COMPUTER SUBJECT:	ENCRYPTION/DECRYPTION
TYPE:	GROUP WORK EXERCISE/DISCUSSION
IDENTIFICATION:	CRYPTOOL No 1/MC
COPYRIGHT:	*Michael Claudius and Homayoon Fayes*
LEVEL:	EASY
DURATION:	1-2 hours
SIZE:	10 lines!! Answering a few questions
OBJECTIVE:	Introduction to classic and modern algorithms
REQUIREMENTS:	**Computer Network Ch.8-8.3**
COMMANDS:	

IDENTIFICATION: CRYPTOOL No 1/MC

CSF Chapter 1 Assignments

Mission

You are to get a general understanding of the basic symmetric encryption and decryption.

Purpose

The purpose of this assignment is to utilize Cryptool to get insight of the algorithms: Ceasar, DES, 3-DES and AES. Cryptool is very comprehensive SW-Tool with both visualizations and simulation of many algorithms (DES 3DES, AES, IDEA etc); and we just look into a few of them.

The following assignments can be solved in groups (1-2 persons).

Useful links

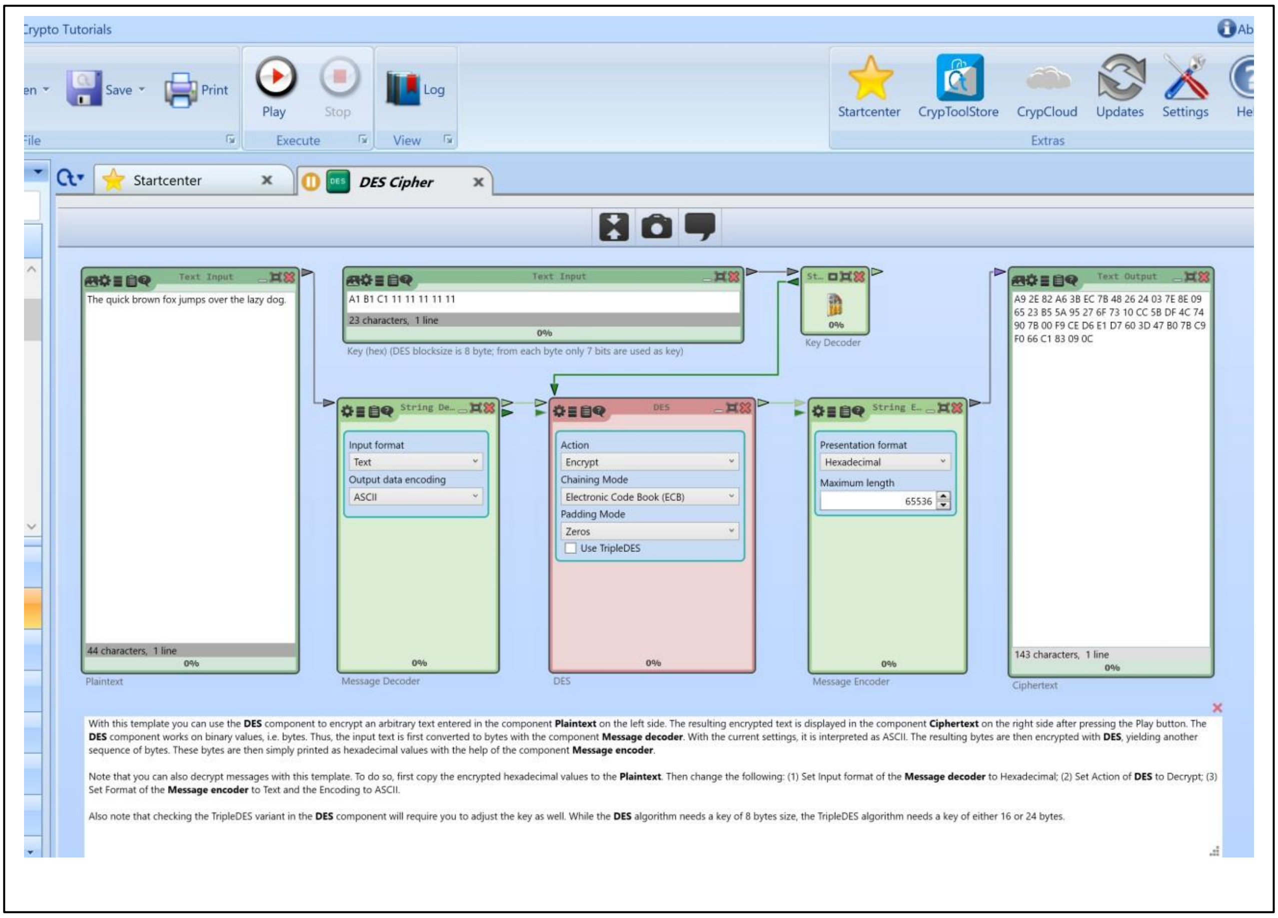
http://www.cryptool.org

1. Download and install Cryptool from http://www.cryptool.org/
   Choose the new stable version 2.1.
   Start the tool

2. You are to encrypt and decrypt a message with a symmetric encryption algorithm for example DES, AES, IDEA, 3DES etc.
   Encryption
   We start with DES to get a feeling of the tool.

   In Cryptool StartCenter use ”Templates”
   Select: Cryptographic -> Modern -> Symmetric -> DESCipher

   Then you will see the following DESCipher



The quick brown fox jumps over the lazy dog.

44 characters, 1 line

A1 B1 C1 11 11 11 11 11

23 characters, 1 line

Key (hex) (DES blocksize is 8 byte; from each byte only 7 bits are used as key)

Input format: Text

Output data encoding: ASCII

Action: Encrypt

Chaining Mode: Electronic Code Book (ECB)

Padding Mode: Zeros

□ Use TripleDES

Presentation format: Hexadecimal

Maximum length: 65536

A9 2E 82 A6 3B EC 7B 48 26 24 03 7E 8E 09 65 23 B5 5A 95 27 6F 73 10 CC 5B DF 4C 74 90 7B 00 F9 CE D6 E1 D7 60 3D 47 B0 7B C9 F0 66 C1 83 09 0C

143 characters, 1 line

With this template you can use the **DES** component to encrypt an arbitrary text entered in the component **Plaintext** on the left side. The resulting encrypted text is displayed in the component **Ciphertext** on the right side after pressing the Play button. The **DES** component works on binary values, i.e. bytes. Thus, the input text is first converted to bytes with the component **Message decoder**. With the current settings, it is interpreted as ASCII. The resulting bytes are then encrypted with **DES**, yielding another sequence of bytes. These bytes are then simply printed as hexadecimal values with the help of the component **Message encoder**.

Note that you can also decrypt messages with this template. To do so, first copy the encrypted hexadecimal values to the **Plaintext**. Then change the following: (1) Set Input format of the **Message decoder** to Hexadecimal; (2) Set Action of **DES** to Decrypt; (3) Set Format of the **Message encoder** to Text and the Encoding to ASCII.

Also note that checking the TripleDES variant in the **DES** component will require you to adjust the key as well. While the **DES** algorithm needs a key of 8 bytes size, the TripleDES algorithm needs a key of either 16 or 24 bytes.

Notice the 7 components and also the ”Play” and ”Stop” buttons at the top-bar.
Discuss shortly the role of the each component.

DesCipher starts with a standard text in the left Plaintext component. It can be changed later.

Click ”Play”

And you get the encrypted message in the right Ciphertext component in Hexadecimal.
Type and try another plaintext and enecrypt it.

<u>Decryption</u>
Now you try decryption.
Copy and paste the Hexadecimal encrypted text into Plaintext.
In MessageDecoder chage input format to Hexadeciamal
In DES change Action to Decryption
In MessageEncoder change PresentationFormat to Text.

Then start decryption i.e. press ”Play”.

So far so good !

3. Encrypt a text message with another symmetric encryption algorithm, and e-mail the encrypted text to one of the other students in this course. Supply her/him with the necessary information to decrypt it.

4. Use the Cryptool template DES Known-Plaintext Analysis to find the key used for encryption. The ciphertext is known and a word (”Encryption”) is known to occur in the plaintext. The KeySearcher component tries to find the DES key using bute-force to search a subset of the entire key space. Finally the full plaintext is shown.
   Change the known word to “Standard“. Run again.
   Then change the known word to “The “. Run again.
   Ups! Does not work, Can you fix the problem !!

5. Use Cryptool template DES BruteForce Analysis to make a brute force attack on a text encrypted by DES. The secret key used is 12 34 56 78 90 11 11 11.
   But You have seen some part of the key 12 34 56 78 90 11 ?? ??.
   Make the necessay changes in the template.
   How long time will it take you to compromise the complete key by using a brute force attack?
   Assume You have seen more part of the key 12 34 56 78 90 11 11 ??.
   How long time will it then take you to compromise the complete key by using a brute force attack?

6. Encrypt a message with DES and decrypt it with triple DES, and opposite encrypt a message with triple DES and decrypt it with DES.

<u>The next assignments are to be made at home !!</u>

7. Use to Vizualization templates to get more insight of of DES, 3-DES or AES.

8. Many classic encryption algorithms exist. One of them is the Caesar algorithm. Read about the Caesar encryption algorithm in “Cryptool – help”. Try to encrypt and to decrypt text messages with the Caesar algorithm.

9. The following message is encrypted with the Ceasar algorithm. Try to decrypt it - first manually and then automatically with one of the tools from Cryptool.

MbizDyyv

Drsc sc k dohd psvo, crygx sx ybnob dy rovz iye dy wkuo iyeb psbcd cdozc gsdr MbizDyyv.

1) Yxo drsxq iye mkx ny o.q. sc dy oxmbizd drsc psvo gsdr dro

Mkockb kvqybsdrw (fsk dro woxe "Mbizd \ Mvkccsmkv").

2) Dro locd yfobfsog klyed kvv pokdeboc yp MbizDyyv sc
yppobon li

dro cdkbdsxq zkqo yp dro Gsxnygc yxvsxo rovz grsmr myxdksxc vsxuc dy kvv bovofkxd pexmdsyxc.

Iye mkx mkvv ez dro cdkbdsxq zkqo fsk dro woxe "Rovz \ Cdkbdsxq zkqo" yb ecsxq dro cokbmr uoigybn

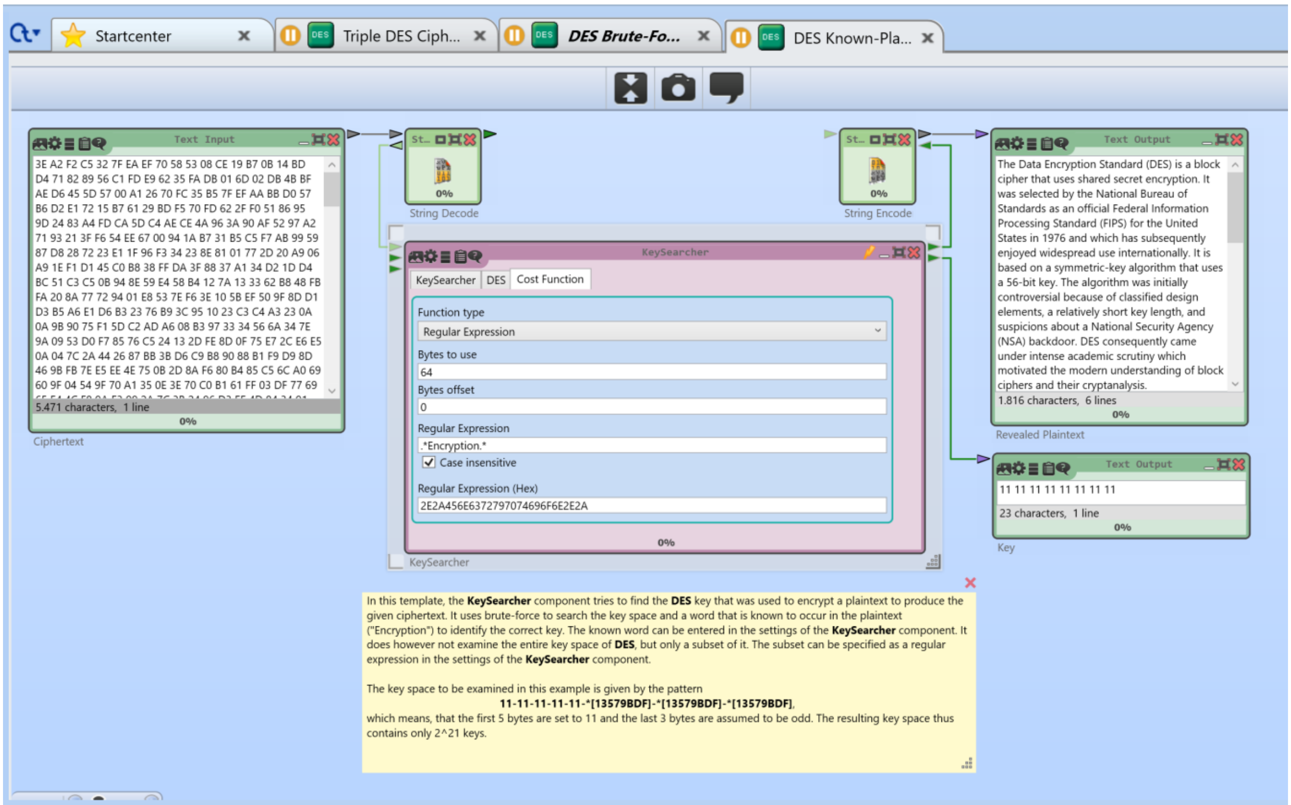
"Cdkbdsxq zkqo" gsdrsx dro sxnoh yp dro yxvsxo rovz.

3) Oczomskvvi dro ohkwzvoc (dedybskvc) zbyfsnon gsdrsx dro yxvsxo rovz wkuo sd okci pyb iye dy qod

ez dy czoon. Droco zkqoc mkx lo pyexn fsk dro woxe "Rovz \ Cmoxkbsyc".

10. Deprecated. Only easy in version 1.4.2.
    A DES encrypted message is placed in Exercise folder on teachers home page (Moodle) the filename is DES. You have been lucky, you have seen some part of the key 12 34 56 78 90 ?? ?? ??. How long time will it take you to compromise the complete key by using a brute force attack?

The next screenshots might be useful when solving some of the assignments



```
3E A2 F2 C5 32 7F EA EF 70 58 53 08 CE 19 B7 0B 14 BD
D4 71 82 89 56 C1 FD E9 62 35 FA DB 01 6D 02 DB 4B BF
AE D6 45 5D 57 00 A1 26 70 FC 35 B5 7F EF AA BB D0 57
B6 D2 E1 72 15 B7 61 29 BD F5 70 FD 62 2F F0 51 86 95
9D 24 83 A4 FD CA 5D C4 AE CE 4A 96 3A 90 AF 52 97 A2
71 93 21 3F F6 54 EE 67 00 94 1A B7 31 B5 C5 F7 AB 99 59
87 D8 28 72 23 E1 1F 96 F3 34 23 8E 81 01 77 2D 20 A9 06
A9 1E F1 D1 45 C0 B8 38 FF DA 3F 88 37 A1 34 D2 1D D4
BC 51 C3 C5 0B 94 8E 59 E4 58 B4 12 7A 13 33 62 B8 48 FB
FA 20 8A 77 72 94 01 E8 53 7E F6 3E 10 5B EF 50 9F 8D D1
D3 B5 A6 E1 D6 B3 23 76 B9 3C 95 10 23 C3 C4 A3 23 0A
0A 9B 90 75 F1 5D C2 AD A6 08 B3 97 33 34 56 6A 34 7E
9A 09 53 D0 F7 85 76 C5 24 13 2D FE 8D 0F 75 E7 2C E6 E5
0A 04 7C 2A 44 26 87 BB 3B D6 C9 B8 90 88 B1 F9 D9 8D
46 9B FB 7E E5 EE 4E 75 0B 2D 8A F6 80 B4 85 C5 6C A0 69
60 9F 04 54 9F 70 A1 35 0E 3E 70 C0 B1 61 FF 03 DF 77 69
```

5.471 characters, 1 line

Function type
Regular Expression

Bytes to use
64

Bytes offset
0

Regular Expression
.*Encryption.*

☑ Case insensitive

Regular Expression (Hex)
2E2A456E6372797074696F6E2E2A

The Data Encryption Standard (DES) is a block cipher that uses shared secret encryption. It was selected by the National Bureau of Standards as an official Federal Information Processing Standard (FIPS) for the United States in 1976 and which has subsequently enjoyed widespread use internationally. It is based on a symmetric-key algorithm that uses a 56-bit key. The algorithm was initially controversial because of classified design elements, a relatively short key length, and suspicions about a National Security Agency (NSA) backdoor. DES consequently came under intense academic scrutiny which motivated the modern understanding of block ciphers and their cryptanalysis.

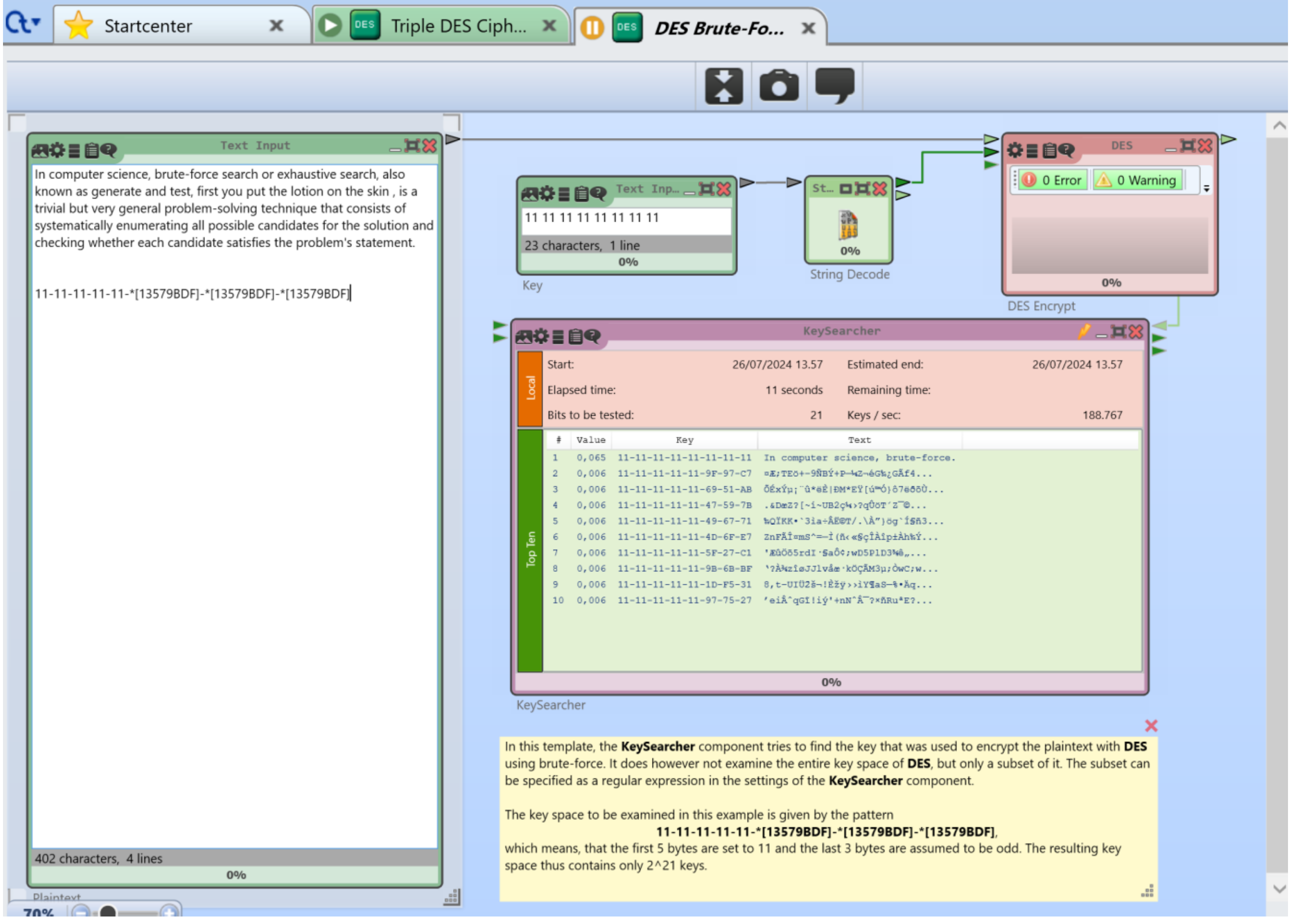
1.816 characters, 6 lines

11 11 11 11 11 11 11 11

23 characters, 1 line

In this template, the **KeySearcher** component tries to find the **DES** key that was used to encrypt a plaintext to produce the given ciphertext. It uses brute-force to search the key space and a word that is known to occur in the plaintext ("Encryption") to identify the correct key. The known word can be entered in the settings of the **KeySearcher** component. It does however not examine the entire key space of **DES**, but only a subset of it. The subset can be specified as a regular expression in the settings of the **KeySearcher** component.

The key space to be examined in this example is given by the pattern
11-11-11-11-11-*[13579BDF]-*[13579BDF]-*[13579BDF],
which means, that the first 5 bytes are set to 11 and the last 3 bytes are assumed to be odd. The resulting key space thus contains only 2^21 keys.



In computer science, brute-force search or exhaustive search, also known as generate and test, first you put the lotion on the skin , is a trivial but very general problem-solving technique that consists of systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem's statement.

11-11-11-11-11-*[13579BDF]-*[13579BDF]-*[13579BDF]

402 characters, 4 lines

11 11 11 11 11 11 11 11

23 characters, 1 line

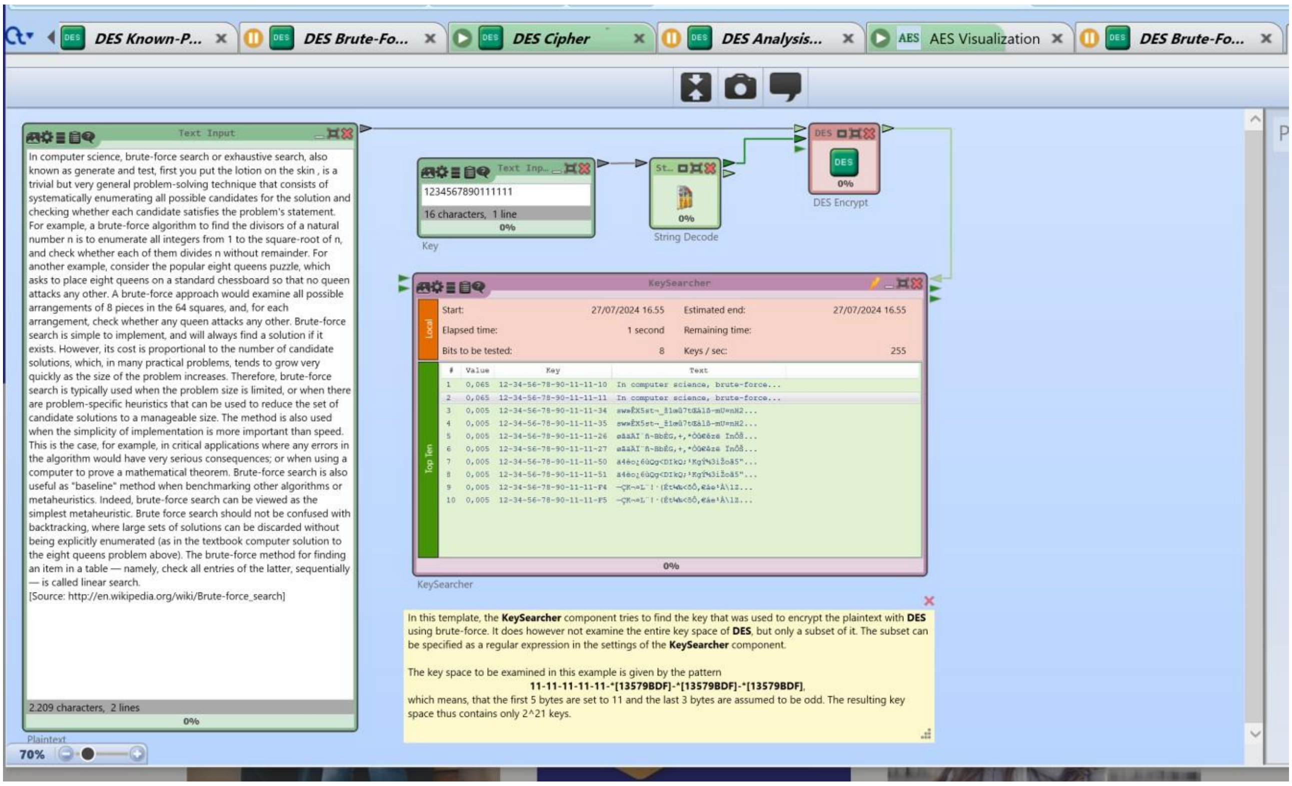
0 Error 0 Warning

Start:	26/07/2024 13.57	Estimated end:	26/07/2024 13.57
Elapsed time:	11 seconds	Remaining time:	
Bits to be tested:	21	Keys / sec:	188.767

#	Value	Key	Text
1	0,065	11-11-11-11-11-11-11-11	In computer science, brute-force.
2	0,006	11-11-11-11-11-9F-97-C7	¤Æ;TEö+-9ÑBÝ+P¬¼Z¬éG‰¿GÃf4...
3	0,006	11-11-11-11-11-69-51-AB	ÕÉxÝµ¡¨û*ëÈ¦ÐM*EŸ[ú™Ó}ô7ëõõÙ...
4	0,006	11-11-11-11-11-47-59-7B	.&DæZ?[~í~UB2ç¼›?qÛöT´Z¯©...
5	0,006	11-11-11-11-11-49-67-71	‰QÏKK•`3ìa÷ÅË©T/.\À”}ög`Í§ñ3...
6	0,006	11-11-11-11-11-4D-6F-E7	ZnFÅÎ¤mS^=—İ(ñ‹«§çÎÀîp±Àh‰Ý...
7	0,006	11-11-11-11-11-5F-27-C1	'ÆûÖõ5rdI·Šaô¢;wD5PlD3¾ê„...
8	0,006	11-11-11-11-11-9B-6B-BF	‘?À¾zíøJJlvåæ·kÖÇÃM3µ;ÒwC;w...
9	0,006	11-11-11-11-11-1D-F5-31	8,t-UIÜ2š¬!Èžý››ìY¶aS—%•Äq...
10	0,006	11-11-11-11-11-97-75-27	’eiÂ^qGI!iý'+nN^Â¯?×ñRuªE?...

In this template, the **KeySearcher** component tries to find the key that was used to encrypt the plaintext with **DES** using brute-force. It does however not examine the entire key space of **DES**, but only a subset of it. The subset can be specified as a regular expression in the settings of the **KeySearcher** component.

The key space to be examined in this example is given by the pattern
11-11-11-11-11-*[13579BDF]-*[13579BDF]-*[13579BDF],
which means, that the first 5 bytes are set to 11 and the last 3 bytes are assumed to be odd. The resulting key space thus contains only 2^21 keys.



In computer science, brute-force search or exhaustive search, also known as generate and test, first you put the lotion on the skin , is a trivial but very general problem-solving technique that consists of systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem's statement. For example, a brute-force algorithm to find the divisors of a natural number n is to enumerate all integers from 1 to the square-root of n, and check whether each of them divides n without remainder. For another example, consider the popular eight queens puzzle, which asks to place eight queens on a standard chessboard so that no queen attacks any other. A brute-force approach would examine all possible arrangements of 8 pieces in the 64 squares, and, for each arrangement, check whether any queen attacks any other. Brute-force search is simple to implement, and will always find a solution if it exists. However, its cost is proportional to the number of candidate solutions, which, in many practical problems, tends to grow very quickly as the size of the problem increases. Therefore, brute-force search is typically used when the problem size is limited, or when there are problem-specific heuristics that can be used to reduce the set of candidate solutions to a manageable size. The method is also used when the simplicity of implementation is more important than speed. This is the case, for example, in critical applications where any errors in the algorithm would have very serious consequences; or when using a computer to prove a mathematical theorem. Brute-force search is also useful as "baseline" method when benchmarking other algorithms or metaheuristics. Indeed, brute-force search can be viewed as the simplest metaheuristic. Brute force search should not be confused with backtracking, where large sets of solutions can be discarded without being explicitly enumerated (as in the textbook computer solution to the eight queens problem above). The brute-force method for finding an item in a table — namely, check all entries of the latter, sequentially — is called linear search.
[Source: http://en.wikipedia.org/wiki/Brute-force_search]

2.209 characters, 2 lines

1234567890111111

16 characters, 1 line

Start:	27/07/2024 16.55	Estimated end:	27/07/2024 16.55
Elapsed time:	1 second	Remaining time:	
Bits to be tested:	8	Keys / sec:	255

```
#   Value  Key                       Text
1   0,065  12-34-56-78-90-11-11-10   In computer science, brute-force...
2   0,065  12-34-56-78-90-11-11-11   In computer science, brute-force...
3   0,005  12-34-56-78-90-11-11-34   swwÊX5st¬_ž1œû7tŒâ1ß-mU=nH2...
4   0,005  12-34-56-78-90-11-11-35   swwÊX5st¬_ž1œû7tŒâ1ß-mU=nH2...
5   0,005  12-34-56-78-90-11-11-26   øââÃÏ¨ñ-BbÈG,+,*ÒûŒâzø ÎnÕ§...
6   0,005  12-34-56-78-90-11-11-27   øââÃÏ¨ñ-BbÈG,+,*ÒûŒâzø ÎnÕ§...
7   0,005  12-34-56-78-90-11-11-50   â4êo¿6ùQg<DIkQ;¹KgÝ¾3iŽoâ5"...
8   0,005  12-34-56-78-90-11-11-51   â4êo¿6ùQg<DIkQ;¹KgÝ¾3iŽoâ5"...
9   0,005  12-34-56-78-90-11-11-F4   ¬ÇK¬»L¨!·(Ét¼k<ðÔ,€áe¹Å\1Z...
10  0,005  12-34-56-78-90-11-11-F5   ¬ÇK¬»L¨!·(Ét¼k<ðÔ,€áe¹Å\1Z...
```

0%

In this template, the **KeySearcher** component tries to find the key that was used to encrypt the plaintext with **DES** using brute-force. It does however not examine the entire key space of **DES**, but only a subset of it. The subset can be specified as a regular expression in the settings of the **KeySearcher** component.

The key space to be examined in this example is given by the pattern
11-11-11-11-11-*[13579BDF]-*[13579BDF]-*[13579BDF],
which means, that the first 5 bytes are set to 11 and the last 3 bytes are assumed to be odd. The resulting key space thus contains only 2^21 keys.