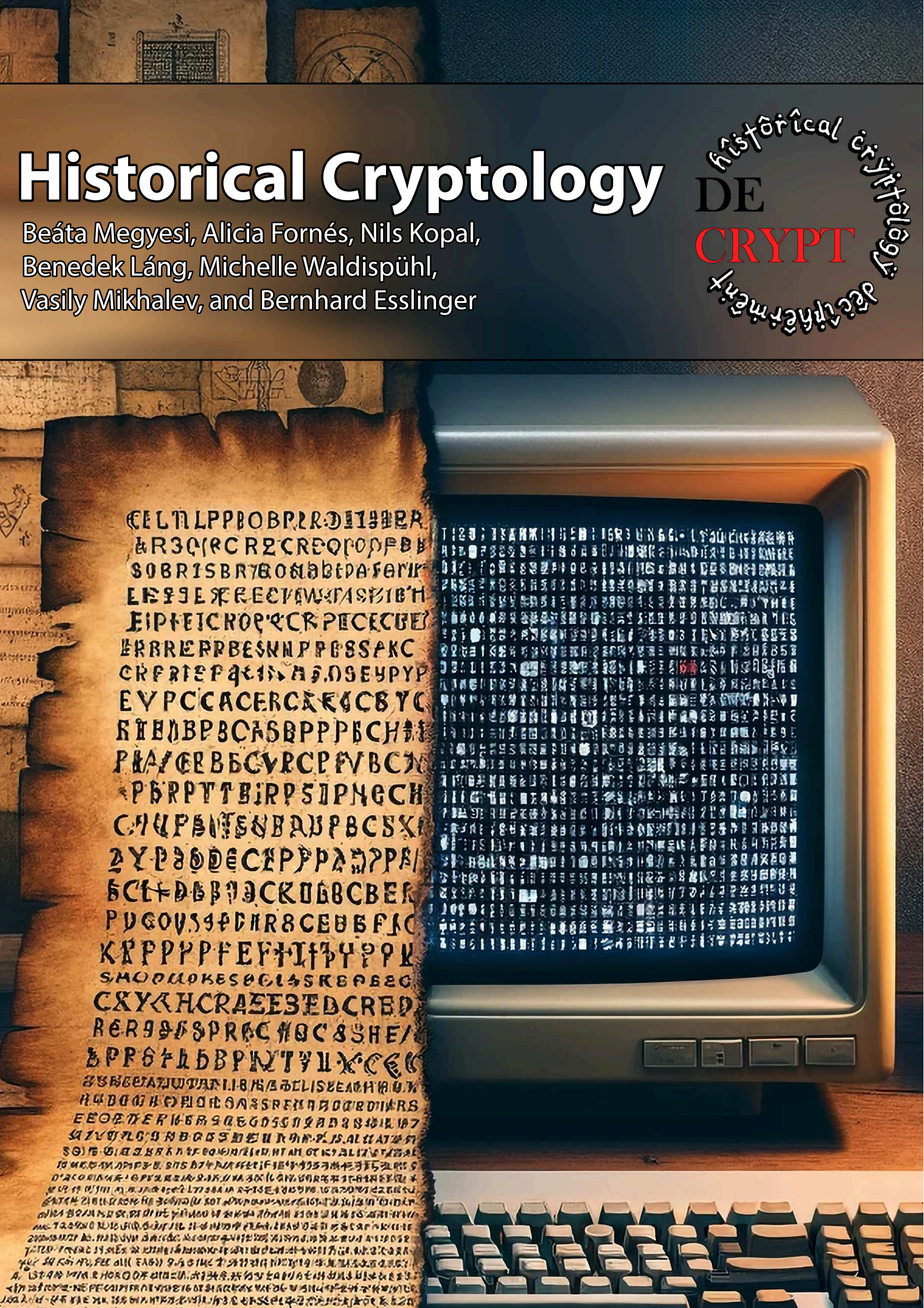
Historical Cryptology

Beáta Megyesi, Alicia Fornés, Nils Kopal,
Benedek Láng, Michelle Waldispühl,
Vasily Mikhalev, and Bernhard Esslinger



LEARNING AND EXPERIENCING CRYPTOGRAPHY WITH CRYPTOOL AND SAGEMATH

Bernhard Esslinger

Chapter 3

Learning and Experiencing Cryptography with CrypTool and SageMath

For a listing of recent titles in the *Artech Computer Security Library*,
turn to the back of this book.

Learning and Experiencing Cryptography with CrypTool and SageMath

Bernhard Esslinger

ARTECH HOUSE
BOSTON | LONDON
artechhouse.com

Library of Congress Cataloging-in-Publication Data
A catalog record of this book is available from the U.S. Library of Congress.

British Library Cataloguing in Publication Data
A catalog record for this book is available from the British British Library.

ISBN 978-1-68569-017-5

Cover design by Joi Garron

Accompanying software for this book can be found at:
https://www.cryptool.org/en/documentation/ctbook.

685 Canton Street
Norwood, MA 02062

10 9 8 7 6 5 4 3 2 1

Contents

CHAPTER 3

Historical Cryptology

Historical cryptology studies (original) encrypted manuscripts, often handwritten sources, produced in our history. These historical sources can be found in archives, often hidden without any indexing and therefore hard to locate. Once found they need to be digitized and turned into a machine-readable text format before they can be deciphered with computational methods. The focus of historical cryptology is not primarily the development of sophisticated algorithms for decipherment, but rather the entire process of analysis of the encrypted source from collection and digitization to transcription and decryption. The process also includes the interpretation and contextualization of the message set in its historical context. There are many challenges on the way, such as mistakes made by the scribe, errors made by the transcriber, damaged pages, handwriting styles that are difficult to interpret, historical languages from various time periods, and hidden underlying language of the message. Ciphertexts vary greatly in terms of their code system and symbol sets used with more or less distinguishable symbols. Ciphertexts can be embedded in clearly written text, or shorter or longer sequences of cleartext can be embedded in the ciphertext. The ciphers used mostly in historical times are substitutions (simple, homophonic, or polyphonic), with or without nomenclatures, encoded as digits or symbol sequences, with or without spaces.

So the circumstances are different from those in modern cryptography which focuses on methods (algorithms) and their strengths and assumes that the algorithm is applied correctly. For both historical and modern cryptology, attack vectors outside the algorithm are applied like implementation flaws and side-channel attacks.

In this chapter, we give an introduction to the field of historical cryptology and present an overview of how researchers today process historical encrypted sources.

3.1 Introduction

Historical cryptology deals with the encryption and decryption of historical, manually constructed ciphers. An encrypted source usually counts as historical if it has been produced no later than the mid-20th century. There is no exact break-even point; however, the development of telegraphy (from the 1830s) led to more sophisticated and complex mathematical methods applied to encryption requiring more advanced cryptanalysis.

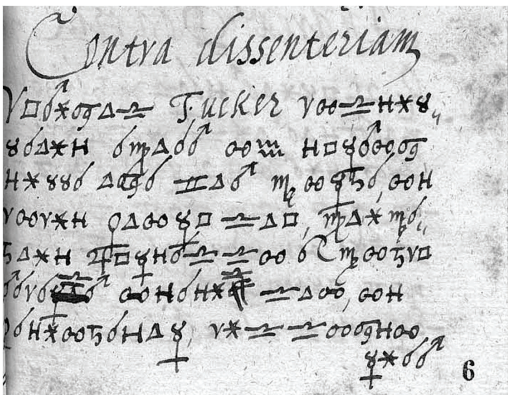
Historical cryptology involves the field of cryptography (the art and science of code making and the encryption of messages), and the field of cryptanalysis (the art and science of code breaking [1], i.e., the decipherment of messages without the

key). In everyday language, the terms "cryptography" and "cryptology" are often used interchangeably.

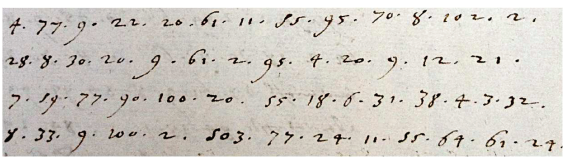
As in all scientific fields, historical cryptology has its own terminology. We start the chapter by introducing the most important terms and give a brief overview of historical ciphers and keys before we move on to the components needed to process and decipher historical ciphers. Given that historical cryptology as a scientific field of study is rather new, the terminology standards and the usage of the terms are still to be established and under discussion in the historical cryptology community (see for example [2] and [3]). We summarize the important terms in Figure 3.6 and Table 3.1, as well as illustrate the crypto process in Figure 3.7.

A cipher (sometimes also "cypher," which is simply the old spelling) refers to an algorithm that describes the procedure of encryption or decryption. The encrypted source itself is called ciphertext, though the terms "code" and "cipher" are often not distinguished from ciphertext in everyday language. A ciphertext consists of a sequence of symbols from a ciphertext alphabet. The ciphertext alphabet can be the same as the plaintext alphabet (e.g., the Latin letters), but often it consists of different symbol systems and alphabets, such as Greek letters, digits, graphic signs (e.g., alchemical or zodiac signs), or Chinese hieroglyphs. Figure 3.1 illustrates the variation of the symbol systems from ciphertext alphabets in three ciphertexts. The ciphertexts are extracts taken from the Borg cipher [4], a digit-based cipher from the National Archives of Sweden [5], and the Copiale cipher [6].

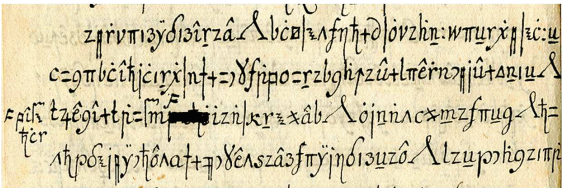
In ciphertexts, we can find regular usage of space marking word boundaries as in the Borg cipher (see Figure 3.1) even though most of the ciphertexts from the past use continuous script (scriptio continua) without any spaces, as shown in the examples from the Swedish National Archives and the Copiale cipher. Word boundaries were often removed in historical ciphers to make codebreaking more difficult. A ciphertext might also contain additional information such as accents and other diacritics, or punctuation marks appearing more or less systematically in connection to symbols, as in the example of the Copiale cipher. We can also find overwritings for corrections, underlined sequences, and unintentional ink spots in the manuscripts.



The Borg cipher



Cipher from the Swedish National Archive Riksarkivet



The Copiale cipher

Figure 3.1 Three examples of ciphertexts. (*From*: [4–6].)

Apart from the ciphertext in an encrypted message, nonencrypted sequences of texts that we call cleartext are also common. For example, the first line of the Borg cipher in Figure 3.1 contains a cleartext "Contra dissenteriam" in Latin. Cleartext passages bear important information about the possible underlying language(s) and the topic of the encrypted source.

The encrypted source might also include decrypted plaintext, written on the same page, often found above the ciphertext lines, as illustrated in Figure 3.2.

A cipher key (Figure 3.3) using a given cipher defines how to encrypt a plaintext and how to decrypt a ciphertext. Historical cipher keys usually contain a list of plaintext elements (letters, syllables, words, names, phrases) and the corresponding symbol or combination of symbols taken from the ciphertext alphabet, henceforth the code elements. Two examples of cipher keys are shown in Figures 3.3 and 3.4.

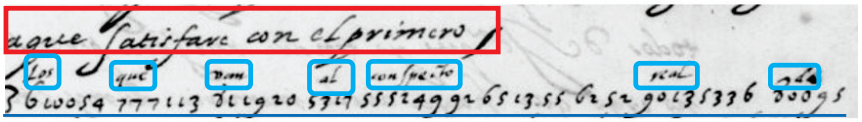


Figure 3.2 Ciphertext (underlined), cleartext (in red), and plaintext (in blue) in an encrypted manuscript.

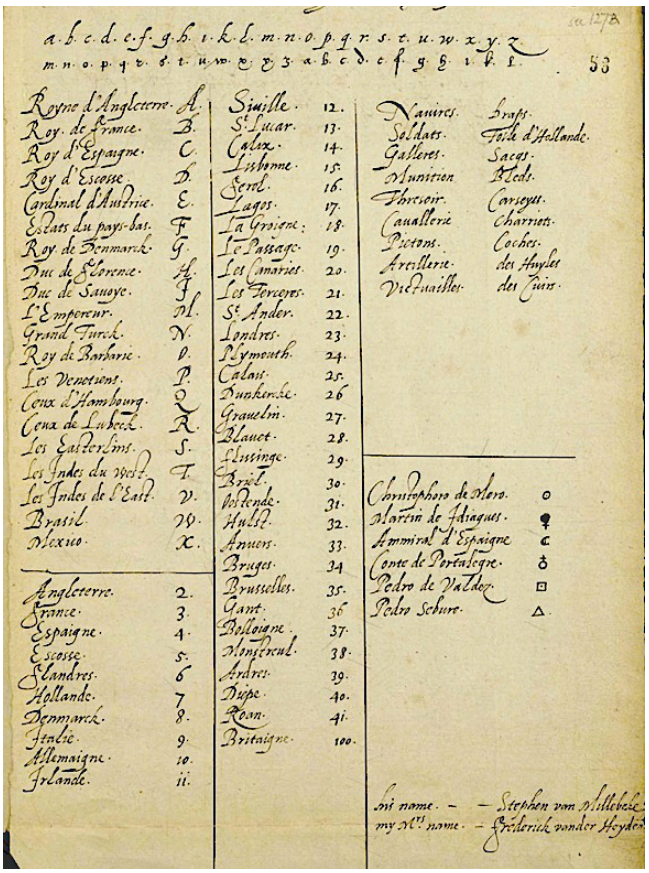


Figure 3.3 Cipher key: simple substitution. (Flanders, 1596 [7].)

In both keys, the letters of the plaintext alphabet (A–Z) are listed horizontally in the first line of the key tables. Moreover, underneath each plaintext letter, we can find either one (Figure 3.3) or several ciphertext symbols (Figure 3.4), henceforth alphabet-code elements, assigned to each plaintext letter. In Figure 3.3 these single ciphertext letters are taken from the plaintext alphabet but in a different position. In Figure 3.4, on the other hand, the lengths of the alphabet-code elements vary; two-digit code elements to encode the plaintext alphabet and three-digit code elements to encode the words. Note that the most frequently occurring plaintext alphabet letters have four alphabet-code elements, whereas the least frequent ones received three code elements. Adding several code elements to the frequently occurring plaintext elements leads to an increased difficulty of decipherment and renders a cipher homophonic.

In the columns of both keys we find a shorter or longer list of plaintext elements (names, content, and function words) with code elements assigned to each. Such a list as part of the key is called nomenclature, sometimes also spelled nomenclator. Sometimes the entire key that contains a nomenclature (i.e., a list of plaintext elements) is called a nomenclator. Here, we make a distinction between the various parts of the key. The nomenclature shown in Figure 3.3 consists of roughly 100 items in which we can see code elements using a single ciphertext symbol, for example "A" for "Royne d'Angleterre" and others with multiple ciphertext symbols, such as "12" for "Siuille." Here, the various types of nomenclature elements receive

Figure 3.4 Cipher key: simple and homophonic substitution. (Hungary, 1703–1711 [8].)

different cipher symbol types: Personal names are encoded by capital letters, place names by numerals, military titles by other words, and dignitaries by graphic signs. However, this assignment is not fully consistent: "Brazil" and "Mexico" are listed among the personal names. Such inconsistencies are not uncommon in historical encrypted sources. In the other key in Figure 3.4, the nomenclature is larger, consisting of over 400 entities. Here, we can find syllables shown as section headings ("Ba," "Ca," "Da," ...), function and content words, and names and phrases, all in French. The last column contains additional information about the key to give instructions or details about the cipher.

Historical cipher keys were typically structured as tables, in which the alphabet elements and the nomenclature elements were graphically clearly separated; the former horizontally as lines and the latter vertically as columns. Content-wise, however, the boundary is not as clear-cut; double letters, syllables, or function words might be listed as part of the alphabet line. It is also noteworthy that the nomenclature tables usually have a certain structure in which plaintext elements can be ordered alphabetically (see the key in Figure 3.4) or thematically (as shown in Figure 3.3), or in a combination where the words in the themes can be alphabetically ordered. In turn, the code elements can be grouped thematically depending on the type of plaintext element they encode (as in Figure 3.3), and/or numerically when the code elements are represented by digits. The key creators often assigned code elements to the alphabetically or thematically listed plaintext elements in some structure. Code elements of the nomenclature list were typically numbered consecutively in increasing or decreasing order, either vertically following the order of the columns or horizontally, following the lines across the columns. The construction of the nomenclature list has an impact on the cryptanalysis (decipherment)—alphabetical order of the plaintext elements with increasing order of numbers can ease cryptanalysis as higher code numbers represent words starting with letters at the end of the alphabet.

To make cryptanalysis more difficult, operational code elements (i.e., code elements that operate either on the plaintext or on other code elements) have been used. A commonly occurring type are nulls, which can also be named in historical cipher keys as nullities and called by the public as "blenders"—fake code elements that encode an empty string in the plaintext. Note that keys might also contain code elements without any given plaintext in the nomenclature table treated as placeholders to be filled in later, which are not defined as nulls but empty code elements. Other types of operational code elements with special function on the plaintext include cancellation signs (also called nullifiers or deleters) that mark the removal of a certain sequence of ciphertext, and repetition signs that repeat the preceding symbol used for the reduplication of a plaintext letter.

Historical cipher keys changed and developed over time leading to the emergence of new ciphers. In fact, all the historical ciphers discussed in this chapter are variations of the substitution cipher. The specific substitution method was entirely determined by the key type used with it. Therefore, when we discuss the development of the keys, we also speak about the evolution of the ciphers. The earliest keys in Europe were based on simple substitution, in which each plaintext element is assigned to exactly one code element represented as a ciphertext symbol. An example of a simple substitution cipher is shown in Figure 3.3. The top two lines

of this document illustrate a nice example of the Caesar cipher (see Section 2.2.1), in which the plaintext alphabet is also used for encryption but shifted (here by 11 positions). To complicate the cryptanalysis, a nomenclature table was added, which became the norm in Europe in the 15th century [9]. Simple substitution ciphers were then further developed into homophonic substitution ciphers, where the same plaintext entities—often the most frequently occurring ones, such as vowels and some consonants—could be encrypted with different code elements, as illustrated in Figure 3.4. The nomenclature list evolved from the 17th century and onward from several hundred elements to thick codebooks, in which not only content words but also grammatical categories (e.g., singular, plural; grammatical cases) or inflected word forms (e.g., "see, sees, saw, seen" for the verb "to see") were listed with their own code elements [9]. In some keys, different plaintext entities could also be assigned to the same code element, intentionally or unintentionally. Ciphers with one code element assigned to several plaintext symbols are called polyphonic substitution ciphers. Figure 3.5 illustrates such a cipher key. Here, the ciphertext symbol "3" can be decrypted as either "A" or "s," and the symbol "6" as either "t" or "r."

The three types of encryption methods—simple, homophonic, and polyphonic—are the most frequently occurring types in European history [9]. The interested reader can find more details about the structure and evolution of cipher keys throughout the centuries in Europe in [9].

In addition, not only monoalphabetic substitution ciphers have been used throughout history. After the early modern time, polyalphabetic substitution ciphers became common, such as the Vigenère cipher (see Section 2.2.4). In these ciphers, the plaintext alphabet is mapped to different ciphertext alphabets—see Section 2.2.4. Transposition ciphers (Section 2.1) are another type, in which the letters of the plaintext are switched around in some systematic way to form the ciphertext. In later centuries, we can also find ciphers that are actually cascades of different ciphers that we call composed ciphers. An example of such a cipher is the ADFGVX cipher [10], which is a combination of substitution (using a Polybius square—see Section 2.3) and (columnar) transposition.

In recent years, by far the greatest attention worldwide for historical cryptology has been given to the successful cryptanalysis of over 50 newly discovered letters written by Mary Stuart between 1578 and 1584. George Lasry, Norbert Biermann, and Satoshi Tomokiyo worked for over one year to transcribe, decipher, and place these letters containing over 150,000 symbols in their proper historical context [11]. Mary Stuart's letters were classified under Italian letters in the French National Library, without telling sender or recipient or the actual language used (French). The procedure used by Mary Stuart was a difficult cipher because she used a nomenclature with 191 different characters, which included well over 100 words in addition to the 26 letters of the alphabet, but also homophones (several symbols representing the same letter), symbols without meaning (nulls or blenders),

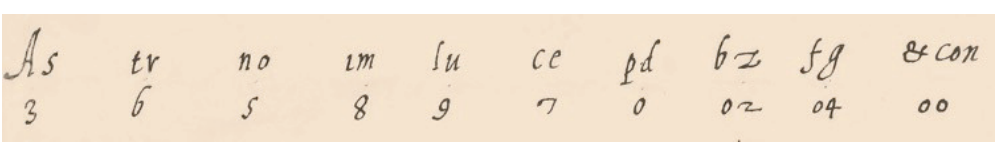


Figure 3.5 Cipher key example: polyphonic substitution from the 16th century.

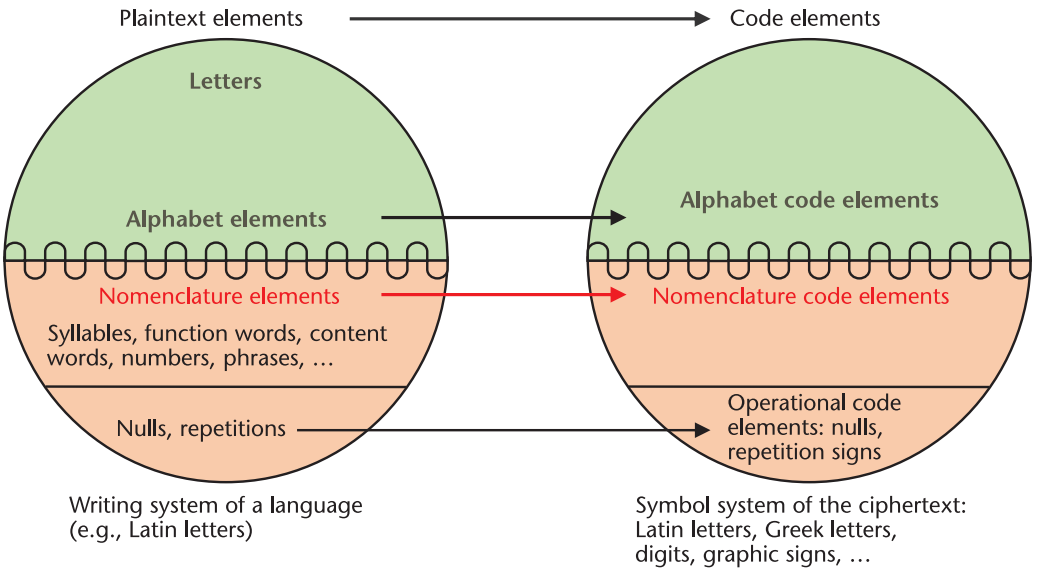


Figure 3.6 Terminology: Mapping of important terms.

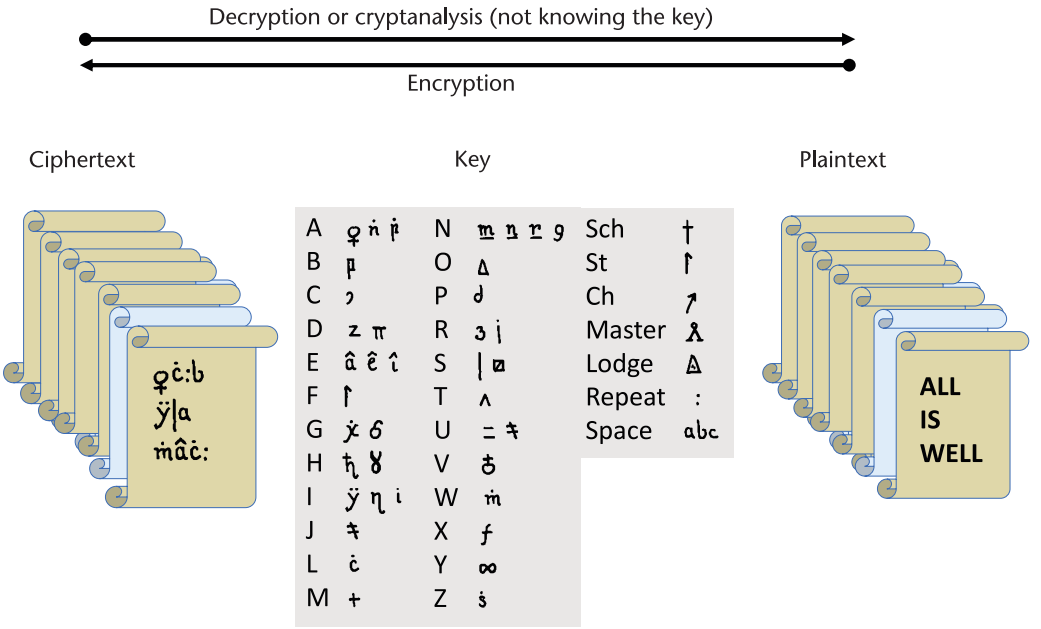


Figure 3.7 The crypto process: Components of encryption and decryption of historical sources.

symbols that cancel the previous symbol (nullifier), and symbols that repeat the previous symbol.

3.2 Analyzing Historical Ciphers: From Collection to Interpretation

Next, we describe the components involved in the processing and analysis of historical encrypted sources.

Historical ciphertexts are handwritten or printed manuscripts buried in archives, libraries, museums, or private collections. They might be difficult to find as they are hardly indexed as ciphers in archive or library catalogs. Only a small but increasing percentage of the historical encrypted sources are digitized and made available online, and even fewer are turned into a computer-readable text format. Finding, analyzing, and deciphering encrypted manuscripts are challenging and need various kinds of expertise. In this section, we give a bird's-eye view on the different steps and components involved in processing encrypted manuscripts from collection through transcription to decipherment, as illustrated in Figure 3.9. Then we describe each step of the process in detail in the subsequent sections.

Collecting encrypted sources requires knowledge about the whereabouts of the documents. Once found, the documents need to be digitized, turned into images, and described with a set of metadata according to some standard. Information can include the sender and receiver of the documents, the time and place when the encrypted source was produced or sent, and a description of its content. Describing historical sources in terms of metadata is as important as the content of the document itself.

Before we can cryptanalyze a ciphertext, we usually need to transcribe it (i.e., turning the ciphertext image(s) into a computer-readable text format). By doing so, we look closely at the symbol set and group the similar ciphertext symbols into types, which helps us in the identification of the entire ciphertext alphabet. A transcription is a text representing the ciphertext symbols from the image(s) symbol by symbol and line by line. This requires interpreting the handwriting style and motion educated guesses about the intentions of the scribe; in other words to interpret the handwriting. The transcription needs to be thorough; all symbols, diacritics, punctuation marks, and spaces must be transcribed to avoid error propagation during decipherment.

Given a (couple of lines of) transcription we can go on with the cryptanalysis. First, we need to segment the ciphertext into code elements and analyze the frequencies and co-occurrences of the various symbol types and code elements. We need to make educated guesses about the cipher type and about the underlying language. Dictionaries and language models for various time periods might be of help on the way when guessing the plaintext underneath. Once we have a decrypted text, we interpret the plaintext, correct wrongly transcribed symbols, and adjust the assumed key to get an appropriate and reasonable plaintext output. We might then translate the text to one or several languages, and set the plaintext in a historical context; what was written, by whom, to whom, and why.

Deciphering a ciphertext—albeit lots of fun—is often challenging. In the past, many historians and people worked individually in an uncoordinated fashion on the identification and deciphering of secret writings. Without access to automatic methods that can accelerate the decipherment, it's a time-consuming process. At the same time, cryptanalysts, computer scientists, and computational linguists develop automatic cryptanalysis algorithms to identify cipher types and to break various ciphers without having access to real historical ciphertexts.

To coordinate the efforts of various expertise and build research infrastructure in terms of resources and tools for historical cryptology, an international research program was created in 2018: the DECRYPT project [12]. The aim of the project

was to establish a new cross-disciplinary scientific field of historical cryptology by bringing the expertise of the different disciplines together to digitize encrypted historical documents, build a database of historical ciphers, and develop software tools for transcription and cryptanalysis. We are not aware of any other cross-disciplinary project in the field that takes a holistic approach from collection through transcription to decipherment by developing open-source resources and tools for historical cryptology in large scale. Therefore, we base this chapter largely on the experiences and findings of the results of the cross-disciplinary cooperation in the DECRYPT project. However, there are many relevant high-quality studies on various aspects of historical cryptology and we will refer to the most prominent ones in the relevant parts of the subsequent sections.

To be able to study the characteristics of historical ciphers with the ultimate goal to decipher all cipher types from historical times, we need a large set of historical sources to be collected and stored from various places and time periods. The DECODE database [13] was created to store images of ciphertexts, encryption keys, and information about their provenance, transcriptions, and possible decryptions. The process of (semi-) automatic decryption involves, as mentioned before, transcription by applying image recognition to automatically convert the images to machine-readable format and a mapping of symbols to a transcription scheme. The detection of the underlying plaintext language of the ciphertext on the basis of historical text sources, the automatic identification of the cipher type, the cryptanalysis of the ciphertext, and finally its decryption are taken care of in the cryptanalysis step. The methods developed are based on a wide range of algorithms: from classical cryptanalysis to advanced deep-learning architectures taken from artificial intelligence. Various (neural network) models for transcription are released in the TranscriptTool [14] (see Figure 3.8), while the algorithms for cryptanalysis have been implemented in CrypTool 2 (CT2) [15]. CT2 is called in a command line version on the webserver of the DECRYPT pipeline. Both tools (TranscriptTool

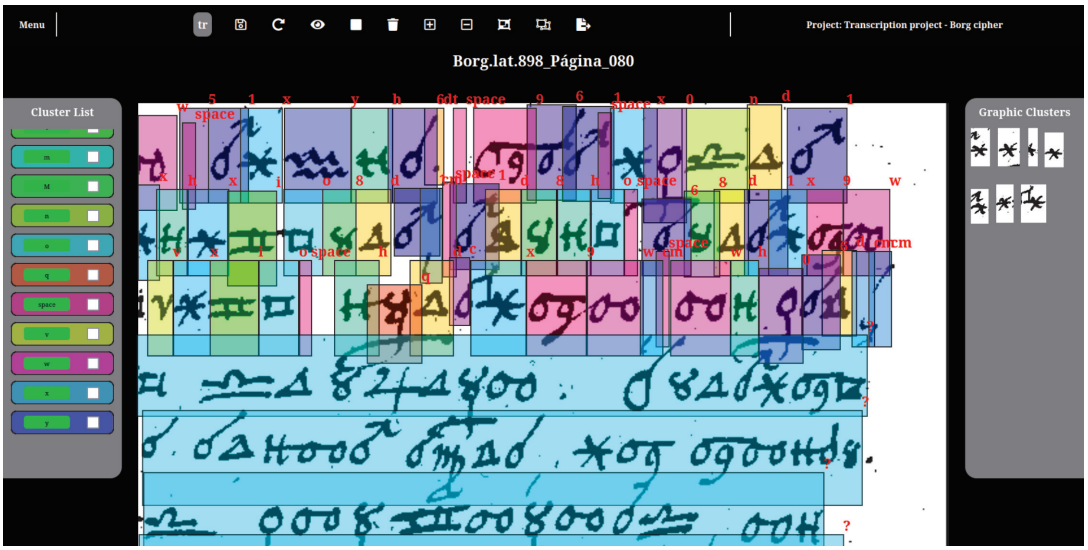


Figure 3.8 TranscriptTool for creating transcriptions of scanned historical manuscripts, offered as part of the DECRYPT pipeline.

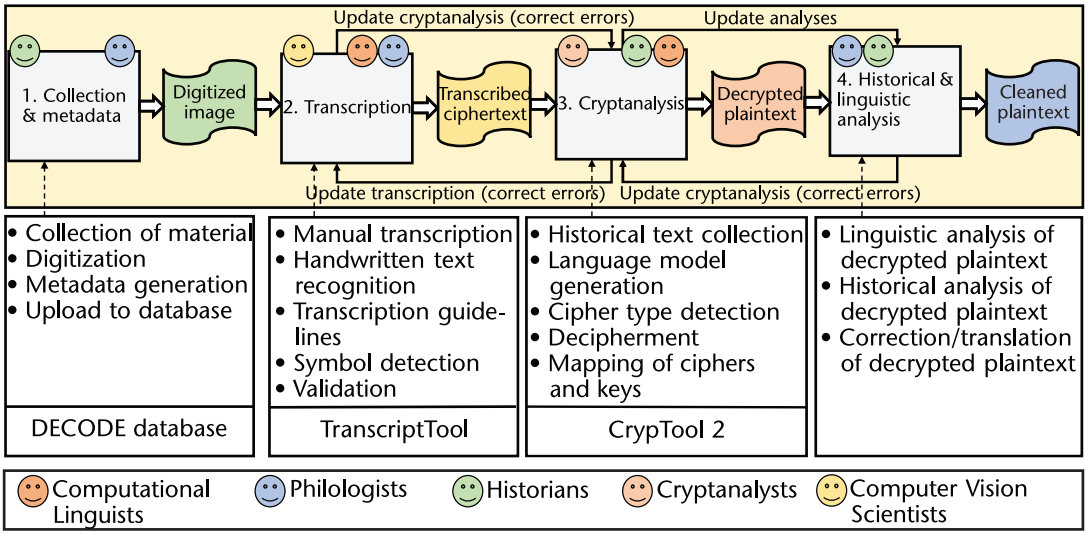


Figure 3.9 Overview of the DECRYPT pipeline (see `https://de-crypt.org/`).

and CT2) are released as open-source and are under continued development (as of 2023). The DECODE database and the two tools are included into a framework as a pipeline for processing the historical encrypted manuscripts to allow feedback loops and error reduction between the various steps in the pipeline. In addition to the TranscriptTool in the pipeline on the web, there is a standalone offline tool called CTTS. See Section 3.4.2. For ciphers that do not consist of numbers, CTTS or TranscriptTool are currently the best choice. For numeric ciphers, Transkribus.ai can be an alternative.

The steps for breaking a cipher need careful combination and cooperation of experts from different fields. Computational linguists provide the database with keys and ciphers, define transcription schemes for various symbol sets, and build and evaluate historical language models generated from historical texts. Historical linguists and philologists collect and analyze historical texts to develop models for language variation and language change. Cryptanalysts develop efficient algorithms for the cryptanalysis of various cipher types, and computer vision scientists provide a typology of symbol transcription and models to turn images into a machine-readable format. Historians contribute to the collection, contextualization, and interpretation of the hidden sources. By doing so the encrypted sources can be systematically handled, studied in large scale, and made available to the public.

The following sections describe the main parts shown in the pipeline and highlight the challenges in each step.

3.3 Collection of Manuscripts and Creation of Metadata

A general experience of experts looking for handwritten cipher keys and encrypted documents is that they are easy to recognize but hard to find (see Section 3.2). It is easy to recognize the keys because they have a typical structure: A plaintext alphabet and a ciphertext alphabet are written next to each other, often followed

by a nomenclature table where words and corresponding code elements are listed. A typical historical key usually looks like a short note on a piece of paper (if it is a monoalphabetic cipher) or a large table on one or two approximately A4-measure pages. They are either separate sheets or part of an extensive collection, with pages in a book entirely dedicated to cipher keys. The encrypted documents are usually easy to *recognize* because they are text-like documents partially or entirely composed of numbers, letters, or graphic signs, often separated by dots. Even though sometimes inventories are mistaken for encrypted documents, and there might be some uncertainty about whether a text is encrypted or written in an unknown writing system or language, most of the time these documents are recognized without any problem. They might be only a few words, a paragraph-long ciphertext in an otherwise readable message, or a several-page (even a book-length) entirely encrypted document.

However, it is not easy to *find* the encrypted sources. Cipher keys and encrypted documents are found in two different places: in the archives and the manuscript collections of libraries. Imagine that a crypto-history expert pays a visit in a foreign country wishing to study that area's cryptology. Such a research trip should be thoroughly prepared because entering an archive and asking for cipher keys without any preparation rarely leads to success. This preparation includes consulting the secondary literature using that specific manuscript collection and writing directly to the archivists/librarians. Asking for advice from historians dealing with the period (but not necessarily with encrypted documents) might also be of considerable help. The importance of personal contacts is not to be underestimated. Finally, precious input can also arrive from blog authors, including the portal about the Voynich manuscript by René Zandbergen [16], Nick Pelling's *Cipher Mysteries* [17], or Klaus Schmeh's science blog [18] with a wide range of encrypted sources.

Manuscript collections in libraries usually have proper catalogs, but the reference materials of archives do not always specify that a given source is encrypted. Even when thoroughly cataloged, their description is rarely on document-level; they remain more frequently on a higher collection level, and thus individual documents remain invisible. Archives usually have boxes with a lot of documents in them. Often, the box is described (e.g., political documents from this or that war), but the individual letters, or documents, are not described one by one. However, even in those rare cases when the indexes list each individual record, a further problem arises: which search word to look for? "encrypted," "cipher," "in cifra" (or ciffra), "enchiffré," "crypté," and "chiffriert" are certainly good choices, but following the results of "en chiffre" in the Bibliothèque Nationale de Paris might be problematic, because one gets thousands of documents, the description of which involves "number" (chiffre).

Usually, it is easier to find the keys because they are often stored together in thematic collections. The two most frequent cases are (1) a whole handwritten book (either in a library or an archive) in which cipher keys are copied, contains one key per page, and (2) a folder (usually in an archive) stores separate sheets of various sizes, one key being on each sheet. Catalogs and reference books usually mention such collections. However, when an individual key occurs somewhere alone, it is hardly mentioned and can only be found by chance.

Encrypted documents are harder to find because the catalogs (of the libraries) and the reference books (of the archives) often do not specify in the indices that they are entirely or partially encrypted. In such cases, the crypto historian can ask for diplomatic or military correspondences of a specific period in general. Diplomatic letters (particularly ambassadors' letters and intelligence reports) and military messages will include encrypted messages with high probability. Even family collections (the kind of sources that make up a large portion of the totality of archival collections) might also contain encrypted documents, not to mention personal diaries and scientific and religious books. There is no systematic way to find them; one has to ask for whole folders and leaf them through. According to the conjecture of a crypto historian, *one percent* of the archival material is partly or entirely encrypted [19].

There is also a problem of matching the encrypted document with the corresponding key. Even if the collectors found both, it is not evident that they recognize the relationship between the two. This task gets harder as the collections grow. It is tough to index the records in a way that corresponding sources become identifiable.

Once crypto historians find cipher keys and encrypted documents, they face several further difficulties. First, the attached metadata might not be correct. The collections are dated, and the origins of the sources are also indicated in the archival folders; however, this information is usually too broad, and the documents and the keys are not dated separately. Some of the records contain dates and names, and in those cases when these are not later additions (by 19th-century archivists and librarians, for example) but historical data, they are reliable. In other cases, they are not always trustworthy, or just contain information that is too unspecific.

Describing a manuscript in terms of its location, structure, origin, and content is invaluable for research. Such descriptions are called metadata, which help us to interpret the manuscript. The more robust and detailed the description is the more accurate analysis we can carry out. Metadata of historical encrypted sources might include—albeit not limited to—information about:

1. The current location of the manuscript (index number in the archive/library, place, city, country).
2. The origin of the document including information about the place and dating, the sender and the receiver of the source, or the creator and/or the user of the cipher key.
3. The content of the document including its type (e.g., a ciphertext, a cipher key, or a manual about cryptology), and the language(s) involved.
4. Additional information might describe the symbol set of the ciphertext alphabet (e.g., digits, alphabets, graphic signs), the cipher type (simple, homophonic, or polyphonic substitution), the nature of nomenclature elements, or instructions.

Unfortunately, such metadata for encrypted sources is difficult to find in the archives and libraries, as they are hardly indexed and only a few know about their whereabouts. As a result of this—hardly operationalizable—process several online collections are available that also offer digital scans. Besides the blog authors already mentioned, Satoshi Tomokiyo's private homepage *Cryptiana* [20] contains original ciphers and keys from the 15th to the 20th centuries and also helpful material on the

cryptanalysis of historical ciphers. Eugen Antal and Pavol Zajac's *Portal of Historical Ciphers* [21] hosts a yet small but growing database of original historical ciphers from the 17th up to the 20th century focusing on Central-European encrypted sources released with a nice graphical interface. And finally, being part of the DECRYPT project, the DECODE database [13] is the largest source for historical ciphers and keys today. At the time of writing (November 2023), the database contains over 7,000 historical encrypted sources, all stored with their original image(s) and annotated with metadata along with related documents such as transcriptions.

All collections of encrypted sources face two difficulties, one legal and one technical. First, the owner of the given records (let them be archives or libraries) usually does not allow making public high-resolution images in the online collection for copyright reasons. Thus, often only a low-resolution reproduction can be shared with the public. Second, visual recognition software requires good quality high-resolution (at least 300 DPI) copies. However, there has been considerable improvement in this second field, and thus sufficiently readable documents can be offered to the transcription tool, the next phase of the pipeline.

3.4 Transcription

Once collected, the images of the encrypted source must be turned into some computer-readable text format needed for the cryptanalysis part of the process. The digitization involves the conversion of the ciphertext as well as cleartext and/or plaintext passages appearing in the manuscript into a text representation. This means in particular that the symbols of the ciphertext in the images are replaced by machine-readable symbols and the cleartext and plaintext sequences are interpreted and transcribed. There are different methods and approaches how this can be done. In the following, we focus on the transcription of ciphertext and describe two methods: a manual option and a semiautomatic option. While the manual option relies entirely on human effort, the semiautomatic option uses computer-vision technology based on artificial intelligence (AI) methods followed by manual postcorrection of the AI output. We show the challenges with both methods and discuss their advantages and disadvantages in the last section.

3.4.1 Manual Transcription

Transcribing a historical source, especially those that are handwritten in a foreign language, is far from easy and needs trained eyes and hands. Here, the main challenges, standards, and current practices are summarized when transcribing encrypted sources.

The aim of the transcription is to convert the text appearing in the image into a text representation. The transcription of the historical document should be as accurate as possible. This concerns of course the delimitation of the distinct ciphertext symbols and the identification of the symbol types that appear in the manuscript. Sometimes it is an easy task if the ciphertext alphabet consists of a limited, known set of symbols such as digits. Oftentimes, the encrypted sources also contain other symbols such as dots, punctuation marks, accents and other diacritic signs, or underlined sequences.

Handwriting styles vary across individuals, and some writing is more clear than others. But it also changed across time periods and geographic areas. However, for these script types scholar descriptions can be found in handbooks of paleography. Script models in tables can serve as support. Also, abbreviations commonly used in historical texts changed over time.

Manual transcription of historical texts in general and probably historical ciphertexts in particular is laborious and time-consuming. It requires a high level of concentration and despite all efforts it is prone to inconsistencies and mistakes. In addition, the personnel needed causes expenses.

Even if the transcription should be as accurate as possible, the transcriber has to make decisions with regard to how detailed a transcription should be. In general, we can differentiate between two different levels of granularity. Either we transcribe very close to the historical writing and represent all word boundaries, all punctuation, all line and page breaks, and give spelling and abbreviations exactly as they appear in the original text (diplomatic transcription), or we modernize for instance punctuation and spelling, correct obvious mistakes, and dissolve abbreviations to help the modern reader (normalized transcription).

For historical ciphertext, we apply a high degree of granularity and aim to capture as many details as possible, for instance spacing, diacritics, and punctuation marks (i.e., everything that might be of relevance to be able to recover the plaintext). In the DECRYPT project, diplomatic transcription is applied.

One of the first tasks of the transcription process is to identify and segment each symbol in the ciphertext. Sometimes it is straightforward, as in the case of the clearly segmented digit-based cipher or the eclectic collection of symbols in the Copiale cipher, shown in Figure 3.1. Sometimes symbol segmentation is rather difficult, especially when the scribe used connected handwriting style with touching symbols, as in the case of the Borg cipher in Figure 3.1. To segment symbols correctly, it is helpful to look at highly similar symbols as they occur in the manuscript, especially in connection to other symbols to see where the symbol boundaries should be drawn. Spaces as shown in the original should not be left out from the observation. Spaces in ciphertexts can be intentional, often marking symbol boundaries and also word boundaries from the plaintext. However, spaces are sometimes just added to make decipherment harder. Spaces can also be unintentional where the scribe happened to put a space during writing that actually can reveal an actual word boundary in the plaintext. Therefore, spaces should be carefully observed and transcribed.

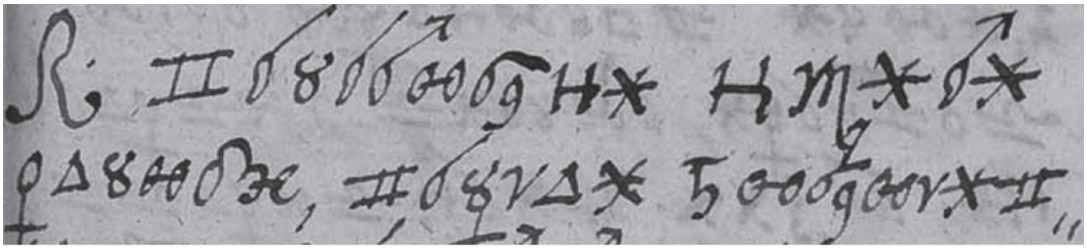
At the same time or as a next step, it is natural to group the similar symbols into a type and assign a unique letter or symbol to each symbol type to be used for transcription. The main difficulty at this step lies in the definition of a group. How similar shall the symbols be in order to be clustered into one group? Should a, a., á, à, å, and ä be one or several groups? How many? Investigating what types of symbols the ciphertext alphabet consists of and how frequent specific symbols are and in what context of other symbols (n-grams) they appear in can be of help. For example, if we can find some digits (1–3), then it is probable that we can find all digits (0–9). Similarly, if we can find some zodiac symbols, we can expect to find more of them, or even all 12. If a symbol with a dot appears only in one or a few cases, the dot could be an ink spot; but if it appears and is used systematically, it should be treated as a symbol type.

A big challenge for the transcription of ciphertexts is with eclectic symbol sets using a large variation of graphic signs; see examples of the Borg and Copiale ciphers in Figure 3.1. Many symbols look similar making it unclear whether we have to do with two distinct cipher symbols or the same symbol with some graphic variation due to the handwriting. For example, the zodiac signs ♍ and ♏ (UTF-8 char: U+264D and U+264F, respectively), look similar at the first sight but if we are familiar with zodiac signs, we can easily distinguish between the two. Human creativity many times invented their own signs with tiny differences between some symbol types, representing different plaintext entities. The challenge of identifying the unique ciphertext alphabet can often be only solved together with the following decipherment process.

To be able to study ciphers and compare them over time and across geographic areas, it is an advantage to have a transcription standard for encrypted sources so that the same symbol types are transcribed similarly across ciphertexts as well as cipher keys. A standardized transcription of all encrypted sources allows matching of ciphertexts with their corresponding key, which makes both decryption and historical contextualization more straightforward.

Within the DECRYPT project, transcription guidelines were developed; see [22] and [23]. The guidelines deal with the systematic transcription of ciphertext images, cipher-key images, and cleartext images.

The basic principle of the transcription is to transcribe the manuscript as close to the original as possible with a special attention directed on the ciphertext itself. Each line is transcribed symbol by symbol with line breaks, spaces, punctuation marks (periods, commas, question marks), diacritics, and underlined sequences marked. Symbols are represented in Unicode using the UTF-8 encoding scheme [24]. Uncertain symbols are transcribed with the guessed symbol followed by a question mark. Unknown letters are marked with an asterisk (*). Figure 3.10 shows



Unicode names

```
R <SPACE> gemini conjunction taurus conjunction ironore arsenic cancer H sextile <SPACE> H
virgo sextile ironore sextile
Copperore fire taurus arsenic delta ? <SPACE> gemini conjunction taurus aries sextile
<SPACE> 5 arsenic cancer arsenic aries sextile gemini
```

Unicode codes

```
R <SPACE> 264A 260C 2649 260C 2642 29df 264b H 26b9 <SPACE> H 264d 26b9 2642 26b9
2640 25b3 2649 29df 03b4 ? <SPACE> 264a 260C 2649 2648 25b3 26b9 <SPACE> 5 29df 264b 29df
2648 26b9 264a
```

Unicode characters

R <SPACE> ♊ ♂ ☿ ♂ ♂ ⚯ ⚷ H ∗ <SPACE> H ♍ ∗ ♂ ∗
♀ △ ☿ ⚯ δ x? <SPACE> ♊ ♂ ☿ ♈ △ ∗ <SPACE> 5 ⚯ ⚷ ⚯ ♈ ∗ ♊

Figure 3.10 Transcription of the Borg cipher [4] represented as Unicode names, converted to Unicode codes, and visualized as original symbols.

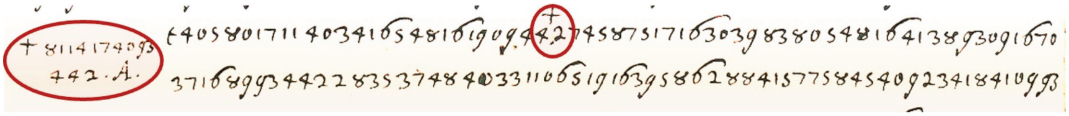
a transcription of the Borg cipher with its eclectic symbol set using Unicode names that can be automatically converted to the actual Unicode codes, and finally represented graphically as icons. It is up to the transcriber's preference to use the Unicode names, which are easier to memorize, or to transcribe graphic signs directly as Unicode codes. Either way, using the keyboard for digits, punctuation marks, and the Latin letters is always preferable for faster progress.

To make the process of decipherment easier, transcription does not always keep to the original image. Instead, the transcription in some cases needs to reflect the intention of the encoder. This means that corrections in the manuscript are transcribed as was presumably intended by the scribe. For example, notes in the margin denoting corrections are transcribed and added to the place as indicated by the given mark in the original, as illustrated in Figure 3.11. Crossed-off symbols in the original are not transcribed but should be added as a comment in the metadata of the transcription file.

Like ciphertexts, cipher keys are transcribed using UTF-8 encoding. However, since cipher keys can be structured in many ways, we do some generalization in the representation of the layout. We separate the plaintext and the code elements onto two sides (different columns), showing this by adding "code" or "plaintext." Each pair is written in a separate line. In cases where several code elements (in the case of homophonic ciphers) or plaintext elements (in the case of polyphonic ciphers) are listed, the alternative elements are transcribed sequentially separated by a bar ("|"), followed by " – " and the plaintext unit(s), regardless of whether the alternatives are written on several lines in the original or not. Special functions in keys (called "operational code elements" in Table 3.1) are also transcribed. A transcription of the cipher key in Figure 3.5 is illustrated in Table 3.2.

The transcription of cleartexts and plaintexts also should represent the original text shown in the image. To be able to distinguish between ciphertext and cleartext sequences, the latter is marked in brackets with a description of the language, as ⟨ `CLEARTEXT LANG-ID Letter_sequence` ⟩. The language ID is a two-letter code defined by ISO 639-1. In addition, catchwords (i.e., a sequence of symbols anticipated as the first symbol(s) of the following page, served to mark page order), are written in brackets. These are marked as ⟨ `CATCHWORD Symbol_sequence` ⟩.

Some documents are damaged and the readability of cipher symbols and other text passages are therefore limited. In these cases, a transcriber marks insecurities in the transcription with a question mark or an asterisk for missing elements. The type of material damage causing the insecurity is described in the metadata, which should be part of the transcription file, and/or as a comment in the transcription. A similar problem might occur when the image quality provided by the archive or library is too poor. Problems caused by low resolution can to some extent be solved



```
1405801711403416548161909 4811417409 3442.A.?2 7458751716303983805481641389309167 0
3716899344228353748403311065191639586288415775845409234184109 93
```

Figure 3.11 Transcribing margin notes.

Table 3.1 Important Terms and Definitions in This Book

Plaintext	The text (or message) intended for encryption and/or the decrypted text.
Cleartext	Intentionally unencrypted text in an encrypted document.
Ciphertext	The encrypted text.
Encryption	The process of transforming a plaintext into a ciphertext using a given key.
Decryption	The process of transforming a ciphertext into a plaintext using a given key.
Cipher	A set of rules (algorithm) describing the process of encryption/decryption.
Key	A piece of information needed for encryption and decryption. A key has to be kept secret for security.
Nomenclature	A part of the key with a list of linguistic entities, such as syllables, words, phrases, or sentences, with their corresponding code elements. Thus, it contains both the nomenclature elements and nomenclature-code elements.
Cryptanalysis (decipherment/ code-breaking)	The process of analyzing a ciphertext without knowing or only partially knowing a key to reveal the original plaintext (and maybe also the key). Some authors emphasize with decipherment that the cryptanalysis process was successful.
Plaintext alphabet	Set of elements used in the plaintext, for example, letters, digits, punctuation marks, spaces.
Ciphertext alphabet	The set of symbols used in the ciphertext (e.g., digits, Latin and Greek letters, alchemical, or zodiac signs). We find these symbols not only in the ciphertext but also in the manuscript containing the key.
Plaintext elements	All types of plaintext entities that have corresponding code elements assigned to them. They usually represent letters, syllables, names, function (e.g., prepositions) and content (e.g., nouns, verbs) words, as well as phrases. The plaintext elements include the alphabet elements and the nomenclature elements.
Alphabet elements	Constitute a subset of plaintext elements. All letters in the alphabet of the writing system that have corresponding code elements assigned to them.
Nomenclature elements	Constitute a subset of plaintext elements. These are above the alphabet level. It may include syllables, names, function and content words, as well as phrases.
Code elements	A symbol or a concatenation of symbols of the ciphertext alphabet used for substitution of the plaintext elements or to indicate that an operation on the revealed plaintext is needed. We distinguish between three types of code elements: alphabet-code elements, nomenclature-code elements, and operational code elements.
Alphabet-code elements	Code elements used for encryption of the alphabet elements.
Nomenclature-code elements	Code elements used for encryption of the nomenclature elements. Nomenclature elements are often encrypted using a different symbol type or of a different length than used for the alphabet-code elements.
Operational code elements	Elements with a special function to carry out an operation on the revealed plaintext. Examples are repetition signs to repeat the preceding letter and cancellation signs (i.e., special code elements that mark the removal of a certain sequence of ciphertext).
Nulls/nullities	A subset of the operational code elements that represent an empty string in the plaintext. Their purpose is to confuse the codebreaker or to mark the start and/or the end of the nomenclature elements.
Code separator/ token separator	A symbol or a concatenation of symbols that separates code elements or groups of code elements from each other. The main intention is to help the receiver to tokenize the ciphertext. In the case of cryptanalysis, it can help to break the cipher more easily.

thanks to methods developed in computer vision science to increase the image quality.

Automatic methods for transcription developed within image processing in general and handwritten text recognition in particular, as parts of one of the scientific fields of artificial intelligence called computer vision, will be the topic of Section 3.4.3.

Table 3.2 Transcription of the Key in Figure 3.5

Code	–	*Plaintexts*
3	–	A | s
6	–	t | r
5	–	n | o
8	–	ı | m
9	–	l | u
7	–	c | e
0	–	p | d
02	–	b | z
04	–	f | g
00	–	& | con

3.4.2 CTTS: Offline Tool for Manual Transcription

To support the time-consuming human labor of manual transcription, George Lasry developed a transcription tool called CrypTool Transcriber and Solver (CTTS). The tool can be executed on Windows, macOS, and Linux, and be downloaded through CrypTool.[1]

CTTS is designed for efficient manual transcription of historical ciphertexts. It also includes a solver for homophonic substitution ciphers. CTTS encourages a cyclic process of review and iteratively editing of transcriptions and decryptions. It provides multidocument support so that users can work on several documents using the same symbol sets simultaneously. CTTS allows to store and load transcription projects and export both the transcribed ciphertexts as well as the decrypted plaintexts.

The nonpublic predecessor version of CTTS was successfully used to crack several real manuscripts (like the Mary Stuart ciphers [11] and the Armand de Bourbon cipher [25]), leading to several publications in *Cryptologia* or at HistoCrypt.

Ciphertexts in historical documents often contain graphic symbols, letters, or digits. The manual process of transcribing such a document with CTTS is as follows:

Step 1: The user loads an image file containing the ciphertext.

Step 2: The user uses the mouse to frame each ciphertext symbol with a box and associates the ciphertext symbols with each other. This is what is described above as grouping the similar symbols.

Step 3: The program generates a transcribed text. In a scenario for a 26-letter alphabet and a simple substitution cipher, it consists of a maximum of 26 clusters of ciphertext letters. Clearly, homophonic substitution ciphers will have many more than 26 clusters, plus additional clusters for punctuation marks, spaces, and other types of delimiters.

Step 4: The user may optionally apply a built-in cryptanalysis algorithm (simulated annealing) on the (so-far) transcribed text to cryptanalyze the cipher and reveal the plaintext.

1. `https://www.cryptool.org/en/ctts`.

Steps 2 through 4 are performed iteratively in a loop to improve transcription and decryption.

Figure 3.12 shows a screenshot of the tool. In the upper right section of the application, a historical encrypted document has been loaded and manually transcribed. Each of the graphical ciphertext symbols is enclosed by a user-drawn box. Boxes of the same color are used to mark ciphertext symbols belonging to the same cluster of symbols. The left side of the application displays a list of all the symbols transcribed so far. Additionally, transcription assignments can be seen; for instance, the first symbol of the list, a 90-degree-rotated letter T, is transcribed as "02." Next to the "02" there is a letter "E," which is the assigned plaintext symbol. Users can manually assign plaintext symbols or an automatic cryptanalysis algorithm can be executed to try and find the best assignments using simulated annealing.

At the bottom of the application, all symbols of the currently selected cluster are visible. Here, all ciphertext symbols transcribed as "02" are grouped in this cluster. This allows users to see which symbols share the same transcription symbol and identify transcription errors. Users can easily correct errors by dragging and dropping incorrectly assigned symbols into a different cluster.

Figure 3.13 shows how the result of step 4 (cryptanalysis) is included into the CTTS GUI again.

3.4.3 Automatic Transcription

Computer vision is the discipline of computer science that makes machines see. In artificial vision, the eyes are the cameras, formed by a matrix of light sensors. These sensors convert the intensity of the light that reaches them into numerical values, generating digital images. But these matrices of points (i.e., pixels), need their brains: computer programs that can associate the sets of pixels with concepts, according to their shape, color, layout, and so forth. In particular, document

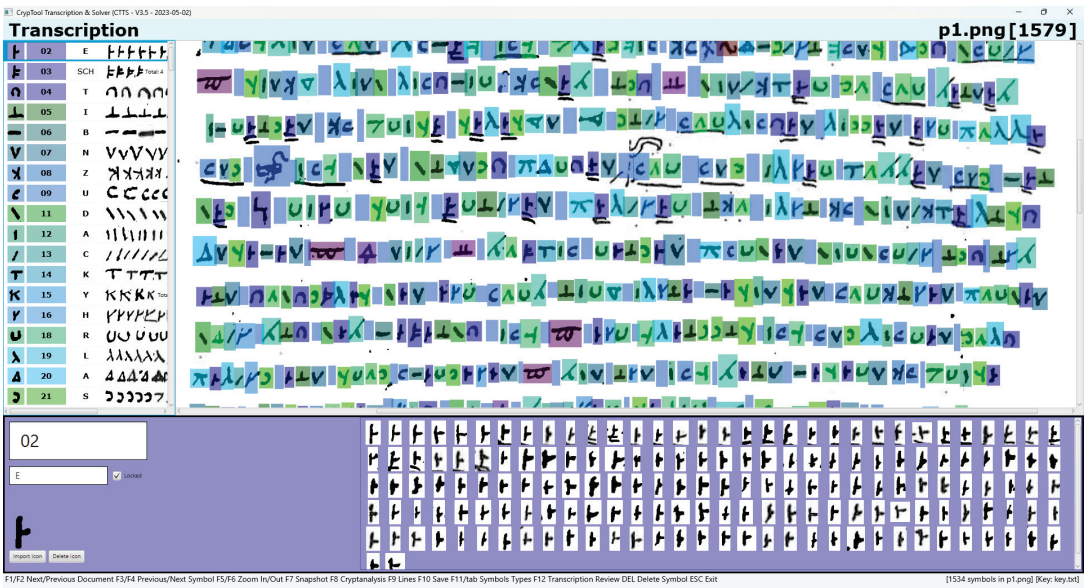


Figure 3.12 Ciphertext transcribed with the program CrypTool Transcriber and Solver.

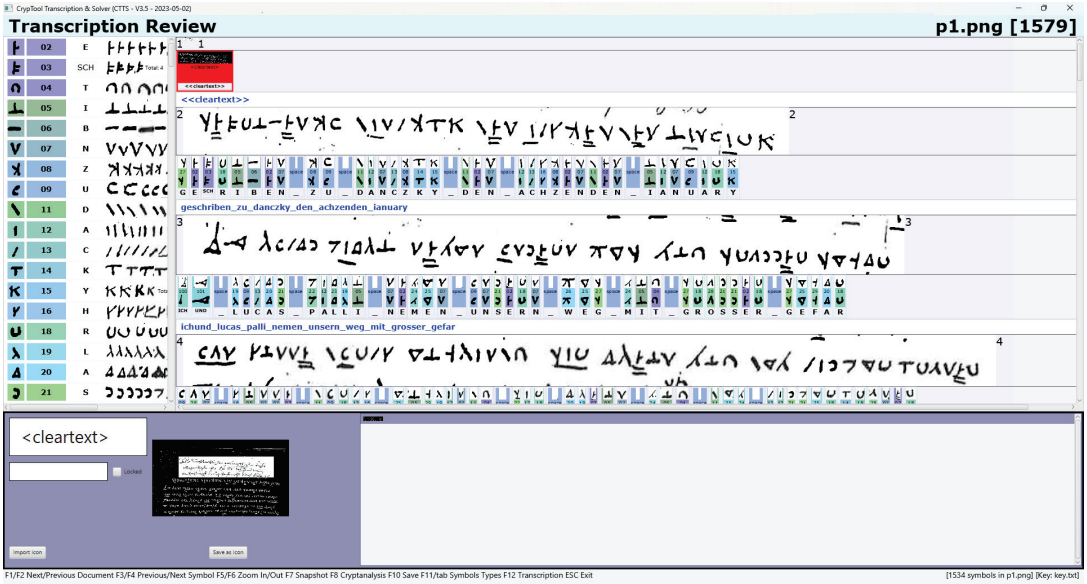


Figure 3.13 Ciphertext cryptanalyzed with the program CrypTool Transcriber and Solver.

analysis addresses the problem of automatically recognizing document content being it printed text, handwritten text, or graphic elements. Traditionally, optical character recognition (OCR) programs recognize clusters of pixels as letters and, at a higher level, validate joint interpretations to end up transforming a digital image into an editable text file. Despite advances during the last decades, reading systems still have limitations, and document analysis research must advance to offer large-scale solutions. In the case of historical handwritten documents, the different handwriting styles, the paper degradation, or the use of ancient languages makes the recognition difficult. Moreover, the use of unknown alphabets, which is commonly the case in such encrypted sources, makes its automatic transcription even more challenging. For this reason, recognition methods must be guided by human experts, and, once the transcription is provided, it must be validated to correct any transcription errors.

Typically, the stages when recognizing text include preprocessing, layout segmentation, and transcription. Given that labeled data (transcribed data) is often not available, the recognition methods are divided into learning-free and learning-based techniques. Next, the main stages of automatic transcription are described.

3.4.3.1 Document Preprocessing

The processing of the image includes those techniques that are usually applied after the digitization of the document. These techniques are essentially applied for improving the quality of the images to make the document more readable, both for people and also for automatic reading systems. In the case of very old and poorly preserved documents, it is necessary to apply document enhancement techniques for minimizing show-through or bleed-through effects, paper discoloration, or loss of ink intensity. Although many document enhancement methods can be directly applied to any input document image, recent deep learning-based methods, such as

generative adversarial networks and transformer networks [26] have demonstrated a superior performance. They need similar (labeled) data to train such systems for a good performance. An example of a document enhancement, which includes binarization, is shown in Figure 3.14. Binarization here means converting a color or grayscale image into a binary image with only black and white pixels.

3.4.3.2 Layout Segmentation

Once the document has been preprocessed and enhanced, the central area of the page must be identified within the image. Layout analysis methods aim to identify the structure and nature of the regions within the document. Many historical documents contain heterogeneous contents, such as text, drawings, or music scores. In the case of ciphertexts, this stage is usually focused on detecting the blocks of text and separating them into lines, words, and ideally, into characters/symbols [27]. However, in many manuscripts, symbols are touching or even overlapping, which makes the segmentation at symbol level difficult, as shown in Figure 3.15 (see the bounding boxes in red color). In such cases, it is preferable to opt for learning-based models.

3.4.3.3 Text/Cipher Recognition

Once the structure of the document has been analyzed and the text regions, lines, and/or symbols have been extracted, these are processed to obtain the final

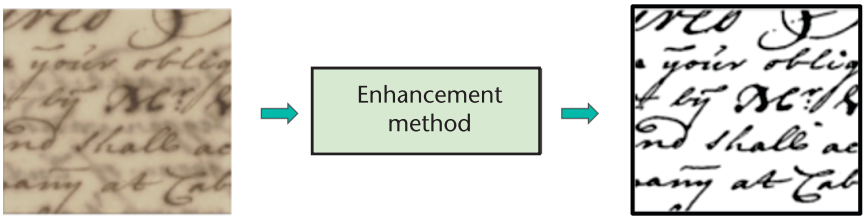


Figure 3.14 Example of a document enhancement method (binarization).

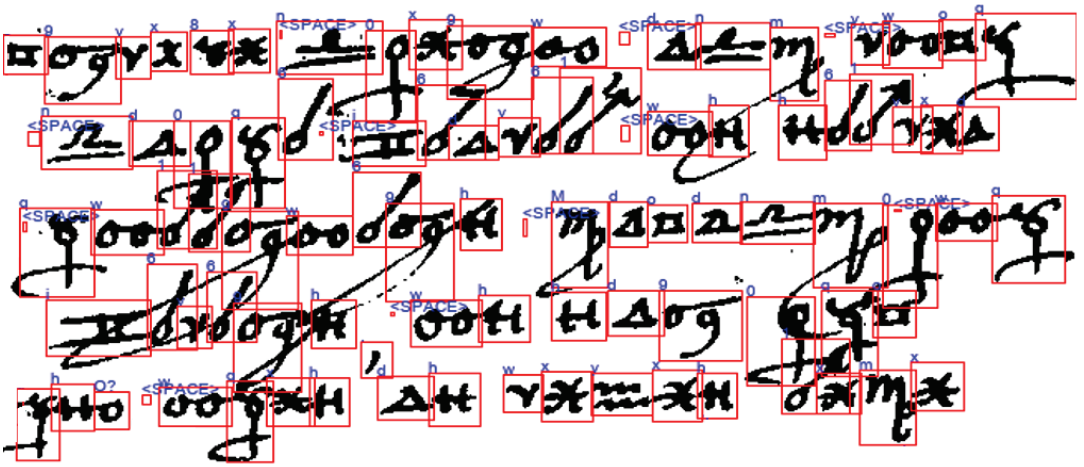


Figure 3.15 Example of symbol segmentation and transcription; segmentation shown in red, transcription in blue.

transcription. Most commercial OCR software only focuses on typewritten text, which means that these programs expect the same visual appearance for each character in the alphabet (e.g., every 'a' looks exactly the same, at pixel level). However, in the case of handwritten documents, the high variability of handwriting styles requires more sophisticated and flexible techniques.

Handwritten text recognition (HTR) methods [28] have been designed for this purpose, which tend to transcribe at line level, avoiding the segmentation into characters that is so prone to errors. Current HTR methods use deep *learning-based* architectures, such as long short-term memory recurrent neural networks (LMRNN), convolutional neural networks (CNN), sequence-to-sequence models (S2S), and transformer networks (TN) [29]. In these systems, the input is usually a text line and the output is the transcribed text. These deep learning-based methods have very good performance, but they require a lot of labeled data to train (more than 100 pages) to learn the shape or visual appearance of each character. But this need for providing examples of text images with their corresponding transcriptions can be a problem in the case of uncommon or unknown alphabets, such as the ones used in many historical encrypted documents. When there is few annotated data to train, the performance of deep learning models dramatically decreases.

For this reason, some researchers opt for learning-free transcription methods, such as learning-free spotting for cuneiform[2] [30] or unsupervised clustering for cipher alphabets like in [31], where the system segments symbols in the document and then groups them according to their visual appearance, using, for example, k-means clustering and label propagation. K-means clustering is an unsupervised method used in machine learning for grouping data into clusters (or groups). It consists in partitioning the elements into k clusters (or groups) so that each element belongs to the cluster with the nearest mean (cluster centers, or prototype of the cluster). Label propagation iteratively propagates the label of each cluster center or prototype through the rest of the nearest elements. The process finishes when all elements are assigned to a cluster, with a label. Then, each cluster corresponds to a particular symbol in the alphabet. Learning-free methods are very flexible and can be applied to any alphabet, but their performance is moderate compared to learning-based approaches, especially when alphabets contain very similar symbols or when characters are difficult to segment, as shown in Figure 3.15.

Lately, different strategies have been explored to deal with the lack of labeled data to train, including few-shot learning, semisupervised and self-supervised learning, transfer learning, and domain adaptation. Few-shot learning aims to mimic how humans learn novel concepts and adapt to unseen data. Concretely, few-shot learning can learn with limited data and the classes (i.e., alphabet symbols) for training and testing can differ. This is especially useful for recognizing manuscripts with rare scripts, unknown alphabets, or very different handwriting styles without retraining the whole model. Rare scripts are those alphabets that are not commonly used today (like Egyptian hieroglyphs, cuneiform, runes, or cipher alphabets). For example, a transcription method based on few-shot learning could

2. Cuneiform is a logosyllabic script used to write several languages of the ancient Near East (from around 3500 BC).

learn how to transcribe symbols from one alphabet, and then use this knowledge when transcribing symbols from an unseen new alphabet. Secondly, semi- and self-supervised learning aim to learn representations from few or no labeled data, which can transfer well to recognition tasks. These types of methods can also be combined with few-shot learning. For example, in [32] a few-shot learning method incrementally transcribes the symbols with a higher confidence rate (namely pseudolabels), assuming that their labels are correct, and uses these pseudolabels as training data for the next iterations, as shown in Figure 3.16. It must be noted that all these types of approaches require only a few annotated examples compared to standard deep learning methods, while reaching a performance only slightly below the typical deep learning-based ones.

3.4.4 The Future of Automatic Transcription

When comparing the manual transcription versus the automatic transcription, it is obvious that, in general, the use of automatic transcription methods are preferable because they minimize the human effort (see Section 3.4.1).

Automatic transcription decreases time-consumption significantly, especially for larger documents. However, for an automatic transcription, the user is required at the beginning to provide labeled data for learning-based methods, and at the end to validate the transcriptions and correct any possible errors. Besides, even though this manual postcorrection can be facilitated since the mistakes by automatic transcriptions are systematic, it requires time. For this reason, a manual transcription can be preferable for transcribing short manuscripts (a few pages). For anything else, the automatic transcription plus manual postcorrection is preferable: In this scenario, semi-interactive software tools are desired, so that the user can guide the automatic transcription (following the idea of *AI in the loop*), and benefit from intuitive graphical user interfaces for the postcorrection. The reader can find a deeper discussion about manual versus automatic transcription in [33].

The field of computer vision develops quickly, as do other branches of AI, and sooner or later we will have access to tools that not only can produce a reliable transcription but also decipher the encrypted manuscript in one step.

Next, we will turn to methods to analyze and decipher encrypted sources.

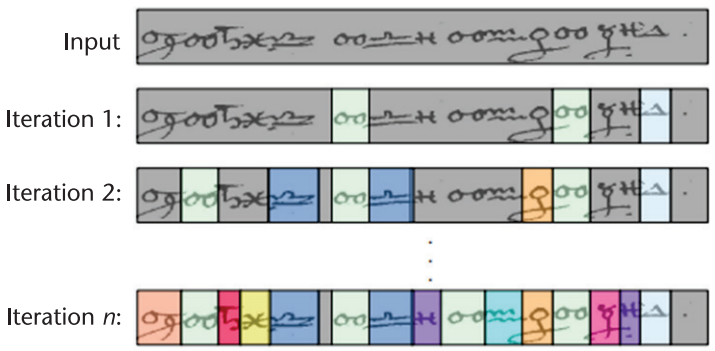


Figure 3.16 Example of incremental transcription by pseudolabeling. At each iteration, the method transcribes the symbols with higher confidence. Each color corresponds to one label. (*From*: [32]. Reprinted with permission.)

3.5 Cryptanalysis

Historical ciphers can be attacked automatically using a computer with heuristic methods like hill climbing. In the previous sections, we presented the different ways in which historical ciphers were built based on alphabet-code elements and nomenclature-code elements. While the alphabet-code elements can be recovered using properties of the original plaintext by methods such as counting frequencies of unigrams, bigrams, and trigrams, as these still show through the encryption, the nomenclature elements cannot really be recovered by automatic cryptanalysis. This is because nomenclature-code elements do not appear as regularly or as frequently as alphabet-code elements do. Nomenclature-code elements can be deciphered either by having access to the original key showing the corresponding plaintext element, or by linguistic and/or historical analysis through contextual interpretation. Contextual analysis (see Section 3.6) might involve the investigation of the surrounding words to reveal the linguistic type in terms of part-of-speech of the plaintext element (e.g., preposition, proper noun, common noun, verb), and/or historical analysis of the entire text to make educated guesses about probable certain persons or places mentioned in the underlying plaintext.

For cryptanalysis, the cipher type and the cipher alphabet used to encrypt the plaintext have to be determined. In the previous sections we showed that, for example, letters, graphic signs, digits, or a combination of them were used as alphabet-code elements. One recognizes only after the decipherment whether, for example, two symbols transcribed together into the same cluster (e.g., A and Ä, whereby one overlooked the points of the Ä) are actually two different symbols, that should have been transcribed differently.

While individual alphabet-code elements with graphic symbols and alphabet symbols are easily distinguishable (mostly, one symbol corresponds to one alphabet-code element), digit-based ciphers are often challenging to segment. Only a few digit-based ciphertexts have visible separations of the code elements (e.g., spaces, a comma, or a dot). Many ciphertexts use scriptio continua with a consecutive sequence of digits without any separation between them (see Section 3.1). Here, tokenization needs to be applied to cut the digit sequences into code elements and identify them, which is far from straightforward as the length of codes can vary within a single ciphertext (e.g., two-digit and three-digit code, or a combination of them).

In the subsequent sections, first we describe the tokenization of ciphertexts. Then, we present two algorithms using heuristics—namely hill climbing and simulated annealing—for the automatic recovery of alphabet-code elements from the transcribed text. Finally, we discuss cost and fitness functions as well as language models used during cryptanalysis.

3.5.1 Tokenization

Tokenization in the context of historical ciphers is defined as the separation of ciphertext into single code elements, be it alphabet or nomenclature codes. Tokenization can be straightforward if the code elements are clearly segmented from each other by separators like a space. Tokenizing a ciphertext that consists of

graphic symbols (e.g., alchemical or zodiac symbols) is often also easy as each symbol being regarded as one token (i.e., one alphabet-code element). However, the tokenization of graphic ciphers sometimes has to be refined or corrected during the cryptanalysis because the creator of a transcription of a ciphertext falsely regarded two symbols as one token.

In contrast, tokenizing digit-based ciphers that are written in a continuous script (scriptio continua), without segmentation between the code elements, is challenging. So far, no solution has been found that allows the generally automated tokenization of such ciphertexts. At the time of writing, tokenizers need to be developed and adapted to individual ciphertexts.

Before attempting to develop a new tokenizer, we can start by applying the most trivial one—tokenizing the ciphertext into two-digit alphabet-code elements, which occur commonly in early modern ciphers. We can also apply already existing tokenizers developed for particular sets of ciphers originating from the same source to new ciphertext of the same collection, such as the papal ciphers from the Vatican or diplomatic correspondence between two sources. If the abovementioned alternatives do not lead to a correctly tokenized ciphertext, a new tokenizer has to be developed. To do so, the ciphers and the corresponding ciphertexts have to be statistically analyzed to find a set of rules the tokenizer is based on. Counting and analyzing unigram, bigram, and trigram frequencies of single digits, two-digit codes, three-digit-codes, and so forth are normally performed. Analysis contains to discover various structures in the code system. For example, if we see that the digit "2" is always in front of an odd digit, it may indicate that the combinations "21," "23," "25," "27," and "29" are valid tokens and may represent alphabet-code elements. In the end, one has to manually look for such peculiarities in the frequencies. The tokenizer can then be applied to the ciphertext and its output be run by the cryptanalysis algorithm(s) of choice (e.g., CT2) to recover the key. If cryptanalysis fails, the tokenizer is probably incorrect and needs adjustment. In the end, the process of tokenization of the ciphertext and the development of a valid tokenizer is a trial-and-error but inevitable process for successful cryptanalysis.

3.5.2 Heuristic Algorithms for Cryptanalysis

A basic flaw (and our advantage) of all simple and homophonic substitution ciphers is the fact that a partially correct key may already allow us to read the content of an encrypted text. Also, text frequencies of the original plaintext may be still visible in the encrypted text. For example the most frequent ciphertext letter in a simple substitution cipher or the most frequent homophone in a homophonic substitution cipher most likely encrypts the most frequent letter. In case of the English language, this would be the letter "E." Both these properties—the ability to have partial correct keys and the appearance of plaintext frequencies in the ciphertext—allow using heuristic algorithms to incrementally solve such ciphertexts. In the following, the two most used and most successful algorithms to break these ciphers are presented. Even though we focus here on the aforementioned two types of substitution ciphers, the algorithms shown can be applied to many other pen-and-paper ciphers as well as to rotor encryption machines. The heuristic algorithms have to be adapted specifically for each cipher.

3.5.2.1 Hill Climbing

The main goal of a hill-climbing algorithm is to find a solution of a search problem that cannot be solved by means of an exhaustive search (i.e., brute force). For example, a simple substitution cipher with 26 ciphertext letters has a total key space (search space) with 26! elements, which are about $2^{88.4} \approx 4 * 10^{26}$ = four hundred octillion different keys. Finding the correct key by testing all possible keys to decrypt the ciphertext is impossible in practice. With hill climbing, the search is possible in practice, but in some cases, for example, very short ciphertexts or poorly transcribed ciphertexts, it might not find the correct key. However, luckily the vast majority of simple monoalphabetically encrypted ciphertexts can be deciphered easily.

The basic hill-climbing algorithm for finding the correct key k_c of a ciphertext ct encrypted with the simple substitution cipher consists of five steps:

1. Select a randomly chosen start key k
2. Decrypt the ciphertext ct to get $pt := \text{decrypt}(ct, k)$
3. Compute the cost value of pt with $f := \text{cost}(pt)$
4. Loop while a defined termination criteria is not met:
   a. Generate a new key k' which is a slightly modified k
   b. Decrypt the ciphertext ct to get $pt' := \text{decrypt}(ct, k')$
   c. Compute the cost value of pt' with $f' := \text{cost}(pt')$
   d. if $f' > f$ then assign $f := f'$ and assign $k := k'$
5. Output the key k (which most likely is the correct key k_c)

The five steps of the hill-climbing algorithm can be clustered into two parts: The first part is the initialization, which is steps (1) to (3). It first generates a random start key and rates its "cost" using a cost (or fitness) function. The higher the cost value, the closer the decrypted plaintext is to real text. In the second part, the algorithm incrementally improves the key. To do so, it generates in step (4a) a slightly modified key, which it then rates in step (4c) using the same cost function as in the initialization part. When the cost value is higher than the previous one it keeps the new cost value as well as the new key. The algorithm loops as long as a defined termination criterion is not met. Finally, in step (5) it outputs the key k, which is with high probability the correct key k_c.

The algorithm can be visualized in a two-dimensional graph as shown in Figure 3.17. Here, the keys are drawn at the x-axis, and the corresponding cost values at the y-axis. The hill-climbing algorithm follows the cost function to find the global maximum (= the key k_c). The figure shows a potential problem of the hill-climbing algorithm, namely local maxima where the algorithm might get stuck (sitting stick figure). Later in this section, we will discuss how to mitigate the effects of local maxima on the success rate of cryptanalysis. Also, keep in mind that while the algorithm can be nicely drawn in a two-dimensional manner, the real problem is a multidimensional problem with, for example, 26 dimensions in the case of the simple substitution cipher with a 26-letter alphabet.

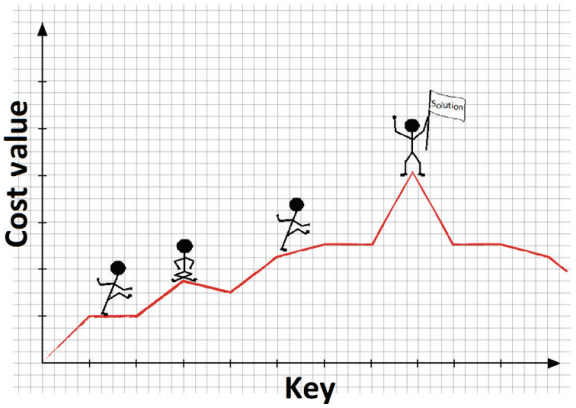


Figure 3.17 A visualization of the hill-climbing algorithm.

In the following, we discuss different aspects and design ideas of the hill-climbing algorithm to break simple substitution ciphers.

Decrypt function and key representation. For the simple substitution cipher, our decryption function requires both the ciphertext and a key as input. The key is represented by a string or array of characters with the same length as the plaintext alphabet. For example, the key "WDNBZCJHOKQRPEISFTUGVXYALM" means that the "W" is decrypted to "A," the "D" is decrypted to "B," ..., and the "M" is decrypted to "Z." The actual decryption is performed by walking letter by letter through the ciphertext and replacing the ciphertext letters with plaintext letters as described before.

Start key. The generation of the start key can be crucial for the success of a hill-climbing algorithm. For some ciphers, a "good" start key is needed to allow the algorithm to converge to the correct solution. With the simple substitution cipher, the start key can just be chosen at random. To do so, we take the alphabet of the assumed plaintext language (e.g., the Latin 26-letter alphabet for the English language) and create a key by shuffling it:

ABCDEFGHIJKLMNOPQRSTUVWXYZ → WDNBZCJHOKQRPEISFTUGVXYALM

With historical encrypted manuscripts, the used alphabet can differ from the alphabet we use today. Some letters may be represented by the same single letter (e.g., "I"="J" and "U"="V"). This depends on the plaintext language and the time of the creation of the manuscript. Sometimes, letters may be intentionally omitted for security purposes, such as by writing a single "L" instead of "LL" or "VV" instead of "W." Sometimes, the alphabets are extended, for example, by adding a symbol for double letters ("LL"), "SCH," or letters with diacritics ("á"). This all has to be taken into account when generating an alphabet and keys with an automated heuristic-based analyzer.

Cost function. The cost or fitness function evaluates the quality (cost or fitness value) of a supposedly decrypted plaintext. Depending on the problem (the cipher), a special cost function may have to be implemented. For the simple substitution

cipher, a language model (n-gram statistics; n being between 3 and 5) is used. In Section 3.5.3 we discuss cost and fitness functions in more detail.

Key modification. The next important part of the hill-climbing algorithm is how to modify the key k to obtain a new key k' during a single iteration. Only a small change in the key allows the algorithm to smoothly follow the curvature of the graph of the cost function to potentially reach its global maximum.

Figure 3.18 shows how the key k is modified to create a new key k' during hill climbing by swapping only two letters (here "C" and "F") at the same time. There are different strategies how to choose which two letters should be swapped:

1. *Perform a single random swap:* In every iteration of the hill-climbing algorithm, use two random indices i and j with $i \neq j$. The two letters at position i and j are swapped. Clearly, only "good" swaps are kept and "bad" swaps are discarded.
2. *Take only the "best" swap:* In every iteration of the hill-climbing algorithm, all indices i and j with $i \neq j$ are tested. The "best" swap of all possible letter swaps is kept. The "best" swap of all possible swaps is the "good" swap, that increases the cost value the most.
3. *Take all "good" swaps:* In every iteration of the hill-climbing algorithm, all indices i and j with $i \neq j$ are tested. Every time a "good" swap occurs, the swap is kept. This means, that during a test of all indices i and j in an iteration, multiple consecutive "good" swaps may occur.

The classical hill-climbing algorithm as described in the literature uses random swaps of two letters—the strategy (1) above. While this works well in most cases, the two other strategies may improve the success rate as well and reduce the computational time needed by the cryptanalysis algorithm. With strategy (2), we test all possible swaps and only take the "best" possible "good" swap. A "good" swap increases the current best cost value while a "bad" swap leads to the same or even a worse cost value. With a 26-letter alphabet, there are $\frac{26 \cdot 25}{2} = 325$ different swaps that need to be tested in every iteration. Clearly, this slows down the algorithm and increases the needed computation time in the worst case by a factor of 325. To mitigate the effect of testing all possible two-letter-swaps, there is strategy (3) that allows already to keep a "good" swap while all remaining swaps still need to be tested. For example, the Vigenère analyzer component of CT2 uses strategy (3), which allows solving really short Vigenère ciphertexts with high success rate in very short times [34].

Termination criteria. In theory, a hill-climbing algorithm should terminate, when it reaches the global maximum (success) or it got stuck in a local maximum (fail).

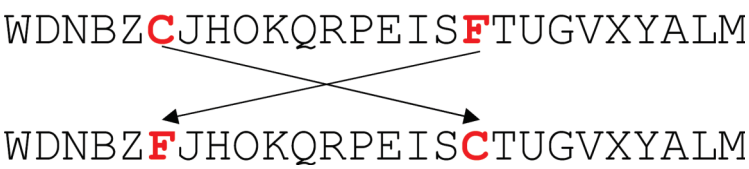


Figure 3.18 Swapping two letters of key k to obtain a modified key k'.

Depending on the selected key-modification strategy, it is possible to detect if the algorithm got stuck or not. For example, with random swaps, it is possible that it by chance never selects a new swap that allows us to increase the cost value, despite there exists another "good" swap. Thus, a suitable termination criterion for random swaps is to count the number of consecutive randomly chosen "bad" swaps and then terminate when a specific number of "bad" consecutively chosen swaps is met. With the two other strategies, (2) and (3), we can actually find out if the algorithm got stuck because in every iteration all possible swaps are tested. If all of these swaps are "bad" swaps, the algorithm terminates.

Strategies to counter getting stuck. There are different strategies to counter getting stuck with hill climbing in a local maximum:

1. *Better start keys*. With some ciphers, it is possible to already generate "good" start keys that are close to the global maximum. In the case of the simple substitution cipher, this is not needed, since any randomly created start key can be used and will lead to the correct solution in nearly all cases. In contrast, with homophonic substitution ciphers, a good start key improves the success rate and performance of the algorithm. We describe this later in Section 3.5.2.2.
2. *Better key modification(s)*. For example, instead of swapping only two elements of the key at the same time, one could perform a triple swap, where element i becomes j, j becomes k, and k becomes i while $i \neq j \neq k$. With the simple substitution cipher and with the homophonic substitution cipher, swapping only two letters at the same time is good enough.
3. *Better cost function*. When hill climbing does not find the correct key, it is probably a good idea to change the cost function. For example, instead of using n-gram models with $n = 2$, we could increase the dimension of the language model to $n = 3$. With simple and homophonic substitution ciphers, $n = 5$ works very well. Sometimes, it can also be useful to change to a lower n, especially with bad transcriptions or many errors in the ciphertext. See Section 3.5.3.
4. *Shotgun hill climbing/random restarts*. Another idea of improving the algorithm is to restart it several times (e.g., 100 times) with different randomly chosen start keys. This is also referred to as shotgun hill climbing, since the start keys are distributed over the key space like shotgun shrapnels. With the simple substitution cipher, this strategy is very effective.
5. *Use of simulated annealing*. This algorithm is an alternative to hill climbing. See Section 3.5.2.2.

When working on a historical ciphertext, all the aforementioned improvements have usually to be tested individually. For example, evaluations with different key modifications and cost functions have to be performed to test the impact of the changes on the cryptanalysis success rates. For CT2, the implemented cryptanalysis algorithms were tested and tweaked with millions of artificially generated test records until sufficient success rates were achieved. Additionally, all CT2 cryptanalysis components allow exchanging the language model or set different parameters in

the corresponding components' settings. A few examples of such CT2 components are the substitution analyzer, the Vigenère analyzer, the homophonic substitution analyzer [35], and the Enigma analyzer.

Figure 3.19 shows a screenshot of the CT2 homophonic substitution analyzer[3] solving an encrypted letter written by Holy Roman Emperor Maximilian II and sent to Polish delegates in 1575. The upper part of the analyzer has some helpful information about the currently analyzed ciphertext, such as the number of used homophones. The large middle part shows the analyzed ciphertext. The lower part shows the currently revealed plaintext. Green marked symbols are already locked, meaning they won't change any more during the ongoing cryptanalysis. Blue marked symbols show German words found in a predefined dictionary. A CT2 user can stop the automatic analysis process at any time and manually change and improve plaintext-ciphertext symbol-mappings on his own.

3.5.2.2 Simulated Annealing

Simulated annealing is a generalization of hill climbing: The basic idea is that with a defined probability modifications of the key are also chosen, which lead to a bad key, which means the cost value may decrease in an iteration. Over time, the probability for selecting a bad key is reduced until it reaches zero. Then, simulated annealing behaves exactly the same way hill climbing does.

The simulated annealing heuristic is inspired by the physical annealing in metallurgy. Here, annealing is a slow process of heat treatment of metals to alter the

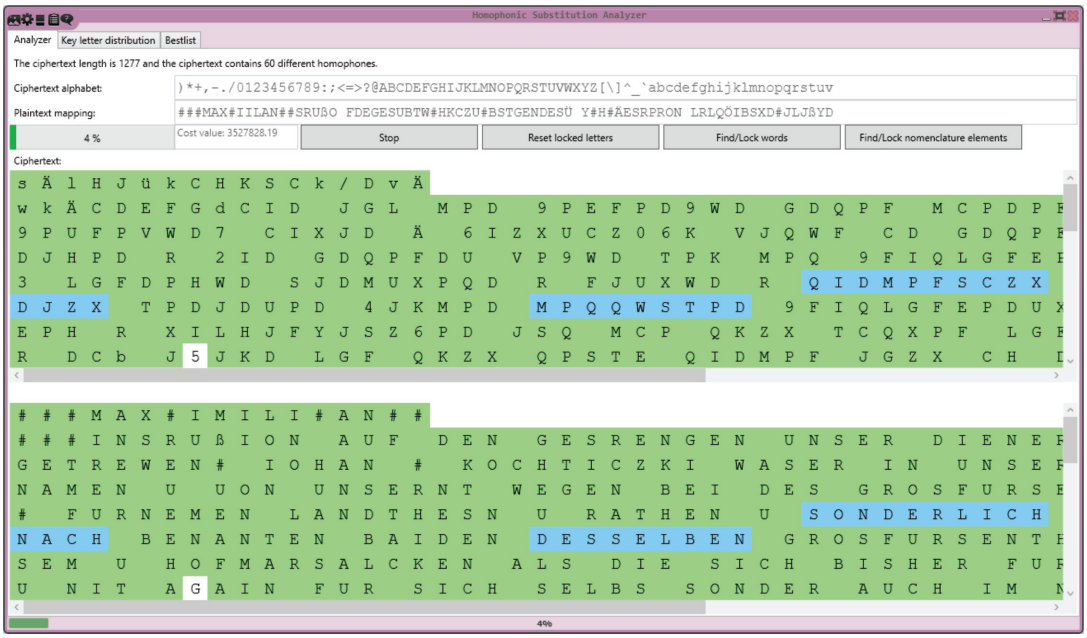


Figure 3.19 The CT2 homophonic substitution analyzer solving an encrypted letter from Maximilian II.

3. In `CT2 Startcenter` ▷ `Templates` ▷ `Cryptanalysis` ▷ `Classical` ▷ `Homophonic Substitution Analysis`. In CTO, a similar homophonic analyzer can be found.

physical properties of the material. While in physical annealing, the real temperature is slowly decreased; with simulated annealing a virtual temperature value is used. The basic simulated-annealing algorithm consists of six steps:

1. Select a randomly chosen start key k
2. Set the temperature to a start value $t := t_{\text{start}}$
3. Decrypt the ciphertext ct to get $pt := \text{decrypt}(ct, k)$
4. Compute the cost value of pt with $f := \text{cost}(pt)$
5. Loop `while` $t > 0$
   a. Generate a new key k', which is a slightly modified k
   b. Decrypt the ciphertext ct to get $pt' := \text{decrypt}(ct, k')$
   c. Compute the cost value of pt' with $f' := \text{cost}(pt')$
   d. If $f' \geq f$ then assign $f := f'$ and assign $k := k'$ `else`
      - Compute a degradation value $d := -\text{abs}(f - f')$
      - Compute an acceptance probability $p = e^{\frac{d}{t}}$
      - Choose a random value r in the interval $]\,0\,;\,1\,[$
      - If $p > p_{\text{min}}$ and $r < p$ then assign $f := f'$ and assign $k := k'$
   e. Decrease temperature, for example, by using a defined step size ss to get $t := t - ss$
6. Output the key k (which most likely is the correct key k_c)

In step (2) a start temperature is set. The start temperature, among other new values needed for simulated annealing, has to be tweaked for each type of cipher and often also for each individual ciphertext, which you want to cryptanalyze. The termination criterion in step (5) now checks if the temperature t is still higher than 0. Inside the main loop of the algorithm, when a key k' is not accepted in step (5d), a probability p based on the degradation value is computed and a random value r is chosen. If r is smaller than the computed probability and the computed probability is greater than a minimum probability p_{min}, the bad key is kept. In practice, we set the minimum probability to $p_{min} = 0.85\%$, which gave us good results. This allows the simulated-annealing algorithm to jump away from local maxima. While the algorithm is being executed, the temperature value t is reduced by a step size ss. The value of s is predefined and can be determined, for example, by dividing the start temperature by the number of wanted steps s, and then the algorithm should perform. So $ss := \frac{t_{\text{start}}}{s}$. Other temperature reduction strategies are also possible. For example, instead of reducing the temperature by the same value ss all the time, it could also be reduced by a percentage value of t with $t := t - 0.01 \cdot t$. The different strategies have to be evaluated to find the best one for the specific case. Figure 3.20 shows a simulation of the key acceptance probability of simulated annealing over time with a fixed temperature step size and Figure 3.21 shows a simulation of simulated annealing with a percentage-based temperature step size.

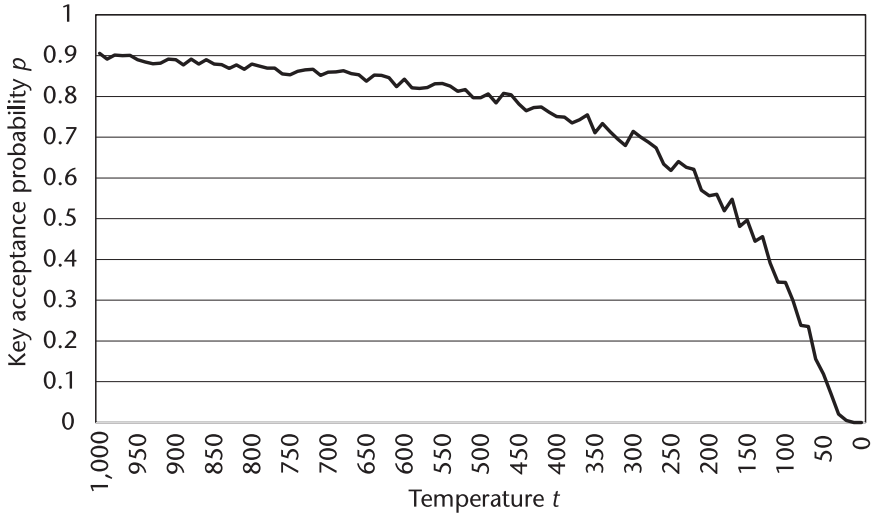


Figure 3.20 Key acceptance probability of simulated annealing with linear decreased temperature over time.

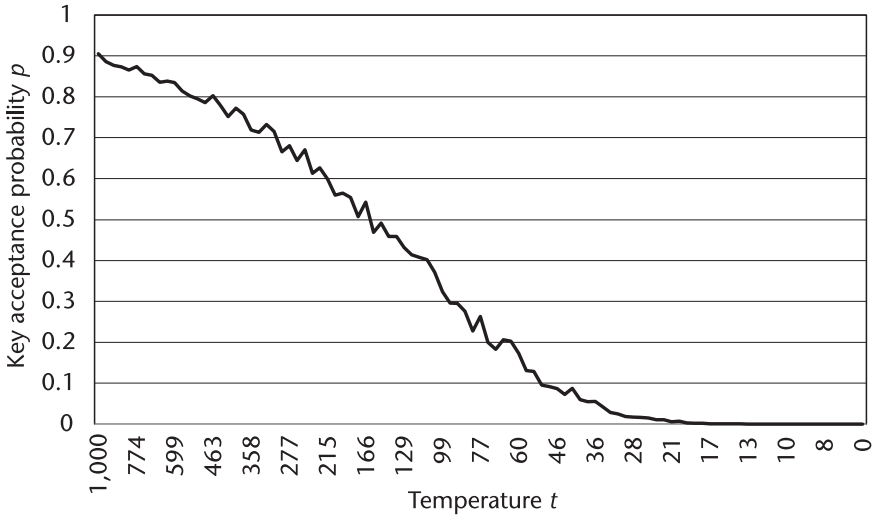


Figure 3.21 Key acceptance probability of simulated annealing with percentage decreased temperature over time.

Improving simulated annealing for homophonic substitution ciphers. During the cryptanalysis of the homophonic substitution cipher, plaintext letters from the plaintext alphabet are assigned to all homophones and the ciphertext is decrypted for testing. During a single iteration of the simulated-annealing algorithm, we swap the assignments of two plaintext letters. As with the simple substitution, we test all possible two-letter swaps of all homophones.

In the following, we present some adaptions and strategies to be applied to the simulated-annealing algorithm to improve its performance, especially for the cryptanalysis of homophonic substitution ciphers.

1. *Good start keys.* With the homophonic substitution cipher, it is helpful when the start keys for the cryptanalysis algorithm are already chosen in a

way that reflects the distribution of letter frequencies of the language. For example, it is better to assign more homophones to more frequent plaintext letters (e.g., the "E" with English) than to less frequent letters (e.g., the "X" with English). Therefore, the Homophonic Substitution Analyzer of CT2 allows distributing the letters among the homophones based on probabilities that are based on the original text frequencies of the language.

2. *Homophone locking (manual).* When analyzing homophonic substitution ciphers, it may improve the cryptanalysis if already correctly assigned letters can be fixed by the user. The Homophonic Substitution Analyzer of CT2 allows this in the semiautomatic mode. Here, the user may pause the analysis and lock homophones, meaning the corresponding assignment of plaintext letters to the homophones cannot be changed anymore by the cryptanalysis algorithm during the further iterations. Also, the user may change and correct the already made assignments.
3. *Homophone locking (automatic with a dictionary).* Besides manually locking homophones as described above, it is possible to automatically lock homophones based on words found in a dictionary. Therefore, the Homophonic Substitution Analyzer of CT2 provides a dictionary to the cryptanalysis algorithm. Every time a new global best value (best key) is found, the analyzer searches for words with a minimum and maximum length. If it finds more words than a specified threshold value, it automatically locks all corresponding homophones to their corresponding plaintext letters. This can also be combined with the manual method for homophone locking described in the second adaption.

3.5.3 Cost Functions

While optimizing a key k with hill climbing or simulated annealing, the algorithm needs a way to decide if a modified key k' is better or worse than the original key k. To rate a key, we use cost or fitness functions on the text previously decrypted with the key k'.

The basic idea of a cost function $cost(t)$ is that it calculates a number that reflects how natural a given text t is. The closer the text is to a real text, the higher the cost value should be. The more random (not natural) a text is, the lower the cost value should be. In the best case, the cost function returns the highest value when we enter the original plaintext. Between the lowest and the highest value, there should be a smooth curve that the cryptanalysis algorithm can follow during the optimization of the key.

A common practice is to use a language model built from a large text corpus. For historical ciphers, it also turned out that the cryptanalysis algorithm can benefit from using a language model based on a historical text corpus [36]. A language model returns the probability of a given text being a text of the language it was built for.

The language models used in our cost functions are n-gram models. Such an n-gram model provides a value (probability) for a given n-gram. Clearly, frequent n-grams of the language, such as "ING" in English, return a higher n-gram value than less frequent n-grams, such as "XYZ." An overview of English and German

language frequencies can be found in CrypTool-Online[4] and a set including different other language n-grams can be found on Practical Cryptography.[5]

We created different language models by using large corpora of text. To create a model, we first count the number of occurrences of all individual n-grams (e.g., from "AAA" to "ZZZ" for a 3-gram model) of the set. Also, we count the total number of all n-grams of the corpus. Then, for each individual n-gram, we divide its number by the number of all n-grams to obtain its probability. To compute the cost value of a given text, we could multiply all the computed values of all n-grams of that particular text to obtain a probability (the cost value) of the text. Here, we have two problems: (1) the probability values of each n-gram are very small numbers, which will result in many precision errors when multiplying these numbers on a computer, and (2) multiplications can be costly, so the performance of the computation may be poor. On modern PCs, problem 2 is negligible, but problem 1 is a huge problem. A common way to get rid of both problems is the usage of logarithmic values. Instead of multiplying all small values of the language model, we add the logarithms of each value. This is possible due to the logarithm law $\log_b(x \cdot y) = \log_b(x) + \log_b(y)$. In the end, to obtain the final value, we could raise the used base b to the power of the sum c, meaning b^c. But this is not needed since the optimization algorithm can also run on the logarithmic values. In CT2, the cost values are normalized to double precision floating point values in the interval of $[0 : 10000000]$. By doing so, the CT2 language models are comparable to each other.

A final note on the data format of language models: During cryptanalysis, the letters are mapped into an integer number space based on the used alphabet. For example, with the 26-letter Latin alphabet, the letter "A" is represented by 0, the letter "B" by 1, ..., and the letter "Z" is represented by 25. A language model is an n-dimensional array. To look up, for example, the 3-gram "ABC," which is encoded as integers 0, 1, 2, we can just look up the language model array using the integers as indices. Doing the encoding of letters this way is easy and fast.

The CT2 language model files have a specific binary file format:

```
Header:
    "CTLM"              4 ASCII characters          (magic number)
    LanguageCode        0-terminated UTF-8 string  (language code)
    GramLength          4 byte integer             (length of n-grams)
    Alphabet            0-terminated UTF-8 string  (alphabet)
Data:
    (Alphabet.Length ^ GramLength) * 4 bytes       (model data)
```

A language model file starts with the four ASCII characters "CTLM" (CrypTool Language Model) to identify the file type. The "LanguageCode" string identifies the language model. The "GramLength" defines the size of the n-gram model. The "Alphabet" defines the used alphabet. In the data section, the actual language model data is stored as 4-byte float values containing the logarithmic values computed using a text corpus. The sizes of the n-gram models increase quickly with n, so the models are compressed using the gzip algorithm. For the English language with 26 characters the file

4. See `https://www.cryptool.org/en/cto/frequency-analysis`.
5. See `http://practicalcryptography.com/cryptanalysis/letter-frequencies-various-languages/`.

sizes on disc are (rounded): 1-gram: 1 kB, 2-gram: 3 kB, 3-gram: 50 kB, 4-gram: 800 kB, 5-gram: 8500 kB. Decompressed in RAM (rounded on 1 kB): 1-gram: 1 kB, 2-gram: 3 kB, 3-gram: 71 kB, 4-gram: 1828 kB, 5-gram: 47526 kB. One observation here is that the more data (texts) are used to create these language models, the smaller the amount of file size reduction achieved by compressing the models. The reason for this is that the increase in entropy (aka amount of information) of the data used leads to lower compressibility.

3.6 Contextualization and Interpretation: Historical and Philological Analysis

Once we have managed to reveal (parts of) the plaintext, we aim to set the manuscript in a historical context to recover what was written, by whom, to whom, and why. Such a contextualization concerns historical and philological interpretation, which will be the topic of this section. These approaches involve a broader type of analysis than cryptanalysis described above, because they do not primarily restitute the message, but rather investigate the linguistic and historical context in which the message was written, encrypted, and sent. *Linguistic* analysis involves the contextualization of the given ciphertext into the contemporary language usage, which presupposes that we have sufficient knowledge about how languages were used in the given time period and geographical area. *Historical* analyses do not only involves the identification of the sender and receiver (and perhaps the code-breaker) of the ciphertext, and the political context, but also the transfer of knowledge in the field of cryptology, as well as the social history of those who applied this technology of secrecy.

3.6.1 Analysis of Historical Languages (Linguistic Analysis)

Historical languages pose some specific challenges to the cryptanalyst. One important aspect is that most languages show a great deal of variation before they were standardized sometime in the eighteenth century. This means, for instance, that one and the same word could be written in many different ways (i.e., orthography was not normalized and even the same scribe could use various spellings for the same word in one text [37]). Moreover, in languages such as English, German, or Italian, we find a lot of different dialectal forms in the same language. Languages also change over time, certain words or word forms disappear, new ones emerge. The pilot study [36] on the decipherment of German and English historical homophonic substitution ciphertexts showed that using 4-gram models derived from century-specific texts leads to significantly better performance than language models built on more modern, contemporary texts for ciphertexts produced in the 17th century or earlier. A corpus of historical texts such as a digital library of online texts like the Project Gutenberg or the collection of historical texts with diplomatic transcriptions for 16 European languages available within the HistCorp collection [38] can serve well as a basis for the creation of language models.

Another general aspect to bear in mind in the use of algorithms for cryptanalysis is that in the plaintext alphabet a historical cipher is based on might differ from modern alphabets in specific languages: In many cases, only one letter is used for both *u* and *v*, for instance, and usually, letters with diacritics (such as *ä, ö, ü* in German; or *á, é, í, ő,* etc. in Hungarian) do not form part of plaintext alphabets. At the same time, plaintext alphabets also might merge commonly co-occurring alphabet letters and treat these as one plaintext element, such as *ss* or *sch* in the Copiale [6] cipher with German as its plaintext language.

In historical ciphertexts, especially in the domain of diplomacy and military correspondence, often more than one language was used [12]. Several languages, such as German and Latin, could be combined in one and the same sentence, as was the case in a letter written by a Lithuanian nobleman to the Habsburg Emperor Maximilian II in 1574 [39]. Initially, this fact caused problems in the decipherment process because the analysis was based on a monolingual German language model and the switching was not detected. Only afterwards, in a closer linguistic analysis, the change of language was identified.

It is also possible that different languages were used for passages in cleartext and passages in ciphertext [40–42], or that the plaintext language used in the key and the language of the plaintext of the encrypted letter are not the same. For example, a Swedish envoy based in Germany during the Thirty Years' War used a German key in his correspondence with the Swedish Lord High Chancellor. However, the underlying plaintext in his letters is in Swedish and Latin [43]. Hence, even when the language of cleartext passages or of a key is identified, other languages may still be encountered in the ciphertext.

These examples show that the linguistic analysis of ciphertexts can form part of the process of cryptanalysis and functions as an auxiliary method to solve a cipher and to reveal information about the underlying language, the provenance, and the dating of a ciphertext. In fact, already in the Middle Ages, codebreakers used linguistic analysis in cryptanalysis: Arabic scholars realized that there is a certain frequency distribution of letters in different languages—a tool they used to decipher monoalphabetic substitution ciphers [44, 45]. Linguistic knowledge also helps to detect transcription errors and to resolve certain decipherment problems. Finally, knowledge in historical languages is often needed to fully understand the content of the deciphered documents.

On the other hand, linguistic analysis can serve its own purpose and be aimed at understanding linguistic patterns and language practices in historical cryptographic texts. Examples for this research path are, for instance, studies on what languages were chosen in ciphers in different geographical areas and different times or which and how different languages were combined in documents [42]. Further, the linguistic analysis of a recovered plaintext can complement the historical analysis and contribute to the understanding of scribal practices and language usage at chanceries and black chambers. Historical ciphertexts can also serve as sources for the analysis of written dialects and languages, and language change.

The linguistic analysis can be fully or partly automatized by algorithms developed within computational linguistics and natural language processing. Spelling variation in historical texts can be automatically discovered and normalized to a modern version, cleartext sequences can be detected and its language(s) identified by applying automatic language identification. The computational analysis of language heavily relies on language models derived from large samples of diplomatic transcriptions of historical texts from various time periods and genres. Such collections are not easy to find and their creation requires linguistic and philological expertise.

3.6.2 Historical Analysis and Different Research Approaches

Similar to the linguistic analysis, historical analysis in historical cryptology plays a double role: It might be the *goal* of the whole procedure described above, or alternatively, it might also be a *tool* used in the process. It is the goal when the historian aims to reconstruct certain past events and study a particular historical context. Solving the ciphers, pairing the keys and the messages, and exploring the ways cryptography was used help her in this task. In other cases, however, it is rather a tool: Most homophonic cipher

keys consist of an alphabet part and a nomenclature table. One needs mathematical and linguistic knowledge to analyze the alphabet, but reconstructing the nomenclature table requires a deep knowledge of the historical context. In this second type of case, history is an auxiliary science of the crypto-historian.

In the following, we provide a—by no means exhaustive—typology of the different (sometimes contradictory, sometimes complementary) approaches when historical analysis comes to the picture, and we exemplify each approach with a corresponding publication.

1. One typical research path aims at getting new, previously unknown knowledge by *solving a given encrypted source*. This approach enriches our picture of a particular historical period and becomes useful for traditional history writing, but the emphasis is more on cryptanalysis, the solution of a riddle [46].

2. A second typical research path follows the agenda of *political history*. Ciphers were primarily used in diplomacy. The analysis of the correspondences of political centers with their ambassadors, messengers, and spies can provide new insight into the history of a given era even if the exchanged letters had always been readable because the historical addressee wrote the solution above the ciphertext characters. Examples for this category include studies on diplomatic history [47–49], analyses on the earliest black chambers, such as codebreaking offices [50], and the reconstruction of particular encryption practices (polyphonic and fixed length ciphers) used in the 16th century in the Vatican [51].

3. It is not the aim but the scope of the *microhistory approach* that makes it different from the previous ones. In this case, a temporarily limited series of events (a few years or a few exchanged letters) is analyzed with a variety of tools in order to have better insight into one particular historical event, such as the study on encrypted letters sent by and to the Habsburg Emperor Maximilian II in 1574–1575 [39, 52, 53].

4. The previous approach might be enriched with a *linguistic analysis* of the sources, as described in the previous section. The two fields have always been close: study of languages and cryptology have walked hand in hand from the earliest times.

5. An opposite approach is followed by those who perform *large-scale statistical analyses* of cipher keys and/or encrypted documents. The emphasis is not on particular sources but on conclusions, tendencies, and correlations that can be pointed out on the basis of relatively big data. An example for this approach are the studies on the typology and change of early modern cipher key documents [9, 54, 55].

6. Cryptology is both a technology and a scientific endeavor neighboring mathematics; thus, it is a genuine topic for a *history of science* approach. Basic issues include knowledge transfer (the ways this secretive knowledge is transferred from one generation to another, from one political center to another), the relations of cryptology to other scientific fields (statistics, algebra, poetics, etc.), its technology use, and the evolution of encrypting and codebreaking practices over time [19].

7. A separate category is populated by articles and book-length studies on specific *famous ciphers*, solved or unsolved, such as the Voynich manuscript [56, 57], the Copiale manuscript [6], the Borg cipher [4], or the Beale ciphers [58].

8. Sometimes it is not the ciphers and keys but the social background of the users that is under study. A *social history of cryptology* relies on the same sources but attempts to answer different questions: Who are the human actors of crypto-history, what are their attitudes to the technology they are using, what do they wish to keep as a secret, and so forth [19].

9. And, finally, further approaches are conceivable and can be exemplified by the continuously growing number of publications, including studies on personal diaries, private ciphers, and so forth.

3.7 Conclusion

Historical cryptology is a cross-disciplinary scientific field aiming at the systematic study of historical encrypted sources: ciphertexts, cipher keys, and related documents. The aim is not only to shed light on the content behind the encrypted sources by breaking their code, but also to study the evolution of cryptography and cryptanalysis over time periods and geographic areas.

As with all scientific disciplines, historical cryptology is in need of research infrastructure including resources and tools for the automatic processing of the encrypted documents. In this chapter, we presented several databases containing smaller or larger collections of historical ciphertexts and cipher keys, with the largest—at the time of writing—being the DECODE database [59]. The collections make it possible to study the evolution of cipher keys over time and to identify the most commonly occurring cipher types. We presented the structure and the peculiarities of three commonly occurring cipher types in early modern times in Europe: simple, homophonic, and polyphonic substitution ciphers, all monoalphabetic with or without nomenclatures. Surprisingly, transposition and polyalphabetic ciphers were used very rarely in Europe in these centuries, even though the cryptographic techniques were known. In contrast, in the U.S. Civil War from 1861 to 1865 the Vigenère cipher was used by the Confederates [60].

To break the historical ciphertexts, we introduced a set of tools for both transcription (to turn the images into a machine-readable text format) and for cryptanalysis (to decrypt the ciphertext). We presented transcription guidelines for the consistent transcription of symbol sets across ciphertexts and described the challenges and pitfalls of manual transcription. We then introduced how current handwritten text recognition techniques developed in computer vision are applied to ease the time-consuming and expensive transcription process. Given the ciphertext in text format, we described algorithms for cipher-type identification, cryptanalysis, and decipherment for the most commonly occurring European historical ciphers. We pointed out the importance of language models and various heuristics for the generation of cipher keys. Lastly, we gave an overview of the linguistic and historical interpretation of encrypted sources and the great challenge of their contextualization.

The latest and rapid development in AI provides us with efficient algorithms and models. It's challenging how AI can be efficiently used to produce an error-free and complete transcription to minimize error propagation to the subsequent step of codebreaking, to identify the cipher type used for producing a given ciphertext, and even to get the original message by breaking the cipher. Another future extension could be the selection and analysis of non-European ciphertexts, especially with languages not using a Latin-based alphabet.

The field of historical cryptology requires expertise from various scientific disciplines in order to collect, describe, transcribe, break, and analyze historical encrypted manuscripts. Historians contribute to the contextualization and interpretation of the hidden sources and linguists analyze the historical plaintext by acquiring models for language variation and language change. Cryptanalysts develop efficient algorithms for breaking of various cipher types, and image processing specialists provide models to process images to a machine-readable format. Computational linguists build and evaluate historical language models generated from historical texts. By close cooperation a hidden class of sources, encrypted to hide the content of importance in the past, can be systematically handled and made available to the public.

The interested reader can find scientific articles on the topic in publication channels of various disciplines from history, linguistics, natural language processing, and digital

humanities to image processing and cryptology. The most well-known scientific publication sources for historical cryptology are, however, the proceedings of the annual International Conference on Historical Cryptology (HistoCrypt) [61] and the journal *Cryptologia* [62]. The community of historical cryptology has also a network called HICRYPT that can be reached through the email address hicrypt@ling.su.se.

The work of this chapter was supported by the Swedish Research Council, grant 2018-06074, DECRYPT – Decryption of Historical Manuscripts https://de-crypt.org/.

References

[1] Friedman, W. F. D., and L. Callimahos, "Military Cryptanalytics, Part I," National Security Agency, United States Government, Washington, DC, 1959 (available through Aegean Park Press, Laguna Hills, CA).

[2] Schmeh, K., *Revisited: A Terminology for Codes and Nomenclators*, 2018, https://scienceblogs.de/klausis-krypto-kolumne/2018/10/07/revisited-a-terminology-for-codes-and-nomenclators/.

[3] Mikhalev, V., et al., "What is the Code for the Code? Historical Cryptology Terminology," in *Proceedings of the 6th International Conference on Historical Cryptology*, 2023, pp. 130–138, https://ecp.ep.liu.se/index.php/histocrypt/article/view/702.

[4] Aldarrab, N., Kevin Knight, and Beáta Megyesi, *The Borg Cipher*, https://cl.lingfil.uu.se/∼bea/borg.

[5] Cipher ID-3816,reproduced image from the Swedish National Archive Riksarkivet 1637, https://de-crypt.org/decrypt-web/RecordsView/189.

[6] Knight, K., B. Megyesi, and C. Schaefer, "The Copiale Cipher," *invited talk at ACL Workshop on Building and Using Comparable Corpora (BUCC)*, Association for Computational Linguistics, 2011.

[7] Key ID-345, Reproduced image from the National Archives in Kew, State Papers. *TNA_SP106/2_ElizabethI_f58(0069)*. 1596. url: https://de-crypt.org/decrypt-web/RecordsView/345.

[8] Key ID-633, Reproduced image from the National Archives in Hungary, G15 Caps. C. Fasc. 44. 01, DECODE ID 633, 1703–1711, https://de-crypt.org/decrypt-web/RecordsView/633.

[9] Megyesi, B., et al. "Keys with Nomenclatures in the Early Modern Europe," *Cryptologia*, 2022, doi: 10.1080/01611194.2022.2113185.

[10] Lasry, G., et al., "Deciphering ADFGVX Messages from the Eastern Front of World War I," *Cryptologia*, Vol. 41, No. 2, 2017, pp. 101–136.

[11] Lasry, G., N. Biermann, and S. Tomokiyo, "Deciphering Mary Stuart's Lost Letters from 1578–1584," *Cryptologia*, 2023, doi: 10.1080/01611194.2022.2160677.

[12] Megyesi, B., et al., "Decryption of Historical Manuscripts: The DECRYPT Project," *Cryptologia*, Vol. 44, No. 6, 2020, pp. 545–559, https://doi.org/10.1080/01611194.2020.1716410.

[13] Megyesi, B., N. Blomqvist, and E. Pettersson, "The DECODE Database: Collection of Ciphers and Keys," in *Proceedings of the 2nd International Conference on Historical Cryptology*, 2019.

[14] Szigeti, F., and M. Héder, "The TRANSCRIPT Tool for Historical Ciphers by the DECRYPT Project," in *Proceedings of the 5th International Conference on Historical Cryptology*, 2022, pp. 208–211, https://ecp.ep.liu.se/index.php/histocrypt/article/view/409/367.

[15] Kopal, N., and B. Esslinger, "New Ciphers and Cryptanalysis Components in CrypTool 2," in *Proceedings of the 5th International Conference on Historical Cryptology*, 2022, pp. 127–136.

[16] Zandbergen, R., *The Voynich Manuscript*, http://www.voynich.nu/.

[17] Pelling, N., *The Cipher Mysteries Blog*, www.ciphermysteries.com.

[18] Schmeh, K., *Cipherbrain*, https://scienceblogs.de/klausis-krypto-kolumne/ (updates on this website stopped end of 2022).

[19] Láng, B., *Real Life Cryptology: Ciphers and Secrets in Early Modern Hungary*, Amsterdam: Atlantis Press, Amsterdam University Press, 2018.

[20] Tomokiyo, S., *Cryptiana: Articles on Historical Cryptography*, http://cryptiana.web.fc2.com/code/crypto.htm.

[21] Antal, E., and P. Zajac, "HCPortal Overview," in *Proceedings of the 3rd International Conference on Historical Cryptology*, 2020, pp. 18–20, doi: 10.3384/ecp2020171003, https://hcportal.eu.

[22] Megyesi, B., "Transcription of Historical Ciphers and Keys," in *Proceedings of the 3rd International Conference on Historical Cryptology*, 2020.

[23] Megyesi, B., and C. Tudor, *Transcription of Historical Ciphers and Keys: Guidelines, version 2.0*, https://cl.lingfil.uu.se/~bea/publ/transcription-guidelines-v2.pdf.

[24] Unicode, The Unicode® Standard Version 12.0–Core Specification, 2019, https://unicode.org/standard/standard.html.

[25] Lasry, G., "Armand de Bourbon's Poly-Homophonic Cipher–1649," in *Proceedings of the 6th International Conference on Historical Cryptology*, 2023, pp. 105–112, https://ecp.ep.liu.se/index.php/histocrypt/article/view/699.

[26] Souibgui, M. A., et al. "DocEnTr: An End-to-End document Image Enhancement Transformer," in *26th International Conference on Pattern Recognition (ICPR)*, 2022.

[27] Axler, G., and L. Wolf, "Toward a Dataset-Agnostic Word Segmentation Method," in *25th IEEE International Conference on Image Processing (ICIP)*, IEEE, 2018, pp. 2635–2639.

[28] Frinken, V., and H. Bunke, "Continuous Handwritten Script Recognition," in *Handbook of Document Image Processing and Recognition* (D. Doermann and K. Tombre, eds.), Springer, 2014, pp. 391–425.

[29] Kang, L., et al., "Pay Attention to What You Read: Non-Recurrent Handwritten Text-Line Recognition," *Pattern Recognition*, Vol. 129, 2022, p. 108766.

[30] Bogacz, B., N. Howe, and H. Mara, "Segmentation Free Spotting of Cuneiform Using Part Structured Models," in *15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, IEEE, 2016, pp. 301–306.

[31] Baró, A., et al., "Towards a Generic Unsupervised Method for Transcription of Encoded Manuscripts," in *Proceedings of the 3rd International Conference on Digital Access to Textual Cultural Heritage*, 2019, pp. 73–78.

[32] Souibgui, M. A., et al., "Few Shots Are All You Need: A Progressive Learning Approach for Low Resource Handwritten Text Recognition," *Pattern Recognition Letters*, Vol. 160, 2022, pp. 43–49, https://doi.org/10.1016/j.patrec.2022.06.003.

[33] Souibgui, M. A., et al., "A User Perspective on HTR Methods for the Automatic Transcription of Rare Scripts: The Case of Codex Runicus," *Journal on Computing and Cultural Heritage*, 2022.

[34] Kopal, N., "Solving Classical Ciphers with CrypTool 2," in *Proceedings of the 1st International Conference on Historical Cryptology*, 2018, pp. 29–38.

[35] Kopal, N., "Cryptanalysis of Homophonic Substitution Ciphers Using Simulated Annealing with Fixed Temperature," in *Proceedings of the 2nd International Conference on Historical Cryptology*, 2019, pp. 107–116.

[36] Megyesi, B., et al., "Historical Language Models in Cryptanalysis: Case Studies on English and German," in *Proceedings of the 6th International Conference on Historical Cryptology*, 2023, pp. 120–129. url: https://ecp.ep.liu.se/index.php/histocrypt/article/view/701.

[37] Waldispühl, M., "Variation and Change," in *The Cambridge Handbook of Historical Orthography* (M. Condorelli and H. Rutkowska, eds.), Cambridge University Press, 2023, pp. 245–264.

[38] Pettersson, E., and B. Megyesi, "The HistCorp Collection of Historical Corpora and Resources," in *Proceedings of the Digital Humanities in the Nordic Countries 3rd Conference*, March 2018.

[39] Kopal, N., and M. Waldispühl, "Two Encrypted Diplomatic Letters Sent by Jan Chodkiewicz to Emperor Maximilian II in 1574–1575," in *Proceedings of the 4th International Conference on Historical Cryptology*, 2021, pp. 80–89, doi: https://doi.org/10.3384/ecp188409.

[40] Pettersson, E., and B. Megyesi, "Matching Keys and Encrypted Manuscript," in *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, October 2019, pp. 253–261.

[41] Gambardella, M.-E., B. Megyesi, and E. Pettersson. "Identifying Cleartext in Historical Ciphers," in *Proceedings of the Workshop on Language Technologies for Historical and Ancient Languages, LT4HALA 2022*, 2022.

[42] Waldispühl, M., and B. Megyesi, "Language Choice in Eighteenth-Century Diplomatic Ciphers from Europe," in *Languages of Diplomacy in the Eighteenth Century* (V. Rjéoutski and G. Kazakov, eds.), Amsterdam University Press, 2023.

[43] Waldispühl, M., "Verschlüsselte Briefe: Mehrsprachigkeit und Geheimschrift im Schwedischen Reich," in *Praktiken der Mehrsprachigkeit im Schwedischen Reich (1611–1721)* (M. Prinz and D. Stoeva-Holm, eds.), Harrassowitz, 2023.

[44] Kahn, D., "The Future of the Past—Questions in Cryptologic History," *Cryptologia,* Vol. 32, 2008, pp. 56–61.

[45] Mrayati, M., Y. MeerAlam, and M. Hassan at-Tayyan, eds., *The Arabic Origins of Cryptology*, Volumes 1–6, KFCRIS & KACST, 2003–2006.

[46] Lasry, G., "Deciphering a Letter from the French Wars of Religion," in *Proceedings of the 5th International Conference on Historical Cryptology*, 2022, pp. 147–152.

[47] Braun, G., and S. Lachenicht, eds, *Spies, Espionage and Secret Diplomacy in the Early Modern Period*, Kohlhammer, 2021.

[48] Bullard, M. M., "Secrecy, Diplomacy and Language in the Renaissance," in *Das Geheimnis am Beginn der europäischen Moderne*, G. Engel, et al. (eds.), Klostermann, 2002, pp. 77–97.

[49] Desenclos, C., "Unsealing the Secret: Rebuilding the Renaissance French Cryptographic Sources (1530–1630)," in *Proceedings of the 1st International Conference on Historical Cryptology*, 2018, pp. 9–17.

[50] De Leeuw, K., "The Black Chamber in the Dutch Republic During the War of the Spanish Succession and Its Aftermath, 1707–1715," *The Historical Journal*, Vol. 42, No. 1, 1999, pp. 133–156.

[51] Lasry, G., B. Megyesi, and N. Kopal. "Deciphering Papal Ciphers from the 16th to the 18th Century," *Cryptologia*, 2020, pp. 479–540, https://www.tandfonline.com/doi/full/10.1080/01611194.2020.1755915.

[52] Kopal, N., and M. Waldispühl, "Deciphering Three Diplomatic Letters sent by Maximilian II in1575," *Cryptologia*, Vol. 46, No. 2, 2022, pp. 103–127, doi: 10.1080/01611194.2020.1858370.

[53] Dinnissen, J., and N. Kopal, "Island Ramanacoil a Bridge too Far. A Dutch Ciphertext from 1674," in *Proceedings of the 4th International Conference on Historical Cryptology*, 2021, pp. 48–57, https://ecp.ep.liu.se/index.php/histocrypt/article/view/156.

[54] Megyesi, B., et al. "Key Design in the Early Modern Era in Europe," in *Proceedings of the 4th International Conference on Historical Cryptology*, 2021.

[55] Megyesi, B., et al. "What Was Encoded in Historical Cipher Keys in the Early Modern Era?" in *Proceedings of the 5th International Conference on Historical Cryptology*, 2022.

[56] Pelling, N., *The Curse of the Voynich: The Secret History of the World's Most Mysterious Manuscript; The Intriguing Story of the People, Places, and Politics Behind the Enigmatic "Voynich Manuscript,"* Compelling Press, 2006.

[57] Kennedy, G., and R. Churchill, *The Voynich Manuscript: The Mysterious Code that Has Defied Interpretation for Centuries*, Rochester, VT: Inner Traditions, 2006.

[58] Kruh, L., "A Basic Probe of the Beale Cipher as a Bamboozlement," *Cryptologia*, Vol. 6, No. 4, 1982, pp. 378–382.

[59] *DECODE Records*, https://de-crypt.org/decrypt-web.

[60] Tomokiyo, S., *Confederate Ciphers During the Civil War: Various Vigenère Keywords*, 2022, http://cryptiana.web.fc2.com/code/civilwar4.htm.

[61] *HistoCrypt–International Conference on Historical Cryptology*, https://histocrypt.org/.

[62] *Cryptologia*, https://www.tandfonline.com/journals/ucry20.