Workshop – Prague October 2018

CrypTool – A Wide-Spread and Free Program to Help Raising Crypto Awareness

Prof. Bernhard Esslinger

Introduction

During this workshop you will learn how to apply CrypTool 2 (CT2) to encrypt and decrypt texts using different ciphers. We will briefly present some attacks on the ciphers as well as a basic introduction to password (in-)security. You will touch understanding for the importance of some cryptographic concepts like randomness. This exercise material can be used as an instruction manual for CT2.

Structure of this workshop

The first chapter **Symmetric Cryptography** shows how to work with ciphers, i.e. encrypting and decrypting texts. First, we use a **Caesar cipher (ca 10 min)**, which was used centuries ago by the Romans. Then, we encrypt a text using a modern symmetric cipher (Advanced Encryption Standard **AES) (ca 10 min**) and how to perform a brute-force attack. The second chapter shows how to use CT2 to encrypt a text using the **RSA cipher (ca 10 min)**. Furthermore, we show how to attack the RSA cipher by factorization (general approach) and in the faulty case when RSA used shared prime factors. The third chapter introduces the **(in-)security of passwords (ca 10 min)** and applies a dictionary attack.

The first (and extensive) appendix introduces the **usage of the CrypTool 2 application**. Due to time restrictions, this chapter is intended for reading at home as a recapitulation of the usage of CT2.

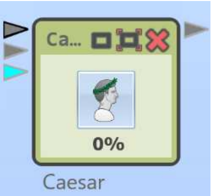
The second appendix contains an overview of all **tasks of the workshop**.

The third appendix offers **links, references, and literature.**

1. Symmetric Cryptography

a. Classic Cipher (Caesar)

CrypTool 2 (CT2) contains various classic ciphers. First, we use the **Caesar** cipher, which is one of the easiest substitution ciphers.



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

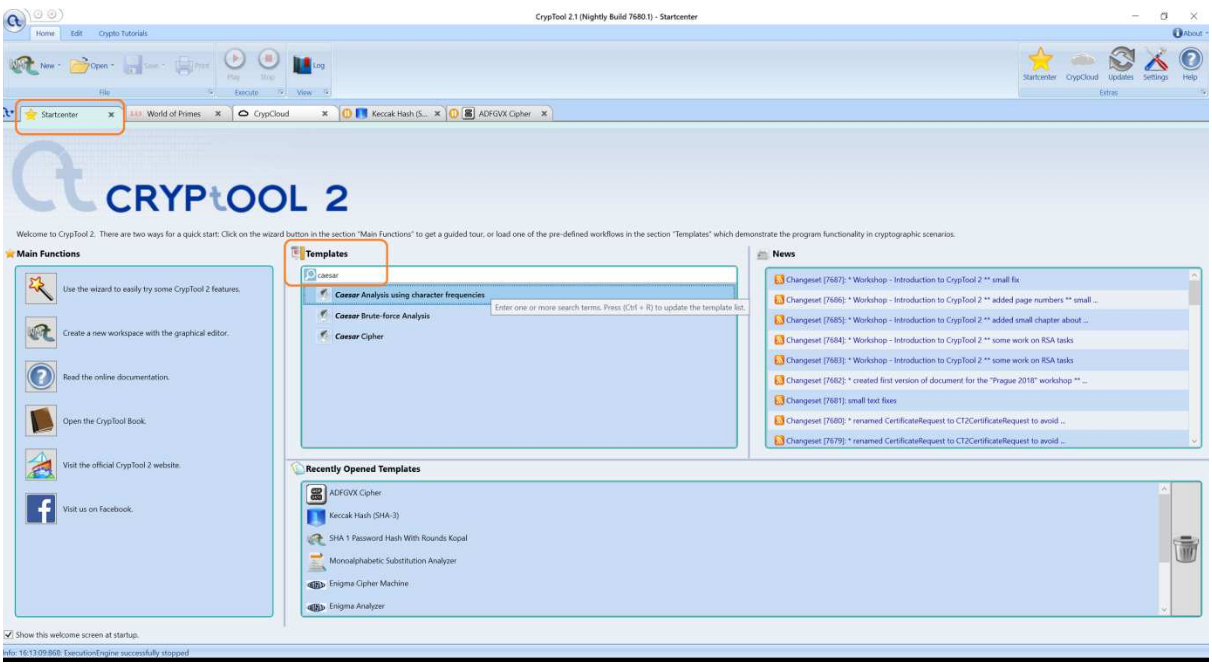
```
Va pelcgbtencul, n Pnrfne pvcure, nyfb xabja nf Pnrfne'f pvcure, gur
fuvsg pvcure, Pnrfne'f pbqr be Pnrfne fuvsg, vf bar bs gur fvzcyrfg
naq zbfg jvqryl xabja rapelcgvba grpuavdhrf. Vg vf n glcr bs
fhofgvghgvba pvcure va juvpu rnpu yrggre va gur cynvagrkg vf
ercynprq ol n yrggre fbzr svkrq ahzore bs cbfvgvbaf qbja gur
nycunorg. Sbe rknzcyr, jvgu n yrsg fuvsg bs 3, Q jbhyq or ercynprq
ol N, R jbhyq orpbzr O, naq fb ba. Gur zrgubq vf anzrq nsgre Whyvhf
Pnrfne, jub hfrq vg va uvf cevingr pbeerfcbaqrapr.
```

Key: 13

Hint 1: Open the template “Caesar Cipher” in CT2 (or use the Wizard).

Hint 2: To copy the ciphertext above, mark it with the mouse and press “Control key + C”. Then in CT2, enter the text by pasting it (pressing “Control key + V”) into the text input component. In order to start the template, click on the “Play” button.

Hint 3: To use the cipher’s corresponding analysis methods, go to the Startcenter in CT2 and enter the search string “Caesar” in the template filter to search for appropriate templates.



<u>Task 2:</u> Encrypt the following text using the Caesar cipher built in CT2:

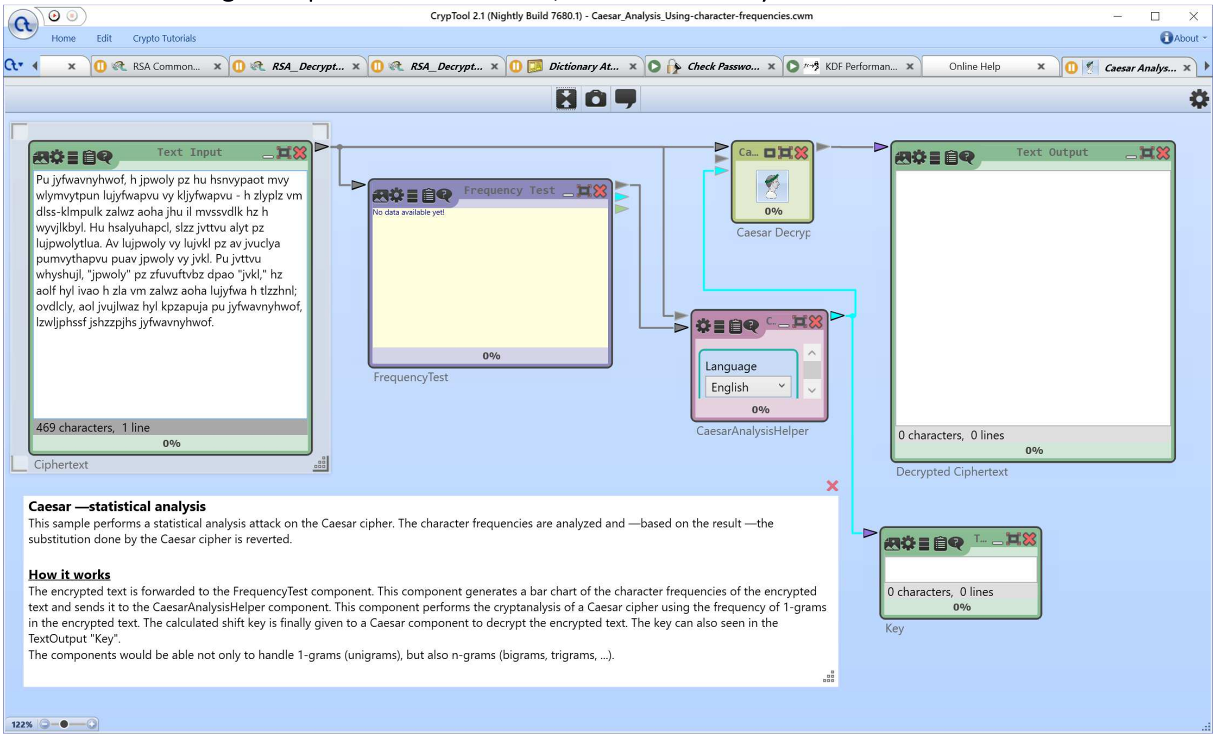
```
Gaius Julius Caesar known by his cognomen Julius Caesar was a Roman
politician and military general who played a critical role in the
events that led to the demise of the Roman Republic and the rise of
the Roman Empire. He is also known as an author of Latin prose.
```

Key: 10

<u>Task 3:</u> Break the following text using the template "Caesar Analysis using character frequencies":

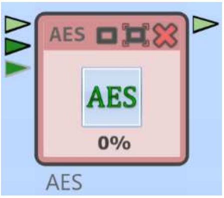
```
Pu jyfwavnyhwof, h jpwoly pz hu hsnvypaot mvy wlymvytpun lujyfwapvu
vy kljyfwapvu -- h zlyplz vm dlss-klmpulk zalwz aoha jhu il mvssvdlk
hz h wyvjlkbyl. Hu hsalyuhapcl, slzz jvttvu alyt pz lujpwolytlua. Av
lujpwoly vy lujvkl pz av jvuclya pumvythapvu puav jpwoly vy jvkl. Pu
jvttvu whyshujl, "jpwoly" pz zfuvuftvbz dpao "jvkl," hz aolf hyl
ivao h zla vm zalwz aoha lujyfwa h tlzzhnl; ovdlcly, aol jvujlwaz
hyl kpzapuja pu jyfwavnyhwof, lzwljphssf jshzzpjhs jyfwavnyhwof.
```

Hint: After entering the ciphertext from above, click on the "Play" button.



b. Modern Cipher (AES)

CrypTool 2 (CT2) contains different modern ciphers. We will have a closer look at the **Advanced Encryption Standard (AES)** cipher, which is the current standard for modern block ciphers. Open the according templates in CT2 – again by using the template filter.

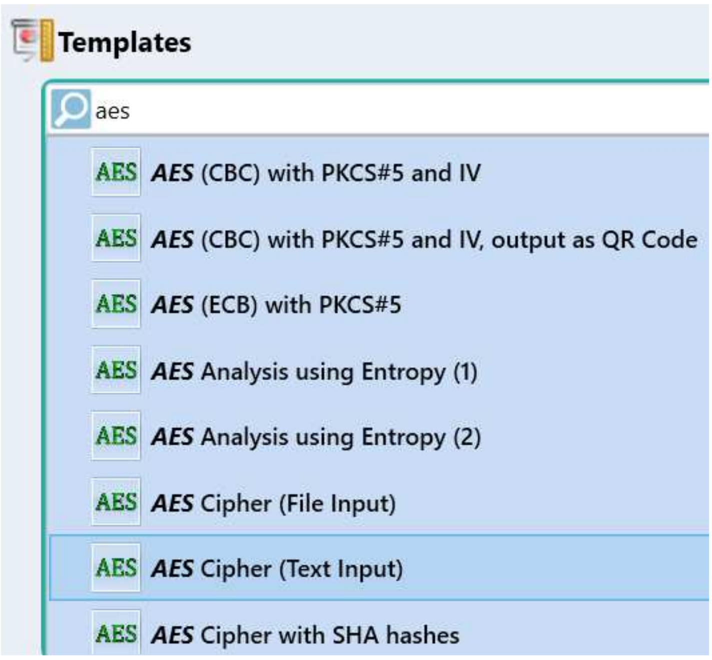


Task 4: Decrypt the following text using the AES cipher built in CT2 (ciphertext data is hex encoded):

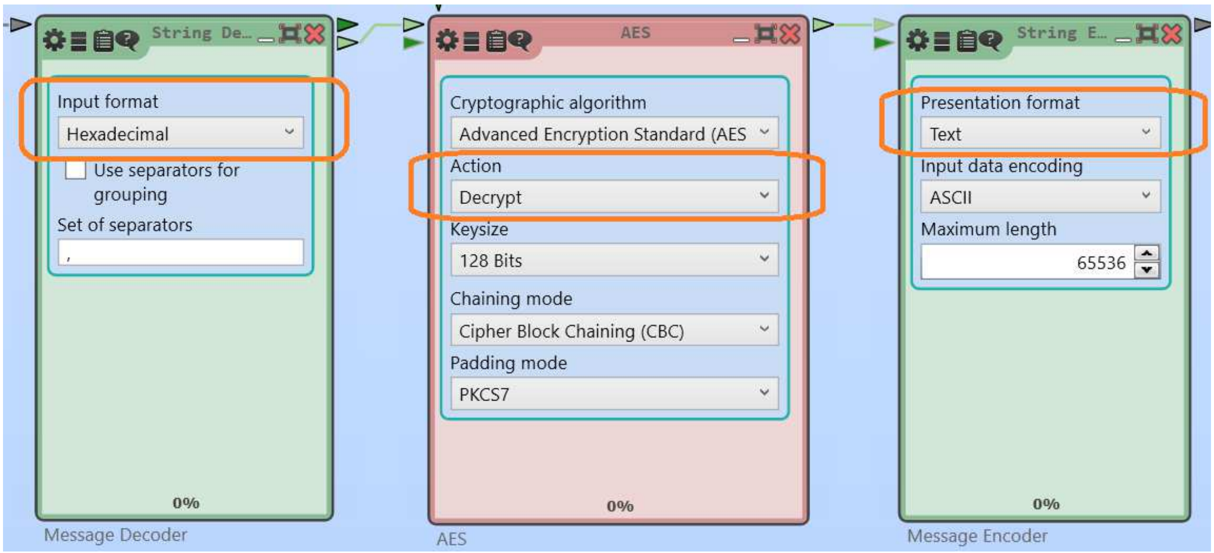
```
0D 96 CF F8 7A 16 54 62 61 07 2D F3 E2 60 FD F9 9A 2C 05 FF 44 D5 B3 13 B8 28 C9 66 48 EC C8
85 A4 39 3D 9A D5 47 6D CD B6 45 27 67 03 F4 E2 19 5D 50 2A F1 2E 9A 48 0D 60 BE A3 DA 4D 36
C0 85 3E 21 34 9D 04 7A DA D4 64 86 BA AA 7B 68 BD A2 AA 05 F4 24 3B 48 BF 62 2D C1 63 1F DE
76 22 2F 95 F2 30 48 CE F9 7A 45 6F 99 74 1E B1 5B 1B AB 4C 29 B6 5C BB 38 94 90 92 51 43 40
DF 46 09 52 70 EB 4B 80 F7 6B 61 C3 68 0B E5 A7 9F FF 71 02 6B DB CD E0 D9 11 D3 E1 56 FE B1
AE E0 57 A1 93 7D 38 7E 6E 4D 1E 4B 92 26 25 E1 6B A9 B1 A1 8F 4B C0 78 2D DC BB 13 84 2A 07
4E F7 DD B3 3A 77 E1 F6 7C A7 9E DA 32 D7 50 51 E4 AE B5 CE C5 67 29 87 31 5C E9 B3 EB 99 80
3C 91 D9 75 30 31 FE 5F CF DC 2D 14 07 6A 1E 33 89 ED D8 D3 3C 98 68
```

Key: **FD E8 F7 A9 B8 6C 3B FF 07 C0 D3 9D 04 60 5E DD**

Hint 1: Open the template “AES Cipher (Text Input)”.



Hint 2: Change the "Message Encoder" input format to "Hexadecimal", set the "AES" action to "Decrypt", and the "Message Decoder" presentation format to "Text".



<u>Task 5:</u> Encrypt the following text using the AES cipher built in CT2:

`AES is a subset of the Rijndael block cipher developed by two Belgian cryptographers, Vincent Rijmen and Joan Daemen.`

Key: **FD E8 F7 A9 B8 6C 3B FF 07 C0 D3 9D 04 60 5E DD**

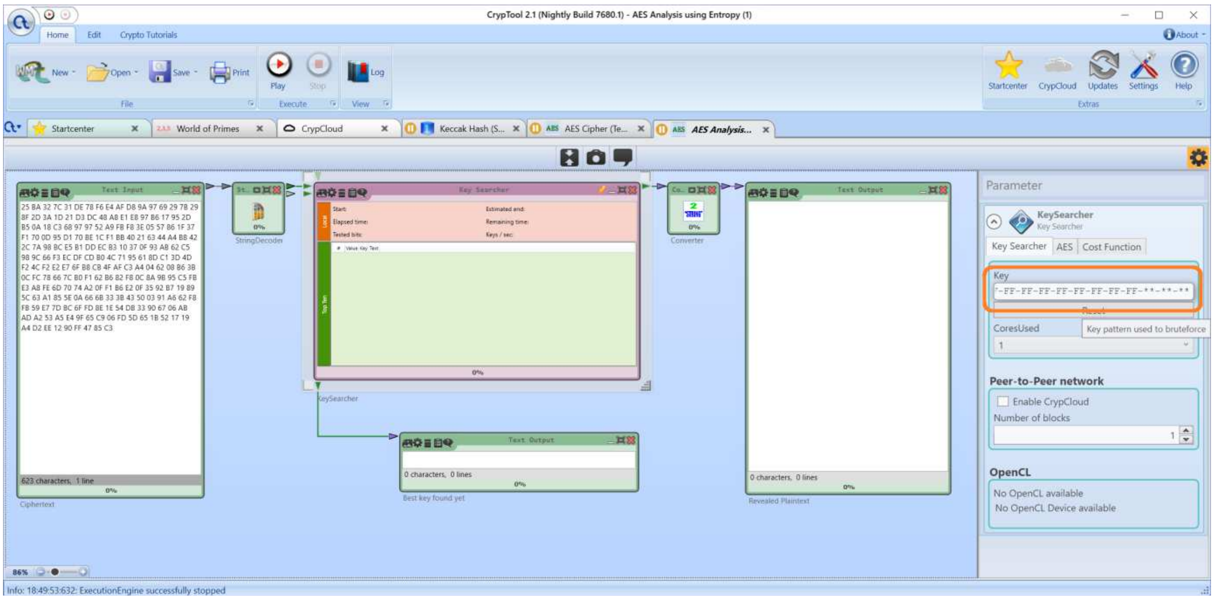
Hint: Open the template "AES Cipher (Text Input)".

<u>Task 6:</u> Break the following ciphertext using the template "AES Analysis using Entropy (1)":

```
25 BA 32 7C 31 DE 78 F6 E4 AF D8 9A 97 69 29 7B 29 8F 2D 3A 1D 21 D3 DC 48 A8 E1 E8 97 B6 17
95 2D B5 0A 18 C3 68 97 97 52 A9 FB F8 3E 05 57 86 1F 37 F1 70 0D 95 D1 70 BE 1C F1 BB 40 21
63 44 A4 B8 42 2C 7A 98 BC E5 B1 DD EC B3 10 37 0F 93 AB 62 C5 98 9C 66 F3 EC DF CD B0 4C 71
95 61 8D C1 3D 4D F2 4C F2 E2 E7 6F B8 CB 4F AF C3 A4 04 62 08 B6 3B 0C FC 78 66 7C B0 F1 62
B6 82 F8 0C 8A 9B 95 C5 FB E3 A8 FE 6D 70 74 A2 0F F1 B6 E2 0F 35 92 B7 19 89 5C 63 A1 85 5E
0A 66 6B 33 3B 43 50 03 91 A6 62 F8 FB 59 E7 7D BC 6F FD 8E 1E 54 DB 33 90 67 06 AB AD A2 53
A5 E4 9F 65 C9 06 FD 5D 65 1B 52 17 19 A4 D2 EE 12 90 FF 47 85 C3
```

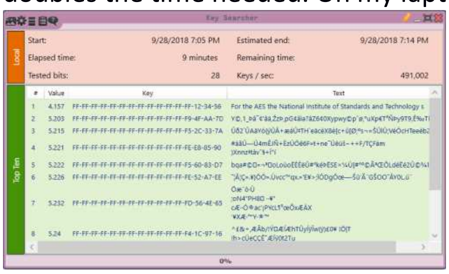
Key: **FF FF FF FF FF FF FF FF FF FF FF FF FF ?? ?? ??**

Hint: The cipher used was AES-128. Set the key in the settings of the "Key Searcher" to
`FF-FF-FF-FF-FF-FF-FF-FF-FF-FF-FF-FF-FF-**-**-**`
One hex character corresponds to 4 bit; a 128 bit key can be described by 32 hex chars (32*4=128). Here, the last 6 hex chars are asterisks, so 24 bit are unknown.



<u>Task 6a:</u> Same task as before, but less known key bits: Try using patterns with more additional asterisks (*) to get a feeling of search time if you know less bits of the key:
`FF-FF-FF-FF-FF-FF-FF-FF-FF-FF-FF-FF-F*-**-**-**`

This means, we know 4 bits less than before (first we knew 128-24=104 bit; now, 7*4=28 bits are unknown, so we know 100 bit of the AES key). For all unknown bits the **KeySearcher** component in CT2 processes an exhaustive search (**brute-force attack**). 4 more unknown bits mean that we can expect that the brute-force attack needs 2^4=16 times the time needed before. Each unknown bit doubles the time needed. On my laptop task 6 needed 35 seconds; task 6a needed 9 minutes.



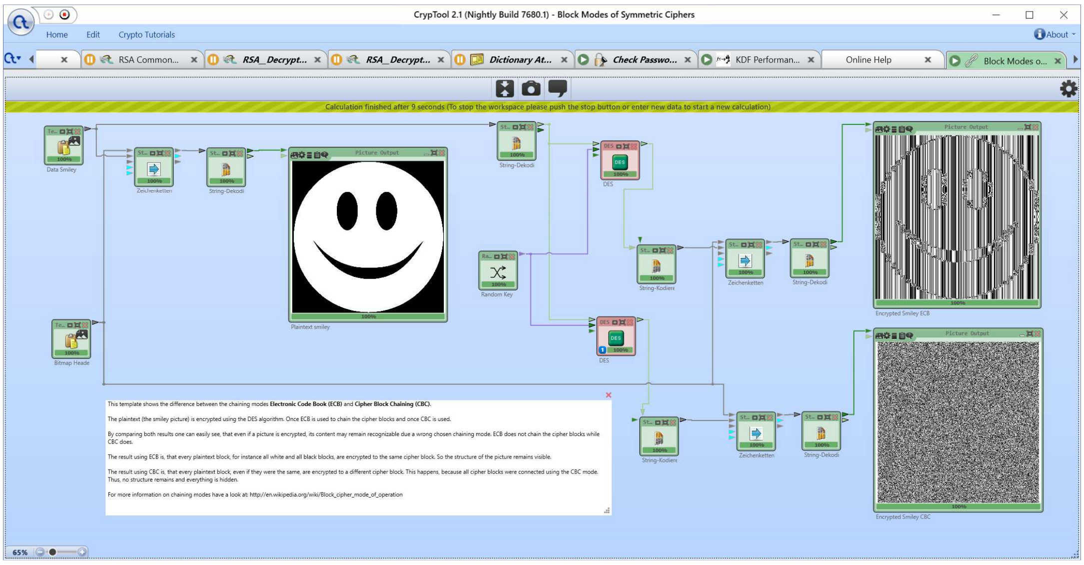
Remark: **Block modes of modern symmetric ciphers**

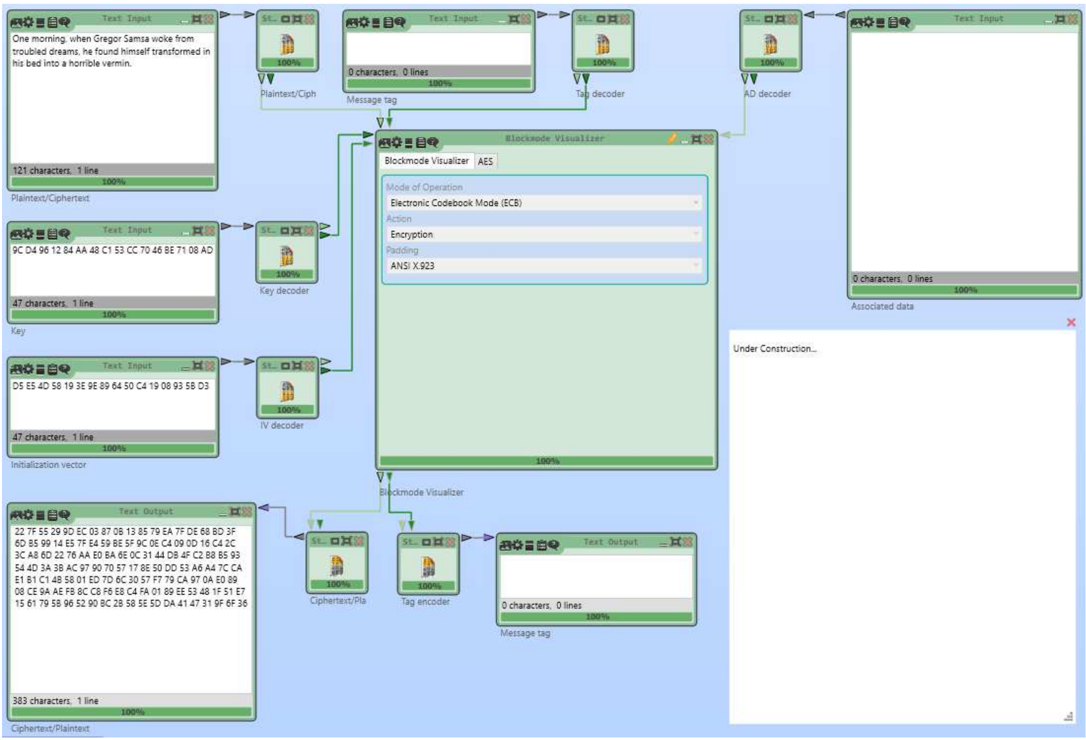
There are two big families of modern symmetric ciphers: block and stream ciphers. Stream ciphers work on single bytes, block cipher work on blocks of n bytes, where n usually is 64, 128, or 256.

AES-128 is a block cipher with a block length of 128 bits (= 16 byte) and a key length of 128 bits. NIST defined three AES standards: AES-128, AES-192, and AES-256, each with a block size of 128 bits, but three different key lengths: 128, 192 and 256 bits.

If you encrypt messages longer than one block, you have to decide how you make sure, that all blocks are full and how to build consecutive blocks.

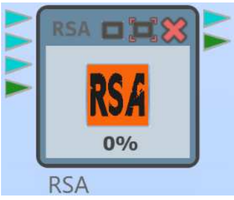
There are different standards – the one recommended today in most cases is the GCM mode. The following 2 templates show how block modes work and what can happen if you choose a wrong one.





2. RSA Cipher

CrypTool 2 (CT2) contains the **RSA Cipher**, named after their inventors Rivest, Shamir, and Adleman.



Task 7: Decrypt the following text using the RSA cipher built in CT2:

```
88 9D D8 2A 7A 14 8C 06 79 BA AC EF BB 85 1E C1 4A AF 84 85 FE A5 0B 1D C3 C6 D9 8D 4C 01 74
57 3F 9A 2D BE CB 14 77 24 24 E1 B6 E9 D6 03 D0 F7 C5 39 5F 72 EF 5E 96 10 A5 21 AB 65 D0 FE
E9 B2 6C 02 EB 61 15 AD E1 70 76 19 C0 0A D4 F2 CD D9 AB 86 54 1A 4D BC 25 7E 0A EC 42 30 FC
17 50 7E F6 D4 B9 7D 58 A1 1D 12 F8 5D E6 63 77 14 16 3B 87 C9 8B B2 EC 02 65 E4 42 71 60 6B
3F E8 7E 64 4B 0D 31 22 8F B7 05 2B C0 36 76 C8 D6 49 94 99 66 26 AD E4 73 18 ED 33 3D 10 C4
77 29 F3 B4 C3 F3 CC 9C 64 B2 2D D9 58 6D 6A 22 11 EF 2F D3 96 EE 33 6B 2B 87 67 01 F2 04 93
FE F0 2C FE E7 3C B1 70 42 3A DD CE 43 5A EE 94 AB 65 78 F7 49 E9 33 AA BB 8C E3 FA 0D A0 B7
85 B4 42 B0 CF C6 9F 15 E9 F8 B0 1B 65 86 8F 6E 07 F9 9E 73 35 36 6A E2 A2 18
```

Key: public key = (e, N); private key = (d, N)

```
N =
97837973726418359868516951718991281325771149750958732944765111213631
32802749392574002300093727799031589158811983556294019011356333461547
11470896455639414844598988543772530316799684342260008657372442996653
93453851802313775580309976978804698982229486068546397607971083305570
96835887020940910268417082718771 2579
```

```
e = 11
```

```
d =
53366167487137287201009246392177062541329718045977490697144606116526
17892408759585819436414760617653594086624718303433101278921636433571
15347761703076044352756218828909257224867912166639117664812409274736
04083681494108652553529557950472379863877351129463207267185120618342
08412930655863198715544210802225 1891
```

Hint: Open the template “RSA Decryption” in CT2 (or use the Wizard). You need the **parameters of the private key** to do the decryption.

Task 8: Encrypt the following text using the RSA cipher built in CT2:

```
The idea of an asymmetric public-private key cryptosystem is
attributed to Whitfield Diffie and Martin Hellman, who published
this concept in 1976.
```

Key: Same as in Task 7. Don’t confuse e and d.

Hint: Open the template “RSA Encryption”.

<u>Task 9:</u> Break the following ciphertext by factorizing the small N (59 decimal chars, 195 bit).

```
65 13 D4 83 A9 FF AD 6F E4 35 F1 80 98 F9 31 08 16 8D C3 86 20 47 E7 D8 05 18 B0 52 EB 1F 3E
CF 0C EA C3 6C 8E A3 63 60 B5 69 D8 9C 55 4F AA 62 E5 00 52 17 F1 BB 1C 43 AC 21 02 98 BE 1E
E0 E4 EA 18 C1 4B D6 4C 18 8C 2F AA 06 41 2A 10 D4 70 B5 2E 4C 58 E4 E1 BE 49 8D E0 B3 0C 26
9C 78 F6 47 67 F9 06 E4 FE E9 3C AB AE 07 89 2F 47 17 F3 BA DD EF C8 95 41 ED 80 15 14 6D 6E
05 88 77 13 62 DB 1E 25 A7 0D 93 3B D4 99 AD 71 95 7E 2D 77 32 04 7D 68 13 03
```

Key:

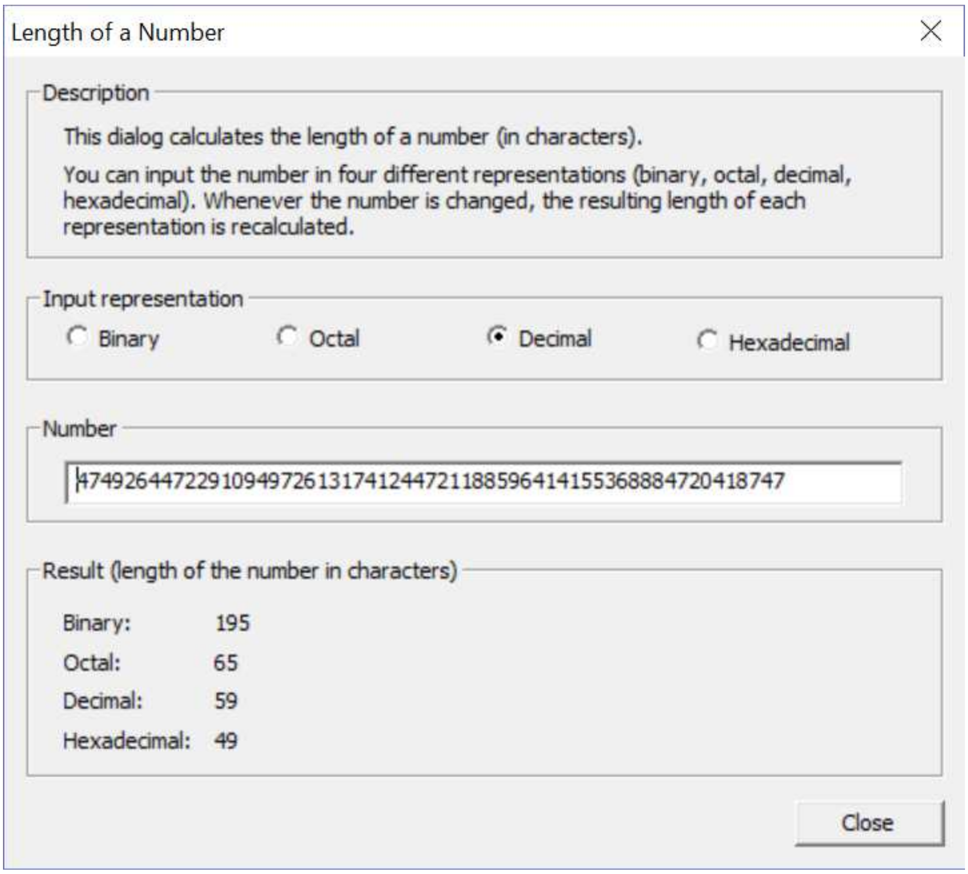
`N = 47492644722910949726131741244721188596414155368884720418747`

`e = 11`

Hint 1: Please **download** the template (cwm file) "*RSA_Decryption2.cwm*".

Hint 2: As you now only know the **parameters of the public key**, you additionally need to do something to get the private key too (the private key is needed for decryption). First, use the pre-delivered template "Factorization with Quadratic Sieve (QS)" to factor N into p and q. Then, use the just-downloaded template "*RSA_Decryption2.cwm*". Here, enter the values for p, q, and the public exponent e in the component "RSA Key Generator". Finally, enter the ciphertext and decrypt it by starting the template.

Remark 1: The following dialog box from CT1 shows the length of the above N in different number representations.



Remark 2: The longest RSA module N (constructed securely) which was factorized yet had a length of 768 bits (as far as known by public research and on conventional computers). Researchers think that NSA is able to factorize 1024 bit RSA keys.

Task 10: Now we consider a case where the RSA keys were not constructed securely. RSA can be broken easily if for the generation of two different RSA keys a common prime factor happened to be used. The following two RSA modules N_1 and N_2 (each ca. 2048 bit) share a common prime factor (Q).

This sample shows how important good **randomness** is when generating keys. (And trust, if you let others generate keys for you.)

```
N1 =
49141803551209330395504421973190597473238749090405070233199020072658
```

```
N1 =
49141803551209330395504421973190597473238749090405070233199020072658
71460562196022395400841808448651461009934527330576802805345915425323
19808565721178537249064673679288577153001041784388578368088056575620
09143435774459487647414972582639899150407081132639220471840819914349
95354947175127829535520828142793005722796657550829443760389287629980
70282128706612255181882738199057572669987924145781933007942531569489
11653110005635248794572997286481811139646753934648804778388154080614
75429601985651079440922017816087886643854620744254796200412349286656
37000624267531265237621614878919822973443876510516840605212742614243
6713
```

```
N2 =
12067917076244409426080107729684712037355257023636344905272038070287
89984263596057450467715472523879222741417755543386090352246400569489
55411667437565745106795114855264615651938837966630189327255839070851
54361425403996314990311667267506567852444204841505663232811865188256
77690389971021198404197353285339799565122021730121605106082462086875
97044371624277156804861097223487055059183664203100234510399938789063
61392522510728167672121520079774799504944950751191091536139445850825
08450615084911820999234616260126548992370216677628710991787412401631
93667494118070550546636431490136983714541466955241063497292949017662
95539
```

`e = 17`   (e is public and the same for both; d1 and d2 are unknown)

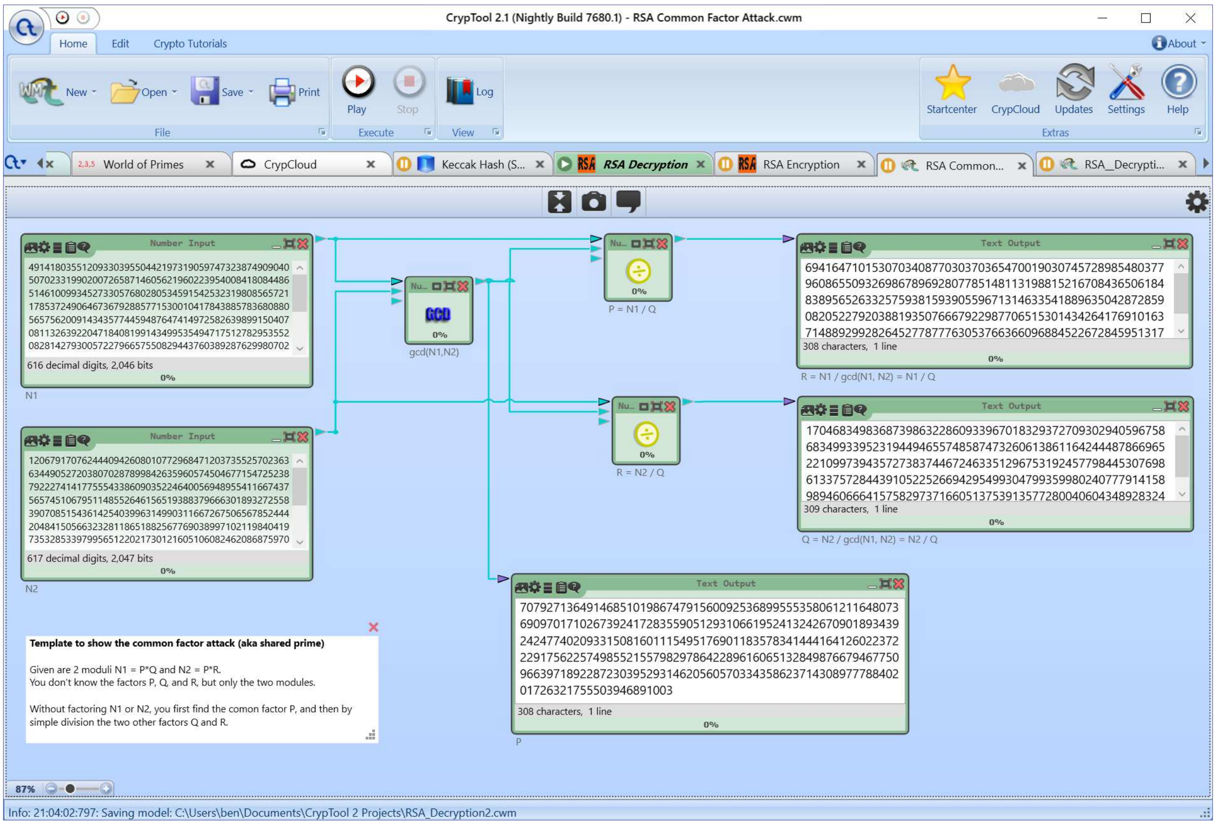
Break the following ciphertext which was encrypted with one of the RSA keys above:

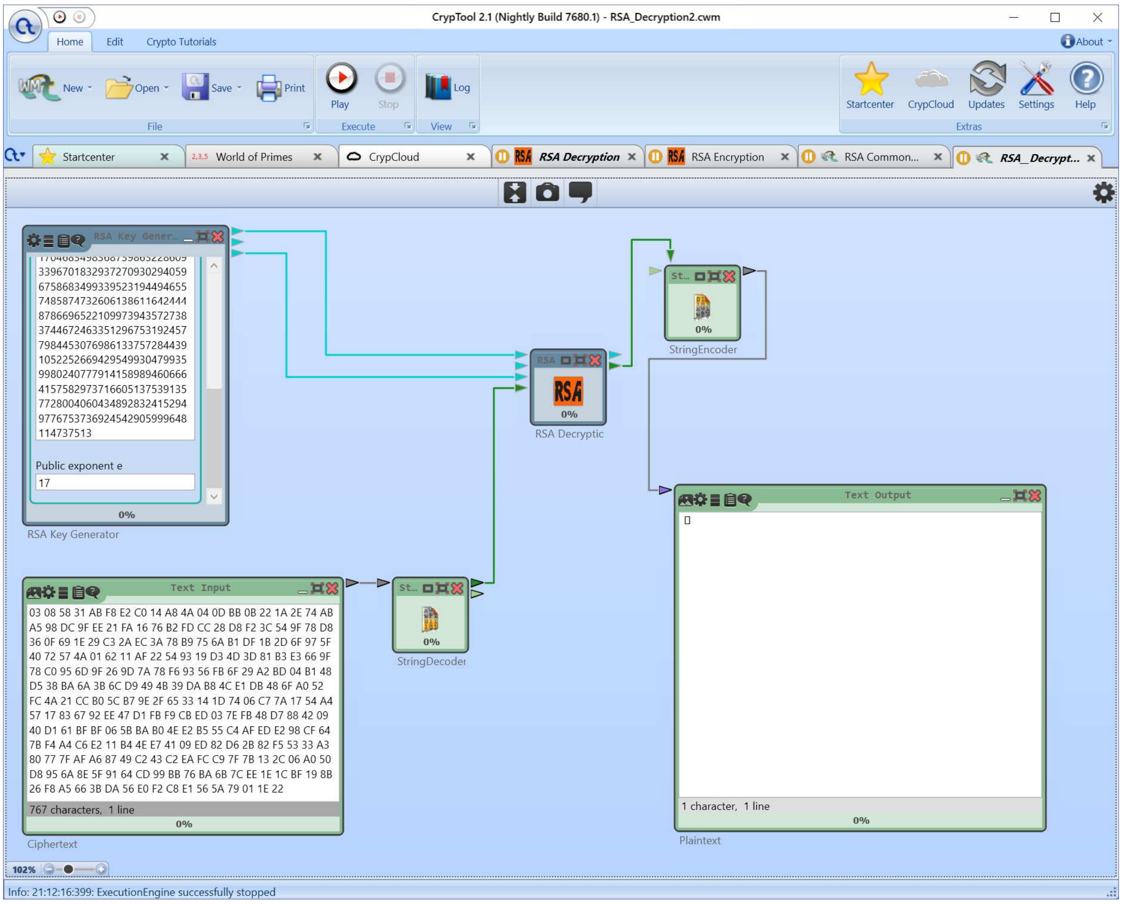
```
03 08 58 31 AB F8 E2 C0 14 A8 4A 04 0D BB 0B 22 1A 2E 74 AB A5 98 DC 9F EE 21 FA 16 76 B2 FD
CC 28 D8 F2 3C 54 9F 78 D8 36 0F 69 1E 29 C3 2A EC 3A 78 B9 75 6A B1 DF 1B 2D 6F 97 5F 40 72
57 4A 01 62 11 AF 22 54 93 19 D3 4D 3D 81 B3 E3 66 9F 78 C0 95 6D 9F 26 9D 7A 78 F6 93 56 FB
6F 29 A2 BD 04 B1 48 D5 38 BA 6A 3B 6C D9 49 4B 39 DA B8 4C E1 DB 48 6F A0 52 FC 4A 21 CC B0
5C B7 9E 2F 65 33 14 1D 74 06 C7 7A 17 54 A4 57 17 83 67 92 EE 47 D1 FB F9 CB ED 03 7E FB 48
D7 88 42 09 40 D1 61 BF BF 06 5B BA B0 4E E2 B5 55 C4 AF ED E2 98 CF 64 7B F4 A4 C6 E2 11 B4
4E E7 41 09 ED 82 D6 2B 82 F5 53 33 A3 80 77 7F AF A6 87 49 C2 43 C2 EA FC C9 7F 7B 13 2C 06
A0 50 D8 95 6A 8E 5F 91 64 CD 99 BB 76 BA 6B 7C EE 1E 1C BF 19 8B 26 F8 A5 66 3B DA 56 E0 F2
C8 E1 56 5A 79 01 1E 22
```

Hint: Here you need to **download** two templates (cwm files) from the same website as from where you downloaded this workshop material (pdf).

First, open the provided template "*RSA Common Factor Attack.cwm*". Here, enter N_1 and N_2 to compute the greatest common divisor (gcd) Q as well as the two other factors P and R.

Then, open the template "*RSA_Decryption2.cwm*" and enter the values P, Q, and e into the "RSA Key Generator". Finally, enter the ciphertext in the according text input component and decrypt it.


CrypTool 2.1 (Nightly Build 7680.1) - RSA Common Factor Attack.cwm
Home
Edit
Crypto Tutorials
About
New
Open
Save
Print
Play
Stop
Log
File
Execute
View
Startcenter
CrypCloud
Updates
Settings
Help
Extras
World of Primes
CrypCloud
Keccak Hash (S...
RSA Decryption
RSA Encryption
RSA Common...
RSA_Decrypti...
Number Input
616 decimal digits, 2,046 bits
N1
617 decimal digits, 2,047 bits
N2
gcd(N1,N2)
P = N1 / Q
R = N2 / Q
Text Output
308 characters, 1 line
R = N1 / gcd(N1, N2) = N1 / Q
Q = N2 / gcd(N1, N2) = N2 / Q
P
Template to show the common factor attack (aka shared prime)
Given are 2 moduli N1 = P*Q and N2 = P*R.
You don't know the factors P, Q, and R, but only the two modules.
Without factoring N1 or N2, you first find the comon factor P, and then by simple division the two other factors Q and R.
87%
Info: 21:04:02:797: Saving model: C:\Users\ben\Documents\CrypTool 2 Projects\RSA_Decryption2.cwm


CrypTool 2.1 (Nightly Build 7680.1) - RSA_Decryption2.cwm
Home
Edit
Crypto Tutorials
About
New
Open
Save
Print
Play
Stop
Log
File
Execute
View
Startcenter
CrypCloud
Updates
Settings
Help
Extras
Startcenter
World of Primes
CrypCloud
RSA Decryption
RSA Encryption
RSA Common...
RSA_Decrypt...
RSA Key Generator
Public exponent e
17
StringEncoder
RSA Decryptic
Text Input
767 characters, 1 line
Ciphertext
StringDecoder
Text Output
1 character, 1 line
Plaintext
102%
Info: 21:12:16:399: ExecutionEngine successfully stopped

3. Password Security

CrypTool 2 (CT2) contains **different hash functions** and **key derivation functions**. To use the, go to the Startcenter and use the template list to search for appropriate templates. Modern password hashing uses key derivation functions like PBKDF2 (better than PKCS#5) – see https://en.wikipedia.org/wiki/Password_hashing or https://en.wikipedia.org/wiki/PBKDF2).
In CT2 this is presented in the templates "PBKDF-1 (PKCS#5 2.0)" and "KDF Performance Comparison (without presentation)".

Here, we show how easy it is to perform dictionary attacks, if passwords are just stored purely as hash values.

Task 11: The following password hashes are built with SHA-256. The passwords are English words, thus, can be easily broken with a **dictionary attack**. Break them:

```
Hash1 = 5E 88 48 98 DA 28 04 71 51 D0 E5 6F 8D C6 29 27 73 60 3D 0D
6A AB BD D6 2A 11 EF 72 1D 15 42 D8

Hash2 = F4 C7 EC 93 81 10 87 E5 C3 DA 50 5C 08 0B E9 BA DA C8 B9 56
D6 2A A6 F1 3E 02 A5 6B BB BA EE 8A

Hash3 = 73 CD 1B 16 C4 FB 83 06 1A D1 8A 0B 29 B9 64 3A 68 D4 64 00
75 A4 66 DC 9E 51 68 2F 84 A8 47 F5

Hash4 = 31 51 08 42 86 F5 95 11 CE 11 D4 FD 23 4B 64 85 01 25 14 58
7A 19 68 FA 55 AF FF 83 B0 5E 21 3E
```

Hint 1: Open the template "Dictionary Attack". Here, enter the different hashes in the text input component "Test password – Hash-value (HEX)". To speed up the search, minimize or remove the two text output components "Found index" and "Actual search password".

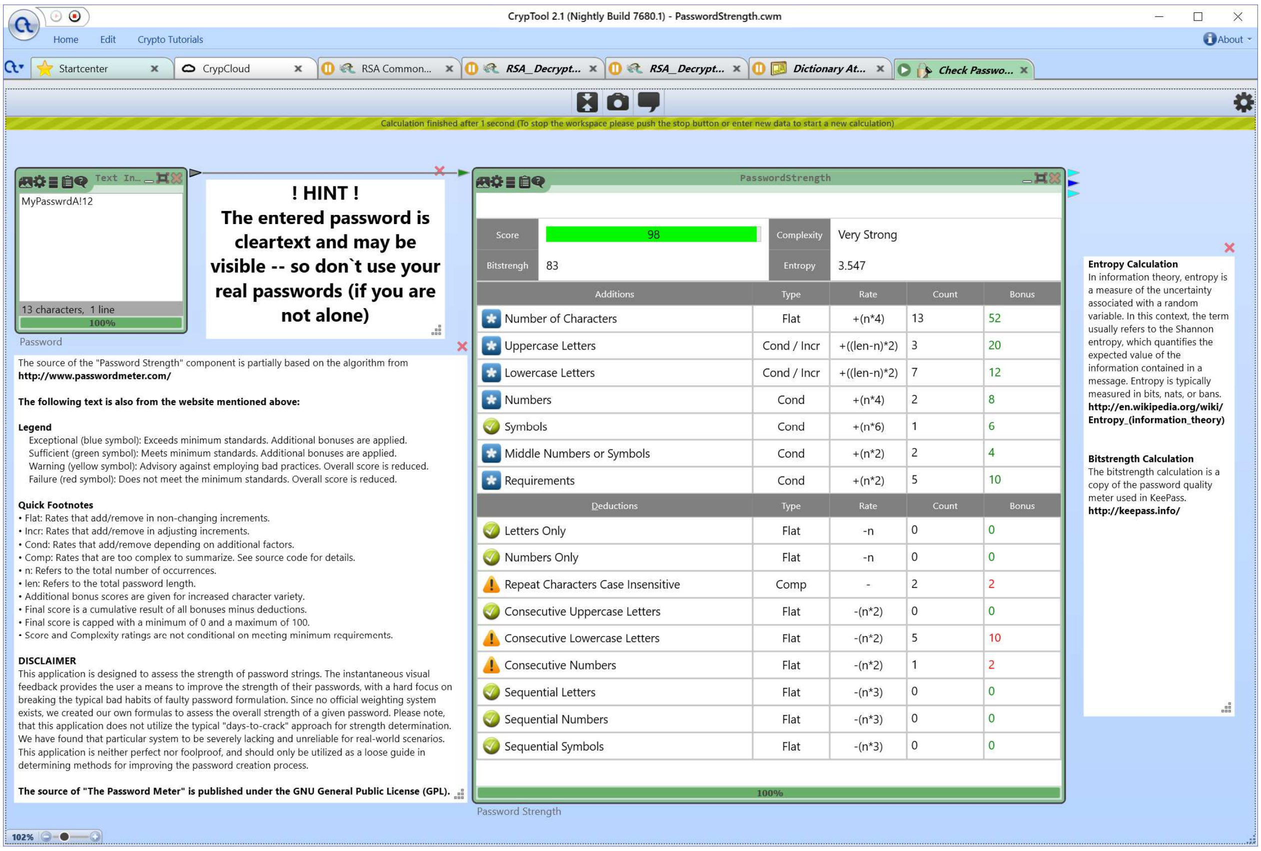
Hint 2: If the password is not derived from a word in a dictionary the attacker has a much more difficult job – he/she has to perform a brute-force attack. On a modern PC around 10^8 hashes/sec can be calculated and checked.

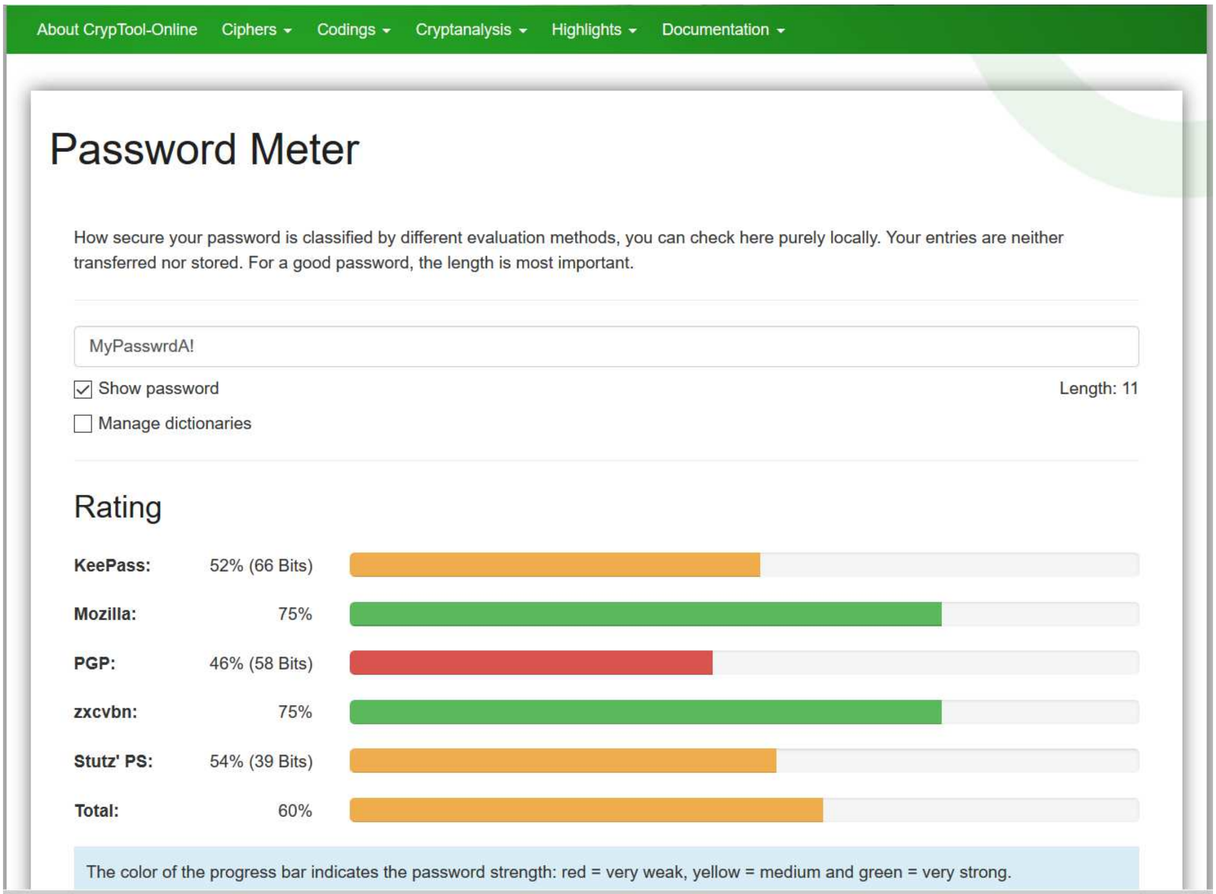
Hint 3: This could be done even much faster with rainbow tables, if the attacker has enough memory. If you buy services, insist that the according vendor uses modern password hashes on their webserver and in their applications. Use your **power of demand** (buyer power).

Task 12: Test the strength of different passwords.

To do so, open the template "Check Password Strength" in CT2. Here, enter the different revealed passwords of Task 11 and give a comment about their strength values. Try to improve these passwords by extending their lengths, adding uppercase letters, numbers, and symbols or try to create a completely new secure password. Please don't enter your real passwords since they are visible in clear. CT2 reacts at once when changing the password. For the password strength, its length is more important than different symbols!

Hint: Alternatively, you could use the plugin "**Password Meter**" on the website CrypTool-Online" (it's entry can be found below the main menu "Highlights"; all calculations are done locally):
https://www.cryptool.org/en/cto-highlights/passwordmeter


CrypTool 2.1 (Nightly Build 7680.1) - PasswordStrength.cwm
Home Edit Crypto Tutorials
Startcenter CrypCloud RSA Common... RSA_Decrypt... RSA_Decrypt... Dictionary At... Check Passwo...
Calculation finished after 1 second (To stop the workspace please push the stop button or enter new data to start a new calculation)
MyPasswrdA!12
13 characters, 1 line
Password
! HINT !
The entered password is cleartext and may be visible -- so don`t use your real passwords (if you are not alone)
The source of the "Password Strength" component is partially based on the algorithm from http://www.passwordmeter.com/
The following text is also from the website mentioned above:
Legend
Exceptional (blue symbol): Exceeds minimum standards. Additional bonuses are applied.
Sufficient (green symbol): Meets minimum standards. Additional bonuses are applied.
Warning (yellow symbol): Advisory against employing bad practices. Overall score is reduced.
Failure (red symbol): Does not meet the minimum standards. Overall score is reduced.
Quick Footnotes
• Flat: Rates that add/remove in non-changing increments.
• Incr: Rates that add/remove in adjusting increments.
• Cond: Rates that add/remove depending on additional factors.
• Comp: Rates that are too complex to summarize. See source code for details.
• n: Refers to the total number of occurrences.
• len: Refers to the total password length.
• Additional bonus scores are given for increased character variety.
• Final score is a cumulative result of all bonuses minus deductions.
• Final score is capped with a minimum of 0 and a maximum of 100.
• Score and Complexity ratings are not conditional on meeting minimum requirements.
DISCLAIMER
This application is designed to assess the strength of password strings. The instantaneous visual feedback provides the user a means to improve the strength of their passwords, with a hard focus on breaking the typical bad habits of faulty password formulation. Since no official weighting system exists, we created our own formulas to assess the overall strength of a given password. Please note, that this application does not utilize the typical "days-to-crack" approach for strength determination. We have found that particular system to be severely lacking and unreliable for real-world scenarios. This application is neither perfect nor foolproof, and should only be utilized as a loose guide in determining methods for improving the password creation process.
The source of "The Password Meter" is published under the GNU General Public License (GPL).
PasswordStrength
Score 98 Complexity Very Strong
Bitstrengh 83 Entropy 3.547
Additions Type Rate Count Bonus
Number of Characters Flat +(n*4) 13 52
Uppercase Letters Cond / Incr +((len-n)*2) 3 20
Lowercase Letters Cond / Incr +((len-n)*2) 7 12
Numbers Cond +(n*4) 2 8
Symbols Cond +(n*6) 1 6
Middle Numbers or Symbols Cond +(n*2) 2 4
Requirements Cond +(n*2) 5 10
Deductions Type Rate Count Bonus
Letters Only Flat -n 0 0
Numbers Only Flat -n 0 0
Repeat Characters Case Insensitive Comp - 2 2
Consecutive Uppercase Letters Flat -(n*2) 0 0
Consecutive Lowercase Letters Flat -(n*2) 5 10
Consecutive Numbers Flat -(n*2) 1 2
Sequential Letters Flat -(n*3) 0 0
Sequential Numbers Flat -(n*3) 0 0
Sequential Symbols Flat -(n*3) 0 0
100%
Password Strength
Entropy Calculation
In information theory, entropy is a measure of the uncertainty associated with a random variable. In this context, the term usually refers to the Shannon entropy, which quantifies the expected value of the information contained in a message. Entropy is typically measured in bits, nats, or bans.
http://en.wikipedia.org/wiki/Entropy_(information_theory)
Bitstrength Calculation
The bitstrength calculation is a copy of the password quality meter used in KeePass.
http://keepass.info/
102%


About CrypTool-Online Ciphers Codings Cryptanalysis Highlights Documentation
Password Meter
How secure your password is classified by different evaluation methods, you can check here purely locally. Your entries are neither transferred nor stored. For a good password, the length is most important.
MyPasswrdA!
Show password
Length: 11
Manage dictionaries
Rating
KeePass: 52% (66 Bits)
Mozilla: 75%
PGP: 46% (58 Bits)
zxcvbn: 75%
Stutz' PS: 54% (39 Bits)
Total: 60%
The color of the progress bar indicates the password strength: red = very weak, yellow = medium and green = very strong.

Appendix 1: Introduction to the CrypTool 2 Application

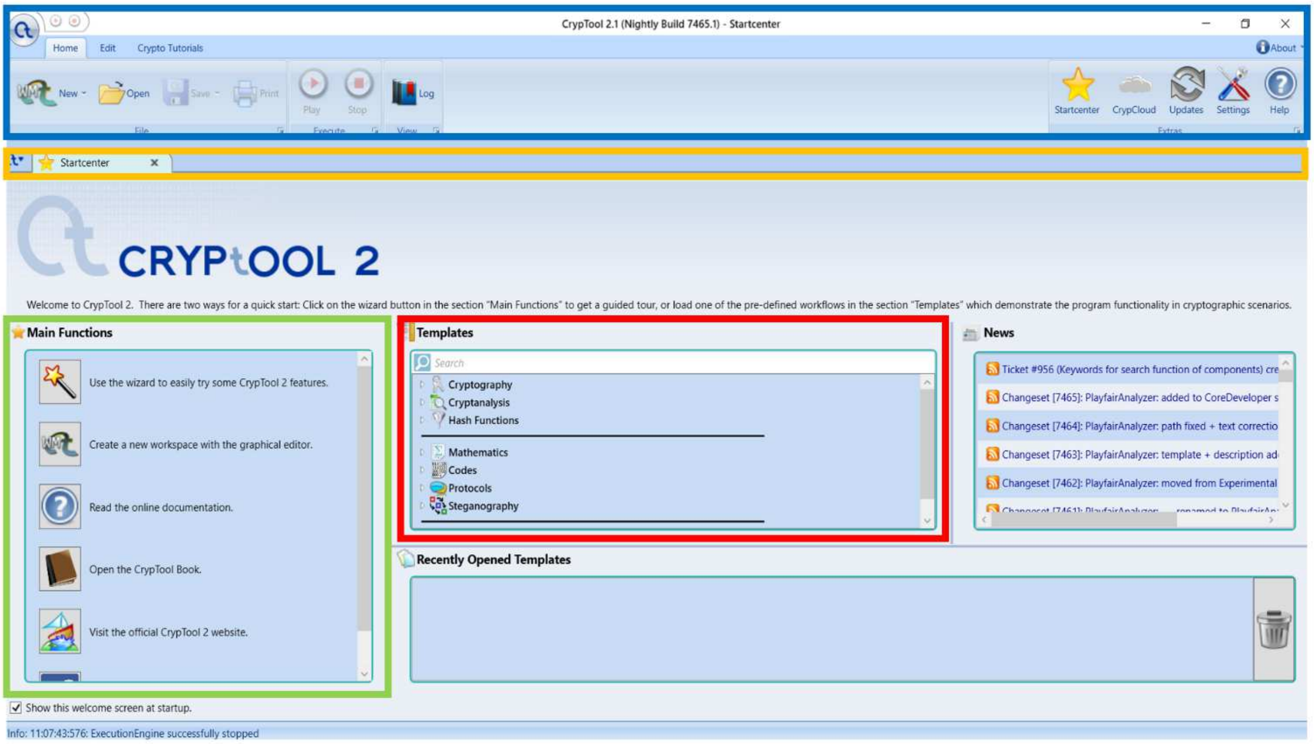
CrypTool 2 (CT2) consists of six main components:

- **Startcenter**,
- **Wizard**,
- **Workspace Manager**,
- **Online Help**,
- **Templates**, and
- **CrypCloud**,

Here, the **Startcenter**, the **Wizard**, and the **Workspace Manager** are explained in more detail.

a. Startcenter

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



CT2 and the Startcenter consist of different areas marked with different colors in the above image.

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

The yellow 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 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 area of the Startcenter contains a list of all "templates" (more than 200) delivered with CT2. A template contains a specific cipher or a ready-to-run cryptanalytic scenario using the graphical programming language of CT2. The list of templates of the Startcenter can be filtered using keywords that can be entered in the search field.

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

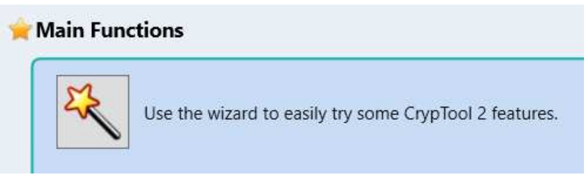
Finally, on the right side of the Startcenter you will see some "news", showing the last changes we did on CT2 with respect to its source code.

b. Wizard

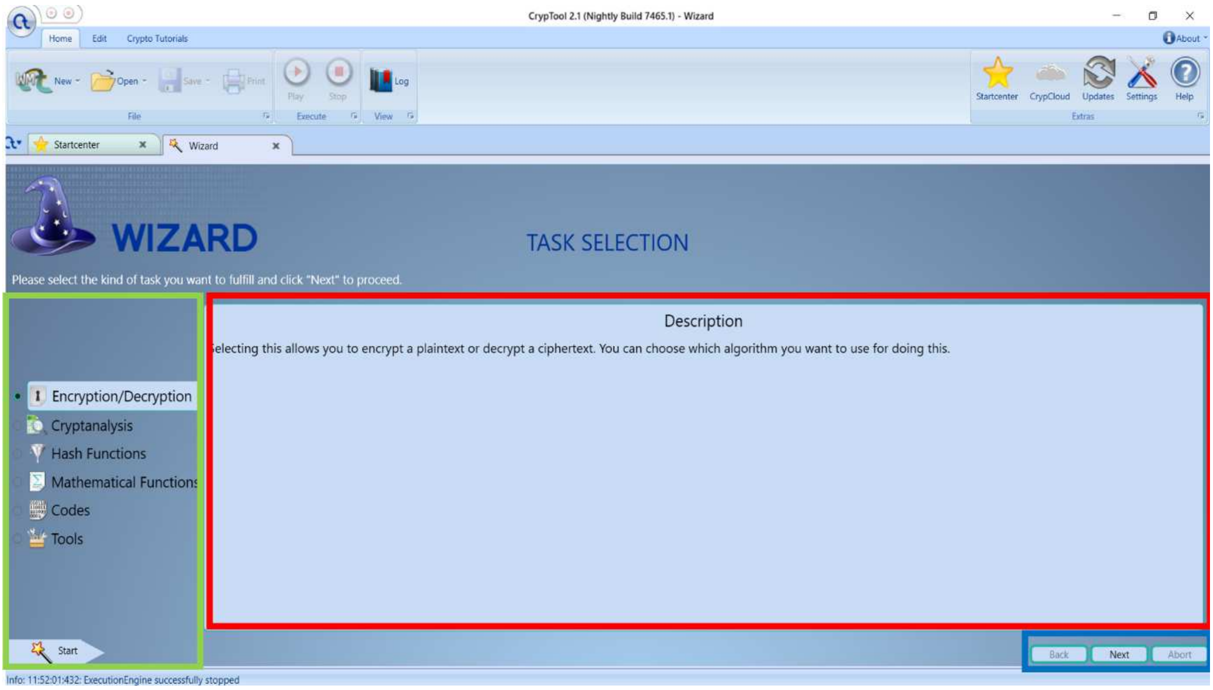
The Wizard is intended for users not familiar with using the graphical programming language of the Workspace Manager and for beginners. It guides you through the different topics of cryptology until you "reach what you want to do", e.g. encrypt something or break something.

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



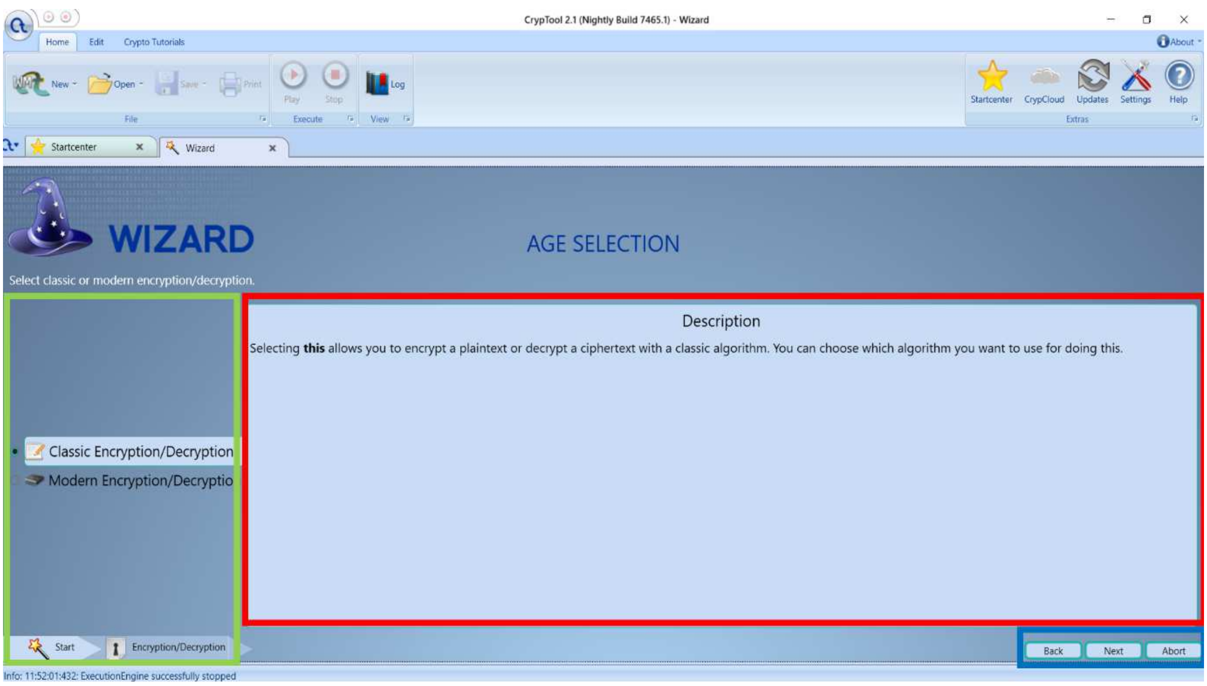
The Wizard consists of three main areas (here marked green, blue, and red).



In the green 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 area.

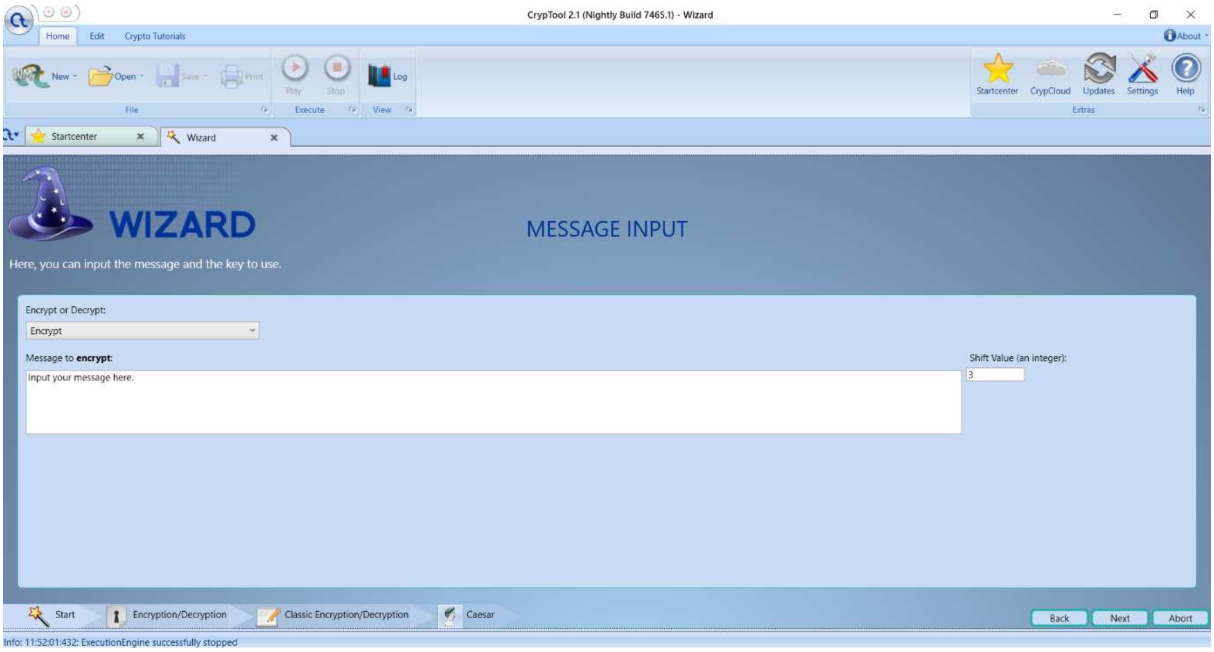
Then, in the red area, the next page will appear.



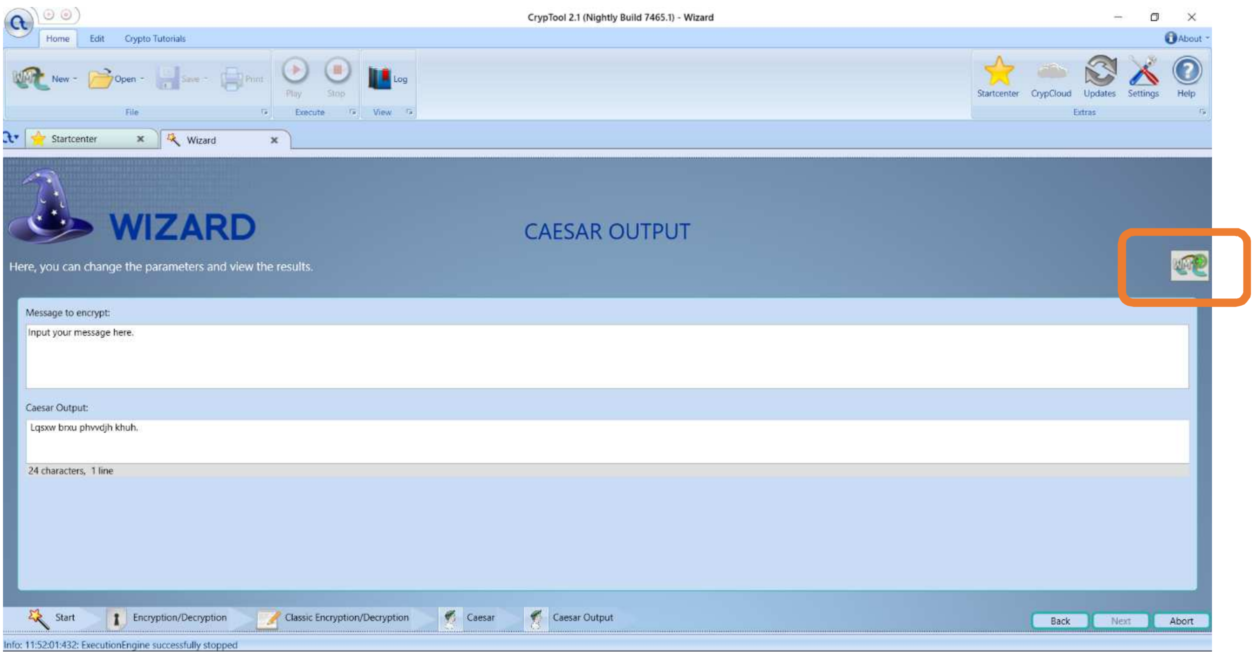
Remark: **Navigation in the Wizard**: Instead of clicking on "Next" you may also double-click in the green area. The fastest way to navigate there is with the 4 arrow keyboard keys.

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. When you finally click “Next”, you will get the encrypted text.



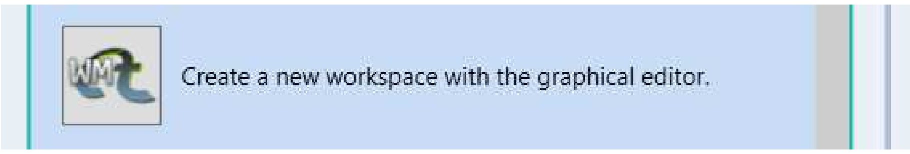
In each final step in the Wizard, you may click on the Workspace Manager icon on the top right side (marked with orange above) of the Wizard to open a template in the Workspace Manager exactly corresponding to the cipher or cryptanalytic method you selected and currently used.

c. Workspace Manager

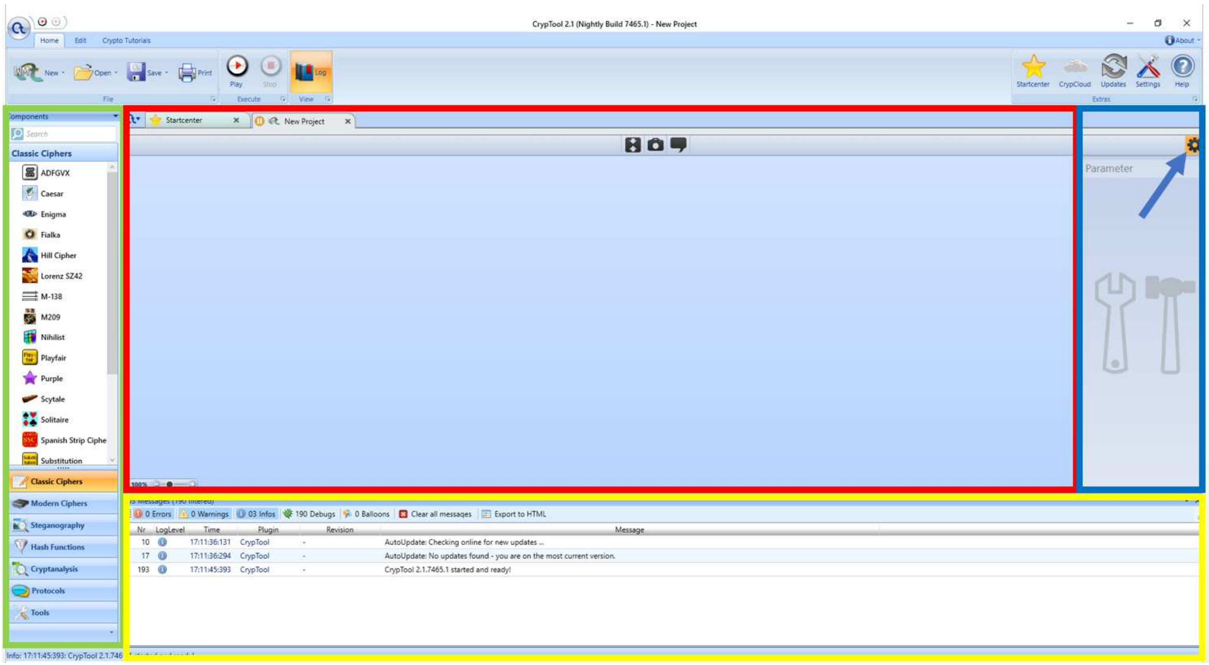
The **Workspace Manager** implements the graphical programming language of CT2. It allows, to create arbitrary cascades of ciphers and cryptanalytic methods.

The Workspace Manager can be started at two different places: First, 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 area in the middle is the actual workspace. It is used to create and present a visual program.

The green area on the left 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.

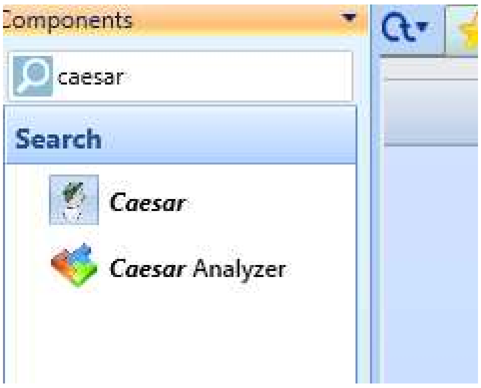
The yellow area is a logging window which contains messages generated by the components during the execution. It can be hidden by clicking the X icon in its upper left corner or via F11.

The blue area on the right side is the settings bar for a selected component. You can select any component in the workspace just by clicking on it. 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).

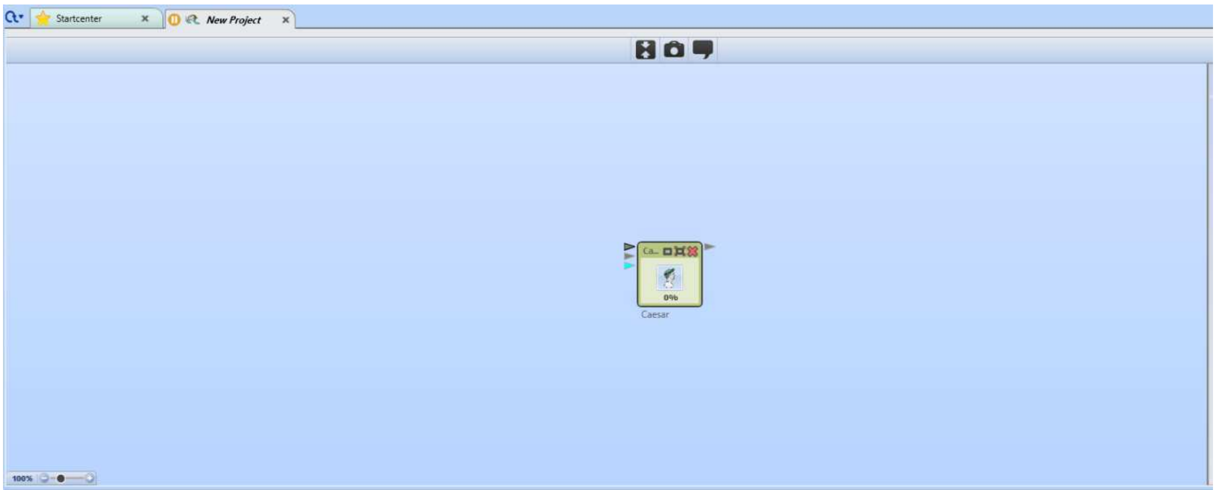
Example: Building a Workflow for the Caesar Cipher

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

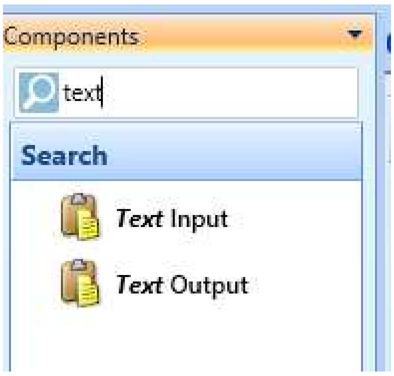
Then, go to the list of components on the left side. Here, enter "caesar" in the search field (it is not case-sensitive).



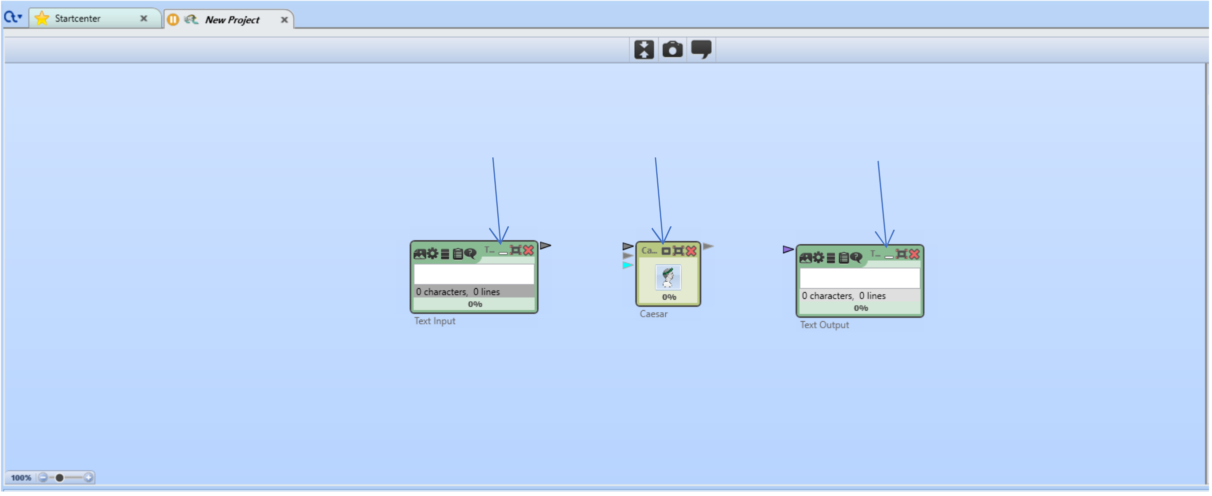
Now, use the left mouse button to drag the "Caesar" component and put it onto the middle of the workspace.



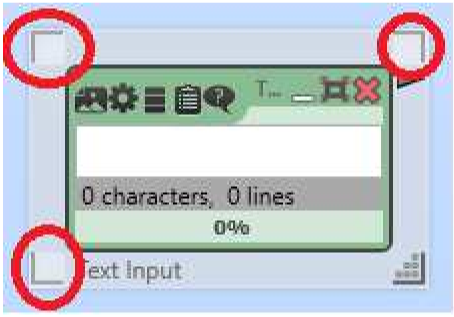
After that, use the components list again to search for “text”.



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.



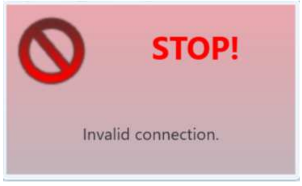
If you want to move them you can always drag a component. A minimized one can be dragged at each position within the icon (like the Caesar component in the picture). If it is “maximized” like the “Text Input” and “Text Output”, select the component by clicking on it. Then you can move the component using 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 its 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.

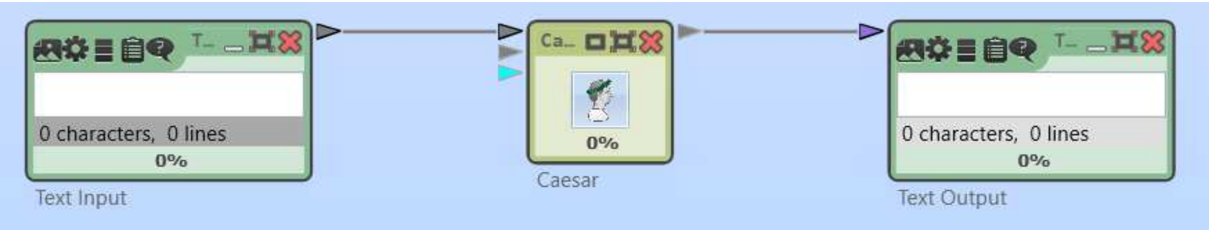
If you want to connect connectors with different colors, you may need converter components. Some data types can be converted implicitly. CT2 will show a hint if this happens.

If a connection is not possible CT2 shows an error.



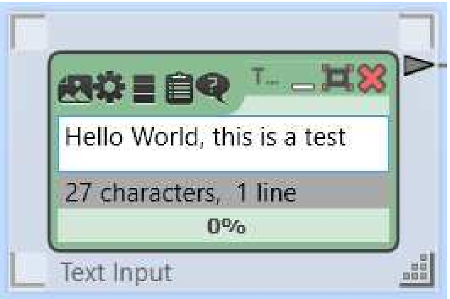
If a connection is valid without any problems CT2 shows a green text.

Now, connect the three components, Caesar, “Text Input”, and “Text Output”, as shown in the next picture. Connecting is done by dragging a line with the mouse from the sender to the receiver.



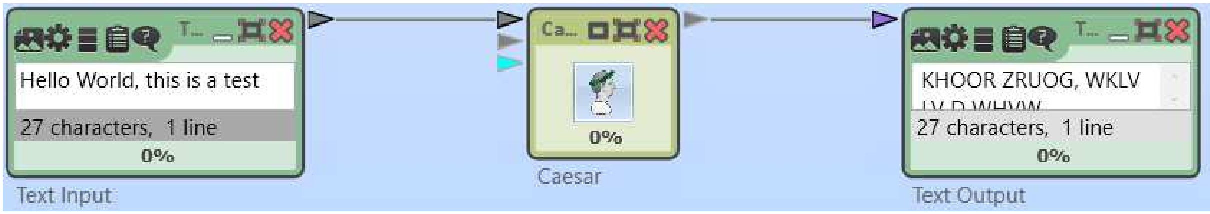
Now, you have built your first graphical program.

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.

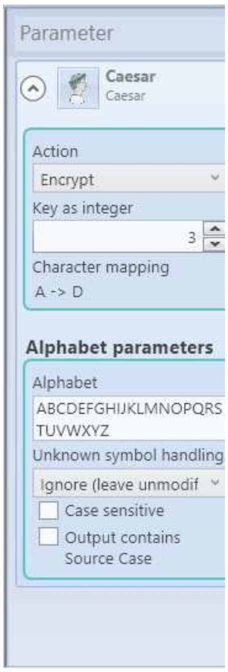
Now, CT2 executes your graphical program. The output should look like this.



Try to type into the “Text Input” while the graphical program is running. 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.

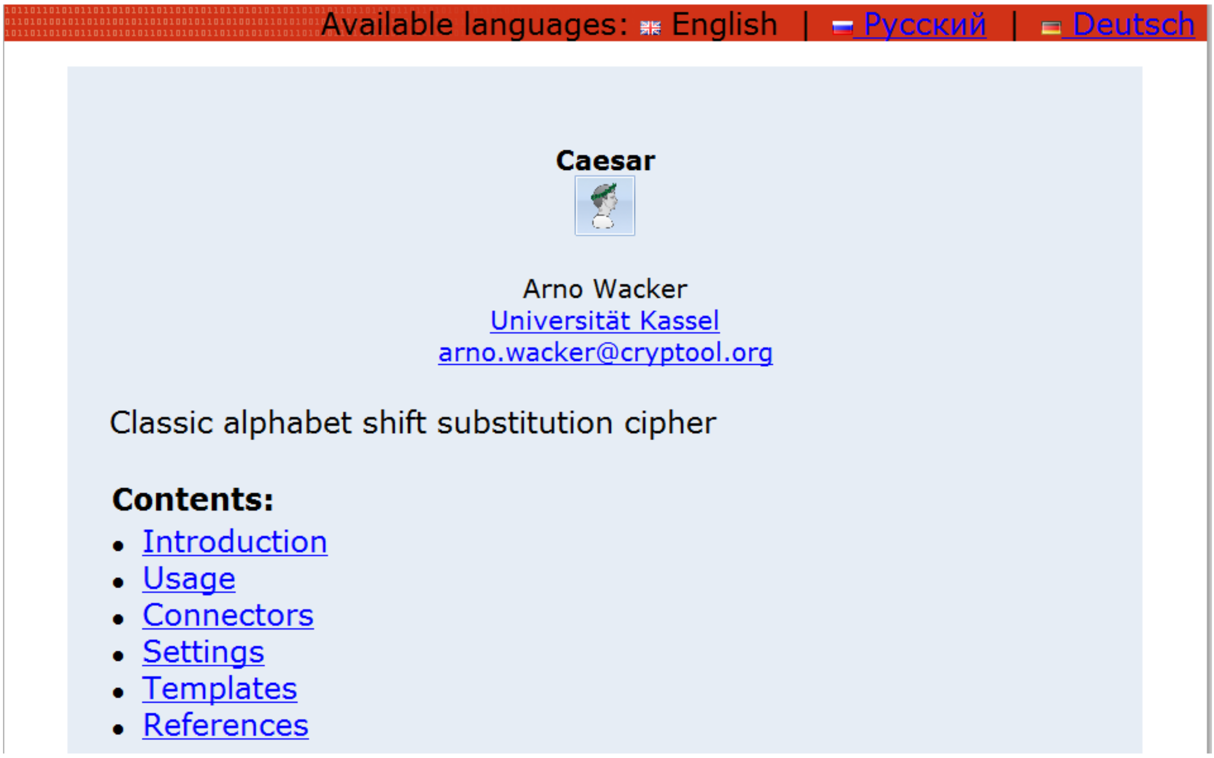
If you want to change for instance the key or any other setting of the Caesar cipher, select the Caesar component (just click on it), and use the toolbar on the right side of the workspace.



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

d. Online help in CT2

For any component in CT2, there in some online help available. If you select a component on the workspace and click on the help icon, the according help file is shown.



Available languages: English | Русский | Deutsch

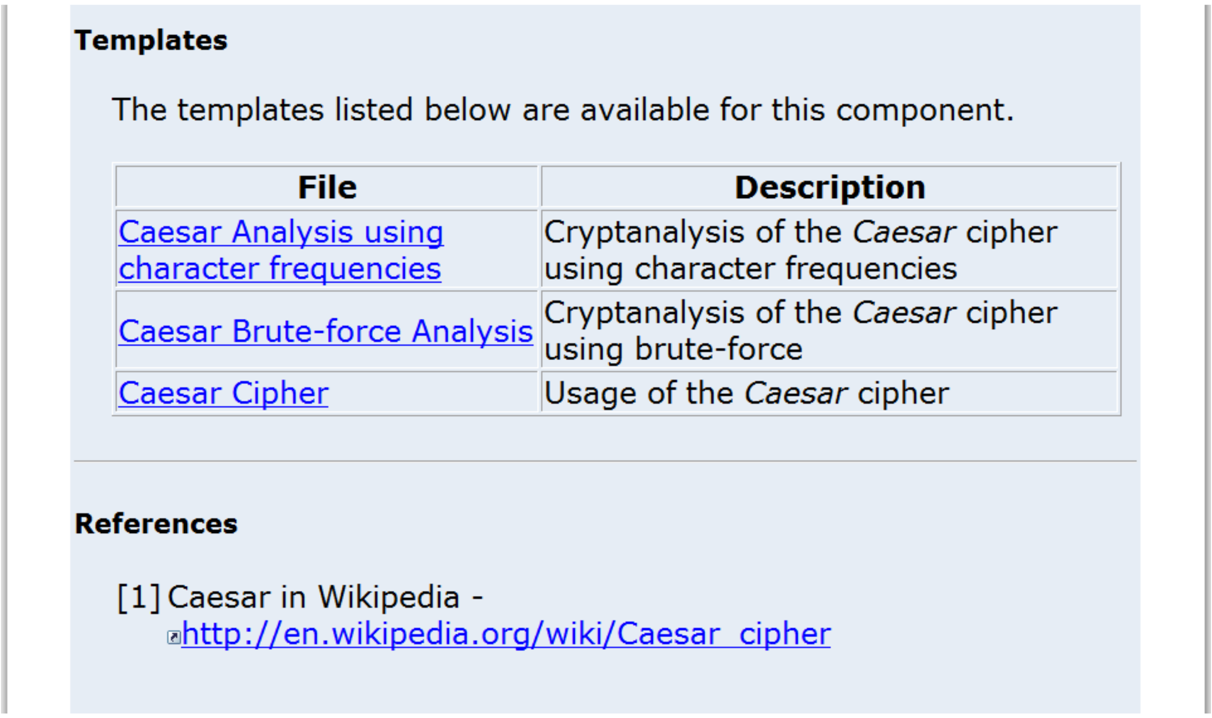
Caesar

Arno Wacker
Universität Kassel
arno.wacker@cryptool.org

Classic alphabet shift substitution cipher

Contents:

- Introduction
- Usage
- Connectors
- Settings
- Templates
- References



Templates

The templates listed below are available for this component.

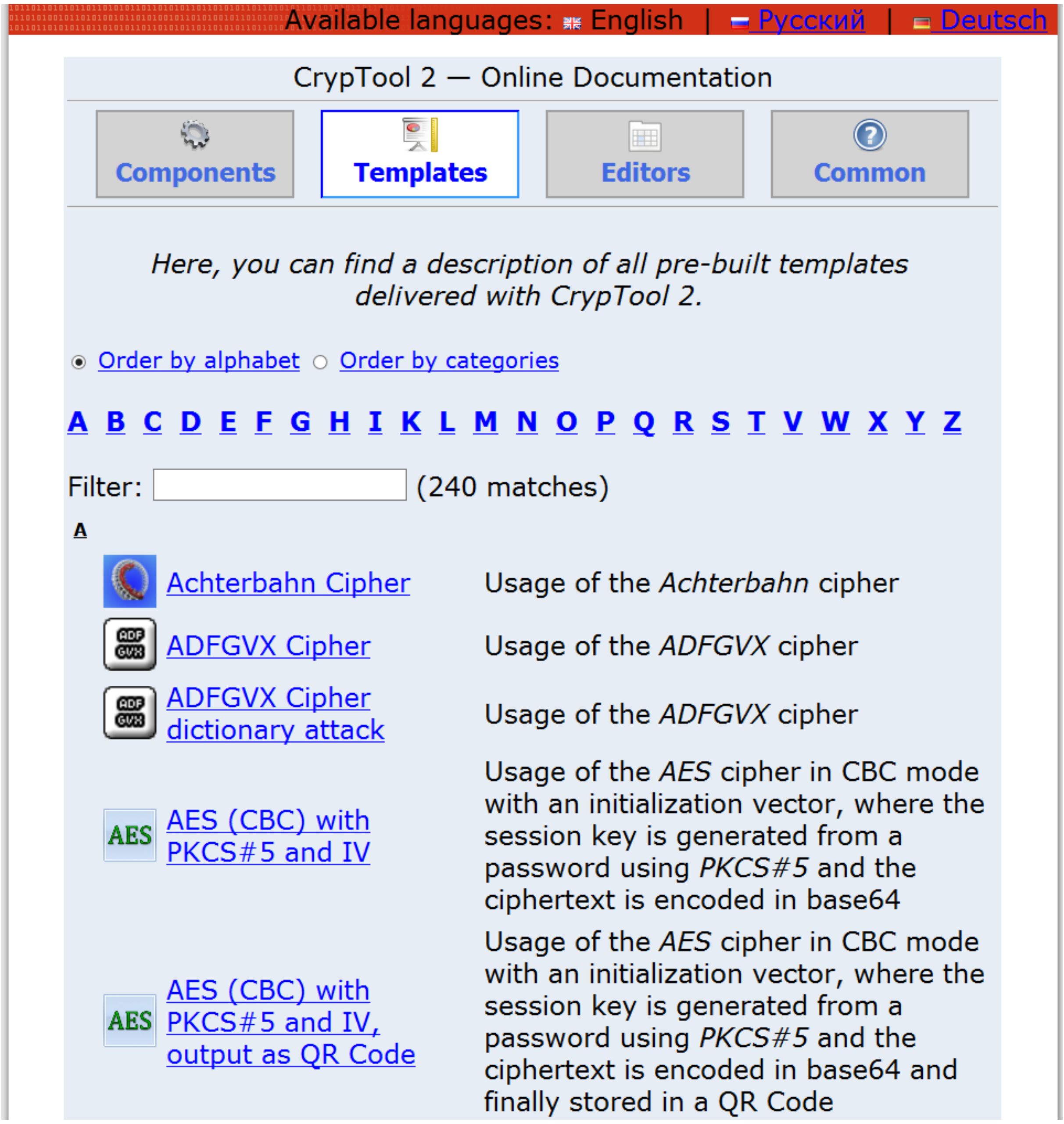
File	Description
Caesar Analysis using character frequencies	Cryptanalysis of the *Caesar* cipher using character frequencies
Caesar Brute-force Analysis	Cryptanalysis of the *Caesar* cipher using brute-force
Caesar Cipher	Usage of the *Caesar* cipher

References

[1] Caesar in Wikipedia - http://en.wikipedia.org/wiki/Caesar_cipher

Here is the beginning of the overview page of the **components** in the CT2 online help:

Here is the beginning of the overview page of the **templates** in the CT2 online help:


Available languages: English | Русский | Deutsch

CrypTool 2 — Online Documentation

Components | Templates | Editors | Common

Here, you can find a description of all pre-built templates delivered with CrypTool 2.

Order by alphabet Order by categories

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

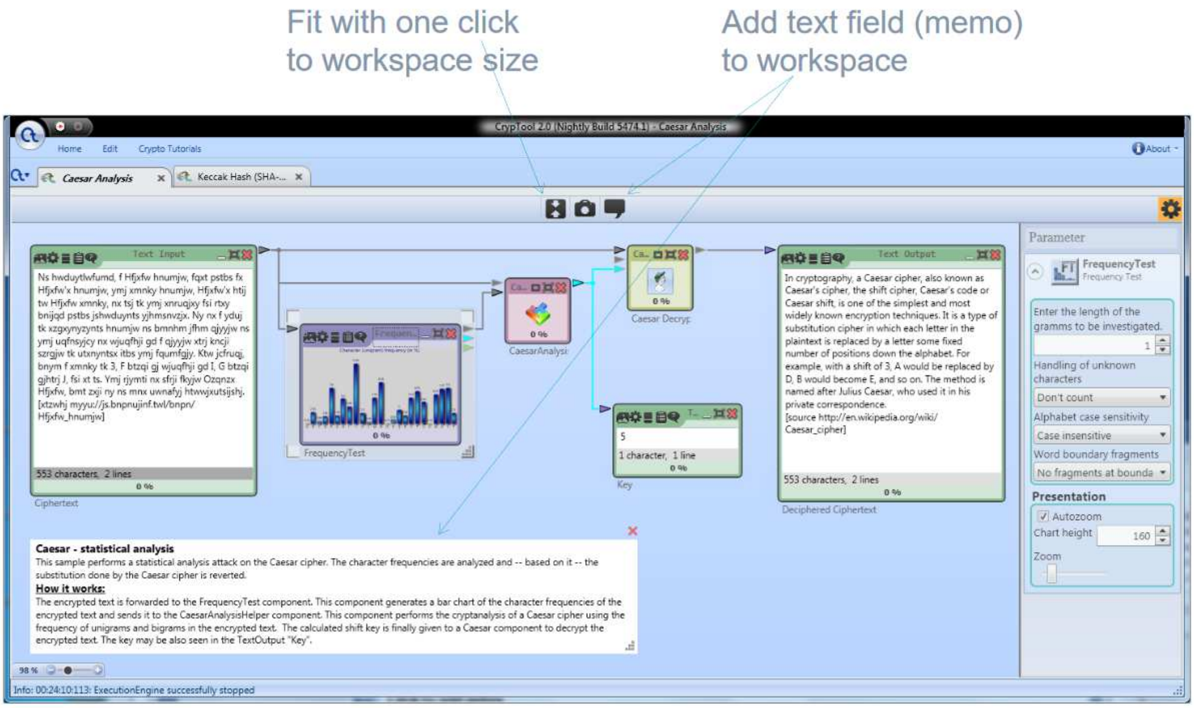
Filter: (240 matches)

A

Achterbahn Cipher	Usage of the *Achterbahn* cipher
ADFGVX Cipher	Usage of the *ADFGVX* cipher
ADFGVX Cipher dictionary attack	Usage of the *ADFGVX* cipher
AES (CBC) with PKCS#5 and IV	Usage of the *AES* cipher in CBC mode with an initialization vector, where the session key is generated from a password using *PKCS#5* and the ciphertext is encoded in base64
AES (CBC) with PKCS#5 and IV, output as QR Code	Usage of the *AES* cipher in CBC mode with an initialization vector, where the session key is generated from a password using *PKCS#5* and the ciphertext is encoded in base64 and finally stored in a QR Code

Remark: **Quickly adapt the layout to your needs**:

1) You can adapt the zoom level of the **workspace** using the left button in the top middle of the Workspace Manager (so with one click all elements optimally fill the given space).



2) Further hint for easy handling: Quickly adapt **the whole CT2 GUI** with the keyboard using F11 and F12 by fading-in or fading-out parts outside the actual workspace. So you can get the maximum space for the workspace within the CT2 window.

Remark: **Storage and persistence:**

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 given 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 or more languages at once.

Appendix 2: Task Overview

This document contains the following tasks:

- **Task 1: Caesar Decryption**
- **Task 2: Caesar Encryption**
- **Task 3: Caesar Cryptanalysis – Brute-Force**
- **Task 4: AES Decryption**
- **Task 5: AES Encryption**
- **Task 6: AES Cryptanalysis – Brute-Force**
- **Task 6a: AES Cryptanalysis – Experience different Search Times**
- **Task 7: RSA Decryption**
- **Task 8: RSA Encryption**
- **Task 9: RSA Cryptanalysis – Factorization of N**
- **Task 10: RSA Cryptanalysis – Attack on Common Prime Factors**
- **Task 11: Break Password Hashes – Preimage Attack using Dictionary**
- **Task 12: Test the Strength of Passwords**

Appendix 3: Links and References / Literature

You can directly download CrypTool 2 (CT2) from here – for this workshop, 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 2 Wiki: https://www.cryptool.org/trac/CrypTool2/

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

CrypTool Project at Wikipedia: https://en.wikipedia.org/wiki/CrypTool

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, 2018,
https://www.cryptool.org/en/ctp-documentation/ctbook

Some more reading for those interested in the cryptanalysis of classical ciphers with CT2:

Nils Kopal: Solving Classical Ciphers with CrypTool 2, 2018,
http://www.ep.liu.se/ecp/149/010/ecp18149010.pdf

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),
http://www.upress.uni-kassel.de/katalog/abstract.php?978-3-7376-0458-1

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

An overview about the whole CrypTool project including more modern algorithms (like post-quantum signatures in JCT):
http://fg-krypto.gi.de/fileadmin/fg-krypto/CrypTool-Project_Crypto_Day_Walldorf_2016-09_v09.pdf