Cryptanalysis of Homophonic Substitution Ciphers Using Simulated Annealing with Fixed Temperature

Nils Kopal
nils.kopal@cryptool.org
University of Siegen

Abstract

This paper describes the current progress of our research in the area of breaking homophonic substitution ciphers. Furthermore, it presents the state-of-the-art of cryptanalyzing this kind of cipher. There is a huge gap between the success rate of methods published in according research papers and the success rate of already available tools on the Internet. This paper also presents a small general taxonomy of monoalphabetic substitution ciphers. Finally, it shows how we broke different homophonic substitution ciphers in an automatic as well as in a semi-automatic way.

1 Introduction

Homophonic substitution ciphers are, when used with a considerably high number of homophones, hard to break, even today. Since they were used in many historical correspondences as the cipher of first choice, many of these historical texts are still unbroken. For example, the DECODE database (Megyesi et al., 2017), which is a collection of historical encrypted books and encrypted documents, contains about 600 encrypted documents of which 22 also have an uploaded solution. Since the kind of cipher is often unknown, only an estimation of the number of homophonic encrypted texts can be made. By assuming that texts consisting of more than 26 (or 27 in Spanish) different ciphertext symbols (letters, number groups, or arbitrary symbols) are homophonically substituted, there are about 480 homophonically encrypted documents in the DECODE database.

Efficient and easy-to-use tools that help researchers cryptanalyzing the texts and revealing their contents are needed. Within the DECRYPT project, one goal is to research and develop such tools. For that purpose, we created an analyzer and integrated it in the open-source software CrypTool 2 (Kopal, 2018). The analyzer is based on simulated annealing with a fixed temperature and allows the user of CrypTool 2 (CT2) to break homophonic substitution ciphers in a semi-automatic as well as in a full-automatic way. Using the analyzer, we were able to break all (already solved) ciphers available on the wiki of the (ZKC, 2019). Additionally, we tested our solver with ciphers from (MTC3, 2018) (i.e. the Spanish Strip Cipher challenges and the Zodiac cipher challenge) and were able to successfully break these as well.

The rest of this paper is structured as follows: Section 2 presents a small taxonomy of substitution ciphers. Section 3 discusses the related work in the area of analyzing homophonic substitution ciphers. Section 4 briefly presents CT2, a tool which is the framework in which we integrate our research results. Section 5 presents our own approach for breaking homophonic substitution ciphers using CT2. Section 6 shows an example of a real-world cipher (Zodiac-408) broken in the full-automatic mode of our analyzer. Finally, Section 7 gives a brief outlook what is planned within the DECRYPT project with regards to the analysis of historical ciphers.

2 Substitution Ciphers Taxonomy

Substitution ciphers in general replace plaintext letters defined by a plaintext alphabet with ciphertext letters defined by a ciphertext alphabet. Substitution ciphers are divided in two general types: (1) monoalphabetic substitution ciphers and (2) polyalphabetic substitution ciphers. With monoalphabetic substitution ciphers, the cipher only uses a single ciphertext alphabet. With polyalphabetic substitution ciphers, the cipher uses more than one ciphertext alphabet. An example for the polyalphabetic substitution cipher is the Vigenère cipher. As we focus on homophonic ciphers in this paper, we do not further specify or analyze polyalphabetic ciphers from now on.

Monoalphabetic substitution ciphers can be furthermore divided into different classes: (1) simple monoalphabetic substitution ciphers (the Caesar cipher is a very simple variant of it; from now on we always consider the general case of monoalphabetic substitution ciphers), (2) homophonic substitution ciphers, (3) nomenclatures, and (4) code books.

Simple Monoalphabetic Substitution (maS): A simple monoalphabetic substitution cipher replaces each plaintext letter using always a single and always the same ciphertext letter. Plaintext letters can be Latin letters but also any kind of symbols. The keyspace of the simple monoalphabetic substitution cipher is, having 26 plaintext/-ciphertext alphabet letters, $26! \approx 2^{88}$. Despite the huge key space, simple monoalphabetic substitution ciphers can easily be broken, also by hand. As the letter distribution of the ciphertext is the same as the one of the corresponding plaintext, language statistics are used to break the cipher. Computerized methods to break simple monoalphabetic substitution ciphers already exist. CrypTool 2 contains powerful algorithms, that break also short (less than 60 letters) simple monoalphabetic substitution ciphers in milliseconds. (Kopal, 2018)

Homophonic Substitution: A homophonic substitution cipher tries to eliminate the aforementioned possibility to analyze the ciphertext using simple language statistics. To do so, it flattens the frequencies of single letters, thus, in the perfect case, the ciphertext letters are uniformly distributed. For example, instead of encrypting the letter 'E' only with one ciphertext letter, it can now be encrypted using one of several different "homophones", e.g. '01', '02', '03', '04', '05'. Then the ciphertext consists of different pairs of digits – this method was often used in history, i.e. in letters kept in the Vatican's secret archive (Archivio Segreto Vaticano, 2019) or in messages of the Spanish Civil War encrypted with the Spanish Strip Cipher (Soler Fuensanta and López-Brea Espiau, 2007). The keyspace size of a homophonic cipher can be calculated by 26^n where n is the number of homophones. For example, a homophonic encrypted text having only 52 homophones has a keyspace size of $26^{52} \approx 2^{244}$, where each homophone may be mapped to one of the 26 letters of the Latin alphabet.

Simple monoalphabetic substitution ciphers as well as homophonic substitution ciphers can be extended to *polygraphic ciphers* like Playfair, where instead of single letters group of letters are encrypted. This further increases the keyspaces of the ciphers. For example, a simple monoalphabetic substitution cipher that works on 2 instead of 1 letters has a total of $26 \cdot 26 = 676$ plaintext and ciphertext "symbols" in their corresponding alphabets. Each symbol consists of two Latin letters, eg 'HE'. The overall keyspace of such a cipher would be $676! \approx 2^{5375}$. Despite this number sounds incredible huge, most of the plaintext alphabet symbols would not be used in practice, since many combinations are seldom or never used in a real language, e.g. 'WX'. This ciphers can still be attacked using language statistics, but are harder to break than their simple cases.

It is also possible to disrupt frequency analysis by adding nulls to a message. Nulls are ciphertext symbols which are added to veil the meaning of the message. They just have to be deleted before decrypting. For example, every second letter in a ciphertext could be a null. If the attacker knows this, the cipher isn't more secure than without, but harder to handle for authorized users.

Nomenclature: A nomenclature cipher is a kind of extension of a simple monoalphabetic or of a homophonic substitution cipher. Additionally to substituting single letters, a nomenclature substitutes groups of letters, like the polygraphic ciphers. On top of that, a nomenclature also substitutes complete words. Nomenclatures are usually built by creating a ciphertext alphabet that consists of groups of letters, often of different lengths. For example, names of persons, objects, or places are substituted using special number groups, e.g. "Pope" $\rightarrow$ 34521, "Rome" $\rightarrow$ 82355, etc. Often, cryptanalysts are able to break nomenclatures partially, but these decryptions have "holes" of such still encrypted words. These holes can sometimes be "filled" using the context of the document, or having other broken documents encrypted with the same key, or even having the original key, e.g. obtained from an archive. Nomenclatures were often used in history. An example for such a nomenclature, which was already broken in its time, is the one used by Mary, Queen of Scots, to communicate with Anthony Babington to plan a complot against Elizabeth I of England. (Kahn, 1996)

Code Book: A code book consists of code words for nearly all words of a language. Both, sender and receiver need the same code book. The sender encrypts the plaintext by replacing each word with the appropriate code word, e.g. "We" $\rightarrow$ 46621, "need" $\rightarrow$ 12315, "supplies" $\rightarrow$ 75123. Often, the corresponding numbers were super-encrypted with a second key changing the numbers using special rules. Code book ciphers are, without being in the possession of the used code book, very hard to break. Things could become easier, if the original code book is sorted based on rules: If two codewords begin with the same letter, e.g. a "T" ("transport" $\rightarrow$ 5100 and "tools" $\rightarrow$ 5200), then a cryptanalyst could assume that a codeword 5150 is "between" the words "transport" and "tools". In history, code books often based on older ones. This made it easier to break a new cipher if the older code book is known. Already broken code words can be put into a list of known code words. Thus, the next time a cipher has to be analyzed, this list could also be used. For further information on breaking code books, see the excellent discussion in (Lasry, 2018b).

3 Related Work

We made extensive investigations concerning research papers and tools about cryptanalysis of homophonic substitutions. In general, there are not many research papers dealing directly with the cryptanalysis of homophonic substitution ciphers. Also, there are not many tools available for breaking homophonic substitution ciphers. Our related work investigation also shows that there is a huge gap: The success rate of methods in the published papers is much worse than the success rate of actual existing tools available on the Internet.

3.1 Research Papers

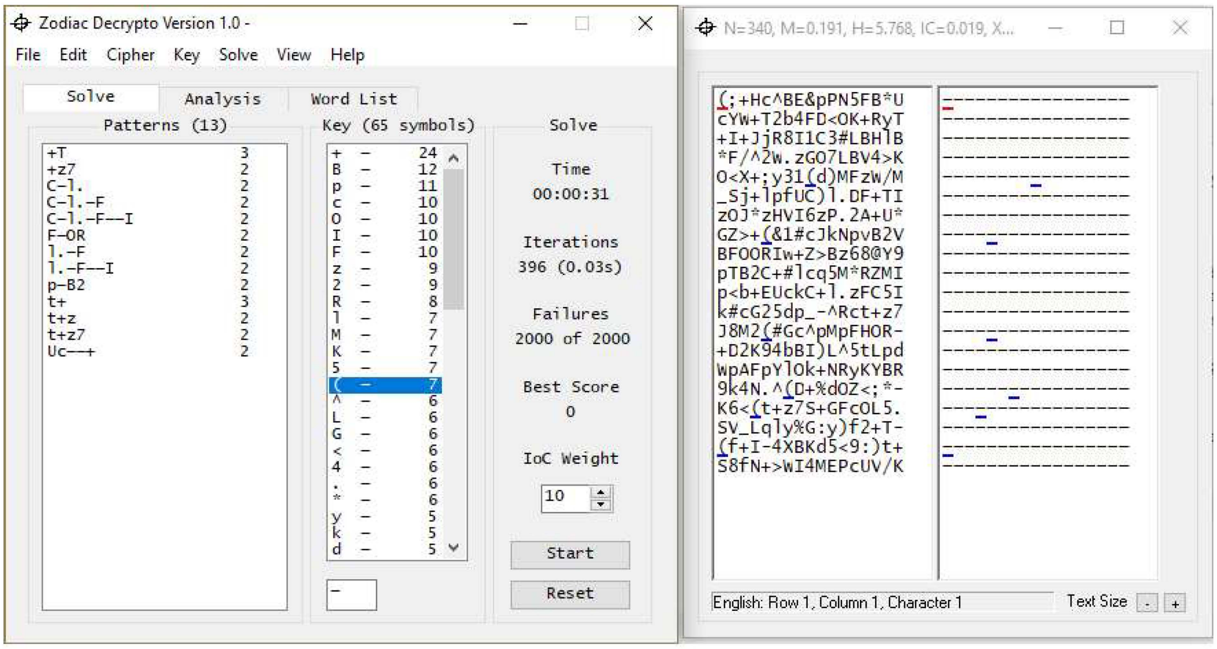
We found different research papers dealing with cryptanalysis of homophonic substitution ciphers:

The first paper (Dhavare et al., 2013) is called "Efficient Cryptanalysis of Homophonic Substitution Ciphers". As baseline for their algorithm the authors use a nested a hill climbing approach. The goal of their research was to solve Zodiac-340, a message sent by the infamous Zodiac killer, which is possibly encrypted using a homophonic substitution cipher. Despite their approach did not decrypt the message, they improved the state-of-the-art of cryptanalyzing homophonic substitution ciphers. Having a ciphertext with a length of 1 000 letters, their algorithm has a success rate of about 100% when there are 28 homophones, about 78% with 35 homophones, about 78% with 45 homophones, about 40% with 55 homophones, and about 45% with 65 homophones (see Figure 6 of (Dhavare et al., 2013)).

The second paper (Campos et al., 2013) is called "Genetic Algorithms and Mathematical Programming to Crack the Spanish Strip Cipher". The Spanish strip cipher is a homophonic cipher used during the Spanish Civil War. It encrypts a plaintext using 3 to 5 different homophones per plaintext letter resulting in a total number of homophones between $27 \cdot 3 = 81$ (since the Spanish alphabet has 27 different letters) and 100. The maximum number is 100, since there are 100 possible homophones $(00, 01, 02, ..., 99)$. The authors present two different methods for analyzing the cipher: (1) a genetic algorithm and (2) mathematical programming. The genetic algorithm performs best in their analyses. To further improve the genetic algorithm, they added a dictionary search that searches for already correct words in their solutions which then are used for new generations in the genetic algorithm. The authors do not present an extensive evaluation of their results nor did we found their code or tools, thus, it is impossible for us to give a success rate. Nevertheless, the authors present an execution time analysis that the genetic algorithm needs about 7 minutes to find a solution.

The third paper (Sanguino et al., 2016) analyzed the Spanish Strip Cipher, but used a different approach compared to Campos et. al. Their attack is based on a three phase attack which consists of (1) homophones-table analysis, (2) letter-frequency analysis, and (3) a dictionary search. Their method is able to successfully analyze and decrypt texts encrypted with the Spanish Strip cipher having a length of 201 letters with a success rate of $\approx 90\%$ in an average of about 2 minutes.

The fourth paper (Oranchak, 2008) presents a method for the decryption of Zodiac-408. His method is based on an evolutionary algorithm and uses a word dictionary. He uses a genetic algorithm to optimize word mappings onto the ciphertext that don't conflict each other by overlapping. Therefore, he uses a small part of the ciphertext that covers over 90% of all homophones of the 408-cipher. He maps words from the dictionary onto that part and then analyzes the rest of the ci-


Zodiac Decrypto Version 1.0 -
File Edit Cipher Key Solve View Help
Solve
Analysis
Word List
Start
Reset
Text Size - +

```
Patterns (13)
+T            3
+z7           2
C-l.          2
C-l.-F        2
C-l.-F--I     2
F-OR          2
l.-F          2
l.-F--I       2
p-B2          2
t+            3
t+z           2
t+z7          2
Uc--+         2
```

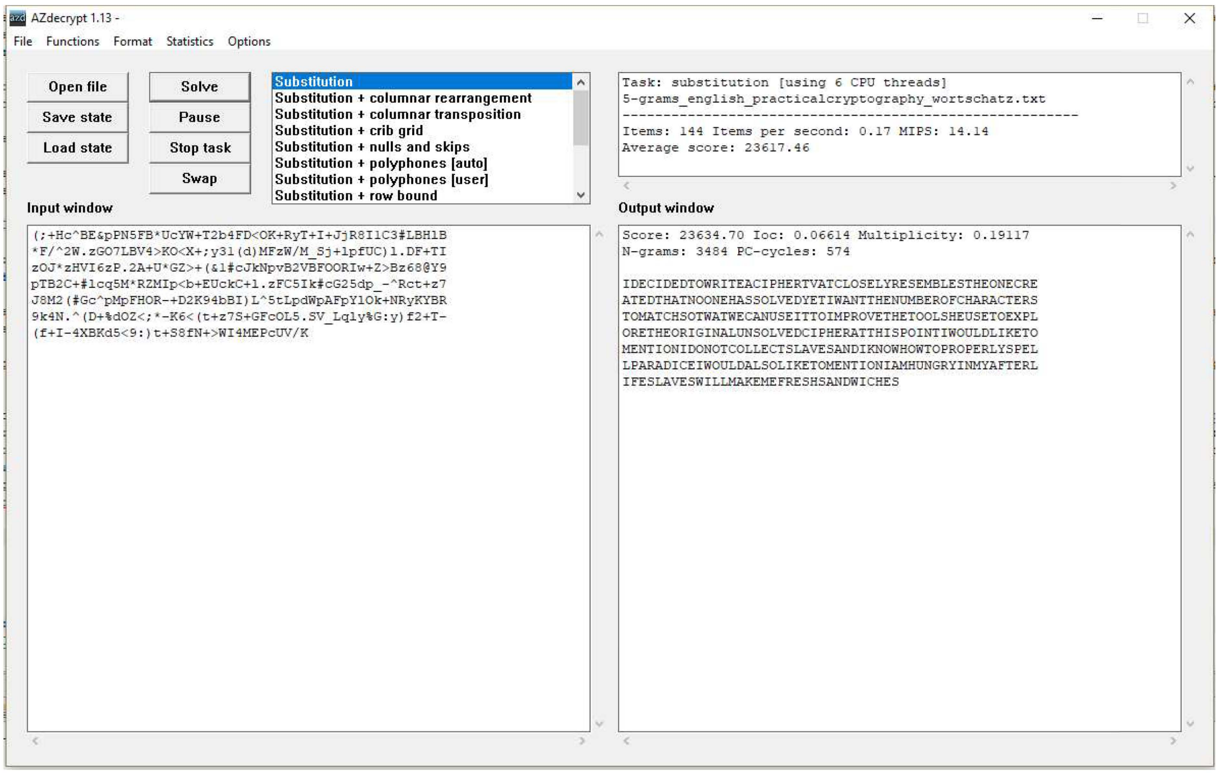
```
Key (65 symbols)
+  -  24
B  -  12
p  -  11
c  -  10
O  -  10
I  -  10
F  -  10
z  -   9
2  -   9
R  -   8
l  -   7
M  -   7
K  -   7
5  -   7
(  -   7
^  -   6
L  -   6
G  -   6
<  -   6
4  -   6
.  -   6
*  -   6
y  -   5
k  -   5
d  -   5
```

```
Solve

Time
00:00:31

Iterations
396 (0.03s)

Failures
2000 of 2000

Best Score
0

IoC Weight
10
```

```
N=340, M=0.191, H=5.768, IC=0.019, X...
```

```
(;+Hc^BE&pPN5FB*U    -----------------
cYW+T2b4FD<OK+RyT    -----------------
+I+JjR8I1C3#LBHlB    -----------------
*F/^2W.zGO7LBV4>K    -----------------
O<X+;y31(d)MFzW/M    -----------------
_Sj+lpfUC)l.DF+TI    -----------------
zOJ*zHVI6zP.2A+U*    -----------------
GZ>+(&1#cJkNpvB2V    -----------------
BFOORIw+Z>Bz68@Y9    -----------------
pTB2C+#lcq5M*RZMI    -----------------
p<b+EUckC+l.zFC5I    -----------------
k#cG25dp_-^Rct+z7    -----------------
J8M2(#Gc^pMpFHOR-    -----------------
+D2K94bBI)L^5tLpd    -----------------
WpAFpYlOk+NRyKYBR    -----------------
9k4N.^(D+%dOZ<;*-    -----------------
K6<(t+z7S+GFcOL5.    -----------------
SV_Lqly%G:y)f2+T-    -----------------
(f+I-4XBKd5<9:)t+    -----------------
S8fN+>WI4MEPcUV/K    -----------------

English: Row 1, Column 1, Character 1
```

Figure 1: Screenshot of the Tool “Zodiac Decrypto”


AZdecrypt 1.13 -
File Functions Format Statistics Options
Open file
Save state
Load state
Solve
Pause
Stop task
Swap
Substitution
Substitution + columnar rearrangement
Substitution + columnar transposition
Substitution + crib grid
Substitution + nulls and skips
Substitution + polyphones [auto]
Substitution + polyphones [user]
Substitution + row bound
Input window
Output window

```
Task: substitution [using 6 CPU threads]
5-grams_english_practicalcryptography_wortschatz.txt
------------------------------------------------------------
Items: 144 Items per second: 0.17 MIPS: 14.14
Average score: 23617.46
```

```
(;+Hc^BE&pPN5FB*UcYW+T2b4FD<OK+RyT+I+JjR8I1C3#LBH1B
*F/^2W.zGO7LBV4>KO<X+;y31(d)MFzW/M_Sj+lpfUC)l.DF+TI
zOJ*zHVI6zP.2A+U*GZ>+(&l#cJkNpvB2VBFOORIw+Z>Bz68@Y9
pTB2C+#lcq5M*RZMIp<b+EUckC+l.zFC5Ik#cG25dp_-^Rct+z7
J8M2(#Gc^pMpFHOR-+D2K94bBI)L^5tLpdWpAFpYlOk+NRyKYBR
9k4N.^(D+%dOZ<;*-K6<(t+z7S+GFcOL5.SV_Lqly%G:y)f2+T-
(f+I-4XBKd5<9:)t+S8fN+>WI4MEPcUV/K
```

```
Score: 23634.70 Ioc: 0.06614 Multiplicity: 0.19117
N-grams: 3484 PC-cycles: 574

IDECIDEDTOWRITEACIPHERTVATCLOSELYRESEMBLESTHEONECRE
ATEDTHATNOONEHASSOLVEDYETIWANTTHENUMBEROFCHARACTERS
TOMATCHSOTWATWECANUSEITTOIMPROVETHETOOLSHEUSETOEXPL
ORETHEORIGINALUNSOLVEDCIPHERATTHISPOINTIWOULDLIKETO
MENTIONIDONOTCOLLECTSLAVESANDIKNOWHOWTOPROPERLYSPEL
LPARADICEIWOULDALSOLIKETOMENTIONIAMHUNGRYINMYAFTERL
IFESLAVESWILLMAKEMEFRESHSANDWICHES
```

Figure 2: Screenshot of the Tool “AZdecrypt”

phertext with respect to minimizing the collision of words. Having 1 000 generations in the genetic algorithm and a minimum dictionary size of about 1 300 words, he is able to decrypt the 408-cipher by 60%. Having a dictionary with only about 850 words, he is able to decrypt the 408-cipher using 600 generations by 100%.

The fifth paper (Ravi and Knight, 2011) presents a Bayesian approach for deciphering complex substitution ciphers. The authors combine n-gram language models and word dictionaries. They were able to decipher Zodiac-408 by 97.8%. They claim, that their solution is the first method that was able to automatically decipher the Zodiac-408 cipher.

The sixth paper (King and Bahler, 1993) presents an algorithm for breaking a special case of homophonic ciphers: sequential homophonic ciphers. With this type, the homophones are not selected randomly but sequentially. This means, if 'E' has homophones '01','15', and '25', a letter 'E' is first encrypted using '01', then '15', and then '25'. After that, the sequence is repeated. Their algorithm reduces the homophonic substitution to a simple monoalphabetic substitution by finding those sequences. Remarkable is, that finding the sequences does not need any language frequencies.

Other publications we found are dealing with (putative) homophonic encrypted ciphertexts, including Zodiac ciphers. An example is the master thesis "Analysis of the Zodiac-340 Cipher" (Dao, 2008) or the paper "How I reconstructed a Spanish cipher from 1591" (Tomokiyo, 2018) in which Tomokiyo analyzed and decrypted an encrypted letter written by Alessandro Farnese, Duke of Parma and sent to Filippo Sega, Bishop of Piacenza. Tomokiyo decrypted the letter by hand. The ciphertext was written using a simple substitution (with some homophones) and vowel indicator symbols.

Clearly, there are more scientific reports about successfully decryptions of homophonic ciphers (by hand), but as this paper focuses on computerized cryptanalysis methods, these reports are not relevant for this paper.

All mentioned papers make use of n-gram statistics wit $n > 1$ as the homophonic substitution only flattens the 1-grams.

3.2 Tools

Additionally to searching for research papers dealing with cryptanalysis of homophonic substitution ciphers, we also investigated different tools available on the Internet. As mentioned before, the success rate of finding the correct decryption of homophonic encrypted texts that these tools offer surpass the success rates of the aforementioned methods in the research papers. Nevertheless, the authors of the tools never published the algorithms nor a scientific analysis dealing with their methods, and only rarely their code.

Many tools are listed on the Zodiackillerciphers website (ZKC, 2019). The purpose of the tools on the list is to support the decryption of the messages of the Zodiac killer. However, the forum of the website also contains useful information for cryptanalyzing homophonic ciphers in general.

In the following, we show the two tools which were cited most often on the Zodiackillerciphers website. One of the two outperforms the other concerning success rate and analysis speed.

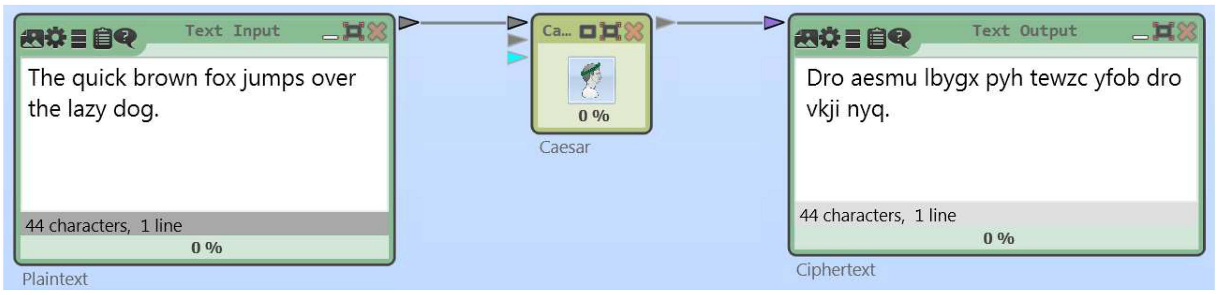
The first tool is called *ZKDecrypto*[1] which stands for "Zodiac Killer Decrypto" and was written by Hopper in 2008. Figure 1 contains a screenshot of the tool which consists of two separate windows. The first window (left) is used to set parameters and load a ciphertext (from text file), while the second window (right) shows it. The tool also allows to select a homophone and see where else it appears in the ciphertext. To test the tool, we used a ciphertext[2] from the Zodiackillerciphers website. According to the Zodiackillerciphers website the "program's original purpose was to attempt to solve the Zodiac killer's unsolved 340-length cipher ... The program has since been advanced to being able to solve general-case homophonic and monophonic ciphers". Unfortunately, we were not able to break the test cipher with ZKDecrypto.

The second tool "AZdecrypt"[3] was written by Eycke since 2016. The newest version (1.13) was

[1]ZKDecrypto can be downloaded from: `https://code.google.com/archive/p/zkdecrypto/`

[2]We used number 35 of the list obtained from `http://zodiackillerciphers.com/wiki/index.php?title=Cipher_comparisons`
The test text has a length of 340 letters and consists of 65 homophones.

[3]AZdecrypt can be downloaded from: `http://www.zodiackillersite.com/viewtopic.php?f=81&t=3198`. There is also a web-based version of this solver.



The quick brown fox jumps over the lazy dog.

44 characters, 1 line

Dro aesmu lbygx pyh tewzc yfob dro vkji nyq.

44 characters, 1 line

Figure 3: A CrypTool 2 Workspace with a Caesar Cipher

published together with the source code (written in FreeBASIC) in January 2019 in a forum thread of the zodiackillerciphers forum. Despite not publishing a scientific publication, Eycke gives insights in his analyzer in that forum. Figure 2 shows a screenshot of the analyzer. Similar to ZKDecrypto, AZdecrypt presents the ciphertext/-plaintext next to each other. At the top, it shows additional information about the currently performed analysis. We used the same ciphertext[2] to test AZdecrypt as we did for ZKDecrypto. After hitting the "Solve" button, the tool presented the valid decryption in less than a second. To our best knowledge, this tool is the most powerful homophonic substitution analyzer publicly available. According to the author, it uses a simulated annealing approach and a combined fitness function using pentragram statistics normalized with the Index of Coincidence.

4 CrypTool 2

CrypTool 2 (CT2) (Kopal et al., 2014) is an open-source e-learning tool aiming to help pupils, students, and crypto-enthusiasts to learn cryptology. Lately, CT2 was enhanced to contain the state-of-the-art cryptanalysis tools for analyzing classic and historic ciphers. CT2 is part of the CrypTool project that was initiated in 1998.

CT2 realizes the visual programming concept, displaying always the process workflow. One of the easiest workflows within CT2 is the Caesar cipher, shown in Figure 3. In the *TextInput* component on the left the user can enter plaintext, the *Caesar cipher* component in the middle is the processor, and the *TextOutput* component at the right displays the encrypted text. The connectors are the small colored triangles. The connections are the lines between the triangles. The color of the connectors and connections indicate the data types (here text). To execute the graphical program, the user has to hit the *Play* button in the top menu of CT2. Currently, CT2 contains more than 150 different components for encryption, decryption, cryptanalysis, etc. For example, there is a component which can add space between words at the appropriate places after decrypting a ciphertext which didn't contain any blanks.

In the DECRYPT project, CT2 is the main software for collecting and implementing cryptanalysis methods and algorithms suitable to break historical ciphers. The goal is to extend CT2 with efficient methods and algorithms for analyzing homophonic substitution ciphers. We started our ongoing research in that area in Dec 2018.

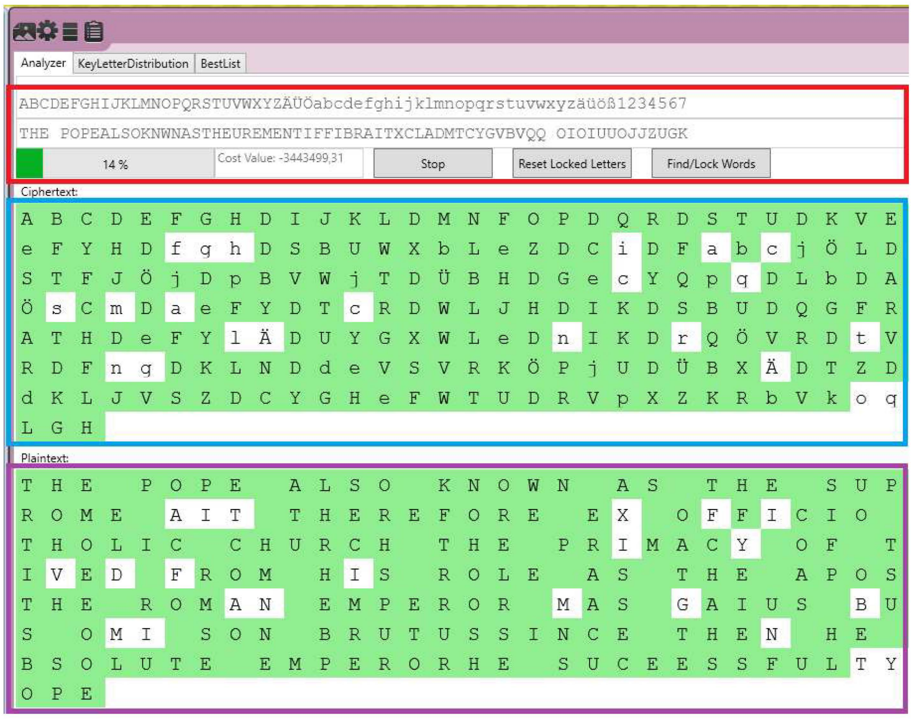
5 Our Approach

This section explains our approach to cryptanalyze homophonic ciphers. First, the requirements for the analyzer are listed. Then, the implementation is described in detail and the intermediate results are discussed.

5.1 Requirements

The requirements for an analyzer for homophonic substitution ciphers are:

1. implement the state-of-the-art cryptanalysis methods and algorithms for breaking homophonic substitution ciphers.
2. be "easy-to-use", thus, also non-computer affine people can use it.
3. allow different plaintext languages.
4. allow the manual change of (wrongly) mapped homophones to plaintext letters during the analysis.
5. show ciphertext and plaintext in parallel, thus, making it easy to identify mappings of homophones to plaintext letters.
6. allow different types of input ciphertexts, i.e. number groups and arbitrary ciphertext letters (UTF-8 transcribed texts).



```
ABCDEFGHIJKLMNOPQRSTUVWXYZÄÜÖabcdefghijklmnopqrstuvwxyzäüöß1234567
THE POPEALSOKNWNASTHEUREMENTIFFIBRAITXCLADMTCYGVBVQQ OIOIUUOJJZUGK
```

```
A B C D E F G H D I J K L D M N F O P D Q R D S T U D K V E
e F Y H D f g h D S B U W X b L e Z D C i D F a b c j Ö L D
S T F J Ö j D p B V W j T D Ü B H D G e c Y Q p q D L b D A
Ö s C m D a e F Y D T c R D W L J H D I K D S B U D Q G F R
A T H D e F Y l Ä D U Y G X W L e D n I K D r Q Ö V R D t V
R D F n g D K L N D d e V S V R K Ö P j U D Ü B X Ä D T Z D
d K L J V S Z D C Y G H e F W T U D R V p X Z K R b V k o q
L G H
```

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

Figure 4: Analyzer Tab of the Homophonic Analyzer in CrypTool 2

7. support spaces between words in the plaintext.

Requirement 4 means to have an *interactive* mode in addition to the *full-automatic* mode. This additional interactive mode is the main difference compared to tools described in Section 3.2.

5.2 Implementation

The analyzer is implemented as an CT2 component and allows the visual analysis of homophonic substitution ciphers. Figure 4 shows the current state of the "Homophonic Substitution Analyzer".

The analyzer has three "tabs", each tab shows a different user interface. The tabs can be changed by clicking on the tab names on the top of the window (this is a maximized view of the component).

In Figure 4, the "Analyzer" tab is selected, which consists of three main parts: (1) the top part, framed with with a red rectangle, contains some control buttons and an indicator field, showing the status of the cryptanalysis (percentage value of done hillclimbing cycles), (2) the given ciphertext, which is framed with a blue rectangle, and (3) the putative plaintext, which is framed with a purple rectangle.

Depending on the mode in which the analyzer is executed (see the component's parameters), the user can start and stop the analysis manually by clicking on the "Analyze/Stop" toggle button. This is only possible in the so-called *semi-automatic or interactive* mode: When stopped the user can "lock" already correct letters, which then appear with a green background. Non-locked letters appear with a white background. "Locked" means, that the analyzer won't change these plaintext letters during restarts of the further analysis process. Also, the user is able to connect a dictionary to the analyzer. Then, the analyzer automatically locks words of defined lengths, that it reveals during the analysis. The user can set a minimum, how often a word has to appear in the plaintext, before the analyzer locks it. Thus, "random words" that may appear during the start of the analysis won't be locked since these are most probably wrong.

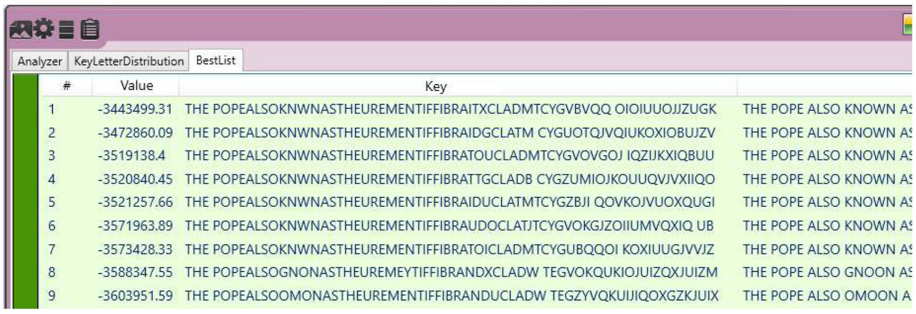
The third tab of the analyzer shows a collection of "best" putative plaintexts (and the according keys) found during the analysis so far. This tab has the title "Bestlist".

As it is also possible to start the analyzer in a *full-automatic* mode, this bestlist will probably contain a text close to the correct plaintext after several automatic "restarts".

In the following, we describe the current state of our cryptanalyzing algorithm.

Baseline Algorithm (for one restart)

1. Initialize a counter c with 0
2. Set a fitness value $f_{globalbest}$ to the smallest



Analyzer | KeyLetterDistribution | BestList

#	Value	Key	
1	-3443499.31	THE POPEALSOKNWNASTHEUREMENTIFFIBRAITXCLADMTCYGVBVQQ OIOIUUOJJZUGK	THE POPE ALSO KNOWN A
2	-3472860.09	THE POPEALSOKNWNASTHEUREMENTIFFIBRAIDGCLATM CYGUOTQJVQIUKOXIOBUJZV	THE POPE ALSO KNOWN A
3	-3519138.4	THE POPEALSOKNWNASTHEUREMENTIFFIBRATOUCLADMTCYGVOVGOJ IQZIJKXIQBUU	THE POPE ALSO KNOWN A
4	-3520840.45	THE POPEALSOKNWNASTHEUREMENTIFFIBRATTGCLADB CYGZUMIOJKOUUQVJVXIIQO	THE POPE ALSO KNOWN A
5	-3521257.66	THE POPEALSOKNWNASTHEUREMENTIFFIBRAIDUCLATMTCYGZBJI QOVKOJVUOXQUGI	THE POPE ALSO KNOWN A
6	-3571963.89	THE POPEALSOKNWNASTHEUREMENTIFFIBRAUDOCLATJTCYGVOKGJZOIIUMVQXIQ UB	THE POPE ALSO KNOWN A
7	-3573428.33	THE POPEALSOKNWNASTHEUREMENTIFFIBRATOICLADMTCYGUBQQOI KOXIUUGJVVJZ	THE POPE ALSO KNOWN A
8	-3588347.55	THE POPEALSOGNONASTHEUREMEYTIFFIBRANDXCLADW TEGVOKQUKIOJUIZQXJUIZM	THE POPE ALSO GNOON A
9	-3603951.59	THE POPEALSOOMONASTHEUREMENTIFFIBRANDUCLADW TEGZYVQKUIJIQOXGZKJUIX	THE POPE ALSO OMOON A

Figure 5: Bestlist Tab of the Homophonic Analyzer in CrypTool 2

possible value that the variable can hold

3. Create a global “best key” $K_{globalbest}$
4. Create a data structure for memorizing “locked mappings” of homophones to plaintext letters
5. Create a random key K_{run} using the letter distribution of the plaintext language (e.g. ’E’ will appear more often in the created key than ’X’) with length of key = number of homophones multiplied by 1.3
6. Set a fitness value f_{run} to the smallest possible value that the variable can hold
7. For each combination of the index i and j from 0 to length of K_{run} -1 do the following
   (a) Swap two elements i and j of K_{run}
   (b) Calculate a fitness value f based on the decryption of the ciphertext using K_{run}
   (c) Check using a simulated annealing based accept-function if the algorithm should keep the change
   (d) If the accept-function returns false, revert the changes in the key
   (e) If the accept-function returns true, set $f_{run} := f$
8. If $f_{run} > f_{globalbest}$
   (a) Set $f_{globalbest} := f_{run}$
   (b) Set $K_{globalbest} := K_{run}$
   (c) Optional: lock words already revealed in the “locked mappings” data structure
   (d) Present a new global best key and fitness value to the user
9. If no better global optimum was found during the last 100 tries, change the last 3 letters of the key to other random letters from the plaintext alphabet
10. Increment the counter c by 1
11. If the counter c is below a maximum cycle number, goto step 7
12. The algorithm terminates here

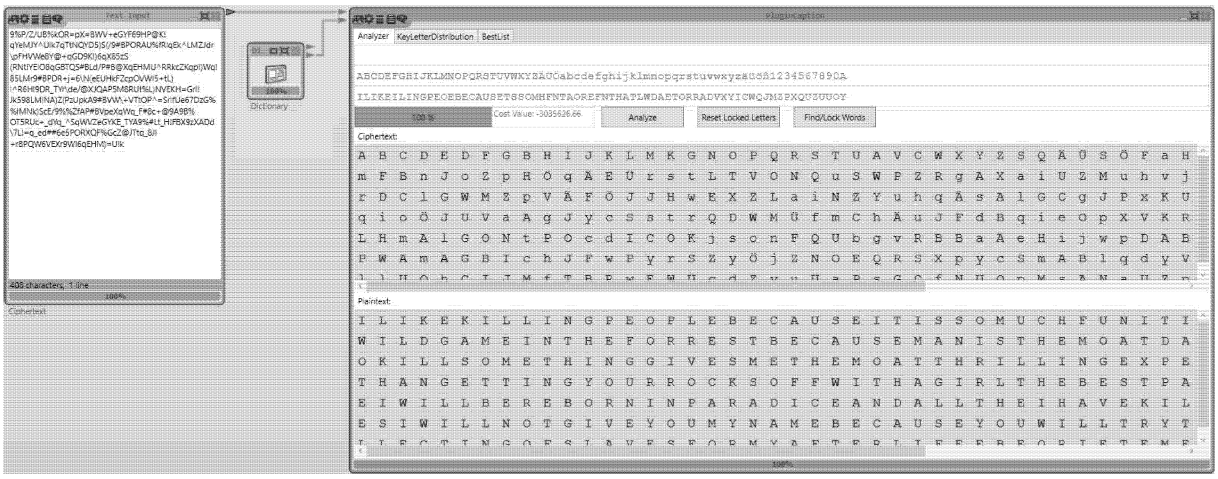
Concept of the Algorithm The idea of our baseline algorithm is hill climbing and a simulated annealing-based accept-function. A key in our algorithm is a mapping of ciphertext homophones array V (defined by a ciphertext alphabet CA) to plaintext letters (defined by a plaintext alphabet PA, for example, the Latin alphabet). An example for such a mapping is

```
CA = ABCDEFGHIJKLMNOPQRST...
V  = THEPOPEALSOKNWASESTZ...
```

Here, the ciphertext homophones ’A’ and ’S’ are mapped both to ’T’, the ’B’ to ’H’, and so on. Our algorithm starts with a random key K_{run}, thus, letters in V are chosen randomly from PA. The choice is done in such a way, that the distribution of letters in V is close to the assumed plaintext language. We furthermore extend the ciphertext alphabet with homophones, which do not exist in the current ciphertext. Thus, it is possible for our algorithm to remove and add plaintext letters to the actual used part of the key that were not used in the previous version of the key. Our hill climber then tries to exchange all possible i and j positions in the key, each position defining a different homophone. The fitness function of our algorithm is the sum of all log_e-pentagrams of the putative plaintext multiplied by a user-defined factor (in our tests 500000). If our algorithm does not find any improvement in 100 steps, it changes the last 3 letters of the key, which are actually not used in decryption, to new random letters. This is done, to get “new letters” into the key which may be exchanged afterwards with letters on the left (which exist in the given ciphertext).

5.3 Key Acceptance Function

In each step (exchange of two key elements), a simulated annealing function (with fixed temperature) accepts or rejects the new key by:



```
9%P/Z/UB%kOR=pX=BWV+eGYF69HP@K!
qYeMJY^UIk7qTtNQYD5)S(/9#BPORAU%fRlqEk^LMZJdr
\pFHVWe8Y@+qGD9KI)6qX85zS
(RNtIYElO8qGBTQS#BLd/P#B@XqEHMU^RRkcZKqpI)Wq!
85LMr9#BPDR+j=6\N(eEUHkFZcpOVWI5+tL)
l^R6HI9DR_TYr\de/@XJQAP5M8RUt%L)NVEKH=GrI!
Jk598LMlNA)Z(PzUpkA9#BVW\+VTtOP^=SrlfUe67DzG%
%IMNk)ScE/9%%ZfAP#BVpeXqWq_F#8c+@9A9B%
OT5RUc+_dYq_^SqWVZeGYKE_TYA9%#Lt_H!FBX9zXADd
\7L!=q_ed##6e5PORXQF%GcZ@JTtq_8J!
+r8PQW6VEXr9WI6qEHM)=UIk
408 characters, 1 line
100%
Ciphertext
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZÄÜÖabcdefghijklmnopqrstuvwxyzäüöß1234567890A
ILIKEILINGPEOEBECAUSETSSOMHFNTAOREFNTHATLWDAETORRADVXYICWQJMZPXQUZUUOY
100 %   Cost Value: -3035626.66
Ciphertext:
A B C D E D F G B H I J K L M K G N O P Q R S T U A V C W X Y Z S Q Ä Ü S Ö F a H
m F B n J o Z p H Ö q Ä E Ü r s t L T V O N Q u S W P Z R g Ä X a i U Z M u h v j
r D C l G W M Z p V Ä F Ö J J H w E X Z L a i N Z Y u h q Ä s A l G C g J P x K U
q i o Ö J U V a A g J y c S s t r Q D W M Ü f m C h Ä u J F d B q i e O p X V K R
L H m A l G O N t P O c d I C Ö K j s o n F Q U b g v R B B a Ä e H i j w p D A B
P W A m A G B I c h J F w P y r S Z y Ö j Z N O E Q R S X p y c S m A B l q d y V
Plaintext:
I L I K E K I L L I N G P E O P L E B E C A U S E I T I S S O M U C H F U N I T I
W I L D G A M E I N T H E F O R R E S T B E C A U S E M A N I S T H E M O A T D A
O K I L L S O M E T H I N G G I V E S M E T H E M O A T T H R I L L I N G E X P E
T H A N G E T T I N G Y O U R R O C K S O F F W I T H A G I R L T H E B E S T P A
E I W I L L B E R E B O R N I N P A R A D I C E A N D A L L T H E I H A V E K I L
E S I W I L L N O T G I V E Y O U M Y N A M E B E C A U S E Y O U W I L L T R Y T
100%
```

Figure 6: Breaking Zodiac-408 with the New Homophonic Substitution Analyzer in CT2

- if *newKeyScore* $>$ *currentKeyScore* return *true*
- $degradation = currentKeyScore - newKeyScore$
- $acceptanceProbability = e^{\frac{-degradation}{fixedTemperature}}$
- if $acceptanceProbability > 0.0085$ and a randomly chosen value between 0.0 and 1.0 is smaller than the $acceptanceProbability$ return $true$
- return $false$

The $fixedTemperature$ value is also user defined (in our tests 15 000). We obtained the acceptance function from (Lasry, 2018a).

5.4 Discussion

Using this analyzer, it was possible to decrypt even difficult (e.g. 81 homophones, 500 letters ciphertext) homophonic ciphertexts. The example ciphertext[2], used also for the tests of the tools in the related work Section 3.2, needed manual input of the user (locking already correctly decrypted parts of plaintext). The quality of the full-automatic mode of our analyzer is between the two tools shown in Section 3.2. Nevertheless, by integrating the analyzer in CT2, a user is able also to use the interactive mode.

In the following section, we show how a user may use our tool to break a real-world ciphertext encrypted homophonically.

6 Breaking a Real Homophonic Substitution Cipher

During the development of the analyzer, we broke the original Zodiac-408, which was sent by the Zodiac killer to different newspapers in 1969.

We used a transcription from the zodiackiller-ciphers webpage, copied it into a *TextInput* component in CT2 and connected it to the Homophonic Substitution Analyzer component (see Figure 6). Additionally, a dictionary component (with ca. 41,000 English words) was connected to the analyzer. The analyzer was set to semi-automatic mode with 1 000 cycles. By chance, we got some parts of partial words, i.e. PEOPLE or AFTERLIFE, that the analyzer either automatically locked or we locked them manually. Then, with these locked words, we incrementally restarted the analyzer and fixed other parts. The final result of our semi-automatic analysis is shown in Figure 6.

Often, having a good random starting key, the analyzer finds a nearly complete solution instantly, so we do not need to fix many letter mappings by our own. If not, the cryptanalyst is able in the semi-automatic mode to correct partially correct found words by himself. To change the mapping of homophones he has just to right click the according plaintext letter.

Our analyzer is able to solve the Zodiac-408 completely on its own in the full-automatic mode. Also two Spanish Strip ciphers from (MTC3, 2018) (the two where the solution already has been known) have been analyzed successfully.

7 Conclusion

This paper shows the current state of developing an analyzer for cryptanalyzing homophonic substitution ciphers in CrypTool 2 (CT2) within the DECRYPT project. Right now, the analyzer is already able to full-automatically and semi-automatically break real world homophonic ciphers like Zodiac-408, which is often used as a default test case for homophonic analysis. The used algorithm consists of hill climbing, uses pentagram language frequencies in a fitness function, and a simulated annealing-based acceptance function with fixed temperature. A dictionary is used to automatically lock already revealed words. Additionally, the current state of the art of cryptanalyzing homophonic ciphers is presented. The best currently available tool is AZdecrypt. Our analyzer is almost comparable in its success rate, but additionally offers an easy-to-use UI to manually change and lock revealed parts of the plaintext. In future work, we'll improve the success rate of the analyzer by investigating the usage of hexagram statistics and more deeply evaluating the parameter sets and the possibilities of other fitness functions. Also, we want to analyze all (homophonic) encrypted ciphertexts of the DECODE database. Finally, we will extend the analyzer to support nomenclatures and code books.

Acknowledgments

This work has been supported by the Swedish Research Council, grant 2018-06074, DECRYPT – Decryption of historical manuscripts.

References

Citta del Vaticano Archivio Segreto Vaticano. 2019. Archivum Secretum Vacticanum, http://www.archiviosegretovaticano.va/.

Fco Alberto Campos, Alberto Gascón, Jesús María Latorre, and J Ramón Soler. 2013. Genetic Algorithms and Mathematical Programming to Crack the Spanish Strip Cipher. *Cryptologia*, 37(1):51–68.

Thang Dao. 2008. Masters Thesis: Analysis of the Zodiac 340-Cipher.

Amrapali Dhavare, Richard M Low, and Mark Stamp. 2013. Efficient Cryptanalysis of Homophonic Substitution Ciphers. *Cryptologia*, 37(3):250–281.

David Kahn. 1996. *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet.* Simon and Schuster.

John C King and Dennis R Bahler. 1993. An algorithmic solution of sequential homophonic ciphers. *Cryptologia*, 17(2):148–165.

Nils Kopal, Olga Kieselmann, Arno Wacker, and Bernhard Esslinger. 2014. CrypTool 2.0. *Datenschutz und Datensicherheit—DuD*, 38(10):701–708.

Nils Kopal. 2018. Solving Classical Ciphers with CrypTool 2. In *Proceedings of the 1st International Conference on Historical Cryptology HistoCrypt 2018*, number 149, pages 29–38. Linköping University Electronic Press.

George Lasry. 2018a. *A Methodology for the Cryptanalysis of Classical Ciphers with Search Metaheuristics.* kassel university press GmbH.

George Lasry. 2018b. Deciphering German Diplomatic and Naval Attaché Messages from 1914-1915. In *Proceedings of the 1st International Conference on Historical Cryptology HistoCrypt 2018*, number 149, pages 55–64. Linköping University Electronic Press.

Beata Megyesi, Kevin Knight, and Nada Aldarrab. 2017. DECODE – Automatic Decryption of Historical Manuscripts. `http://stp.lingfil.uu.se/~bea/decode/`.

MTC3. 2018. MysteryTwister C3 The Crypto Challenge Contest, https://www.mysterytwisterc3.org/.

David Oranchak. 2008. Evolutionary Algorithm for Decryption of Monoalphabetic Homophonic Substitution Ciphers Encoded as Constraint Satisfaction Problems. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1717–1718. ACM.

Sujith Ravi and Kevin Knight. 2011. Bayesian Inference for Zodiac and other Homophonic Ciphers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 239–247. Association for Computational Linguistics.

Luis Alberto Benthin Sanguino, Gregor Leander, Christof Paar, Bernhard Esslinger, and Ingo Niebel. 2016. Analyzing the Spanish Strip Cipher by Combining Combinatorial and Statistical Methods. *Cryptologia*, 40(3):261–284.

José Ramón Soler Fuensanta and Francisco Javier López-Brea Espiau. 2007. The Strip Cipher—The Spanish Official Method. *Cryptologia*, 31(1):46–56.

Satoshi Tomokiyo. 2018. How I reconstructed a Spanish cipher from 1591. *Cryptologia*, 42(6):477–484.

Zodiac Killer Ciphers ZKC. 2019. Webpage and wiki, http://zodiackillerciphers.com/.