15 29 51
88 46 43 09 10 52 99 34 36 14 30 51 88 16 33 44 38 29 99 30 01 43 88
12 52 05 35 13 03 29 21 99 36 42 88 02 04 15 99 09 03 41 44 88 35 37
99 16 48 04 43 38 30 03 42 04 47 88 44 37 45 43 51 28 36 14 15 77 99
01 44 88 50 43 38 44 41 03 29 43 46 99 42 13 35 11 88 52 37 99 04 38
41 36 14 12 51 10 88 44 45 18 48 52 30 03 35 37 99 36 42 88 29 17 30
35 13 16 99 51 38 46 88 27 04 15 03 29 16 99 41 14 36 11 88 13 43 37
35 19 38 44 45 99 15 47 02 36 09 52 14 16 77 88 01 04 15 99 12 35 16
30 88 42 51 11 36 18 15 99 20 35 13 08 88 03 16 99 43 37 29 04 30 10
44 46 88 12 52 39 03 51 43 99 38 52 29 17 14 51 09 04 15 77
```

Hint: First copy the ciphertext into the “Ciphertext” TextInput. Then, start the workspace. To actually start the analysis process, click on the “Analyze”-button of the “Homophonic Substitution Analyzer”.

In the analyzer’s semi-automatic mode, the 3 buttons "Analyze/Stop", "Reset locked letters", and "Find/Lock words" within the analyzer component are enabled. To start the analysis process click on the "Analyze" button. After some time, the auto-locker will automatically lock already revealed words. You can stop ("Stop") and restart ("Analyze") the analysis process at any time. When the analysis is stopped, you can also lock and unlock single letters of the revealed plaintext by yourself with a left mouse button click. After locking a letter with the left mouse button, you can change letters using the right mouse button: Each click on the right mouse button shows the next valid letter; clicking the right mouse button plus Shift selects the previous letter in the plaintext alphabet.

Task 13: Break the following ciphertext using the template "Homophonic Substitution Analysis":

```
§ a ä / @ o + $ ) u Ä e q s m @ p - ö f + w g e = Ä n b § f ) - ( h c
d g e a 1 + o # h m r @ f Ä = - q g + $ b ä / @ - ö e + x h f ) Ä n a
§ e = - ä p + $ f - + e g v @ b - w m ö r q Ä f + % 0 - @ e ^ a ä o '
+ § t r # h n - ( ' $ m b Ä p + a s q x ö ) ° @ - = g ) ^ o h f + t e
= Ä n - r # @ + i p Ä s ) g f 1 c - b @ w ä o + / § m n h a b : - ö q
+ r Ä a b p - g Ö + $ - 0 h t e ° + ^ ä m a - f § d @ = + $ b ö ( Ä -
Ö § a b ä e ° + q ' n g s ^ # - $ + m § & % ö r - ' h a @ + ä f q g -
$ + Ö § e r $ o 1 - x h n b ) + j g i t a § q Ä = - & 0 + j @ / s b ö
$ m - § f r # n h i g c h m j ' ä ( + / n Ä $ q t m @ p
```

Hint: First copy the ciphertext into the "Ciphertext" TextInput. Then, start the workspace. To actually start the analysis process, click on the "Analyze"-button of the "Homophonic Substitution Analyzer". You may have to change the "Block" name component to blocksize = 1.

Task 14: Break the following ciphertext using the template "Homophonic Substitution Analysis":

```
19 20 03 21 31 11 53 41 14 24 02 13 25 51 15 39 06 05 30 59 56 03 54
31 35 29 34 14 10 48 57 09 27 06 39 53 16 04 56 05 20 25 35 01 18 36
57 07 55 59 03 26 05 41 14 24 31 33 02 04 35 54 29 39 45 20 47 21 58
59 42 27 47 38 12 55 15 02 16 47 53 48 14 56 28 49 34 57 36 03 19 13
39 16 05 26 35 41 25 27 54 28 59 55 22 31 20 14 45 39 57 52 02 21 36
05 41 27 15 51 32 07 53 48 15 30 13 21 55 59 14 56 01 18 39 59 58 03
54 14 47 02 47 24 31 16 12 27 36 48 35 20 29 34 55 42 24 02 26 28 53
21 57 33 56 47 27 39 03 36 10 55 16 47 02 41 25 31 26 06 19 16 01 27
47 38 15 21 53 59 14 56 16 48 39 45 04 03 15 42 51 29 25 36 31 28 58
13 59 19 30 32 34 55 47 14 21 53 10 01 02 16 38 49 27 48 42 05 15 36
56 26 35 54 39 45 59 24 57 20 44 05 41 29 34 03 47 21 31 14 39 53 16
48 12 56 47 03 15 31 07 51 16 53 49 24 35 36 56 57 54 47 03 55 21 05
59 10 14 34 31 38 12 53 16 56 28 03 32 35 18 24 31 47 53 26 02 20 28
16 56 27 26 33 10 49 55 36 48 57 41 25 34 02 01 06 47 58 42 39 24 03
15 31 21 53 59 14 56 16 48 05 39 12 27 15 07 36 03 55 47 59 34 02 14
01 27 16 38 24 55 28 48 02 54 32 39 35 19 20 31 26 59 34 53 27 14 39
56 42 47 59 03 28 55 15 48 02 15 48 57 41 27 14 05 45 54 58 30 31 21
35 08 55 04 53 39 24
```

Hint: First copy the ciphertext into the "Ciphertext" TextInput. Then, start the workspace. To actually start the analysis process, click on the "Analyze"-button of the "Homophonic Substitution Analyzer". The ciphertext contains no substituted spaces, thus, uncheck "Use spaces" in the analyzer's settings.

Task 15: Use the ciphertexts from the previous tasks and copy them one by one into the template "Statistic Tests for Classical Ciphers" ("Plaintext" text input component). Compare the results of the "Friedman Test" component with the different types of ciphers. Hint: You have to delete the Vigenère component from the workspace and afterwards connect the "Plaintext" text input component with the "Frequency Test" component. Otherwise, all texts would be encrypted using the Vigenère cipher every time you insert a new text.

8. Challenge Part

Here, we have some tasks with ciphers of "unknown" type. Happy breaking!

Task 16: Analyze the type of the following ciphertext and break it!

```
iw no lh lj is lj nr no no iw nm iu ic ld ib nk lk nv ip is iq lz ih
if is iw ig iw lr lr ib ll ih nx ns nu ih ia lh ng lh if in ib lq ld
nu ih ia ls iv nz lm lj lh lk lk nr ia lf ij lh no nw lf io nn is iw
nm ig ns nv it ln lq ie ld lr ig ip nv nx lz ih if ld ll io ia iw if
ng lg is ll ib nz ls nw lz lm lf nv ie ng lt is lz lm nz ll io no ib
it lz no lk ng ib np nr no lk nh nl in ld ng ns iw nm iu lf lh ne ld
if nn ld ig ns is in ib lz ls ig lg ni nr no iz lh ia lf ld nc nk nv
ie is lm iq is lh ls nr if is lu is ia ip nv ng ig is ie ng ns lz nm
lf ld ng ls lh ia nt il ib ih ni ni nl iq lj lr ib le le lv iw ng lg
io lf lh ni no ng lg ld la is nh ls ic io ni ng ln it lh ng nr io ig
lg lz ld ij lg is ia lh lc iw nv iw lv lh no no ip ld ie nv ip ln ie
nm iw ia ic nz ie lz nw iw lb is lr nm nw io no iz ig lg is nr iv lz
ii is iy iw no iz is ir lv nr no lk ny is lb ib in ld in lx if iz nz
ne ld lr iw nd iw lk no ia ln ig nt nr ii nv il ln lt in il ia lz in
is la ld nx nz lt if is nb ln nf ij iw iz lk ig ni lx ls ln lr iz nl
iw lc ib ij lm ln ni lr ig nl ic in il iq nl iz iz nv nx ig nr nm lf
ib nu lr iz nz lu is nh it nl ie nn il lz nu ls ld ni no iw it ld nv
ip nv nl ni iw ld ls nv nn nv ls lg iv lo iw ng nr
```

Task 17: Analyze the type of the following ciphertext and break it!

```
yitkt1qkt1q1fxdwtk1gz1rozztktfy1ynhtl1gz1lxwlyoyxyogf1eohitkl1oz1yit
1eohitk1ghtkqytl1gf1lofust1styytkl1oy1ol1ytkdtr1q1lodhst1lxwlyoyxyog
f1eohitk1q1eohitk1yiqy1ghtkqytl1gf1sqkutk1ukgxhl1gz1styytkl1ol1ytkdt
r1hgsnukqhioe
```

Task 18: Analyze the type of the following ciphertext and break it!

```
JxQ8ubMS7bqaJSrelaIRJbE56bWctQ8bqaJHluayDav8EcJbVkbJxuaxUK7QbqXtbJxl
aCTHsxaiTHlbpcmbYxlbxSJYlHb3Q8QbiS4ZDjb6lpaSJayJaTbmU8dUKIQbMpIb7yYJ
ZDjbrRYMQRDbJxlCavq7Jbq7eulFc
```

Task 19: Analyze the type of the following ciphertext and break it!

```
526555515015509520522555185512508055507509522055555195205155512523555
551352252618555550751905555552214520550901551352405555515552155550700
855225550755261850450551455552055195225555095125080551955555201851555
504518513075555512145555095507555235522509055555504085518205522555552
550650755520551905518552255523505550952255550708518505552255552001555
095230555135221855085555262055550955075550252151955181255025555110150
951452055185130755555075190555085225555090550455055551955065230455221
214551525555520508505509518555552250205550855503551952651352452252355
555071555550650151255155555215165512145555526125518035055
```

(Hint: maybe use 3 digits)

Task 20: Analyze the type of the following ciphertext and break it!

```
ap ao bb au ai aj at cc xg xs af dd ax hx xr hj xv bb as ab ac xy hy
au cc hr am hu xd dd xg ai xi af xv bb ac xo xz xh hj hi cc xl xm dd
au xs hu bb xg hh av an aq af hj cc au hx xv hd dd xf ao as he hb am
hu ae bb xg ai af cc xk hq xi ad xs xn xv xm hj at xx hh ap xo hb dd
ab hd xw bb as hu xz ht cc hq xh dd ag xl am xo he hm hi bb au hx af
cc ar hk xv hu ao dd ap xu bb ai af ab xi xg at cc xh xs xv dd hc xz
ae hu bb hi xl an af cc hj hq hh au at dd ab hb am bb he xm cc xz dd
xh av xn hc xv as bb xw hq az cc xg hx hu dd al hd ab aw af bb ap hv
cc ai xv xz xi hj hi dd xs hu bb at au xl xo af cc xg hx he xh xv dd
hj hq hh au hi bb ab ao ht cc xg ap xl xp dd hj ai hu an bb xj xf aj
au af cc xz ax hq xb dd hs ab hb am bb xg xs xv cc ag xr as at hj dd
xd hy au xm hu xh hi bb at xz aj ae cc xg hx af dd ha xr hd ah bb hq
ao xw cc hj ai xv dd hm xs hy au hu bb xi ab xy hr aj xg cc ac xo af
ax dd hj hx hh xv hu bb xy hb xz xh au hi cc he xm dd xg ai af bb hj
as hk xn hf xv au cc hq hd ht dd ad ab am xo hu ae bb ap av xg cc xu
xr xi at hj dd xd hy au ao af xh hi
```

Task 21: Analyze the type of the following ciphertext and break it!

```
DBCEFXFXVTRCVRMMNQHPRLFANBNEGTFWNQHPRXYITCYIQXYIZIXXUWDEBVKXVGWMZTJB
GLVMINUVZEKMJVCEFKRXCMXIQEEHOZGRFQZXOMJYAHVVBZAYAHKSVVARXRFAITUGNXZS
IFJSRWCEIORILOESRIHSHXKLDAZLRCJLJCRHVXJFPZOIQSLXOPKVRWFQZENIEIOACWQR
BAJXCMKBNGKPJKGXVSESITEAJXYMNEGWUMJPVAZQRWJEBMDXUMIXTMOKUXUIBZKIFJZJ
OGYIIIEQDVAXRWJMSXUMAXWMQMYIPSEHNVUVGLJIQMTXLWVZZVJITVVINMOKUXDMICZI
FJFVOGLSHVJIXWTHFAVWOQJFLVFAN
```

(Hint: only WW knows)

Task 22: Analyze the type of the following ciphertext and break it!

```
SEANWIEUIIUZHDTGCNPLBHXGKOZBJQBFEQTXZBWJJOYTKFHRTPZWKPVURYSQVOUPZXGG
OEPHCKUASFKIPWPLVOJIZHMNNVAEUDXYFDURJBOVPASXMLVFYYRDELVPLMFYSINXYFQE
ONPKMOBPCFYXJFHOHTASETOVBOCAJDSVQUMZTZVTPHYDAUFQTIUTTJJDOGOAIAFLWHTX
TIQLTRSEALVLFLXFO
```

(Hint: maybe period 15)

Task 23: Analyze the type of the following ciphertext and break it!

```
47 17 03 21 13 23 24 29 35 15 51 25 23 22 43 26 16 24 11 18 48 44 21
17 23 33 43 47 14 13 37 44 27 52 36 25 18 43 15 12 30 14 35 16 13 38
44 41 17 43 44 43 29 44 51 18 26 36 25 39 15 24 42 14 07 52 17 43 48
18 04 22 13 23 24 30 35 29 30 11 16 14 17 51 13 36 26 40 47 18 03 21
14 23 39 17 52 22 37 35 48 51 25 45 47 18 04 21 13 52 26 51 29 03 14
36 48 47 24 25 48 44 22 13 15 52 47 48 23 17 46 35 26 40 14 24 38 51
31 36 25 26 35 25 43 47 18 04 21 13 23 24 30 36 16 08 23 17 29 45 07
35 46 44 14 37 43 28 24 15 13 08 36 45 44 16 22 18 43 52 46 44 30 43
51 17 26 35 25 39 15 23 41 14 07 52 18 44 36 26 13 38 43 42 35 44 29
45 24 41 48 17 03 21 14 23 30 24 40 04 47 18 03 22 13 23 24 29 36 27
21 30 43 28 44 25 14 16 27 23 17 43 13 37 51 26 42 24 12 18 38 11 25
46 17 44 14 52 29 39 23 18 35 13 37 28 15
```

Task 24: Analyze the type of the following ciphertext and break it!

```
37141100126520152803006604075019265899623900042500126508345267372032
25005511426207118805124112633167120500042555997439585599622500711358
00580433217899716800666205003876582637621237139907126538443451887132
88293332371156370056682766583356195463000519302231285437200700042506
88272021207103347888073166274426190704712032659920388848043988122729
21685258060011753811267262735821789901510025045062991612696603267888
05446919651699486869220699765434002020991965005421228802335465565012
40003157003814120060113427546588282021193703335100266215280399140440
00042500126312563834326612565004252008542188333171686988661107505426
20402700371454388872563354660163113900381312883748582537784020759963
11383858337200685700381458990363675054011237
```

Task 25: Analyze the type of the following ciphertext and break it!

```
bxdbbefeppcfwwhcdhddbydhbefxcxbyeedyfddcddgfbxdbbdaecgcgfawwcafffxqq
acfxgybgexyyeaadbfttdeggadgffdrrqqbcafcaahdhgccffxaefdacdbdxfegfffae
cydbcbhxggadhbbdehfagxexadhcaeahdhbyecwwcacgdxfawwbfbbgychyyqqddbbpp
aegceahcaecfaxbbyyhcaecbbbhfbyahdcbggyyyghfafxdefgbegyzzcfcafgccbcde
axdefccydxdcbgdeaeghbdwwgyadqqffgxahahedgddxcahcdhfabcbeadagfgzzppea
caagfxddyyacccfxcappaycbeerrcfahbbgxcgdhberrafrrebrrcbzzdaggrrgfdhdc
bafxaedecchxgggxafwwgycfadcgagggdcafttgycfgxaxfgedcbdhcahccxfagchcdy
dheeffttcgdxdybyppahgdedagwwafbettcaqqdecafgfafgdcddgffxcccahedyadga
ddgccyghadyyppffghaegyadaxfxcbdyedeahbbewwabhxcfaechbbhbdebydhfawwcx
caeeeagybcfhrrdedydxzzbdcadycfaedxbyebrrhbdbehcfdcfhhbgycadxdhwwcgfa
rrgcggdbcghbfattchdbdybcccgdttddgchdrrddfedxdhbefxdhbyggwwdafegydhgx
aydhbebcbbccwwaxhbdbgyaddxwwdcbahbddbybbdd
```

9. Links and References / Literature

You can directly download CrypTool 2 (CT2) from here:
(For this course, please use the current "Nightly Build" of CT2.)

https://www.cryptool.org/en/ct2-downloads

If you are further interested in CT2 or the CrypTool project, have a look at these pages:

CrypTool-Project / CrypTool-Portal: https://www.cryptool.org/

CrypTool-Wiki: https://www.cryptool.org/trac/CrypTool2/

If you want to read more about cryptology and CT2, have a look at this free 500-page book:

B. Esslinger, et al: CrypTool-Book, 12th edition, https://www.cryptool.org/en/ctp-documentation-en/276-ctp-script (2018)

Several of the cryptanalysis algorithms are based on implementations of George Lasry:

G. Lasry, N. Kopal, A. Wacker: Solving the Double Transposition Challenge with a Divide-and-Conquer Approach. In: Cryptologia, 38, 3 (2014), 197–214

G. Lasry, N. Kopal, A. Wacker: Ciphertext-only Cryptanalysis of Hagelin M-209 Pins and Lugs. In: Cryptologia, 40, 2 (2016), 141–176

G. Lasry, N. Kopal, A. Wacker: Cryptanalysis of Columnar Transposition Cipher with Long Keys. In: Cryptologia, 40, 4 (2016), 374–398

G. Lasry: A Methodology for the Cryptanalysis of Classical Ciphers with Search Metaheuristics. kassel university press GmbH (2018)

G. Lasry, I. Niebel, N. Kopal, A. Wacker: Deciphering ADFGVX Messages from the Eastern Front of World War I. In: Cryptologia, 41, 2 (2017), 101–136