

# Proceedings of the

LMN QRST  
LMNO QRSTUVW  
E J LMNOPQRSTUVWXYZ  
EF KLMNOPQRSTUVWXYZA  
FGH KLMNEGATELUNAXYZABCD  
GHI LMNOPQRSTUVWXYZABCDE  
GHIJ LMNOPQRSTUVWXYZABCDEFGHI  
HIJKLMNPOQRSTUVWXYZA  
IJKLMNO P  
J MNOPQ

2019

# HistoCrypt

2<sup>nd</sup> International Conference on Historical Cryptology

TUVWXYZ ABCD  
UVWXYZ ABCDE  
XYZABCDE  
Y CDE  
DE

NO P  
NO P Q  
GHIJKLMNOPQRS  
IJKLMNPOQRST  
JKLMNPOQRSTU  
LMNOPQRSTU  
NOPQRSTU  
OPSTU  
T



June 23-26, 2019 Mons, Belgium

Editors: Eugen Antal, Klaus Schmeh



**Proceedings of the  
2<sup>nd</sup> International Conference on Historical  
Cryptology**

**HistoCrypt 2019**

**Editors  
Eugen Antal and Klaus Schmeh**

**June 23-26, 2019  
Mons, Belgium**

Published by

NEALT Proceedings series volume 37  
Linköping University Electronic Press, Sweden  
Linköping Electronic Conference Proceedings 158  
ISSN: 1650-3686  
eISSN: 1650-3740  
ISBN: 978-91-7685-087-9



## Preface

I am happy to present the Proceedings of this 2<sup>nd</sup> International Conference on Historical Cryptology (HistoCrypt 2019) in the Mundaneum in Mons, Belgium.

HISTOCRYPT addresses all aspects of historical cryptology/cryptography including work in closely related disciplines (such as history, history of ideas, computer science, AI, computational linguistics, linguistics, or image processing) with relevance to historical ciphertexts and codes. The subjects of the conference include, but are not limited to the use of cryptography in military, diplomacy, business, and other areas, analysis of historical ciphers with the help of modern computerized methods, unsolved historical cryptograms, the Enigma and other encryption machines, the history of modern (computer-based) cryptography, linguistic aspects of cryptology, the influence of cryptography on the course of history, or teaching and promoting cryptology in schools, universities, and the public. HISTOCRYPT represents a continuation of the friendly events of European Historical Ciphers Colloquiums (EuroHCC) held in Heusenstamm (2012), Kassel (2016), and Smolenice (2017) to discuss on-going research in historical cryptology in Europe. Considering EuroHCC's growing popularity among the crypto-historians and cryptographers and the established HICRYPT network on historical cryptology with over 100 members from 20 countries around the world, our aim is to establish as an annual, world-wide event. The first HISTOCRYPT in the series was organized in 2018 in Uppsala, Sweden. The second event in the series takes place in 2019 at Mundaneum in Mons, Belgium.

The conference topics include:

- the use of cryptography in military, diplomacy, business, and other areas,
- analysis of historical ciphers with the help of modern computerized methods,
- unsolved historical cryptograms such as the Voynich manuscript,
- the Enigma and other encryption machines,
- the history of modern (computer-based) cryptography,
- linguistic aspects of cryptography,
- the influence of cryptography on the course of history,
- teaching and promoting cryptography in schools, universities, and the public.

The Program Committee has selected 23 papers out of 25 submissions for presentation (13 in the research track, 6 in the exposition track and 4 as poster/demo). Three papers accepted were later withdrawn by the authors. Papers accepted for the research track are collected in these Proceedings.

I would like to thank the Program Committee, Steering Committee and Local Organizers for their hard work in establishing HISTOCRYPT 2019. Their dedication and invaluable input is highly appreciated. The thanks also goes to all the reviewers, subreviewers, keynote speakers and all authors without whom this conference would not have taken place.

*Klaus Schmeh (Program Chair)*

## **Program Committee**

- Paolo Bonavoglia, Mathesis Venezia, c/o Convitto Liceo "Marco Foscarini", Venezia, Italy
- Camille Desenclos, Université de Haute-Alsace, France
- Bernhard Esslinger, University of Siegen, Germany
- Joachim von zur Gathen, Bonn-Aachen International Center for Information Technology, Germany
- Marek Grajek, Freelance cryptography consultant and historian, Poland
- Otokar Grošek, Slovak University of Technology in Bratislava, Slovakia
- Benedek Láng, Budapest University of Technology and Economics, Hungary
- Karl de Leeuw, IDTopiQ, the Netherlands
- Beáta Megyesi, Uppsala University, Sweden
- Karol Nemoga, Mathematical Institute of the Slovak Academy of Sciences, Slovakia
- Anne-Simone Rous, Saxon Academy of Sciences and Humanities, Germany
- Klaus Schmeh, private scholar, Germany
- Gerhard F. Strasser, Emeritus, Pennsylvania State University, USA
- John Dermot Turing, Kellogg College, Oxford, United Kingdom

## **Steering Committee**

- Joachim von zur Gathen, Bonn-Aachen International Center for Information Technology, Germany
- Marek Grajek, Freelance cryptography consultant and historian, Poland
- Klaus Schmeh, private scholar, Germany
- Arno Wacker, Privacy and Compliance, Bundeswehr University Munich, Germany

## **Local Organizing Committee**

- Jean-Jacques Quisquater

## **Subreviewers**

- Eugen Antal, Slovak University of Technology in Bratislava, Slovakia
- Nils Kopal, University of Siegen, Germany
- George Lasry, The CrypTool Team, Germany
- Pavol Zajac, Slovak University of Technology in Bratislava, Slovakia





## INVITED TALKS

- Vincent Rijmen, Joan Daemen:  
*The History of AES*
- Bernard Fabrot:  
*Breaking LCS35*
- Bart Preneel:  
*Lumumba; or Breaking 1961 Hagelin Ciphertexts*
- Marc McMenamin:  
*Codebreaker Richard Hayes*
- Ingo Niebel:  
*The German Crypto-Crisis of 1917 - How the Zimmermann Telegram Completed the Genesis of German Cryptology*
- René Zandbergen:  
*No News About the Voynich MS?*



## POSTER AND DEMO

- Nils Kopal, Tobias Schrödel:  
*Crypto Books*
- Mauran Philippe:  
*Cyphers and Shadow Diplomacy in England During 17th Century: Prince of Condé Looking for Support*



## EXPOSITION TRACK

- Gerhard F. Strasser:  
*Johann J. H. Bücking (1749-1838) – Medical Doctor, Inventor, and Cryptologist*
- Jerry McCarthy:  
*Recreating the Polish Bomba, Predecessor to the Turing-Welchman Bombe*
- Carola Dahlke:  
*From Antiquity to Post-Quantum Cryptography: A New Gallery on Cryptology at the Deutsches Museum, Munich*
- Monir Azraoui, Solenn Brunet, Sébastien Canard, Aïda Diop, Lélia Eveillard, Alicia Filipiak, Adel Hamdi, Flavie Misarsky, Donald Nokam Kuate, Marie Paindavoine, Quentin Santos and Bastien Vialla:  
*CYBERCRYPT: Learn Basic Cryptographic Concepts while Playing*
- Giuseppe Bianchi:  
*Voynich, Hardware and Software*



# Contents

Preface . . . . .	iii
INVITED TALKS . . . . .	vii
POSTER AND DEMO . . . . .	ix
EXPOSITION TRACK . . . . .	xi
RESEARCH TRACK	
<i>Hieronimo di Franceschi and Pietro Partenio: Two Unknown Venetian Cryptologists</i>	
Paolo Bonavoglia . . . . .	3
<i>Decrypting the Hill Cipher via a Restricted Search over the Text-Space</i>	
Florent Dewez, Valentin Montmirail . . . . .	13
<i>Cryptology in the Slovak State During WWII</i>	
Eugen Antal, Pavol Zajac, Otokar Grošek . . . . .	23
<i>The Typex Scare of 1943: How Well Did the British React to a Cypher-Security Scare?</i>	
Dermot Turing . . . . .	31
<i>A Practical Meet-in-the-Middle Attack on SIGABA</i>	
George Lasry . . . . .	41
<i>Dead Ends in Breaking an Unknown Cipher: Experiences in the Historiography of the Rohonc Codex</i>	
Benedek Láng . . . . .	51
<i>Uruguayan Cryptography: Printed Book Covers</i>	
Juan José Cabezas, Francisco Castro, Joachim von zur Gathen, Jorge Tiscornia, Alfredo Viola . . . . .	59
<i>The DECODE Database Collection of Historical Ciphers and Keys</i>	
Beáta Megyesi, Nils Blomqvist, Eva Pettersson . . . . .	69
<i>Beyond Schlock on Screen: Teaching the History of Cryptology Through Media Representations of Secret Communications</i>	
Peter Krapp . . . . .	79
<i>Solving a 40-Letter Playfair Challenge with CrypTool 2</i>	
George Lasry . . . . .	87
<i>Cryptanalysis of an Early 20th Century Encrypted Journal</i>	
Tony Gaffney, Klaus Schmeh . . . . .	97

<i>Cryptanalysis of Homophonic Substitution Ciphers Using Simulated Annealing with Fixed Temperature</i>	
Nils Kopal . . . . .	107
<i>Using the Entropy of N-Grams to Evaluate the Authenticity of Substitution Ciphers and Z340 in Particular</i>	
Tom S Juzek . . . . .	117



## RESEARCH TRACK



# Hieronimo di Franceschi and Pietro Partenio: Two Unknown Venetian Cryptologists

**Paolo Bonavoglia**

Former teacher of Mathematics and Computer Science  
Mathesis Venezia c/o Convitto "Marco Foscarini" Venezia  
paolo.bonavoglia@liceofoscarini.it

## Abstract

In 1596 the powerful Council of Ten, the secret service of the Republic of Venice, sent a message to the new Baylo in Constantinople, warning him to use, for ordinary messages which needed to be encrypted, Pietro Partenio's cipher, but for questions of extraordinary importance to use the *Zifra delle caselle*<sup>1</sup> cipher invented by Hieronimo di Franceschi. But who were Partenio and Franceschi?

This paper is the report of the first results of a research in the State Archive of Venice about these two unknown cryptologists, still in progress.

## 1 Two unknown cryptologists

Hieronimo<sup>2</sup> di Franceschi, Pietro Partenio, who were they?

If one searches the web with Google<sup>3</sup> for these names the result is a long list of results having nothing to do with cryptology.

And still the State Archive of Venice has plenty of documents about them, dispersed in several funds and envelopes. And there is plenty of documents having to do with Franceschi and Partenio.

Let us start with a 1596 letter.

<sup>1</sup>The word *zifra* or *ziffra* is used in the XVI century for cipher; beginning at the end of that century, *cifra* replaces more and more *zifra*

<sup>2</sup>*Hieronimo* is a very common name in the XVI century; towards the end of the century the Italian form *Gerolamo* or *Girolamo* takes over.

<sup>3</sup>Google is today the most powerful tool for fast searches, very useful also for serious researches, most notably Google Books gives access to a huge library of old books otherwise hard to find; so a Google negative result is meaningful. Of course I had searched also the indexes of the most authoritative cryptology books like (Kahn, 1967), (Bauer, 1997), and the archives of Cryptologia, with the same negative result. As far as I know Franceschi's and Partenio's names are mentioned in passing and without details only in (Pasini, 1872), and (Preto, 1994). So the use of the adjective *unknown* seems appropriate.

## 2 Two statements of the Council of Ten

Inside the archive there is an interesting letter, dated 30 August 1596, written by the Chiefs of the Council of Ten,<sup>4</sup> to the new baylo of Constantinople.

The text translated into English is:

We recommend with the Chiefs of the Council of X, that when it is necessary to write in cipher you continue using the ordinary cipher, but, when treating affairs of extraordinary importance, you will use the [*Zifra delle caselle*] of the cautious and most loyal secretary of the Senate Hieronimo di Franceschi, abstaining from using those of the most loyal Pietro Partenio, up to our new order.

The message is signed by Piero Lando, and two of the chiefs of CCX. Here Franceschi's cipher is seen as better than Partenio's.

But, as we will see in the following, in 1593 another document of CCX had stated just the contrary.

Now we will examine some of these ciphers of Franceschi and Partenio. The most surprising aspect is that both of them used super-encryption as a method to enforce security. But first I will give a short description of a typical Venetian code.

## 3 A XVI century Venetian nomenclator

So to begin let us see a typical Venetian nomenclator<sup>5</sup> used in the second half of the XVI cen-

<sup>4</sup>The Council of Ten was the secret service of the Republic of Venice, and was in charge for ciphers; in the following I will use the two short forms used in the archive: CX for Council of Ten; CCX for Chiefs of the Council of Ten; and ASVE is the common acronym for *Archivio di Stato di Venezia*.

<sup>5</sup>The words "nomenclator" and "code" are in some way synonyms in the cryptographic lexicon; usually a nomencla-

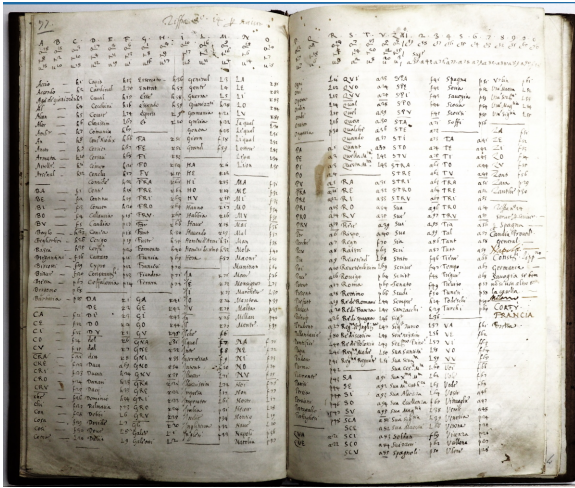


Figure 1: The *ziffra grande* in the book of ciphers 1578-1587. ASVE Cifre, chiavi e scontri di cifra ... b.4, r.16. For no profit use only

tury. Hundreds of diplomatic messages encoded this way are stored in the Venetian archives.

A good source is a book of ciphers<sup>6</sup> having at the first page a decree of the CX dated August 18, 1578 and at the last page another CX decree dated August 26, 1587.

Both decree mention Hieronimo de Franceschi as the reference person of the CX for ciphers. The last page mentions a *falso scontro* (fake key) cipher proposed by Franceschi, to be given to the baylo of Constantinople for saving the keys even in the case the Turks should seize the baylo and his secretary and force them to handle the key. No technical details are given about this cipher.

At the date of this paper, I couldn't find any other trace of this cipher; as we will see below, ciphers of the like were designed by Pietro Partenio in the following years.

The book has many nomenclators approved by the CX, among them is the *Ziffra n. 14*<sup>7</sup> found at *carta 77* of In the following figure we see the *lista per scriuer* i.e. the encrypting list: As we see the nomenclator has different parts:

- An alphabet, here with three homophones for each letter.

tor is small, typically one or two sheets, while a code is larger, a booklet at least; for obvious reasons in this paper I will use the word nomenclator.

<sup>6</sup>ASVE, CX Cifra, chiavi e scontri di cifra con studi successivi, busta 4, reg. 16. Calligraphy is very similar to that of Franceschi, so it is very likely that the book was written by his own hand .

<sup>7</sup>Copies of this cipher known also as *Ziffra Granda*, the big cipher, are found on loose sheets in the Venetian archive

SYLLABARY [27 syllables]																			
ba	be	bi	bo	bu	gra	gre	gri	gro	gru	qua	que	qui	quo	quu	tra	tre	tri	tro	tru
f <sub>1</sub>	f <sub>2</sub>	f <sub>3</sub>	f <sub>4</sub>	f <sub>5</sub>	r <sub>21</sub>	r <sub>22</sub>	r <sub>23</sub>	r <sub>24</sub>	r <sub>25</sub>	a <sub>21</sub>	a <sub>22</sub>	a <sub>23</sub>	a <sub>24</sub>	a <sub>25</sub>	a <sub>51</sub>	a <sub>52</sub>	a <sub>53</sub>	a <sub>54</sub>	a <sub>55</sub>
ca	ce	ci	co	cu	ha	he	hi	ho	hu	ra	re	ri	ro	ru	ua	ue	ui	uo	uu
f <sub>11</sub>	f <sub>12</sub>	f <sub>13</sub>	f <sub>14</sub>	f <sub>15</sub>	r <sub>11</sub>	r <sub>12</sub>	r <sub>13</sub>	r <sub>14</sub>	r <sub>15</sub>	a <sub>31</sub>	a <sub>32</sub>	a <sub>33</sub>	a <sub>34</sub>	a <sub>35</sub>	f <sub>61</sub>	f <sub>62</sub>	f <sub>63</sub>	f <sub>64</sub>	f <sub>65</sub>
ca	ce	ci	co	cu	la	le	li	lo	lu	sa	se	si	so	su	ta	te	ti	to	tu
f <sub>21</sub>	f <sub>22</sub>	f <sub>23</sub>	f <sub>24</sub>	f <sub>25</sub>	r <sub>71</sub>	r <sub>72</sub>	r <sub>73</sub>	r <sub>74</sub>	r <sub>75</sub>	a <sub>81</sub>	a <sub>82</sub>	a <sub>83</sub>	a <sub>84</sub>	a <sub>85</sub>	f <sub>51</sub>	f <sub>52</sub>	f <sub>53</sub>	f <sub>54</sub>	f <sub>55</sub>
da	de	di	do	du	la	le	li	lo	lu	sca	sce	sci	sco	scu					
r <sub>1</sub>	r <sub>2</sub>	r <sub>3</sub>	r <sub>4</sub>	r <sub>5</sub>	r <sub>81</sub>	r <sub>82</sub>	r <sub>83</sub>	r <sub>84</sub>	r <sub>85</sub>	a <sub>61</sub>	a <sub>62</sub>	a <sub>63</sub>	a <sub>64</sub>	a <sub>65</sub>					
fa	fe	fi	fo	fu	ma	me	mi	mo	mu	spa	spe	spi	spo	spu					
f <sub>51</sub>	f <sub>52</sub>	f <sub>53</sub>	f <sub>54</sub>	f <sub>55</sub>	f <sub>31</sub>	f <sub>32</sub>	f <sub>33</sub>	f <sub>34</sub>	f <sub>35</sub>	f <sub>41</sub>	f <sub>42</sub>	f <sub>43</sub>	f <sub>44</sub>	f <sub>45</sub>					
fra	fre	fri	fro	fru	na	ne	ni	no	nu	sta	ste	sti	sto	stu					
r <sub>61</sub>	r <sub>62</sub>	r <sub>63</sub>	r <sub>64</sub>	r <sub>65</sub>	r <sub>71</sub>	r <sub>72</sub>	r <sub>73</sub>	r <sub>74</sub>	r <sub>75</sub>	a <sub>71</sub>	a <sub>72</sub>	a <sub>73</sub>	a <sub>74</sub>	a <sub>75</sub>					
ga	ge	gi	go	gu	pa	pe	pi	po	pu	stra	stre	stri	stro	stru					
r <sub>41</sub>	r <sub>42</sub>	r <sub>43</sub>	r <sub>44</sub>	r <sub>45</sub>	a <sub>1</sub>	a <sub>2</sub>	a <sub>3</sub>	a <sub>4</sub>	a <sub>5</sub>	a <sub>61</sub>	a <sub>62</sub>	a <sub>63</sub>	a <sub>64</sub>	a <sub>65</sub>					
gna	gne	gni	gno	gmi	pra	pre	pri	pro	pru	ta	te	ti	to	tu					
r <sub>31</sub>	r <sub>32</sub>	r <sub>33</sub>	r <sub>34</sub>	r <sub>35</sub>	a <sub>11</sub>	a <sub>12</sub>	a <sub>13</sub>	a <sub>14</sub>	a <sub>15</sub>	a <sub>41</sub>	a <sub>42</sub>	a <sub>43</sub>	a <sub>44</sub>	a <sub>45</sub>					

Figure 2: The syllabary of the *ziffra grande* the ordered lists are clearly visible.

- An abacus, the ten digits encrypted with one or more groups.
- A syllabary in group of 5, each with a different vowel at the end, for instance **ba, be, bi, bo, bu**.
- A dictionary with common words.

Every letter or group is encrypted with a cipher made of one letter followed by a number of one or two digits, often written like exponents, for instance letter **A** is encrypted with three ciphers (homophones):  $o^{18}t^8u^{15}$  the syllable **FA** is encrypted with  $r^{51}$ , the word *Guerra* is encrypted with  $L^{54}$  and so on, for about five hundred ciphers. The heart of this cipher is the syllabary, these signs are the most used. Here is a more readable table; it appears a strong regularity, syllable ending with **A** always end with **1**, syllable in **B** always end with **2** and so on. This is an obvious weakness, the enemy will get great help in rebuilding the syllabary. This cipher was also known as *ziffra grande* and it was widely used by ambassadors in European capitals. For not so important matters a smaller cipher was used a *ziffra piccola* (small cipher). An example in the same book is in figure 2.

This cipher has an alphabet with two homophones for each letter, with the exception of **H** who has only a cipher the number 20; the **A** has two homophones 16 and 36, **B** has 13 and 33, **C** has 1 and 21, strangely all homophones have a difference of 20. There is also a small dictionary of 60 words, all with two digits ciphers, from 40 to 99, for instance **con** encrypted with 50, **Re di Spagna** with 73 and so on.

According to Pasini classification<sup>8</sup> this cipher is

<sup>8</sup>Luigi Pasini, see also footnote 1, was the last archivist to reorder the papers having to do with cryptography, and classified ciphers using the cipher for the first letter: A

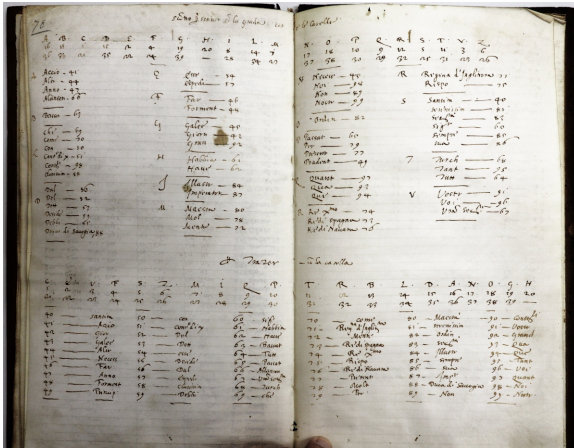


Figure 3: The *cifra piccola* used as the base cipher by the *cifra delle caselle*. ASVE CX Cifre, chiavi e scontri di cifra ... b.4, r.16. For no profit use only

named **A 16-36**. Many copies of this cipher are found inside the folders where Pasini collected the ciphers and key sheets.

But the real importance of this small cipher, will be seen in the next paragraph; a hint can be read in the headline of the page. *Sono per scriuer su la grada cioè le caselle* = They are for writing on the grid, that is the boxes.

#### 4 Hieronimo di Franceschi

Very little is known about this cryptologist; his name is frequently mentioned in the CX papers, in 1578 he is mentioned in a CX book of ciphers<sup>9</sup> as a notary at the Doge's chancellery; in 1587 he is mentioned as a secretary of the Venetian Senate. His name appears in many deeds of the notary Pietro Partenio between 1577 and 1596, acting as an attorney for other people, or as a landlord renting flats. He was the reference person of the CX for cryptography in those years, known above all for his *cifra delle caselle*.

In the first page of the book there are these Franceschi's rules for *scriuer ben la zifra* (to write well the cipher):

1. Use signs that mean words or syllables as much as possible.
2. Having to use simple letters, the signs meaning these letters must be changed, and especially the vowels.
3. When using the superfluous (nulls) put these nulls in the middle between words, between

<sup>9</sup>See footnote 6, page 2.

consonants, and the vowels, and especially behind the Q, behind the S, the T, L, P and so on

#### 5 The cifra delle caselle

Now let's talk about this *cifra delle caselle* one of the most interesting ciphers found in the State Archives of Venice. A cipher which was used in the real world for many years.<sup>10</sup>

First of all let us see a real message from the archives, encrypted with the *caselle*.<sup>11</sup>

It is well visible the ordered and regular way the two digits numbers were written down. This immediately recalls the grids contained in one of the book of ciphers found in the CCX envelope, were four different grids are present.

Three grids have 24 columns, while the fourth, the one for France, for some reason, is thinner having only 21 columns.

But what is important is the perfect correspondence between a grid and an encrypted text.

Above each window in the grid there are three numbers in the range 0..19. What's the purpose of these numbers? The answer is in the *ziffra piccola* seen in the previous chapter, which used numbers in the range 1..20 as ciphers. The reason for those strange homophones differing by 20 is now clear; it is just an *escamotage* to realize a modulo 20 arithmetic.<sup>12</sup>

The plaintext was first encrypted with this small nomenclator, then the resulting encrypted text was written inside the dedicated grids, and the grid number were subtracted to the single ciphers giving the final cryptogram to be transmitted.

The reverse process of deciphering was just the opposite, one had to add numbers of the cryptogram to those of the grid to recover the nomenclator ciphers.

This method of encrypting twice is best known, as **superencryption**, a method which came in

<sup>10</sup>As previously stated, this cipher is mentioned in (Preto, 1994); Preto says only that Franceschi was known as the inventor of this cipher, in fact he is just reporting news found in the deeds of the CX and CCX archives.

<sup>11</sup>The complete method was recovered by the author in December 2018 and a detailed report about the matter will be published on Cryptologia; *The "Cifra delle Caselle", a XVI century superencrypted cipher* (ID: 1609132 DOI:10.1080/01611194.2019.1609132). An updated report is on the web, starting from page: <http://www.crittologia.eu/storia/cifraCaselle.html> [in Italian]

<sup>12</sup>Modular arithmetic was formalized by Gauss in the XIX century, so both Franceschi and Partenio had to invent complicated procedures for this purpose,

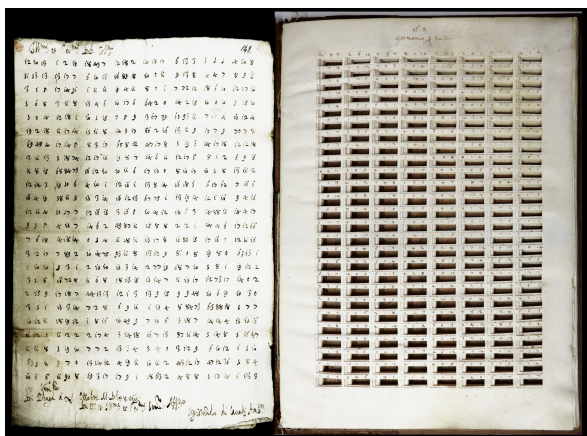


Figure 4: On the left a diplomatic message from the Venetian ambassador in Prague dated 1578-10-11, encrypted with the *cifra delle caselle*. On the right one of the grids used for super-encrypting a message; this is the one used by the ambassador in Germany (Holy Roman Empire). *ASVE Senato, dispacci ambasciatori in Germania, f13, c142 and ASVE CX Cifre, chiavi e scontri di cifra ... b.4. For no profit use only*

common use in the XIX century or immediately before. So a superencryptor cipher is something in advance of two centuries! As far as I know is the oldest of this kind<sup>13</sup>.

## 6 Pietro Partenio

Pietro Partenio was a notary active from 1563 to 1618 according to the register of notary deeds stored in the Venetian archive.

As stated above there are several Partenio's deeds since the 1570s where Hieronimo de Franceschi is named, a proof that Partenio and Franceschi knew each other and had professional links. Partenio is never mentioned in the book of ciphers 1578-1587, so we can guess he became interested in ciphers in the following years and designed several interesting ones.

We find detailed descriptions of six ciphers in a fine CCX parchment book (1592-93)<sup>14</sup>, other ciphers on loose sheets and finally a book of ciphers

<sup>13</sup>Update: the idea of combining two ciphers is rather simple and goes back to the beginnings of cryptography, if it is true that the well known Arab cryptologist Al-Kindi in his IX century book wrote about something like super-encryption, but gave no details or examples. As far as I know, Franceschi's cipher is the first super-encrypted cipher well documented and used in the diplomatic messages of the real world.

<sup>14</sup>ASVE CCX Raccordi 1 1593

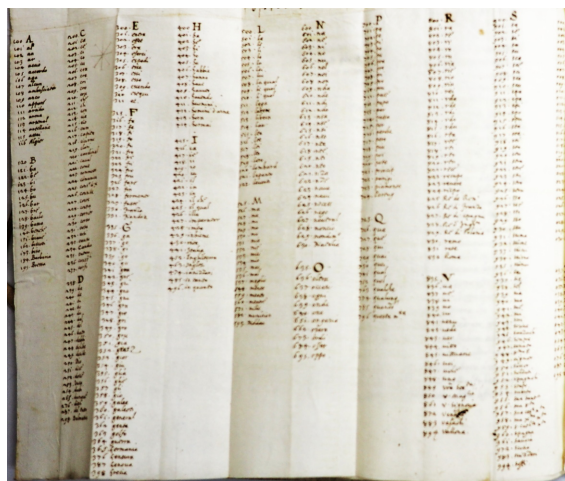


Figure 5: The nomenclator of the second cipher. *ASVE CX Cifre, chiavi e scontri di cifra ... b.2, f.23. For no profit use only*

dated 1606 with six ciphers, some of them already described in the CCX book.

Partenio divides his ciphers into two categories: 1) *cifre sospette* (suspicious ciphers): the suspicious enemy easily recognizes them as encrypted messages; 2) *cifre di senso corrente* (ciphers of current sense) that is ciphers that produce messages of common language, a sort of steganography. This paper is about the first kind, the second deserves further research.

## 7 Partenio's ciphers

Now we will describe and examine some of these ciphers, from the 1592/93 CCX book and from the 1606 booklet. Let's begin with a cipher of the latter, because it is the most similar to Franceschi's *caselle*.

### 7.1 Second cipher (1606)

This second cipher of the 1606 booklet is interesting because Partenio explicitly mentions the Franceschi's *cifra delle caselle* boasting the superiority of his own.

The base cipher is a 3-digit nomenclator shown in the following figure.

The nomenclator is almost totally ordered; there are exception, the syllables are ordered separately, as seen in the alphabet and syllabary shown here.

But, of course, the most interesting part is super-encryption: indeed the method is similar to Franceschi's cipher; one had to do a subtraction to encrypt and an addition to decipher, here using a

ALFABETO																																									
a	b	c	d	e	f	g	h	i	l	m	n	o	p	q	r	s	t	u	z																						
100	120	200	238	300	315	335	400	415	500	525	600	635	700	725	800	900	000	835	035																						
SILLABARIO [29 sillabe]																																									
ba	be	bi	bo	bu	gra	gre	gi	gro	gru	qua	que	qui	quo	quu	ta	te	ti	to	tu																						
121	122	123	124	125	346	347	348	349	350	726	727	728	?	?	001	002	003	004	005																						
ca	ce	ci	co	cu	ha	he	hi	ho	hu	ra	re	ri	ro	ru	tra	tre	tri	tro	tru																						
201	202	203	204	205	401	402	403	404	405	801	802	803	804	805	006	007	008	009	010																						
cra	cre	cri	cro	cru	ia	ie	ii	io	iu	sa	se	si	so	su	tta	tte	ttr	tto	ttu																						
206	207	208	209	210	416	417	418	419	420	901	902	903	904	905	011	012	013	014	015																						
da	de	di	do	du	la	le	li	lo	lu	sca	sce	sci	sco	scu	ua	ue	ui	uo	uu																						
239	240	241	242	243	501	502	503	504	505	906	907	908	909	910	836	837	838	839	840																						
fa	fe	fi	fo	fu	ma	me	mi	mo	mu	spa	spe	spi	spo	spu	ta	te	ti	to	tu																						
316	317	318	319	320	526	527	528	529	530	911	912	913	914	915	036	037	038	039	040																						
fra	fre	fri	fro	fru	na	ne	ni	no	nu	ssa	sse	ssi	ssu																												
321	322	323	324	325	601	602	603	604	605	926	927	928	929	930																											
ga	ge	gi	go	gu	pa	pe	pi	po	pu	sta	ste	sti	sto	stu																											
336	337	338	339	340	701	702	703	704	705	916	917	918	919	920																											
gna	gne	gni	gno	gnu	pra	pre	pri	pro	pru	stra	stre	stri	stro	stru																											
341	342	343	344	345	706	707	708	709	710	921	922	923	924	925																											

Figure 6: Alphabet and syllabary of the second cipher.

modulo 10 arithmetic, instead of the modulo 20 of Franceschi.

Partenio, seeking as usual a key that could be memorized without writing, uses a different method to generate the obscuring sequence of numbers.

He starts with a verse, from a poetry or other text, and writes it on three rows. Let's use his own example, the verse is:

*"Iam in me sperauit liber abacum protega me um quantam c;"* where the final *c* is a null, used to fill the three rows schema. The verse has to be written on three rows, and will be read per columns:

```

iaminmesperauit
liberabacumprot
egameumquantamc

```

Now let us transform these letters in numbers using this table:

1	2	3	4	5	6	7	8	9	0
a	b	c	d	e	f	g	h	i	l
m	n	o	p	q	r	s	t	u	z

Now every letter of the verse is converted into the number above, and the numbers are read per columns forming group of three number to match the ciphers of the nomenclator.

In this example the sequence is:

ile	aig	mba	iem	nre	mau	ebm	...
905	197	121	931	265	119	521	...

The super-encrypting procedure is similar to the *caselle*. The numbers of the single digits of the nomenclator's ciphers are subtracted modulo ten by the numbers of the key.

For deciphering just do a sum instead of a subtraction.

Let us see the example of Partenio:

*Ha questa Maestà intendimento con alcuni de capitani in Corfù.*

The following table shows the procedure; the first row has nomenclator ciphers, the second has:

ha	questa maestà	inte	ndi	mento	con ...
401	796	430	611	559	213 ...
905	197	121	951	265	119 ...
506	609	319	760	394	104 ...

So the cryptogram to send is:

506609319760394104 ...

To decipher just do an addition modulo 10.

Of course Partenio does not use the "modulo 10" arithmetic, introduced by Gauss in the XIX century, and has to write two pages of instructions explaining how to subtract and sum this way.

### 7.2 A comparison with the caselle

At the end of the instructions Partenio makes a comparison between his cipher, defined *fortissima* (very strong) and *quella del Franceschi* (the one of Franceschi), remarking his nomenclator may reach 1000 among words, syllables and single letters, while Franceschi's nomenclator had only two digits and only "40 or 50 among syllables and words"<sup>15</sup>

Partenio makes also an important remark: he is afraid that secretaries may use his ciphers in a simplified and easier manner, using only the nomenclator without the super encrypting tools.

This is exactly what did happen; in the XVII century the most used ciphers were similar to Partenio's, 3-digit ciphers, ordered lists but super-encryption was forgotten.

### 7.3 The false sense

Finally Partenio describes a complicate device to give the cryptogram a false meaning to disguise the enemy. For this he uses this Latin square of numbers<sup>16</sup> (On the left the original<sup>17</sup>, on the right a more readable view):

Suppose you want to add this fake message:  
*Sarà guerra tra questa m.tà et Re di Polonia*

<sup>15</sup>Indeed Franceschi's small cipher has 20 letters and 60 words; it does not have syllables.

<sup>16</sup>Latin square is a square of  $n \times n$  objects where every object appears once and only once on each row and on each column. Mathematically this is the Pythagorean table of a binary operation that is invertible; the associated algebraic structure is called quasi-group. For this reason the Latin squares have been widely used in cryptography beginning with Trithemio, Vigenère and so on. This one is peculiar being disordered.

<sup>17</sup>ASVE Cifre, chiavi, scontri di cifra ... busta 3

	1	2	3	4	5	6	7	8	9	0
1	4	3	8	0	9	6	1	7	5	2
2	3	8	0	9	6	1	7	5	2	4
3	8	0	9	6	1	7	5	2	4	3
4	0	9	6	1	7	5	2	4	3	8
5	9	6	1	7	5	2	4	3	8	0
6	6	1	7	5	2	4	3	8	0	9
7	1	7	5	2	4	3	8	0	9	6
8	7	5	2	4	3	8	0	9	6	1
9	5	2	4	3	8	0	9	6	1	7
0	2	4	3	8	0	9	6	1	7	5

Figure 7: The latin square, original on the left, more readable on the right. *ASVE CX Cifre, chiavi e scontri di cifra ... b.2, libro Partenio. For no profit use only*

The first syllable of the fake message is *sa*; the nomenclator has 901 as the cipher of it. Now you apply the binary operation defined by the Latin square to the digit of the fake text and the corresponding digit of the cryptogram, as in the following table,

sa	Ra	guerra	tra	questa m.tà	et	...
901	801	364	006	796	311	...
506	609	319	760	394	104	...
856	855	963	699	515	820	...

Finally we intercollegiate the numbers of the true cryptogram with the fake one, so obtaining the following fake cryptogram:

58056668059539169376690935984518044

Now the secretary receiving this cryptogram knows that only the odd placed numbers are good and will easily recover the plaintext.

But, to use Partenio's example, if the Baylo of Constantinople or his secretary are forced by the Turks to deliver ciphers and keys, they will give them the nomenclator and the Latin square and these false instructions: take the numbers in pairs, follow the first number row until you find the second and write the column number, group the numbers obtained by three and use the nomenclator to retrieve the normal text. Due to the property of the Latin square, the Turks will get the false message. Try it and believe it.<sup>18</sup>

#### 7.4 Remarks

This is an amazing cipher, undoubtedly. It has also a pair of weakness: 1) violation of Knockoff's

<sup>18</sup>Hint: take the first two numbers 5 and 8, look the 5 row and find 8 under 9, 9 is the first digit; take 0 and 5 and in the 0 row find 5 under 0, the second digit is 0; take 6 and 6, look 6 row and find 6 under 1, the third digit is 1, so the first cipher is 901, from the nomenclator you get "sa". And so on ...

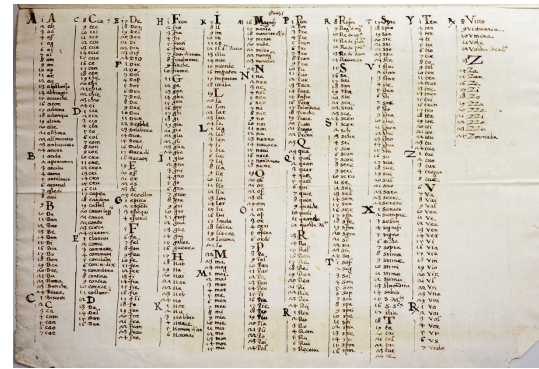


Figure 8: The nomenclator of Partenio's third cipher. *ASVE Cifre, chiavi e scontri di cifra ... b.2, f. Partenio. For no profit use only*

rule; if the enemy discovers the method, the fake effect is lost. 2) the nomenclator is too regular, it is almost a ordered list.

#### 7.5 Third cipher (1592)

The following cipher is the third one of the 1592 CCX register<sup>19</sup>, and may be considered another Partenio's reply to Franceschi's *caselle*. Instead of a grid, we have a paperboard slider as a poly alphabetic tool.

The base cipher, a nomenclator has about a thousand signs formed by a letter from a 24 letters alphabet (Italian with K X Y &) followed bay a two digits number in the range 1..24. For instance **a** is encrypted with **A**, **DA** with **E12**, **Il Signor Turco** with **K12**, **Galee** with **I15**.

Let's see the cipher procedure with the example used by Partenio: the message to be encrypted is. "*Il Signor Turco arma galee*"<sup>20</sup>. We find **k12** as the cipher of "**Il Signor Turco**"; now we have to use a paperboard slider (see following figure) made of a fixed part (top and bottom in the figure) and a sliding part (middle in the figure).

We find copies of this strip with instructions in several parts of the Venetian archive. The one shown below is pasted on an instruction sheet found in the Venetian Archives.

here is a more readable presentation:



We start with the slider in the aligned position, as in the following figure:

<sup>19</sup>ASVE CCX Raccordi 1, 1592 p.28

<sup>20</sup>English: The Turkish Master is arming galleys





Figure 9: Partenio's strip pasted on a cipher sheet. *ASVE CX Cifre, chiavi e scontri di cifra ... b.2, f.23. For no profit use only.*

a	b	c	d	e	f	g	h	i	k	l	m	n	o	p	q	r	s	t	u	x	y	z	&	
y	i	r	k	z	a	s	b	t	g	q	f	x	u	n	m	d	h	c	k	e	p	o	l	y
18	2	19	7	16	6	21	20	13	12	4	8	3	11	5	15	14	11	22	9	17	24	23		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	

Now under **k** in the top row we find **g**, while above **12** there is **8**, so **k12** becomes **g8**.

We find **Arma** has cipher **b4**; under the letter **b** we find **i**, while above **4** there is **7**. So, **b4** is with **i7**. In a similar way the cipher of Galee, **i15** becomes **t5**. So the cryptogram for "Il Signor Turco arma galee" is:

**g8 i7 t5**

Partenio does not give detailed rules to when and how to move the slider, changing the alphabet. He writes this is something to be agreed between the two parts. As an example he proposes to move the slider one step to the left, every one or two lines.

The deciphering procedure is just the inverse of the previous. In the above example to decrypt **g8 i7 t5** we just look for **g** and **8** using the slider from bottom to top, and finding **k12**. And so on with the rest.

### 7.6 Remarks

Again, the weakness of this cipher is the base cipher, too regular. A super-encryption with a mono-alphabetic substitution could be enough to overcome this weakness; the poly-alphabetic substitution is a plus giving a good level of safety for the XVI century.

### 7.7 Sixth cipher

The sixth cipher of the book is basically a transposition cipher based on a long key. Partenio uses this example: as a key-phrase take the Latin "En lex tua meditatio mea in corde meo" phrase as a message to send: *Vi sono in Brescia capi ribelli*.

One writes down the key-phrase on a row, finds the first letter, in the alphabetic order, here as in most cases **a**, and writes **1** exactly under this **a**; then find the second letter, another **a** and under it writes **2** and so on until the end of the message.

	S	P	I	R	T	Q	V	A	N	D	C	O	H	L	F	Z	G	E	M	B
I	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
L	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1
M	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1	2
N	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1	2	3
O	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1	2	3	4
P	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1	2	3	4	5
Q	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1	2	3	4	5	6
R	8	9	10	11	12	13	14	15	16	17	18	19	20	1	2	3	4	5	6	7
S	9	10	11	12	13	14	15	16	17	18	19	20	1	2	3	4	5	6	7	8
T	10	11	12	13	14	15	16	17	18	19	20	1	2	3	4	5	6	7	8	9
V	11	12	13	14	15	16	17	18	19	20	1	2	3	4	5	6	7	8	9	10
X	12	13	14	15	16	17	18	19	20	1	2	3	4	5	6	7	8	9	10	11
Z	13	14	15	16	17	18	19	20	1	2	3	4	5	6	7	8	9	10	11	12
A	14	15	16	17	18	19	20	1	2	3	4	5	6	7	8	9	10	11	12	13
B	15	16	17	18	19	20	1	2	3	4	5	6	7	8	9	10	11	12	13	14
C	16	17	18	19	20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
D	17	18	19	20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
E	18	19	20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
F	19	20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
G	20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
H	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Figure 10: The square of the sixth cipher.

Next, the plain text is written on a third row following the numbers order; when the text is over, the remaining places are filled with random letters.

Then text resulting on the third row is the cryptogram to dispatch. Here is the example, step by step:

```

E n l e x t u a m e d i t a t i o m e a i n c o r d e m e o
7 20 16 8 30 26 29 1 17 9 5 13 27 2 28 14 22 18 10 3 15 21 4 23 25 6 11 19 12 24
i r c n o i d u a b n c l i b i b p r s a i o e l o e i s l

```

So we have a cipher that does not require a written sheet, may be taken by heart, that's the main goal of Partenio's ciphers<sup>21</sup>.

### 7.8 The fake key

This is a device similar to the one seen in cipher 2, for the same purpose; it is somehow easier to use.

Partenio's idea is to add a fake cipher key (in Italian a *falso scontro*) that the ambassador could give to the enemy.

In the previous example, we could add this fake message: *Dalle sue parole io spero buona pace* of the same length, in a way the enemy, using the fake key would get this fake meaning.

```

u i s o n o i n b r e s c i a c a p i r i b e l l i
e n l e x t u a m e d i t a t i o m e a i n c o r d e m e o
6 19 15 7 29 25 28 0 16 8 4 12 26 1 27 13 21 17 9 2 14 20 3 22 24 5 10 18 11 23
i r c n g i d u a b n c c i a i b p r s a i o e l o e i s l

```

For this purpose Partenio proposes another Latin square table, this time  $20 \times 20$ . The square is regular but row labels are shifted and the column labels scrambled according to a keyword.

The first letter of the encrypted message is **i**, the first of the fake message is **d**, so we look on the first column of the square for the **i** and look on this

<sup>21</sup>Three centuries after Auguste Kerckhoffs included a similar principle as his rule number 3, see (Kerckhoffs, 1883).

row until below the letter **d**. The number found is 10, and we write  $i^{10}$ . The following letter of the encrypted text is **r**, to get the second letter of the fake message **a** we need to reach number **15**, so we write  $r^{15}$ , and so on; finally the encrypted text looks so:

$i^{10}r^{15}c^8n^{17}b^{17}i^1z^{13}u^{28}a^{14}b^1n^7c^6s^{22}...$

a cryptogram who has the typical look of a Venetian nomenclator encrypted message, a perfect fake.

So, in case of capture, the ambassador should give up to the enemy the table square, with instructions leading to recover the fake messages instead of the true ones.

### 7.9 Is this fake perfect?

Indeed the true cryptogram here is the sequence of the letters, the numbers being only a fake leading to the fake meaning. Only half of the cryptogram is good, like in cipher 2. Indeed, the cipher is just a transposition disguised as a nomenclator.

An enemy examining such a cryptogram could at glance observe that the statistical distribution of the letters resembles a plausible language distribution: many vowels, e, i, a the most frequent, and could guess a transposition is the real cipher.

So far, the fake looks weaker than the one seen in the second cipher.

And like cipher 2, this cipher does not satisfy Kerckoffs principle; if the enemy discovers the method, the whole contraption is unmasked.

On the other hand, a transposition for 30-40 letters message is not so easy to break.

## 8 Were Partenio's ciphers used in the real world?

A difficult question; the 1596 CCX letter shown at the beginning of this paper, explicitly refers to Partenio's ciphers as used before 1596 by the Baylo; and still at the current date not a single such message was found in the archives, of the Baylo or other ambassador, to the Doge or to the Council of Ten.<sup>22</sup>

<sup>22</sup>Last update: two paragraphs encrypted with a cipher similar to Partenio's n.2 were found at the beginning of two messages of Piero Duodo, Venetian ambassador in France, dated August 1595; in June 1595 the CX had recommended the use of Partenio's cipher, after learning from Giovanni Mocenigo, the previous ambassador in France, that Francois Viète, the well known French mathematician, boasted to be able to decrypt Venetian ciphers. The cipher seems to have been used for a very short period of time

## 9 Conclusion

From the above examples Franceschi and Partenio have in common the use of super-encryption, but have different priorities: Franceschi cares more about safety, while Partenio, as already stated, cares more about ease of use, and keys easy to memorize.

Ease of use is important: a procedure too complicated may induce bad behaviors of the cipher operators; a classical example is a monoalphabetic cipher with homophones; the operator should change homophone very often, as recommended by Franceschi's rules, but this is annoying and demanding, so an operator may memorize only one cipher for letter going back to a simple monoalphabetic cipher; a secret letter by the CCX to the governor of Candia, has reprimands about bad ciphering habits, and at the end tells: "to use only one alphabet would be like not writing in cipher at all."<sup>23</sup>. The reprimand had little effect and reducing an homophonic cipher to a trivial monoalphabetic remained a common practice.

On the other hand safety is important too: an easy to use cipher may be also an easy to decrypt one. A typical example: the use of an ordered list in a nomenclator, a step in the direction of ease, one just needs a single list, but also a big help for the enemy, an ordered list nomenclator is much easier to break than a disordered one. And yet in the XVII century Venetian cryptography used more and more ordered lists instead of the disordered of the XV and XVI century.

Franceschi used a small but disordered list with super-encryption; Partenio used ordered lists also with super-encryption.

The followers used similar ordered lists but without the burden of superencryption! Precisely the fear expressed by Partenio in his postscript to the second cipher. The golden age of Venetian cryptography had come to an end.

## 10 Acknowledgments

I wish to thank Elio Canestrelli, former professor of mathematics at the Ca' Foscari University of Venice, for providing me with the starting point and information about the encrypted dispatches of Alberto Badoer. Special thanks also to the archivists Giovanni Caniato, Marilena Bonato and many others for the assistance provided at the

<sup>23</sup>CCX Lettere Segrete 10, 25-08-1583

State Archives of Venice.

## References

- Ibrahim Al-Kadi. 1992. *Origins of Cryptology: the Arab Contributions*. Cryptologia, 16:2, 97-126, Philadelphia, Pa. DOI: 10.1080/0161-119291866801
- Friedrich Ludwig Bauer, 1997. *Decrypted Secrets*. Springer, Berlin. ISBN: 3-540-24502-2
- Paolo Bonavoglia. 2019. *The Cifra delle Caselle, a superencrypted XVI Century Cipher*. Cryptologia, Philadelphia, Pa. DOI:10.1080/01611194.2019.1609132
- David Kahn. 1967. *Codebreakers*. Scribner, New York, NY. ISBN: 978-0-684-83130-5
- Auguste Kerckhoffs. 1883. *La cryptographie militaire*. Paris.
- Luigi Pasini 1872, 2019. *Delle scritture in cifra usate nella Repubblica di Venezia*. Aracne, Roma ISBN: 978-88-255-1926-6
- Luigi Pasini. 1885. *Archivio di Stato di Venezia. Consiglio di Dieci - Cifre, Chiavi e Scontri di cifra, a cura di L. Pasini e G. Giomo*.
- Paolo Preto 1994. *I servizi segreti di Venezia*. EST, Milano
- Luigi Sacco. 1958, 2012. *Un primato italiano, la crittografia nei secoli XV e XVI*. Ist.Poligrafico dello Stato, Roma, Italy.



# Decrypting the Hill Cipher via a Restricted Search over the Text-Space

**Florent Dewez**

INRIA, Modal team-project  
Lille–Nord Europe research center, France  
florent.dewez@inria.fr

**Valentin Montmirail**

Université Côte d’Azur  
I3S, CNRS, Nice, France  
vmontmirail@i3s.unice.fr

## Abstract

Developed by L. S. Hill in 1929, the Hill cipher is a polygraphic substitution cipher based on matrix multiplication. This cipher has been proved vulnerable to many attacks, especially the known-plaintext attack, while only few ciphertext-only attacks have been developed. The aim of our work is to study a new kind of ciphertext-only attack for the Hill cipher which is based on a restricted search over an explicit set of texts, called orbits, and not on a search over the key-space; it is called Orbit-Based Attack (OBA). To explain in a convenient setting this approach, we make use of basic notions from group action theory; we present then in details an algorithm for this attack and finally results from experiments. We demonstrate experimentally that this new method can be efficient in terms of time-execution and can even be faster on average than the classical Brute-Force Attack in the considered settings.

## 1 Introduction

The Hill cipher is a relatively old polygraphic substitution cipher based on linear algebra and invented by Lester S. Hill in 1929 (Hill, 1929; Hill, 1931). For a plaintext of  $M$  characters composed of  $m$  blocks of  $n$  characters in an alphabet with  $p$  elements, the Hill cipher considers each block as an element of the vector space  $(\mathbb{Z}_p)^n$  and multiplies each of them by the same  $n \times n$  invertible matrix, called the secret key, to compute in output the whole ciphertext.

Because of its linear nature, it suffers mainly from the known-plaintext attack, *i.e.* attacker can obtain one or more plaintexts and their corresponding ciphertexts, as stated in (Stinson, 2002). This weakness has led to many modifications of the original version of this cipher; see for instance (Ismail et al., 2006; Mahmoud and Chefranov, 2009; Toorani and Falahati, 2009; Toorani and Falahati, 2011). Regarding the ciphertext-only attack, *i.e.* the attacker is assumed to have access only to a set of ciphertexts, it is said in (Wagstaff, 2002; Stinson, 2002) that performing a ciphertext-only attack on the Hill cipher is “much harder” than performing

a known-plaintext one. Indeed it seems that only few such attacks have been developed, all of them supposing an a priori knowledge on the language and making a search over the key-space; we refer the reader to the papers (Bauer and Millward, 2007; Yum and Lee, 2009; Leap et al., 2016; McDevitt et al., 2018).

Further it is known that, in the case of no restrictions on the considered language or alphabet, “the best publicly known ciphertext-only attack on Hill cipher requires full search over all possible secret keys”, as stated in (Khazaei and Ahmadi, 2017). Note that this paper indeed proposes a new attack but only in the case of meaningful English texts with an alphabet of size 26. In the case where  $p$  is a prime number, the Brute-Force Attack tests almost  $p^{n^2}$  matrices (Overbey et al., 2005, Lemma 4.3).

In view of this, we propose in the present paper to study another kind of ciphertext-only attack which is not based on a search over the key-space but rather over restricted regions of the text-space, regions called orbits. This attack, denoted Orbit-Based Attack (OBA), lies on a partition into orbits of the text-space induced by the Hill cipher. In this paper, we prove the existence of this partition exploiting group action theory and we make explicit the orbits by exploiting the property that Hill preserves the linear combinations of blocks in a given text. The ciphertext and the associated plaintext being necessarily in the same orbit, our theoretical results assure that the size of their orbit, and hence the maximal number of texts to test, is smaller than the number of keys. To make clear the ideas of our approach and avoiding too technical computations, we assume that the size of the alphabet is a prime number; similar results are expected to hold true in more general settings.

An algorithm for the OBA is then proposed. Our aim here is to show that this attack can be faster in terms of time-execution on average over random texts than the above mentioned Brute-Force Attack (BFA) in the case where the only assumption is that the size of the alphabet is a prime number. Even though the computational complexities are proved to be roughly the same, we illustrate by means of numerous experiments that the OBA permits to speed-up on average the runtime of the decryption process as compared to the BFA.

## 2 Preliminaries

In this section, we define rigorously the Hill cipher for the sake of completeness and we introduce some notations which will be used throughout the rest of this paper.

### 2.1 The Hill cipher

We start by the definition of the Hill cipher we use in this paper; we refer to (Hill, 1929) for the original one.

**Definition 1** (Hill cipher). *A plaintext string  $X$  of size  $M = mn$  over an alphabet having  $p$  characters is defined as a vector of size  $M$  over  $\mathbb{Z}_p$  using an arbitrary bijection between the elements of the alphabet and the elements of  $\mathbb{Z}_p$ . The plaintext  $X$  is splitted into  $m$  blocs of size  $n$  such that  $X = X_1X_2 \dots X_m$ . An invertible  $n \times n$  matrix  $K$  over  $\mathbb{Z}_p$ , called the key-matrix, is then chosen. Afterwards we construct a block diagonal matrix  $A$  of size  $M \times M$  over  $\mathbb{Z}_p$  whose main diagonal sub-matrices are equal to  $K$ . The encryption is finally performed by considering each  $X_i$  as a vector of  $(\mathbb{Z}_p)^n$  and by computing the ciphertext  $Y = Y_1Y_2 \dots Y_m$  as follows:*

$$Y = AX \pmod{p},$$

which is equivalent to  $Y_i = KX_i \pmod{p}$ , for all  $i \in \{1, \dots, m\}$ . Thanks to the invertible nature of  $K$ ,  $A$  is invertible as well and the decryption is performed by computing:

$$X = A^{-1}Y \pmod{p}.$$

Throughout the rest of this paper, we choose  $p$  as a prime number for the sake of simplicity. This implies in particular that the set  $\mathbb{Z}_p$  is the field of  $p$  elements and so the division is well-defined, making the arguments and computations easier. However, one may plan to generalise the present results to the case where no assumption on  $p$  is made, covering hence more realistic cases.

We define now the set of invertible block diagonal matrices.

**Definition 2.** *Let  $GL_n(\mathbb{Z}_p)$  be the set of invertible matrices of size  $n \times n$  over  $\mathbb{Z}_p$ . A matrix  $A$  of size  $M \times M$  over  $\mathbb{Z}_p$  belongs to  $\mathcal{G}_{M,n}$  if and only if there exists  $K \in GL_n(\mathbb{Z}_p)$  such that*

$$A = \begin{pmatrix} K & & & \\ & K & & \\ & & \ddots & \\ & & & K \end{pmatrix}.$$

We note that  $GL_n(\mathbb{Z}_p)$  and  $\mathcal{G}_{M,n}$  are clearly in bijection.

We are now in position to define the Hill cipher map, which will be proved to be a group action in the following section.

**Definition 3** (Hill cipher map). *Let  $\mathcal{H} : \mathcal{G}_{M,n} \times (\mathbb{Z}_p)^M \rightarrow (\mathbb{Z}_p)^M$  be the map defined by*

$$\forall (A, X) \in \mathcal{G}_{M,n} \times (\mathbb{Z}_p)^M \quad \mathcal{H}(A, X) := AX.$$

### 2.2 Group action theory

Group action theory offers a convenient setting to describe the attack proposed in this paper. While the results can be actually proved without invoking this theory, the latter may be helpful to make clear the effects of Hill on the texts. Consequently, we recall some abstract results from group action theory for the sake of completeness; their proofs can be found for instance in (Smith, 2008, Chapter 10).

In the rest of the present section, the notation  $G$  will refer to a group whose group law and identity element are respectively represented by  $\cdot$  and  $e$ . We start by recalling the notion of a (left) group action on a set.

**Definition 4** (Group action). *Let  $G$  and  $S$  be respectively a group and a set. A map  $\varphi : G \times S \rightarrow S$  is said to be a group action of  $G$  on  $S$  if and only if it satisfies the two following properties:*

- Identity: for all  $s \in S$ , we have

$$\varphi(e, s) = s;$$

- Compatibility: for all  $g, h \in G$  and  $s \in S$ , we have

$$\varphi(g \cdot h, s) = \varphi(g, \varphi(h, s)).$$

We define now the orbit and the stabiliser of an element  $s \in S$ : the orbit of  $s$  is the set of elements of  $S$  to which  $s$  can be sent by the elements of  $G$ , while the stabiliser of  $s$  is the set of elements of the group  $G$  which do not move  $s$ . Let us emphasise that an element  $s \in S$  can not be sent outside its orbit by definition.

**Definition 5** (Orbit and stabiliser). *Let  $\varphi : G \times S \rightarrow S$  be a group action of a group  $G$  on a set  $S$  and let  $s \in S$ .*

1. *The orbit  $Orb_\varphi(s)$  of  $s$  is defined as follows:*

$$Orb_\varphi(s) = \{y \in S \mid \exists g \in G \quad y = \varphi(g, s)\}.$$

2. *The stabiliser  $Stab_\varphi(s)$  of  $s$  is defined as follows:*

$$Stab_\varphi(s) = \{g \in G \mid \varphi(g, s) = s\}.$$

These two notions are closely related: it is shown that the orbit of an element  $s \in S$  is isomorphic to the quotient of the group  $G$  by the stabiliser of  $s$ . Roughly speaking, this means that it is sufficient to move  $s$  by all the elements of the group  $G$  which do not fix  $s$  to recover the whole orbit of  $s$ . In the finite group case, this permits to compute the cardinal of the orbit of a given element  $s \in S$ :

**Corollary 1.** *Let  $\varphi : G \times S \rightarrow S$  be a group action of a finite group  $G$  on a set  $S$  and let  $s \in S$ . Then we have*

$$|Orb_\varphi(s)| = \frac{|G|}{|Stab_\varphi(s)|},$$

where  $|Z|$  denotes the cardinal of a given set  $Z$ .

To conclude this subsection, we mention that an action of a group  $G$  on a set  $S$  defines an equivalence relation on  $S$  whose equivalence classes are given by the orbits. Since two equivalence classes are either equal or disjoint, the set of the orbits under the action of  $G$  forms a partition of  $S$ ; this is recalled in the following result:

**Theorem 1.** *Let  $\varphi : G \times S \rightarrow S$  be a group action of a group  $G$  on a set  $S$ .*

1. *Let  $s, t \in S$ . Then we have either*

$$\text{Orb}_\varphi(s) = \text{Orb}_\varphi(t)$$

or

$$\text{Orb}_\varphi(s) \cap \text{Orb}_\varphi(t) = \emptyset.$$

2. *Let  $\mathcal{R} \subseteq S$  be a set of orbit representatives, in other words a subset of  $S$  which contains exactly one element from each orbit. Then the family  $\{\text{Orb}_\varphi(s)\}_{s \in \mathcal{R}}$  forms a partition of  $S$ .*

An illustration of Theorem 1 is given in Figure 1.

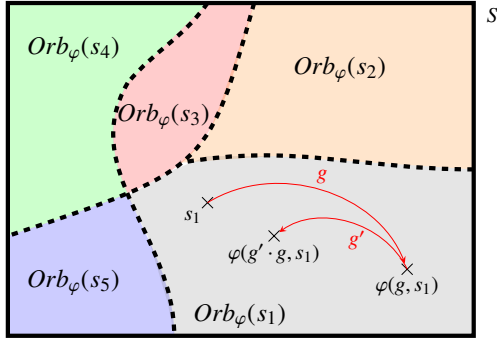


Figure 1: Illustration of a partition of  $S$  created by the group action  $\varphi$

### 3 Group Action Theory for Hill Cipher

We start this section by proving the following property of Hill inherited from its linear nature: it preserves the linear combinations of any given text. This is stated in the following proposition:

**Proposition 1.** *Let  $X = X_1 \dots X_m \in (\mathbb{Z}_p)^M$  be a plaintext. Suppose that the block  $X_i$  is a linear combination of  $q$  other blocks  $X_{i_1}, \dots, X_{i_q}$ , i.e.,*

$$\exists \lambda_1^{(i)}, \dots, \lambda_q^{(i)} \in \mathbb{Z}_p \quad X_i = \sum_{k=1}^q \lambda_k^{(i)} X_{i_k}. \quad (1)$$

Then the  $i$ -th block of the ciphertext  $Y = \mathcal{H}(A, X)$ , with  $A$  an element of  $\mathcal{G}_{M,n}$  associated with a key-matrix  $K \in GL_n(\mathbb{Z}_p)$ , satisfies

$$Y_i = \sum_{k=1}^q \lambda_k^{(i)} Y_{i_k}.$$

*Proof.* Since we have  $Y_i = KX_i \forall i \in \{1, \dots, m\}$ , it is sufficient to multiply equality (1) by  $K$  to obtain the result.  $\square$

Our goal in the present section is to study some consequences of this property. Especially, we shall prove the existence of a partition of the text-space due to the Hill cipher map. To do so, we shall exploit group action theory, whose principles will be illustrated in the setting of Hill, to structure our arguments. The results obtained here are at the root of the Orbit-Based Attack presented in the next section. Further, we mention that our work seems to formalise the principle of the approach developed in (McDevitt et al., 2018).

It is important to note that, the Hill cipher being a symmetric-key cipher, the results presented here can be interpreted in two ways: the relation  $Y = \mathcal{H}(A, X)$  can describe either the cipher of the plaintext  $X$  leading to the ciphertext  $Y$ , or the decipher of a ciphertext  $X$  leading to the plaintext  $Y$ . Therefore an input text  $X$  can be interpreted as a plaintext (resp. ciphertext) and the output text  $Y$  as a ciphertext (resp. plaintext) if we consider a cipher (resp. decipher).

We start our study by showing that the Hill cipher map given in Definition 3 is actually a group action.

**Theorem 2.** *The Hill cipher map given in Definition 3 is a group action.*

*Proof.* It is easy to show that the set  $\mathcal{G}_{M,n}$  is actually a subgroup of  $GL_M(\mathbb{Z}_p)$ , so it is itself a group. Further the *Identity* and *Compatibility* points of Definition 5 are satisfied thanks to the basic properties of matrix multiplication (multiplication by the identity matrix and associativity).  $\square$

From this theorem, it follows that the text-space is split into orbits which are stable under the Hill cipher map; in other words, if we choose an input text and we apply the Hill cipher map to it, then the resulting output text is necessarily inside the orbit of the input text, as illustrated in Figure 1 in an abstract setting.

**Corollary 2.** *1. Let  $X, X' \in (\mathbb{Z}_p)^M$ . Then we have either  $\text{Orb}_{\mathcal{H}}(X) = \text{Orb}_{\mathcal{H}}(X')$  or  $\text{Orb}_{\mathcal{H}}(X) \cap \text{Orb}_{\mathcal{H}}(X') = \emptyset$ .*

*2. Let  $\mathcal{X} \subseteq (\mathbb{Z}_p)^M$  be a set of orbit representatives, in other words a subset of texts which contains exactly one text from each orbit. Then the family  $\{\text{Orb}_{\mathcal{H}}(X)\}_{X \in \mathcal{X}}$  forms a partition of  $(\mathbb{Z}_p)^M$ .*

*3. For all  $X \in (\mathbb{Z}_p)^M$ , we have*

$$|\text{Orb}_{\mathcal{H}}(X)| = \frac{|GL_n(\mathbb{Z}_p)|}{|\text{Stab}_{\mathcal{H}}(X)|}.$$

*Proof.* Simple application of Theorem 1 and Corollary 1.  $\square$

According to Corollary 2, the number of elements of an orbit given by an input text  $X$  depends on the cardinal of the stabiliser of  $X$ . In the following proposition, we describe explicitly the stabiliser of any input text  $X$  by exploiting the property that the Hill cipher preserves linear combinations (see Proposition 1); in particular, this will permit to derive the cardinal of the orbit of  $X$  in Corollary 3. Further let us mention that we do not treat the case of the input text given by  $0_{(\mathbb{Z}_p)^M}$  since any matrix belonging to  $\mathcal{G}_{M,n}$  is in its stabiliser.

**Proposition 2.** *Let  $X = X_1 \dots X_m \in (\mathbb{Z}_p)^M \setminus \{0_{(\mathbb{Z}_p)^M}\}$ . Suppose that there exist  $1 \leq q \leq n$  and  $i_1, \dots, i_q \in \{1, \dots, m\}$  such that  $X_{i_1}, \dots, X_{i_q}$  are linearly independent and, for each  $i \notin \{i_1, \dots, i_q\}$ ,  $X_i$  is a linear combination of  $X_{i_1}, \dots, X_{i_q}$ . Then  $A \in \text{Stab}_{\mathcal{H}}(X)$  if and only if*

$$A = \begin{pmatrix} P\tilde{K}P^{-1} & & & \\ & P\tilde{K}P^{-1} & & \\ & & \ddots & \\ & & & P\tilde{K}P^{-1} \end{pmatrix},$$

with

- $P = (X_{i_1} | \dots | X_{i_q} | V_{q+1} | \dots | V_n) \in GL_n(\mathbb{Z}_p)$  where  $V_{q+1}, \dots, V_n$  are vectors of  $(\mathbb{Z}_p)^n$  such that  $\{X_{i_1}, \dots, X_{i_q}, V_{q+1}, \dots, V_n\}$  is a basis of  $(\mathbb{Z}_p)^n$ ;
- $\tilde{K} \in GL_n(\mathbb{Z}_p)$  is of the form

$$\begin{pmatrix} 1 & 0 & \dots & 0 & \tilde{k}_{1,q+1} & \dots & \tilde{k}_{1,n} \\ 0 & 1 & \dots & 0 & \tilde{k}_{2,q+1} & \dots & \tilde{k}_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & \tilde{k}_{q,q+1} & \dots & \tilde{k}_{q,n} \\ 0 & \dots & \dots & 0 & \tilde{k}_{q+1,q+1} & \dots & \tilde{k}_{q+1,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & 0 & \tilde{k}_{n,q+1} & \dots & \tilde{k}_{n,n} \end{pmatrix}. \quad (2)$$

If  $q = n$  then  $\text{Stab}_{\mathcal{H}}(X) = \{I_M\}$ , where  $I_M$  is the identity matrix of size  $M$ .

*Proof.* Choose  $X = X_1 \dots X_m \in (\mathbb{Z}_p)^M \setminus \{0_{(\mathbb{Z}_p)^M}\}$  and assume that  $X_{i_1}, \dots, X_{i_q}$  are linearly independent, where  $1 \leq q \leq n$  and  $i_1, \dots, i_q \in \{1, \dots, m\}$ , and that

$$\exists \lambda_1^{(i)}, \dots, \lambda_q^{(i)} \in \mathbb{Z}_p \quad X_i = \sum_{k=1}^q \lambda_k^{(i)} X_{i_k},$$

for all  $i \notin \{i_1, \dots, i_q\}$ . If  $q \neq n$ , choose  $V_{q+1}, \dots, V_n \in (\mathbb{Z}_p)^n$  such that the family  $\{X_{i_1}, \dots, X_{i_q}, V_{q+1}, \dots, V_n\}$  is a basis of  $(\mathbb{Z}_p)^n$ ; let us mention that such vectors exist according to the incomplete basis theorem (Artin, 2011, Proposition 3.15). Hence the matrix  $P$  defined in the statement of Proposition 2 is invertible and satisfies for all  $k \in \{1, \dots, q\}$ ,

$$PE_k = X_{i_k} \iff P^{-1}X_{i_k} = E_k, \quad (3)$$

where  $E_k$  is the  $k$ -th vector of the canonical basis of  $(\mathbb{Z}_p)^n$ . Furthermore, for a given matrix  $\tilde{K}$  of the form (2), the following relation is true for each  $k \in \{1, \dots, q\}$ ,

$$P\tilde{K}P^{-1}X_{i_k} = P\tilde{K}E_k = PE_k = X_{i_k}. \quad (4)$$

Then we deduce that, for all  $i \notin \{i_1, \dots, i_q\}$ ,

$$\begin{aligned} P\tilde{K}P^{-1}X_i &= P\tilde{K}P^{-1} \left( \sum_{k=1}^q \lambda_k^{(i)} X_{i_k} \right) \\ &= \sum_{k=1}^q \lambda_k^{(i)} P\tilde{K}P^{-1}X_{i_k} = \sum_{k=1}^q \lambda_k^{(i)} X_{i_k} \\ &= X_i. \end{aligned} \quad (5)$$

We are now in position to prove the equivalence stated in Proposition 2.2.

$\Leftarrow$  If a matrix  $A \in \mathcal{G}_{M,n}$  is given by

$$A = \begin{pmatrix} P\tilde{K}P^{-1} & & & \\ & P\tilde{K}P^{-1} & & \\ & & \ddots & \\ & & & P\tilde{K}P^{-1} \end{pmatrix},$$

where  $\tilde{K}$  is an invertible matrix of the form (2), then  $A$  satisfies

$$AX = \begin{pmatrix} P\tilde{K}P^{-1}X_1 \\ P\tilde{K}P^{-1}X_2 \\ \vdots \\ P\tilde{K}P^{-1}X_m \end{pmatrix} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{pmatrix} = X,$$

according to the relations (4) and (5). This proves that  $A \in \text{Stab}_{\mathcal{H}}(X)$ .

$\Rightarrow$  Let  $A \in \mathcal{G}_{M,n}$  be an element of the stabiliser of  $X$ . It follows

$$\forall k \in \{1, \dots, q\} \quad KX_{i_k} = X_{i_k},$$

where  $K \in GL_n(\mathbb{Z}_p)$  is the key-matrix. By using relation (3), we obtain

$$\forall k \in \{1, \dots, q\} \quad P^{-1}KPE_k = E_k.$$

We observe then that the matrix  $\tilde{K} := P^{-1}KP$  is of the form (2) and is invertible since it is similar to  $K \in GL_n(\mathbb{Z}_p)$ . This finally proves that

$$A = \begin{pmatrix} P\tilde{K}P^{-1} & & & \\ & P\tilde{K}P^{-1} & & \\ & & \ddots & \\ & & & P\tilde{K}P^{-1} \end{pmatrix}.$$

□

As a consequence of the preceding result, we are able to give the cardinal of the orbit of any input text, *i.e.* the number of texts which can be attained from this input. We note that this cardinal depends only on the number  $q$  of linearly independent blocks within the input text.



**Corollary 3.** Let  $X = X_1 \dots X_m \in (\mathbb{Z}_p)^M \setminus \{0_{(\mathbb{Z}_p)^M}\}$ . Suppose that there exist  $1 \leq q \leq n$  and  $i_1, \dots, i_q \in \{1, \dots, m\}$  such that  $X_{i_1}, \dots, X_{i_q}$  are linearly independent and, for each  $i \notin \{i_1, \dots, i_q\}$ ,  $X_i$  is a linear combination of  $X_{i_1}, \dots, X_{i_q}$ . Then we have

$$|\text{Orb}_{\mathcal{H}}(X)| = \prod_{k=0}^{q-1} (p^n - p^k).$$

*Proof.* First of all, we recall from (Rotman, 1965, Theorem 8.13) that the cardinal of  $GL_n(\mathbb{Z}_p)$  is given by

$$|GL_n(\mathbb{Z}_p)| = \prod_{k=0}^{n-1} (p^n - p^k).$$

Let us now compute the cardinal of  $\text{Stab}_{\mathcal{H}}(X)$ . According to Proposition 2, this is actually equal to the number of invertible matrices of the form (2), namely

$$\left( \begin{array}{c|c} I_q & \widetilde{K}_{1,2} \\ \hline \mathbf{0} & \widetilde{K}_{2,2} \end{array} \right).$$

Such a matrix being invertible, the sub-matrix  $\widetilde{K}_{2,2}$  is invertible as well; hence we have

$$\prod_{k=0}^{n-q-1} (p^{n-q} - p^k)$$

choices for the sub-matrix  $\widetilde{K}_{2,2}$ . Once this sub-matrix is fixed, it remains to choose  $\widetilde{K}_{1,2}$ , which does not have any restriction: thus there are  $p^{q(n-q)}$  choices for the sub-matrix  $\widetilde{K}_{1,2}$ . Consequently, we obtain

$$\begin{aligned} |\text{Stab}_{\mathcal{H}}(X)| &= p^{q(n-q)} \prod_{k=0}^{n-q-1} (p^{n-q} - p^k) \\ &= \prod_{k=q}^{n-1} (p^n - p^k). \end{aligned}$$

Finally, by using Corollary 2.3, it follows

$$|\text{Orb}_{\mathcal{H}}(X)| = \frac{\prod_{k=0}^{n-1} (p^n - p^k)}{\prod_{k=q}^{n-1} (p^n - p^k)} = \prod_{k=0}^{q-1} (p^n - p^k).$$

□

The previous corollary shows that the number of elements of an orbit given by an input text is always smaller or equal to the number of elements in the key-space  $GL_n(\mathbb{Z}_p)$ . Theoretically this means that if we consider an oracle able to answer in  $O(1)$  whether a matrix is the key or whether a text is the corresponding plaintext of the ciphertext of interest, then performing an exhaustive search on the key-space would be in  $O(\prod_{k=0}^{n-1} (p^n - p^k))$  and on the orbit of the ciphertext in

$O(\prod_{k=0}^{q-1} (p^n - p^k))$  with  $q \leq n$  ( $q$  being the number of linearly independent blocks in the ciphertext). Since in most of the cases  $q = n$ , this assures that these two kinds of search share the same worst-case complexity. The key idea of the OBA lying on a search on the orbit of the ciphertext, the preceding remark assures that it will be theoretically at worst as efficient as the BFA.

It remains to make practicable such a ciphertext-only attack. To do so, we have to describe explicitly the orbit of any given text. This is provided in the following theorem whose proof exploits once again the property that the Hill cipher preserves linear combinations; see Proposition 1.

As previously, we do not treat the case of the input text given by  $0_{(\mathbb{Z}_p)^M}$  since its orbit is equal to the singleton  $\{0_{(\mathbb{Z}_p)^M}\}$ .

**Theorem 3.** Let  $X = X_1 \dots X_m \in (\mathbb{Z}_p)^M \setminus \{0_{(\mathbb{Z}_p)^M}\}$ . Suppose that there exist  $1 \leq q \leq n$  and  $i_1, \dots, i_q \in \{1, \dots, m\}$  such that  $X_{i_1}, \dots, X_{i_q}$  are linearly independent and

$$\exists \lambda_1^{(i)}, \dots, \lambda_q^{(i)} \in \mathbb{Z}_p \quad X_i = \sum_{k=1}^q \lambda_k^{(i)} X_{i_k},$$

for all  $i \notin \{i_1, \dots, i_q\}$ . Then  $Y = Y_1 \dots Y_m \in (\mathbb{Z}_p)^M$  belongs to  $\text{Orb}_{\mathcal{H}}(X)$  if and only if  $Y_{i_1}, \dots, Y_{i_q}$  are linearly independent and

$$\forall i \notin \{i_1, \dots, i_q\} \quad Y_i = \sum_{k=1}^q \lambda_k^{(i)} Y_{i_k}.$$

*Proof.* Choose  $X = X_1 \dots X_m \in (\mathbb{Z}_p)^M \setminus \{0_{(\mathbb{Z}_p)^M}\}$  and suppose that there exist  $1 \leq q \leq n$  and  $i_1, \dots, i_q \in \{1, \dots, m\}$  such that  $X_{i_1}, \dots, X_{i_q}$  are linearly independent and

$$\exists \lambda_1^{(i)}, \dots, \lambda_q^{(i)} \in \mathbb{Z}_p \quad X_i = \sum_{k=1}^q \lambda_k^{(i)} X_{i_k},$$

for all  $i \notin \{i_1, \dots, i_q\}$ . Define now the set  $E_q(X)$  as follows:  $Y = Y_1 \dots Y_m \in (\mathbb{Z}_p)^M$  belongs to  $E_q(X)$  if and only if it satisfies

$$\left\{ \begin{array}{l} Y_{i_1}, \dots, Y_{i_q} \text{ are linearly independent} \\ \forall i \notin \{i_1, \dots, i_q\} \quad Y_i = \sum_{k=1}^q \lambda_k^{(i)} Y_{i_k} \end{array} \right.$$

Hence we have to show  $\text{Orb}_{\mathcal{H}}(X) = E_q(X)$ . To do so, we prove an inclusion and the equality between the two cardinals.

⊆ Let  $Y = Y_1 \dots Y_m \in (\mathbb{Z}_p)^M$  be an element of  $\text{Orb}_{\mathcal{H}}(X)$ . Then, by definition, there exists  $A \in \mathcal{G}_{M,n}$  such that  $Y = AX$ , i.e.,

$$\forall i \in \{1, \dots, m\} \quad Y_i = KX_i,$$

where  $K$  is the key-matrix. As an immediate consequence of the linear independence of

$X_{i_1}, \dots, X_{i_q}$ , the vectors  $Y_{i_1}, \dots, Y_{i_q}$  are linearly independent, and by Proposition 1, we have

$$\forall i \notin \{i_1, \dots, i_q\} \quad Y_i = \sum_{k=1}^q \lambda_k^{(i)} Y_{i_k}.$$

This shows that  $Orb_{\mathcal{H}}(X)$  is included in  $E_q(X)$ .

□ The cardinal of the set  $E_q(X)$  is given by the number of linearly independent families of  $q$  vectors belonging to  $(\mathbb{Z}_p)^n$ , that is to say

$$\prod_{k=0}^{q-1} (p^n - p^k).$$

We employ then Corollary 3 which shows that  $Orb_{\mathcal{H}}(X)$  and  $E_q(X)$  have the same cardinal.

This finally proves  $Orb_{\mathcal{H}}(X) = E_q(X)$ . □

The benefit of this result lies on the fact that it enumerates all the possible output texts from any given input text via the Hill cipher map: this permits to develop an algorithm for the OBA decrypting Hill cipher without studying the key-space. Moreover, even if it is unlikely in practice that the number  $q$  of linear independent blocks is strictly smaller than  $n$ , we expect an efficiency gain for the OBA in this case.

#### 4 Algorithm and computational complexity of the Orbit-Based Attack

The preceding theoretical results provide all the ingredients to create an algorithm for the Orbit-Based Attack, which consists in making a search in the orbit of the ciphertext of interest, and to determine its computational complexity. We emphasise that the algorithm proposed here is not intended to be optimised and some steps could be refined. The reader can find information on computational complexity in (Papadimitriou, 1994).

Throughout the rest of this section, we consider a ciphertext  $C$  of  $m$  blocks, each block having a size of  $n$  characters (with  $n \leq m$ ) in an alphabet of size  $p$ . Moreover we assume that the ciphertext has  $q$  linearly independent blocks (with  $q \leq n$ ), thus  $m - q$  dependent ones. The only condition we put here is that  $p$  is a prime number.

The main idea of the algorithm is to build sequentially the elements of the orbit of  $C$  by exploiting Theorem 3 until the plaintext associated with  $C$  is found. From a practical point of view, this can be achieved by placing firstly  $C$  in a  $n \times m$  matrix over  $\mathbb{Z}_p$ , called  $C_q$  and such that its  $k$ -th column corresponds to the  $k$ -th block of  $C$ , and by applying then the three following steps:

1. Performing a Gaussian elimination (Golub and Van Loan, 2012) column by column on  $C_q$  to make explicit the linear combinations of the blocks within  $C$ ; the same computations are

---

#### Algorithm 1: LU-type decomposition of $C_q$

---

**Data:**  $C_q$  the matrix storing the ciphertext,  $\text{Id}$  the identity matrix of size  $m \times m$

**Result:**  $C_q$  is lower triangular,  $\text{LC}$  contains the indices of the linearly independent blocks and the coefficient of the linear combinations

```

1 for ( $k=1, k \leq n, k++$ ) do
2    $j \leftarrow \text{firstNotNull}(C_q[k, i], 1 \leq i \leq n)$ ;
3   if ( $C_q[k, j] \neq 0$ ) then
4     divide column  $j$  by  $C_q[k, j]$  in  $C_q$  and in  $\text{Id}$ ;
5     swap columns  $j$  and  $k$  in  $C_q$  and  $\text{Id}$ ;
6     for ( $i=k+1, i \leq m, i++$ ) do
7       subtract to column  $i$  the column  $k$ 
8         multiplied by  $C_q[k, i]$  in  $C_q$  and in  $\text{Id}$ ;
9     end
10  end
11   $\text{LC} = \text{Id}$ ;
```

---

made on an identity matrix of size  $m \times m$  to store the indices and coefficients of the linear combinations.

2. Extracting the indices of the  $q$  independent blocks and the coefficients giving the  $m - q$  linear combinations.
3. Building the elements of the orbit of  $C$  in a random manner until the right plaintext is found. To do so, each element is built by first choosing randomly  $q$  independent blocks and the  $m - q$  remaining blocks are then deduced by using the linear combinations determined in the two preceding steps.

#### Lower-Upper decomposition of $C_q$

Lower-upper (LU) decomposition or factorisation factors a matrix as the product of a lower triangular matrix and an upper triangular matrix. It can be obtained by the Gaussian elimination and the factors contain the information on the linear dependencies of the columns. Let us specify step 1. with Algorithm 1 which performs an LU decomposition on the matrix  $C_q$ .

Let us determine the complexity of this initialisation step. Roughly speaking, at the  $k$ -th step, we search for a non-zero element on the  $k$ -th row (this step is omitted in the complexity computation), then we multiply the chosen column of size  $n - k + 1$  by the inverse of its non-zero element on the  $k$ -th row (such a computation is supposed to be cost-less) and we swap two columns if necessary, and we multiply the  $m - k$  remaining columns by scalars and make  $m - k$  additions. Furthermore, the same operations are made on the matrix  $\text{Id}$ : at the  $k$ -step, we work with columns of size  $k$ . We repeat these operations  $q$  times since the rank of  $C_q$  is equal to  $q$ . Adding the computational complexity of

each operation up gives:

$$\begin{aligned}
C_1(m, n, q) &= \sum_{k=1}^q \left( (m-k+1)(n-k+1) + (m-k)(n-k+1) \right) \\
&\quad + \sum_{k=1}^q \left( (m-k+1)k + (m-k)k \right) \\
&= 2(n+1) \sum_{k=1}^q (m-k) + (n+1) \sum_{k=1}^q 1 \\
&= 2(n+1) \left( mq - \frac{q(q+1)}{2} \right) + (n+1)q \\
&= 2nmq + 2mq - nq^2 - q^2.
\end{aligned}$$

### Obtaining the linear combinations within $C_q$

In the second step, we search in the output matrix LC the indices of the linearly independent blocks of the ciphertext  $C$  and the coefficients of the linear combinations. In view of the construction of the matrix LC in Algorithm 1, it is an upper triangular matrix up to a permutation if columns have been swapped. We note that  $q$  rows have been filled, corresponding to the  $q$  steps to make lower triangular  $C_q$ ; moreover, if the  $k$ -th column of the lower triangular matrix  $C_q$  is zero, then the  $k$ -th column of LC gives the coefficients of one of the  $m-q$  linear combinations within  $C$ . Hence determining the indices of the independent blocks of  $C$  consists in making searches in  $\text{Id}$ , such operations are supposed to be negligible in terms of computational complexity. Thus this second step is cost-less.

The algorithm that can perform this search is depicted in Algorithm 2.

### Recovering gradually the orbit of $C$

The two preceding steps combined to Theorem 3 permit to build all the elements of the orbit of the ciphertext  $C$ . Here we do not aim at recovering the whole orbit but rather to build element by element and to test whether the right plaintext is obtained. To do so, for each text to build, we choose randomly  $q$  linearly independent blocks of size  $n$  and we put them in the text in such a way that their indices are given by the matrix LC from the second step; this generation is supposed to be negligible. Then we compute the  $m-q$  associated linearly dependant blocks of size  $n$  from the  $q$  independent blocks by using once again the information contained in LC. This computation is depicted in Algorithm 3.

We observe that, for each linearly dependent block, we multiply the  $q$  independent blocks of size  $n$  by coefficients contained in LC and we add these  $q$  resulting blocks up to find the linearly dependent block. Supposing from now on that  $\tau_1$  tries are necessary to find the  $q$  right independent blocks of the plaintext  $P$ , we obtain then the computational complexity for the third step:

$$\begin{aligned}
C_3(m, n, q, \tau_1) &= \tau_1(m-q)(qn + (q-1)n) \\
&= 2\tau_1 mnq - \tau_1 mn - 2\tau_1 q^2 n + \tau_1 nq.
\end{aligned}$$

---

### Algorithm 2: Get the linear combinations

---

**Data:** LC the matrix storing the different linear combinations,  $C_q$  the lower triangulated  
**Result:** the indices of the free columns and the coefficient of the linear combinations.

```

1 size = 0;
2 ind = ∅;
3 for (i=1, i ≤ Cq.nbLines(), i++) do
4   | if (Cq[i][i] ≠ 0) then size = size + 1 ;
5 end
6 for (i=1, i ≤ LC.nbLines(), i++) do
7   | if (ind.size() == size) then return ind;
8   | for (j=1, j ≤ size, j++) do
9     | | if (LC[i][j] ≠ 0) then ind.add(i) ;
10    | end
11 end
12 for (col=ind.size(), col ≤ LC.nbCols(), col++) do
13   | for (line=1, line ≤ LC.nbLines(), line++) do
14     | | lineOK = true;
15     | | for (k=1, k ≤ ind.size(), k++) do
16       | | | if (ind[k] == line) then lineOK =
17         | | | false ;
18     | | end
19     | | if (lineOk ∧ LC[line][col] ≠ 0) then
20       | | | result[0][col-ind.size()] = line;
21     | | end
22   | end
23 for (i=1, i ≤ ind.size(), i++) do
24   | for (col=ind.size(), col ≤ LC.nbCols(), col++)
25     | | do
26       | | | result[i+1][col-ind.size()] =
27         | | | -LC[ind[i]][col];
28   | end
29 return result;

```

---

Note that  $\tau_1$  is bounded by the number of families having  $q$  linearly independent blocks of size  $n$ , namely,

$$\prod_{k=0}^{q-1} (p^n - p^k) \leq p^{nq}, \quad (6)$$

which can be very large. Nevertheless, we emphasise that if some prior knowledge on the text are available, such as the language, then the mean number  $\tau_1$  of tries can drastically diminish, reducing the computational complexity of this step.

Finally, the final cost of the Orbit-Based Attack (OBA) is:

$$\begin{aligned}
C^{OBA}(m, n, q, \tau_1) &= C_1(m, n, q) + C_3(m, n, q, \tau_1) \\
&= 2(1 + \tau_1)mnq + 2mq \\
&\quad + \tau_1 nq - (1 + 2\tau_1)nq^2 \\
&\quad - \tau_1 mn - q^2.
\end{aligned}$$

---

**Algorithm 3:** Re-build the linearly dependent blocks

---

**Data:** *res*: a randomly initialised matrix, *LC* the matrix storing the different linear combinations

**Result:** a potential plaintext where each linearly dependant block has been computed

```

1 for (c=1, c ≤ LC.nbCols(), c++) do
2   col = LC[0][c];
3   for (line=1, line ≤ LC.nbLines(), line++) do
4     res[line][col] = 0;
5     for (k=1, k ≤ ind.size(), k++) do
6       res[line][col] = res[line][col] +
7         (res[line][ind[k]] × LC[k+1][c])
8     end
9 end
10 return decrypt(res);

```

---

### Soundness, completeness and termination of the OBA

The Orbit-Based Attack to decrypt the Hill cipher is depicted in Algorithm 4. Note that the line 10 works as an odometer, meaning that if the loop does not stop, all the matrices will be generated.

---

**Algorithm 4:** Orbit-Based Attack

---

**Data:** *cipher*: the ciphertext that we want to decipher

**Result:** the corresponding plaintext

```

1 <in,LC> = Algorithm_1(cipher,Id);
2 ind = Algorithm_2(LC,in);
3 for (c=1, c ≤ LC.nbCols(), c++) do
4   for (l=1, l ≤ LC.nbLines(), l++) do
5     guess[l][c] = random() % (p);
6   end
7 end
8 guessMessage = Algorithm_3(guess,LC);
9 while (¬findWord(guessMessage)) do
10  guess = guess + 1;
11  guessMessage = Algorithm_3(guess,LC);
12 end
13 return guessMessage;

```

---

We prove now that Algorithm 4 satisfies the soundness, the completeness and the termination, which are the three main characteristics of an algorithm.

First let us prove the soundness, which means that if the algorithm gives an answer, then it is the expected one.

**Proposition 3.** *Algorithm 4 is sound.*

*Proof.* The soundness is a direct consequence of the use of the oracle “findWord”. Indeed, since the only way to return a solution is to exit the while-loop which

is possible only if “findWord” returns true. The solution is thus necessarily the corresponding plaintext. □

Now let us prove the completeness, meaning that the algorithm gives an answer for any input.

**Proposition 4.** *Algorithm 4 is complete.*

*Proof.* By having a look at the line 10 of Algorithm 4, we can see that each time we did not find the corresponding plaintext, we increment one value in the guess matrix, in the same way as in an odometer. By searching in such way all the matrices present in the orbit text for any ciphertext, we will perform an exhaustive search over its orbit, guaranteeing in such way the completeness of the algorithm. □

Finally let us prove the termination, stating that the time needed by the algorithm to terminate is finite.

**Proposition 5.** *Algorithm 4 terminates.*

*Proof.* The proof of termination is straightforward. There is no choose, nor backtrack in the algorithm. Moreover each loop iterates over finite domains: the columns or rows in matrices for Algorithms 1, 2 and 3, and the texts belonging to the orbit of the ciphertext for Algorithm 4. Therefore Algorithm 4, based on Algorithms 1, 2 and 3, terminates. □

## 5 Experiments

In this section, we present and comment on results from numerous experiments. Our aim here is to compare the new Orbit-Based Attack with the classical Brute-Force Attack, which is the only efficient ciphertext-only attack for the Hill cipher in case where no information on the text is available (Khazaei and Ahmadi, 2017).

### Computational complexity of the Brute-Force Attack

Before commenting on the experiments, let us talk about the Brute-Force Attack (BFA). We recall that it consists in testing all the invertible matrices of size  $n \times n$  over  $\mathbb{Z}_p$  until the right key is found.

Here we propose an algorithm for the Brute-Force Attack (BFA). For the sake of completeness, we inform the reader that our implementation of the BFA is quite naive. However, no matter the implementation, the empirical results should remain the same. Indeed, both approaches being mainly based on the same basic matrix operations, an optimisation for the BFA would also be an optimisation for the OBA as a side-effect. The Brute-Force Attack (BFA) is working as follows: we pick randomly a matrix of size  $n \times n$ , this generation being supposed negligible as previously. Such a random matrix has a high probability to be invertible, as stated in (Overbey et al., 2005). Moreover we suppose that we need  $\tau_2$  tries on average to find the correct matrix. It remains to multiply each  $m$  block of size  $n$  by

the chosen matrix of size  $n \times n$ . For each  $m$  block, it corresponds to make  $n$  multiplications and  $n - 1$  additions of columns of size  $n$ . We thus have:

$$\begin{aligned} C^{BFA}(m, n, q, \tau_2) &= \tau_2 mn(n + n - 1) \\ &= 2\tau_2 mn^2 - \tau_2 mn. \end{aligned}$$

Let us now remark that the quantity  $\tau_2$  is bounded by the number given in (6) with  $q = n$  and we have  $\tau_1 = \tau_2$  without any prior knowledge on the corresponding plaintext or on the key-matrix. In this setting, one can conclude that both the Orbit-Based Attack and the Brute-Force Attack are in  $O(p^{n^2})$ . Nevertheless these results are mainly theoretical and, in practice, one or the other attack may be faster in terms of time-execution, motivating the following subsections.

### Empirical results for the Orbit-Based and Brute-Force Attacks

In this section, we present empirical results generated on a cluster of Xeon, 4 cores, 3.3 GHz with CentOS 7.0 with a memory limit of 32GB and a runtime limit of 9,000 seconds per text per attack. Each text is uniformly randomly generated to avoid any statistical bias. The size of the text is given by  $n \times m$ , where  $m$  is the number of blocks and  $n$  the number of characters in each block, and we select randomly each character with a probability of  $\frac{1}{p}$ , where  $p$  is the size of the alphabet.

Here is our experimental protocol:  $p \in \{5, 7, 11, 13, 17, 19, 23, 29\}$ ,  $m = 100 : n = \{1, 2, 3, 4, 5, 6\}$ ,  $m = 200 : n = \{2, 4, 6, 8, 10, 12\}$ , ...,  $m = 900 : n = \{9, 18, 27, 36, 45, 54\}$

For each triplet  $\{p, m, n\}$ , we generate 300 different random texts. Thus we consider 129,600 different random texts, each of them is enciphered via the Hill cipher with random key-matrices. To compare properly the OBA and the BFA, we decipher each ciphertext with the two attacks; therefore 259,200 experiments are conducted with a time-out of 9,000 seconds, representing 73 years of computations in the worst-case.

First of all let us give the ratio of how many times the OBA has been faster than the BFA over the 129,600 different texts: it is equal to 99.91%. This shows that the OBA is faster in most of the cases considered here than the BFA. This first result shows that the OBA seems to furnish an attack for the Hill cipher more efficient than the classical OBA on the set of considered parameters.

To access the results, we redirect the reader to the following external link: <https://bit.ly/2QeYhH2>. They permit to compare the time-executions for the two attacks: in each sub-figure, each point represents a text whose x-coordinate and y-coordinates are respectively the time needed by the BFA and OBA. A point localised below the diagonal means that the OBA has been faster than the BFA. The first part of the Figures focuses on the parameters  $p$  and  $m$  (the different values for  $n$  are not distinguished) while the second part focuses on  $p$  and  $n$  (the different values for  $m$  are not distinguished). Basically, these Figures demonstrate that

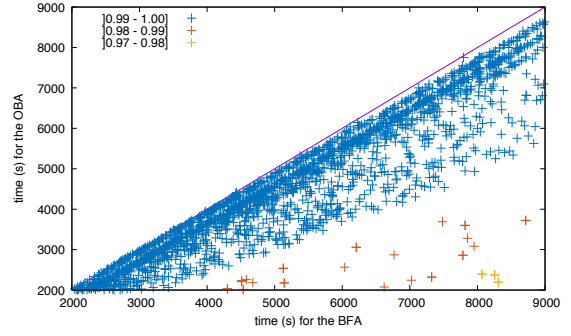


Figure 2: Scatter-plots of BFA vs OBA. Each dot corresponds to a random text where there is, at least, one block which is a linear combination of the others.

the time-execution tends to increase when  $m$  or  $n$  become large: this is logical since, in these cases, the text contains more characters and requires hence more computations to be decrypted. We also observe a small impact of the parameter  $p$  on the gain: the larger  $p$  is, the larger the number of points below the diagonal seems to be. This means that the OBA seems to be more efficient than the BFA when one studies ciphertexts in rich alphabets.

### Impact of the number of independent blocks

As explained in Section 2.2, the size of the orbit of a ciphertext  $C$ , whose influence on the computational complexity of the OBA can be seen through  $\tau_1$ , depends on the number  $q$  of linearly independent blocks within  $C$ ; by standard linear algebra arguments, we have  $q \leq n$ . Obviously the number of possible key-matrix does not depend on this parameter  $q$  and so, whatever the value of  $q$  is, the complexity of the BFA remains the same. Hence one expects the OBA deciphers faster on average ciphertexts having  $q < n$  independent blocks than the BFA does. Our aim here is to illustrate this fact in practice by using our experiments.

Min	1 <sup>st</sup> Qu.	Median	Mean	3 <sup>rd</sup> Qu.	Max
0.973	0.987	0.993	0.995	0.999	1.00

Table 1: Summary of  $\frac{q}{n}$  over all our examples.

For the sake of completeness, a simple statistical study of the ratio  $\frac{q}{n}$  over our 129,600 texts is given in Table 1. As expected, this ratio is generally close to 1 for our random texts, as expected, only few texts do not have the maximum number of independent blocks. One interesting issue would be to make the same simple study for meaningful texts in a given language.

In Figure 2, we observe the distribution of the time-executions but with respect to the ratio  $\frac{q}{n}$ ; note that the plot starts at 2,000 seconds. It is clear that the percentage of independent blocks impacts strongly the time-execution of the OBA: the fewer independent blocks the ciphertext has, the faster the decryption via OBA

is. While such texts are rare according to the above table, the OBA takes a huge advantage against the BFA in such cases, even in the case where the ratio is close (but different) to 1.

## 6 Conclusion

In this paper, we formalised the Orbit-Based Attack, whose principle has been firstly introduced in (McDevitt et al., 2018), by applying basic notions from group action theory; this provides a new type of ciphertext-only attack for Hill. This attack is based on the fact that Hill can not cipher an input text to any output one: the latter belongs necessarily to an explicit set associated to the former, namely its orbit, whose size is proved to be smaller than the one of the key-space. This attack consists then in making a search over only the orbit of the ciphertext.

We focused then on the computational complexity and time-execution of an algorithm for the OBA. Even if this algorithm has the same complexity as the one of the classical Brute-Force Attack (consisting in testing all the invertible matrices) in the worst case, our experiments show that this algorithm is faster on average than the Brute-Force Attack in practice. Discussions on the influences of some parameters of the text on the gain in terms of runtime of the OBA over the BFA are given. In particular, our theoretical and experimental results exhibit an interesting gain in the particular situation where the text has not the maximal number of linearly independent blocks.

We finish by discussing on the outlook. Considering ciphertexts for which some prior knowledge on the language are available is an interesting issue, reducing potentially the computational complexity of the OBA by using relevant statistical tools. A hope would be to refine the results obtained in (McDevitt et al., 2018) thanks our formalisation. A comparison with the Row-By-Row Attack from (Bauer and Millward, 2007; Yum and Lee, 2009; Leap et al., 2016) would be interesting as well.

We mention that some steps of the algorithm could be refined in view of optimisation of the attack. Moreover, except the LU-type decomposition, the rest of the algorithm can be parallelized, reducing potentially the runtime of the OBA. An empirical evaluation could be then performed as future works.

A last interesting issue would be to study whether the principle of the Orbit-Based Attack can be applied to other polygraphic substitution ciphers.

**Acknowledgments** Valentin Montmirail has been supported by IDEX UCA<sup>IEDI</sup>. The authors thank also T. Defourneau for discussions on group action theory.

## References

Michael Artin. 2011. *Algebra*, volume 2. Pearson Prentice Hall, Advanced Mathematics Series edition.

Craig P. Bauer and Katherine Millward. 2007. Cracking Matrix Encryption Row by Row. *Cryptologia*, 31(1):76–83.

Gene H. Golub and Charles F. Van Loan. 2012. *Matrix computations*, volume 3. JHU Press.

Lester Sanders Hill. 1929. Cryptography in an Algebraic Alphabet. *The American Mathematical Monthly*, 36(6):306–312.

Lester Sanders Hill. 1931. Concerning Certain Linear Transformation Apparatus of Cryptography. *The American Mathematical Monthly*, 38:135–154.

I. A. Ismail, Mohammed Amin, and Hossam Diab. 2006. How to Repair the Hill Cipher. *Journal of Zhejiang University-Science A*, 7(12):2022–2030, Dec.

Shahram Khazaei and Siavash Ahmadi. 2017. Ciphertext-only Attack on  $d \times d$  Hill in  $O(d \times 13^d)$ . *Inf. Process. Lett.*, 118:25–29.

Tom Leap, Tim McDevitt, Kayla Novak, and Nicolette Siermine. 2016. Further Improvements to the Bauer-Millward Attack On the Hill Cipher. *Cryptologia*, 40(5):452–468.

Ahmed Y. Mahmoud and Alexander G. Chefranov. 2009. Hill Cipher Modification Based on Eigenvalues HCM-EE. In *Proc. of SIN'09*, pages 164–167.

Tim McDevitt, Jessica Lehr, and Ting Gu. 2018. A parallel time-memory tradeoff attack on the hill cipher. *Cryptologia*, 42(5):408–426.

Jeffrey Overbey, William Traves, and Jerzy Wojdylo. 2005. On the Keyspace of the Hill Cipher. *Cryptologia*, 29(1):59–72.

Christos H. Papadimitriou. 1994. *Computational complexity*. Addison-Wesley.

J. Rotman. 1965. *The Theory of Groups*. Boston: Allyn and Bacon.

Jonathan D.H. Smith. 2008. *Introduction to Abstract Algebra*. Textbooks in Mathematics. Chapman and Hall/CRC.

Douglas Stinson. 2002. *Cryptography: Theory and Practice*. CRC/C&H, second edition.

Mohsen Toorani and Abolfazl Falahati. 2009. A Secure Variant of the Hill Cipher. In *Proc. of 14th IEEE Symposium on Computers and Communications (ISCC'09)*, pages 313–316.

Mohsen Toorani and Abolfazl Falahati. 2011. A Secure Cryptosystem Based on Affine Transformation. *Security and Communication Networks*, 4(2):207–215.

Samuel S. Wagstaff. 2002. *Cryptanalysis of Number Theoretic Ciphers*. CRC Press, Inc., Boca Raton, FL, USA.

Dae Hyun Yum and Pil Joong Lee. 2009. Cracking Hill Ciphers with Goodness-of-Fit Statistics. *Cryptologia*, 33(4):335–342.

# Cryptology in the Slovak State During WWII

**Eugen Antal**  
Slovak University of  
Technology in Bratislava  
Slovakia  
eugen.antal@stuba.sk

**Pavol Zajac**  
Slovak University of  
Technology in Bratislava  
Slovakia  
pavol.zajac@stuba.sk

**Otokar Grošek**  
Slovak University of  
Technology in Bratislava  
Slovakia  
otokar.grosek@stuba.sk

## Abstract

We explore the up-to now unknown details of the history of cryptology in Slovakia found in Slovak archives. This contribution focuses on cryptology of the Slovak State, which was a German puppet state during WW2. We identify three main types of ciphers in use. Firstly, ciphers from the former Czechoslovakia were adopted. During main military campaigns, the ciphers were mostly dictated by Germany. Finally, we describe a series of hand ciphers A-x specifically designed in Slovakia, mostly for internal use.

## 1 Introduction

The territory of modern Slovakia was for a long time a part of the Kingdom of Hungary. After the proclamation of the first Czechoslovak Republic on October 28, 1918, it has become a part of the new republic. A good overview of the situation of cryptology in the former Czechoslovakia is given by Š. Porubský in (Porubský, 2017).

On March 14, 1939, a separate Slovak State was created as a puppet state of the Nazi Germany. Czech territory was directly absorbed by Germany as a Protectorate. Former representatives of Czechoslovakia escaped to France, and later to the UK, to form the foreign resistance. Top intelligence officers of Czechoslovakia managed to escape to London along with intelligence files. However, the cryptology in Czechoslovakia, and later in London resistance movement was very weak, as it is demonstrated in books written by J. Janeček (Janeček, 1998; Janeček, 2001; Janeček, 2008), as well as in Cigáň's manuscript analysed by Š. Porubský (Porubský, 2017). Communications with the exile movement played an important role during the anti-nazi Slovak National Uprising that started in August 1944. The situation with

exile movement was complicated by cooperation with communist exile, which was connected to the Soviet Union, and partisan movement.

While the state of cryptology and secret communications of the exile government of Czechoslovakia are relatively well-known, as far as we know, the situation of the cryptography during the Slovak State was not studied in details yet. As mentioned, the Slovak State was a puppet state, and ally of Nazi Germany. The Slovak State declared war against German enemies, including the USA (curiously, there was never a peace treaty signed, because the Slovak State was not recognized in the aftermath of war). Slovak Army participated in military campaigns against Poland in 1939, and against the Soviet Union. In June 1944 remnants of the two Slovak divisions were disarmed due to low morale, and possibly due to mistrust by German command.

In our contribution we present some of cryptologic facts uncovered in the Military History Archive (part of the Institute of Military History established by the Ministry of Defence of the Slovak Republic). We want to clear a common misconception that the Slovak State cryptology was only directly dictated by Germany. We show some of the means of the cryptologic cooperation between German and Slovak armies, as well as some specific ciphers developed in Slovakia during WWII.

## 2 At the beginning of the war

In June 1939, the MNO - "Ministerstvo národnej obrany" (Ministry of National Defence) ordered the subordinate headquarters to report the list of officers with a cryptologic training. One month later it was ordered to report all the available ciphers and cryptographic directives. The goal of the ministry was to review the current state of secrecy in the newly established Slovak State.

From these reports we can conclude that the

available ciphers (and codes) belong to the pre-war era, namely:

- code "ZSD",
- hand-cipher "Q" (also called as key "Q"),
- cipher-machine (without any name in the archival documents).

All these ciphers (and codes) have been used before the war by the Czechoslovakian army<sup>1</sup>. We were unable to find any document mentioning the cipher-machine's name, but based on (Šklíba, 2007- 5; Šklíba, 2007- 7/8), only the ŠTOLBA cipher-machine was in use before 1938<sup>2</sup>.

The encryption service was reopened on 15th of July, 1939 - reusing the available ciphers. The hand-cipher "Q" was selected as the main cipher system. The document "Spojovací rozkaz č. 1" (Communication directive no. 1) from August 1939<sup>3</sup> was an order to encrypt all internal radio-telegraphic messages using this cipher. In the same month, on the 15th of August, 1939, the use of available cipher-machines was also (re)started. Document called *G-VII-8* named "šifrování" (encryption) was the main cryptographic directive in use with attachments describing the cipher systems such as the key "Q".

The available materials and directives show only internal use of these ciphers. Unfortunately it's not known, whether these ciphers were also used in a communication with the allies. This hand-cipher "Q" with the cipher-machine was still in use in December 1942, and the keys and passwords were distributed at least up to April 1943<sup>4</sup>. The daily keys for the cipher-machine were distributed for the whole year of 1943.

The *G-VII-8* was extended in 1943 with directives from Germany (without changing the name of the document). At this time the Slovak State also adapted some German ciphers including the Enigma - as described in the next section.

<sup>1</sup>Document 20.800/Taj.3.odd.1939 in (Military History Archive in Bratislava, 2019), fund KVV, box n. 2.

<sup>2</sup>ŠTOLBA is a cipher-machine with 6 main rotors and with 3 rotors controlling the rotor stepping. The daily keys distributed in 1939 for the "cipher-machine" also contains a 3 letter word and a 6 letter word.

<sup>3</sup>Document k. č. 77/39/Taj.3.odd.1939 in (Military History Archive in Bratislava, 2019), fund MNO tajné, box n. 2.

<sup>4</sup>Document 404621/Taj.obr.1942 in (Military History Archive in Bratislava, 2019), fund 53 (53-88/1-19).

### 3 Ciphers from Germany

"...The encryption is performed by the commander of the division using a cipher-machine. ... The cipher-machine is a box of dimension appr. 40x50 cm. The machine has keys like the typewriter and letters that lights during the encryption. The encryption is enabled by a 3-wheel system...  
Created in Germany (Berlin)."<sup>5</sup>

In September 1942 Ján Morvic completed a signal corps training in Germany (Nachrichtenschule, Halle). One part of the course was about secrecy, describing the German Enigma (without mentioning the name of equipment in the report).

In March 1943, an encryption training was designed<sup>6</sup> to learn the German hand-ciphers and the cipher-machine.

In April 1943, a new document called "návod k šifrovaniu" (manual of encryption) was created<sup>7</sup>. This document contained a description and instructions for two German hand-ciphers NS42<sup>8</sup> and TS42a<sup>9</sup> - introducing the German ciphers to the Slovak State departments.

Germany also gave an order<sup>10</sup> to unify the encryption among the allies. The Slovak State received directives for translation, extending the existing crypto-directive *G-VII-8*. The new directive consisted of four parts:

- general encryption rules (H.Dv.g.7) as *G-VII-8-a*,
- instructions to the "Enigma" (H.Dv.g.13, H.Dv.g.14) as *G-VII-8-b*,
- instructions to NS42 as *G-VII-8-d*,
- instructions to TS42a as *G-VII-8-c*.

When the German ciphers were in use, the daily keys were distributed monthly. There were 2 types of Enigma keys:

<sup>5</sup>Document 83375/spoj.2.1942 in (Military History Archive in Bratislava, 2019), fund MNO tajné, box n. 18.

<sup>6</sup>Telegram from 24. III. 1944 as an attachment to 2879/Dov.3./6.1944 in (Military History Archive in Bratislava, 2019), fund MNO dôverné, box n. 475.

<sup>7</sup>Document 7.632/Taj.3.odd.1943 in (Military History Archive in Bratislava, 2019), fund RD, box n. 45.

<sup>8</sup>Gen.StdH/Chef HNW IV.89 b 30 Nr.7.370/42.

<sup>9</sup>Gen.StdH/Chef HNW IV.89 b 30 Nr.7.360/42.

<sup>10</sup>Höherer Wehrmacht - Nachrichtenführer Mittelost Nr. 2241/geh.1942 referring to Gen.StdH/Chef HNW IV Nr. 8537/42.g.v.10.1942.



- marked as "Slovensko" (Slovakia) - to be used in the country,
- to be used with the allies.

The NS42 keys were distributed alongside with the TS42a, where the TS42a was designed to replace the NS42 in case of offensive army movement.

The knowledge of German ciphers among the Slovak signal corps officers wasn't on the required level, so there was an effort to train the staff to use these ciphers<sup>11</sup>.

#### 4 Design of a Slovak cipher

The Slovak State started the war with the available Czechoslovakian ciphers. Before adapting German ciphers and directives, the Slovak State developed own ciphers (called "A-2") for internal use. Note, that the cipher development was still overseen by Germany.

A new cipher called "A-2" was firstly introduced in May 1940. The cipher was described as a complicated transposition designed to encrypt 50 – 600 letters in a case of less-important radiograms. After the first distribution, all headquarters were asked to encrypt some radiograms with this cipher, and to send them to a corresponding place for the analysis<sup>12</sup>. The received reports describe the cipher as a practical, fast and secure enough<sup>13</sup>. It was also tested by the OKW Berlin. OKW Berlin allowed to use (see Figure 1, the stamp "Tajné" means "Secret") this cipher.

But the use of this cipher wasn't always without problems. Due to a large amount of errors made by encryption officers, it was ordered to re-train the use of the "A-2" cipher.

In 1941 a new directive for the encrypted communication was implemented replacing the previous one. Based on the directive, it was allowed to use only "A-2", the cipher-machine and the "ZSD" code. Most of the departments and battalions were allowed to use the "A-2" cipher only<sup>14</sup>.

Later on, in 1943, after the German ciphers were adapted, the Slovak one was still in use and

trained alongside with the German hand-ciphers and the cipher-machine<sup>15</sup>.

During the WW2 years, several upgrades/versions were created, the known versions are from "A-2" up to "A-5". The main contributor on the development was Michal Kmeřo-Dovina<sup>16</sup>.

#### 4.1 M. Kmeřo-Dovina

Michal Kmeřo-Dovina was the commander of the "hlavná voj. radiostanica MNO" (the main military radio-station of the Ministry of National Defence) and later on from 1943 worked as an encryption officer of second department of the Ministry of National Defence (VHU, 2013). He completed a cryptographic course "Písemné kursy kryptografie"<sup>17</sup> before WWII in 1938 - with a good score.

Due to a lack of officers experienced with encryption, in 1940, a creation of an encryption course was ordered. One of the instructors from this course was Michal Kmeřo-Dovina<sup>18</sup>.

During the Slovak National Uprising (SNP), Kmeřo-Dovina was helping the anti-nazi movement, keeping communication channels and developing new encryption system for the Uprising and later guerrilla fighters (VHU, 2013). From the available documents, it is not clear whether the development of the specific Slovak cryptographic systems was a part of the Uprising preparations. However, the cryptography during SNP is a very large and specific topic that is out of the scope of this article.

#### 5 The "A-x" hand-cipher

The "A-2" hand-cipher was developed as a first cipher from the "A-x" series. The cipher was updated several times during the years. Versions "A-2" and "A-3" consisting of 4 tables, and "A-4" consisting of 2 tables. We don't have detailed information about the other versions.

The "A-2" is a transposition cipher, designed to encrypt less important messages of length up to 600 letters. Main transposition was defined by a

<sup>11</sup>Document 2823/dov.spoj.1944 in (Military History Archive in Bratislava, 2019), fund MNO dôverné, box n. 475.

<sup>12</sup>Document 156.458/9.1940 in (Military History Archive in Bratislava, 2019), fund MNO dôverné, box n. 34.

<sup>13</sup>Document 135.992-II/9.Taj.1940 in (Military History Archive in Bratislava, 2019), fund MNO dôverné, box n. 34.

<sup>14</sup>Document 30.196/Taj.spoj.1941 in (Military History Archive in Bratislava, 2019), fund 55 (55-27-5).

<sup>15</sup>Telegram from 24. III. 1944 as an attachment to 2879/Dov.3./6.1944 in (Military History Archive in Bratislava, 2019), fund MNO dôverné, box n. 475.

<sup>16</sup>Documents 173.074/Taj.spoj.1941, 592/Taj.spoj.1941 and 22/Taj.1941 in (Military History Archive in Bratislava, 2019), fund MNO tajné, box n. 10.

<sup>17</sup>Document 151332-II/9.1940 in (Military History Archive in Bratislava, 2019), fund MNO dôverné, box n. 57.

<sup>18</sup>Document 151332-II/9.1940 in (Military History Archive in Bratislava, 2019), fund MNO dôverné, box n. 57.

series of secret tables, and each message had also a specific message key. It was forbidden to encrypt text of length under 50 letters. Each table has four logical sides — two are printed on the front page (the second one is upside-down version of the same page), and two on the back page. The logical side and the table's identification is marked with a red and black color on each side as *side/table*, later on in "A-4" this was flipped to *table/side*.

On each side, in the first row (header) of the table are two strings:

1. table identifier: 13 letters (unique for each table),
2. side identifier: 6 or 7 letters (different for each side).

The rows of the table are also labelled with one or two letters from the alphabet. Before encrypting a message, the message key is constructed from the letters identifying the order of tables, pages and starting positions within pages.

Before encrypting a message, several rules<sup>19</sup> were defined how to pre-process the input text:

1. Replace:
  - . with *X*,
  - : with *XX*,
  - , with *QW*,
  - . (full stop) with *XW*.
2. Write special characters with a full name, such as:
  - " as *UVODZOVKY* (*quotation mark*),
  - ( as *ZÁTVORKA* (*parenthesis*).
3. Write numbers with a full name, divided to digits:
  - 14 as *JEDNA ŠTYRI* (*one four*).
4. Replace accent, like:
  - *á* with *AA*,
  - *č* with *CV*.
5. When an accent is removed from a name, a letter *Q* should be put after this name.
6. When the text does not divide the number 5 a padding should be used (no longer as 4 letters) using some of the *QXW* letters or using the "STOP" word.

<sup>19</sup>Document 164/Taj.spoj.1941 in (Military History Archive in Bratislava, 2019), fund MNO tajné, box n. 10.

The cipher "A-2" had 4 tables. Later on, in version "A-4", this was reduced to only 2 tables. As "A-4" encryption mechanism is otherwise the same as "A-2", for the sake of simplicity, we will continue the description with "A-4" procedure. In Figure 2 we show one of the pages showing two sides of one table.

To start an encryption, sender of the group created a message key called "skupina oznamovatel'a" (*sender group*), consisting of 5+5 letters. He selected

1. the order of the tables,
2. the order of the four available sides for each table,
3. the offset (row identifier), defining where to start writing the text,

and encoded his selection as a group of first 5 letters. The first letter of this group contains a randomly chosen letter from the identifier label of the first selected table, and the remaining four letters are randomly chosen letters from the labels of the sides (in the corresponding order). During the encryption, after the four sides of the first selected table were used, the encryption would automatically continue with the remaining table. The second 5 letters were used to encode the row-offset identifier, and then the order of the sides of the second table.

As an example (using the Figure 2), we can choose to start with the table marked 2. We select randomly one letter from the selected table identifier string "RKGJXDOQHFB". If we want to start the encryption from side 2 (marked 2/2), we select randomly one letter from the side identifier "JLIXZA". Next we choose the side 1 (marked 2/1) as the second side, selecting a letter from "CURKTG", and so on.

The encryption table itself consists of small rectangles forming a matrix, where some of the cells are cut off. This is essentially a variant of the Fleissner grille. Each table realizes multiple  $5 \times 5$  grilles in parallel, randomized by a message key.

The plain-text letters are filled into these cells based on the message key. The text is written in an order defined by the red arrow painted on the corresponding side. On every side, we start to fill the first available (cut off) cell on the row labelled with the chosen row identifier. It is necessary to note, that in case of a shorter text, only a part

of the matrix is used (not ending on the last free cell). The encryption grille was designed to automatically form five letter groups of the encrypted text. On each side, there are columns labelled with numbers. Since the first row is known, we can save the last number from the previous row. If we add the plain text length (after the pre-processing) to this number, we obtain the position of the final cell.

To continue in our example, we choose the row offset "Q" (starting in the seventh row). The last number from the previous row is 180 (red color). Suppose the text length to be 50 letters long. Our ending position will be  $180 + 50 = 230$ , so we cannot use cells after this position.

The decryption is straightforward, using the same order of tables, sides and using the same cell range.

For the interested reader, we include an example of the encrypted telegram (Figure 3), with the following transcript:

```
VVXKW QUKQW
PZJXY AVTQA ZVRPO DAOSV PLLQA XIMVS
UOLNT IURTC DVEAL ATSNE WPDSE EUOPU
LSOTR OYKOJ UAOXV ECDTQ AKXLS JSSPD
SCAXR VPEAI RVIOT UOXAO SXNRK ESAPU
```

## Acknowledgments

This work was partially supported by grant VEGA 1/0159/17.

## References

- Military History Archive in Bratislava (Vojenský historický archív v Bratislave).
- Jiří Janeček. *Gentleman (ne)čtou cizí dopisy* (in Czech). 1998. Books - bonus A. ISBN:8072420232.
- Jiří Janeček. *Válka šifer* (in Czech). 2001. Votobia. ISBN:8071985058.
- Jiří Janeček. *Rozluštěná tajemství* (in Czech). 2008. XYZ. ISBN:8086864545.
- Štefan Porubský. Application and Misapplication of the Czechoslovak STP Cipher During WWII. 2017. *Tatra Mountains Mathematical Publications*, 70(1):41–91.
- Karel Šklíba. Z dějin československé kryptografie, část I., Československý šifrátor MAGDA (in Czech). 2007. *Crypto-World*, (5).

Karel Šklíba. Z dějin československé kryptografie, část II., Československý šifrovací stroje z období 1930-1939 a 1945-1955 (in Czech). 2007. *Crypto-World*, (7/8).

František Cséfalvay et al. 2013. *Vojenské osobnosti dejín Slovenska 1939-1945* (in Slovak). Vojenský historický ústav Bratislava. ISBN:9758089523207.

# Appendices

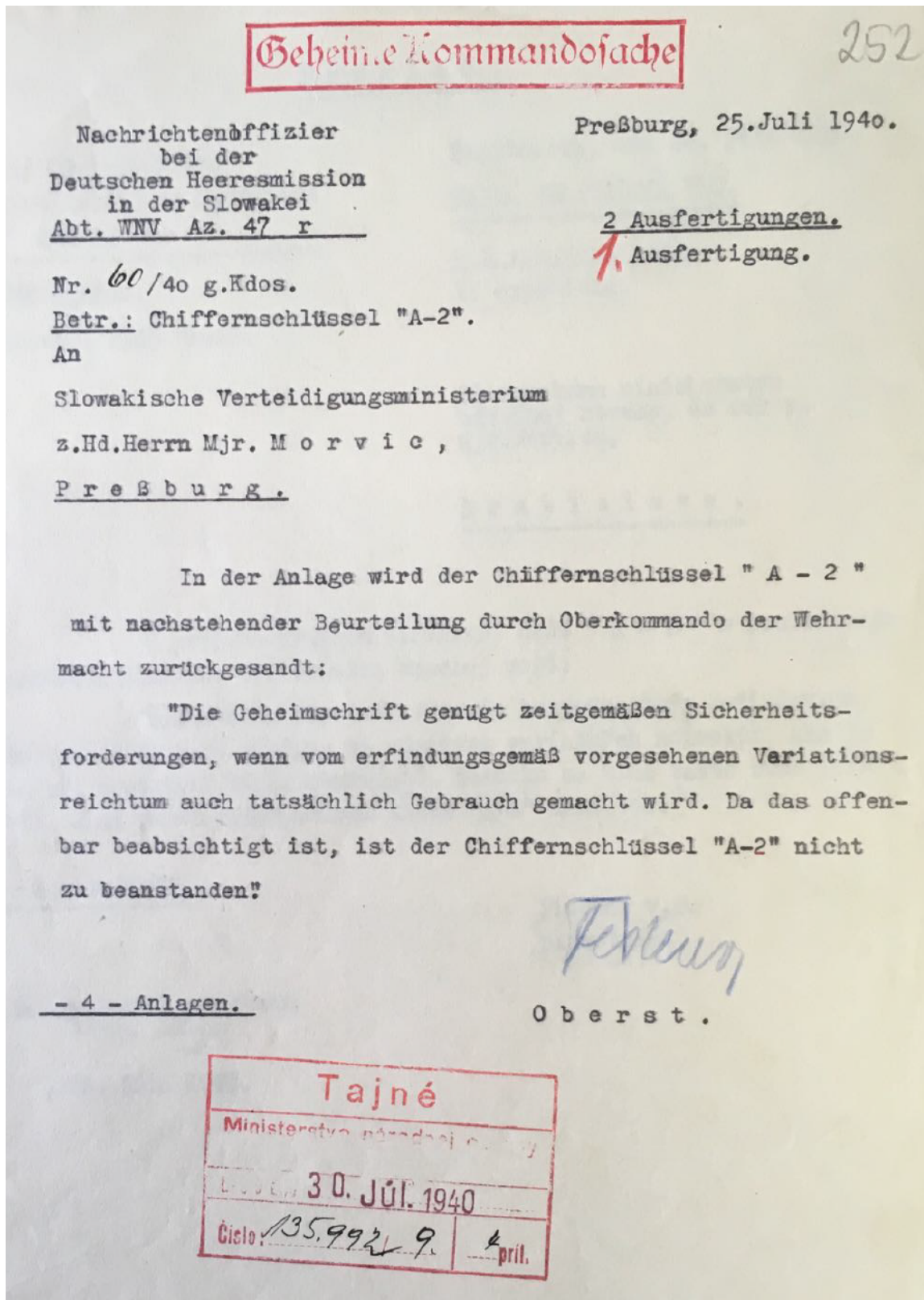


Figure 1: Report of the "A-2" from the OKW Berlin - in (Military History Archive in Bratislava, 2019), fund MNO dôverné, box n. 34.

4	2/1 = RKGJXZDOQHFB	= CURKTG	Tajné	1		
KE	5	10	15	20	25	30
CN	35	40	45	50	55	60
SW	65	70	75	80	85	90
AB	95	100	105	110	115	120
Z	125	130	135	140	145	150
DJ	155	160	165	170	175	180
Q	185	190	195	200	205	210
T	215	220	225	230	235	240
U	245	250	255	260	265	270
IY	275	280	285	290	295	300
V	305	310	315	320	325	330
M	335	340	345	350	355	360
LG	365	370	375	380	385	390
O	395	400	405	410	415	420
R	425	430	435	440	445	450
H	455	460	465	470	475	480
X	485	490	495	500	505	510
F	515	520	525	530	535	540
P	545	550	555	560	565	570
KE	575	580	585	590	595	600
4	2/2 = RKGJXZDOQHFB	= JIIXZA	Tajné	4		

Figure 2: The "A-4" hand cipher (Table 2, sides 1 and 2) - in (Military History Archive in Bratislava, 2019), fund MNO tajné, box n. 10.

545

Stanice (ústředna): .....

Došel	T e l e g r a m				Odeslán
z .....	do .....	z .....	pres .....	do .....	
dne .....	Číslo	Třída	Slov	Čas podání	dne .....
hod. ....					hod. ....
telegrafista .....	dne..... hod.....				telegrafista .....
Služební poznámka .....	Příkaz podatele .....				Služební poznámka .....
SKUPINY OPEROVATELA					
I. ✓ → II					
V <del>Q</del> V X K W Q Ü K Q W					
P Z J X Y	A V T Q A	Z V R P O	D A O S V	P L L Q A	X I M V S
U O L N T	I Ü R T C	D V E A L	A T S N E	W P D S E	E Ü O P Ü
L S O T R	O Y K O J	Ü A O X V	E C D T Q	A K X L S	J S S P D
S C A X R	Y P E A I	R V I O T	Ü O X A O	S X N R K	E S A P Ü

Figure 3: Text encrypted with the "A-4" hand cipher - in (Military History Archive in Bratislava, 2019), fund MNO tajné, box n. 10.

# The Typex Scare of 1943: How Well Did the British React to a Cypher-Security Scare?

**Dermot Turing**  
Visiting Fellow  
Kellogg College  
62 Banbury Road  
Oxford OX2 6PN, UK  
dermotturing@btinternet.com

## Abstract

The response of German Naval Intelligence, at various points in World War Two, to suspicions that the Enigma cipher had been broken is well known. In 1943 the British were faced with evidence about the possible compromise of the Typex machine, the highest-level communications device in use across their armed forces. This paper compares the response of the British to the Typex scare to the German investigations concerning Enigma.

Reconstructing the story, it seems that the Germans had, in fact, read some Typex messages. Although Allied code-breaking during the war operated on a higher plane than Germany's, it was inappropriate to assume that the Germans could not do so. Various similarities between the German and British responses emerge: the British were ill-adapted to investigating their own security; they were reluctant to chase down the truth, using arguments to justify their desire to be reassured that all was safe.

## 1 Introduction

The British answer to high-security enciphered radio communication was the Typex machine. Typex has been discussed elsewhere (Erskine, 1997; Ferris, 2005). Suffice it to say here that was an electro-mechanical machine, whose principle, as with the more famous Enigma machine, was a set of wired rotors each of which

switched a letter of text for another, with the rotors stepping on to create a different cipher overlay each time a key was pressed. Additional security features included settable entry and exit 'stators', rotor-cores which could be inserted backwards so as to invert their left-right behaviour, a set of ten rotors from which to choose five, and in some versions a plugboard (like the German military version of Enigma).

All this was good for British communications security, but there were still vulnerabilities. First, the British armed services were not universally equipped with the plugboard version of Typex. Then there were elementary errors of cryptographic security, such as (in particular) the operational error where the machine operator chose as his start-position for encipherment a non-random orientation of rotors based on the end-position of the preceding message – facilitating a cryptanalytical attack known to Bletchley Park as 'Herivelismus'.<sup>1</sup> Together with the risk of capture of a machine, its operating instructions, and rotors, there was a high risk that the German code-breaking organisations in the Oberkommando der Wehrmacht (German Armed Forces General Staff), the Oberkommando des Heeres (German Army General Staff) or the Luftwaffe Chi-Stelle (German Air Force Cipher Bureau) would be able to crack messages sent in Typex.

Despite this, the idea has persisted that the Germans never broke Typex (Hinsley et al., 1981; TICOM, 1946, vol 4). That received view is not clearly wrong, but there was certainly a point

---

<sup>1</sup> TICOM document D-83, UK National Archives (TNA) HW 40/87.

during the war when it appeared that the Germans could read Typex messages and had done so. This paper examines the evidence which the British had before them, and how the British reacted. A comparison can then be drawn to the – perhaps notorious – lack of effective response on the German side to a series of equivalent scares about the security of Enigma (cf. Mulligan, 1985; Ratcliff, 1999, 2006).

## 2 The Typex Scare of 1943

The news came out, as news does, in bits and pieces. First there was an alarming message from Bertie. ‘Bertie’ was the name given to Commandant Gustave Bertrand, the signals intelligence officer from the French Army’s ‘Deuxième Bureau’ who had managed somehow to re-establish himself under the Vichy régime and run a small code-breaking operation, which despite everything remained in contact with the British Secret Intelligence Service, the parent of the Government Code & Cypher School at Bletchley Park. As well as running a team of Polish, Spanish and French code-breakers, Bertie had his ear to the ground in many ways, receiving news relating to radio communications security and similar issues from a range of sources.

On 19 July 1942, Bertie signalled to the British that there were reports that the German Air Ministry was using adapted versions of two ‘English cipher machines’ captured at Dunkirk.<sup>2</sup> It is not difficult to imagine that the British had lost a Typex machine in the chaos of the Dunkirk evacuation, and indeed they had.<sup>3</sup> However, Typex would be difficult to crack without the rotors, and with a set of ten rotors from which to choose five (as opposed to the three when Marian Rejewski achieved his feat of reverse-engineering the Enigma machine in 1932) the reconstitution of the wiring of Typex rotors would have been a herculean task without a physical capture; it seems that the Germans did not manage to capture Typex rotors at Dunkirk. (This probably reflects the fact that the Typex equipment was too bulky to be easily moved, so had to be left behind, whereas the rotors could be much more easily transported or disposed of.)

---

<sup>2</sup> TNA HW 40/88.

<sup>3</sup> There are many TICOM interviews which stated this, for example, D-83 (TNA HW 40/87), D-40 (US National Archives and Records Administration (NARA) RG 457 HMS P11 Box 24).

Whether Bertie’s message needed to be taken seriously, amid all the noise of contrary intelligence heard in the cacophony of war, is difficult to judge even with hindsight. But in the same month there was another snippet of news. A certain Dr Vögele had, according to a German signal intercepted and decrypted at Bletchley Park, been sent to visit cipher material captured following the fall of Tobruk to Rommel: the signal was sent only five days before Bertie’s telegram, and only just over three weeks after the capture of Tobruk. The British probably did not know at that stage that Vögele was a senior cipher specialist (‘Regierungsrat’) in the Luftwaffe’s Chi-Stelle, thus establishing a link to Bertie’s note which showed that German Air Force intelligence could be engaged in an attack on Typex security. A note on the telegram, apparently in the handwriting of Lt Cdr Russell Dudley-Smith RNVR, the officer at Bletchley Park tasked with cipher security questions, says ‘no typex machines forward of Gambut [a military airfield complex in Libya] during recent fighting’: maybe, then, this was more noise.<sup>4</sup>

The third tiny piece of the jigsaw-puzzle came on 15 August. ‘The following message in code has been received from a British Prisoner of War in GERMANY... COLONEL STEVENSON C.S.O. S. AFRICA DIVISION REPORTS THAT ALL CODES CIPHERS TELEX MACHINES AND DRUMS DESTROYED TOBRUK BEFORE JERRY’S ENTERED.’<sup>5</sup> Why would a prisoner go to such lengths to report that what ought to have been done, had been done, unless perhaps there was a question about it?

## 3 The British response

Whatever the immediate cause, something seems to have prodded the British into action on Typex security. It may have been an intercept. On 20 January 1943, the commanding officer of a signals intelligence regiment, believed to be stationed in Greece, wired ‘OKH/In 7/VI’ asking whether the recipient dealt with a certain kind of British five-letter traffic. ‘In 7/VI’ was the cryptanalytic division of the Oberkommando des Heeres, and it was noted that this was the first mention in secret signals of what seemed to be a programme of interception of Typex messages.

---

<sup>4</sup> TNA HW 40/88.

<sup>5</sup> TNA HW 40/88.



To begin with, Alan Turing was asked to advise on the maximum secure message length for a Typex signal – on the basis that long signals should be split into pieces, each enciphered using a different rotor start-position, thereby reducing the exploitability of ‘cribbing’ to reveal the settings in use. Turing reported on 10 July 1943, ‘it seems that 1000 letters would not be too long with the form of the machine with a pluggable Umkehrwalze [reflector rotor], but that with the other form of the machine the question turns on the crib-avoiding discipline’.<sup>6</sup> The problem which this short sentence reveals was that the British Typex machine was fundamentally insecure unless they were using the pluggable reflector. In any case, Turing’s assessment was hardly a thoroughgoing examination of security relating to Typex.

What the sentence from Turing’s paper does not reveal is that the Typex machine existed in a form which lacked not just a pluggable reflector, but any kind of plugboard at all. At least some of the time, the British Army in North Africa was using only the simplest version of the Typex machine without the all-important plugboard. Alan Turing – who had worked on the Enigma problem in the earliest years of the war had just confirmed the ease of breaking a rotor-based cipher machine based on cribbing. Dilly Knox had broken such machines from the mid-1930s onwards. Post-war German interviews and documents confirm that the Germans knew how to do this too.<sup>7</sup> Worse still, Commander Edward Travis, then deputy head of the GC&CS, had explained in 1940 that he had concerns about Typex.<sup>8</sup> The central problem with Typex in 1943 was that the British were ignoring what they themselves knew about rotor-machine security: allowing the services to use a bad machine with lax operational practice, a combination of affairs which surely ought to have rung alarm bells.

The bells were not going to ring, though, without an accumulation of evidence which was overwhelming, and that would only happen if the Germans had actually broken Typex and then told the British that they had done so. This is, in practice, what happened when the Allies got notice that Colonel Fellers’s ‘black code’

messages back to Washington from Cairo were being read, and when the Allies got notice that the Royal Naval Cypher number 3 was being read by the German Navy’s B-Dienst with appalling consequences for the security of convoys in the North Atlantic (Tighe, 1945). Thus, by 1943 there was a growing body of concern that not all was well in the world of Allied signals security. The most damning case regarding Allied army and air force signals was presented by the Germans themselves, when their field signals security regiment NFAK (Nachrichtenfernaufklärung) 621 was captured in Tunisia. Not only did the prisoners explain that American signals security on the battlefield (where lower-grade ciphers were in use) was rotten, but there was a fresh set of indications that Typex itself might have been read. It only required the Germans to keep quiet about their successes for the British complacency to continue.

During the final stages of the battle for Tunisia in May 1943 the German signals unit NFAK 621 was among those who surrendered to Allied forces. NFAK 621 had a difficult history. On the one hand, the unit had been immensely successful at providing Rommel with real-time signals intelligence, based on both traffic analysis and in-the-field cryptanalysis, but on the other hand its star officers and much of the unit had been captured in July 1942 during the early stages of the second Alamein battle. Now even more men from the rebuilt unit had once again fallen into Allied hands. Lieutenant Bode was interrogated in June 1943, and revealed that he had been engaged on translating and emending British machine messages from 1937 until June 1940. The interrogator asked what kind of a machine; Bode said ‘a sort of typewriter. A man just typed the nonsense stuff, and the English came out on a tape.’ That sounded rather like a Typex. The intelligence captain from MI8(a) who interrogated Bode added that ‘Unfortunately, BODE, at that time, was a very junior N.C.O., and the knowledge of the machine was very restricted. It was in fact treated very much as our own CX/MSS knowledge.’ (CX/MSS was the designation to intelligence received from ‘most secret sources’, in other words decrypts resulting from cryptanalysis.)<sup>9</sup>

But ‘the trouble with BODE is that he is trying to tell us more than he knows and is only too

---

<sup>6</sup> TNA HW 40/87.

<sup>7</sup> OKH In 7-VI Kriegstagebuch for July 1941, Politisches Archiv Berlin, TICOM collection S8 (PA-S8), nos T-2755 to T-2764; TICOM document D-83, TNA HW 40/87.

<sup>8</sup> TNA ADM 223/505.

---

<sup>9</sup> TNA HW 40/88.

ready to agree with anything one says.’ On the other hand, ‘there is a possibility of obtaining confirmation of BODE’s story from other members of 621 Intercept Coy. when they arrive in this country for interrogation.’ So, another pair of prisoners were interrogated on 23 August 1943. One – Leutnant Haunhorst – had been a divisional intelligence officer working closely with NFAK 621. The other – Oberleutnant Possel – was a senior radio officer in the 10<sup>th</sup> Panzer Army Intelligence Regiment.<sup>10</sup> Independently, these two officers said that one or more Typex machines had been captured at Tobruk, and a certain ‘Warrant Officer Wagner’ using ‘reference books’ containing settings had been able to set the machine and decode messages. Some of the reference books came from the ‘Haupt Chiffrier Stelle OKH’ – the Head Cipher Office of the Oberkommando des Heeres.<sup>11</sup>

Obviously, further action was needed. But the investigation was, almost perversely, directed in the wrong way. Rather than look at the vulnerability of Typex, it seems that evidence to confirm the security of Typex was sought out.

#### 4 Confirming confirmation bias

So the first thing actually done was to track down what had happened to the Typex machines at Tobruk. Questionnaires were sent out and an inquiry as to destruction procedure was undertaken. ‘Navy report no typex equipment held by RN ships or staffs using Tobruk. RAF report no machines or drums [rotors] held RAF in Tobruk relevant dates. They add all drums held RAF during retreat to Alamein safely returned.’ ‘Your [question] five One set black drums Number 1270 handed over on authority CSO 8 Army to Captain MacFarlane Cipher Officer 2 SA Division reported by latter destroyed night before Tobruk file reference 8Army X2/883 of 20 June 42. Destruction certificate black drums 1270 based on this cipher message which stated all cipher equipment except one “W” Book one local recyphering table destroyed.’<sup>12</sup>

Another task was to locate the other members of NFAK 621 who might be able to cast further light on the alleged reading of Typex in North

Africa, as revealed by Possel and Haunhorst – ideally the Warrant Officer called Wagner. After some months, ex-NFAK 621 prisoners Habel and Bremer were located, but there was no trace of any Wagner. Habel, who had been the commander of NFAK 621 at the time of its capture, was transferred from the United States to London together with Bremer, where they arrived on 22 December 1943, and interrogated. ‘Although every means possible were used to induce these two men to talk, their inherent security which is of an abnormally high standard has completely defeated normal methods of approach.’<sup>13</sup> So the British were none the wiser. Plus, the proper procedures had been followed at Tobruk: destruction certificates had been prepared, which should not have happened if the Typex equipment had not been properly destroyed, so clearly prisoners like Bode, Haunhorst and Possel must be mistaken or attempting to mislead.

Indeed, it was quite possible that they had been mistaken. Various forms of rotor-based cipher equipment was being used in North Africa. If Typex were being read regularly, through cryptanalysis rather than capture of a few weeks’ worth of settings, surely a more robust response to Allied plans would have been experienced on the ground, whereas it had been possible to take the Axis by surprise in relation to major operations like TORCH (the landings in French North Africa) and HUSKY (the invasion of Sicily). By the spring of 1944, with the preparations for the main invasion of continental Europe well under way, those events seemed a long time ago, and the Typex scare of 1943 something one could stop worrying about.

It was therefore, perhaps, not surprising to find Gordon Welchman of Bletchley Park reaching that conclusion, even before Habel and Bremer had been grilled. Welchman was not only highly intelligent and highly persuasive but he also carried the confidence of Commander Travis, who by this time was head of Bletchley Park. Travis appointed Welchman to lead a Machine Coordination and Development Section in September 1943, which meant that among other things Welchman was (occasionally – there was a question over his terms of reference) in charge of security of cipher machinery. Welchman’s memo is interesting:

<sup>10</sup> TICOM document I-16, TNA HW 40/89.

<sup>11</sup> TNA HW 40/88.

<sup>12</sup> TNA HW 40/88.

<sup>13</sup> TNA WO 208/5109.

This story about the mysterious Wagner sounds like a garbled account of something true. I imagine that, having captured a few Type X machines, the Germans would have the sense to maintain a forward decoding party to take full advantage of any captured keys, but it seems unlikely that we should have lost any Type X keys in Tunisia and the story suggests that there is far more in it than that....

As regards breaking, I have always felt that the Germans could not be breaking any of our Type X traffic because, if they were, they would take steps to prevent us breaking their enigma traffic....

I have never thought seriously about possible methods of breaking Type X, but should have guessed that the equipment necessary would be pretty bulky unless the problem is being simplified by extreme carelessness....

On the whole I feel that a thorough investigation would be a good thing, but I don't see who could do it. However it may be possible to shoot down the Wagner story after further discussion here and further interviews with P.O.W's. It is quite possible that Haunhorst was merely shown how the Type X machine worked, and it would be interesting to know whether he actually saw an English message decoded.<sup>14</sup>

So the idea of an investigation into Typex security was not pursued. Despite being told that the Typex project was clothed with the utmost secrecy within the German radio intelligence regiment, the idea that 'Wagner' might have been a cover-name for the Warrant Officer actually involved does not seem to have occurred to those involved in the interrogations. The British had decided to look away from the possibility of 'extreme carelessness' and rely on the specious idea that 'if they were, they would take steps to prevent us breaking their enigma traffic'. But this argument was nonsense: putting it in reverse, the British were taking no cryptanalytical steps (despite the success against Enigma) to check up on Typex, and using that as an illogical excuse to take no steps to protect Typex.

In a note of 3 June 1944, Lt Cdr Dudley-Smith stated that 'Five months of interrogation has produced no additional information on German exploitation of Typex in N. Africa.' He sounds

weary of the whole business. And that was how things stood, until after the war was over.<sup>15</sup>

## 5 TICOM

As is now well known, a 'Target Intelligence Committee' (TICOM) was set up in the summer of 1944 to track down German codebreakers and interrogate them as to their successes or otherwise (Rezabek, 2017; Jackson, 2013; TICOM, 1946). Over the course of the months and years following the invasion of Germany, many German individuals were asked to describe their attacks on Typex and the successes which they had had.

Some of the answers were confusing, and some prisoners seem to have changed their testimony. However, the consistent story from the team at OKH In 7/VI (who had had Typex on their to-do list for years) was that they had initially put a great deal of effort into the attack on Typex, knowing that they could reconstitute the keys through cribbing, if they knew the wiring in the coding rotors; that they could reconstitute the stepping arrangement of the rotors, if they knew the key; and that partial cribs were available because through statistical analysis they knew that RAF messages began stereotypically with the letters AIRX and ended with a series of Xs as filler to make up to a round multiple of five characters for a group. They had Hollerith machinery in abundance and were adept at using it for sorting and statistics – as indeed had been done at Bletchley Park. They had captured keys for May and June 1940 and a memorandum from the War Office (MI1(b)) which remonstrated against lax cipher discipline. They were using 'Herivelismus' to predict rotor start-positions. All in all, they had had a good start on Typex.<sup>16</sup>

But later on, the Typex experts – notably Dr Erich Hüttenhain, Regierungsrat in the OKW's cipher research division – had abandoned work. Recovering the rotor wirings through statistical analysis would demand excessive amounts of Hollerith time, relative to the resources available and other priority work which was using the machinery to good effect. What is more, it is not clear that they had any answer to the more challenging problem of the plugboard – by contrast to the attack in Britain on Enigma, for

---

<sup>14</sup> TNA HW 62/5.

---

<sup>15</sup> TNA HW 40/88.

<sup>16</sup> TICOM document D-83, TNA HW 40/87.

which Alan Turing's Bombe had specifically been designed. Yet Hüttenhain's boss, Oberstleutnant Mettig, who at the time of the Tobruk capture was in command of OKH In 7/VI, said under interrogation that Typex was read in North Africa in 1942.<sup>17</sup> But Mettig's evidence seemed to be contradictory. He had said, only five days before, that the official in charge of the British section of In 7/VI, Referat Zillmann, 'despite great efforts was unable to break the English cypher machine'.<sup>18</sup> Mettig later retracted the statement about North Africa, but doubt lingered.

None of the OKH codebreakers interrogated seem to have been personally present in North Africa in 1942. Dr Vögele had been there, and he was accompanied to Africa by Inspektor Harms, from Mettig's team in OKH In 7/VI. Our knowledge of this visit comes partly from a bugged discussion between Hüttenhain and another German codebreaker, Dr Fricke, which took place in the evening of 25 September 1945. Harms, aged 50, was not happy in Africa, and came home after two weeks complaining of the heat. It seems that Harms didn't think too highly of Vögele: Vögele had 'done nothing' while they were there, except that he had filled two suitcases with material and taken them back to Germany while, apparently, Harms came back empty-handed. Furthermore, Hüttenhain was certain that Harms had seen nothing of Typex in North Africa – he would have said so, and he hadn't. Perhaps, hinted Hüttenhain, Vögele (of the German Air Force's Chi-Stelle) had had his own Typex operation in Potsdam?<sup>19</sup>

The basics of what Hüttenhain was telling the TICOM interrogators are confirmed by the war diary of Inspektorat 7/VI.<sup>20</sup> Harms certainly went to Africa in July 1942, when there was much excitement about the finds at Tobruk. The war diary also confirms that Zillmann had made no progress on Typex cryptanalysis in OKH. But lack of success on the part of Zillmann, as confirmed by Mettig, did not confirm lack of success in every place and by every agency. What is more, there were liaison meetings at various times between OKH In 7/VI and the

Luftwaffe's Chi-Stelle, with Vögele in particular, while Mettig was commanding In 7/VI;<sup>21</sup> at the time of Tobruk and after he was in a position to control the liaisons with other services. These liaisons included a link-up connecting the intercept team of the Luftwaffe in Athens and the NFAK 621 outpost in North Africa.<sup>22</sup> The missing part of the Typex puzzle was in the Chi-Stelle, and held by Dr Vögele in particular.

Luckily the TICOM group had captured Dr Ferdinand Vögele in August 1945. Vögele was a reluctant captive.<sup>23</sup> Vögele wrote an extensive CV, and catalogued numerous breakthroughs on various American code and cipher systems; as it was a British system, Vögele did not mention Typex.<sup>24</sup> Vögele's colleague Lieutenant Pick wrote about British systems and repeated the mantra that an attempt had been made against Typex in 1940 but abandoned. Although Vögele was interrogated specifically about Typex, the interrogation took place on 25 September, the same day that Hüttenhain and Fricke were discussing the Typex question under the British microphones. So the interrogation of Vögele about Typex lacked the input from Hüttenhain and Fricke, and was superficial:

VOEGELE stated that he would certainly have heard had Typex been broken, and reiterated most emphatically his belief that Typex was never broken. His considered opinion was that the breaking of Typex was impossible... He ceased taking the messages in 1940. When informed that a P.O.W. taken in Cyrenaica had described what appeared to be the registration of Typex traffic at Athens in 1942 or 43, VOEGELE said that one of his staff there, a cryptographer named ROSSKATH, had unofficially arranged that they take Typex traffic again for 4-6 weeks.<sup>25</sup>

Despite this feeble examination, the evidence was beginning to fall into place. Even that old telegram from the early days of the scare, when In 7/VI had been asked by Athens about Typex, could be seen to be part of the picture, if someone went through the war diary and joined the pieces up.

<sup>17</sup> TICOM document I-48, TNA HW 40/166.

<sup>18</sup> TICOM document I-78, TNA HW 40/167; TNA HW 40/89.

<sup>19</sup> Transcript of bugged conversation between Hüttenhain and Fricke, TNA HW 40/89.

<sup>20</sup> See fn 7; also PA-S8 T-2762.

<sup>21</sup> PA-S8 T-1620.

<sup>22</sup> PA-S8 T-2762.

<sup>23</sup> TICOM document I-87, NARA RG 457 HMS P4 Box 35.

<sup>24</sup> Seabourne Report, Vol XIII, NARA RG 457 HMS A1-9032 Boxes 974-6.

<sup>25</sup> TICOM documents I-87, I-119, NARA RG 457 HMS P4 Box 35.

Yet, oddly, the conclusion on 22 September had been that it was Hüttenhain who was misleading the interrogators on the story of Typex. 'Great emphasis is laid on the idea that they considered enigma, and therefore Typex insoluble.' By the next month, it was thought that Vögele was 'the problem' – he had been caught out on the business of Typex interception in Greece, and 'again we have no definitive evidence but the whole story does not ring true.'<sup>26</sup> Vögele was a civilian and the time was up; the TICOM team could not hold him indefinitely, and by the time of that report it is likely that Vögele had gone back to Germany. The secret of Typex in North Africa would remain just that.

The conclusion? It seems that the Eighth Army had left behind a Typex machine, complete with rotors and keys, at the time of the capture of Tobruk. Both OKH and the Chi-Stelle had sent someone over, but it was Vögele who had filled two cases with the materials and it was his agency, not OKH's Harms, which had exploited it. For some period after that, decryption of Typex messages was possible by figuring out the message settings being used by the British. Mettig was in the loop, but his subordinates and OKW colleagues seem not to have been: the to-and-fro between NFAK 621 and Berlin, noted by Prisoner Haunhorst, may not have involved OKH, especially if it concerned RAF issues. The main cryptanalytic player was Vögele of the Luftwaffe's Chi-Stelle. The TICOM interrogators do not seem to have picked up on the differences, or the rivalries, between the German agencies. The limited duration of the German success against Typex was probably due to the gradual adoption of the plugboard model of Typex, against which even Vögele's Chi-Stelle had no answer.

So Typex messages probably had been read, although on the basis of battlefield captures rather than as a result of Bletchley-style general cryptanalysis. Even so, that poses the question whether the British should have reacted differently when confronted with evidence that their most secure communications device was either compromised or under cryptanalytic attack.

## 6 Comparing German and British responses to security scares

The German Marine-Nachrichtendienst (naval intelligence service) is widely perceived to have scored not just one but three own-goals in failing to detect the Allied breaks into naval Enigma during the Battle of the Atlantic. Investigations took place in 1941, 1943 and 1944 into the security of Enigma. On each occasion, too-good-to-be-true coincidences were drawn to the attention of those in charge of signals security, and on each occasion alternative explanations, fantastic if not wholly absurd, were preferred to the simple, obvious, and correct interpretation that Enigma messages were being read by the Allies (Tighe, 1945; Ratcliff, 2006; Mulligan, 1985). German intelligence preferred its own narrative that Enigma was secure, so it had (consciously or unconsciously) to be shown to be so, and all evidence was interpreted in that sense. The consequences for German, and Allied, losses in the North Atlantic are well known.

To quote Dr Rebecca Ratcliff (1999):

In concluding that Enigma was not the source of enemy information, the investigators set out to prove only what could not have been the leak, Enigma. They did not set out to prove what *was* the source and did not produce a scenario which explained the [British] Admiralty's information... German intelligence assumed the enemy would either be able to read the ciphers completely – and within a three to five day period or not at all.

In parallel, though on a smaller scale, the British were shown evidence in 1942 and 1943 which indicated that (as predicted by Travis in 1940) Typex was not secure. Gordon Welchman's memo seems to have many of the same errors as the German investigations, as explicated in Dr Ratcliff's analysis. Welchman should not carry the blame for British failings, though: in 1942 the number of personnel in charge of own-systems security at Bletchley Park was one – the diligent Lt Cdr Dudley-Smith – and he a non-specialist to boot (Erskine, 2002). The Admiralty appointed Lt Cdr George Bull RNVR as adviser on cipher matters to the Naval Intelligence Department's Security Panel in the spring of 1942;<sup>27</sup> by 1943 Alan Turing had also taken on a communications security role (Turing, 2015, p 164), so things were slowly beginning to

---

<sup>26</sup> TNA HW 40/89.

---

<sup>27</sup> TNA ADM 223/505.

improve on the British side at the time of the Typex scare, but a security mindset was not yet embedded. (By contrast, when the British spotted defects – through decrypts of German messages – in American cipher security in North Africa, they were quick to point them out, and the Americans as quick to implement change.<sup>28</sup>)

Perhaps because of the paucity of personnel available for the task, the British reaction to the Typex scare of 1943 was almost as weak as that of the Marine-Nachrichtendienst. Confirmation bias seems to have influenced the decision to pursue further investigations – trying to prise more details out of more German prisoners, and checking for destruction certificates – rather than looking at the actual security issue, which was already known and ought to have been better understood. Surely it was wrong to own up to a problem with Typex only after it had been firmly established that the Germans had actually exploited Typex – to wait for sight of the horse bolting before checking the fastening on the stable door?

It seems that if they had been equipped with what Dr Vögele came away with from Africa in 1942, the talented team of German codebreakers in OKH or OKW might have been able to make some headway with the Typex problem. They might have been defeated by an earlier adoption of plugboard-equipped Typex and with more rigorously enforced operating discipline, but that does not seem to have been the outcome of the British examination of the post-Tobruk evidence. In fact, what saved the British from a potentially devastating reading of Typex messages in the months following Tobruk was the organisational split between the armed services' own cryptanalytical organisations, OKW, OKH and the Luftwaffe's Chi-Stelle. Vögele did not, apparently, have direct access to Hüttenhain, whose know-how on Typex was not shared.

## 7 Concluding comments

The British response to the Typex scare of 1943 has similarities to the German responses to the Enigma alarms throughout the war. Neither side wanted to know that its high-security communications machine was insecure. Neither side wanted to investigate properly or to implement changes which would fix a risk without overwhelming evidence which would

arrive too late, in other words after breach – potentially a breach with monumental consequences – had actually occurred. A risk-based assessment was done by neither side.

There the similarities stop. Fortunately for the Allies, the British machine was more secure than the German one, and with the use of secure indicator procedures and the plugboard, significantly so; Bletchley Park was ahead of the game on mechanical cryptanalysis, thanks to the work of the Poles before the war and the invention by Turing and Welchman of the new cryptanalytic bombe; and the Germans had not managed to invent cryptanalytic techniques which could crack a plugboard-adapted rotor cipher machine. Much of the difference was a mismatch in brilliant inventiveness. Some of that brilliance might just as easily have been on the other side; it was the Allies' good fortune that they possessed it in greater measure.

## References

- Erskine, Ralph. 1997. *The Development of Typex*. Enigma Bulletin No.2 p 69-86.
- Erskine, Ralph. 2002. *The Admiralty and Cipher Machines during the Second World War: Not so Stupid After All*. Journal of Intelligence History Vol 2 (2) p 49-68.
- Ferris, John. 2005. *The British "Enigma": Britain, signals security and cipher machines, 1906-1953*. Chapter 3 in 'Intelligence and Strategy – Selected Essays'. Routledge.
- Hinsley, F.H., et al. 1981. *British Intelligence in the Second World War*, vol 2, appendix 1. Cambridge University Press.
- Jackson, John. 2013. *Hitler's Codebreakers – German Signals Intelligence in World War 2*. BookTower Publishing.
- Mulligan, Timothy P. 1985. *The German Navy Evaluates Its Cryptographic Security, October 1941*. Military Affairs Vol 49 (2) p 75-79.
- Ratcliff, R.A. 1999. *Searching for Security: The German Investigations into Enigma's Security* in Alvarez, D. (ed), 'Allied and Axis Signals Intelligence in World War II'. Frank Cass, p 146-167.
- Ratcliff, R.A. 2006. *Delusions of Intelligence*. Cambridge University Press.
- Rezabek, Randy. 2017. *TICOM: the Hunt for Hitler's Codebreakers*. Privately published.
- TICOM (Target Intelligence Committee). 1946. *European Axis Signal Intelligence in World War II*

---

<sup>28</sup> TNA HW 40/92.

as Revealed by 'TICOM' Investigations and by Other Prisoner of War Interrogations and Captured Material, Principally German. <https://www.nsa.gov/news-features/declassified-documents/european-axis-sigint/>

Tighe, W.G.S. 1945. *Review of the Security of Naval Codes and Cyphers – September 1939 to May 1945*. Unpublished, TNA ADM 1/27186.

Turing, Dermot. 2015. *PROF: Alan Turing decoded*. The History Press.





# A Practical Meet-in-the-Middle Attack on SIGABA

George Lasry

The CrypTool Team

george.lasry@cryptool.org

## Abstract

The SIGABA is an electromechanical encryption device used by the US during WWII and in the 1950s. Also known as ECM Mark II, Converter M-134, as well as CSP-888/889, the SIGABA was considered highly secure, and was employed for strategic communications, such as between Churchill and Roosevelt. The SIGABA encrypts and decrypts with a set of five rotors, and implements irregular stepping, with two additional sets of rotors generating a pseudo-random stepping sequence. Its full keyspace, as used during WWII, was in the order of  $2^{95.6}$ . It is believed that the German code-breaking services were not able to make any inroads into the cryptanalysis of SIGABA (Mucklow, 2015; Budiansky, 2000; Kelley, 2001).

The most efficient attack on SIGABA published so far is a known-plaintext attack that requires at least  $2^{86.7}$  steps.<sup>1</sup> Although it is more efficient than an exhaustive search, it is not practical, even with modern computing (Stamp and Chan, 2007; Stamp and Low, 2007).

In this paper, the author presents a novel meet-in-the-middle (MITM) known-plaintext attack. This attack requires  $2^{60.2}$  steps and less than 100 GB RAM, and it is feasible with modern technology. It takes advantage of a weakness in the design of SIGABA. With this attack, the author solved a MysteryTwister C3 (MCT3) Level III challenge (Stamp, 2010). The author also presents a series of new challenges, which will also appear in MTC3.

<sup>1</sup>To date, no ciphertext-only attack has been proposed, except for an attack that requires multiple messages in depth (Savard and Pekelney, 1999).

This paper is structured as follows: In Section 1, the SIGABA encryption machine is described, including a functional description and an analysis of its keyspace. In Section 2, prior attacks on SIGABA are surveyed, and a novel MITM known-plaintext attack is presented, including an analysis of its work-factor, and how it was used to solve MysteryTwister C3 (MCT3) challenges (Stamp, 2010). In Section 3 and in the Appendix, new challenges are presented, as well as the reference code for a SIGABA simulator used to create those challenges.<sup>2</sup>

## 1 The SIGABA Encryption Machine

In this section, a short functional description of the SIGABA is given, as well as an analysis of its keyspace size.

### 1.1 Functional Description of SIGABA

The functional description of SIGABA presented here focuses on the features essential to the understanding of the new attack presented in Section 2. A complete description of the machine and its history may be found in the references (Savard and Pekelney, 1999; Sullivan, 2002b; Stamp and Chan, 2007; Mucklow, 2015; Kelley, 2001; Pekelney, 1998; Sullivan, 2002a).

The SIGABA encryption and decryption mechanism consists of three banks of five rotors each, the *cipher* bank, the *control* bank, and the *index* bank, as depicted in Figure 1. Each rotor of the *cipher* bank has 26 inputs and 26 outputs (similar in concept to the Enigma rotors, but with different wirings). The *cipher* rotors implement encryption (from left to right), and decryption (from right to left). The rotors of the *cipher* bank step according to an irregular pseudo-random pattern generated by the *index* and the *control* rotor bank.

<sup>2</sup>This work has been supported by the Swedish Research Council, grant 2018-06074, DECRYPT - Decryption of historical manuscripts.

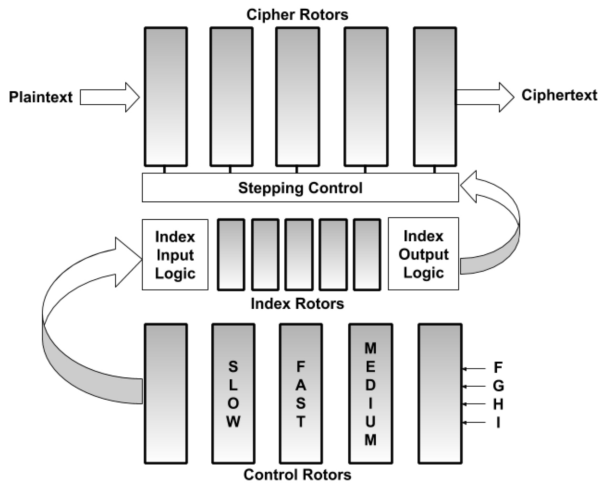


Figure 1: SIGABA – Functional Diagram

The *control* bank consists of five rotors, each with 26 inputs and 26 outputs. The *cipher* rotors and the *control* rotors are interchangeable, and are selected from a set of ten rotors. Furthermore, those rotors can be installed in two possible orientations – forward or reverse (thus increasing the size of the keyspace by a factor of  $2^{10} = 1,024$ ). Interestingly, the *cipher* rotors and the *control* rotors move through the alphabet in reverse order (e.g., from D to C, or from C to B) when installed in forward orientation, and in alphabetical order (e.g., from D to E, or from E to F) when installed in reversed orientation. The leftmost and rightmost *control* rotors are stationary and do not rotate. The *fast* rotor always steps (interestingly, this rotor is located between the *slow* and the *medium* rotors). If the *fast* rotor steps from O to N (while in forward orientation) or from O to P (while in reversed orientation), the *medium* rotor also steps (Pekelney, 1998; Sullivan, 2002a).<sup>3</sup> Similarly, if the *medium* rotor steps from O to N (while in forward orientation) or from O to P (while in reversed orientation), the *slow* rotor also steps. At each encryption step, the inputs F, G, H, and I of the rightmost (stationary) rotor are activated and fed with electrical current (and the 22 remaining input are always inactive). The 26 outputs of the leftmost *control* rotor enter the *index input logic*, described in Figure 2.

The *index* bank consists of a set of five stationary rotors, which do not rotate during encryption or decryption, and they each have 10 inputs and

<sup>3</sup>(Stamp and Chan, 2007; Savard and Pekelney, 1999) describe different implementations for the stepping mechanism.

Input (to index rotor bank)	Logic (A-Z are the outputs of the control rotor bank, V indicates logical OR)
Input 1	B
Input 2	C
Input 3	D V E
Input 4	F V G V H
Input 5	I V J V K
Input 6	L V M V N V O
Input 7	P V Q V R V S V T
Input 8	U V V V W V X V Y V Z
Input 9	A
Input 10	(inactive)

Figure 2: Index Input Logic

Stepping Control (of cipher rotor)	Logic ( $I_1$ - $I_{10}$ are the outputs of the index rotor bank, V indicates logical OR)
Rotor 1	$I_1 V I_{10}$
Rotor 2	$I_8 V I_9$
Rotor 3	$I_6 V I_7$
Rotor 4	$I_4 V I_5$
Rotor 5	$I_2 V I_3$

Figure 3: Index Output Logic

10 outputs. Those rotors are not interchangeable with the *cipher* and *control* rotors, and they can only be installed in a forward orientation. The *index input logic*, described in Figure 2, maps its 26 inputs into 10 outputs, which enter the leftmost *index* rotor. The *index output logic*, described in Figure 3, maps the 10 outputs of the *index* rightmost rotor into five *stepping control* signals, controlling the stepping of the five *cipher* rotors. The design of the *control* and *index* rotor banks, in conjunction with the *index input logic* and the *index output logic*, ensures that at least one of the five *cipher* rotors will step, but no more than four *cipher* rotors ever step (Stamp and Chan, 2007, p. 203).

Encryption is performed as follows, assuming that the 15 rotors have been installed. The machine must be set to the encryption mode. The operator selects the starting positions of the rotors,

and types the plaintext on the SIGABA keyboard. The plaintext symbol is applied to the *cipher* rotors from left to right, producing the ciphertext symbol on a printing device. After encryption of a symbol, the *cipher* rotors step according to the state of the *stepping control* (see Figure 1). After the *cipher* rotors have stepped, some of the *control* rotors step, thus generating (via the *index* rotors) a new state for the *stepping control* of the *cipher* rotors. The process is repeated for the next plaintext symbols.

Decryption works similarly, except that the device must be set to the decryption mode, and the cipher symbols (typed on the keyboard) are applied to the *cipher* rotors from right to left, the resulting plaintext being printed.

## 1.2 Analysis of the Keyspace

Assuming that there is a set of ten rotors from which the *cipher* and *control* rotors are selected, there are  $10!$  possible selections for those rotors. Each one of those rotors may be installed in either a forward or reverse orientation. The size of the keyspace for the settings of the ten rotors of the *cipher* and *control* banks is therefore  $10! \cdot 2^{10} \cdot 26^{10} = 2^{78.8}$ .

There are  $5!$  possible ordering of the *index* rotors. The size of the keyspace for the settings of the rotors of the *index* bank is therefore  $5! \cdot 10^5 = 2^{23.5}$ . The combined size of the SIGABA keyspace is  $2^{78.8+23.5} = 2^{102.3}$ .

However, the size of the keyspace for the *index* bank is limited by the fact that the five rotors implement a (stationary) permutation of the ten inputs and the ten outputs are mapped by the *index output logic* into only five outputs. Therefore, the size of the practical keyspace for the *index* rotors including the *index output logic* is only  $10!/2^5 = 113,400 = 2^{16.8}$ , and the combined size of the practical keyspace of SIGABA is as follows:

$$2^{78.8+16.8} = 2^{95.6} \quad (1)$$

For comparison, the size of the keyspace the German Enigma I was in the order of  $2^{77}$  (Stamp and Low, 2007, p. 31), and it is  $2^{56}$  for the more recent DES.

## 2 Cryptanalysis of SIGABA

In this section, prior attacks on SIGABA are reviewed, and a novel MITM attack is presented.

### 2.1 Prior Attacks

(Savard and Pekelney, 1999) describe a ciphertext-only attack on SIGABA which requires a series of 10 to 15 messages in depth, that is, encrypted with the same key.<sup>4</sup> First, the plaintexts are recovered using Kirchoffs superimposition.<sup>5</sup> The alphabets which represent the effect of the five *cipher* rotors are reconstructed, for each position of the ciphertexts/plaintexts. Positions at which only the leftmost or the rightmost *cipher* rotor move are identified, and the wiring of those rotors are recovered, by comparing the alphabet at such a position with the alphabet at the following position. The authors describe how the wiring of the inner *cipher* rotors can also be recovered, and they suggest additional methods to recover the wiring of the *control* rotors.

In (Lee, 2003), attacks on simplified and weakened versions of the SIGABA are presented.

(Stamp and Chan, 2007; Stamp and Low, 2007) describe a known-plaintext attack in two phases. It assumes that the wiring of the rotors is known, while their order, orientation, and starting positions, are unknown. At the first phase, only the *cipher* rotors are considered, and all their possible settings (rotor selection, orientation, starting positions) are evaluated. For each such setting, the first ciphertext symbol is applied across the five *cipher* rotors, and the decrypted symbol is compared to the known-plaintext symbol. To check the second symbol, all options for the stepping of the *cipher* rotors are tested (there are 30 such options, as at least one rotor steps, and at most four), and the *cipher* rotors step accordingly. If after stepping, decrypting the second ciphertext symbol produced the expected known-plaintext symbol, the process is repeated for the next symbols. Only those *cipher* rotor settings that survive the test for all the known-plaintext symbols are retained. With 100 letters of known-plaintext, about  $2^{34.5}$  *cipher* rotor settings are expected to survive, out of  $2^{43.4}$  possible *cipher* rotor settings.

At the second phase, all the surviving *cipher* rotor settings are exhaustively tested against all possible *control* and *index* settings. The total workfactor of the attack is  $2^{86.7}$ , and it is therefore more

<sup>4</sup>An unlikely scenario, given the US Army and Navy's strict operational security procedures. The method, however, requires little processing time, unlike the other attacks presented in here.

<sup>5</sup>The authors do not provide any detail on how the plaintexts can be recovered using superimposition. Furthermore, the required length for the messages in depth is not specified.

efficient than a simple brute-force attack by a factor of  $2^{95.6-86.7} = 2^{8.9}$ . The authors also propose a method with a workfactor of  $2^{84.5}$ , but with only a 0.82 probability of success.

## 2.2 A New Meet-in-the-Middle Known-Plaintext Attack

This new attack assumes that the wiring of the rotors is known, but their order, orientation, and starting positions are unknown. It was inspired by the attack described in (Stamp and Chan, 2007), and also consists of two phases. While (Stamp and Chan, 2007) is essentially an (optimized) exhaustive search, the new attack is a divide-and-conquer MITM attack. It is significantly more efficient than a simple brute-force attack.<sup>6</sup> It only requires a minimum of 8 known-plaintext symbols.<sup>7</sup> The first phase (described in Section 2.3) generates a set of *cipher* rotor settings and stepping sequences so that the expected known plaintext is accurately reproduced when decrypting the ciphertext. The second phase (described in Section 2.4) generates feasible *cipher* stepping sequences, by testing all possible *control* and *index* settings, and matching the resulting *cipher* rotor stepping sequences against those gathered during the first phase. The whole process – phase 1 and phase 2 – is repeated for all possible partitions of the ten *cipher* and *control* rotors, into two sets of five rotors.

<sup>6</sup>MITM attacks are applicable to modern multi-stage encryption systems (Diffie and Hellman, 1977). The approach is illustrated here with a system with two sequential encryption stages,  $E_1$  and  $E_2$ . For simplicity, we assume each stage has a separate key,  $K_1$  and  $K_2$ , with  $N_1$  and  $N_2$  bits, respectively. A known-plaintext MITM attack for such a system could potentially be developed. The attack has two phases. In the first phase, the plaintext is encrypted with  $E_1$  only, for each value of  $K_1$ . The resulting encryptions are stored in a hash table mapping each such partial encryption to the relevant  $K_1$ . The second phase checks for all possible values of  $K_2$ , decrypts the ciphertext using only  $E_2$ , and checks whether this partial decryption (via  $E_2$ ) matches one of the partial encryptions (via  $E_1$ ) stored in the hash table. The overall complexity of this attack is the maximum of  $2^{N_1}$  and  $2^{N_2}$ , compared to  $2^{N_1+N_2}$  for a brute-force search. This comes at the expense of additional memory for the hash table, which the two phases use to “meet in the middle”. Such an attack is also effective against 2-DES (seriated DES with two stages, each using a 56-bit key), and to achieve a level of security higher than with DES (or 1-DES), three stages (or 3-DES) are required.

<sup>7</sup>The attack also works, albeit less efficiently, with less than 8 known plaintext symbols. It may also utilize more than 8 symbols, but the author found that this number allows for a good trade-off between the size of the required memory and the overall processing time for the attack.

## 2.3 Phase 1 of Meet-in-the-Middle Attack

At the first phase, only the *cipher* rotors are considered. For each partition of the ten *cipher* and *control* rotors (divided into two sets of five rotors), the first phase produces a hash table mapping all *cipher* rotor stepping sequences, to their corresponding *cipher* rotor settings, that together produce a decryption matching the 8 known-plaintext letters. The structure and contents of the hash table is described later in this section. This is different from the first phase of (Stamp and Chan, 2007), which instead generates a simple list of matching *cipher* rotor settings, but other than that, the first phase of the new attack and of (Stamp and Chan, 2007) are similar.

All orders of the five rotors (allocated for the *cipher* bank in the partition), their orientations, and starting positions are tested. The first ciphertext symbol is applied through the five *cipher* rotors, and the output is compared to the expected known-plaintext symbol. If they match, all possible stepping options are tested, and the second symbol is processed and checked. The next symbols are (recursively) checked, and if there is a match for all the known-plaintext symbols, the corresponding *cipher* rotor stepping sequence and the *cipher* rotor settings are added to the hash table.

Stepping Sequence (Hash Key)	(Maps to) Cipher Rotor Settings					Starting Positions
	1	2	3	4	5	
01011 01000 11001 00111 00111 11110 00010	⇒ 8R	0	4R	7	1	H Y J N H
	⇒ 1	7R	0	8R	4	T U A L M
01111 01100 11100 11001 00010 10101 10001	⇒ 1R	4	8R	7	0	K H J N M
11010 10011 00111 10101 00111 00110 10011	⇒ 0	8R	7R	4	1R	E Q A M B

Figure 4: Meet-in-the-Middle Attack – Hash Table

The structure of the hash table is illustrated in Figure 4. This example is for a partition in which rotors 0, 1, 4, 7, and 8 are allocated to the *cipher* rotors. The hash key represents the stepping sequence of the *cipher* rotors, applied during the decryption of the 8 symbols for which there is known plaintext. Since stepping occurs after decryption, only the first 7 stepping patterns are relevant, as the eighth stepping pattern only affects the ninth symbol. Each stepping pattern consists of 5 Boolean values, one for each *cipher* rotor. The hash table key therefore consists of 7 groups of 5 bits.

The first group represents the stepping of the *ci*-

*pher* rotors after decrypting the first symbol. In the first entry in the hash table illustrated in Figure 4, 01011 indicates that *cipher* rotors in slots 2 (from the left), 4, and 5 (the rightmost rotor) step after decrypting the first symbol. Similarly, the next group, 01000, indicates that only the *cipher* rotor in slot 2 (from the left) steps after encrypting the second symbol.

A hash key (the stepping sequence) maps into one or more *cipher* rotor settings.<sup>8</sup> Each such setting includes the selection and order of the rotors in the 5 slots of the *cipher* rotor bank, their orientation, and their starting positions. In the first entry in Figure 4, the order of the *cipher* rotors is 8, 0, 4, 7, and 1 (installed in *cipher* bank slots 1 to 5, from left to right). Rotors 8 and 4 (first and third from the left) are in the reversed orientation (marked as 8R and 4R, respectively). The starting positions of the *cipher* rotors are H, Y, J, N, and H, respectively. The same stepping sequence (the first in Figure 4) also maps to a second *cipher* setting, with 1, 7, 0, 8, and 4 as the order of the rotors (7 and 8 are in the reversed orientation), and T, U, A, L, and M as their starting positions. This illustrates the fact that the hash table implements a one-to-many mapping, as there might be several distinct *cipher* rotor settings which reproduce the known plaintext, while the rotors step in an identical manner (as represented by the stepping sequence which is also the hash key).

Similarly, the second stepping sequence (hash key) in the hash table (Figure 4), which starts with 01111, maps to only one setting, with 1, 4, 8, 7, and 0 as the order of the rotors (1 and 8 are in reversed orientation), and set at starting positions K, H, J, N, and M.

Note that the presence of a combination of a stepping sequence and *cipher* rotor setting in the hash table, only indicates that under that same combination, the 8 symbols of the ciphertext can be decrypted to match the expected known plaintext. It does not indicate that such a stepping sequence is feasible and can be produced by the *control* and *index* rotor banks. Therefore, the need for a second phase, in order to generate all feasible *cipher* stepping sequences, and check whether they appear in the hash table created by the first phase.

<sup>8</sup>This is different from most MITM attacks, where the shared memory structure maps partial encryption or partial decryption results to partial keys. Instead, in this attack on SIGABA, the hash table maps stepping sequences of the *cipher* rotors to partial keys (the *cipher* rotor settings).

This second phase is described in Section 2.4.

## 2.4 Phase 2 of Meet-in-the-Middle Attack

The second phase looks at all possible *control* and *index* settings, generates their resulting *cipher* rotor stepping sequences, and checks whether those stepping sequences exist in the hash table. If such a stepping sequence exists, then the combination of the *control* and *index* settings, together with the *cipher* setting associated with the stepping sequence, constitutes a candidate key. When applying such a candidate key to decrypt the 8 ciphertext symbols, the decryption is guaranteed to match the expected known plaintext.

Still, this is only a candidate key, as the fact that it properly decrypts the given 8 ciphertext symbol does not mean it will necessarily properly decrypt the rest of the message. To validate a candidate key, there are two options. Either more than 8 plaintext symbols are known, and the candidate key can be validated by checking whether it properly reproduces the remaining known-plaintext symbols. Alternatively a quality measure such as the Index of Coincidence can be applied to the decryption of the full message, together with a minimal threshold. Both methods may be combined, if the known plaintext is not long enough to safely rule out wrong candidate keys.<sup>9</sup> Both validation methods also impact the workfactor. The choice of processing 8 letters of known-plaintext is a trade-off between the complexity of phase 1 (the longer the plaintext, the more steps in phase 1), the storage requirements, and the need to validate phase 2 matches (the longer the known-plaintext, the lower the probability for phase 2 false positives and the need for validation).

## 2.5 Workfactor Analysis

Phase 1 and phase 2 of the attack are applied repeatedly, on each partition of 10 rotors (those with 26 input and outputs) into a set of five *cipher* rotors and another set of five *control* rotors. The number of such partitions is equal to the number of ways to select 5 unordered rotors from a set of 10 rotors, that is,  $10!/(5! \cdot 5!) = 252 = 2^{7.9}$ .

### Workfactor Analysis for Phase 1

For each partition, the number of possible *cipher* rotor settings is  $5! \cdot 2^5 \cdot 26^5 = 2^{35.4}$ . When evaluating a certain *cipher* setting, all options for the

<sup>9</sup>Based on experiments, 12 to 22 letters of known plaintext are necessary to rule out all 'false-positive' candidate keys.

stepping of the rotors are checked for each known-plaintext-ciphertext pair (the stepping after the 8th symbol is ignored). Since between one to four *cipher* rotors step after decryption, there are only  $2^5 - 2 = 30$  possible stepping options, out of the theoretically possible  $2^5 = 32$  stepping patterns of the 5 rotors (patterns 00000 – none of the rotors step, and 11111 – all rotors step, are not feasible). For each symbol tested, the probability or ruling out such a stepping option is  $(26 - 1)/26$ . So on average,  $30 \cdot (1/26) = 1.154$  option for stepping of the *cipher* rotors survive after each decryption step, out of 30. The total number of operations for each *cipher* rotor setting is therefore  $\sum_{i=0}^7 1.154^i = 13.9 = 2^{3.8}$ . The total workfactor for phase 1 for a single partition is  $2^{35.4+3.8} = 2^{39.2}$ , and  $2^{39.2+7.9} = 2^{47.1}$  for all partitions.

### Workfactor Analysis for Phase 2

For a given partition, the number of possible *control* rotor settings is  $5! \cdot 2^5 \cdot 26^5 = 2^{35.4}$ , and the number of feasible options for the *index* rotors is  $10!/2^{32} = 113,400 = 2^{16.8}$ . Therefore, the workfactor for phase 2 is  $2^{35.4+16.8} = 2^{52.2}$  for a single partition, and  $2^{52.2+7.9} = 2^{60.2}$  for all partitions.

### Overall Workfactor

The workfactor for phase 2 is the dominant one, and therefore, the overall workfactor for the attack is  $2^{60.2}$ . This attack is more efficient than a brute-force attack by a factor of  $2^{95.6-60.2} = 2^{35.4}$ , and it is feasible with modern technology. For comparison, a brute-force attack on a 56-bit DES key was successfully carried out already in 1998 (Gilmore, 1998).

### Storage Requirements

Since the attack is a MITM attack, we still need to address the size of the hash table, which is generated in phase 1, separately for each partition of the *cipher* and *control* rotors. Based on simulations, the ratio between the number of stepping sequences generated by phase 1 and the number of possible *cipher* rotor settings is on average 0.107. There are therefore approximately  $5! \cdot 2^5 \cdot 26^5 \cdot 0.107 = 2^{32.1}$  sequences generated for each partition of the ten rotors.

To represent a stepping sequence in the hash table (see Table 4),  $7 \cdot 5 = 35$  bits are required. To represent the *cipher* rotor setting, 7 bits are required for the order ( $5! = 120 \leq 2^7$  options), 5 bits

for their orientation, and  $5 \cdot 5$  bits for their starting positions, with a total of  $7 + 5 + 5 \cdot 5 = 37$  bits. Since the vast majority of the entries in the hash table map to only one *cipher* rotor setting, the total size for an entry is about  $35 + 37 = 72$  bits.<sup>10</sup> Implementing and storing this information in a practical hash table in RAM requires some additional overhead, and based on measurements using Java 10 Hashmap library, the overall amount of space required for each entry is approximately twice the amount of space for just the data of the entry, that is,  $2 \cdot 72 = 144$  bits or about 18 bytes.<sup>11</sup> Therefore, the memory size required for the per-partition hash table can be estimated to be  $2^{32.1} \cdot 18 = 4.6 \cdot 10^9 \cdot 18 = 80$  GB.

A possible optimization to reduce the size of the hash table consists of checking for additional matching known-plaintext-ciphertext symbols, if more than 8 plaintext symbols are known. For example, by checking an additional 7 symbols (total of  $8 + 7 = 15$ ), about half of the stepping sequences may be ruled out. When checking the additional symbols (following the initial 8 symbols), we do not extend the size of the stepping sequence stored as the key in the hash table, and we still keep only the first 7 stepping patterns. We only check whether there is at least one possible continuation of this stepping sequence, following the 8 initial decryption steps. However, this optimization would also incur additional processing at phase 1, which would probably not affect the overall workfactor, as the workfactor of phase 2 is overwhelmingly dominant.

The results in this section are based on simulations performed on 200 hundreds of random keys and plaintexts. Further analysis is required to determine the optimal parameters (e.g. trade-off between processing time and storage space, derived from the length of the known plaintext processed), as well as more precise workfactor and storage analyses, for a practical attack on the full keyspace.

## 2.6 Solving the MysteryTwister C3 SIGABA Level III Challenge

In (Stamp, 2010), a known-plaintext SIGABA challenge is given. Due to the particular method the challenge was created, the effective size of keyspace is reduced, making the attack described above prac-

<sup>10</sup>Based on simulations.

<sup>11</sup>A more (or less) efficient implementation of a hash table may require different amounts of overhead.

tical on a consumer PC. With a probability of  $31/32$ , it may be assumed that the starting position of four out of the five *cipher* rotors is A, and the same applies to the *control* rotors. If we assume this is indeed the case, the size of the keyspace of the *cipher* rotors and the *control* rotors are each both reduced by a factor of  $26^4 = 2^{18.8}$ . Under the same assumption, the workfactor of phase 1 for this challenge is therefore  $2^{47.1-18.8} = 2^{28.3}$ , and for phase 2, the workfactor is  $2^{60.2-18.8} = 2^{41.3}$ .

It took a few days on a 10-core Intel Core i7 6950X 3.0 GHz PC to complete the attack and solve the challenge. The assumption stated above was found to be true.<sup>12</sup>

### 3 New Challenges

A series of new SIGABA known-plaintext challenges is presented in Table 1 (see the Appendix), with various levels of difficulty.<sup>13</sup> For most challenges, the size of the keyspace is limited by setting several of the *cipher* and *control* rotors to a fixed position A. Challenge #2 is against a keyspace of a size similar to the keyspace for the challenge in (Stamp, 2010). The last challenge (#6) is against the full keyspace of SIGABA. Java source code, used to generate the challenges, is listed in the Appendix. It is compatible with the source code given in (Pekelney, 1998), and has been tested against another simulator (Sullivan, 2002a).<sup>14</sup>

### 4 Conclusion

The functional separation between the *cipher* rotor bank, and the *control* and *index* rotor banks, is a significant weakness, and it allows for a practical MITM attack. This attack is not feasible on systems like the Siemens and Halske T52d, the Russian Fialka, and the Hagelin CX-52, in which rotors have two functions – encryption/decryption,

<sup>12</sup>*Jerva* and *Integral* had previously found the solution to the challenge. *Jerva* used methods described in (Stamp and Chan, 2007). *Integral*'s methods are unknown.

<sup>13</sup>All the plaintexts were extracted from Shakespeare writings. Each plaintext consists of the concatenation of two segments, extracted from different places. The first segment, with 100 letters, is given as a crib. The letter Z is used to represent a space.

<sup>14</sup>The SIGABA simulator source code given in (Stamp, 2010) and used to generate previous MysteryTwister C3 challenges has several incompatibility issues. The first one, described in the forum discussion, has to do with several rotors being unintentionally reset to position A. Another issue is with the stepping logic for the *medium* and *slow* rotors. In addition, the mappings for rotors in reversed orientation are incorrect.

and controlling the stepping of other rotors.<sup>15</sup> Still, the attack on SIGABA proposed here would not have been feasible given WWII technology.

### References

- Stephen Budiansky. 2000. *Battle of wits: the complete story of codebreaking in World War II*. Simon and Schuster.
- Whitfield Diffie and Martin E Hellman. 1977. Exhaustive cryptanalysis of the NBS data encryption standard. *Computer*, 10(6):74–84.
- John Gilmore. 1998. *Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design*. O'Reilly.
- Stephen J. Kelley. 2001. *Big Machines: Cipher Machines of World War II*.
- Michael Lee. 2003. *Cryptanalysis of the SIGABA, Master's Thesis*. University of California, Santa Barbara.
- Timothy Jones Mucklow. 2015. *The SIGABA/ECM II Cipher Machine: "a Beautiful Idea"*. National Security Agency, Center for Cryptologic History.
- Richard S. Pekelney. 1998. ECMApp - Emulation of ECM Mark II. <https://maritime.org/tech/ecmapp.txt>, [Accessed: January, 18th, 2019].
- John J. G. Savard and Richard S. Pekelney. 1999. The ECM Mark II: Design, History, and Cryptology. *Cryptologia*, 23(3):211–228.
- Mark Stamp and Wing On Chan. 2007. SIGABA: Cryptanalysis of the Full Keyspace. *Cryptologia*, 31(3):201–222.
- Mark Stamp and Richard M. Low. 2007. *Applied Cryptanalysis: Breaking Ciphers in the Real World*. John Wiley & Sons.
- Mark Stamp. 2010. MysteryTwister C3 (MTC3), SIGABA Part 2 (Level III). <https://www.mysterytwisterc3.org/en/challenges/level-iii/sigaba-part-2>, [Accessed: December, 16th, 2018].
- Geoff Sullivan. 2002a. CSG Sigaba (ECM Mark II) Simulator for Windows. <http://cryptocellar.org/simula/sigaba/index.html>, [Accessed: January, 18th, 2019].
- Geoff Sullivan. 2002b. The ECM Mark II: some observations on the rotor stepping. *Cryptologia*, 26(2):97–100.
- <sup>15</sup>A MITM attack is also not feasible against Enigma. Although there are multiple components involved, there is no feasible 'meet-in-the-middle point' as the encryption path traverses the plugboard and the rotors back and forth via the reflector.

## 5 Appendix – Source Code and Challenges

Listing 1: SIGABA Simulator Source Code

```

package simulator;

class Sigaba {
    private Rotor cipherBank[] = new Rotor[5];
    private Rotor controlBank[] = new Rotor[5];
    private IndexRotor indexBank[] = new IndexRotor[5];
    Sigaba(String cph, String ctl, String idx,
        String cphP, String ctlP, String idxP) {
        for (int i = 0; i < 5; i++) {
            cipherBank[i] =
                new Rotor(cph.charAt(i * 2) - '0',
                    cph.charAt(i * 2 + 1) == 'R',
                    cphP.charAt(i) - 'A');
            controlBank[i] =
                new Rotor(ctl.charAt(i * 2) - '0',
                    ctl.charAt(i * 2 + 1) == 'R',
                    ctlP.charAt(i) - 'A');
            indexBank[i] =
                new IndexRotor(idx.charAt(i) - '0',
                    idxP.charAt(i) - '0');
        }
    }
    String encryptDecrypt(boolean decrypt, String in) {
        String outString = "";
        for (char c : in.toCharArray()) {
            outString +=
                (char)(cipherPath(decrypt, c - 'A') + 'A');
            cipherBankUpdate();
            controlBankUpdate();
        }
        return outString;
    }
    private void controlBankUpdate() {
        if (controlBank[2].pos == (int) 'O' - 'A') {
            // medium rotor moves
            if (controlBank[3].pos == (int) 'O' - 'A') {
                // slow rotor moves
                controlBank[1].advance();
            }
            controlBank[3].advance();
        }
        // fast rotor always moves
        controlBank[2].advance();
    }
    private static final int INDEX_IN[] =
        {9, 1, 2, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6,
         6, 6, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8};
    // rotor stepping magnet
    private static final int INDEX_OUT[] =
        {1, 5, 5, 4, 4, 4, 3, 3, 2, 2, 1};
    private void cipherBankUpdate() {
        boolean move[] = new boolean[5];
        for (int i = (int) 'F' - 'A';
            i <= (int) 'I' - 'A';
            i++) {
            int indexIn = INDEX_IN[controlPath(i)];
            move[INDEX_OUT[indexPath(indexIn)] - 1] = true;
        }
        for (int i = 0; i < 5; i++) {
            if (move[i]) cipherBank[i].advance();
        }
    }
    private int cipherPath(boolean decrypt, int c) {
        if (decrypt) {
            for (int r = 4; r >= 0; r--)
                c = cipherBank[r].rightToLeft(c);
        } else {
            for (int r = 0; r <= 4; r++)
                c = cipherBank[r].leftToRight(c);
        }
        return (c);
    }
    private int controlPath(int c) {
        for (int r = 4; r >= 0; r--)
            c = controlBank[r].rightToLeft(c);
        return (c);
    }
    private int indexPath(int c) {
        for (int r = 0; r <= 4; r++)
            c = indexBank[r].indexPath(c);
        return (c);
    }
}

```

```

static class Rotor {
    private static final String[] WIRINGS = {
        "YCHLQSUGBDXNZKRPVJTAWFOM",
        "INXPBWETGUYSAOCHVLDMQKZJFR",
        "WNRDIOZPTAXHFYQBMSEKUCGL",
        "TZGHOBKRUVXLQDMPNFWCJYEAS",
        "YWTAHRQVLCFXUNGBIPZMSDFOK",
        "QSLRBTEKOGAICFWYVMHJNXZUDP",
        "CHJDQIGNBSAKVTUOXFWLEPRMZY",
        "CDFAJXTMNBEOHSUGRYLWZKVPO",
        "XHFESZDNBRCGKQJLTVMUOYAPW",
        "EZJQXMOGYTCSFRIUPVNADLHWBK"};
    // Index for left to right.
    private static final int TO_RIGHT = 0;
    // Index for right to left.
    private static final int TO_LEFT = 1;
    private int wiring[][] = new int[2][26];
    int pos;
    private boolean reversed;
    Rotor(int wiringIndex, boolean reversed, int pos) {
        for (int i = 0; i < 26; i++) {
            wiring[TO_RIGHT][i] =
                WIRINGS[wiringIndex].charAt(i) - 'A';
            wiring[TO_LEFT][wiring[TO_RIGHT][i]] = i;
        }
        this.reversed = reversed;
        this.pos = pos;
    }
    void advance() {
        if (reversed) {
            pos = (pos + 1) % 26;
        } else {
            pos = (pos - 1 + 26) % 26;
        }
    }
    int leftToRight(int in) {
        if (!reversed) {
            return
                (wiring[TO_RIGHT][(in+pos)%26]-pos+26)%26;
        }
        return
            (pos-wiring[TO_LEFT][(pos-in+26)%26]+26)%26;
    }
    int rightToLeft(int in) {
        if (!reversed) {
            return
                (wiring[TO_LEFT][(in+pos)%26]-pos+26)%26;
        }
        return
            (pos-wiring[TO_RIGHT][(pos-in+26)%26]+26)%26;
    }
}
static class IndexRotor {
    private static final int WIRINGS[][] = {
        {7, 5, 9, 1, 4, 8, 2, 6, 3, 0},
        {3, 8, 1, 0, 5, 9, 2, 7, 6, 4},
        {4, 0, 8, 6, 1, 5, 3, 2, 9, 7},
        {3, 9, 8, 0, 5, 2, 6, 1, 7, 4},
        {6, 4, 9, 7, 1, 3, 5, 2, 8, 0}};
    private int wiring[] = new int[10];
    private int pos;
    IndexRotor(int wiringIndex, int pos) {
        System.arraycopy(WIRINGS[wiringIndex], 0,
            wiring, 0, 10);
        this.pos = pos;
    }
    int indexPath(int in) {
        return (wiring[(in + pos) % 10] - pos + 10) % 10;
    }
}
public static void main(String[] args) {
    Sigaba sigaba =
        new Sigaba("0R1N2N3N4R", "5N6N7R8N9N",
            "01234", "ABCDE", "FGHIJ", "01234");
    String out = sigaba.encryptDecrypt(false,
        "AAAAAAAAAAAAAAAAAAAA");
    System.out.printf(
        "%s (expecting JTSCALXDRWOQKRXXHKMVD) \n", out);
    sigaba =
        new Sigaba("0R1N2N3N4R", "5N6N7R8N9N",
            "01234", "ABCDE", "FGHIJ", "01234");
    String in = sigaba.encryptDecrypt(true, out);
    System.out.printf(
        "%s (expecting AAAAAAAAAAAAAAAAAA)\n", in);
}
}

```



	<b>Ciphertext</b>	<b>First 100 Plaintext Letters</b>	<b>Hint</b>
#1	GSZQEMAGFULNFZHHRVUTCUEXU FBMPDGOROJRPMAUDOZMJWJCVH YCBZDELOWKVLYJLSZBQJXWXL WOIMBVUTBAVRHPPPYQDTIURLV IQGIZSEVGXOYCMGESFOXDLPFT UQQCRDSRNFDTBDDULFJKQGXZB XKKIMSBSIUZSZNOOLCFRRVTOD XFQRRXLDEMSLORKXUCGDKCZKY ULDORUGEGDLTTROBUIVWJTBVH YWOKANYJCGQUYGPHSMWJRILZP SQJOXKKMEGMWQKXWVKF	AHZFOULZSHREWDZNEWSZBESHR EWZTHYZVERYZHEARTZIZDIDZN OTZTHINKZTOZBEZSOZSADZTON IGHTZASZTHISZHATHZMADEZME	All <i>cipher</i> and <i>control</i> rotors are at position A.
#2	ZMJHMLJTJSSHZBBMYXJRVZCUS PMETNBPZQCAHGYJDHJNQNMTHY EJAOOQYFSURONLTGQVKOMABX QXGKRAVBZYWBRWYGLBYFZNN XIVJVOJYYBQGTWJIIIZESYBRAN XEWYDRMYAINJWWDFWBVCTHRL ZCTNHWWBRYSJSZSYMSSLUXBLZ STDBARVGCSTJOWIRFXIBZCF CCYRUXMUCISNUIFLCOJYZQTBY DWVFDHJZBJNSAPYAUYWQGFY ZJYWPCWVRSVCQTPHTFPGHCJAM CFZRHYNFXYJVVWNNN	WOULDSTZTHOUZNOTZBEZGLADZ TOZHAVEZTHEZNI GGARDLYZRAS CALLYZSHEEPBITERZCOMEZBYZ SOMEZNOTABLEZSHAMEZFABIAN	The last 4 <i>cipher</i> rotors and the last 4 <i>control</i> rotors are at position A.
#3	HYQUSBFHVDVKSLSKSGUQIVZAR QKQZBLLGCTCLQHZNBEQVUOJH BROKUKRYXWPGSPDJSWLLTDASB MTTPRPFHMSXPLBDENAYJWAQZD JDXGBJCWXNARABTTSEZBJDYHT NEIQCQRTFUAZDTTVBNHJGWQHF UHAPBPYJAIXGELTILPULVSNC BJJIGFJNYDURTI VVYHTNKFSL ALTHLBHYQBYXUK	TISZWONDERFULZWHATZMAYZBE ZWROUGHTZOUTZOFZTHEIRZDIS CONTENTZNOWZTHATZTHEIRZSO ULSZAREZTOPFULZOFZOFFENCE	The last 3 <i>cipher</i> rotors and the last 3 <i>control</i> rotors are at position A.
#4	CEXZZGZOYLDYPAGJQTFJSEYZP ORHMSTYLQVSJARJLCDBYXFPKB NREAEYVOPBQKYFYETXOUQNMAT CBWIFKJWZJFWZHMJYQALVNXV UDUVEJGJNBWZRCVMIHDHLPD LSBPTFNEGWAIRZZPIPPVEBWB VBGLNCGBKWFUUCVGTGKGEHJQ XGEHVP LDD LALNWNDOXTPPWCQ HNAWFTXVOWIZFVRWXBIIJDFAU TMCNWDHLSCHNOBQRURVLCXLVB YDXKMPYIWPYOXPFXBNEBU WZECXOUDTVVNRGGHPTE	IZWILLZBESPEAKZOURZDIETZW HILESZYOUZBEGUILEZTHEZTIM EZANDZFEEDZYOURZKNOWLEDGE ZWITHZVIEWINGZOFZTHEZTOWN	The last 2 <i>cipher</i> rotors and the last 2 <i>control</i> rotors are at position A.
#5	JJJWJZMPUKYDGRHSPIXTYPAPA IVGFOTXMFWRZLBRXQPNRYLCPF WNMZFHF SMVIEEDAHWZOMBIVPA RTAOWYOWRFACGAI TUAFDCTEV YZAQIQXVHZFCIBSVSQJAMYPTS YNWXBFBKDKVDOXZQQE VVGAAWI LRFYRGIPJCKVVPMAEIAIMOPY XCSJFDAUHYZYVQJXGGZTMCAGW BEICRYROYCPNGEZQFVVQTSZBP SZYWCNNWMUBCNYQX	HOWZMIGHTZWEZSEEZFALSTAFF ZBESTOWZHIMSELFZTONIGHTZI NZHISZTRUEZCOLOURSZANDZNO TZOURSELVESZBEZSEENZPOINS	The last <i>cipher</i> rotor and the last <i>control</i> rotor are at position A.
#6	FWEYNOPSTLFMWXQITVTMRVHOL YDEIROBXPVZVBLCSJPSYIXIY IJHJMCHAWSAQBH SUVASAGYLR DJREKIFQUXBEJZUFVIJBIMWVT VSPHOQTRAECHEEJLBRCDTGXRP OVSJKDYWNWIUTPXKXSHDCBC WVYDGBVJLMCPZJROXKDPDTMC PHXGCTHPDLVHYQHFRRTTKSOTE IWAXEDMUOVBLSLZUWFTY GNCQY YPHZRNJRBYVVS NPYWAEMXOIV UQWAXAECBOODIPLWGC VQJVDCX GKCBXHCUK	TOZHAVEZNOZSCREENZBETWEEN ZTHISZPARTZHEZPLAYDZANDZH IMZHEZPLAYDZITZFORZHEZNEE DSZWILLZBEZABSOLUTEZMILAN	No hint given.

Table 1: New SIGABA Challenges



# Dead Ends in Breaking an Unknown Cipher: Experiences in the Historiography of the Rohonc Codex

**Benedek Láng**

Budapest University of Technology and  
Economics  
1111, Budapest, Egry József u. 1  
benedeklang@gmail.com

## Abstract

A close reading of unsuccessful breaking attempts of unsolved historical cryptograms (particularly, if they happened to be solved at a later moment) is more useful than it would be obvious at first glance. While the specific “solutions” offered by code breakers differ from each other, the attempts in the case of the Rohonc Codex, the Voynich manuscript, and other historical ciphers (or codes) exhibit surprising structural similarities. These attempts can be classified into promising and unpromising subcategories on structural grounds even when the final solution is not available. The paper aims at supporting this argument in a case study of the amateurish “solutions” of the Rohonc Codex, which include an old-Hungarian, a proto-Rumanian (11<sup>th</sup> century “vulgar Latin”) and an old Sanskrit script theory. A more convincing thread also emerges, details of which are in the process of being published.

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

## 1 Introduction

As a result of the last decade, reliable literature on the Rohonc Codex became available in English (Láng 2010, Király and Tokai 2018). The history of this entirely enciphered 450 long source is described, the characteristics of its script are analyzed. Most recently a convincing

code breaking attempt is offered (Király and Tokai 2018), the continuation of which will be also submitted to the decisive journal of the history of cryptography, *Cryptologia*. However, much less attention is paid to the earlier, aborted attempts at breaking the code, which are accessible only in Hungarian.

## 2 The codex

The Rohonc Codex (also called Codex of Rohonc, and even: Codex Rohonczi) is a handwritten paper book filled with unknown sign-strings and more than 80 seemingly biblical illustrations. It consists of nearly 450 pages. The first and last few dozen leaves were detached from the book itself, thus rendering the original order of these pages as unknown. There is no title page. The small pages (3.9 x 4.7 inches / 10 x 12 centimeters) on average contain nine to fourteen lines of characters of some unknown origin, and eighty-seven illustrations altogether. The leather binding was attached to the pages in the 19<sup>th</sup> century, and thus it does not add anything about the early history of the book. At present, the book is kept in the library of the Hungarian Academy of Sciences (MS K 114).

What we know of its history dates back to 1838 only, when, as part of the thirty-thousand-book library of the late Hungarian magnate, Gusztáv Batthyány, it was incorporated into the collection of the Hungarian Academy of Sciences. Since no other information is available to us about its origin, the codex is named after the town of Rohonc (today Rechnitz, in Austria), the Batthyány family seat. However, since the Batthyánys had been amassing their book collection from a great array of sources through a succession of centuries, there is no proof that the codex is either Hungarian or Central European in origin.

Soon after it emerged, the mysterious codex earned considerable academic attention, but only as long as it could be considered a potentially valuable piece of old Hungarian writing. The initial enthusiasm soon died out, giving place to disappointment, skepticism and suspicion. By the end of the 19<sup>th</sup> century the academic public had decided to regard it as a forgery, and virtually no serious study was published on it until the turn of the 21<sup>st</sup> century.

### 3 The first attempts

The codex was first examined shortly after it was found by the Hungarian language historian, János Jerney. Judging by the watermark, he soon identified the paper as coming from 16<sup>th</sup> century Italy. This type of paper was indeed fairly common in early-modern Hungary. From the Biblical topic of the illustrations Jerney concluded that the author must have belonged to a Christian culture. He then went on to compare the writing to various Asian writings since it displayed some Eastern characteristics. Jerney was not yet tempted to regard the source as an ancient Hungarian document. Instead, he suspected that if the language was a natural one, it could have been written by Tartars, who had settled in medieval Hungary and become Christians. His theory was that the Tartars used their own Asian letters to write the Rohonc Codex or the original book that the codex was copied from. Jerney did not believe that the text was an imitation of an ancient Hungarian source or any other natural language. He did not think the purpose of the book had been to “deceive coming generations or to create a counterfeit just for the sake of a game”. He did toy with the idea of the text being a cipher though (Jerney 1844).

In the following years, a succession of scholars tried to identify the symbols, Hungarians and foreigners alike: Ferenc Toldy, Pál Hunfalvy, Josef Jireček from Prague, Bernath Jülg from Innsbruck, Alois Müller from Graz. Later, Mihály Munkácsy, the famous Hungarian painter, even took the codex with him to Paris to have it examined. The first systematic and published attempt at breaking the codex was by Kálmán Némäti, who started working on the book after it was brought back from its 18-month long sojourn in Paris.

Kálmán Némäti (1855-1920), the ‘educator of the nation’ – as he called himself, had a life so unique that it should be described in a separate

monograph. He certainly did not belong to the institutionalized mainstream of the historiography of Hungarian literature. After giving up on educating the nation, he spent two years in an empty bear cave where, according to his entry in a biographical encyclopedia, “he wore underclothes and a monk’s habit made by his own hands; ate wild fruit and roots; and was often visited by the people of the land who would listen to his speech and give him wheat, fruit and bread” (Szinnyei, 1903, 954). Later, living on alms from his relatives like a “beggar-writer”, he published a long line of articles not only on the Khazars, the Turks and the origins of the Hungarian people, but also on a proposed reformation of the teaching of the alphabet in primary schools, and the laws of nutrition (he himself was a hardcore vegetarian). As for the Rohonc Codex, he correctly identified the writing as running from right to left, and incorrectly argued for the ancient Hungarian origin of the text. He published his views on his own. Besides, he submitted a manuscript typology to the Academy that listed and grouped the symbols of the codex, of which he had found almost 800. This high number of symbols made him suspect that the codex is a syllable-writing (Némäti 1884 and 1889).

Némäti’s research received some scholarly attention when he requested a grant from the Academy. The Committee of Linguistics took his proposal seriously and, according to the official record of the meeting on 12 November 1898, decided that they primarily needed a “palaeographic study in order to judge whether the manuscript is an ancient Hungarian source”. So they asked four palaeographers, experts on ancient writings, who, based on a variety of evidence, came to the conclusion that though the paper “was indeed from the first quarter of the 16<sup>th</sup> century, the writing on it is a later forgery.” Their main argument was

“it is impossible to encipher a text using 900 symbols because no man on Earth could possibly read such a text, not even the person who had created it. Handling an alphabet of 900 secret symbols is beyond the capacity of human memory. The words are not separated, making the text difficult, even impossible to read. Moreover, no corrections have been made, which is unheard of in a manuscript of this length.”

Two of these observations are simply incorrect: there are ciphers that use 900 symbols or more, and the codex does contain a number of corrections, deletions and strikethroughs. Still, these comments imply that the palaeographers did not believe Némäti's ancient Hungarian script hypothesis was correct. Instead, they examined whether the string of symbols could be a cipher. Though their arguments will be discussed later, the final conclusion of the Committee must be quoted here,

“All of these convinced the committee that Mr. Kálmán Némäti had been wasting his rare tremendous zeal on an impossible task, and that anyone encouraging him to continue this work would do a bad thing to him.” (strikethrough in the original record) (The records of the Committee of Linguistics).

When declaring the Rohonc Codex to be a forgery, the 12 November 1898 meeting of the Committee of Linguistics of the Hungarian Academy of Sciences silenced a long wave of attempts that were losing their initial fervor and were becoming more doubtful. The Academy's opinion actually discouraged the desire to break the codex for almost a century (with the exception of one attempt). Upon the arrival of the third millennium, however, many voices broke the silence.

#### 4 The ancient Hungarian theory

In the last few decades, Attila Nyíri, was one of those who proposed a solution for the codex. Nyíri is neither a professional historian, nor a paleographer, instead he is an electrical engineer, but we should bear in mind that good insights in the codebreaking of historical texts often emerge from non-professional sources. In the late 1990's, when copies of the codex were not yet easily available, Nyíri used those two complete pages he had access to. He read the symbols of the codex as a prayer written with ancient Hungarian letters, i.e. a natural language. This means that he did not *correspond* the characters of the codex to the letters of the Hungarian language, but he simply *recognized* them as letters that could be read spontaneously (Nyíri, 1996). He happened to read the text upside down, something that he himself realized later, but it is not this mishap that proves him to be completely wrong. For one thing, Nyíri allowed for one letter (one sound) to correspond to several symbols, a method perfectly common in the case of ciphers, though

not so much in natural languages. For another, he read the same character as several different letters. He furthermore claimed that the order of the letters is sometimes jumbled up in the text. This method of decoding, nevertheless, drops to such an arbitrary level where every deciphering attempt is successful by nature, and thus any text can be read in any way.

Turning the page upside down, a few lines from the Rohonc Codex as Attila Nyíri reads it from right to left in “the ancient Hungarian language” sounds like this:

“Your God has arrived. Oh, the Lord is flying. There are the holy angels.

Oh, yes, them. Sung with decorous words, send the song, pour it.

The Lord is to come, I am flowing everywhere. Oh. The Lord is honored.

The peace of the Lord flies far. Those holy words...”

#### 5 The Daco-Romanian hypothesis

Nyíri was no philologist, and he only had access to a limited section of the original text. These qualifications cannot be made in defense of that Romanian archaeologist who summarized her twenty years of studies of the Rohonc Codex in an eight-hundred-page book (Enăchiuc, 2002). Viorica Enăchiuc claims that the text is in Vulgar Latin, in other words, proto-Romanian, from the 11<sup>th</sup> century. She transcribed the complete string of symbols, provided the dictionary of this hitherto unknown language, published the complete reproduction of the codex without ever asking the library for permission to do so, and then translated the text to modern Romanian and French. Furthermore, in her voluminous book she also made room for studies in Romanian (and their French translations) on topics such as the “musical notes” and “musical content” of the codex, various maps of Dacia, and even the author's list of publications.

There was only one thing left out from this thick volume, something that is the basis of all successful solutions, namely the one-or-two-page code table that would reveal which symbols correspond to which letters in Enăchiuc's view. Her hints in the book seem to imply that we will only see this table when the “second volume” is

published. If the readers, however, attempt to create the table on his own, they will quickly find that the various symbols correspond to a different letter every time, carefully tailored to the meaning that Enăchiuc's Vulgar Latin text carries.

Throughout the twenty years she spent on studying the codex, the archeologist never realized she was reading the text the wrong way. She did notice that the symbols read from right to left, which is obvious from the facts that the text is aligned to the right, and that there are sometimes hyphens at the left end of the lines. Why she read the text from bottom up, however, is hard to explain. Even a relatively quick study of the text identifies coherent strings of characters that, when broken at the end of a line, always continue in the right end of the line below. Furthermore, there is a fairly conspicuous phenomenon where the end of a chapter or section produces a blank area at the bottom of a page, and not its top. Looking at these features, even the earliest researchers could correctly determine the direction of the writing, right to left, top to bottom, of the Rohonc Codex.

Equally bizarre is the fact that the Romanian researcher failed to notice that certain symbols always stand together, such as the frequent IO:O and several dozen other examples that she decodes as separate letters every time they occur. If they really were a string of separate letters, instead of carrying a meaning as a complex character, then they would be signs of such a high level of structure unmatched by any other language. (In other words, the number of letters that always stand together like q+u in Latin, are too high.) Enăchiuc also believes the second symbol of one specific digraph to be a sentence delimiter. If that were the case, all sentences would start with the same letter, i.e. the first symbol of this digraph. This weird feature of Vulgar Latin went unnoticed during the process of translation only because the same symbol is always translated by different letters. This is yet another method that enables the reader to translate any kind of ciphertext in any way she chooses.

All of these peculiarities make sense, naturally, once we glimpse into the reconstruction of the text and discover her motivation. In her rendering, the codex describes the centralized Blaki (early Romanian) state,

located between the Tisza and the Dniester rivers, at the peak of its glory in the 11th and 12th centuries, led by emperor Vlad. The codex contains speeches, prayers and songs in connection with this state, but mostly battle songs to inspire the 11th century Blaki youth to glorious victory over the Oghuz and the Hungarian people. The Oghuz, if I understand her correctly, are in fact the Pechenegs, who were allies of the Hungarians and posed a threat on the centralized Blaki state as well as the Byzantine Empire around the year 1100.

Let us read the transcript of the codex by the Romanian archeologist, Viorica Enăchiuc. It goes right to left, bottom to top in Vulgar Latin, i.e. in Daco-Romanian language,

“Deteti lis vivit neglivilu iti iti itia niteren

Titius suonares imi urast ucen” (Enăchiuc 2002, 22)

Chances are that the reader would not quite be able to read this in Vulgar Latin. Vulgar Latin, to the best of our knowledge, has no other surviving source, let alone a language book or dictionary. This does not bother Enăchiuc in the slightest, as she attaches a detailed dictionary where she assigns each word of this language to another word, usually Latin. For example, the Vulgar Latin ITI comes from the Latin “eo, ire, ivi, itum” word, which means go. The strange-looking NEGLIVLU is not listed in the dictionary, but could be the verb neglect, based on a similar-sounding word, the Latin “negligo”. SUONARES stands for the Hungarians, because it sounds like Hunor, a character in *Gesta Hungarorum*, an early history of the Hungarians written by the medieval Hungarian chronicler, Simon of Kézai. The Vulgar Latin lines above were translated to French by Enăchiuc and here you, the reader, can read them in English,

“In great numbers in the fierce battle, go without fear, go heroically!

Advance thunderingly, to sweep away the Hungarians and win!” (Enăchiuc 2002, 674)

Elsewhere an encouraging speech is quoted, that was delivered at the forts of Inau, Transylvania, before the battle over the river Tisza against the Hungarians,

“A suoar noas suoar striol / inou iu oi iura fidi  
tenis nitioi / inou nevi tenes sedani dit = / iu

elicen vasi abdi bini / sunar edo lidi sunar titi tisa  
/ ti inou to veiki uti nititi acira / ti deti atr dira  
sati sunara / ot nis tenen vi ulcer iurai sunar /  
dica er uti veik iuku inou a roi / suoar osorai  
suoar striool / isti is etia vi iker uti iti ser”  
(Enăchiuc 2002, 8)

In English:

“In our defense, in defense of the Strei! Go to  
Ineu and swear!

Defend it with glory and defend the united  
Ineu in continuum, completely.

Go together, I have pushed forward. Together  
fight back the Hungarians; I encourage you to  
fight over the Hungarians, not letting go of the  
Tisza at your Ineu; push forward, to shine with  
glory, by your bravado

stop the cruel tragedy caused by the  
Hungarians.

To defend us strongly, swear to wound the  
Hungarian.

Decide my lord, to push forward at Ineu with a  
hawk’s cry again in defense, decide about the  
defense of Strei in advance.

Go and now you will strike with greater force,  
now that you go united.” (Enăchiuc, 2002, 669)

The reconstruction by Enăchiuc about the  
centralized 11th century Romanian state and its  
soldiers who would defeat the Hungarians over  
and over raises a series of historical problems,  
and little wonder that even Romanian historians  
have criticized it (Ungureanu, 2003). Let us for a  
moment accept this theory and assume that ever  
since Roman times there had been a Romanian  
state, and that its people spoke “Vulgar Latin”.  
We will still find it impossible to accept  
Enăchiuc’s rendering of the codex: she reads it  
the wrong way, she decodes the same strings of  
symbols into different sentences, and she makes  
up a non-existent language that has sources  
nowhere else.

Although a close study of the illustrations of  
the codex is beyond the scope of this paper, we  
must cite how the archaeologist identifies  
emperor Vlad, his subjects and the ambassadors  
arriving to the emperor from Byzantium in  
images where every other researcher sees typical  
Biblical scenes. There is an image, for example,

of Christ entering Jerusalem riding on a donkey,  
with people laying their clothes in front of him, a  
palm tree that people have taken the branches  
off, and also the money changers being driven  
out of the temple. This is entitled, “Vlad is  
preparing for the alliance to be made with the  
Byzantines against the 1064-65 conquest of the  
Oghuz, the metropolitan archbishop of the Blak,  
Sova Trasiu, blesses the warriors in the temple  
with battle signs”. The Hail Mary, depicting  
Mary, the winged angel and Joseph bears the  
title, “Sova Trasio metropolitan archbishop is in  
a wooden church with a bell tower, sending a  
book to Jaroslav I, Prince of Kiev so they would  
unite with the Blak in the war against the  
Oghuz”, as Enăchiuc sees it. On the right, where  
we see Joseph and the angel, she sees “the Prince  
of Kiev, who, receiving the news, accepts the  
alliance”. The adoration of the Magi, which, to  
make things clear, also depicts the star of  
Bethlehem, is seen by her as, “Vlad, head of the  
Blak, is standing with Sova Trasiu metropolitan  
archbishop and a general, receiving the envoy of  
Byzantine emperor Constantine, in an army base  
somewhere in the Sub-Carpathian region.” She  
does not stray so far from the traditional  
rendering in the obvious scene of the crucifixion,  
“Vlad, governor of the Blak, wearing a helmet  
and weaponry, is praying for victory at the foot  
of the crucified Savior, before leaving for the  
war against the Oghuz.” The arrest of Christ in  
the Garden of Gethsemane, however, is actually  
the meeting between the Goths, who were  
crossing the Blak territories, and the Byzantine  
army leaders, while Christ meeting Pilate is  
indeed Vlad, Prince of the Blak, before the  
Byzantine emperor Alexander I. The scene  
depicting Christ wearing a crown of thorns  
standing before Herod is explained as the Cuman  
king receiving blessing from the Cuman high  
priest. The king is about to leave for battle where  
he is going to fight on the side of the Blak and  
the Byzantines. The events unfolding around the  
grave of Christ (the noli me tangere scene, the  
empty cave with the angel) in Enăchiuc’s book is  
the scene where the Goths are preparing to fight  
back the invasion of the Pechenegs. All of these  
illustrations are typical and unambiguous  
depictions of Biblical scenes using conventional  
iconographic symbols.

## 6 The Sanskrit theory

Perception is gravely influenced by a priori  
nationalistic expectations, a fact not only

illustrated by the work of Enăchiuc. Turán, the journal specializing on research on the early history of Hungary published the solution of Dr. Mahesh Kumar Singh. Dr. Singh is from India, and at the same time a descendant of the Hunnish royal family – as he himself claims. The accounts of the chief editor of the journal inform us, how Dr. Mahesh started browsing through an earlier edition of the journal, reading in English without a pause the facsimile of the mysterious Rohonc Codex (Turan, 2004-2005, 6-7), as if the language was familiar to him. The chief editor of the Turán, despite contrary advice from some fellow editors, published the first 24 pages of the Singh transcript, along with its Hungarian version (Singh, The Rohonc Codex, 2004-2005).

The following are a few lines from the Rohonc Codex as read by Dr. Mahesh Kumar Singh, from left to right, top to bottom, in Old Indian script, in Sanskrit:

“Oh, Lord, the people here are very poor, sick and hungry, / give them skills and strength to satisfy their needs / provider, do not harden your heart / do not take your hand from this needy people / their needs that they desire for themselves / grant these to them for their sake / whenever you help them these people / that they may find this help perfect.”

Credit should be given to the journal Turán for publishing a reader’s letter in a subsequent issue with a detailed refutation that argues, like we do with regard to Nyíri and Enăchiuc, that the Singh decoded the same strings of symbols in different ways: one digraph was for example interpreted in eleven different ways, and one other, longer string of symbols had four meanings in four different places within the text (Varga, 2005).

## 7 Systematic attempts

The list of attempts at decoding the Rohonc Codex is not yet finished. The following ones are relevant because of the method applied in them, tools that a professionals would also use. And although they were considerably more successful than the attempts described so far, they have modestly refrained from thinking they were a full solution. The first such attempt is that of lieutenant colonel Ottó Gyürk, who became known in 1969 as the person who deciphered the numbers in the encrypted diary of the novel writer, Géza Gárdonyi (the script of the diary was cracked by Gábor Gilicze, then a university

student). Having successfully transcribed the secret text, Gyürk felt confident enough to have a go at breaking the Rohonc Codex too (Gyürk 1970). Studying the continuous strings of symbols that broke up at the end of the lines, as well as the incomplete lines and pages, he quickly realized that the text goes from right to left, top to bottom. He also correctly identified the double line that often appeared on the left side of the page as a hyphen. This symbol is familiar to researchers studying handwritten texts from the 16th to 19th centuries, but we must remember that Gyürk was not a philologist. Following the procedure that proved to be successful with Gárdonyi’s diary, Gyürk went on to try and identify certain strings of symbols as numbers. He confesses to spending years on creating tons of systematic statistics, yet he could not achieve any more results. His method, which focused on studying the patterns and statistics of (groups of) symbols was the first in the history of the Rohonc Codex that was promising.

A similar, but computerized analysis was carried out by Miklós Locsmándi, who published his results in the same 2004 issue of Turán that also contained the Singh-transcript. Locsmándi apparently received less support from the editor (Locsmándi, 2004-2005). He realized fairly quickly that the language of the codex is a constructed one and he was willing to let go of the early Hungarian script theory, a feature somewhat alien to the journal Turán. He distinguished between the simple and the complex symbols, a real achievement in itself, because neither Némäti, nor Nyíri, nor Enăchiuc seem to have recognized that certain symbols always stand together and must be translated as one unit. Locsmándi then examined the frequency of the symbols, to find out if they represented letters, syllables or perhaps complete words. He drew attention to a queer feature of the codex, namely that the text contained repetitions that were too frequent. “The language of the text probably uses a small number of simple basic units,” he concluded. This structure is strongly characteristic of “prayers of litany, perhaps charms from the folk tradition” which suits the proposition based on the illustrations that the codex is the prayer book of a religious, perhaps sectarian movement. Locsmándi, however, did not produce the real solution either.

## 8 Epilogue



Finally, in 2010, after a ten-year pause, another important step was made on the road paved by Gyürk and Locsmánda. Gábor Tokai and Levente Zoltán Király started cooperating on the code of the codex, which led finally to a common series of articles, the first of which have been published in *Cryptologia*. This is however, a different story, which is – in contrast to the earlier, aborted attempts – accessible for the wider public.

Csaba Varga 2005. “A Rohonczi Kódex M K Singh-féle olvasatának ellenőrzése” (Checking M K Singh's transcript of the Rohonc Codex), *Turán*, 198-202.

## References

- Viorica Enăchiuc 2002. *Rohonczi Codex: descifrare, transcriere si traducere*, Editura Alcor, Bucarest.
- Ottó Gyürk. 1970. “Megfejthető-e a Rohonci-kódex?” (Is it possible to decipher the Rohonc codex?), *Élet és Tudomány* 25, 1923-1924.
- Ottó Gyürk. 1996. “Megszólal a Rohonci-kódex?” (Will the Rohonc codex finally speak?), *Theologiai Szemle*, 39, 380-381.
- János Jerney, 1844. “Némi világosítások az ismeretlen jellemű rohonczi írott könyvre,” (Some information concerning the unknown book of Rohonc) *Tudománytár*, 8, 25–36.
- Levente Zoltán Király, Gábor Tokai, “Cracking the code of the Rohonc Codex” *Cryptologia* 2018, 285-315.
- Benedek Láng, “Why don't we decipher an outdated cipher system? The Codex of Rohonc,” *Cryptologia* 34 (2010): 115 – 144.
- Miklós Locsmánda 2004. “A Rohonci Kódex. Egy rejtélyes középkori írás megfejtési kísérlete,” (The Rohonc codex: An attempt at solving a mysterious medieval writing) *Turán*, 41-58.
- Kálmán Némäti. 1892. *Rohonczi Codex Tantétel*, (Statements about the Rohonc Codex), Budapest.
- Kálmán Némäti. 1889. *Rohonczi Codex Ábéczéje*, (The Alphabet of the Rohonc Codex) 1889, MTA Manuscript Library, Ms 884.
- Attila Nyíri. 1996. “Megszólal 150 év után a Rohonci-kódex?” (Will the Rohonc codex finally speak, after 150 years of silence?), *Theologiai Szemle*, 39, 91-98.
- Maresh Kumar Singh 2004-2005. “Rohonci Kódex” (The Rohonc Codex) (The transcript of the first 13 folio by M K Singh, as well as the Hungarian translation by László Bárdi), *Turán*, 9-40;
- József Szinnyei. 1903. *Magyar írók élete és munkái* (The lives and works of Hungarian writers) Hornyánszky Viktor, Budapest, 9. vol., 954-955.
- Dan Ungureanu. 2003. “Nu trageti in ambulanta,” *Observator Cultural*, 167.



# Uruguayan Cryptography: Printed Book Covers

**Juan José Cabezas**  
Facultad de Ingeniería  
Universidad de la República  
Montevideo, Uruguay  
jcabezas@fing.edu.uy

**Francisco Castro**  
Facultad de Ingeniería  
Universidad de la República  
Montevideo, Uruguay  
fcr@adinet.com.uy

**Joachim von zur Gathen**  
B-IT, Universität Bonn  
Germany  
gathen@bit.uni-bonn.de

**Jorge Tiscornia**  
Presidencia Uruguay  
Montevideo, Uruguay  
jtiscornia@  
presidencia.gub.uy

**Alfredo Viola**  
Facultad de Ingeniería  
Universidad de la República  
Montevideo, Uruguay  
viola@fing.edu.uy

## Abstract

During the military dictatorship in Uruguay in the 1970s and 1980s, three encrypted documents were sent between members of the urban guerrilla *MLN Tupamaros*. In this work we present the second of these documents, whose text was encrypted by an array of colored circles on the covers of a book.

We introduce some of the history and the processes of image recognition and decryption of this document.

## 1 Introduction

In Uruguay, a small buffer country between Argentina and Brazil, an urban guerrilla *MLN Tupamaros* emerged in the 1960s and 1970s under the influence of the Cuban Revolution. After years of fame, it was finally defeated in the mid-seventies and a large part of its members were imprisoned in EMR Nº1 (Libertad Prison) and EMR Nº2 (Punta de Rieles Prison), for men and women respectively. The rest sought political asylum in European countries, or died in combat or torture. Prisoners in EMR Nº1 and some exiles played a leading role in this story in the early 1980s involving Uruguay, Germany, and Sweden.

When nine members of their leadership were held hostages (“rehenes políticos”) in different barracks in September 1973, the Tupamaros

left in EMR Nº1 focussed on self-criticism inside the prison and felt the need to make their political positions known to their comrades in exile. The elaboration of self-criticism took them three years (1977-1980) in severe prison conditions on the second floor of the EMR Nº1 where the military kept the prisoners that they considered “dangerous”. The discussions necessary to reach a certain consensus, more or less representative, went through several stages and led to the final draft of the document and a detailed revision among many of them. Two prisoners were charged with the task of elaborating the key and deciding on the vehicle that would carry the message, as well as looking for ways to get the key to family members, so that they could transmit it into exile. A cryptographic *tapestry* (called *carpet* in (Cabezas et al., 2018)) emerged as a medium for clandestinely passing their elaborations and proposals to the outside.

They agreed on an 18-letter alphabet, reduced from the standard 27 letters of Spanish, and its representation in six columns and three rows:

<i>m</i>	<i>a</i>	<i>r</i>	<i>k</i>	<i>o</i>	<i>s</i>
<i>d</i>	<i>i</i>	<i>n</i>	<i>t</i>	<i>e</i>	<i>l</i>
<i>j</i>	<i>u</i>	<i>p</i>	<i>v</i>	<i>h</i>	<i>f</i>

Figure 1: Encryption key used in the *tapestry*.

A total of six colors was used. Each column

represents one of the six colors, and each row one of three among them. The encryption of a letter is the pair of colors at the intersection of its column and row. It was implemented as a *tapestry* elaborated with threads of wool of six different colors and made in the "cross stitch" technique on burlap. This was a craft accepted by the military and many inmates sent similar items outside, passing censorship.

It took many months to make the *tapestry*. First, the proportion and the size of the fabric, and then a reading direction were defined. The agreed text was written in the new alphabet, obtaining the pairs of colors for each letter of the alphabet. Then a *tapestry* design was made on common squared paper, and finally letter by letter, stitch by stitch, the text was transferred from the squared paper to the burlap.

In spaced and controlled visits, the prisoners tried to pass the code to the outside, with the logical difficulties implied in committing family members to things unknown to them. An inmate who was soon to be released and then to be expelled from the country was chosen to remember the key and transmit it to Germany. In exile, the key and the *tapestry* should converge to allow reading of the message. It was also intended that a message with news from the outside be returned and that the same key be used.

What happened to the key, to the alphabet? It came out of prison on thin cigarette rolling paper wrapped in nylon, forming a tiny *pill* that the released prisoner carried in his mouth, as a backup to his memory. But since life is always complex and much more so for those who in the midst of a dictatorship are "expelled" from their country, in a moment of tension he swallowed the pill containing the alphabet. The *tapestry* remained in Uruguay and never reached its intended recipients in exile.

Memory is deceitful and elusive. The exiled prisoner tried to reconstruct the code in order to send a message with news from exile, but only came up with one somewhat inspired by the *tapestry*'s. The released prisoner's new life, the health care given in Germany, the reunions with family and colleagues, the commitment to denounce the living conditions of

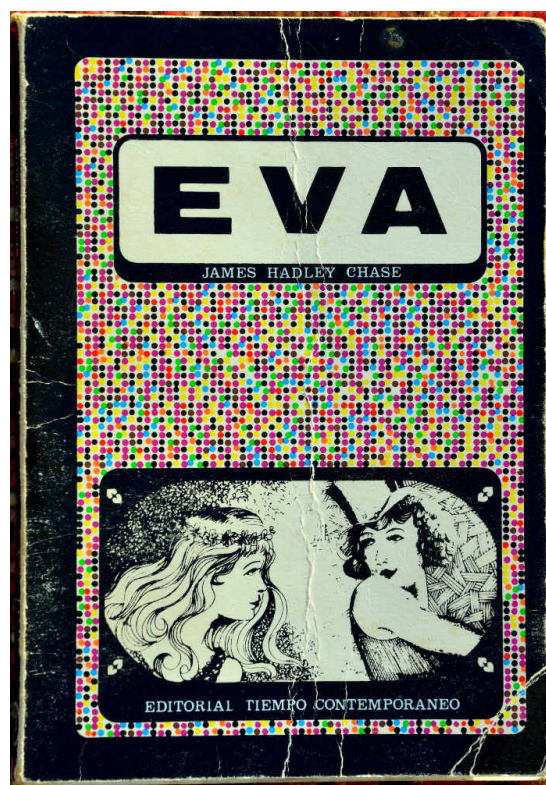


Figure 2: Front cover of the book "Eva".

prisoners in the Libertad Prison, the eagerness to know what had happened in the organization and in the world in all those years, undoubtedly conspired to hinder the recovery of the key. This inconvenience became the greatest obstacle for the successful deciphering effort almost forty years later ((Tiscornia and Cabezas, 2015), also (Cabezas et al., 2018)). That is the first part of this story.

The exiled Tupamaros, who did not even know of the *tapestry*'s existence, sent a message from Sweden to Uruguay. It was encrypted by colored dots on the covers of a little book entitled "Eva" (Figures 2 and 3) whose contents is irrelevant to our story. Two copies were printed in Sweden. One arrived in Uruguay at the time, but did not pass prison censorship. The coding used a method vaguely similar to that of the *tapestry*, but with some changes that complicated its recent analysis, which is the subject of the present work. This is the second part of the story.

Its third part is a *notebook* (*cuadernola* in Spanish) having on its covers the same plaintext as "Eva", but encrypted differently, with a key similar to that of the *tapestry*. The

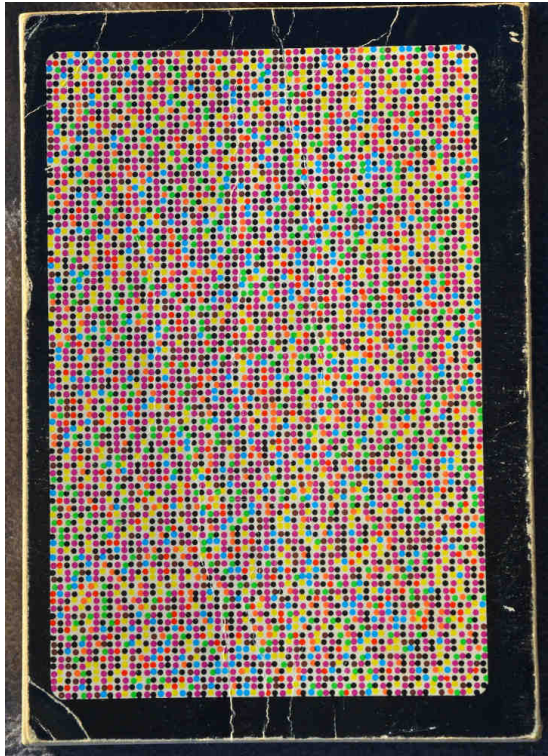


Figure 3: Back cover of the book “*Eva*”.

three key words MARKOS-DINTEL-JUPVHF were identical, but with the rows and columns encoded in “*Eva*”, using the run-length of black and magenta dots as the row indicator. It was printed in Sweden and arrived safely in the Libertad prison and had to be deciphered without any technical help. This demanded a great deal of effort due to alterations, as mentioned above, in the *tapestry*’s encryption method. Enclosed in their cells, against all odds and dedicating weeks of permanent work, eventually they managed to do this. It completed the whole route, but we do not have the *notebook* with us because it was destroyed inside the prison for security reasons as soon as its contents was copied. And this is the third part of the story.

In 2015, thirty-five years later, someone coming from exile in Sweden brings a small book entitled “*Eva*”, a copy of the original that had not passed censorship. Slightly embarrassed, he informs the ex-prisoner that its front and back covers contain the message that came to the Libertad prison in the *notebook*. They consist of regular arrangements of a myriad of colored dots. Whoever has been able to see the *tapestry* and “*Eva*” together under-

stands that the strategy has been the same. The little book is given to an ex-prisoner who has been working on the *Memory of the Libertad Prison*, not knowing the story we have been telling so far. So this researcher started from scratch, consulting his comrades. No prisoner had ever seen “*Eva*”, but some knew about the existence of the *tapestry*, and others had held it in their hands; step by step he advanced until finding the author of the *tapestry*. Although this person had never seen “*Eva*”, he recognized the colored dots. In the first discussion with him, the surprise was great when the researcher found the already moth-eaten *tapestry*. And then, in successive meetings, 35 years later, began a process of retrieving memories; trying to recover the key and the whole process of its creation.

The researcher, besides being surprised, tried to put a rigorous logic to the events in order to help memory, on the one hand, and on the other hand to construct an understandable story. Thus, first questions arose, and after the answers, certainties of what had happened, in order to arrive at the following conclusions. Only one person saw, no matter at what time, the *tapestry*, the *notebook*, and “*Eva*”. This is the author of the *tapestry*. Thirty-five years later, when shown “*Eva*”, of whose existence he had no prior knowledge, a circle is closed. The long-awaited message contained in the *tapestry* never left the home of the author’s family. Miraculously, thirty-five years later it was still there. After so much effort, so many dangers, and so much inventiveness and just out of EMR Nº1, the *tapestry* ran aground in his family’s house.

Almost forty years later, after a deciphering effort at INCO<sup>1</sup>, someone unrelated to the authors of the message was able to read it (Tiscornia and Cabezas, 2015). The *tapestry* ended its journey and revealed its content and the thoughts of those imprisoned Tupamaros.

For deciphering “*Eva*”, we have in our hands the *tapestry* and its alphabet and key, and the almost certainty “*Eva*”’s code should be the same. We also know that the *notebook* is a backup of the message sent as “*Eva*”, which never reached the hands of the prisoners be-

<sup>1</sup>Computing Science Department at the Universidad de la República, Montevideo, Uruguay

cause the censorship did not allow it. In other words, the *notebook* was a second attempt of a message from exile - this time successful - with many months, or years, between the two.

Time made more memory come to light in that expelled prisoner and with it the key to the *notebook*, in spite of small differences with the *tapestry*'s code. Eventually, the *notebook* could be deciphered. However, this did not help to solve the problem of “*Eva*”, but rather complicated it. Deciphering the *tapestry* was considered more complex due to the digitization itself, the discoloration of the wool, the structure of the fabric, the waves that are generated, and the moth eaten areas. These are the reasons why we started with the *tapestry*. Once again, chance, memory and oblivion intersect. In the course of “*Eva*”'s decipherment, it was detected that the encoding method was not the same as the one used in the *tapestry* and the complications multiplied. But the task was solved (Castro, 2019). All these steps were finished by confirming the correctness of the discovered plaintext with those who received the *notebook* and verifying with the “forgetful” ex-prisoner that the new alphabet was the one in Figure 10.

## 2 Previous Knowledge

For decoding “*Eva*”, the following previous information was known, or at least could be assumed as a starting point for decrypting the book covers:

- The text was written in Spanish without spaces and using an 18-character alphabet.
- The encoding method of the *tapestry*, namely a substitution cipher with each character encoded as two colors, one for the row and another for the column. (This assumption turned out to be wrong.)
- The key of the *tapestry* (forming a  $3 \times 6$  matrix): MARKOS-DINTEL-JUPVHF (Figure 1).
- The encoding method of the book covers and particularly the key should not be too different from the one on the *tapestry* since it was decoded by hand.

A few days before final success, an additional hint was provided: the author of the book cover remembered that there was a word in Latin somewhere. Which later proved to be true, as MORTIS was found (by computer) as part of the book cover's key.

## 3 Image Recognition

Compared to the *tapestry*, the book was found in good condition (Tiscornia and Cabezas, 2015). The inks used in the cover were sufficiently clear and well-preserved to easily distinguish the colors from each other and from the white background. This allowed us to use simple algorithms for the recognition.

The back cover consists of a  $61 \times 91$  array of colored circles, missing one per corner, totaling 5547 circles. On the front cover there is a rectangular space lacking circles where an image is located instead. That is why the analysis began with the back cover only.

Two difficulties during the recognition process are illustrated in Figure 4:

- Minor cracks, probably originating from bending the cover, causing thin yellow lines (from the acid paper) to appear behind a few circles.
- Each of the eight inks was manually printed from a different plate, and the plates were not perfectly aligned. The amount of misalignment was only problematic on the front cover.

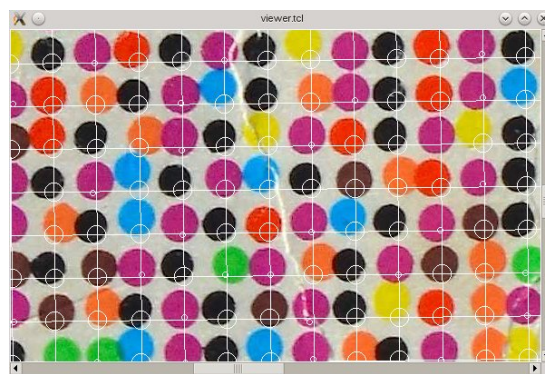


Figure 4: Snapshot of the recognition tool on the back cover. Smaller overlaid white circular marks show which circles were recognized.

A tool was created to assist in the recognition process.

1. First, the pixel coordinates for the centers of the circles located at the corners are specified, along with the number of rows and columns.

With this information the program is able to overlay a grid where an approximation to the center of each circle is calculated. This is done using bilinear interpolation.

2. When the application is run, it shows a list of the circles that it couldn't recognize. For each one of them the color at their center point is also shown (including the RGB values, each in the 0...255 range) at the center point.

The user may then specify the RGB values of recognized colors with their names, like `color 30 192 44 g` for green.

A circle is recognized when a pixel of a valid color is located in the area given by the center point ( $\pm 5, \pm 5$ ) pixel area. Valid colors are those whose RGB triplets differs less than 10 in the 2-norm of a recognized color. In case of ambiguity the point nearest to the center is used.

The front cover, being significantly shorter, was recognized and transcribed by hand once the back was decoded and the key was known.









It was then found that in fact the front cover is the continuation of the back, with the bottom half being the repetition of the beginning of the text.

## 4 Encoding deduction

### 4.1 Frequency count

After the image was recognized we only had a text file with a character per circle, and the knowledge that it was encoded with a method similar to the one used in the *tapestry*.

There are eight colors with the following number of occurrences:

Code	Count	Samples for each color
k	1501	 Black
m	1497	 Magenta
y	622	 Yellow
g	466	 Green
c	424	 Cyan
n	394	 Brown
o	350	 Orange
r	293	 Red

The first difference from the *tapestry* is that 8 colors were used instead of 6. Black and magenta are the two most common colors, yet no two consecutive black nor magenta circles can be found.

### 4.2 Conditional probabilities

When we started, we did not know the reading direction. In order to determine it, Figure 5 exhibits for a circle with color 'y', the probability to be immediately followed by a circle of color 'x'. Here,  $P(x)$  is the overall probability for color  $x$ , say  $P(k) = 1501/5547$ . Additionally, this table can also be seen as the transition table of a first order Markovian model.

The background hue of each cell in Figures 5 and 6 indicates the distance from a memoryless process; the more colorful, the more distant it is. Thus the stronger colors in Figure 5 say that reading is more likely to be horizontal than vertical. Formally the hue is calculated as  $h = c_1 \cdot \text{sigmoid}(c_2 \cdot (P(yx|y) - P(x)))$ , where green indicates positive and red negative  $h$ .<sup>2</sup>

By building the same table from the transposed array of circles, which would be in effect equivalent to reading the circles vertically, as done in Figure 6, we can see that the transition probabilities are much more similar to a memoryless model.

From this we can infer that the text is encoded horizontally.

### 4.3 Text direction via compression

Another hint that the text is encoded horizontally can be obtained by compressing the text file with the arrays of circles using a pure variant of LZ-77 (Ziv and Lempel, 1977)<sup>3</sup>, resulting in 2064 bytes compressed horizontally and 3014 bytes vertically.

### 4.4 Text direction via common substrings

A conclusive method to ensure the text was indeed written horizontally is to find the longest common repeated substrings in both directions. Figure 7 shows a text `mkmnkomkgn-gnmkmykycmkrmgc` of 26 circles appearing

<sup>2</sup> $c_1 = 32, c_2 = 10, \text{RGB color} = [224 - h, 224 + h, 224]$ .

<sup>3</sup>Variant with a sliding window of 4kB coding in 9 bits each uncompressed byte and in 17 bits (1 bit for the type, 12 for offset and 4 for length) the matches of length 3 to 18.

$P(yx y)$	$x=k$	$x=m$	$x=y$	$x=g$	$x=c$	$x=n$	$x=o$	$x=r$
$y=k$	0.00	44.40	15.53	10.67	8.73	7.07	6.53	7.07
$y=m$	45.16	0.00	15.76	9.95	7.82	8.08	8.15	5.08
$y=y$	31.99	31.19	1.93	2.89	8.04	10.61	7.40	5.95
$y=g$	30.69	30.26	4.08	1.29	7.30	15.45	5.58	5.36
$y=c$	29.48	33.49	8.96	14.39	3.54	4.01	1.89	4.25
$y=n$	34.26	32.74	5.84	3.30	7.87	1.02	9.90	5.08
$y=o$	35.14	31.71	8.57	12.57	6.86	2.29	0.29	2.57
$y=r$	33.79	38.91	10.58	5.12	7.51	0.00	3.41	0.68

Figure 5: Conditional probabilities from the text read horizontally.

$P(yx y)$	$x=k$	$x=m$	$x=y$	$x=g$	$x=c$	$x=n$	$x=o$	$x=r$
$y=k$	25.13	27.87	11.73	8.40	7.13	7.33	5.93	6.47
$y=m$	29.39	27.59	10.89	7.82	7.35	6.81	6.01	4.14
$y=y$	26.37	26.53	13.18	8.68	7.88	6.11	6.75	4.50
$y=g$	28.76	25.75	10.52	8.37	6.87	7.51	6.44	5.79
$y=c$	28.54	23.82	12.74	5.42	8.02	8.49	6.13	6.84
$y=n$	25.63	26.14	7.36	11.42	10.66	7.61	7.61	3.55
$y=o$	26.86	26.00	9.43	10.29	6.86	7.43	7.71	5.43
$y=r$	23.89	29.01	12.29	8.87	8.87	5.80	5.46	5.80

Figure 6: Conditional probabilities from the text read vertically.

nkmckcmkmokmykymkymyookmknmkycmrkmgccckomkmnkcomkgngnmkykycmkr  
 gmgcroygkmykomkynymkmnkcomkgngnmkykycmkrmgckmnkomkrymykmykmk  
 rcnkymrkykyomkygmrokmckomkymkmnkcomkgngnmkykycmkrmgckmrykmk

Figure 7: Example showing the longest repeated string.

thrice in the text, with a newline breaking the first two lines which are contiguous in the original.

#### 4.5 Kasiski test

The Kasiski test is commonly used to obtain the key length for Vigenère cryptosystems, and even though we assume from the start that Vigenère was not used, this test can provide additional information.

In the Figure 8 we present the number of pairs of circles of the same color (*#matches*) with the same distance *offset* between them, *factors* shows the prime factorization of the offset.

#matches	offset	factors
1112	173	173
1112	1842	2,3,307
1115	1071	3,3,7,17
1119	782	2,17,23
1121	2679	3,19,47
1122	2294	2,31,37
1126	289	289
1130	2148	2,2,3,179
1130	2538	2,3,3,3,47
1166	6	2,3
1175	530	2,5,53
1562	3	3

Figure 8: Top values of the Kasiski test ( $\bar{x} = 1015.80$ ,  $\sigma = 41.47$ )

Since there are only 8 colors to encode an alphabet of 18 characters (assuming it was the same alphabet as in the *tapestry*), we have a multi-symbol cryptosystem, where more than one symbol is needed to encode each charac-



ter. The Kasiski test could then be used to obtain a clue about the number of circles per characters needed, in the case that such fixed value existed.

Even though a higher number of coincidences were found at offsets 3 and 6 (vs 1, 2, 4 or 5), the number of matches for offsets congruent with  $i$  modulo 3 show a difference small enough to discard trilateral or trinomic cryptosystems (US Army, 1990):

offset	$\bar{x}$	$\sigma$
$\cong 0_{(3)}$	1015.08	37.63
$\cong 1_{(3)}$	1015.23	47.14
$\cong 2_{(3)}$	1017.10	38.97

#### 4.6 Black and magenta runs

For any subset of two or more colors, we consider the substream of those circles whose colors belong to the subset. When we compress these substreams with LZ77, we obtain a significant result: black and magenta circles reach the maximum compression ratio. This occurs because they are interspersed (k m k m k m . . .).

If we treated black and magenta as identical colors, then the ciphertext would still have the same meaning. We observed that the run lengths for consecutive black and magenta circles follow a pattern: There are 638 runs of length 1, 694 of length 2, 324 of length 3, but none of length 4 or higher.

At this point we correctly suspected that the run length was pointing out the row number of the  $3 \times 6$  key, with the other 6 colors referencing the column. This was one of the key observations leading to our cryptanalysis.

**Example 1** *The first few circles of the back cover can then be decoded as:*

- **kmyc**: in row 2: yellow and cyan,
- **ky**: in row 1: yellow,
- **mkgr**: in row 2: green and red,
- **mo**: in row 1: orange,
- **kmkn**: in row 3: brown,
- **moy**: in row 1: orange and yellow,
- **kmo**: in row 2: orange,
- **ko**: in row 1: orange,
- **mkmr**: in row 3: red.

The sequence of black and magenta run lengths starts with (21213121321232312121-2112121232. . .). Few numbers are repeated consecutively. This suggests that in most places when there are two or more consecutive letters from the same row, the row number is indicated only once.

A few repetitions can be seen in the first part of the back cover. However from about midpage on there are no more consecutive runs with the same length. This can be observed in Figure 9. Maybe they realized during the production process that repeated run lengths could be optimized out.

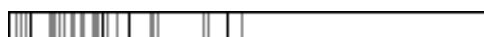


Figure 9: Positions where consecutive repeated black and magenta run lengths are found on the back cover.

Assuming that our hypotheses are correct at this point, the “Eva” code is **not** a simple substitution. The alphabet is split into three parts (rows) of six letters each. In each part, a letter is indicated by one of six colors. Two special colors (black and magenta) indicate the part to be used, but not by colors, rather by the black/magenta run length. We are not aware of any historical cipher that employs such a two-step encryption.

However, we can rewrite it as a simple substitution. The first entry **kmyc** in Example 1 then becomes  $((2, y), (2, c))$ , and we now work with this simple substitution. But it required a major effort to discover this unusual and clever step in the encryption.

## 5 Simple Substitution

Once we have reduced the encoding to a simple substitution, we need to know which letter corresponds to each of the 18 entries in this  $3 \times 6$  table. This table forms the secret key of the resulting cipher.

Traditionally such ciphers are broken by frequency analysis, a process which begins by counting the number of times each character appears in a corpus and in the encrypted text. Since the ratio depends mainly on the language used and applying the substitution does not change the ratios, it is expected to have

a similar order both in the corpus as in the encrypted text.

Instead of using just the frequencies for the individual characters, digrams are commonly used as well (Clark and Dawson, 1998).

In our case we used the simple hill climbing ILS greedy algorithm (iterative improvement local search) which starts from a random solution *key* and keeps looking for the best neighboring one until a local maximum is reached (Lourenço et al., 2003):

```

procedure ILS(key)
   $best \leftarrow \max_{n \in \text{neighbours}(key)} \{\text{fitness}(n)\}$ 
  if  $\text{fitness}(best) > \text{fitness}(key)$  then
    return ILS(best)
  else
    return key
  end if
end procedure

```

A key is a permutation of the 18 letters in our alphabet  $\mathcal{A}$ , and two keys are neighbors if one can be obtained by swapping two entries of the other.

### 5.1 Fitness function

For the criterion used to prefer one key over another we use a “fitness” function. Such functions return higher values whenever the plain text obtained (by using a “better” key) has some property closer to a given corpus, like the relative frequencies of letters or polygrams.

A first approach to a fitness function based on a dictionary is just breaking the text in words, so that the sum of the lengths of the words that can also be found in a Spanish dictionary is maximized.

Furthermore we can power the length of the words to a positive constant  $\alpha$  to increase the weight of longer words. The longer the word, the lower its probability is to appear inside garbled text.

**Example 2** Given the text HELLONE and an English dictionary,  $\alpha = 1.8$  would prefer a higher coverage selecting “hell”+“one”, while  $\alpha = 2.2$  would prefer a longer match “hello”.

$\alpha$	‘hello’	‘he’, ‘lone’	‘hell’ ‘one’
1.8	18.1	15.6	19.4
2.0	25	20	25
2.2	34.5	25.7	32.23

Formally, our fitness function  $\text{fitness} : \mathcal{A}^* \mapsto$

$\mathbb{R}$  is defined as:

$$\begin{aligned} \text{fitness}(z) &= \max \{ \text{fitness}(\phi) + |\omega|^\alpha \\ &\quad \forall (\phi || \sigma_1 || \omega || \sigma_2) = z, \omega \in \mathcal{D} \}, \\ \text{fitness}(\varepsilon) &= 0. \end{aligned}$$

where:

- $\phi, \sigma_1, \omega, \sigma_2 \in \mathcal{A}^*$ ,
- $\varepsilon$  is the empty string,
- ‘||’ is the string concatenation operator,
- $|\omega|$  is the length of the string  $\omega$ ,
- $\alpha$  is a constant applied to the length, and
- $\mathcal{D} \subset \mathcal{A}^*$  is the Spanish dictionary whose characters were projected to the alphabet  $\mathcal{A}$  (without accents or diresis).

The  $\alpha$  parameter can be calibrated to make the function have stronger preference to finding longer words vs. finding more words (a higher percentage of the text covered by recognized words). A value of 1.8 was used.

Trying all possible partitions of a text has exponential cost. By remembering (dynamic programming) the fitness result of the first  $i$  characters and keeping the dictionary in a trie, we can implement the fitness function with  $\mathcal{O}(m)$  memory usage and  $\mathcal{O}(n \cdot m)$  CPU usage, where  $n$  is the text length and  $m$  is the length of the longest word in the dictionary.

### 5.2 Result

After several executions of the algorithm, the correct key is obtained when the maximum fitness value is obtained:

	r	c	y	n	o	g
1	m	o	r	t	i	s
2	k	l	a	n	d	e
3	j	u	p	v	h	f

Figure 10: The key of “Eva”.

The *tapestry* and the book cover share the third row of the key. However instead of “markos”/“dintel” here we have “mortis” as Latin for death and “klande” as short of “clandestino” (Spanish for clandestine).

Obtaining or not the correct key in a series of independent executions can be modeled as a Bernoulli process of a given probability  $p$ .

Since this key was obtained in 643 of 10000 independent executions, we know that the probability of *not* obtaining a correct key in 500 independent executions given  $p > 0.05$  has a trivial upper bound of  $(1 - p)^{500} < 7 \cdot 10^{-12}$ .

On the other hand, given  $p \leq 0.05$  the probability of finding the correct key at least 643 times in 10000 independent executions would be less than  $1 - \text{bin}(k = 643, n = 10000, p = 0.05) \approx 1.8 \cdot 10^{-16}$ , where  $\text{bin}$ , the binomial cumulative distribution function, equals  $\sum_{i=0}^{\lfloor k \rfloor} \binom{n}{i} p^i (1 - p)^{n-i}$ .

Thus running 500 independent executions of this algorithm implies testing an average of 1.15 million keys, with the probability of *not* finding the correct key upper-bounded by  $10^{-10}$ . That said, if we were to do a 1.15 million naïve random searches for the key, the probability of *finding* it would be less than  $1.8 \cdot 10^{-10}$ .

## 6 Parts of the document's cleartext.

We present the initial part of the cleartext. The opinions and political points of view expressed in this document do not, in any way, reflect necessarily those of the authors or their institutions.

Al recibir digan: Magdalena  
llegó.

Colores marcan renglones.

Cuando salí no revisaron boca.

[...]

Hay agrupamiento tupas en base a acuerdos, con discrepancias llamado proceso, o simposio. Coincido bastante con ellos, todavía embrión, falta mucho.

Fracasos anteriores, recelo y desesperanza, exilio jode gente.

Va a ser largo y difícil, derrota muy gruesa.

Viejas desviaciones, Cros en Chile, Argentina, Cuba.

Europa: fui bienvenido por todos, hablo con todos grupos, homogéneos y sueltos, no pude hablar (con) seis puntos.

Empezamos a organizar trabajos prácticos, necesarios para MLN, algunos quieren arrimarse América.

Hay unos pocos en Salvador, Nicaragua, Cuba, Colombia.

Intento solidaridad, reorganización, impulso tarea, me queda enorme.

Gran respeto pensamiento (del segundo. Estamos mas frescos que exilio. PUMA (Pautas Unificadoras Mínimas y Amplias) muy bien recibido, coincidencia. Olvidé partes, saquen de nuevo. Importante sigan trabajando ustedes. <sup>4</sup>

## 7 Conclusions

None of the Uruguayan Tupamaros had any in-depth knowledge of cryptography, but still they succeeded in encrypting three fairly long messages in different ways: the *tapestry*, the *notebook*, and the small book “*Eva*”. The *notebook* was deciphered and then destroyed on purpose in its time, the 1980s, but the other two items survive. Both are esthetically pleasing. Their contents were deciphered only in recent years.

For the *tapestry* and “*Eva*”, special-purpose image recognition software was designed in order to translate the sequences of colors into machine-readable text. Understanding the *tapestry* encryption gave several hints for

<sup>4</sup>When you receive this, notify by saying “Magdalena has arrived”. Colors mark rows. When I left [the prison], they did not check my mouth.

There are groups of Tupamaros based on agreements, with discrepancies called process or symposium. I basically agree with them, still in embryonic state, that much is missing. Previous defeats, retreat, and despair, exile fucks up people. It will be a long and arduous way, after a big defeat.

Old disagreements, comrades from Chile, Argentina, Cuba. In Europe, it was welcomed by everybody, I talk to all groups, homogeneous and unattached, I could not talk (to) six groups. We start organizing practical work, necessary for MLN, some want to go with America. There are a few in Salvador, Nicaragua, Cuba, Colombia.

I go for solidarity and reorganization, push towards the goal, but much remains to be done. Great respect for the thinking on the second floor [where the “dangerous” prisoners were housed]. We are fresher [our political thoughts here in prison are clearer] than those in exile. PUMA (minimal and broad unifying guidelines) well received, a coincidence. I forget some parts [of PUMA], please send them again. The important thing is for you to keep working.

“Eva”. Some of them proved useful, for example the arrangement of 18 letters in a  $3 \times 6$  array. Others did not. The arrangement of the 18 row/column pairs was done just by colors in the *tapestry*, but no such arrangement worked for “Eva”.

Statistical experiments led to the insight that the run length of black and magenta circles indicates the row number, and any color(s) following such a run indicate the column(s) to be encoded. Once this unusual method was understood, a hill climbing algorithm with an appropriate fitness function broke the resulting simple substitution of “Eva”.

Overall, we have unique encryption systems designed by amateurs. They embedded sufficient steganographic cleverness to pass critical inspection by prison guards, were hand-coded with substantial efforts, and the coding method is sufficiently strong to resist attacks by equal adversaries, namely by hand.

## References

- Juan José Cabezas, Joachim von zur Gathen, and Jorge Tiscornia. 2018. Uruguayan cryptographic carpet. *Proceedings HistoCrypt 2018*, 21-27, <http://www.ep.liu.se/ecp/contents.asp?issue=149>.
- Francisco Castro. 2019. Printed book cover cryptanalysis: A formal approach. Master’s thesis, Facultad de Ingeniera, Universidad de la Republica. In progress.
- Andrew Clark and Ed Dawson. 1998. Optimisation heuristics for the automated cryptanalysis of classical ciphers. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 28:63–86.
- Helena Ramalinho Lourenço, Olivier C. Martin, and Thomas Stützle, 2003. *Iterated Local Search*. In: *Handbook of Metaheuristics*, pages 320–353. Springer US, Boston, MA.
- Jorge Tiscornia and Juan José Cabezas. 2015. El código secreto del tapiz del MLN hecho en el Penal de Libertad. *Reportes Técnicos 15-10*.
- US Army. 1990. *Basic Cryptanalysis. Chapter 5: Monoalphabetic Multilateral Substitution Ciphers*. Headquarters Department of the Army. Washington DC.
- Jacob Ziv and Abraham Lempel. 1977. A universal algorithm for sequential data compression. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 23(3):337–343.

# The DECODE Database

## Collection of Historical Ciphers and Keys

Beáta Megyesi, Nils Blomqvist and Eva Pettersson

Department of Linguistics and Philology

Uppsala University, Sweden

firstname.lastname@lingfil.uu.se

### Abstract

We present an on-line database DECODE consisting of encrypted historical manuscripts, aiming at the systematic collection of ciphers and keys to create infrastructural support for historical research in general, and historical cryptology in particular. The collected material is annotated with a metadata scheme developed specifically for historical ciphers. Information includes provenance and location of the manuscript, computer-readable transcription, possible decryption(s) of the ciphertext and translation(s) of the plaintext, images, and any additional materials of relevance to the particular manuscript. The database allows search in the existing collection and upload of new encrypted sources by users.

### 1 Introduction

According to some historians, 1% of the national archives and libraries in Europe contain secret messages, encrypted hand-written manuscripts, intended to hide the content of the message with the exception of the intended receiver(s). Keys used for encryption might also be found, often without being stored together with the encrypted or decrypted message. The manuscripts in the libraries and archives are seldom indexed as ciphers, which makes it difficult to find them unless you know the librarian with extensive knowledge about the library's selection. Historians and other scholars interested in our history stumble on these manuscripts when searching for sources from a particular period of their interest. They try to decrypt the hidden source to shed some new light on our history — to find new interpretations and explanations.

However, it is far from trivial how to crack a secret message and there is a clear lack of infrastructural support in terms of data resources and automatic tools that historians and others without any knowledge in cryptology can use to reveal the content of the hidden message. In order to develop tools for automatic decryption, we need large(r) collections of ciphertexts and keys to develop better algorithms and systematically evaluate them on various cipher types.

In this paper, we describe an on-line database, DECODE<sup>1</sup> aiming at the systematic collection and description of ciphers, keys and related documents. The database comes with a graphical user interface that allows simple and advanced search in the existing collection for all users, and upload of new ciphertexts and keys by users with an account.

The DECODE database is one of the first steps towards an infrastructure for historical cryptology. Our goal is to collect ciphertexts, codes, keys, and codebooks from various archives and libraries as well as from the public, and to develop tools to support the transliteration of images into computer-readable format, cryptanalysis, and to make the resources and tools available to people interested in historical cryptology. Our hope is that users will contribute to enlarge the database by uploading new material for a growing collection, a monitor corpus of historical ciphers and keys.

The paper is structured as follows. In Section 2, we describe the general architecture of the database followed by a detailed description of the metadata, a set of features with their possible values for the description of the encrypted manuscripts in Section 3. Information

---

<sup>1</sup><https://cl.lingfil.uu.se/decode>

about cryptanalysis and decryption, including standards for transcription/transliteration of the images is described in Section 4. Then, we present the search design in Section 5, and the upload and editing functions in Sections 6 and 7. User access roles are described in Section 8 followed by a brief technical description of the database in Section 9 and some tools for the automatic processing of ciphers in Section 10. Lastly, in Section 11, we conclude the paper and give directions for future research.

## 2 The DECODE Database

The database was developed between 2015-2018, to cover a large range of different cipher types and keys for various plaintext languages from early modern time. We expected that the Vatican archives would be the right place for our endeavor, given papal correspondence throughout the centuries with many countries, with many (European) languages. Luckily, the Secret archives of the Vatican and the main library of the Vatican are well-organized archives with indexes over the encrypted manuscripts' whereabouts and description of their provenance. Within a few weeks, we could collect over 300 ciphers and some keys, and order images from the archive. This dataset became the starting point i) for a systematic description of ciphers and keys, ii) to develop the database with a search function, iii) to draw up guidelines for transliteration of ciphers, iv) to develop tools for semi-automatic transcription using hand-written text recognition, v) to implement tools for cryptanalysis, and vi) to map available ciphers with their corresponding key in the database.

During the past year, the database has been publicly released and three historians were asked to upload their cipher and key collection to test the functionality of the database. At the time of writing, the collection contains nearly 1000 records, ciphers and keys with images, and description of their current location, provenance, content, along with related documents including transcriptions/transliterations, cryptanalysis, related generated key, deciphered plaintext, possible translation(s), references to publications and other information of interest. Each record is

marked with the name of the user who uploaded the record and the date for entering the record into the database.

The ciphers and keys are collected at various archives and libraries in Austria, Belgium, Germany, Hungary, Italy, the Netherlands, UK, and the Vatican City. Among the dated records, the earliest ones originate from the 15th century, and the latest from 1793. About 33% of the material consists of original keys. Out of 634 ciphers, 205 are decrypted, and 232 are transcribed as running text allowing further processing for cryptanalysis. Among the records, we find plaintext languages in Dutch, English, French, German, Hungarian, Italian, Latin, or a combination of these (e.g. English-Latin, Hungarian-Latin, Italian-Spanish).

The majority of the records are short, one-page images, but we also find longer ciphers, the longest 410 pages. The great majority of the ciphers are encrypted with numbers, but ciphers with alphabetic characters and esoteric symbols such as zodiac and alchemical signs are also present. The known cipher types in the database are mostly based on simple substitution or homophonic substitution, with or without nomenclatures, but polyphonic substitutions also appear.

To browse the database, we developed simple and advanced search functions that are available to the public. The graphical interface is accessed using a web browser, making it usable on various operating systems and devices (smartphones, tablets, etc.) Experts in the field of historical cryptology may also apply for a database account to be allowed to add new records, and edit existing ones. In order to be able to enter and store a record in the database, image(s) showing the original cipher or key are required to ensure that the record actually exists. Images and other related documents that cannot be distributed for copyright reasons can be marked as private by the user. In such cases, the private documents/images are accessible only to the owner of the record, i.e. the person who uploaded the original document, and the database system owner(s). The database is publicly available for search but private images or documents related to the records are neither visible nor accessible to anyone, except for the owner. For

users with login, private images/documents are shown as miniature pictures but the documents are not downloadable.

On the basis of the initially collected records present in the database, we developed a metadata scheme for the description of ciphers and keys, which will be described next. Our hope is that the metadata can serve as basis for a standardized description of ciphers and keys.

### 3 Describing Ciphers: Metadata

Each manuscript is described according to a subset of metadata structured as attribute-value pairs. The structured information is divided into three fields: the current location of the manuscript, information about its content, and the form of the manuscript.

#### 3.1 Current Location

The current location of the manuscript, being it a ciphertext or a key, is mandatory information divided into three attributes: *Country*, *City*, and *Holder*. Country and City relate to the current location of the document, while Holder refers to an institution or a person who owns or keeps the document today.

#### 3.2 Origin

The field *Origin* gives information about the origin of the manuscript: the *Dating* or time period when the document was created, the name of the *Author* of the manuscript, the *Sender(s)* and the *Receiver(s)*, which can be an institution or a person, as well as the place, the *Region* and the *City* of origin. Given that we often do not know much about the provenance of the manuscript, information about the origin of the document is supplemental. However, knowing something about the provenance of the manuscript might be highly valuable during the puzzle of the decryption process, for example to make educated guesses about the possible plaintext language(s) behind the encrypted document.

*Dating* can be given as a specific date or a time period during which the document is assumed to have been created. Dates are represented in the proleptic Gregorian calendar, and follow the convention of ISO 8601 where 1 BCE is represented as year 0, 2 BCE is represented as -1, and so on. The year, month

and day (in that order) must be separated by a dash (-). The user can enter the year only, and it is interpreted as an interval spanning over the entire year. For example, 1542 is interpreted as an interval between 1542-01-01 to 1542-12-31 (i.e. from January 1, 1542 to December 31, 1542). If the exact dating of the document is not known, an interval can be specified with a starting and ending date, separated by a colon (:). Thus, 1542:1570 is interpreted as the interval between 1542-01-01 to 1570-12-31. If the month or day is not specified in the ending date, they will default to the last day of the last month. For example, 1542-05:1570-03 is interpreted as the interval between 1542-05-01 to 1570-03-31.

Concerning naming the sender and receiver of the manuscript, and naming the city or region where the manuscript originates from, the user is allowed to fill in the full or partial name of the *Author*, the *Sender(s)* and the *Receiver(s)*. The *Region* and *City* of origin might be given as original names, i.e. the historical name of the region or city used when the document was created. For example, if a manuscript originates from the current capital of Slovakia, Bratislava, it is up to the user to decide whether to name the city by its old German name *Pressburg*, use the old Hungarian name *Pozsony*, or give the current name *Bratislava*. We are aware of the fact that the lack of standard concerning the name of cities and regions throughout the history might confuse users, and raises higher demands on the user when searching in the database.

#### 3.3 Content

Describing the content of the encrypted document includes ten different types of optional attributes. *Number of pages* gives, as its name indicates, the number of pages of the ciphertexts or key given the image, excluding entirely non-encrypted plaintext pages, such as title pages or envelope. This is intended for quick search, for example, for shorter or longer ciphertexts.

The manuscript can either be a cipher, or a key, which is defined in the *Cipher/Key* field as predefined values. Each cipher can also be marked on the basis of its *Status*, in terms of whether the cipher is decrypted, non-decrypted, or partly decrypted. For keys, the

value of *Status* is non-applicable (N/A). For each record, it is indicated if it is a publicly available record with downloadable images and documents, i.e. if the record is *Public cipher/key* or not.

*Cipher type* is predefined and can be added to keys, ciphers and codes. If the type is not known, the value UNKNOWN applies. In case the type is known, the following predefined values might apply: homophonic substitution, nomenclatures, polyalphabetic, simple substitution, transposition, or a combination of those. If the cipher is of another type than the predefined values, there is an option to fill in a user-defined one in the OTHER field.

Encrypted documents consist of many different *Symbol sets*, numbers, alphabetical characters from various alphabets (e.g. Latin or Greek letters), diacritics, esoteric symbols such as zodiac or alchemical signs, logograms, punctuation marks, and a combination of those. In the database, the user is allowed to give information about the symbol set used in a particular manuscript, predefined values for alphabet, esoteric, and/or numerical symbols. White space in the original ciphertext might be present intentionally to keep word boundaries during encryption, between code groups, or unintentionally when producing the encrypted message. The user can indicate whether the ciphertext includes white space or not. If the encrypted manuscript contains other symbols than the pre-defined list of symbols, the user can define his/her own.

Lastly, the encrypted manuscript might contain encrypted sequences, i.e. ciphertext only, but plaintext, non-encrypted sequences of words, sentences, paragraphs and even pages might occur embedded inline in the message. Figure 1 shows an encrypted message with the plaintext <comé la mi comanda> in the ciphertext (ASV, 2016a).

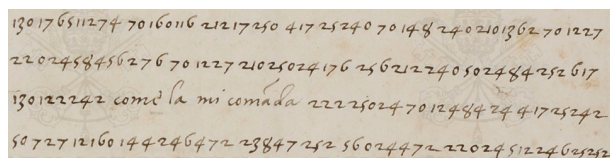


Figure 1: Extract from a cipher, with cleartext embedded in the ciphertext.

The cipher might contain not only em-

bedded non-encrypted plaintext, but also decrypted plaintext, often written over the ciphertext sequences by the receiver, see Figure 2 (ASV, 2016b).

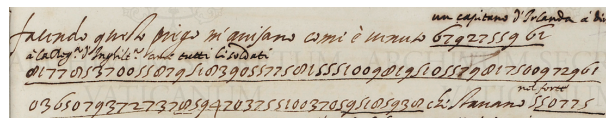


Figure 2: Extract from a cipher with cleartext embedded in the ciphertext, and decrypted plaintext.

Therefore, there is a need to differentiate between the plaintext, written in the original language and in which the ciphertext is embedded (or vice versa), and plaintext representing the decrypted ciphertext. We define plaintext as the decrypted ciphertext, and cleartext, as a non-encrypted text embedded in the message. According to the above, we indicate in the metadata description whether the text contains *Inline cleartext*, *Inline plaintext*, or both. In Figure 2, both *Inline cleartext* and *Inline plaintext* are added as values.

Similarly, the *Cleartext language(s)* (if any) and the original underlying language of the cipher, i.e. the *Plaintext language* can be defined by the user, as optional fields.

### 3.4 Format

Another optional field is the *Format* aiming at the description of the *paper* or *ink type*, for codicological studies to date a particular manuscript. The user can fill in these fields as free text. Typically, these fields are used for notes about the quality of the paper/parchment, whether it is damaged, or difficult to read or interpret due to bleed-through ink, or just a bad photocopy.

### 3.5 Other

There is also an option to provide links to available publications and other information about the manuscript in text format in the field *Additional information*.

### 3.6 Related Documents

In addition to the description of the ciphertext or key given as metadata fields, there is the possibility to add various types of documents describing the manuscript. These can



be transcriptions or transliterations of the manuscript (*Transcription*), the decrypted or decoded plaintext of the cipher (*Deciphered text*), generated key(s) (*Key*), statistics and cryptanalysis of the ciphertext (*Cryptanalysis*), or any other relevant document (*Miscellaneous*).

Many manuscripts containing ciphertexts or keys are buried among letter correspondences written in cleartext. These might be of relevance for the historical interpretation and contextualisation of the manuscript and could also be helpful for cryptanalysis to make educated guesses about the topic of the document, to crack encoded named entities such as place or personal names, and so on.

The plaintext might be translated and here, the user can upload *Translation(s)* of the plaintext to various languages.

If there is any published material about the cipher or key, the user can upload these (*Publications*), or add references as a single file.

The documents can be uploaded in various formats (txt, doc, docx, pdf) and most image types are allowed (e.g. png, tiff, jpeg). When uploading a document, the user is asked to name the document and categorize it by its type. The user can decide whether to create the documents with free access and downloadable to everyone, or to keep these private so other users won't be allowed to access those. The images can be uploaded as a single file, or as multiple files stored in the same folder by holding the Control key (Windows/Linux) or the Command key (Mac) while clicking on the desired files.

## 4 Analyzing Ciphers

In addition to the cipher collection and browsing function provided, we develop tools for the automatic transcription and decryption of ciphers. Given an image representing a cipher or key, the first step is to transcribe or transliterate the ciphertext into a computer-readable format. Then, the transcribed ciphertext can be statistically analyzed using various metrics (n-gram frequency, clusters, index of coincidence, entropy measures), and decrypted. Keys and ciphers can also be mapped automatically, calling for language models for various European languages. In the subsequent

sections, we give an overview of the transcription guidelines, and tools for cryptanalysis and cipher-key mapping.

### 4.1 Transcription/Transliteration

Usually, the first step in attacking a cipher is the conversion of the image into a machine-readable format, represented as text. There are many different ways of transcribing or transliterating a manuscript. Therefore, we developed guidelines so that the transcriptions available in the database have a common format.

Each transcript file of a particular cipher (which may consist of multiple images) starts with comment lines with information about the file. Each comment line starts with "#" followed by a transcription attribute and its value, as illustrated below:

- #CATALOG NAME: your own index, i.e. file location: e.g. /Segr. Stato Francia 6/1
- #IMAGE NAME: the name of the image(s) representing the cipher: e.g. image interval 234r-237v.jpg
- #TRANSCRIBER NAME: full name or initials of the transcriber: e.g. BeMeg
- #DATE OF TRANSCRIPTION: the date the transcription was submitted
- #TRANSCRIPTION TIME: the time it took to transcribe all images of a cipher in hours and minutes without counting breaks and quality checks
- #COMMENTS: description of e.g. difficulties, problems

Next, the content of the image is transcribed. Each new image in a cipher starts with a new comment line with information about the name of the image followed by a possible comment line:

- #IMAGE NAME: the name of the image, e.g. 234v.jpg
- #COMMENTS: any comments, e.g. difficult to read line 3, bleed-through

The transcription is carried out symbol by symbol and row by row keeping line breaks,

spaces, punctuation marks, dots, underlined symbols, and cleartext words, phrases, sentences, paragraphs, as shown in the original image. Line breaks are kept so that when a new line starts, a new line is added in the transcription. Punctuation marks, such as periods, commas, and question marks are transcribed as such. Space is represented as space. If there is a larger width of a space in relation to other spaces in the ciphertext, two or more space characters can be entered in the transcription. The reason for allowing several space characters is that a larger space in the original might mark word boundaries which the encryptor unintentionally left there when encrypting the manuscript, which can be helpful in the decryption process as they might denote word boundaries.

Sometimes, punctuation marks (e.g. dots, commas, underscores) appear above or under specific symbols. It could be ink splash, but if they appear in a systematic way, they are transcribed as well. If the mark appears above the symbol, the sequence is transcribed as the symbol, followed by “^” and the specific mark (e.g. dot or comma). If the mark appears under the symbol, it is marked by an “\_” placed between the symbol and the mark “.” (e.g. s\_.). Similarly, underlined symbols are marked with “\_” immediately following the symbol, except when the whole ciphertext is underlined.

Uncertain symbols are transcribed with added question mark “?” immediately following the uncertain symbol. Possible interpretations of a symbol can be transliterated by transcribing the options using the delimiter “/”. For example, if it is not clear if a symbol represents a 0 or 6, it is transcribed as “0/6?”.

The cipher sequences might be embedded in cleartext, or cleartext might be embedded in ciphertext, see Figure 3. We can also find cleartext in keys, often explanations about the key. To be able to distinguish between ciphertext and cleartext sequences, the latter is clearly marked in brackets as <CLEARTEXT LANG letter/word sequence>, where the tag <CLEARTEXT ... > denotes where the cleartext starts and ends. LANG represents the language the cleartext is written in, marked by ID as defined by ISO 639-1 two-letter codes

for languages (e.g. IT for Italian), and UN for unidentified languages.

```
130176511274701601162121725041725240701482402101362701227
220245845627670122721025024176 25 621224 0502484252617
1301222 2 <CLEARTEXT ES comè la mi comanda> 22225024701248474417 25242
50727121601442464723847252 560244722202951224625212
```

Figure 3: Transcription of the cipher image in Figure 1.

Sometimes we can find the decrypted plaintext written above the ciphertext. Similarly to cleartext, plaintext is transcribed as <PLAINTEXT LANG letter/word sequence> in a separate line. Transcription of the image containing cleartext and plaintext of the original image in Figure 2 is shown in Figure 4. Note that if the cipher is cracked, the entirely or partly deciphered parts, i.e. the plaintext, can be uploaded as a separate file (see Section 3.6).

Transcription reflects the intention of the encoder, i.e. the corrected segments are transcribed. For example, if numbers are crossed-off in the original, these are not transcribed. Similarly, insertions of corrections between symbols are transcribed, as they intended to appear in the original. Ciphertext/cleartext written in the margin is added into the specific space as indicated by the given mark/note in the original cipher.

Not seldom, historical manuscripts contain catchwords placed at the foot of the page to mark page order (instead of numbers). Catchwords are a sequence of symbols anticipated as the first symbols of the following page. In ciphers, catchwords might denote an actual word, unintentionally, and therefor transcribed as <CATCHWORD symbol sequence> (e.g. <CATCHWORD 1 1 2 0 8 9>).

The transcriptions are uploaded as text files, represented in Unicode (utf-8) format. Several transcription files are allowed to be uploaded of the same cipher/key, and they should be uploaded as text files (.txt, docx, etc).

## 4.2 Cryptanalysis

The transcribed ciphertext can be analyzed by using various metrics. Attacking and eventually cracking ciphers might involve many different types of cryptanalysis. We implemented the ManuLab statistical analyzer (Antal and Zajac, 2018) for ciphers containing metrics for

```

<CLEARTEXT IT facendo questo prego n'anisano? come è menuto> 6_7_9_2_7_5_5_9_6_1_<PLAINTEXT IT un capitano d'Irlanda à dire>
<PLAINTEXT IT à la Reg.a l'??? arta? tutti le soldati>
8_1_7_7_8_5_3_7_0_0_5_5_8_7_9_5_1_8_3_9_0_5_5_7_7_5_8_1_5_5_5_1_0_0_9_8_1_9_5_1_0_5_5_7_9_8_1_7_5_0_0_9_7_2_9_6_1_
0_3_6_5_0_7_9_3_7_2_7_3_7_8_5_9_4_7_0_3_7_5_5_1_0_0_3_7_0_5_9_5_1_8_5_9_3_8_<CLEARTEXT IT che Stauano?> 5_5_0_7_7_5_<PLAINTEXT IT nel forte>

```

Figure 4: Transcription of the cipher image containing ciphertext, cleartext and plaintext in Figure 2.

index of coincidence, n-grams up to 8 characters, and entropy. The user can upload a ciphertext, run the appropriate metrics, and get analyzed data back. Then, the user can upload his or her own analysis to the database for the particular cipher, see Section 3.6. The analysis can be represented in terms of statistics, or a structured description of a study.

### 4.3 Mapping Ciphers to Keys

Given many ciphers and keys (being it original or generated), we developed a tool that maps a key/code/nomenclature and a ciphertext of the user's choice and creates a plaintext (i.e. decrypted text) on the basis of the provided key. The result is compared against 14 European historical language models, to see which language the plaintext matches best. The language models are provided by HistCorp, a collection of historical texts and language models for 14 European languages (Pettersson and Megyesi, 2018).

## 5 Searching in the Collection

The search function of the database allows for simple and advanced search depending on the user's need. Simple search, illustrated in Figure 5, allows for keyword searching, which looks for the occurrence of the search term in the record. The search is not sensitive to capitalization. The system matches the search term (i.e. the entered text) as a substring against all text fields with the exception of fields with checkbox and fields with numerical values. To search in these field types, the advanced search interface is used.

Advanced search allows the user to limit (or target) the search to each attribute or a combination of attributes defined by the metadata scheme with specific value(s). The Boolean operators AND, OR, and NOT are used in a graphical interface. The user first selects the attribute specified by the metadata. Most metadata fields use a text field for matching and the entered text is matched as a substring.

The screenshot shows a search interface titled "Simple search". Below the title, there is a note: "The entered text will be matched as a substring against all text fields. Checkbox and number fields are not matched. To match against these field types, use the advanced search interface." There is a "Switch to advanced search" link. A search input field contains the text "BAV" and a "Search" button. Below the search bar, it says "Found 101 records" and "20 per page" with pagination links "Pages: 1 2 3 4 ... 6". A table lists the search results with columns: Name, Current location (Country, City), and Holder. The results are all from the Vatican City State, with various "BAV\_Barb.lat\_6956-XX" identifiers, and the holder is "Bibliothec".

View	Name	Current location	Holder
		Country City	
	BAV_Barb.lat_6956-1	Vatican City State Vatican City	Bibliothec
	BAV_Barb.lat_6956-10	Vatican City State Vatican City	Bibliothec
	BAV_Barb.lat_6956-11	Vatican City State Vatican City	Bibliothec
	BAV_Barb.lat_6956-12	Vatican City State Vatican City	Bibliothec
	BAV_Barb.lat_6956-13	Vatican City State Vatican City	Bibliothec
	BAV_Barb.lat_6956-14	Vatican City State Vatican City	Bibliothec
	BAV_Barb.lat_6956-15	Vatican City State Vatican City	Bibliothec
	BAV_Barb.lat_6956-16	Vatican City State Vatican City	Bibliothec
	BAV_Barb.lat_6956-17	Vatican City State Vatican City	Bibliothec
	BAV_Barb.lat_6956-18	Vatican City State Vatican City	Bibliothec
	BAV_Barb.lat_6956-19	Vatican City State Vatican City	Bibliothec
	BAV_Barb.lat_6956-2	Vatican City State Vatican City	Bibliothec
	BAV_Barb.lat_6956-20	Vatican City State Vatican City	Bibliothec
	BAV_Barb.lat_6956-21	Vatican City State Vatican City	Bibliothec
	BAV_Barb.lat_6956-22	Vatican City State Vatican City	Bibliothec
	BAV_Barb.lat_6956-23	Vatican City State Vatican City	Bibliothec
	BAV_Barb.lat_6956-24	Vatican City State Vatican City	Bibliothec
	BAV_Barb.lat_6956-25	Vatican City State Vatican City	Bibliothec
	BAV_Barb.lat_6956-26	Vatican City State Vatican City	Bibliothec
	BAV_Barb.lat_6956-27	Vatican City State Vatican City	Bibliothec

Figure 5: Simple search with matched records.

The NOT checkbox can be set to negate the expression. To the right of the selected field the user can delete the expression.

Search functions can be built upon each other for sequential search. The order of the operating functions are made explicit by grouping the expressions; composite expressions are shown in the same grey box appearing vertically, while consecutive operations, visualized horizontally, are executed in sequence. Figure 6 illustrates the advanced search function where we searched for all keys originated from either Germany or France from all times except from the years between 1700 and 1800.

The result of the search is shown either as a list of matched items line by line with metadata information shown in the columns for each item (see Figures 5 and 6), or as a

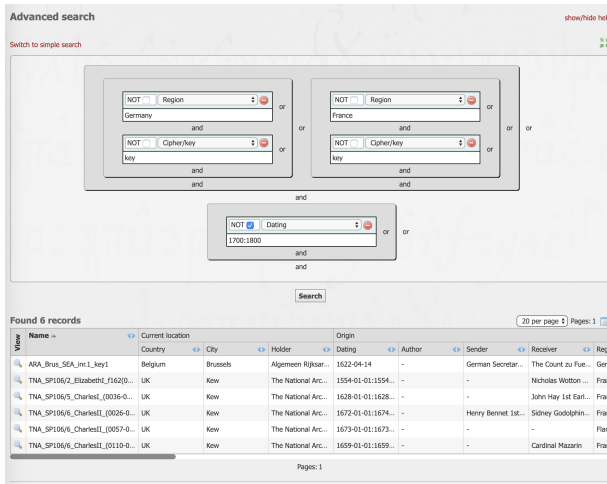


Figure 6: Advanced search.

"Google-style" result showing each record with its metadata listed item by item, see Figure 7. The magnifying glass shows all information about the particular record.

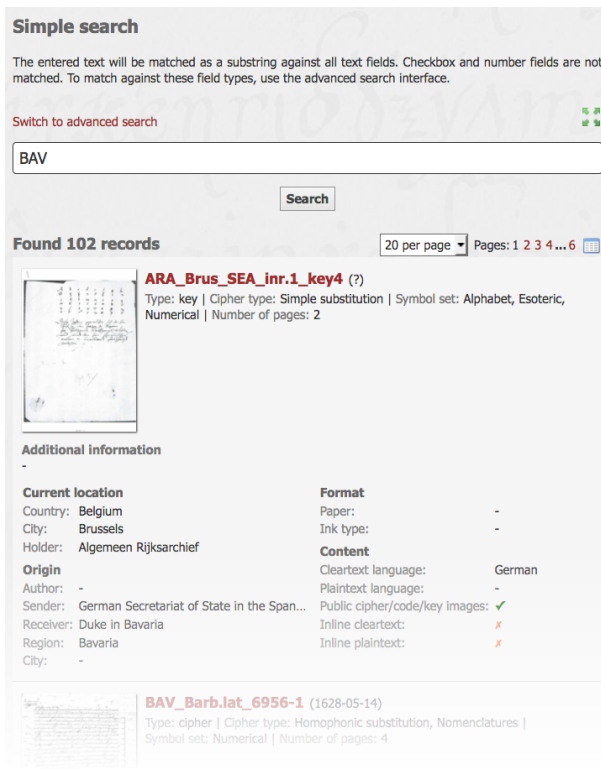


Figure 7: Search result.

Upon choosing a record from the list of matched documents, the record with all metadata is visualised, as illustrated in Figure 8 for the Copiale cipher with all metadata entered in the database.

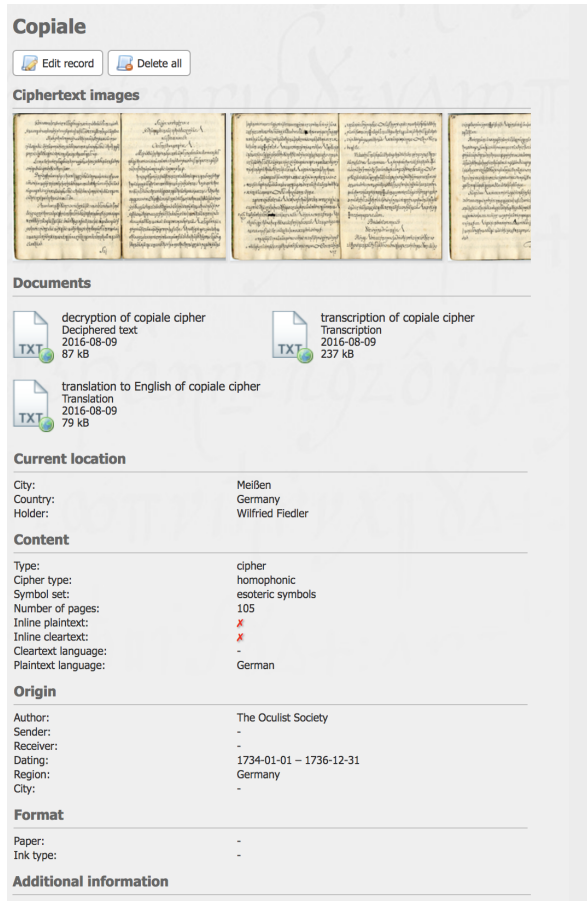


Figure 8: A cipher with metadata.

## 6 Adding New Data

One of the main goals of the DECODE database with its graphical user interaction is to allow registered users to create new records, ciphertexts or keys, by uploading an image of the encrypted document, and filling in metadata information about the manuscript. Mandatory fields are naming the manuscript, the current location (Country, City and Holder) of the manuscript, and the number of pages the manuscript consists of. All other metadata fields and related documents are optional.

For each uploaded document, being it an image, transcription, cryptanalysis, or publications, the user can choose to make it private, i.e. to not allow access to the file to other users.

## 7 Editing Existing Data

The user can edit information about existing records if she/he is the owner of the record. Uploaded documents, such as transcriptions, decrypted plaintexts, or translations of the

Figure 9: Entering new records.

plaintext can be deleted. New documents can be added and the metadata can be corrected. The owner can also choose to delete the whole record.

## 8 User Access and Roles

The DECODE database with a hosting web-service is publicly available with open access. In order to edit or add ciphers, codes or keys and related documents, registration is required of interested parties, typically specialists in historical cryptology. Personal data about the registered user includes information about the name of the user, affiliation, and scientific background (computational linguistics, computer science, cryptology, history, linguistics, literature, mathematics, politics, and other). The personal information that the user provides is handled according to GDPR. The personal information is stored and used to enable

users of the site to see who has uploaded and modified certain content, and to simplify collaboration between users. The user may unsubscribe as a registered user upon request at any time without any explanation.

Registered users of the site may upload text, pictures and other information ("content") which is stored by DECODE, and is shared with other registered users and other users of the site, except in circumstances where registered users choose to make their content inaccessible to other users (i.e. private); this functionality is noted where available.

The user can remove the content that she/he uploaded (i.e. the owner of the record) at any time. Upon registration, the registered user agrees to not upload any content to which he/she does not hold the necessary rights. We do not claim any ownership rights of the content uploaded by the users. We do not use the content commercially. However, we share the content with third parties in order to perform image analysis and other types of analyses, except for the content that the user has chosen to make inaccessible. Should the content be in violation of applicable copyright law or other laws on intellectual property rights or be in any way abusive or illegal, we reserve the right to remove such content without any prior warning.

Should a user find any content on the site which is or may be in violation of applicable copyright law or other laws on intellectual property rights or in any way abusive or illegal, we ask the user to report this to us via email. Such information may be deleted by us at any time without any prior warning. The user has to tick a box that she/he agrees to the Terms and Conditions which are provided on the site upon registration, as described above.

## 9 Technical Description

The DECODE database web application is written in Python and runs as a WSGI application using Flask on the Apache web server via `mod_wsgi`. Secure session handling and logins are handled via the Flask extension Flask-Login. The codebase has two layers: The core logic, for which a test suite has been written to make sure that the system always remains in a consistent state, and the web part, which

calls into the application logic on behalf of the user.

All data itself (user data, record metadata) is stored in a PostgreSQL database, and is interacted with using Psycopg (a PostgreSQL-Python adapter). Image files are stored directly in the filesystem, although any access to these is done by first consulting the database. Search queries are served directly by the database.

Because Python does not handle dates before year 0 (ISO 8601), and because documents dated before that might be introduced into the database, the `mxDateTime` library from eGenix is used in place of the `datetime` module that the Python standard library provides.

Creation of thumbnails is done using Pillow, a fork of the Python Imaging Library. The `'pdfimages'` utility from Poppler does the PDF image extraction, enabling users to directly upload PDF files of a scanned cipher instead of having to extract these themselves.

The server that the database web application is running on provides the SMTP connection, which is used to (via the Python standard library) send email to users, for example in order to reset one's password.

## 10 Tools

The database content is currently connected to CrypTool2 (CrypTool2, 2018) through an HTTP API, and connection to the Manulab system (Antal and Zajac, 2018) is underway.

To allow automatic processing of cipher images and transcription for decryption of the documents, we also develop historical language models extracted from authentic historical texts, and on-line tools for semi-automatic transcription, further develop cipher-key mapping, and the statistical analysis of ciphertexts. Currently, CrypTool2 is directly accessible for automatic decoding of ciphertexts where the user can test various decryption algorithms to decipher the encrypted elements.

## 11 Conclusion

We presented an on-line database aiming at the collection and systematic description of encrypted historical manuscripts. The database allows the user to search among ciphertexts and keys, and upload new encrypted histor-

ical sources with their metadata information and other relevant documents.

Future improvements include a mapping between ciphers and matching original and generated keys, and standardised forms for personal names, holding institutions and locations, like GND-number, VIAF or geonames-Number to be able to connect these in a systematic way for reliable search, as these entities are freely decided by the owner of the record today.

Most importantly, tools for automatic transcription, cryptanalysis and decryption are underway and will be connected to the database to allow self-help of the users.

Our hope is that many professionals interested in historical cryptology could make use of the data collected and enlarge and enrich the database with new ciphertexts, codebooks, and keys, or transcriptions, additional improved cryptanalysis, or in the best of worlds solution(s) to the undecrypted, still secret documents.

## Acknowledgements

We would like to thank our colleagues Nicklas Bergman, Bengt Dahlqvist, and Per Starbäck for their help throughout the project, and all users and contributors who are willing to share these fascinating historical sources. This work was supported by the Swedish Research Council, grants E0067801 and 2018-06074.

## References

- Eugen Antal and Pavol Zajac. 2018. ManuLab System Demonstration. In *Proceedings of the 1st International Conference on Historical Cryptology, HistoCrypt 2018*, pages 125–128.
- Archivio Segreto Vaticano ASV. 2016a. `Asv16:segr.di.stato/portogallo/1a/16v@2016` archivio segreto vaticano. All rights reserved.
- Archivio Segreto Vaticano ASV. 2016b. `Asv16:segr.di.stato/portogallo/8/93r@2016` archivio segreto vaticano. All rights reserved.
- CrypTool2. 2018. CrypTool2. last entry: 15/3/2019.
- Eva Pettersson and Beáta Megyesi. 2018. The HistCorp Collection of Historical Corpora and Resources. In *Proceedings of the Third Conference on Digital Humanities in the Nordic Countries*.

# Beyond Schlock on Screen: Teaching the History of Cryptology Through Media Representations of Secret Communications

**Peter Krapp**

Professor, Film & Media Studies / Informatics

Humanities Gateway 2321

University of California, Irvine

krapp@uci.edu

## Abstract

This paper lays out the course design for, and teaching experiences with, a class that introduces students in the humanities to the history of cryptology, with particular attention to film and media studies. The course covers principles of secret communication from ancient times to the 21<sup>st</sup> century, and encourages students to develop creative solutions that may help portray computing, computer networks, and cybersecurity issues in more informed and accurate ways on screen.

## 1 Introduction

While it may seem paradoxical to combine communication and secrecy, in fact media history can be told as the story of secret communications - from long before the earliest radio transmissions and interceptions to long after the commercial union of military technology and entertainment in television. In this course, students discover the media history of codes and ciphers from ancient cultures to the advent of computing, with a focus on secret communication mostly before the proto-computers of Bletchley Park, while connecting historical methods to contemporary questions of secrecy, privacy, and security in the Internet age. Readings include short stories and selected articles and chapters from the history of encryption and code breaking. Each week also features hands-on exercises (in class as well as in homework), and workshops on 21st century applications of historical models, with particular attention to the often fundamentally irreconcilable demands of privacy, security, trust, data integrity, and freedom of speech (Diffie/Landau 2007). This, however, is not a class in computer science or informatics, and it requires neither facility with number theory nor an appetite for algorithms.

The overall arc of the course leads students to ponder the motivations for secure and secret transmissions, from assurances of communication integrity to various forms of authentication, and beyond a naïve identification of concealment with security. Once they wrap their heads around the difference between steganography and cryptology and begin to appreciate the nuances of substitution and transposition ciphers, historical examples from the ancient Skytale to forms of the Pigpen cipher allow hands-on decoding experience. After an aside about steganography and invisible ink, it is time for methods of cracking monoalphabetic substitution (Kahn 1967, Macrakis 2014, Singh 1999). Most students at the University of California have sufficient prior exposure to American history to enjoy exploring secret communications from Independence to the Civil War, amplified by selected screenings from TV shows like *Turn* (2014-2017) about the Culper Ring - which also makes an appearance in season 4, episode 6 (2012-2013) of the TV show *White Collar* in contemporary New York City. Many also enjoy tracing the mechanization of ciphers from the Alberti disc to the Mexican Army disc and from the Jefferson wheel cipher to the US Army's M-94 cylinder used as late as 1942.

Media representations of secrecy and security in communications allow students to tackle the Vigenère table, try taking the Kasiski test, visualize implementations of ciphers from Polybius to ADFGVX, and wrap their heads around Friedman's index of coincidence (Kackman 2005, Bauer 2013). In order to introduce the enduring mystique of Number Stations, it helps to have them listen not just to a few tracks from the CONET Project (1997), but also from the Wilco album *Yankee Hotel Foxtrot* (2001) from a band many of them recognize; moreover, the thriller *Numbers Station* (2013) starring John Cusack and Malin Akerman can lead to a fruitful discussion of how faithful film and

television should be to real technology. One pivotal homework assignment therefore is a film review focusing on ciphers and codes as they are explored in cinema and television. The list of acceptable titles ranges from *Rendezvous* (1935) and *Cipher Bureau* (1938) to *The Imitation Game* (2014) and *Mr Robot* (2015-2019), but most students tend to pick neither the most recent nor the oldest examples, preferring to stick instead with espionage fare from the eighties or nineties – often several vie for writing up *Sneakers* (1992) or *Pi* (1998).

The course pivots on storytelling and visualization, but since it cannot require advanced mathematics or informatics knowledge, it focuses mainly on ciphers and codes before the advent of computing, without omitting Navajo code talkers, Enigma and Colossus, etc. Students quickly grasp why linguists developed frequency analysis as a way to crack simple substitution ciphers, and how nomenclators and book codes aided trade and diplomacy (Kahn 1967). Yet without requiring students in such a course to tackle prime factorization or hash functions, it is nonetheless important to discuss how much their internet use depends on cryptographic principles – once they share how much their daily life revolves around trust in online communications with banks and stores, doctors and pharmacies, educational institutions and entertainment, it is not difficult to illustrate how much the infrastructure of secure communications depends on asymmetric ciphers (Bauer 2007, Quisquater 1990).

Finally, the course ends with a survey of unsolved cryptological puzzles, touching on Voynich, Beale, Zodiac, and Kryptos (Schmeh 2015, Clemens 2016, Bauer 2017). At the conclusion of this writing-intensive seminar, students are expected to have gained greater facility with the relevant historical and critical vocabulary, deeper knowledge about media history, a keener appreciation for codes and ciphers, an ability to critically evaluate conflicting demands on communication, and a better understanding of both fictional and real secrecy (Glass 2013, Koblitz 2010, Koss 2014). In addition to familiarizing students with conceptual and historical content, this course involves advanced information literacy skills by locating, evaluating and integrating information gathered from multiple sources into discipline-specific writing.

## 2 Overcoming the schlock paradigm

What does encrypted communication look like? The problem with audiovisual representations of cybersecurity in particular and computer networks in general is that they are all too often turned into ludicrous caricatures on screen. Even computer-focused TV dramas like *CSI: Cyber* (2015) get numerous details so wrong that few computer-literate viewers can stand to watch. Computing is not about blinking lights and pixels - and it does not help to lard the script with misused and mispronounced jargon. Malicious code does not show up in red on your screen, and a forensic review takes more than a few minutes. Cybercrime is more likely to involve phishing for credit card numbers or trade secrets rather than kidnapping. When it comes to real or imaginary risks online, movies and TV shows dish up tired iterations that not only perpetuate stereotypes of hacking as (usually male) teenage flirtation with crime, but perhaps offend even more in how they depict data 'space' as an arcade game.

The countless TV shows and movies that get computing, encryption, and decryption wrong tend to make two types of mistakes: they want to glamorize the actions of a person in front of a computer, and they try to visualize the flow of data in networks, often in a ludicrous manner. Take the recent Michael Mann movie *Blackhat* (2015), which has Chris Hemsworth's character come out of prison to help combat international cybercrime. Setting aside the idea that Thor should be a computer nerd (and that he reads Baudrillard and Derrida in his prison cell instead of, say, Schneier or Kahn), while the plot features tricking a government employee into changing his password (so an outsider may gain access), its garish attempt to visualize data on a network is a major throwback to the bad old days of films like *The Net* (1995), *Hackers* (1995), *Sneakers* (1992), *War Games* (1983), or *Tron* (1982). Of course, *Hackers* (1995) is remembered mainly because it featured Angelina Jolie as one of two high schoolers involved in corporate extortion, but it also featured a virus that can speak and has a face, and its protagonists spend more time trash talking and partying than using computers. *The Net* (1995) has Sandra Bullock stumble around bulletin boards as if ordering pizza online was a radical act of subversion, and while this film does show some true aspects of the net (such as IP addresses), it does not do enough with them to ground it in computing reality - you could not



connect via telnet to an email address, for instance, you need an IP address and a TCP port number: only once you are connected can you initiate an email login. Nor would a Macintosh virus from 1995 infect a mainframe computer.

Interestingly, *Sneakers* (1992), an ensemble caper conceived during the making of *Wargames* (1983), features a black box capable of breaking any and all computer encryption, threatening to destabilize the world economy (which already raises the issue of post-quantum cryptography). Both movies tend to play fast and loose with computer technology - while you may be able to change your high school grades from a home computer if your school is sloppy, you most definitely won't be able to launch ICBMs with the same machine over the same dial-up modem. *Tron* (1982) is venerable for its pioneering use of computer graphics, but the idea that you can enter the network and act there as if it was a wireframe videogame has had a pernicious influence on film and television.

But perhaps one of the funniest transgressions against computing in a movie is *Swordfish* (2001), a Travolta extravaganza that sees Hugh Jackman's character forced to remotely access a computer of the Department of Defense (via an ancient PDP10 in a CalTech basement), which the actor does by typing really fast and clapping his hands while being threatened by thugs and at the same time sexually engaged by a young woman... and of course he pulls it off within 85 seconds, despite having suffered a long-term ban from using computers; and of course his old software, extant in some basement on tape, may look like vintage graphics software but it acts like a destructive worm... At least in *Superman III* (1983), Richard Pryor's character was allowed to concentrate on hacking a weather satellite, as unlikely as it is that you would do that in BASIC with some PRINT and LIST commands - let alone change payroll data, mess with traffic lights, and other exploits of Pryor's role as a recent computing acolyte.

Yet sadly, things have not become much more sophisticated over the years - consider, for instance, the recent TV show *Homeland* (2011-2019): who believes that the CIA server two Berlin-based online activists accidentally chance upon (in the fifth season, 2015) would grant them access to a directory full of files whose inordinately long names all contain the character string CIA? Do all your file listings contain your

employer's name? We are supposed to believe that pulling a physical cable out of the wall is the only thing experts in Langley can do to defend the CIA against an onslaught of internet connections seeking pornographic cam-shows? Shows like that tend to take computers less seriously than Indiana Jones is serious about archeology...

This is not just a question of verisimilitude or realism. While sci-fi author Arthur Clarke stipulated that "Any sufficiently advanced technology is indistinguishable from magic," this relies on a notion of widespread ignorance that is a legacy of pre-literate times and incompatible with the aims of education. Magic may be acceptable in fantasy fiction but not at university; we are interested in applicable concepts. So a student who wants to discuss Harry Potter in this class should consider two-factor authentication - the combination of something you have (e.g. a wand) or something you are (not a "muggle") and something you know (a passphrase) for common-room access at Hogwarts; not to mention parseltongue access to the Chamber of Secrets, or maybe the "blood password" needed to access the Horcrux Cave... How can films portray restricted access to The Leaky Cauldron, to Diagon Alley, and to Platform 9 3/4 at King's Cross? Who has access (and how) to the prefects' bathroom or to Dumbledore's office? These questions may seem fanciful, but the stakes are very real.

Before the recent TV show *Mr Robot* (2015-2019), television did not often show computer security issues in a realistic light. But *Mr Robot* is a show that pivots on the activities inside a cybersecurity firm, the code it displays on computer screens is real, and there are no hokey sound effects or flights of fancy. Even the hack on this show of an android phone, by inserting a chip that runs a bootloader, is a reference to realistic technology, in this instance the Flexispy software. Remarkably, *Mr Robot* does not shy away from discussing TOR routers, a distributed denial of service attack on corporate servers, and getting people to install malware - in this case, a remote access trojan that resembles an actual piece of software called DarkComet.

Ironically, the sci-fi conspiracy pastiche of *The Matrix Reloaded* (2003) is one of the few movies in the entire schlock genre to show a realistic scene: eschewing for once the usual antics of visualizing cyberspace as a vertiginous flight through the dim canyons of some badly rendered

Data-Manhattan, the movie shows Trinity (neither male nor a teen, but played by Carrie-Anne Moss), working at a keyboard instead of some futuristic interface contraption, using an actual piece of software to scan a power grid for weaknesses: NMAP, a port scanner on the command line, known to system administrators around the world. The German cyber-thriller *Who am I? No System is Safe* (2014) features a similar presentation of software exploits on the power grid – protagonist Benjamin seeks to remotely compromise a local utility using a script that seems to be endowed with universal powers in a command shell. In the thriller *The Bourne Ultimatum* (2007), the CIA hacks the mail server of a British newspaper, and the screen shows the realistic use of SSH, Postfix SMTP, and a domain name server in a UNIX shell. Another franchise thriller from the same year featured an interesting exploit within the first ten minutes: *Live Free or Die Hard* (2007) had its protagonist team up with a young hacker to fight a cyberterrorist. Meanwhile, the Swedish cinematic adaptation of the Stieg Larson book *The Girl with the Dragon Tattoo* (2009) shows her computer skills, again right in the first ten minutes, while the Hollywood remake (2011) does not. At least in the superhero flick *Fantastic Four* (2015, based on the Marvel Comics), you can see Sue Storm (played by Kate Mara) tracking down a companion online: her screen flashes "IPSCAN", "TRACEROUTE" and "PORTSCAN" - it is all too rare to see actual network technology represented.

Of course, getting computer technology right on screen is not just about software and hardware; cybersecurity is also about social engineering – the exploitation of patterns of behavior, vulnerabilities and opportunities. While hardware manufacturers clearly work closely with film and television producers to show off their wares, the software industry and the educational sector both miss out on opportunities to show computing as interesting, stimulating, and challenging - without faking it. Indeed, popular culture no longer celebrates hacking as the generally innocuous but occasionally very profitable pursuit of the computer hobbyist. Television stopped romanticizing the obsessions of talented nerds, the press no longer touts the bootstrapping spirit of digital capitalism. Instead, journalists are busy selling the sinister specter of hacking as an irreducible systemic threat of digital media. Never mind that until the late 1980s, a hacker was someone who, by trial and error and without

referring to any manuals, ended up successfully operating computers. Only a few years later, commentators already began to fret that malicious hacking might pose a serious and costly problem. For the longest time, digital culture had focused on access, learning, privacy, and free speech (Bamford 1982, Levy 2001, Schneier 2004). Yet in a sea change in popular opinion as well as legal and economic policy regarding network technology and education, alarmist commentators began to demonize anyone who tried to access more than the official, limited interface allowed.

### 3 Assignment Design

A cult of secrecy can easily lead to a global resurgence of irrational rumor, and unfortunately this is indeed what one sees in a lot of internet culture. When conspiracy theory takes the place of critical computer culture, our future is seriously impoverished. Arguably, teaching students in film and media studies about basic concepts of cybersecurity, and getting them interested in the outlines of the history of cryptology, may in time greatly increase the chances for scripts and scenes that provide more accurate and more intelligent audiovisual representations of computing and of communication security.

In order to heighten attention to both the problematic audiovisual representation of cybersecurity as well as to the possibilities of visualizing cryptology in persuasive and plausible ways, students are tasked with writing synoptic treatments for films or TV pilots based on assigned short stories. While a synopsis is different from a full treatment in industry parlance (a synopsis distills the narrative into a brief pitch, while a treatment gets into nuts and bolts of representing a story audio-visually), for the pedagogical purposes of this course, what is solicited is a document that highlights the necessary details with some cinematic style and rhythm, giving a feeling for characters, mood, and visual settings evocative of a time and place. Such a synoptic treatment is not simply a retelling of the story; it should be a document that might allow a decision-maker to evaluate the idea as well as its intended audiovisual execution. Marking story beats with particular attention to how to present issues of secret communication on screen, it needs to include a title and logline, introduce major characters, set the scene, dramatize the main conflicts leading to a crisis, and envision the dramatic resolution. Unlike a short story, the synoptic treatment cannot tell us a character's

thoughts but needs to show them, it cannot provide background but needs to outline dialogue; for pedagogical reasons (and because writing actual dialogue is hard), the task here is to succinctly describe the dialogue that a fully-fledged screenplay would provide. The short stories assigned were culled mostly from American and British detective fiction dating to the turn of the 20th century. In turning such old-fashioned material into a synoptic treatment, students are not simply retelling a piece of fiction, but updating and retooling its story beats to suit their own contemporary taste, with particular attention to how one can present secret communications on screen.

In addition, students compile reference materials resembling 'encyclopedia entries' about certain names and concepts that are important to the history of cryptology. Here the task is to conduct some research (a minimum of 4-5 references), define/describe/discuss what the keyword denotes or who the person is/was, and how exactly this entry relates to course topics. At least one reference source must be drawn from an online database for academic research (such as JSTOR or Project MUSE) in order to familiarize students with library systems for academic work. With names ranging from Alberti and Trithemius to Diffie and Schneier, and concepts including Kerckhoff's Principles, a Zero Knowledge Proof, Atbash, PGP, or the Clipper Chip, students sometimes struggle to fit all their findings into an encyclopedia entry that is concise yet comprehensive in scope.

An in-class midterm mixes multiple choice questions about historical facts with a few open-ended prompts that solicit a few paragraphs of reflection. How do you use a keyword to enhance the Caesar cipher? What is the second most common trigram in English? What kinds of sympathetic stains can you list? What was the name for the ancient Greek method to secure confidential messages? Which early US diplomat is associated with a wheel cipher? What is the first step to begin cracking a message if you know it was enciphered with the Vigenère method but you do not have the key? What led Babbage to a statistical breakthrough in cryptanalysis, and why did he not publish it, but Kasiski later did? A final take-home essay on a research topic directly related to course materials is expected to be more substantial, and again at least one source must be drawn from an online database for academic

research (such as JSTOR or Project MUSE), and the individual research topics and essay drafts are workshopped in class. Should governments be able to access anyone's encrypted communication to prevent crimes, or should technology companies deploy encryption as unbreakable as possible to protect widespread privacy and security? What are contemporary forms of steganography, and how practical does it seem to store and/or transmit secret information in superficially unaltered sound files, images, videos, etc.? As with all assigned writing, both in class and at home, students provide peer review on multiple drafts through a shared course portal, so that graded assignments are never first drafts with all their usual flaws.

#### 4 Feedback

Students quickly find out for themselves why the depiction of secret communications on screen is often so stilted and wrong-headed, but a few found rather creative solutions that their peers justifiably celebrated in peer-review sessions. Unlike the analytic and critical mode university students in the humanities are commonly expected to exercise, many of these assignments are not explanations or comments on what they read. If your writing must prepare for telling a story with audiovisual means, you are neither spelling out a character's thoughts, nor providing biographical or technical background. Granted, one must not expect truly creative writing – students know that they should neither copy nor invent dialogue or characters for their synoptic treatments. Once they see that they can remain faithful to the conceptual dimensions of each short story and yet put considerable inventiveness into the pitch for a screenplay based on it, they deliver with impressive ingenuity.

On the other hand, even if they enjoy puzzling out how to write their own names in Pigpen, or how best to define the differences between substitution and transposition ciphers, one should not expect humanities students to install JCrypt on their computers, or to study the math involved in public key crypto (Koblitz 1997, Kaur 2008, Winkel 2008, Kurt 2010). But one certainly can expect them to work through a curriculum that surveys the history of secret communication, albeit mostly pre-computing, and draw conclusions for their own lives in the 21st century. Even or especially if they are not budding computer scientists, they need to be able to debate the role of cryptography during World War II,

reading about Enigma and Purple, about Alan Turing and the Polish mathematicians who made seminal discoveries. Students invariably show themselves engaged with current cybersecurity, from password management to the trust they put in various communication platforms, whether WhatsApp or Messenger or Telegram or Signal. They tend to be enthusiastic about social media, and generally much less skeptical about data mining and advertising than national surveys suggest they might be (Pew Research Center 2018). They not only get a kick out of the Zimmerman Telegram, but even more so out of imagining its contemporary equivalent as a *casus belli*. They tend to be rather skeptical about some of David Kahn's claims about why the Germans lost to US intelligence in World War II; they neither vilify nor venerate characters like Julian Assange or Edward Snowden, and their discussions and homework show a clear preference for discussing the security of gaming servers over that of credit card companies, or whether to trust social media platforms rather than political institutions.

Students tend to have animated and well-informed discussions of the stakes of online identity, anonymity, and pseudonyms. They never acquiesce to a majority viewpoint, though, and some maintain reservations throughout the entire course about certain conflicts between privacy and public security, trust and advanced technology, freedom of speech and personal accountability online.

Writing synoptic treatments that seek to update classic detective fiction like the Sherlock Holmes yarn about the Dancing Men or an Isaac Asimov short story about encryption, students show ingenuity in visualizing a code that might be like graffiti, hidden in plain sight. Perhaps not surprisingly given that they are less likely to own printed books and more likely to read digital files on their various devices, fewer students become interested in book codes, as featured in season two of the TV show *Burn Notice* (a stolen bible dominates the entire season's plot, 2008-2009), in the movie *National Treasure* (where coordinates on the back of the Declaration of Independence lead to elusive treasure, 2004), or in the Sherlock Holmes mystery *The Valley of Fear* (1915), despite the enduring popularity of Holmes as a character on TV and on the big screen. Also perhaps unsurprisingly, students tend to be less interested in codes that involve more than one

language, even though they appreciate that the history of cryptology for a long time was entwined with translation and philology, and not only in prominent ways like the Rosetta Stone or in decoding German and Japanese World War II communications.

By the same token, students tend to show a refreshing lack of respect for old-fashioned aspects of the short stories they were tasked to update; instead of snuff boxes, poisoned pens, and handkerchiefs, their versions of these stories feature smartphones, dance clubs and graffiti, and rather than see their characters scandalized by allegations of infidelity or fiscal impropriety, their envisioned plots twists include social media gaffes and wet t-shirt contests. In the end, the aims of the course are demonstrably achieved, as evident in the students' confident command of historical and conceptual dimensions of cryptology in their final essays. There are usually quite a few essays on questions of identity theft and how to protect oneself against it. There are usually competent arguments for, as well as against, government access to encryption backdoors; some ambitious and more computer-literate students are also game to puzzle over unsolved ciphers (Schmeh 2015, Bauer 2017), or to tackle the task of demonstrating how Auguste Kerckhoffs' principles (from 1883) are still valid in today's mobile media culture.

Students also enjoy a brief weekly exercise that introduces various technical implementations of the issues studied in the class, from the proper set up of browsers and virtual private networks to comparing password managers, from multi-factor authentication to safe use of social media. Now that the university they enrolled with is moving to systems that require multi-factor authentication, they see that passwords alone are no longer sufficient in preventing unauthorized access of individual and institutional resources; and most students turn out to be rather passionate about protecting their personally identifying information. Since passwords, even in the academic environment, are now routinely compromised through malware, brute force attacks, phishing, and other exploits, the history and future of secure communication raises new questions that are directly relevant to their everyday lives. Part of the impact of this course is to lead students to discover basic principles for themselves, instead of nudging their behavior as institutions tend to try.

Integrating online resources into the course has also greatly aided the contextualization. An Instagram account is a more readily accessible repository of images from the history of cryptography than slides used to be. Students can see on the calendar of the National Cryptologic Museum Foundation what happened on this day in cryptologic history, they can use morse coders and decoders, play with Pigpen fonts and anagram servers, and cast a far more informed look at the cyber security training resources of the university. Several times, I teamed up with IT specialists on campus responsible for propagating safe online behavior among students, and whether it was screenings and discussions or more focused panel presentations, the students who had taken this class did far better on the Cybersecurity training modules of the university than the general public does according to relevant surveys (Pew Research Center Cybersecurity Knowledge Quiz, 2017).

### Acknowledgments

I thank the students in my writing seminars on the media history of secret communications at the University of California, Irvine in 2005, 2011, 2017, and 2019.

### References

- James Bamford. 1982. *The Puzzle Palace: Inside the National Security Agency*. Penguin, New York.
- Craig P Bauer. 2013. *Secret History: The Story of Cryptology*. CRC Taylor & Francis, Boca Raton.
- Craig P Bauer. 2017. *Unsolved! The History and Mystery of the World's Greatest Ciphers from Ancient Egypt to Online Secret Societies*. Princeton University Press, Princeton NJ.
- Friedrich L Bauer. 2007. *Decrypted Secrets: Methods and Maxims of Cryptology*. Springer, Berlin.
- Raymond Clemens. 2016. *The Voynich Manuscript*. Yale University Press, Yale CT.
- Whitfield Diffie and Susan Landau, 2007. *Privacy on the Line: The Politics of Wiretapping and Encryption*, MIT Press, Cambridge MA.
- Darren Glass. 2013. "A First-Year Seminar on Cryptography", *Cryptologia*, 37:4, 305-310
- David Kahn. 1967. *The Codebreakers – The Story of Secret Writing*. Macmillan, New York.
- Michael Kackman. 2005. *Citizen Spy: Television, Espionage, and Cold War Culture*. University of Minnesota Press, Minneapolis.
- Manmohan Kaur. 2008. "Cryptography as a Pedagogical Tool", PRIMUS v18 n2 (March 2008), 198-206
- Neal Koblitz. 1997. "Cryptography as a Teaching Tool," *Cryptologia* 21:317–326.
- Neal Koblitz, 2010. "Secret Codes and Online Security: A Seminar for Entering Students," *Cryptologia*, 34:145–154
- Lorelei Koss. 2014. "Writing and Information Literacy in a Cryptology First-Year Seminar," *Cryptologia* 38:223-231
- Yesem Kurt. 2010. Deciphering an Undergraduate Cryptology Course, *Cryptologia*, 34:2, 155-162
- Stephen Levy. 2001. *Crypto*. Penguin Books, London
- Kirstie Macrakis. 2014. *Prisoners, Lovers, and Spies: The Story of Invisible Ink from Herodotus to al-Qaeda*. Yale University Press, Yale CT.
- National Cryptologic Museum Foundation. <https://cryptologicfoundation.org>
- Pew Research Center: Americans and Cybersecurity <https://www.pewinternet.org/2017/01/26/americans-and-cybersecurity/>
- Pew Research Center Cybersecurity Quiz, 2017 <https://www.pewinternet.org/quiz/cybersecurity-knowledge/>
- Klaus Schmeh. 2015. List of Encrypted Books. <http://scienceblogs.de/klausiis-kryptokolumne/klaus-schmehs-list-of-encrypted-books/>
- Simon Singh. 1999. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. Anchor Books, New York.
- Bruce Schneier. 2004. *Secrets and Lies. Digital Security in a Networked World*. Wiley, Indianapolis.
- Jean-Jacques Quisquater et al. 1990. "How to Explain Zero-Knowledge Protocols to your Children", in G Brassard ed., *Advances in Cryptology – Crypto 89*. LNCS 435, pp 628-631.
- Brian Winkel. 2008. Lessons Learned from a Mathematical Cryptology Course, *Cryptologia*, 32:1, 45-55



# Solving a 40-Letter Playfair Challenge with CrypTool 2

George Lasry

The CrypTool Team

george.lasry@cryptool.org

## Abstract

Playfair is a manual substitution cipher invented in 1854 by Charles Wheatstone. Its name and popularity came from the endorsement of his friend Lord Playfair. The Playfair cipher encrypts bigrams (pairs of letters), and is considered more secure than monoalphabetic substitution ciphers which encrypt single letters. It was used by several countries in the 19th century and in the first half of the 20th century.

Playfair ciphers can often be solved with the help of a crib. Ciphertext-only attacks usually require hundreds of letters when carried out manually (Mauborgne, 1918). More recently, computerized attacks based on hill climbing and simulated annealing have been published, that require between 60 to 100 letters of ciphertext (Cowan, 2008; Al-Kazaz et al., 2018).

In this article, the author presents a novel ciphertext-only attack, implemented in the open-source e-learning CrypTool 2 (CT2) platform, that is effective against ciphertexts as short as 40 letters (CrypTool 2 Team, 2019). This attack is based on a specialized adaptation of simulated annealing and uses hexagrams in the scoring method. With CT2, a Playfair public challenge with only 40 letters was solved, establishing an unofficial world record for decrypting short Playfair messages, encrypted with random keys, from ciphertext only (Schmeh, 2018b). The author also offers a series of new Playfair challenges.<sup>1</sup>

<sup>1</sup>This work has been supported by the Swedish Research Council, grant 2018-06074, DECRYPT - Decryption of historical manuscripts.

## 1 Description of Playfair

Playfair enciphering and deciphering are based on a key square, with a  $5 \cdot 5$  grid of letters. Each of the 25 letters must be unique and one letter of the alphabet (usually J) is omitted from the square, as there are only 25 positions in the square, but 26 letters in the alphabet.

While it is possible to use a random key square, it is often more convenient to derive a key square from a keyword (or sentence). The keyword is written horizontally with duplicate letters being removed. The rest of the square is filled with the remaining letters of the alphabet, in alphabetical order.

For example, the key square derived from the keyphrase HELLO WORLD is:

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

To encrypt a message, the plaintext is split into bigrams. If there is an odd number of letters, a Z or X is added as the last letter. To encrypt the message HIDE THE GOLD, we first split it into bigrams, and add Z at the end:

HI DE TH EG OL DZ

Next, for each pair, we locate its two letters in the square. We replace them according to the following rules:

- If the two letters are corners of a rectangle, take the letters on the horizontal opposite corners of the rectangle. For example, HI is encrypted as LF.
- If both letters are in the same column, select the letters below each one in the square (go-

ing back to the top if at the bottom). For example, DE is encrypted as GD.

- If both letters are in the same row, select the letters to the right of each one (going back to the left if at the farthest right). For example, OL is encrypted as WO.

Using these rules, we obtain:

LF GD NW DP WO CV

## 2 Cryptanalysis of Playfair

In this section, the security of Playfair is discussed, and a survey of prior attacks on Playfair is presented.

### 2.1 Security of Playfair

The Playfair cipher has several weaknesses, which may be exploited when trying to recover a Playfair key square:

- With long enough ciphertexts, statistical analysis can be applied on bigrams, and matched against the frequencies of common bigrams in the language (e.g., English). The ciphertext bigrams corresponding to the most common bigrams in the language (such as TH or IN in English) can often be easily identified.
- The most frequent ciphertext letters are likely to be near the most frequent plaintext letters (e.g., E, T, I, O, N) in the key square.
- Each mapping of a plaintext bigram to a ciphertext bigram reveals the mapping of another bigram, where the letters of the bigrams have been reversed. For example, if HI is encoded as LF, then IH will necessarily be encoded as FL.
- If the Playfair key is derived from a keyword, then the last row often contains the last alphabet letters such as X, Y and Z. Also, the letters which did not appear in the keyword and are used to fill the bottom part of the square will always appear in alphabetical order.

### 2.2 Prior Cryptanalysis of Playfair

Historically, Playfair was often solved by hand with the help of cribs (partially-known plaintext attack). Based on the crib, some entries of the key square can be guessed or reproduced, and additional entries reconstructed by trial and error.

Manual ciphertext-only cryptanalysis involves frequency analysis of ciphertext bigrams, and usually requires hundreds of ciphertext letters, not an uncommon scenario if multiple messages were encoded using the same key. In (Mauborgne, 1918), a manual method is described, to solve a ciphertext composed of 800 letters. The frequencies of the most common ciphertext bigrams are matched against those most common in English, e.g. TH, ER, and ET. A tentative initial square is built, and completed in a trial-and-error process.

In (Monge, 1936), a challenge ciphertext with only 30 letters is solved by taking advantage of the characteristics of a key square built from a keyword (see Section 2.1). This is considered to be the shortest Playfair ciphertext ever solved, that was encrypted using a key derived from a keyword.<sup>2</sup>

(Cowan, 2008) presents an attack based on simulated annealing. It uses quadgrams frequencies (applied on a logarithmic scale) as the scoring function. A constant temperature is employed (Hoos and Stützle, 2004, p. 76). With this method, ciphertexts as short as 80 letters can be solved. Also, in the now-defunct website [www.cryptoden.com](http://www.cryptoden.com)<sup>3</sup>, (Cowan, 2015) proposes a *churn* algorithm, described in Section 3.2. The *churn* algorithm was designed to mimic the process of simulated annealing with constant temperature, while reducing software code complexity and runtime. Cowan describes how his *churn* method was found superior to hill climbing for attacks on various ciphers (Cowan, 2015). Cowan's implementation of *churn* also produces an interesting but probably unintended side-effect, described in Section 3.3.

In (Al-Kazaz et al., 2018), a compression-based technique combined with simulated annealing is described, and demonstrated on several ciphertexts. The shortest one, with only 60 letters, was successfully decrypted with only two errors. The com-

<sup>2</sup>The unicity distance for Playfair and English is 22.69 letters (Deavours, 1977). For any Playfair cryptogram of that length or shorter, it is likely that there exist one or more keys, different from the original key, which decrypt the cryptogram so that the resulting decryption is a plausible English text (and different from the original plaintext). The unicity distance can be viewed as a theoretical lower-bound for the length of a cryptogram, so that its key may be recovered via cryptanalysis. The length of the cryptogram solved by Monge (30 letters) is very close to that limit. On the other hand, the unicity distance is only 16.56 letters if it can be assumed that the last row in the key square is VWXYZ (as for most keys derived from keywords) (Deavours, 1977).

<sup>3</sup>[www.cryptoden.com](http://www.cryptoden.com) is still accessible via [www.wayback.com](http://www.wayback.com) (Cowan, 2015).



pression technique proposed in (Al-Kazaz et al., 2018) is essentially analog to using hexagram statistics (on a logarithmic scale) as the scoring method.

### 3 A New Ciphertext-only Attack

In this section, a novel attack, successfully employed to solve several public challenges, is presented.

This new attack extends Cowan’s method (Cowan, 2008). While it is also based on constant-temperature simulated annealing, it uses hexagrams statistics, instead of quadgrams, converted to a logarithmic scale. It implements an extended set of transformations applied to candidate keys, adding new types of transformations, as described in Section 3.4. Furthermore, the new attack exhaustively applies and tests the full set of transformations on candidate keys, at each step of simulated annealing (instead of applying only a random subset of transformations as in (Cowan, 2008)).

#### 3.1 An Initial Implementation

Initially, this new attack was implemented using a standard constant-temperature simulated annealing algorithm. As described in Listing 1, higher scores are always accepted. If the new score is lower than the current score, the probability of acceptance  $p$  is computed using the Metropolis formula (Hoos and Stützle, 2004, p. 75), based on the score degradation  $d$  and the constant temperature  $t$ .

$$p = e^{-d/t} \quad (1)$$

The first implementation of this new attack, after tuning and optimizing the temperature  $t$ , was able to solve ciphertexts with only 70 letters, and rarely, with 60 letters (for comparison, (Cowan, 2008) requires between 80 to 100 letters). Also, hexagrams were found to be more effective than quadgrams or pentagrams as the scoring method (all using a logarithmic scale).

#### 3.2 Improved Implementation Using Churn

The attack was modified to use Cowan’s *churn* implementation of constant-temperature simulated annealing (Cowan, 2015). The *churn* acceptance function is described in Listing 2.<sup>4</sup> Cowan does not explain why he employs the term *churn*, however,

<sup>4</sup>The code in Listing 2 is different from the original code given in (Cowan, 2015). It was adapted for clarity, but it preserves the original functionality.

the process could be described as candidate keys being accepted with a decreasing probability, or discarded (‘churned’) with a increasing probability, as the score of the current key increases over time during simulated annealing.<sup>5</sup>

A lookup table with degradation values,  $D$ , is precomputed. Cowan does not describe how he computed  $D$ , but his original values can be reproduced and closely approximated. From Equation 1, it follows that:

$$d = t \cdot \ln(1/p) \quad (2)$$

$D$  has 100 entries (with an index  $i$  from 0 to 99). For each  $i$ , the acceptance probability is computed as follows:

$$p_i = (i + 1)/100 \quad (3)$$

and therefore:

$$D_i = t \cdot \ln(100/(i + 1)) \quad (4)$$

The *churn* acceptance function selects a (random) degradation threshold from the lookup table by generating a random index  $i$  from 0 to 99. If the actual degradation  $d$  is lower than this threshold, then the new key is accepted.<sup>6</sup>

After modifying the new attack on Playfair to use the *churn* acceptance function, the attack was again tested, and surprisingly, not only its runtime could be reduced, but the attack’s performance was also improved. The algorithm was able to consistently solve ciphertexts with 50 letters. Cowan’s *churn* algorithm was originally designed to mimic a constant-temperature simulated annealing process. There was therefore no apparent reason for such an improvement. After further investigation, the root cause of this phenomena was found, as described in Section 3.3.

#### 3.3 An Unintended Side Effect

With a regular constant-temperature simulated annealing process (without *churn*), since the Metropolis acceptance function is a continuous function

<sup>5</sup>The acceptance probability decreases exponentially for candidate keys with a score lower than the score for the current key, as a function of the score degradation - see Equation 1. As the score of the current key increases over time (as better current keys are being selected), the degradation for a given candidate key increases, and its probability of acceptance decreases.

<sup>6</sup>The same functionality could in principle be achieved without a lookup table. However, the implementation using a lookup table plays an important role, described in Section 3.3.

(see Equation 1), a candidate key with a score significantly lower than the score of the current key can theoretically be accepted, albeit with a low but non-zero probability  $p$ . In other words, a candidate key resulting in a very high degradation  $d$  might still be accepted.

With *churn*, the lookup table stores 100 discrete degradation values, and from Equation 4, it can be seen that the highest degradation value is:

$$D_0 = t \cdot \ln(100/(0 + 1)) = t \cdot \ln(100) \quad (5)$$

As a result, with *churn*, no key resulting in degradation  $d$  greater than  $D_0$  may ever be accepted. Similarly, it can be seen that there is also a lower bound for the acceptance probability,  $p_{min}$ , so that:

$$p_{min} = (0 + 1)/100 = 0.01 \quad (6)$$

Therefore, with *churn*, keys with a score degradation resulting in an acceptance probability  $p < p_{min} = 0.01$  will always be rejected, unlike with regular simulated annealing, where there is a low but non-zero probability they might be accepted. It was suspected that particular side-effect of the implementation of *churn* could be the root cause for the higher performance of the attack with *churn* compared to the attack with regular constant-temperature simulated annealing.

To validate this hypothesis, a third version of the attack was implemented, using the standard acceptance function (for constant-temperature simulated annealing – see Listing 1), but this time only accepting keys with acceptance probabilities  $p \geq 0.01$ . With this modification (described in Listing 3), the attack on Playfair achieved the same performance as when using *churn*, confirming the hypothesis. The  $p_{min}$  parameter was further fine-tuned and set to an optimal value of 0.0085.<sup>7</sup>

### 3.4 Transformations on Candidate Keys

In all versions of the new attack on Playfair, the set of transformations applied at each stage of simulated annealing includes:

- Swaps of any two elements in the square
- Swaps of any two rows in the square
- Swaps of any two columns in the square
- Permutations of the five rows
- Permutations of the five columns
- Permutations of the five elements of any row
- Permutations of the five elements of any column

All possible transformations listed here are tested at each step of simulated annealing. In contrast, in (Cowan, 2008), only randomly selected transformations are applied and tested (from a smaller set of transformation types, which only includes the swaps, as well as a few special transformations).

## 4 A New Partly-Known Plaintext Attack

The algorithm described in Section 3 was also adapted to support a crib-based attack. The scoring function was modified, so that the score (computed using hexagrams statistics) is increased for each known-plaintext symbol correctly reproduced, when decrypting the ciphertext with a candidate key. With this modification, ciphertexts with 40 letters can easily be solved given a crib of 10 letters.

## 5 Solving Playfair Challenges with CrypTool 2 (CT2)

The new attacks described in Section 3 were first implemented as command-line programs. A first ciphertext-only challenge with 50 letters published by Klaus Schmech was solved (Schmech, 2018c). It took a few seconds on a 10-core Intel Core i7 6950X 3.0 GHz PC to complete the attack and to recover the key and the plaintext.

The attack was also integrated into CT2, taking advantage of the convenient user interface of CT2, which shows useful details about the progress of the attack, such as a list of top keys (CrypTool 2 Team, 2019). Klaus Schmech published a second challenge, this time with only 40 letters, stating that its solution would constitute a world record for solving the shortest Playfair ciphertext encrypted with a random key (Schmech, 2018b).

This new challenge was attacked with CT2. Initial runs only produced spurious solutions. At some stage, CT2 displayed a decryption (in the 4th place

<sup>7</sup>In preliminary experiments with attacks on other ciphers, this seemingly minor adaptation of simulated annealing significantly improved their performance. One possible explanation is that accepting candidate keys with scores significantly lower than the score of the current key, might completely disrupt the convergence of simulated annealing towards the correct key. Whereas accepting keys with score slightly or moderately lower than for the current keys helps in surveying more diverse areas of the keyspace.

in the list), starting with MEETYOU, but only for a few seconds, before the decryption quickly disappeared from the list as new higher-score decryptions were inserted. The partial known-plaintext attack (see Section 4) was then run with MEETYOU as a crib, and the solution was quickly found (see Figure 1). The plaintext, after adding spaces, is as follows:

MEET YOU TOMORROW AT FOUR TWENTY  
AT MARKET PLACE

## 6 CrypTool 2

CrypTool 2 (CT2) is an open-source e-learning tool that helps pupils, students, and crypto enthusiasts to learn cryptology.<sup>8</sup> CT2 is part of the CrypTool project which includes widely-used e-learning tools for cryptography and cryptanalysis.<sup>9</sup> CT2 is the successor of CrypTool 1 (CT1), and it is based on a graphical programming language allowing the user to cascade different ciphers and methods and to follow the encryption or decryption steps in real-time (CrypTool 2 Team, 2019).

CT2 is maintained by the CrypTool team. Contributions and voluntary support to this open-source project come from all over the world. CT2 implements classical and modern cryptographic methods, including cryptanalytic methods. It is also used to implement real-world prototypes of distributed cryptanalysis using the so-called CrypCloud. CT2 is maintained in English and German.

State-of-the-art algorithms, such as the attack against the double transposition cipher described in (Lasry et al., 2014) and shown in Figure 2, are also integrated in CT2.

## 7 New Challenges

A series of new Playfair challenges is presented in Table 1 (Appendix 9), with short ciphertexts. The plaintexts were extracted from English books. The keys were generated either from an English keyphrase, or randomly. For some, the first eight letters of the plaintext are given as a crib. This crib is always PLAYFAIR, but the continuation of the plaintext is a sentence unrelated to Playfair.

In addition, (Schmeh, 2018a) has published a new challenge with 30 letters only and encrypted using a random key-square.

## References

- Noor R Al-Kazaz, Sean A Irvine, and William J Teahan. 2018. An Automatic Cryptanalysis of Playfair Ciphers Using Compression. In *Proceedings of the 1st International Conference on Historical Cryptology HistoCrypt 2018*, number 149, pages 115–124. Linköping University Electronic Press.
- Michael J Cowan. 2008. Breaking short Playfair ciphers with the simulated annealing algorithm. *Cryptologia*, 32(1):71–83.
- Michael J Cowan. 2015. Churn Algorithm. <https://web.archive.org/web/20150308125149/http://www.cryptoden.com:80/index.php/algorithms/churn-algorithm>, [Accessed: January, 18th, 2019].
- CrypTool 2 Team. 2019. CrypTool Portal – Cryptography for Everybody. <http://www.cryptool.org/>, [Accessed: January, 18th, 2019].
- C.A. Deavours. 1977. Unicity Points in Cryptanalysis. *Cryptologia*, 1(1):46–68.
- Holger H. Hoos and Thomas Stützle. 2004. *Stochastic local search: Foundations and applications*. Elsevier.
- George Lasry, Nils Kopal, and Arno Wacker. 2014. Solving the Double Transposition Challenge with a Divide-and-Conquer Approach. *Cryptologia*, 38(3):197–214.
- Joseph Oswald Mauborgne. 1918. *An advanced problem in cryptography and its solution*. Army Service Schools Press.
- Alf Monge. 1936. *Solution of a Playfair cipher*. US Signal Corps.
- Klaus Schmeh. 2018a. Playfair cipher: Is it unbreakable, if the message has only 30 letters? <http://scienceblogs.de/klausis-krypto-kolumne/2019/04/15/>, [Accessed: May, 18th, 2019].
- Klaus Schmeh. 2018b. Playfair cipher: Is it unbreakable, if the message has only 40 letters? <http://scienceblogs.de/klausis-krypto-kolumne/2018/12/08/>, [Accessed: January, 18th, 2019].
- Klaus Schmeh. 2018c. Playfair cipher: Is it unbreakable, if the message has only 50 letters? <http://scienceblogs.de/klausis-krypto-kolumne/2018/04/07/>, [Accessed: January, 18th, 2019].

<sup>8</sup><https://www.cryptool.org/en/cryptool2>

<sup>9</sup><https://en.wikipedia.org/wiki/cryptool>

## 8 Appendix – Listings and Figures

Listing 1: Simulated Annealing Acceptance Function – Constant Temperature

```
package common;
import java.util.Random;
public class FixedTemperatureSimulatedAnnealing {
    private Random random = new Random();

    // Fixed temperature optimized for hexagram scoring
    private static final double FIXED_TEMPERATURE = 20_000.0;

    /**
     * Simulated annealing acceptance function.
     *
     * @param newKeyScore - score for the new key
     * @param currentKeyScore - score for the current key
     * @return true if new key should be accepted
     */
    boolean accept(double newKeyScore, double currentKeyScore) {
        // Always accept better keys
        if (newKeyScore > currentKeyScore) {
            return true;
        }

        // Degradation between current key and new key.
        double degradation = currentKeyScore - newKeyScore;

        double acceptanceProbability =
            Math.pow(Math.E, - degradation / FIXED_TEMPERATURE);

        return random.nextDouble() < acceptanceProbability;
    }
}
```

Listing 2: Simulated Annealing Acceptance Function – With Churn Lookup Table

```

package common;
import java.util.Random;

public class ChurnSimulatedAnnealing {

    private Random random = new Random();

    // Fixed temperature optimized for hexagram scoring
    private static final double FIXED_TEMPERATURE = 20_000.0;

    // Size of degradation threshold lookup table.
    private static final int LOOKUP_TABLE_SIZE = 100;

    // The churn algorithm lookup table of degradation thresholds.
    private final double[] degradationLookupTable = new double[LOOKUP_TABLE_SIZE];

    // Compute the churn algorithm lookup table of degradation thresholds.
    void computeDegradationLookupTable() {
        for (int index = 0; index < LOOKUP_TABLE_SIZE; index++)
            degradationLookupTable[index] =
                FIXED_TEMPERATURE * Math.log(LOOKUP_TABLE_SIZE / (index + 1));
    }

    /**
     * Simulated Annealing acceptance function - Churn implementation.
     *
     * @param newKeyScore - score for the new key
     * @param currentKeyScore - score for the current key
     * @return true if new key should be accepted.
     */
    boolean accept(double newKeyScore, double currentKeyScore) {

        // Always accept better keys
        if (newKeyScore > currentKeyScore) return true;

        // Fetch a random degradation threshold from the lookup table.
        int randomIndex = random.nextInt(LOOKUP_TABLE_SIZE);
        double degradationRandomThreshold = degradationLookupTable[randomIndex];

        // Degradation between current key and new key.
        double degradation = currentKeyScore - newKeyScore;

        return degradation < degradationRandomThreshold;
    }
}

```

Listing 3: Simulated Annealing Acceptance Function – Constant Temperature – Modified

```
package common;

import java.util.Random;

public class ImprovedFixedTemperatureSimulatedAnnealing {

    private Random random = new Random();

    // Fixed temperature optimized for hexagram scoring
    private static final double FIXED_TEMPERATURE = 20_000.0;

    /**
     * Simulated Annealing acceptance function.
     *
     * @param newKeyScore - score for the new key
     * @param currentKeyScore - score for the current key
     * @return true if new key should be accepted.
     */
    boolean accept(double newKeyScore, double currentKeyScore) {

        // Always accept better keys
        if (newKeyScore > currentKeyScore) {
            return true;
        }

        // Degradation between current key and new key.
        double degradation = currentKeyScore - newKeyScore;

        double acceptanceProbability =
            Math.pow(Math.E, - degradation / FIXED_TEMPERATURE);

        return acceptanceProbability > 0.0085
            && random.nextDouble() < acceptanceProbability;
    }
}
```

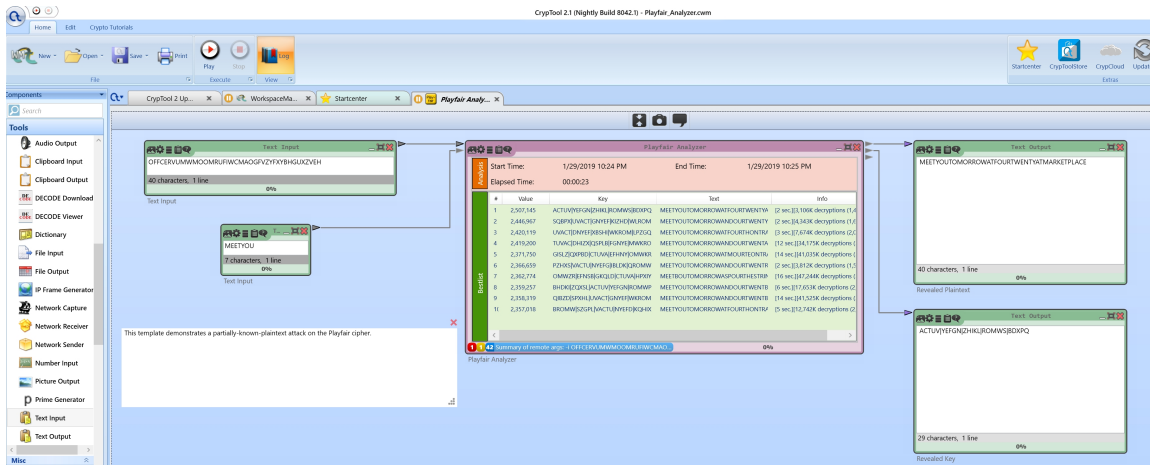


Figure 1: CT2 – Cryptanalysis of Playfair with Crib

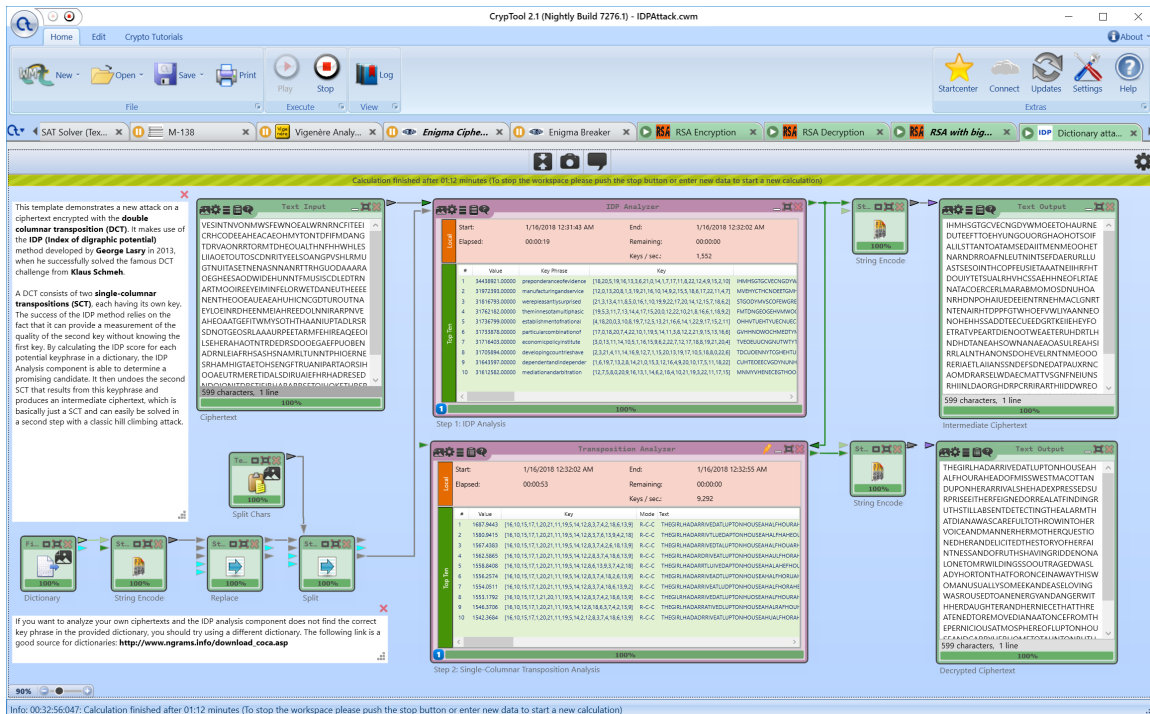


Figure 2: CT2 – Cryptanalysis of the Double Transposition Cipher

## 9 Appendix – Challenges

	<b>Ciphertext</b>	<b>Length</b>	<b>Key</b>	<b>Crib</b>
1	QONACDBLNKHIOTWDUEISOITFIDQBVOUTNRZOU CPC	40	From key phrase	PLAYFAIR
2	LVNXDNMHHHLHIUIEGENXTHEGQH XUHFQ	30	From key phrase	PLAYFAIR
3	HNGFDIRFAMAVHFOVXLGLTVOAZMYLQGRXHAHRNHGF	40	Random key	PLAYFAIR
4	ZAYNWPSYEMYQTIRXICMCKVHQHTHUKY	30	Random key	PLAYFAIR
5	IROAWMDQLRNCTUOCFHMQQKMAALCQMGHIQOQKLCAP	40	From key phrase	
6	BQUWLODQTOODLXWKEGAQOGHQQTQQZI	30	From key phrase	
7	ILPMPEOIIIZIRTPPRQRUYFUVXLIRCVANBVTWRCE CRVSLIQOVS	50	Random key	
8	TVCIYVGFVOGWPEFPDASNIXWKDISDRQVQLGSDZQXB	40	Random key	
9	PBILKMXFPDMDHCYHIVECOOUTGBNUC	30	Random key	
10	PROMGDUGVBNYXKEADCHTHM	22	Random key	

Table 1: New Playfair Challenges



# Cryptanalysis of an Early 20th Century Encrypted Journal

**Tony Gaffney**

Private Scholar

tony\_baloney\_macaroni@yahoo.co.uk

**Klaus Schmeh**

Freelanced Journalist

klaus@schmeh.org

## Abstract

Ernest Rinzi (1836-1909), a London-based jeweler, goldsmith and miniaturist, left behind an encrypted journal, which went unsolved until 2017. Dozens of artistically designed illustrations, which interact with text written in tiny letters, make this 175 pages journal one of the most outstanding cryptograms in existence. In 2017 one of the two authors of this paper brought Rinzi's journal to the attention of a wider audience, which led to the other author examining and breaking the encryption. The cipher Rinzi used proved a monoalphabetic substitution cipher (MASC) that replaces letters and numbers. The cryptanalysis work was complicated by the unusual and hard-to-read miniature writing.

## 1 Introduction

Ernest Rinzi (1836-1909), born as Ernesto Rinzi in Milan, Italy, was a jeweler, goldsmith and miniaturist of Italian decent. He was brought to London by the renowned jeweller Alessandro Castellani around 1859 and became a naturalized British citizen in 1867. Rinzi's oeuvre mainly consists of miniature paintings (mainly portraits) that were used as necklace pendants. He had a wealthy clientele and was very prolific. His works were exhibited at the Royal Academy, the Modern Gallery, the Royal Colonial Institute and other art galleries. Rinzi was a member of the Society of Miniature Painters.

Today, Rinzi's works can be found in art databases like MutualArt, Blouin, or Artnet. His creations are frequently traded at auction houses like Woolley & Wallis, Lofty's, and Bonham's. Nevertheless, there seems to be as good as no literature about Ernest Rinzi. The only owner of a

Rinzi miniature we have found is The Royal Collection Trust (Royal Collection Trust, 2019). We are not aware of a museum or gallery that currently exhibits a Rinzi work.

## 2 The Encrypted Journal

The Rare Book & Manuscript Library at the University of Illinois at Urbana-Champaign owns a 175 pages hand-written journal Rinzi left behind (Rinzi, 1903). Rinzi created it from 1898 to 1903. He started in the age of 62, which means that this journal belongs to Rinzi's late work (he died in the age of 73). As is easy to see, Ernest Rinzi's manuscript is encrypted. Only small parts of the text have been left in the clear.

The Rare Book & Manuscript Library acquired Rinzi's manuscript a couple of years ago from the Librairie Paul Jammes in Paris. They inquired about additional provenance information but the book dealer did not know much, except that he had obtained the manuscript as part of the remaining stock of a fellow bookseller who had specialized in all things "curious, mysterious and unusual." No other other writings by Rinzi are known to us, let alone encrypted ones.

Rinzi wrote his enciphered journal in a minuscule hand that requires magnification to see clearly. While each journal page (sized 18 cm x 12 cm) is ruled with twenty-one lines, Rinzi managed to fit over a hundred lines of text. Based on these numbers, we estimate that the total amount of characters contained in the manuscript is about 1.5 million, which corresponds to a novel of about 800 pages. Rinzi used a non-standard alphabet consisting of astrological symbols as well as Chinese, Greek and Hebrew letters. It can be assumed that Rinzi assembled this alphabet himself. The first page of the manuscript appears to contain a list of the characters used, but no substitution table.

Among the passages in Rinzi's journal that are

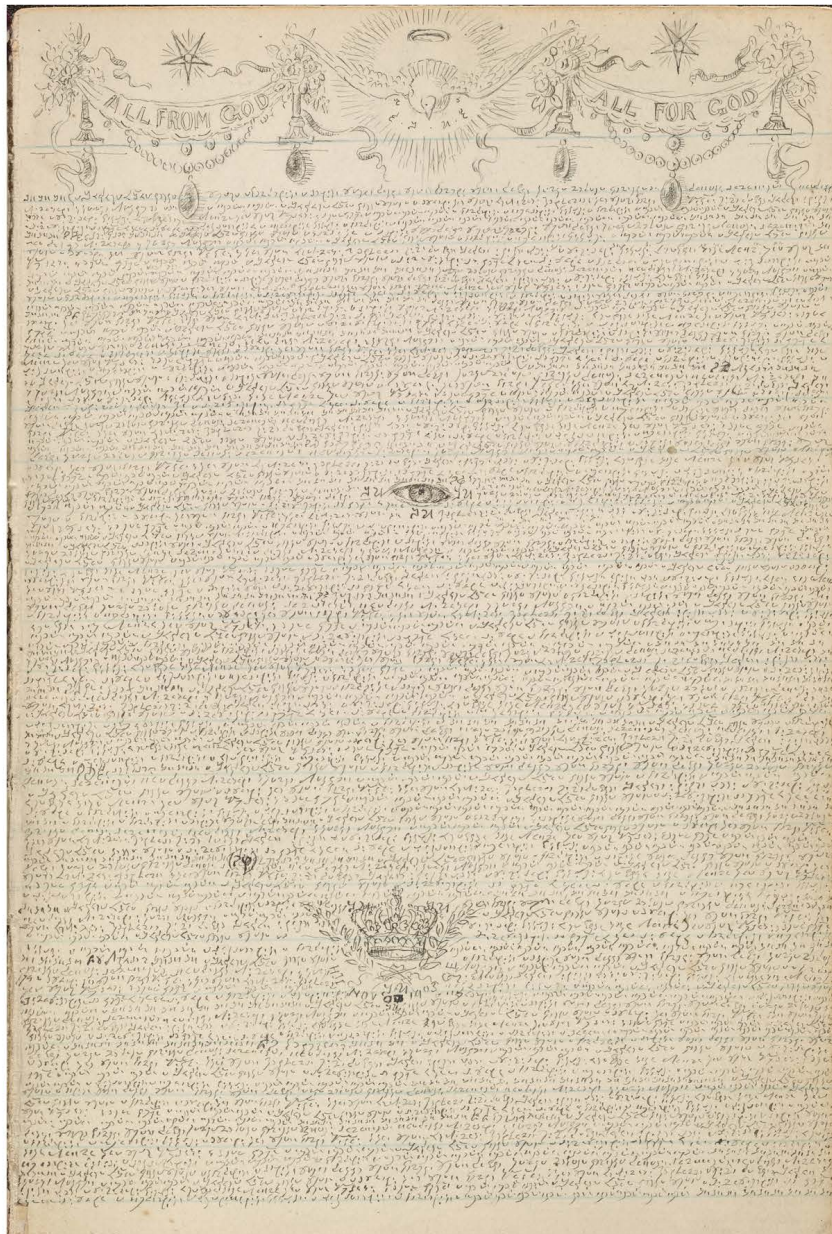


Figure 1: A typical page from Ernest Rinzi's encrypted journal. There are about 100 written lines per page.

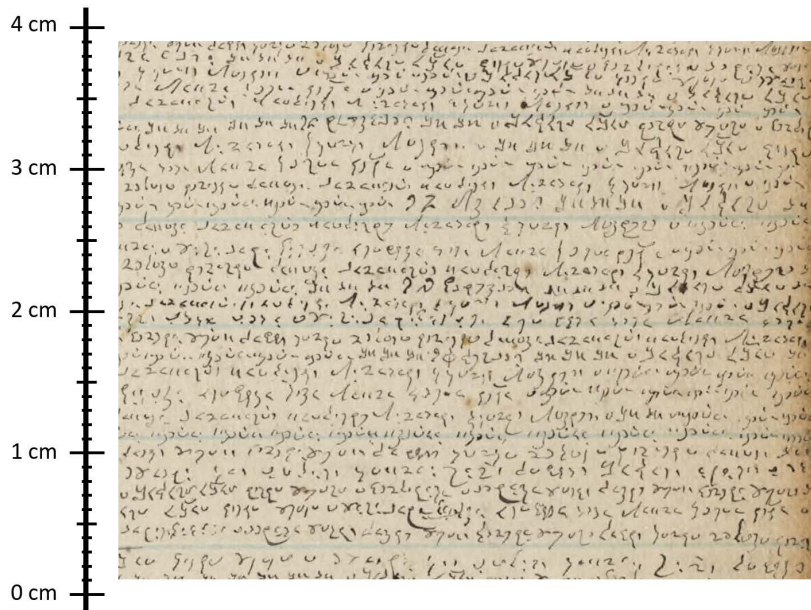


Figure 2: Rinzi’s miniature writing is hard to read. There are up to eight lines per centimeter. Rinzi used a non-standard alphabet containing of astrological symbols as well as Chinese, Greek and Hebrew letters.

left in the clear are some names and events, such as the assassination of King Umberto I in 1900, the death of British queen Victoria in 1901, the coronation of British king Edward VII in 1901, and the death of Rinzi’s wife Jessie in 1902. Rinzi’s writing and drawing shows religious devotion. Most pages are topped with illustrations that feature the Holy Spirit in the form of a dove, angels, or banners, and the motto “All from God, all for God, all to God” (written in English, Italian, or French). While the angels first appear mostly in black and white, they become progressively more colorful throughout the journal. Apart from the “All from God, all for God, all to God” illustrations on many pages, there are other drawings spread throughout the manuscript. Illustrations and text always form a unity – sometimes the pictures are integrated into the text, sometimes it’s the other way round.

### 3 Analysis

As far as we know, Ernest Rinzi’s journal is not mentioned in the crypto history literature. The only public source mentioning it originally were two blog posts published by the Rare Book & Manuscript Library in 2017 (Anonymous, 2017) (Anonymous, 2017). David Scheers, a fellow crypto history scholar, made the second author of this paper aware of these posts. This second author subsequently informed other crypto history

scholars via his blog (Schmeh, 2017). The two blog posts described Rinzi’s journal as unsolved. A Google search we conducted did not reveal any other sites mentioning it, let alone the solution. Later an employee of the Rare Book & Manuscript Library told us: “We’ve puzzled over this document for the last couple of years, and come up with all sorts of guesses and fantasies as to what its content might be.”

What is especially fascinating about Rinzi’s journal, is the combination of art and encryption. The drawings are of high artistic quality. Rinzi’s encrypted journal reminded us of a number of other notable encrypted books (which doesn’t mean, of course, that a similar encryption method was used):

- *The Voynich Manuscript*: The Voynich Manuscript is the most famous unsolved cryptogram in the world. Like Rinzi’s journal, it combines text written in an unknown script with illustrations. However, the artistic quality of the illustrations is considerably lower.
- *The Rohonc Codex*: Similar as the Voynich Manuscript, the Rohonc Codex is written in an unknown script and contains illustrations. A solution was recently published but is not generally accepted to date (Király and Tokai,



Figure 3: Most pages in Rinzi’s journal are topped with illustrations that feature the Holy Spirit in the form of a dove, angels, banners, and the motto “All from God, all for God, all to God”.

2018).

- *Charles Dellschau’s books*: The manuscripts created by US outsider artist Charles Dellschau contain encrypted text and illustrations, just like Rinzi’s journal (Schmeh, 2017). However, the focus in Dellschau’s books lies on the paintings, while (encrypted) text plays only a minor role. Dellschau’s ciphertexts have been solved.
- *James Hampton’s journal*: US outsider artist James Hampton left behind a journal of over 100 pages written in an unknown script (Schmeh, 2018). This journal mainly consists of text; there are only few drawings. Hampton’s notebook as well as a few other writings in the same script he left behind have never been deciphered.

After having read the blog post about Rinzi’s journal, the first author of this paper started to analyze it. He assumed that the journal was written in Italian, English, or French, as these are the languages that appear in the cleartext passages of the journal. Italian, which was Rinzi’s mother language, appeared to be the most likely choice. Considering the amount of text and the fact that the journal probably was written for himself (and not for an English-speaking audience), it seemed likely that Rinzi wrote in the language he was most fluent in. English appeared to be the second option. Rinzi emigrated to London in the age of 23, and so it seemed plausible that after almost four decades he spoke English well enough to easily write such a huge amount of text. As Rinzi’s biography is not documented very well, it was hard to say whether French was another plausible option. The French passages in the journal are not proof that Rinzi really spoke this language.

The large amount of encrypted text the journal contains made it unlikely that Rinzi had used a complex encryption system. The first author of this paper assumed that a Monoalphabetic Substitution Cipher (MASC) had been applied, which meant that each of the glyphs in the journal corresponded with a certain letter, number, or sign. The number of glyphs in the alphabet Rinzi used is 36. The most obvious explanation was that these glyphs stood for the letters from A to Z and the numbers from 0 to 9.

#### 4 Solution

One method to solve a MASC is to guess plaintext words. When leaving through the journal, the first author of this paper found that the lower part of page 127, which is dedicated to the death of Rinzi’s wife Jessie (figure 5), provided a mixture of encrypted and non-encrypted content. This looked like a good place for guessing words. The lower part of page 127 can easily be located in the journal, as it is this only passage with a black background. Ten lines up from the bottom left of this page there is a three-letter word followed by a seven-letter word in the next line. Both words occur numerous times in the journal. As the text and the illustrations appeared to contain religious content, DIO (Italian for “god”) seemed a good candidate for the three-letter word. The English translation GOD worked, too, while the French equivalent DIEU did not fit. Substituting the letters D, I, and O in the seven-letter word resulted in DI?I?O, while G, O, and D rendered GO?O?D. So, the first author of this work concluded that the Italian variant (D, I and O) made sense and that the seven-letter word was DIVINO (“devine”).

On the same page, just up and to the right beneath the flowers, the first author of this work

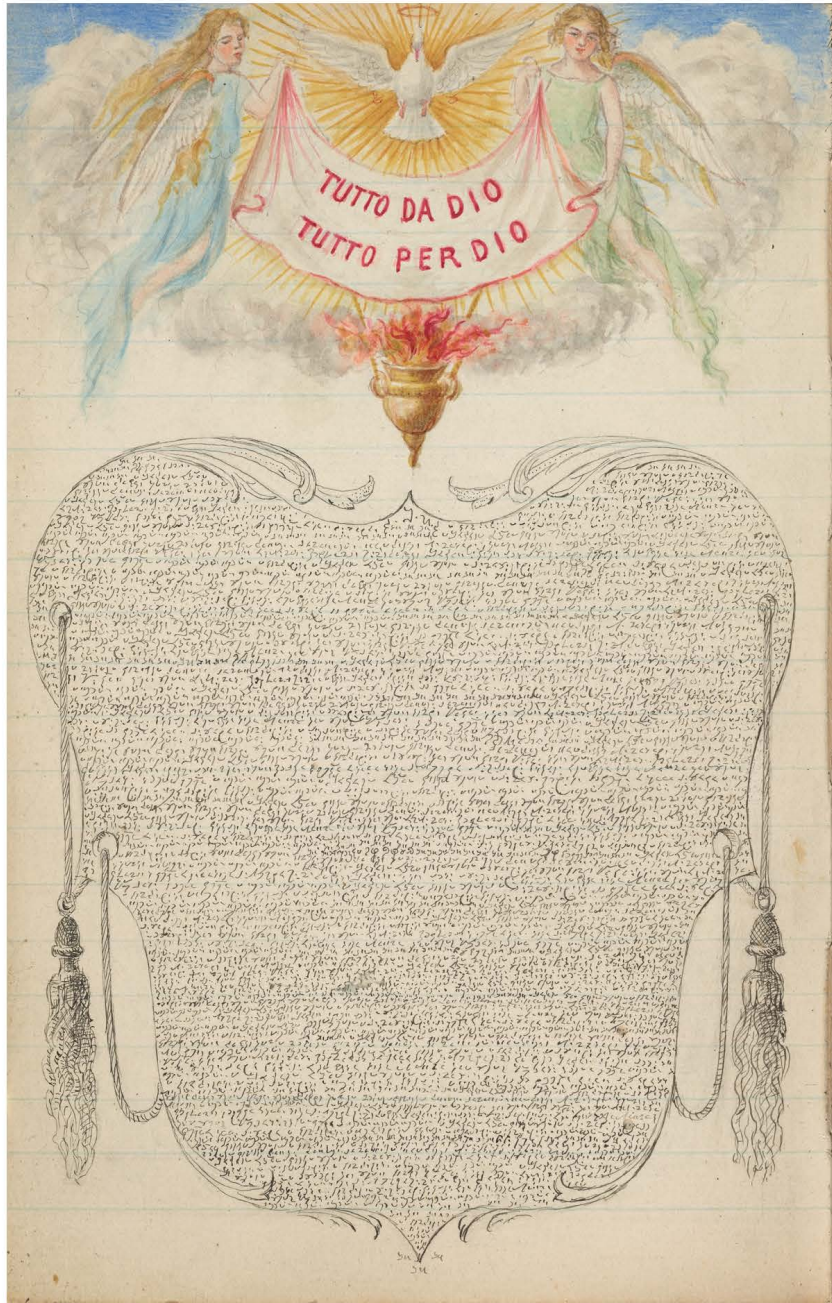


Figure 4: Some of the 175 pages in Rinzi's journal show additional illustrations. They are usually integrated into the text.

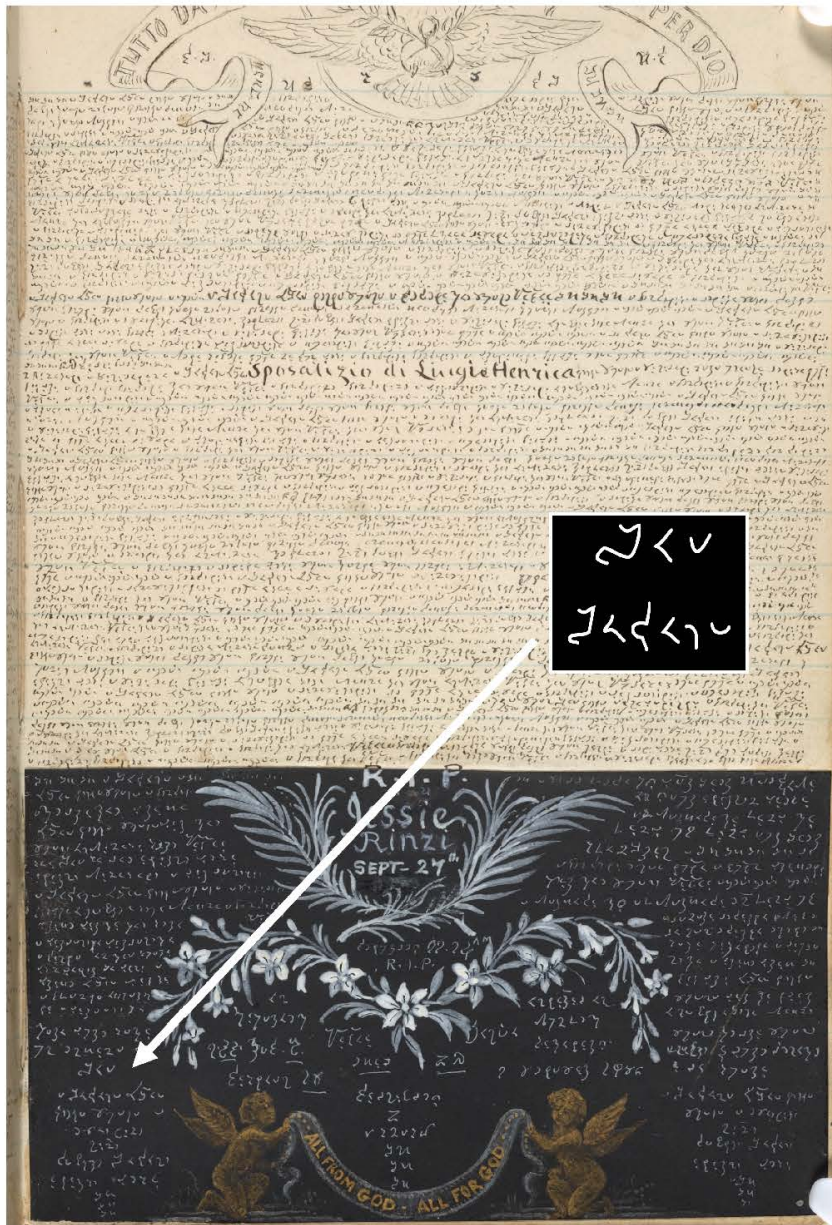


Figure 5: The first author of this work used two words on this page to break into the cipher.

found a two-letter word that, assuming that the Italian hypothesis was correct, became IN. Beneath this, an eight-letter word appeared, where the initial letter occurred twice more in the word. The character used for this letter occurs in many ancient alphabets and is the precursor of our modern M. Assuming that this character actually stood for the M, the expression M?MO?I?M was received. In context of the page it probably meant “in memory of”, which is MEMORIAM in Italian. With nine letters identified, the remaining ones could be guessed, too.

The first author of this work now assumed that the ten symbols not standing for letters were numbers. The numbers under the flowers (? OCTOBER ???? ) on page 127 most likely indicated the burial date. As Jessie Rinzi died in 1902, the ???? apparently stood for 1902, which rendered the 1st October 1902 as the day when she was buried. The time of death and her age also appear in this section. Substituting the known digits gave the ciphertext representations of 0, 1, 2, and 9.

A fellow crypto history enthusiast provided us the information that Jessie Rinzi was 33 years old at the time of the British census of 31 March 1901. As “September 27th” is written in the clear below “R.I.P.” (at the top of the passage with the black background), we concluded that Jessie died on 27 September 1902, at the age of 35. This is confirmed by the fact that above the second “R.I.P.” (between the willows and the flowers) it reads SATURDAY ???? AM (this is the only English expression we have found in the encrypted text so far) and 27 Sept 1902 was a Saturday. Based on this information a few more digits could be guessed. The lower line of figure 6 shows the substitution table we derived. It seems possible that the digits 5 and 6 have to be switched. So far, we haven’t found a number appearing in the encrypted text that allows for a definite identification of these two digits.

Figure 7 shows a part of the ciphertext that has been decrypted. The first line reads as follows: O DIVINO IDDIO SANTO BUONO O SALVATERIO GRAZEI GRAZI. As far as we can tell based on this short plaintext part, Rinzi’s religious beliefs play an important role in his journal.

## 5 Conclusion and Outlook

The encrypted journal of Ernest Rinzi is a remarkable document. Especially, the amount of text

(probably about 1.5 million characters), the miniature writing and the illustrations make Rinzi’s journal something very special. The cipher Rinzi used turned out to be a monoalphabetic substitution (MASC) with an alphabet of 36 letters. The first author of this paper broke it based on word guessing. The plaintext language turned out to be Italian. Rinzi’s bad penmanship (probably owed to the small size of the writing) complicated cryptanalysis.

There are a number of open questions about Rinzi’s journal. Especially, we ask ourselves the following:

- *What’s the content of Rinzi’s journal?* So far, only a few sentences of the journal have been decrypted. Decrypting more, let alone the whole journal, will require much more time and effort. Perhaps, somebody interested in Ernest Rinzi’s life will be interested in such a project.
- *Are there other encrypted documents Rinzi left behind?* The journal this article is about was written between 1898 and 1903. Provided that we deal with a diary or a similar document, it is well possible that other journals of this kind exist. In addition, Rinzi might have created other encrypted documents. Perhaps, he even included encrypted text in some of his artworks.
- *Why did Rinzi write this book?* Encryption is usually used to hide information from others. It is therefore an interesting question, from whom Rinzi’s wanted to hide his writing. Did he want to keep his family members from reading his journal? And why did he include all these elaborate illustrations, if he didn’t plan to reveal his journal to others? We have no answers to these questions.
- *Are works created by Rinzi on display in museums?* Though we found many artworks of Ernest Rinzi on the websites of auction houses and art registers, we are still not aware of a Rinzi creation that is on display in a museum or public collection.
- *Can we find out more about Rinzi’s biography?* Finally, it would be interesting to know more about Ernest Rinzi’s biography.

A	B	C	D	E	F	G	H	I	J	K	L	M
⊃	⊂	⊃	⊂	⊃	⊂	⊃	⊂	⊃	⊂	⊃	⊂	⊃
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
⊂	⊃	⊂	⊃	⊂	⊃	⊂	⊃	⊂	⊃	⊂	⊃	⊂
0	1	2	3	4	5	6	7	8	9			
⊂	⊃	⊂	⊃	⊂	⊃	⊂	⊃	⊂	⊃			

Figure 6: This substitution table shows how the cipher works. It is a simple substitution cipher (MASC), but things are complicated by the facts that the writing is tiny and the alphabet contains 36 letters.

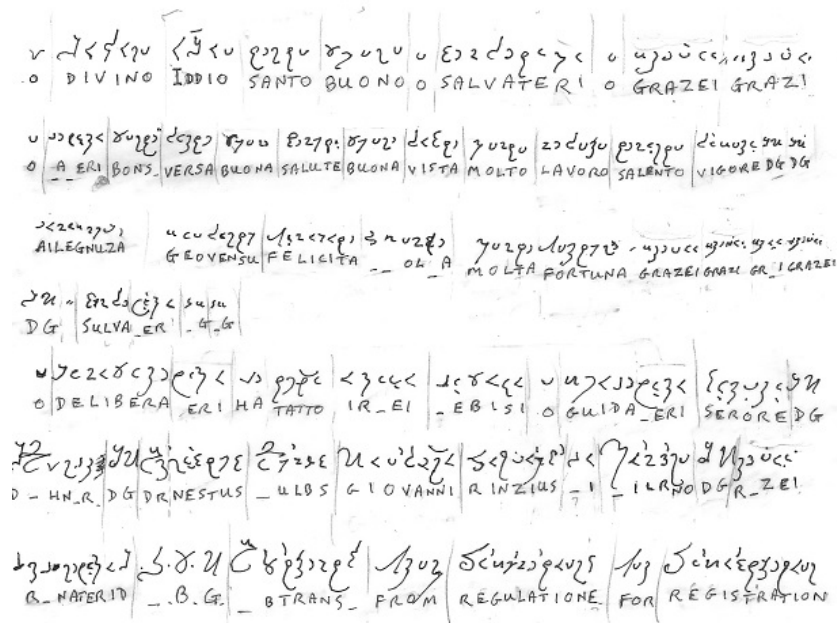


Figure 7: A piece of text from Rinzi's journal that has been decrypted. Decrypting the whole journal would be a laborious project.



These questions have to be answered by historians, art historians, and psychologists. From a cryptographers point of view, the case of Ernest Rinzi's journal is solved.

## Acknowledgements

We would like to thank the Rare Book & Manuscript Library at the University of Illinois at Urbana-Champaign (especially Associate Professor Caroline Szylowicz), David Scheers, Norbert Biermann, Paolo Bonavoglia, and Thomas Bosbach.

## References

- Anonymous. 2017. *Ernest Rinzi: All from God*. <http://illinoisrbml.tumblr.com/post/165049087613/ernest-rinzi-all-from-god-ernest-rinzi-born-as>.
- Anonymous. 2017. *Ernest Rinzi: Master of Miniatures*. <http://illinoisrbml.tumblr.com/post/164792448156/ernest-rinzi-master-of-miniatures-ernest-rinzi>
- Király, Levente Zoltán; Tokai, Gábor. 2018. *Cracking the code of the Rohonc Codex*. Cryptologia Volume 42, 2018 - Issue 4, p. 285-315
- Rinzi, Ernest. 1898-1903. *Bound manuscript journal*. From the Rare Book & Manuscript Library at the University of Illinois at Urbana-Champaign, Post-1650 Manuscript Collection.
- Royal Collection Trust. 2019. *Ernest Rinzi (1836-1909): Queen Victoria (1819-1901) Signed and dated 1897*. <https://www.rct.uk/collection/422305/queen-victoria-1819-1901>.
- Schmeh, Klaus. 2017. *A phantastic discovery: The encrypted journal of Ernest Rinzi*. <http://scienceblogs.de/klausis-kryptokolumne/2017/09/21/a-phantastic-discovery-the-encrypted-journal-of-ernesto-rinzi/>
- Schmeh, Klaus. 2017. *The mysterious paintings and cryptograms of Charles Dellschau*. <http://scienceblogs.de/klausis-kryptokolumne/2017/01/31/the-mysterious-paintings-and-cryptograms-of-charles-dellschau/>
- Schmeh, Klaus. 2018. *The Top 50 unsolved encrypted messages: 10. James Hampton's notebook*. <http://scienceblogs.de/klausis-kryptokolumne/2018/03/04/the-top-50-unsolved-encrypted-messages-10-james-hamptons-notebook/>



# 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 polyalpha-

betic 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" → 34521, "Rome" → 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 plot 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” → 46621, “need” → 12315, “supplies” → 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” → 5100 and “tools” → 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 substitu-

tion ciphers. Having a ciphertext with a length of 1000 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-

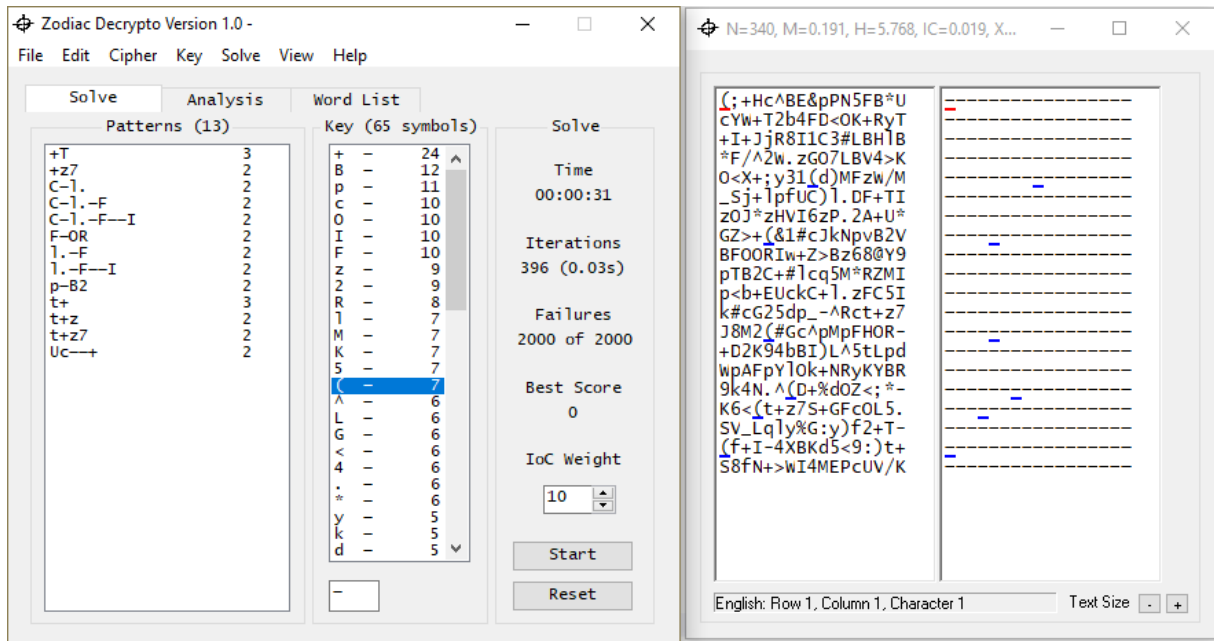


Figure 1: Screenshot of the Tool “Zodiac Decrypto”

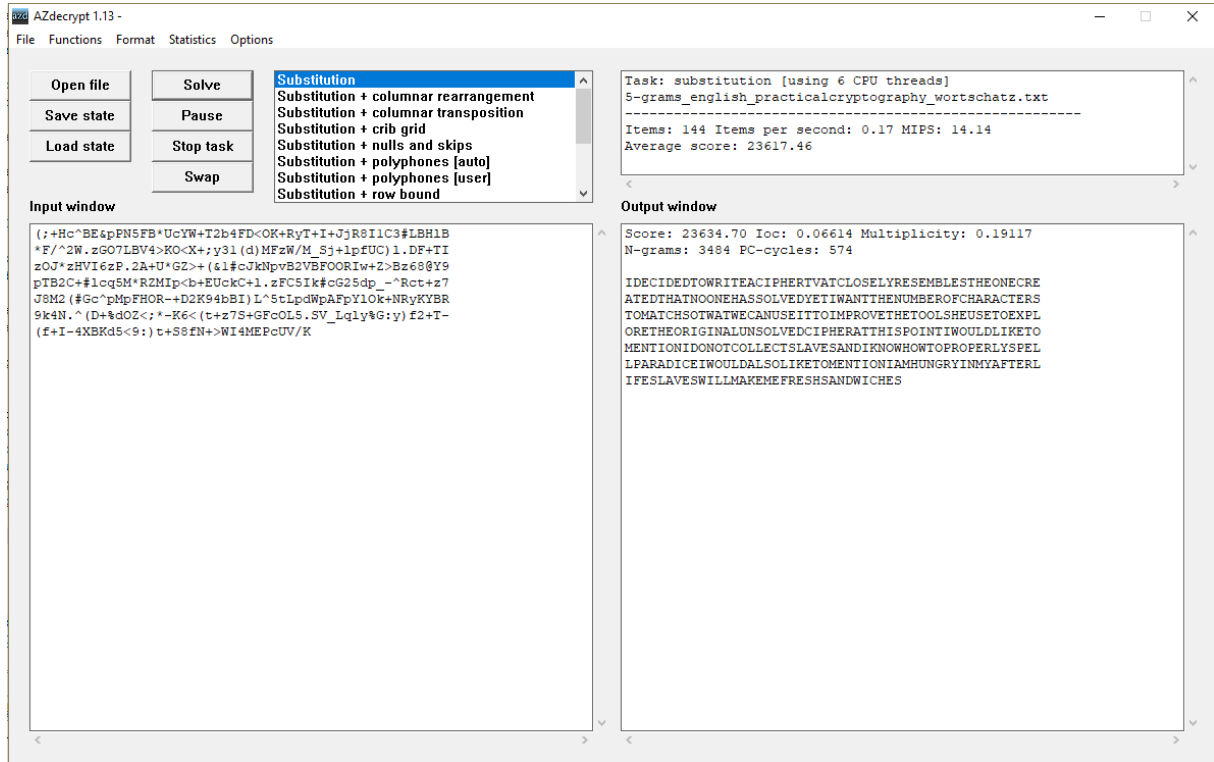


Figure 2: Screenshot of the Tool “AZdecrypt”

phertext with respect to minimizing the collision of words. Having 1000 generations in the genetic algorithm and a minimum dictionary size of about 1300 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 with  $n > 1$  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*<sup>1</sup> 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<sup>2</sup> 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"<sup>3</sup> was written by Eycke since 2016. The newest version (1.13) was

<sup>1</sup>ZKDecrypto can be downloaded from: <https://code.google.com/archive/p/zkdecrypto/>

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

<sup>3</sup>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.

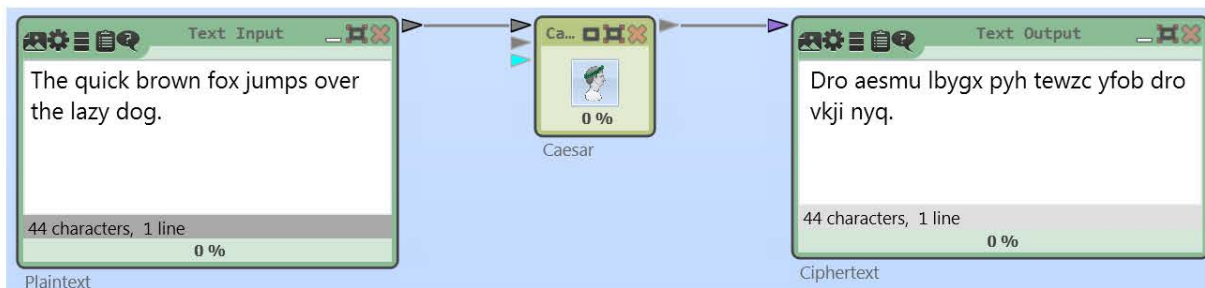


Figure 3: A CrypTool 2 Workspace with a Caesar Cipher

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<sup>2</sup> 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 pentagram 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 pro-

gram, 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).



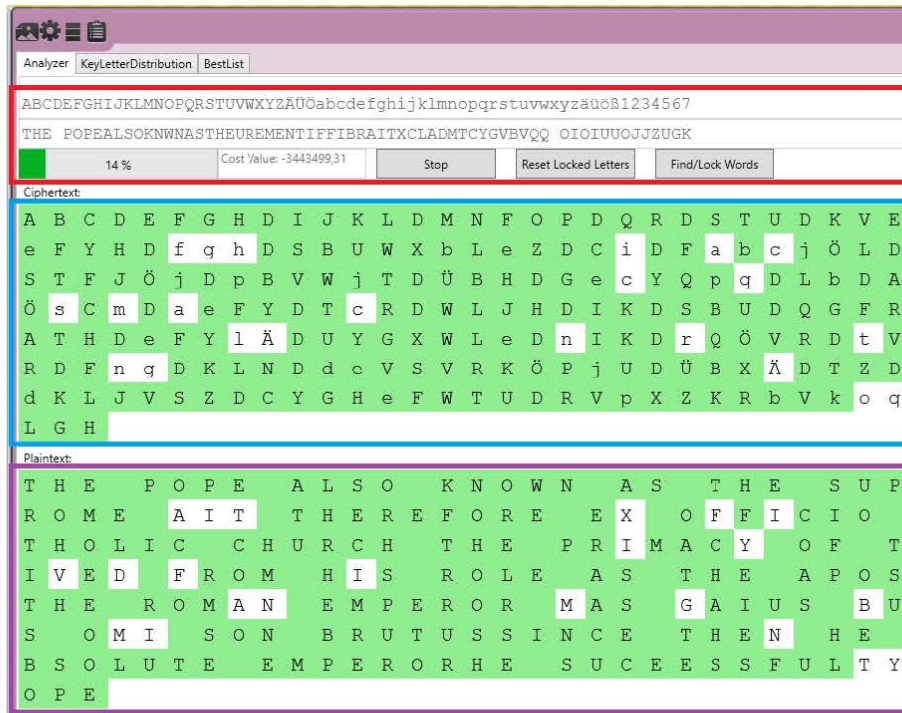


Figure 4: Analyzer Tab of the Homophonic Analyzer in CrypTool 2

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

#	Value	Key	
1	-3443499.31	THE POPEALSOKNWNASTHEUREMENTIFFIBRAITXCLADMTCYGVBVQQ OIOIUUOJJZUGK	THE POPE ALSO KNOWN AS
2	-3472860.09	THE POPEALSOKNWNASTHEUREMENTIFFIBRAIDGCLATM CYGUOTQJVQIKOXIOBUJZV	THE POPE ALSO KNOWN AS
3	-3519138.4	THE POPEALSOKNWNASTHEUREMENTIFFIBRATOUCLADMTCYGVVOVGJ IQZIKXIQBUU	THE POPE ALSO KNOWN AS
4	-3520840.45	THE POPEALSOKNWNASTHEUREMENTIFFIBRATTGCLADB CYGZUMIOJKOUUQVJVXIIQO	THE POPE ALSO KNOWN AS
5	-3521257.66	THE POPEALSOKNWNASTHEUREMENTIFFIBRAIDUCLATMTCYGBZBJ QOVKOJVUOXQUGI	THE POPE ALSO KNOWN AS
6	-3571963.89	THE POPEALSOKNWNASTHEUREMENTIFFIBRAUDOCCLATJTCYGVOKGJZOIIMVQXIQ UB	THE POPE ALSO KNOWN AS
7	-3573428.33	THE POPEALSOKNWNASTHEUREMENTIFFIBRATOICLADMTCYGVBBQOI KOXIUUGJVJZ	THE POPE ALSO KNOWN AS
8	-3588347.55	THE POPEALSOGNONASTHEUREMENTIFFIBRANDXCLADW TEGVOKQKIOJUIZQXJUIZM	THE POPE ALSO GNOON AS
9	-3603951.59	THE POPEALSOOMONASTHEUREMENTIFFIBRANDUCLADW TEGZYVQKUIUQOXGZKJUIX	THE POPE ALSO OMOON AS

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  = THEPOPEALSOKNWAESTZ...
```

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:

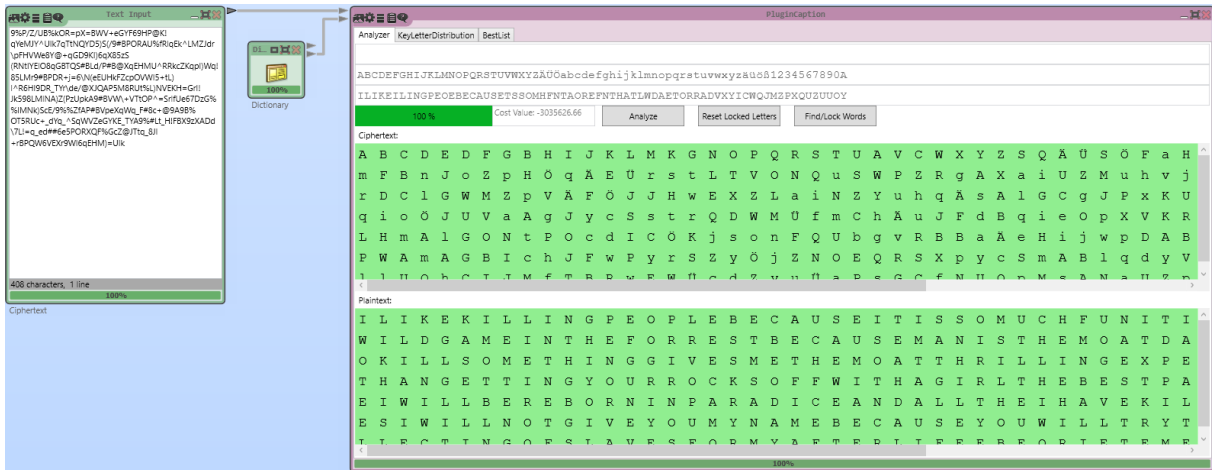


Figure 6: Breaking Zodiac-408 with the New Homophonic Substitution Analyzer in CT2

- 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 15000). 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<sup>2</sup>, 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 homophonicly.

### 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 1000 cycles. By chance, we got some parts of partial words, i.e. PEOPLE or AFTER-LIFE, 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 CryptTool 2 (CT2) within the DECRYPT project. Right now, the analyzer is already able to full-automatically and semi-automatically break real world homophonic ci-

phers 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 Vaticanum, <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/>.

# Using the Entropy of N-Grams to Evaluate the Authenticity of Substitution Ciphers and Z340 in Particular

Tom S Juzek

Saarland University

Campus A2.2, R1.25

66123 Saarbrücken, Germany

tom.juzek@uni-saarland.de

## Abstract

The present paper uses information theoretic entropy as a means to evaluate the authenticity of homophonic substitution ciphers. We motivate the use of entropy on n-grams and then validate its applicability, by using it on various true ciphers and pseudo-ciphers. Differences in entropy allow us to apply further formal analyses, e.g. support-vector machines, in order to make predictions about a potential cipher's status. We train several support-vector machines and validate them. We then apply the models to two classic ciphers, the Zodiac Killer's first major cipher, z408, which has been solved, and his second cipher, z340, which remains unsolved. The models correctly identify z408 as a substitution cipher. z340 is classified as an advanced cipher or pseudo-cipher.

## 1 Introduction

For unsolved ciphers, it is often difficult to determine which type of cipher is applicable. A seemingly encrypted text might be a simple homophonic substitution cipher, a transposition cipher, a Vigenère cipher, and the like, or it might not even be a *bona fide* cipher at all. Even if one suspects that a cipher is a simple homophonic substitution cipher, then there are only a few known analyses that count as formal evidence, such as e.g. Ravi and Knight (2008), Corlett and Penn (2010), and Ravi and Knight (2011).

The present paper presents another formal measure that can be used to detect underlying language-like information in a possible homophonic substitution cipher (in the following shortened to just *substitution cipher*). The measure quantifies differences in (information theoretic) entropy between a cipher and a meaningless baseline. Critically, in contrast to previous analyses,

the measure can be used to train classifiers like support-vector machines or k-means clusters to make predictions about the status of a text.

The present paper is structured as follows: In Section 2, we review common types of substitution ciphers, including one-to-one ciphers, one-to-many ciphers, and many-to-many ciphers. Our focus is on one-to-many ciphers and many-to-many ciphers.

The Zodiac Killer's second major cipher, z340, is often assumed to be a substitution cipher, but it is generally accepted that it has not been solved yet. Section 3 briefly introduces z340 and compares it to the Zodiac Killer's first major cipher, z408, which has been deciphered, cf. Graysmith (1987).

We give basic analyses of z340 and z408, viz. character frequency analyses and n-gram analyses, in Section 4.1. We compare the two ciphers to a pseudo-cipher that we have created, to illustrate the limitations of character frequency analyses. N-gram analyses, on the other hand, give useful hints about the authenticity of a cipher. However, even more powerful measures are needed to give more definite insights.

For this, we use the information theoretical measure of entropy, cf. Shannon (1948). We calculate entropy based on bigrams and trigrams and we apply the measure to various texts and ciphers. To be able to train a classifier model, some sort of baseline is needed and we establish such baselines for plain texts and ciphers. All this is done in Section 4.2. In Section 4.3, we then use the results to train a support-vector machine, which we validate on further test data. The classifiers give very good results for plain texts and one-to-many substitution ciphers and good results for many-to-many substitution ciphers. The models correctly classify z408, the Zodiac Killer's solved cipher, as a substitution cipher. z340, the unsolved cipher, is classified as an advanced cipher or pseudo-cipher.

Section 5 discusses a few issues with the measure and Section 6 concludes the paper.

## 2 Substitution ciphers

Common substitution ciphers include one-to-one ciphers, one-to-many ciphers, and many-to-many ciphers. For an introduction to ciphers (and cryptography in general), see e.g. Hoffstein et al. (2008). In the following, we use “letter” to refer to a unit of an unencrypted plain text, “symbol” for a unit of a cipher, and “character” for both.

In a one-to-one cipher, any plain text letter is mapped to exactly one cipher symbol, e.g. “A” to “X”, “B” to “Y”, etc. Let  $l$  be a letter of the set of letters  $\mathbf{L}$ ,  $s$  a symbol in the set of symbols  $\mathbf{S}$ . This is expressed through the relation  $r$  in Equation 1: Any given letter  $l$  maps to exactly one symbol  $s$ ; and any given symbol  $s$  maps to exactly one letter  $l$ .

In a one-to-many cipher, some or all plain text letters are mapped to more than one cipher symbol, e.g. “A” might be mapped to both “X” and “Ω”, “B” might be mapped to “Y” and “Ψ”, etc. Here, too, each cipher symbol is used for one plain text letter only. In our example, “Ω” is always “A”, etc. One-to-many ciphers are expressed through the relation in Equation 2: Any given letter  $l$  maps to at least one symbol  $s$ ; but any given symbol  $s$  maps to exactly one letter  $l$ .

In a many-to-many cipher, there is no restriction of having exactly one mapping back from symbol to letter. Each plain text letter can map to more than one cipher symbol and vice versa. “A” might be mapped to both “X” and “Ω”, “B” might be mapped to “Y”, but also “Ω”, etc. “Ω” now represents both “A” and “B”. Many-to-many ciphers are expressed in Equation 3: Any given letter  $l$  maps to at least one symbol  $s$ ; and any given symbol  $s$  maps to at least one letter  $l$ .

$$r(l) \mapsto \exists!s \ s \in \mathbf{S} \quad \text{and} \quad r(s) \mapsto \exists!l \ l \in \mathbf{L} \quad (1)$$

$$r(l) \mapsto \exists s \ s \in \mathbf{S} \quad \text{and} \quad r(s) \mapsto \exists!l \ l \in \mathbf{L} \quad (2)$$

$$r(l) \mapsto \exists s \ s \in \mathbf{S} \quad \text{and} \quad r(s) \mapsto \exists l \ l \in \mathbf{L} \quad (3)$$

Table 1 gives an example of a many-to-many cipher. Note how e.g. “S” maps to both “10” and “57” and how “31” represents both “Y” and “G”.

Figure 1: The beginning of the z340 cipher.

The text is the beginning of *The Confession*, the first letter attributed to the Zodiac Killer, cf. Gray-Smith (1987). The letter can be found on Wikisource (The Wikimedia Foundation, 2018b). *The Confession* also features as one of the texts used in Section 4.2.

One-to-one substitution ciphers are relatively easily detected – unless some other trick is employed. The focus of the present paper is on one-to-many ciphers and many-to-many ciphers. Such ciphers come in degrees. While weak instances can be rather easy to decrypt, strong instances can be very hard, sometimes virtually impossible to decrypt (see below for details).

## 3 z340

The Zodiac Killer’s second major cipher, z340, is often assumed to be a substitution cipher. However, even 50 years after its appearance, a generally accepted solution is lacking. Figure 1 illustrates the beginning of the cipher and the full cipher can be found on Wikisource (The Wikimedia Foundation, 2018b). This contrasts to the Zodiac Killer’s first major cipher, z408. z408 is generally assumed to have been solved by Donald and Bettye Harden. Parts of the cipher and the full solution can also be found on Wikisource (The Wikimedia Foundation, 2018b). [Zodiackillerciphers.com](http://Zodiackillerciphers.com) (2012) contains the full cipher and the Harden solution.

## 4 Analyses

### 4.1 Character frequencies and n-gram frequencies

To gain first insights into a potential substitution cipher, there are two basic analyses: A character frequency analysis and an n-gram analysis. Character frequencies provide the number of occurrences per character of a plain text or cipher. N-gram analyses of characters give the number of occurrences per n-gram (bigram, trigram, etc.) in a text – see e.g. Jurafsky (2019) for an introduction to n-grams.

<i>Plain text:</i>	S	H	E	W	A	S	Y	O	U	N	G	A	N	D	B	E	A	U	T	I	F
<i>Encrypted:</i>	10	21	48	40	57	47	31	53	51	28	31	7	44	26	6	25	4	30	1	8	29

Table 1: An example of a many-to-many substitution cipher. The top row is the plain text, below it is the encrypted message, using the encryption algorithm described in Section 4.2.

For example, if the cipher text was simply “DADAISM”, then the character frequencies are as follows: ‘A’:2, ‘D’:2, ‘I’:1, ‘M’:1, ‘S’:1. The bigram counts are as follows: ‘DA’:2, ‘AD’:1, ‘AI’:1, ‘IS’:1, ‘SM’:1.<sup>1</sup> The trigrams are: ‘DAD’:1, ‘ADA’:1, ‘DAI’:1, ‘AIS’:1, ‘ISM’:1.

However, character frequency analyses are only of limited value when one wishes to assess the authenticity of substitution ciphers. When using character frequencies, it is difficult to distinguish a semi-random string that observes common character frequencies and that is then encrypted using a true cipher. However, differences will show up in an n-gram analysis. This is illustrated in Figure 2. Figure 2 (left) plots the character frequencies of a true cipher, z408, and those of a comparable pseudo-cipher, which is based on a semi-random string. The pseudo-cipher is comparable in length and has a similar symbol set to z408. For details regarding the encryption algorithm, see Section 4.2.

Character frequencies of both ciphers are plausible. However, as Figure 2 (right) illustrates, the bigram frequencies reveal differences – while the bigram frequencies of z408 are plausible, the bigram frequencies for the pseudo-cipher fall “flat”, i.e. the pseudo-cipher seems to lack a plausible bigram count.

It has been noted that the Zodiac Killer’s unsolved z340 also seems to lack plausible n-gram counts (Knight, 2013, p. 91). Figure 3 gives the character frequencies and bigram frequencies for z340. Note how z340 rather resembles the pseudo-cipher than the true cipher.

The n-gram analysis of z340 indicates that it might not be a *bona fide* cipher. However, a more formal analysis is needed for more conclusive evidence. We provide such evidence by analysing the entropy of various texts, including true ciphers and pseudo-ciphers, and then training several support-vector machines on the results.

<sup>1</sup>And ‘D’:1 and ‘M\$’:1 if one wishes to account for beginning of line (“”) and end of line (“\$”). In the following, we will not include those two.

## 4.2 Entropy as a measure of authenticity

Entropy in information theory, as introduced by Shannon (1948), is a measure of order of a system and can be applied to various levels of a linguistic sequence, including characters, n-grams, words, multiple words, and entire sentences. The general formula for entropy,  $H$ , is given in Equation 4. In our case,  $F$  is the frequency of the respective bi- and trigrams.

$$H = -\sum_i F_i \log(F_i) \quad (4)$$

For instance, the bigram entropy for “DADAISM” is 0.68, the entropy for “IADS-DMA” is 0.78. However, it can be difficult to make sense of the values, especially when it is compared across sequences with different symbol sets and of different lengths. Consider the bigram entropy for the pseudo-cipher from above: It comes out at 2.52 – but it is not clear what this exactly means.

Thus, for ease of interpretation, we compare a sequence’s entropy,  $H_s$ , to the entropy of a meaningless baseline,  $H_b$ . For any sequence, we pseudo-randomly shuffle the sequence in question and use the shuffle as the baseline. To avoid distortions of an unlucky shuffle, we take 1000 shuffles and average their entropy values. This is  $\overline{H}_b$ . The absolute difference between  $H_s$  and  $\overline{H}_b$  is what we abbreviate by  $|\Delta_H|$ . Creating the pseudo-random baselines and calculating  $|\Delta_H|$  is done with a script that we wrote in Python (Python Software Foundation, 2018) (all scripts can be found on GitHub<sup>2</sup>.  $|\Delta_H|$  is easier to interpret: A value of 0 means that the sequence lacks any order, just like the pseudo-random baselines. The greater the value, the more ordered a sequence is. Accordingly, the  $|\Delta_H|$  for “IADSDMA” is 0.0, but the  $|\Delta_H|$  for “DADAISM” is 0.1.  $|\Delta_H|$  for the pseudo-cipher from above is 0.0, but  $|\Delta_H|$  for the true cipher z408 comes out at 0.06.

However, even a  $|\Delta_H|$  can be hard to interpret. What does a  $|\Delta_H|$  of 0.06 mean? Some contextualisation is needed and in order to provide it, we

<sup>2</sup><https://github.com/superpumpie/z340.2>.

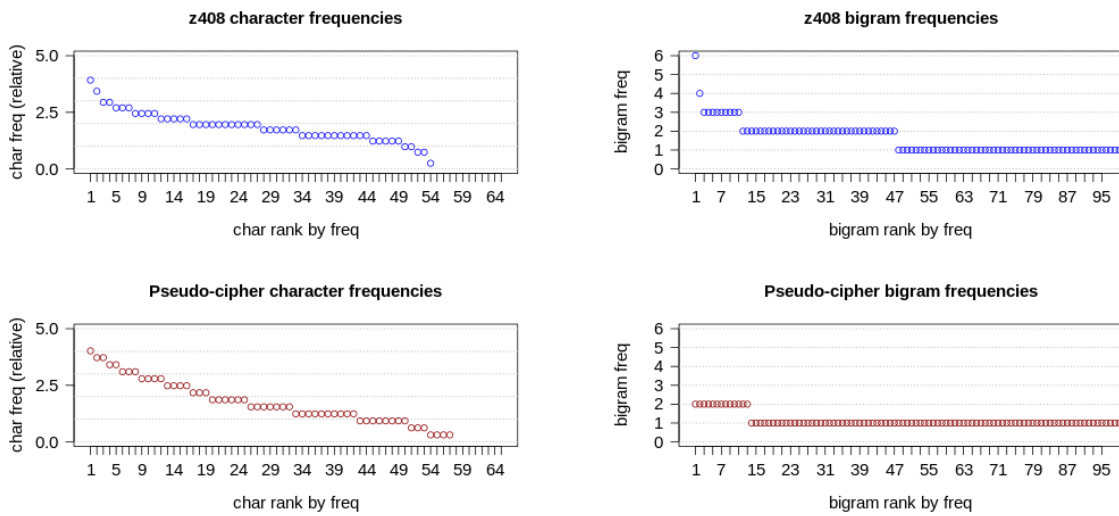


Figure 2: Left: The relative character frequencies, y-axis, for the true cipher z408 (top) and for a pseudo-cipher based on a semi-random string (bottom). Items are ordered on the x-axis by their frequencies in descending order. Right: The bigram frequencies, y-axis, for the same two ciphers, again in descending order as per rank.

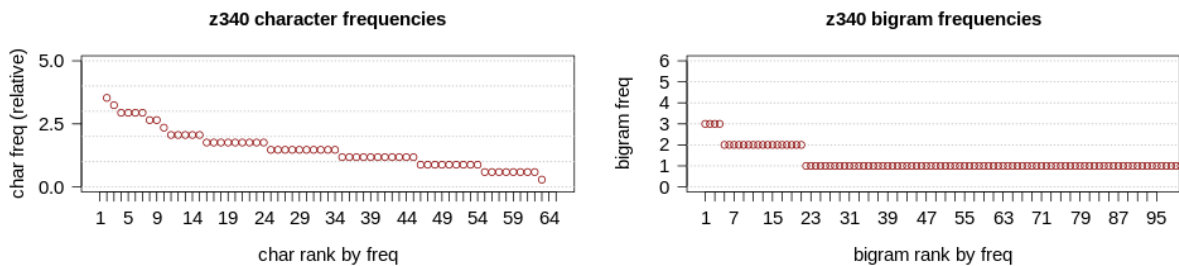


Figure 3: Left: The relative character frequencies, y-axis, for the unsolved z340 cipher. Items are ordered on the x-axis by their frequencies in descending order. Right: The bigram frequencies, y-axis, for the z340 cipher, again in descending order as per rank.



analyse various plain texts vs semi-random strings and various true ciphers vs pseudo-ciphers. We begin with a training set, which we then use to train several support-vector machines. We later validate the models with further test data sets.

As a first step, we analyse 32 plain texts and 32 semi-random strings. The majority of those are snippets from the Top 100 books on Project Gutenberg (2018), others were extracted from various sources, incl. Wikipedia (The Wikimedia Foundation, 2018a) and Wikisource (The Wikimedia Foundation, 2018b). All data sets, whose length varies between 255 and 459 characters, can be found in the above mentioned GitHub repository, including their sources. The semi-random texts, with a length of 255 to 425 characters, were created with a script that we wrote in Python (Python Software Foundation, 2018). The texts are semi-random in the sense that they observe English letter frequency. The corresponding  $|\Delta_H|$ 's for bigrams and trigrams are illustrated in Figure 4 (left).

We encrypt all true texts and pseudo-texts with an algorithm modelled after the encryption method used for z408, also using one of our Python scripts. Each letter is, partly depending on its frequency, pseudo-randomly mapped to one to five unique symbols, resulting in one-to-many ciphers with symbol sets of 52 to 66 symbols. The encryption script can also be found in the above mentioned GitHub repository. The  $|\Delta_H|$ 's in entropy for the one-to-many ciphers are illustrated in Figure 4 (right).

We also create many-to-many ciphers. The encryption algorithm for this has two layers. First, similar to the algorithm above, each letter is, partly depending on its frequency, pseudo-randomly mapped to one to five symbols. This is the first encryption layer. Then, the first layer is encrypted again: Each first layer symbol is pseudo-randomly mapped to one to four second layer symbols. The second mapping is not unique, in the sense that most second layer symbols map to more than one first layer symbol, resulting in a many-to-many cipher. The ciphers have second layer symbol sets of 60 to 64 symbols, which is similar in size to the symbol sets of the one-to-many ciphers.

The  $|\Delta_H|$ 's for the many-to-many ciphers are illustrated in Figure 5 (left). As Figure 5 (left) indicates, the many-to-many encryption algorithm is a

lot stronger than the one-to-many algorithm. True many-to-many ciphers and pseudo-ciphers overlap to some degree, illustrating that a very strong many-to-many cipher might be indistinguishable from a pseudo-cipher.

In a last step, we add the two ciphers by the Zodiac Killer to the picture. Their  $|\Delta_H|$ 's are illustrated in Figure 5 (right), including a comparison with our other ciphers. Z340 sits among the pseudo-ciphers. And compared to our encryption algorithms, z408 uses a fairly weak encryption technique. The latter is not a surprise. In their selection of symbols, humans are biased. For instance, if the mappings for "A" are "X" and "Ω", then a human might tend to choose "X" if e.g. "A" precedes an "N" but choose "Ω" if "A" precedes a "T". Our pseudo-random encryption algorithm has no such biases.

### 4.3 Training and testing support-vector machines for classification

We use the results from above to train three support-vector machines (SVMs), cf. Ben-Hur et al. (2002). One SVM for the plain texts vs the semi-random strings, one for the true one-to-many ciphers vs the one-to-many pseudo-ciphers, and one for the true many-to-many ciphers vs the many-to-many pseudo-ciphers. The one-to-many SVM and the many-to-many SVM are illustrated in Figure 6.

In a second step, we validate the SVMs on further test data sets. We use another 32 true plain texts and 32 semi-random strings, 32 true one-to-many ciphers and 32 one-to-many pseudo-ciphers, and 32 true many-to-many ciphers and 32 many-to-many pseudo-ciphers. This gives us a 50-50 training-testing split. The texts and ciphers were obtained in a similar fashion as described in Section 4.2. The results of the testing phase are given in Table 2.

## 5 Discussion

The SVM for plain texts makes excellent predictions, the one-to-many SVM also makes very good predictions. The many-to-many still makes good predictions, but with a somewhat lower accuracy than the other models, presumably because the encryption technique is rather strong.

According to the one-to-many model and the many-to-many model, z408 is classified as a true substitution cipher. z340, on the other hand, is

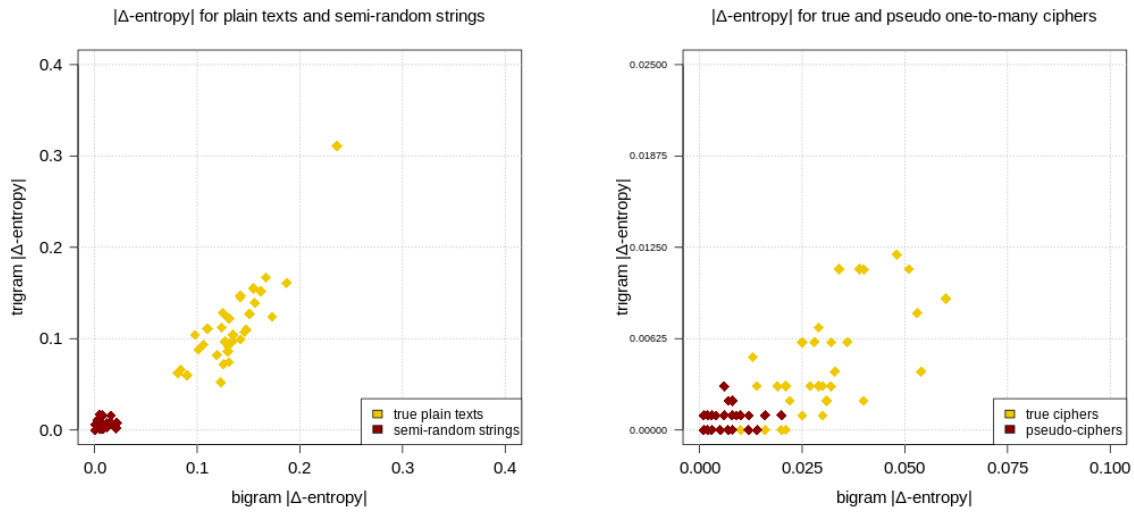


Figure 4: Left: The bigram  $|\Delta_H|$ 's, on the x-axis, and the trigram  $|\Delta_H|$ 's, on the y-axis, for the 32 plain texts and 32 semi-random texts in our training set. Right: The  $|\Delta_H|$ 's for the 32 true one-to-many ciphers and the 32 one-to-many pseudo-ciphers in our training set.

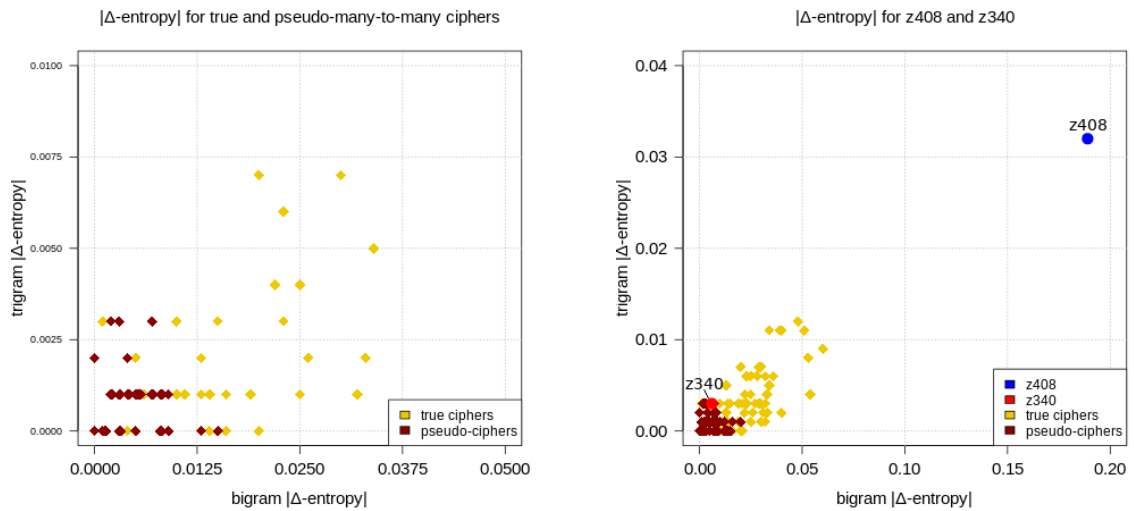


Figure 5: Left: The bigram  $|\Delta_H|$ 's, on the x-axis, and the trigram  $|\Delta_H|$ 's, on the y-axis, for the 32 true many-to-many ciphers and the 32 one-to-many pseudo-ciphers in our training set. Right: The  $|\Delta_H|$ 's for z408 and z340, with the results for the other ciphers for contextualisation.

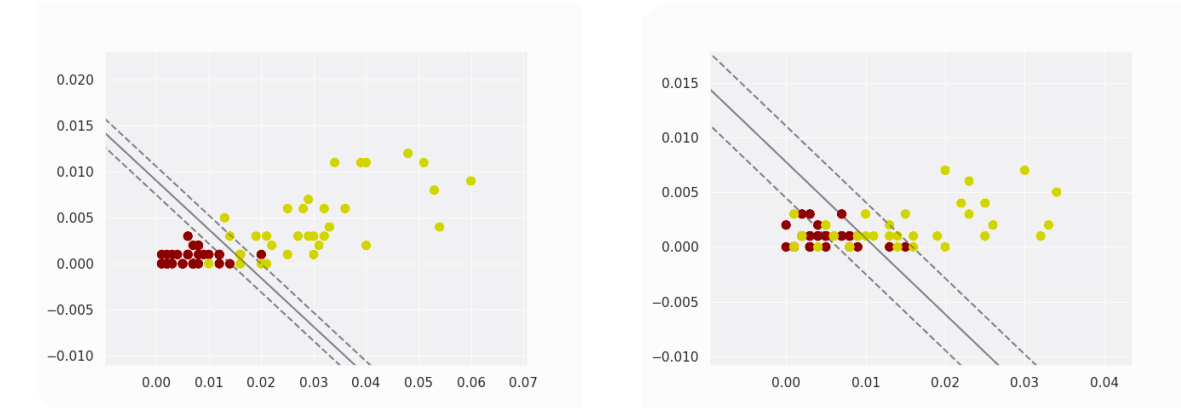


Figure 6: Left: The results of an SVM trained on the 64 one-to-many ciphers in our training data sets, 32 of which are true ciphers, the other 32 being pseudo-ciphers. The decision boundary is in grey. Items left of the boundary are predicted to be pseudo one-to-many ciphers, items right of it true one-to-many ciphers. Right: The same for the 64 many-to-many ciphers in our training set. Visualisation of the SVM was partly done with code from <https://jakevdp.github.io>.

<b>text</b>	predicted true	predicted pseudo
actual true	32	0
actual pseudo	0	32
$F_1 = 1.0$		

<b>o-t-m</b>	predicted true	predicted pseudo
actual true	30	2
actual pseudo	0	32
$F_1 = 0.96$		

<b>m-t-m</b>	predicted true	predicted pseudo
actual true	25	7
actual pseudo	6	26
$F_1 = 0.80$		

Table 2: Confusion matrices and F1-scores for the three support-vector machine models. Top: The model that classifies plain texts vs semi-random strings (*text*). Middle: The model that classifies true one-to-many ciphers vs one-to-many pseudo-ciphers (*o-t-m*). Bottom: The model that classifies true many-to-many ciphers vs many-to-many pseudo-ciphers (*m-t-m*).

extremely close to zero and both models predict that it is not a true substitution cipher. However, it should be noted that z340 sits relatively close to the decision boundary of the many-to-many model and that in some of the re-runs of the procedure, the many-to-many model places z340 right above the decision boundary.

Considering that z408 is a rather weak one-to-many cipher, we think that it is unlikely that the same author had been able to produce another substitution cipher, i.e. z340, such that its encryption mechanism became stronger by several orders of magnitude. We interpret this as evidence that z340 is either not a *bona fide* substitution cipher or uses a different, more sophisticated encryption mechanism altogether. For instance, it might be a transpose cipher or a Vigenère cipher.

There are a few things to note about the presented measure. First, the measure is not absolute. Consider for instance Kryptos Passage 4 by Jim Sanborn, also available on Wikipedia (The Wikimedia Foundation, 2018a). Here are the first three lines of Kryptos 4:

```

NGHIJLMNQUVWXZKRYPTOSABCDEFGHIJL
OHIJLMNQUVWXZKRYPTOSABCDEFGHIJL
PIJLMNQUVWXZKRYPTOSABCDEFGHIJLM

```

The entropy for Passage 4 comes out as 1.55 and the  $|\Delta_H|$  is 0.92. This, of course, does not mean that Passage 4 is a plain text or a substitution cipher. Upon visual inspection, it becomes immediately clear that neither is likely. This is a

limitation one has to keep in mind using entropy on potential ciphers.

Another issue is that there is no definite cut off point between strong many-to-many ciphers and pseudo-ciphers. Very strong many-to-many ciphers can become indistinguishable from pseudo-ciphers. However, this is a good reflection of the underlying reality: As the strength of the encryption mechanism increases, the probability of being able to make sense of it decreases.  $|\Delta_H|$  reflects this inverse relationship.

Finally, while we use SVMs, other analyses can be used as well. For instance, one could use a k-means clustering analysis, cf. Lloyd (1982) and Forgy (1965), in addition to an SVM.

## 6 Conclusion

We have shown that using differences in information theoretical entropy can be used to evaluate the authenticity of substitution ciphers. We created 64 true ciphers and another 64 pseudo-ciphers and split those into training and test data sets. We then trained and tested support-vector machines on our data sets. The model for one-to-many ciphers makes very good predictions, the model for many-to-many ciphers makes decent predictions. We applied those two SVM models to the the Zodiac Killer's two major ciphers, z408 and z340. z408, which has been solved, is correctly predicted to be a real substitution cipher. z340, which remains unsolved, is predicted to not be a substitution cipher. We think that it is likely that z340 is either another type of cipher, e.g. a transpose cipher or a Vigenère cipher, or that it might not be a *bona fide* substitution cipher after all.

As a next step, it would be interesting to apply the measure to other, unsolved ciphers. Also, ideally, further measures could be developed for other types of ciphers, like transposition ciphers or Vigenère ciphers.

## Acknowledgments

Many thanks to asetniop, Jarl, K. L., Sasja, and the anonymous reviewers for their valuable feedback.

## References

- Asa Ben-Hur, David Horn, Hava T. Siegelmann, and Vladimir Vapnik. 2002. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137.
- Eric Corlett and Gerald Penn. 2010. An exact a\* method for deciphering letter-substitution ciphers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1040–1047, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Edward W. Forgy. 1965. Cluster analysis of multivariate data: Efficiency versus interpretability of classification. *Biometrics*, 21(3):768–769.
- Robert Graysmith. 1987. *Zodiac*. Berkley, New York City, NY.
- Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. 2008. *An Introduction to Mathematical Cryptography*. Springer Publishing Company, Incorporated, 1 edition.
- Dan Jurafsky, 2019. *Lecture CS 124: From Languages to Information. Introduction to N-grams – lecture notes*. <http://web.stanford.edu/class/cs124/>.
- Kevin Knight, 2013. *Decipherment Tutorial. Workshop at the 2013 Annual Meeting of the Association for Computational Linguistics – lecture notes*. <https://kevincrawfordknight.github.io/extra/acl-tutorial-13-decipher-final.pdf>.
- Stuart P. Lloyd. 1982. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137.
- Project Gutenberg, 2018. *Project Gutenberg*. <http://www.gutenberg.org>.
- Python Software Foundation, 2018. *Python: A dynamic, open source programming language*. <https://www.python.org/>.
- Sujith Ravi and Kevin Knight. 2008. Attacking decipherment problems optimally with low-order n-gram models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 812–819, Stroudsburg, PA, USA. Association for Computational Linguistics.
- 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*, HLT '11, pages 239–247, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 7.
- The Wikimedia Foundation, 2018a. *Wikipedia, the free encyclopedia*. <https://wikipedia.org/>.
- The Wikimedia Foundation, 2018b. *Wikisource, the free library*. [https://en.wikisource.org/wiki/Author:Zodiac\\_Killer#Letters](https://en.wikisource.org/wiki/Author:Zodiac_Killer#Letters).

Zodiackillerciphers.com, 2012. *Annotated solution to the 408 cipher; based on the Harden worksheets.* <http://zodiackillerciphers.com/408/key.html#1>.



## **Published by:**

---

**NEALT** Proceedings series volume 37

Linköping University Electronic Press, Sweden

Linköping Electronic Conference Proceedings 158

ISSN: 1650-3686

eISSN: 1650-3740

ISBN: 978-91-7685-087-9