Deciphering papal ciphers from the 16th to the 18th Century

George Lasry, Beáta Megyesi & Nils Kopal

To cite this article: George Lasry, Beáta Megyesi & Nils Kopal (2021) Deciphering papal ciphers from the 16th to the 18th Century, Cryptologia, 45:6, 479-540, DOI: 10.1080/01611194.2020.1755915

To link to this article: https://doi.org/10.1080/01611194.2020.1755915

© 2020 The Author(s). Published with license by Taylor & Francis Group, LLC

Published online: 23 Jun 2020.

Submit your article to this journal

Article views: 2534

View related articles

View Crossmark data
Deciphering papal ciphers from the 16th to the 18th Century

George Lasry, Beáta Megyesi, and Nils Kopal

ABSTRACT

In Meister’s 1906 landmark study, “Die Geheimschrift im Dienste der päpstlichen Kurie von ihren Anfängen bis zum Ende des XVI Jahrhunderts”, the 16th Century papal cryptographic service is described as a vibrant, highly professional organization, at the forefront of the science of cryptography in the Late Renaissance. In his work from 1993, Alvarez concluded that by the 19th Century, “the reputation of papal cryptography, once so lustrous, has sadly faded.” However, until now, very little was known about the evolution of papal cryptography from the 16th to the 18th Century. In this article, we describe how we obtained a large collection of original papal ciphertexts from the Vatican archives, transcribed them, and how we were able to recover most of the keys, and to decipher the original plaintexts using novel cryptanalysis methods and the open-source e-learning CrypTool platform. The recovered keys and decipherments provide unique insights into papal cryptographic practices from the 16th to the 18th Century. The 16th Century is characterized by innovation and a high level of sophistication, with a primary focus on cryptographic security. From the 17th Century, only the simpler but also less secure forms of ciphers remain in use, and papal cryptography significantly lags behind other European states.

1. Introduction

Any research on historical cryptography usually involves a multidisciplinary effort. It covers multiple facets that are critical to achieve a deeper understanding of cryptographic methods and practices at a specific historical place and time. The historical context for the use of encrypted communications and the cultural and political aspects are of primary interest. Research also involves the development of linguistic tools, tools for transcription, as well as computerized algorithms for cryptanalysis. The DECRYPT program was established with the purpose of tackling such large-scale and
multidisciplinary historical decipherment projects (Megyesi, Blomqvist, and Pettersson 2019; Megyesi et al. 2020).1

The study of the papal ciphers in the 16th, 17th, and 18th Centuries poses additional and specific challenges. Some of the material is not easy to find and obtain a copy of. There is no systematic indexing of encrypted material and cryptographic keys, and one often needs to systematically search large volumes of mostly unencrypted material. The quality of some of the material, or of the digital copies, is often problematic.

Fortunately, we were able to locate large amounts of encrypted documents from various time periods. The volume and diversity of the material created the challenge of storing, classifying, and transcribing the data, before cryptanalysis. As many of the cipher methods are idiosyncratic and unique to specific periods and places, new codebreaking techniques needed to be developed to decrypt the various sources. Another challenge is the scarcity of existing sources on the subject; while some aspects are partially covered by existing sources for the 16th Century, there is a lack of studies for the 17th and 18th Centuries.

The study described in this article is an attempt to answer the following questions:

- What types of ciphers were used by the papal cipher secretaries and papal envoys in the 16th, 17th, and 18th Centuries? What were their specific practices and habits?
- How did the methods and practices evolve over time?
- How secure were those ciphers? How secure are they against modern computing and cryptanalysis techniques?
- Is it possible to develop a set of tools for the automated or semi-automated cryptanalysis of those historical ciphers?

This article contributes to insights into the development and usage of crypto systems over time by investigating the papal ciphers and keys from the 16th to the 18th Century.

The remainder of this article is structured as follows: In Section 2, the existing sources on papal cryptography are surveyed. Section 3 presents the main types of cipher types in use by the papal secretaries in the 16th Century, also introducing the terminology used in this article. Section 4 describes the collections of ciphers obtained from the Vatican archives and the process of their digitization and transcription. Section 5 describes the process of recovering the keys and deciphering the encrypted material. Section 6 provides additional details on the codebreaking techniques, and

1https://cl.lingfil.uu.se/decode/about.html
their integration into CrypTool 2. Section 7 summarizes the main findings of this study.

2. Sources on papal ciphers

In this section, the main sources on papal cryptography are surveyed. A comprehensive study of numerous categories of ciphers and variants covering the 16th Century can be found in Meister’s essential “Die Geheimschrift im Dienste der päpstlichen Kurie von ihren Anfängen bis zum Ende des XVI Jahrhunderts”, 1906 (Meister 1906). It also briefly covers the 15th Century, and to a lesser extent, the 14th Century. Meister meticulously collected keys from various Vatican libraries and archives, including the papers of Giovanni Battista Argenti and his nephew Matteo, the highly influential Cipher Secretaries of the popes in the second half of the 16th Century and the beginning of the 17th Century, and reproduced them in his book. A pattern of extreme diversity emerges, and Meister identifies at least 12 types of ciphers using digits (Meister 1906, 69), but those cover only a subset of the keys documented in the appendices. Meister also reproduces a number of treatises on cryptography, including treatises written by the Argenti. Furthermore, he describes a high degree of professionalization involving extensive training in both the creation and the cryptanalysis of ciphers. On the flip side, Meister does not include original ciphertexts in his study. As a result, it is difficult to assess how the ciphers were used in practice and to study the habits of the cipher clerks who encrypted and decrypted the messages.

Enciphered messages from the 16th Century are also mentioned in the Acta Nuntiaturae Gallicae Series, which documents the correspondence between the Papal Curia and the papal nuncios in France. The series include several keys recovered based on original plaintexts found in the collections covered by the series (Blet 1962; Lestocquoy 1972; Toupin 1967, 1984). Additional material regarding the correspondence of nuncios in Portugal may be found in de Witte (1980).

Leighton (1969) describes the decipherment of a 1573 cipher used by a papal envoy in Poland.

The website cryptiana2 contains references to recovered papal ciphers, as well as additional information about Italian ciphers in the 16th Century, which are similar to papal ciphers.

Also, in 2016, several ciphertexts collected as part of the current project were made available as public cipher challenges in MysteryTwisterC3.3 In

---

2 http://cryptiana.web.fc2.com/code/crypto.htm
3 https://www.mysterytwisterc3.org/en/search?search_term=vatican&type=all
2018, two of them, from the years 1625 to 1628, were solved, the keys recovered, and the ciphertexts deciphered. Those samples are the only prior sources of information about papal cryptography in the 17th Century. No sources are available that cover the 18th Century.

Finally, in (Alvarez 1993), Alvarez describes the evolution of papal cryptography in the 19th Century. After a long period of inertia, the Vatican cipher clerks slowly started to adopt newer techniques such as polyalphabetic ciphers, but still lagged behind other European countries, at least until WW1.

3. Papal ciphers in the 16th Century

In this section, the knowledge on 16th Century papal ciphers, prior to the current study, is surveyed, and the various types and categories of ciphers are described. The primary source is Meister (1906), in which hundreds of variants are included. This section also introduces the terminology used throughout this article, and it focuses on ciphers that are composed of digits, possibly enriched with special symbols and markings. While letters or pictographic representations were also used in the 14th and 15th Centuries and the first half of the 16th Century, the vast majority of the papal ciphers in the second half of the 16th Century employ digits.

Meister provides an extensive catalog of keys and systems collected from the Vatican archives. He makes an attempt at classifying the variants, using selected examples, but his classification is not comprehensive enough to cover the large number of variants in his catalog. Instead, the authors propose here a different taxonomy and classification.

In most cases, the ciphertexts consist of continuous sequences of digits, usually without any external clue on how to divide them into separate entities, as illustrated in Figure 1 (Archivio Segreto Vaticano 2016b).

In some ciphertexts, some of the digits have a special marking, such as a dot on top of the digit. In other cases, additional “digit-like” symbols are employed, to enrich the set of regular digits (from 0 to 9). The meaning of the special markings and symbols differs per cipher type. Often, they provide additional clues on how the continuous ciphertext sequence should be split into separate items. A list of common special markings and symbols is given in Table 1.

---

5 Meister 1906, p. 176–460.
6 Meister 1906, p. 69–73.
7 Meister 1906, p. 171-460.
8 F18-1 in the DECODE database: https://cl.lingfil.uu.se/decode/database/record/15
Figure 1. Example of continuous sequences of digits in a ciphertext (Archivio Segreto Vaticano 2016b).
3.1. Plaintext and cipher elements

We introduce here the concept of plaintext elements and cipher elements, the basic entities for enciphering and deciphering.

- When enciphering a plaintext, the plaintext must be first decomposed into distinct plaintext elements, which are each encoded separately. Some ciphers may only encipher letters, others may also encode syllables or words as distinct plaintext elements. Each plaintext element is then encoded and replaced by its corresponding cipher element.
- When deciphering, the ciphertext continuous sequence of digits must first be decomposed into distinct cipher elements, which are then decoded and replaced by their corresponding plaintext elements.

Based on the sources on 16th Century papal ciphers, it is possible to distinguish between three main types of cipher elements:

- Regular elements
- Nomenclature elements
- Nulls

The types of cipher and plaintext elements are described in the following sections.

3.2. Regular elements

Regular plaintext elements (or simply, regular elements) are the most common types of plaintext elements. They usually represent letters as well as the most common syllables and some common prepositions.\(^9\) Not all letters are represented, for example, the letter h is most often omitted, as well as the letter u after qu. Also, u and v are interchangeable and represented by the same regular element. In doubled letters, such as ss or ll, one of the letters is usually omitted.

Regular elements can be encoded using a single digit (e.g., 7), a sequence of 2 digits (e.g., 72) or a sequence of 3 digits (e.g., 721), denoted in this article as regular element codes or regular codes. In some cases, the first digit or the second digit of a 2-digit regular (element) code is marked, e.g., with a dot on top of it. In other cases, it is restricted to specific digits (e.g., only odd digits are allowed, or only even digits, or only a specific digit).

\(^9\)Sometimes, some punctuation marks are also regular elements. In other cases, punctuation marks are nomenclature elements.
Such markings or restrictions assist the deciphering clerk in separating the cipher digits into cipher elements.

Figure 2 shows an example of the mapping of a few regular elements (letters and prepositions) to regular codes.\(^\text{10}\)

### 3.3. Nomenclature elements

Nomenclature (plaintext) elements may include names, places, common words, function words (e.g., prepositions), or syllables. Nomenclature elements are most often encoded using 3 or 4 digits. Those digits are denoted as **nomenclature element codes** or simply **nomenclature codes**. When deciphering a ciphertext, it is necessary to first recognize the nomenclature codes. Nomenclature codes can be marked using a variety of methods:

- One of the digits is marked with a sign, e.g., a dot on top of the second digit, as illustrated in Figure 3.\(^\text{11}\)
- Nomenclature codes are enclosed within nulls.
- They are followed by a null. For example, in a cipher where all regular codes have 2 digits, and nomenclature codes have 3 digits, a null at the end of a subsequence of digits with a noneven length indicates that the last 3 digits represent a nomenclature code.
- They start with a reserved digit. For example, a cipher where nomenclature codes have 4 digits and always start with 2, e.g., 2098 or 2956.
- The second digit (or the first digit, or the third digit) of 3-digit nomenclature codes is a reserved digit, e.g., 0 as in 208 or 309.
- They include a special symbol, e.g., the second digit in a 3-digit nomenclature code is a special symbol.
- The digit right before a nomenclature code has a dot on top.

\(^\text{10}\)Meister 1906, p. 393.

\(^\text{11}\)Meister 1906, p. 393.
Examples of nomenclature codes marked using a dot on top of the second digit are shown in Figure 3.12.

3.4. Nulls

Nulls do not represent any language entity (such as letters, syllables, or words). Instead, nulls have several purposes. One is to confuse the cryptanalyst, who first needs to recognize the nulls and discard them. In some cases, the purpose is to assist the (intended) decipherer. Sometimes, nulls are added by convention after encoding each word. In some ciphers, a null follows each nomenclature code. Some ciphers do not employ nulls.

Nulls are most often encoded using one or more dedicated digits (e.g., all 1’s and 8’s are nulls). In some cases, nulls are represented by reserved pairs of digits (e.g., 21, 22, 23, … , 29).

3.5. Types of ciphers

We distinguish between three primary classes of ciphers, based on how regular elements relate to corresponding regular code(s):

- **Monoalphabetic**
- **Homophonic**
- **Polyphonic**

In **monoalphabetic ciphers**, each regular element can be encoded using only one regular code.

In **homophonic ciphers**, regular elements may be represented by more than one regular code. For example, the letter e may be encoded as either 02 or 20. The set of regular codes that map to a certain regular element is called **homophones**.

In **polyphonic ciphers**, a particular regular code may correspond to more than one regular element. Usually, regular codes in polyphonic ciphers consist of a single digit, which represents two possible letters (and one of the digits may represent a null). The set of letters represented by the same code is known as the **polyphones** of the code. For example, the regular code 4 may represent either the letter t or the letter e. The intended decipherer is expected to recognize the correct polyphone from the context.

We further distinguish between ciphers, as follows:

- In a **fixed-length cipher**, all regular codes consist of the same number of digits, usually 2 digits.

---

12Meister 1906, p. 393.
In a **variable-length cipher**, regular codes may have different numbers of digits, e.g., 1, 2, or 3. For example, the letter e may be encoded as either 2, 21, or 212.

Note that the terms fixed-length and variable-length, even though they qualify a cipher (e.g., fixed-length cipher), refer only to the length of the regular codes. In fixed-length ciphers as defined above, nulls or nomenclature codes may have a different length (e.g., one digit for nulls, and three digits for nomenclature codes). Furthermore, it is not required that all nomenclature codes have the same length (some nomenclature codes could have four digits, and others only three). The same applies for nulls.

### 3.6. Deterministic vs. nondeterministic ciphers

To decipher a ciphertext enciphered using a key with one of the variants described above, the cryptanalyst who does not know the key or the cipher secretary (who does) must first divide the sequence of digits and special signs into separate cipher elements.

We denote the process of decomposing a ciphertext composed of digits and special symbols into separate cipher elements (nulls, regular codes, and nomenclature codes), as **parsing** the ciphertext. The process of replacing the codes with the original (plaintext) elements is denoted as **decoding**.

For any given cipher, the process of **parsing** a ciphertext may either be:

- **Deterministic**, so that parsing the continuous sequence of cipher elements in order to split it into distinct cipher elements (nulls, regular codes, and nomenclature codes) can be implemented in a predictable way.
- Or **nondeterministic**, so that parsing may require a choice between several options at each step of the process.

Parsing of fixed-length ciphers is in most cases deterministic. Parsing of variable-length ciphers is usually nondeterministic, unless some special marking or symbol indicates the beginning of a regular code. Historically, in the case of a nondeterministic cipher, the clerk deciphering the ciphertext would recognize the correct regular code (and corresponding plaintext element), based on the context, choosing from several options.

Similarly, **decoding** may either be deterministic or nondeterministic for a given cipher. Decoding (after correctly parsing) for monoalphabetic and homophonic ciphers is always deterministic. It is nondeterministic for polyphonic ciphers, unless for ciphers where the choice between the polyphones

---

13Unless the digit(s) used for null(s) is also used as part of regular codes, for example.
is indicated using a special marking (e.g., a dot on the bottom of the digit indicates the second out of two choices of regular elements, and the absence of such a dot indicates the first choice).

### 3.7. Cipher schemes

We introduce here the notion of a **cipher scheme**. A cipher scheme characterizes a set of ciphers and cipher keys that are of the same type (e.g., non-deterministic variable-length homophonic) and have the same structure, that is, they have:

- The same structure for the regular codes (e.g., their length, possible markings), while the actual mapping of the regular elements to regular codes differs per each cipher or key that belongs to the same scheme.
- The same structure for nomenclature codes, e.g., all having 4 digits, the first being a dedicated digit, which may differ for different ciphers or keys.
- The same structure for nulls, e.g., a single digit from a set of reserved digits, while the specific reserved digit(s) may differ for different ciphers or keys that belong to the same scheme.

Two ciphers that share the same scheme are essentially the same cipher system, with different keys.

### 4. The Vatican cipher collections

The Vatican City has two main institutes for the collection of manuscripts: the main library of the Vatican and the Secret Archives of the Vatican, where we collected encrypted sources.

#### 4.1. Cipher collections from the secret archives of the Vatican

The Secret Archives of the Vatican, or its official name Archivio Segreto Vaticano, henceforth abbreviated ASV, contain a large collection of preserved papal correspondence related to the government and diplomatic correspondence of the Catholic church from the 8th Century to the 20th Century. The pope grants free access to most of the documents up to 1939 to qualified scholars with a university degree conducting scientific studies, after permission from the Prefect of the ASV.\(^{14}\)

We got the kind permission from the Prefect to visit the archives with our aim to collect as many ciphers (and keys) as possible from various

\(^{14}\)http://asv.vatican.va/content/archiviosegretovaticano/en/consultazione/admission-request.html
countries and time periods in Europe and start building a research infrastructure for historical cryptology. We visited the archives over 2 weeks in 2014 and 2015 and were pointed to one, well-structured index (Indice 1025) by the Prefect containing reference to many folders with encrypted letters and related documents from France, Spain, and Portugal (Segretario di Stato Francia, Spagna, and Portogallo, respectively). The reference to each letter in the index gives neatly written short description of the sources categorized by enciphered letters (Lett. Orig. e cifre), deciphered letters (Registro di Cifre), and the combination of these (Lett. Orig. e decifre). We ordered 45 folders (doss.) in total, each containing between 300 and 600 pages of letters, in which we were able to find encrypted sources and keys, as follows:

- Segr. di Stato Francia [F]: doss. 2, 3, 4, 6, 7, 17, 18, 22, 32, 58, 59, 62, 64, 104, 111, 114, 129, 173, 283C, 346, 392A, 392C
- Segr. di Stato Portogallo [P]: doss. IA, 3, 7, 8, 17, 117

The letters originate from the 16th, 17th, and 18th Centuries, written on paper or parchment. We turned several hundreds of original letters page by page chasing for ciphers and keys. We found that most of the ordered folders contained enciphered sequences from a few cipher symbols to several pages of encrypted sequences. About 1% of the material was actually encrypted. The cipher sequences turned out to be represented by digits, easily distinguishable from the Roman characters in the letters. Once we found a page with a cipher sequence, be it short or long, we put the index reference on our shopping list to order a colored copy of an image with high resolution, and took a note of the metadata as stated in the archive’s index. Because messages with encryption were mixed with letters written in cleartext, we also ordered copies of cleartext letters that were immediately preceding or following the message with encrypted content as these messages might bring some light to the content of the correspondence and help decryption. During the 2 weeks, we were able to collect over 200 encrypted letters and order images of each letter page for private use for ca. 5 Euros each. The scanned high resolution images of 400 dpi were posted to us on CD-ROMs within a few weeks in jpg-format. The images in each folder (doss.) were then manually divided into groups (before decipherment), where we identified each letter with encrypted content and created a cipher record together with the associated metadata given by the archive and the plausible corresponding letters in cleartext.
4.2. Cipher collections from the Vatican Library

Our second target for finding encrypted sources was the Vatican Library, Bibliotheca Apostolica Vaticana, henceforth abbreviated BAV, holding ca. 180,000 manuscripts and 1.6 million books. It is a research library located inside Vatican City, next to the ASV. Admission is granted for researchers with appropriate qualifications and/or relevant scientific publications. Access was easy to get and upon our arrival, a dedicated librarian, whom we contacted beforehand and told about our wish to collect ciphers and keys, waited for us with relevant indexes and pointed us to enciphered manuscripts. Similar to the ASV, the manuscripts are organized in folders containing letters that we had to scan through. In 3 days, we could collect and order images from the library in black-and-white, low resolution of 72 dpi for ca. 5 Euros per image. We chose the letter correspondence of Monsignor Giovanni Battista Pallotto, archbishop of Thessalonica to the emperor referring to the following folders:

- Vaticana Barberiniani Latini-6956 [Barb.lat.6956]: containing 70 encrypted letters from 1628.

The BAV makes its material, consisting of 80,000 codices, accessible through their ambitious digitization project to share it with scholars by making these images available for consultation, at low resolution, on the web. After our order, the library made the two folders publicly available on their website: https://digi.vatlib.it/view/MSS_Barb.lat.6956 and https://digi.vatlib.it/view/MSS_Barb.lat.6960. There are many other folders with encrypted sources, which we are planning to extend our collection with for further studies in the near future. The scanned images were then grouped into cipher records, similar to the encrypted letters from the ASV collection, and some were transcribed.

4.3. The DECODE database

To store and share these and other encrypted sources, we developed a database, called DECODE, aiming at the systematic collection and description of ciphers, keys, and related documents (Megyesi et al. 2019), where these sources are uploaded along with all metadata that have been available to us. Currently we have over 1,100 ciphertexts and keys from Early Modern

---

15https://digi.vatlib.it/
16The DECODE database https://cl.lingfil.uu.se/decode/database/
Times collected from various archives and libraries in Europe. The collected ciphertexts are annotated with metadata schema developed specifically for historical ciphers. Information includes the current location of the manuscript, its provenance, computer-readable transcription, possible decryption(s) and translation(s) of the ciphertext, images, cleartext found in connection to the encrypted source, and any additional materials of relevance to the particular manuscript.

The database comes with a graphical user interface and allows for search in the existing collection for all users and the upload of new encrypted manuscripts by registered users (professionals in historical cryptology) with an account. The DECODE database aims at the systematic collection of ciphertexts to create infrastructural support for historical research in general and historical cryptology in particular. Our hope is that knowledgeable users will contribute to enlarge the database by uploading new material for a growing collection, a monitor corpus of historical ciphers and keys.

In order to process the ciphers, the images are turned into computer-readable format by manually transcribing many sources in the collection. We trained students to transcribe the content of images and developed transcription guidelines (Megyesi et al. 2020). Ciphertexts contain symbol sequences, characters from existing alphabets, digits, special symbols, or a mixture of these. They might contain spaces, or the symbols follow each other one by one without any space to hide word boundaries. Diacritics are usually not marked by special symbols, however, dots or other marks might be used to indicate special codes or code groups. Similar to historical text, punctuation marks are not frequent, sentence boundaries are typically not marked, and capitalized initial letters in the beginning of the sentence are often missing.

Transcriptions are currently available for 270 out of 800 ciphertexts and more data is on the way. The transcription is aiming to represent the original cipher as shown in the image, keeping line breaks, spaces, punctuation marks, dots, underlined symbols, and cleartext words, phrases, sentences, paragraphs, as shown in the original image. Punctuation marks are transcribed and might also appear above or under specific symbols (e.g., dots, commas, underscores). If the mark appears above the symbol, it is transcribed as the symbol, followed by “^” and the specific mark (e.g., dot or comma). If the mark appears under the symbol, it is marked by a “_” placed between the symbol and the mark “.” (e.g., _). Similarly, underlined symbols are marked with “_” immediately following the symbol, except when the whole ciphertext is underlined. Some of the special signs and how they are transcribed in DECODE are listed in Table 1. Uncertain symbols are transcribed with an added question mark “?” immediately following the uncertain symbol. Possible interpretations of a symbol can be
transcribed by using the delimiter “/”. For example, if it is not clear if a symbol represents a 0 or 6, it is transcribed as “0/6?”. An example of a cipher with continuous digits and special signs is shown in Figure 1.

4.4. Cipher collections included in this study

The ciphers included in this study contain encrypted papal letter correspondence with Colonia, France, Germany, Portugal, and Spain between 1538 and 1767, in total 229 texts consisting of nearly 900,000 symbols. Table 2 gives an overview of the ciphers investigated in this article presented with the folder (doss.) and folio (page) as indexed in the archive collection, the folder’s abbreviated name in this article, together with the time period the collection originates from, the number of letters in each folder, and the total number of symbols in the ciphertext in each folder.

5. Recovering the keys

As part of this project, between 24 April and 9 June 2019, 16 keys were recovered, out of a total of 21 keys used to encode messages in the
Table 2. Cipher collections used in this study.

<table>
<thead>
<tr>
<th>Collection/Folder</th>
<th>Abbr. Collection</th>
<th>Pages/Folio</th>
<th>Period</th>
<th>Texts</th>
<th>Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arch. Nunz. Colonia-5</td>
<td>C5</td>
<td>26r</td>
<td>1721</td>
<td>1</td>
<td>300</td>
</tr>
<tr>
<td>Arch. Nunz. Colonia-13</td>
<td>C13</td>
<td>1r, 5r</td>
<td>1656</td>
<td>2</td>
<td>1,800</td>
</tr>
<tr>
<td>Segr. Stato Francia-3</td>
<td>F3</td>
<td>117rv, 128rv 230r, 263r</td>
<td>1552-1554</td>
<td>4</td>
<td>4,100</td>
</tr>
<tr>
<td>Segr. Stato Francia-4</td>
<td>F4</td>
<td>220r-221r</td>
<td>1587</td>
<td>1</td>
<td>2,300</td>
</tr>
<tr>
<td>Segr. Stato Francia-6</td>
<td>F6</td>
<td>234r-236r</td>
<td>1573</td>
<td>1</td>
<td>10,100</td>
</tr>
<tr>
<td>Segr. Stato Francia-7</td>
<td>F7</td>
<td>249r-250v</td>
<td>1574</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Segr. Stato Francia-17</td>
<td>F17</td>
<td>30r</td>
<td>1583-1585</td>
<td>1</td>
<td>300</td>
</tr>
<tr>
<td>Segr. Stato Francia-18</td>
<td>F18</td>
<td>7rv, 206rv, 233r-235v</td>
<td>1585-1586</td>
<td>3</td>
<td>11,300</td>
</tr>
<tr>
<td>Segr. Stato Francia-22</td>
<td>F22</td>
<td>32r-35v, 71r-74v, 87r-90r, 94r-96v, 132r-135v, 137r, 142r-145r, 152r-153v, 160r-162v, 172r-174v, 181r-183v, 189r-192v, 198r-200v, 202r-203r, 249r-250v, 256r-257v, 282r-284v, 289rv, 293r-294r, 298r-299v, 303r-304v, 307r-309v, 313r-315r, 357r-361v, 381r-384r, 385r-386v, 399r-402v, 404v, 466r-467r, 473r-474v, 499v, 502r, 516rv, 522r-523r, 524r-525r, 533r-534r</td>
<td>1588-1589</td>
<td>36</td>
<td>231,800</td>
</tr>
<tr>
<td>Segr. Stato Francia-64</td>
<td>F64</td>
<td>22r-31v, 38r-42v, 49r-51v, 56rv, 60rv, 64rv, 68rv, 93r-100v, 119r-128r, 130rv</td>
<td>1625</td>
<td>10</td>
<td>130,500</td>
</tr>
<tr>
<td>Segr. Stato Francia-62</td>
<td>F62</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Segr. Stato Francia-283C</td>
<td>F283C</td>
<td>89r-90r</td>
<td>1580</td>
<td>1</td>
<td>2,800</td>
</tr>
<tr>
<td>Segr. Stato Francia-346</td>
<td>F346</td>
<td>71r, 102rv, 109rv, 129r</td>
<td>1632-1634</td>
<td>4</td>
<td>7,800</td>
</tr>
<tr>
<td>Segr. Stato Portogallo-1A</td>
<td>P1A</td>
<td>22r-27r</td>
<td>1535-1536</td>
<td>1</td>
<td>11,000</td>
</tr>
<tr>
<td>Segr. Stato Portogallo-8</td>
<td>P8</td>
<td>92r-95r, 98r-101r</td>
<td>1579-1581</td>
<td>2</td>
<td>4,200</td>
</tr>
<tr>
<td>Segr. Stato Portogallo-117</td>
<td>P117</td>
<td>95rv, 98r, 101r, 103r, 104r, 106r, 115rv, 280r</td>
<td>1757-1761</td>
<td>8</td>
<td>7,900</td>
</tr>
<tr>
<td>Segr. Stato Spagna-1A</td>
<td>S1A</td>
<td>70r-73v</td>
<td>1538-1548</td>
<td>1</td>
<td>7,000</td>
</tr>
<tr>
<td>Segr. Stato Spagna-1</td>
<td>S1</td>
<td>165r, 236r, 249r-250r, 301rv, 314rv, 324r</td>
<td>1566-1571</td>
<td>6</td>
<td>7,900</td>
</tr>
<tr>
<td>Segr. Stato Spagna-6 I</td>
<td>S6/I</td>
<td>6r, 42r, 55r, 81r, 88r-89v, 136r, 169rv, 291rv, 306r</td>
<td>1568-1570</td>
<td>9</td>
<td>9,600</td>
</tr>
<tr>
<td>Segr. Stato Spagna-6 II</td>
<td>S6/II</td>
<td>381r, 397r, 424r, 443r, 474r</td>
<td>1569-1572</td>
<td>13</td>
<td>11,900</td>
</tr>
</tbody>
</table>
collections. The keys were recovered using a variety of cryptanalytic methods, from known-plaintext to ciphertext-only. The process of recovering the key involved several phases:

- Clustering the ciphertexts into groups of ciphertexts enciphered using the same key.
- Recovering the keys of fixed-length homophonic ciphers.
- Recovering the keys of variable-length homophonic ciphers.
- Recovering the keys of polyphonic ciphers.

| Table 2. Continued. |  |
|---|---|---|
| **Segr. Stato Spagna-304** | S304 | 485rv |
| 28rv, 37v, 67r—73v, 101r-105v, 125r-127v, 184rv, 190r-191r, 194r-195r, 222r-224r, 226r-228v, 230r-231r, 262r-264r, 266rv, 268r-269r, 300r-302v, 304r-308r, 320r-322v, 328r-333r, 352r-354v, 376rv, 384rv, 386r-387v, 390r-391v, 398r-399r, 418rv, 426r-427v, 446rv, 448r-449r, 454r-455r, 456rv, 464r-465v, 478r-479v | 1767 | 1 | 1,100 |
| **Segr. Stato Spagna-364C** | S364C |  |
| 12r-15r, 24v, 47r-51r, 53v, 63r-66v, 133r, 139r, 145rv, 147rv, 181r-183v, 235r, 266rv, 288rv, 302r-312r, 314v, 320rv, 322r-324v, 332r-333r, 340r-341r, 344r-344v, 346r, 348r, 358r-360r, 362v | 1717 | 32 | 104,000 |
| **Segr. Stato Spagna-364D** | S364D |  |
| 281r-283r, 294rv, 297r, 300rv, 374r, 384rv, 388rv, 391r, 415rv, 491r-492v, 510r-511v | 1718-1720 | 24 | 86,000 |
| **Segr. Stato Spagna-423** | S423 |  |
| 2r-3v, 6r, 8r-9r, 13r, 15r, 17r-18v, 21r, 23r-24v, 29rv, 32r, 34rv, 42rv, 63rv, 72r-73v, 76r, 82r, 83r, 85r-86v, 89r, 91r-94r, 96r-97v, 102r-105r, 115r, 116r-117v, 118r-120v, 125r-126v, 132r-133v, 136rv, 138r-142r, 144rv, 146rv, 148rv, 149rv, 157r-158r, 160rv, 162r-163v, 166rv, 168r-169v, 172r-173r, 175rv, 178r-179r, 182r, 184rv, 186r-187v, 191rv, 193r, 195r-196v, 199r-202r, 206r, 209r-210v, 213rv, 229r-230v, 233r-234r, 236r-237r, 240rv, 242rv | 1736 | 11 | 17,900 |
| **Vaticana Barberiniani Latini-6956** | 86956 |  |
|  | 1628 | 56 | 227,800 |
The phases are described in the following subsections, as well as a preliminary analysis of the remaining unsolved ciphers.

### 5.1. Clustering the ciphertexts

The first phase of the project consisted of dividing all transcripts into distinct clusters so that all ciphertexts in a cluster represent plaintexts enciphered with the same key. To do so, a key Clustering Algorithm (see Section 6.3) that is based on the frequencies of the digits (0 to 9) was developed and applied to all ciphertext transcripts.

In most cases, the key Clustering Algorithm found a one-to-one matching between the original Vatican collections and the resulting clusters, with some exceptions, marked in color in Table 3. For example:

- The transcripts of the Segr. Stato Francia-22 collection [F22] are split into two clusters. One of them is the cluster that also contains most of the transcripts from the Segr. Stato Francia-18 collection [F18].
- All the transcripts from the collections Segr. Stato Spagna-1 [S1], Segr. Stato Spagna-6 I [S6/I], and Segr. Stato Spagna-6 II [S6/II] are clustered together.
- All the transcripts from the collections Segr. Stato Spagna-364C [S364C] and Segr. Stato Spagna-364D [S364D] belong to the same cluster.

The list of key clusters and their associated collections is given in Table 3. Interestingly, after the project was completed and most of the keys recovered, the assumption that each cluster represents a unique key (and vice-versa) was found to be correct.

### 5.2. Recovering the keys of fixed-length homophonic ciphers

Two collections, B6956\(^{17}\) and F64\(^{18}\), had previously been presented as public challenges in the crypto-challenge site MysteryTwisterC3 [Part I, Part II]\(^{19}\) and solved. Details about how the solutions were found are given in the website cryptiana\(^{20}\) by Satoshi Tomokiyo, who also recovered the keys. The two collections date from 1625 to 1628 and they use fixed-length homophonic ciphers with the same scheme. Two digits are reserved as nulls, regular codes have 2 digits, and nomenclature codes have 3 digits, always followed by a null.

\(^{17}\)B6956 in DECODE database: https://cl.lingfil.uu.se/decode/database/record/215

\(^{18}\)F64 in DECODE database: https://cl.lingfil.uu.se/decode/database/record/62

\(^{19}\)https://www.mysterytwisterc3.org/en/search?search_term=vatican&type=all

\(^{20}\)http://cryptiana.web.fc2.com/code/crypto.htm
The recovery of those keys and the deciphering of the ciphertexts provided important insights into the cryptographic practices in the Vatican, such as the spelling of words, and the use of syllables and prepositions as
The decipherments also provided for a corpus of regular elements, from which a language model with \( n \)-gram statistics could be generated.\(^{21}\)

The next phase consisted of recovering keys for additional fixed-length homophonic ciphers, S364, S304, P8, and F346,\(^ {22}\) based on plaintexts that were written next to the corresponding ciphertexts. While this process is obviously easier than recovering a key from ciphertext-only, this is not an easy task. The challenges involved in recovering a key from plaintext are described in Section 6.1. The aforementioned fixed-length homophonic ciphers use schemes that are different from the scheme used by B6956 and F64. Two of them, S364 and S304, were found to share the same scheme (1 null digit, regular codes with 2 digits, nomenclature codes with 4 digits, and starting with a reserved digit).

Next, an attempt was made to find additional collections matching any of the schemes identified so far, but with different parameters, using a newly developed tool for Parsing Tests (see Section 6.4). This tool tests various schemes for parsing the ciphertexts, with all possible values for parameters such as the digit(s) reserved as null(s). After discarding nulls and nomenclature code, the sequences of regular codes are evaluated using statistical measures such as their index of coincidence.

As a result of this analysis, S423 was identified as having the same scheme as S364 and S304. The next challenge was to recover the assignments of the 2-digit regular codes (homophones). A new Automated Homophones Recovery algorithm (see Section 6.5) was implemented for that purpose. It uses simulated annealing to find optimal assignments of homophones so that, after decoding the ciphertexts, the resulting sequence of (regular) elements, e.g., letters or syllables, is as close as possible to an \( n \)-gram model built from B6596 decoded elements. With this method, most of the codes for the regular elements of S423 were recovered correctly, and the ciphertexts were deciphered.\(^ {23}\) Some matching plaintexts in the collections could then be identified, allowing for further corrections. The same process was successfully applied to P117.\(^ {24}\)

---

\(^{21}\)This language model differs from regular word-based or letter-based language models that are generated from an Old Italian corpus (Pettersson and Megyesi 2018). Regular models do not reflect the actual spelling (before encoding, e.g., the removal of doubled letters), and the use of regular elements which are not single letters, such as syllables, 3-grams, or prepositions. Furthermore, preposition regular elements may be used to encode parts of words, e.g., ‘con’ as the prefix of ‘confidente’, thus encoding as con-f-i-de-n-te. In Italian, ‘con’ is both a prefix, and a preposition (i.e., with).

\(^{22}\)P8, P346, and S364 were recovered more or less at the same time (April 24th 2019). S304 was recovered two weeks later (May 7th 2019).

\(^{23}\)April 24th 2019

\(^{24}\)April 24th 2019
The key for F283 was recovered from ciphertext only,\textsuperscript{25} using a Digit Clustering Algorithm (see Section 6.6). The odd digits (1, 3, 5, 7, and 9) were found to have common statistical attributes that differ from those of the even digits (0, 2, 4, 6, and 8). Inspired by Albert Leighton’s work on another Vatican cipher, it was hypothesized that only an odd digit might be allowed as the second digit of a 2-digit regular code. After parsing the ciphertexts under this assumption, the Automated Homophones Recovery algorithm was applied (see Section 6.5), and most of the key elements could be recovered. Matching plaintexts were then located in the collection images so that the key could be validated and corrected.

The key for F18 was found using a new Interval Analysis algorithm (see Section 6.7) that looks at the distribution of the intervals between two consecutive occurrences of the same digit.\textsuperscript{26} The digit 8 was found to have a preponderance of even-length intervals, mostly 2, 4, 6, and 8. This indicates that 8 is probably a null and that the regular codes are composed of 2 digits. After parsing the digits under those assumptions, the Automated Homophones Recovery algorithm was applied, and the key was recovered. The key was later validated (and improved) using plaintexts that matched the decipherments. The key was also found in two sources (Meister 1906, 324/2; Blet 1962).

Finally, the key C13 was recovered from plaintexts.\textsuperscript{27} C13 has a scheme similar to B6956 and F64, except that nulls are formed of two digits vs. a single digit for B6956 and F64.

\subsection*{5.3. Recovering the keys of variable-length ciphers}

Next, attempts were made to apply the algorithms mentioned in the previous section on the remaining ciphers, without success. It became clear that those were employing different schemes. One hypothesis was that some of the ciphers might use a variable-length scheme.

Keys for variable-length homophonic ciphers are more difficult to recover, and they are resilient to attacks against fixed-length homophonic ciphers. In this project, the recovery of keys for variable-length homophones could only be achieved from matching plaintexts or by finding the key in the sources (e.g., Meister), based on characteristics identified via statistical analysis.

\textsuperscript{25}May 5th 2019
\textsuperscript{26}May 18th 2019
\textsuperscript{27}Much later in the process, on June 8th 2019
First, the key for F17 was recovered, from a short matching plaintext.\(^{28}\) The key was also found in (Toupin 1967, 27). As with this cipher, the first digit of 2-digit regular codes is usually marked with a dot on top, parsing is in theory deterministic, although in practice many dots in the ciphertexts could not be identified, or were missing, probably due to the poor state of the manuscript.

Next, the key for F3 was recovered, also based on matching plaintext.\(^{29}\) Parsing for this cipher is not deterministic, as there is no way to (automatically) determine whether a given digit is part of a 1-digit or 2-digit regular code.

The key for F6 was recovered from a small fragment of matching plaintext.\(^{30}\) Parsing for this cipher is deterministic, as the second digit of a 2-digit regular code is always 2 with a dot on top. Interestingly, it is likely that this cipher is not homophonic, but rather, monoalphabetic. Every regular element (e.g., a letter or syllable) is mapped to only one 2-digit code, and there seem to be no homophones.

The key for F22 presented a special challenge as, despite the availability of large amounts of ciphertext material, no matching plaintext could be found. A new Homophone Detection algorithm (see Section 6.8) was developed to identify possible homophones, that is, two codes with 1, 2, or 3 digits that likely represent the same regular element (e.g., the same letter), based on their context. The algorithm identified several pairs of likely homophones, such as $26 = 62$, $2 = 212$, and $19 = 76$. The first pair looked quite promising, as in other ciphers, a code with 2 digits (e.g., 26) and the same digits in reverse order (62) are often homophones, to make enciphering easier. The analysis also indicated that the digit 8 might be a null.

In addition, a tool for the analysis of nomenclature codes (see Section 6.9) was developed to identify the likely structure of the nomenclature codes, for those ciphers where the nomenclature codes are marked by special signs. Some likely candidates for F22 nomenclature codes were identified using this tool.

A search in the sources produced one cipher key with the expected characteristics (likely pairs of homophones, 8 as the null, and the expected structure of nomenclature codes), found in (Meister 1906, 393/40). The metadata in Meister also matched the metadata for the collection (date, place, and names). A (tedious) manual decipherment of a sequence further validated the hypothesis that the key is indeed the correct one.\(^{31}\) For this

---

\(^{28}\) May 24th 2019

\(^{29}\) May 26th 2019

\(^{30}\) May 27th 2019

\(^{31}\) May 27th 2019
cipher, parsing is not deterministic, as no clues are given on how to separate the variable-length regular codes correctly. Furthermore, nulls are not used systematically, so the distance between 2 nulls (or between a null and a nomenclature code) might be long, resulting in long subsequences of regular codes. The number of possible decipherment option raises exponentially w.r.t. the length of the subsequence of regular codes. A new tool for the Probabilistic Decoding of Variable-Length Homophones (see Section 6.10) was developed, to suggest likely decipherment options, based on an Old Italian language model. However, when applied to F22, this algorithm often fails to produce sound suggestions for subsequences, in most cases due to numerous errors in the transcription of poorly preserved manuscripts.

Finally, the key to the last variable-length homophonic cipher, F22b, was recovered from plaintext.\textsuperscript{32} Parsing for this cipher is not deterministic.

\section*{5.4. Recovering the keys of polyphonic ciphers}

The key for S6 was recovered using matching plaintext.\textsuperscript{33} At first, all attempts to match pairs of digits to specific letters under the hypothesis of a homophonic cipher failed. It became evident that new hypotheses and cipher schemes needed to be tested. One of the schemes often mentioned in Meister refers to polyphonic ciphers (Meister 1906), in which 1-digit regular codes may represent two possible 1-letter regular elements, and one of the 10 digits is usually reserved as the null. Using this assumption, the key to S6 was successfully recovered from the matching plaintext.

The parsing of polyphonic ciphers (with single digit regular codes) is deterministic and straightforward, provided the digit assigned as null and the structure of the nomenclature codes are known. Decoding, however, is ambiguous, as there are two choices per each regular code. A new tool for the Probabilistic Decoding of Polyphones (see Section 6.11) was developed to generate suggestions for the decipherment of subsequences of regular codes. This algorithm uses a language model (\textit{n}-grams of letters) and a dictionary to rank possible decipherments. In S6, nulls are frequently used, inserted usually after each word. As a result, the number of possible decoding options for subsequences of digits is limited, and most subsequences in S6 could be successfully decoded.

Finally, an Automated Polyphonic Key Recovery algorithm (see Section 6.12) was developed, to recover the key of a polyphonic cipher, from ciphertext only, based on a language model (\textit{5}-grams of letters). This

\textsuperscript{32}June 3rd 2019

\textsuperscript{33}June 1st 2019
algorithm was tested on S6, and could reliably reproduce a key identical to
the one recovered manually from plaintext.

This algorithm was then applied to the ciphers that had remained
unsolved. Luckily, it produced successful results for F4. The recovered
key could also be located in Meister (1906, 345/21). Further analysis
showed that decoding is deterministic, as a dot under a certain digit indi-
cates that the second polyphone mapped to that digit is to be selected
(and the absence of a dot indicates that the first polyphone is to be selected).

5.5. Unsolved ciphers

All the algorithms developed so far failed to produce any significant results
that might lead to the recovery of the keys for 3 remaining ciphers.
Furthermore, no matching plaintexts could be found in those collections.
The ciphertexts from those collections (C5, P1, and S1) were therefore
offered as challenges in MysteryTwisterC3.

P1 was solved on 22 August 2019 by Thomas Bosbach, who was able to
locate matching plaintexts in a book (de Witte 1980). P1 turned out to be
a nondeterministic monoalphabetic substitution cipher with variable-length
regular codes.

C5 and S1 are still unsolved at the time of publication.

C5 stands out from all other ciphers, as its cipher elements are separated
using commas, and parsing is straightforward. Almost all the codes have 2
digits, except a few (probably nomenclature codes) that have 3 or 4 digits.
Attempts to map the 2-digit codes (probably the regular codes) to their
corresponding plaintext elements using the Automated Homophones
Recovery algorithm described in Section 6.5 failed. This might be due to
the very short length of the ciphertext or to the original plaintext not being
in Italian.

S1 seems to contain texts enciphered using two different keys. The
ciphertexts in the first part (IA-1) include a large number of pairs of digits
starting with 8 and preceded by a digit with a dot on top. This pattern also
appears in a polyphonic cipher described in Meister (1906, 176/2), used by
Mons. Poggio, the papal nuncio to Spain at the same period; however,
atttempts to decipher the ciphertexts using that key did not produce any
result. But if the scheme is similar to the scheme for the key in Meister
(1906, 176/2), that would imply that the first part of S1 is a variable-length

34June 9th 2019
vatican-challenge-part-5
polyphonic cipher, a unique case in the Vatican collections in DECODE (other polyphonic ciphers use fixed-length 1-letter codes). The ciphertexts in the second part (IA-2) seem to belong to another key.

5.6. Summary

The classifications of the keys are listed in Table 4, together with the status of their key recovery (see legend shown in Table 5). Table 6 summarizes the number of keys solved using various methods, as well as those solved not as part of the project, or unsolved.
6. New cryptanalytic tools and algorithms

In this section, we present the tools we developed so far in this project. Most of the tools were implemented as console applications and then used for analyzing the ciphertexts. To allow for a more user-friendly access to our tools, we started implementing these in the open-source software CrypTool 2 (Kopal et al. 2014). At the end of each tool section, we also show the specific implementation in CrypTool 2. Some tools are already integrated in CrypTool 2, others will be integrated in the near future.

6.1. Recovering keys from matching plaintexts

Several of the keys for the Vatican collections were recovered from plaintexts, in a mostly manual, nonautomated process. Recovering the key from matching plaintext should in principle be easier than recovering the key from ciphertext only. Several challenges are involved in that process, however.

In theory, plaintext matching some ciphertext may originate from three possible sources:

- **Historical cryptanalyst files and worksheets**, where plaintext guesses are inscribed, together with statistics and probable key assignments. This scenario is not relevant to the Vatican collections in DECODE, as the senders and recipients had full knowledge of the relevant keys.

---

**Table 5. Legend of the classification of keys and status of key recovery.**

<table>
<thead>
<tr>
<th>Condition</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solved independently by N. Biemann and T. Bosback</td>
<td>Green</td>
</tr>
<tr>
<td>Solved from ciphertext only</td>
<td>Green</td>
</tr>
<tr>
<td>Reconstructed from plaintexts</td>
<td>Blue</td>
</tr>
<tr>
<td>Key found in Meister based on homophone analysis</td>
<td>Purple</td>
</tr>
<tr>
<td>Unsolved</td>
<td>Red</td>
</tr>
</tbody>
</table>

**Table 6. Key recovery summary.**

<table>
<thead>
<tr>
<th>Key Recovery Type</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solved from ciphertext only</td>
<td>5</td>
</tr>
<tr>
<td>Reconstructed from plaintexts</td>
<td>10</td>
</tr>
<tr>
<td>Key found in Meister based on homophone analysis</td>
<td>1</td>
</tr>
<tr>
<td>Solved independently by N. Biemann and T. Bosbach</td>
<td>3</td>
</tr>
<tr>
<td>Unsolved</td>
<td>2</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>21</strong></td>
</tr>
</tbody>
</table>
In the sender files, where a copy of the original plaintext is kept, together with the enciphered version. The Vatican collections in DECODE, however, mostly consist of files kept by the recipient (see below).

In the recipient files, and created as the cipher secretary deciphers the ciphertext and writes the decoded plaintext in one of the following forms:

- On the same page as the ciphertext, on top of the relevant line.
- In a separate paragraph on the same page as the ciphertext. It can often be distinguished from possible cleartext (text not intended to be enciphered), as the handwriting is different, usually less neatly written, and possibly with corrections and other markings.
- On separate pages. It can also be distinguished from cleartext, based on the handwriting style, as messages sent in cleartext are usually neat and orderly.

Figure 4 shows an example of deciphered plaintext written on top of ciphertext (Archivio Segreto Vaticano 2016a). Figure 5 shows an example of deciphered plaintext in a separate paragraph (Archivio Segreto Vaticano 2016d). For comparison, Figure 6 shows cleartext (not enciphered) in the same collection (F283) (Archivio Segreto Vaticano 2016c). The handwriting style is clearly different.

If the plaintexts are separated from the ciphertexts, it is often difficult to identify those that match ciphertexts, as long as the key is not yet known. In this research, matching plaintexts could often be identified only after the key was recovered from ciphertext only (e.g., F18, F283, P117, S423). In such a case, the matching plaintext is still useful in order to validate and correct the recovered keys. In other cases, no matching plaintexts could be found, even after recovering the key from ciphertext only (e.g., F4) or from one of the sources (e.g., F22).

---

36 F17-1 in the DECODE database: [https://cl.lingfil.uu.se/decode/database/record/13](https://cl.lingfil.uu.se/decode/database/record/13)
37 F283C-1 in the DECODE database: [https://cl.lingfil.uu.se/decode/database/record/72](https://cl.lingfil.uu.se/decode/database/record/72)
38 F283C-1/50v in the DECODE database: [https://cl.lingfil.uu.se/decode/database/record/72](https://cl.lingfil.uu.se/decode/database/record/72)
A commonly-used method to match plaintext and ciphertext is to look in the files for a plaintext page right before or right after a ciphertext page. Another method to identify a potentially matching ciphertext is to count the number of plaintext symbols, and divide this number by the number of...
ciphertext symbols. This ratio can be compared with the ratio computed from other texts with known keys.

In most cases, however, there is no choice other than simple trial-and-error and attempting to associate plaintext elements with corresponding codes in the ciphertext. It would naturally make sense to start the process...
at the beginning of the plaintext or ciphertext or at their ending. However, the beginning and the ending of the plaintext are often hard to read, with acronyms or signatures.

Another technique is to look for repeated sequences of digits in the ciphertext. Unless the cipher is polyphonic, those repeated digits should translate into repeated words or sequences of letters in the plaintext.

Alternatively, we can first look for repeated words or expressions in the plaintext. If the cipher is not homophonic, those should translate into matching repeated sequences of ciphertext digits. With homophonic ciphers, the plaintext repetitions would likely result in similar ciphertext sequences, with some differences due to the use of different homophones for the same letter.

Another (and simpler) technique consists of first identifying the structure of the nomenclature codes, which might be marked using specific signs. The next step consists of looking for nomenclature codes in the ciphertext (including repetitions), and trying to match them with names or locations in the plaintext. For example, the acronym N.S.\textsuperscript{39} is used frequently in the Vatican collections, and so is the preposition ‘che’ (which is most often a nomenclature element). Those nomenclature codes can then serve as ‘anchors’, so that the texts/codes right before and right after them are the next candidates for matching. As an alternative, some frequent prepositions or syllables (like Italian ‘di’, ‘of’ in English) can also serve as anchors.

The process requires a fair amount of trial-and-error, making educated guesses and trying them out. Several hypotheses w.r.t. the scheme of the cipher (fixed or variable length, homophonic or polyphonic, the structure of nomenclature codes, and the values for the null(s)) need to be tested. The process is obviously harder if a cipher scheme not (yet) encountered is being used.

Plaintext of historical sources often includes abbreviations, titles, spelling variations, and code-switching from one language to another. We cannot be certain that the corresponding ciphertext follows the same principles in these particular cases.

The main challenge, however, often has to do with the difficulty of transcribing the plaintext handwriting, which, as mentioned, is often messy or unreadable, and requires philological expertise of the particular language and time period the message was written. Transcribing the plaintext handwriting can be no less difficult than recovering the key from ciphertext only.

\textsuperscript{39}Abbreviation for Nostra Sua, ‘our own’, the pope.
6.2. Deciphering ciphertexts with known key

At first, deciphering (parsing and decoding) was implemented using a console application. A more user-friendly CrypTool 2 component was later implemented for that purpose.

6.2.1. The DECRYPT decipherer component in CrypTool 2

This component allows for the automatic deciphering of most of the ciphers presented in this work. The DECRYPT decipherer takes as input a ciphertext as well as a key, which both follow the DECRYPT transcription guidelines (Megyesi et al. 2020). Also, the parser for tokenizing the ciphertext and splitting it into the correct cipher elements is selected as a component setting. We already implemented 13 different parsers, of which some are created for a specific cipher type, e.g., an F4 parser for ciphertexts based on the F4 cipher. Other parsers implement a more generic cipher scheme and parameters such as the digit used as a null can be specified in the key file.

After that, the component decodes the parsed cipher elements based on the given key. Figure 7 shows a screenshot of the DECRYPT decipherer which successfully decrypts the C13-1 cipher of the DECODE database. The decipherer also shows additional information from the transcription file metadata, e.g., the date of transcription, the number of tokens (cipher elements), and more. Appendix A shows different ciphertext/plaintext decryption screenshots, which were also generated with the DECRYPT decipherer.

6.3. Key clustering

The idea for the key Clustering Algorithm used in this project came from observing the relative frequencies of monograms (the digits 0 to 9, as well as of special symbols) in ciphertexts. The monogram frequencies turned out to be similar for most ciphertexts in the same original collection, with a few exceptions. For illustration purposes, the frequencies of the digits for several transcripts from the Segr. Stato Francia-22 collection are shown in Table 7.

It can be seen that the frequencies of the digits are very similar for all cryptogram transcripts except for 32 and 33, for which the digit frequencies are similar but differ from the digit frequencies of the other transcripts. It should be noted that after the keys were recovered for all the cryptograms in this collection, the key used for those two cryptograms (F22b) indeed turned out to be different from the key for the rest of the cryptograms (F22).

The Clustering Algorithm is a variant of the standard K-Clustering Algorithm, using a self-adapting number of clusters $K$, and works as follows:
1. As initial clusters, assume that each (original) collection consists of texts enciphered using a distinct and unique key for that collection and that each such collection is a distinct cluster. In other words, the archive collection groupings form the initial clusters.

2. For each cluster \( c \), compute the total number of occurrences of each monogram \( i \) (a digit or special symbol), as well as \( f_{c,i} \), their relative frequencies.

3. For each ciphertext \( y \):
   a. Compute the relative frequencies \( f_{y,i} \) of the monograms for the ciphertext.
   b. For each cluster \( c \) (either the cluster that \( y \) currently belongs to, or any other cluster), compute a matching score \( M_{y,c} = \sum f_{c,i} \cdot f_{y,i} \).
   c. Assign ciphertext \( y \) to the cluster \( c \) for which \( M_{y,c} \) is the highest, provided this highest \( M_{y,c} \) is above a given threshold \( M_{\text{min}} \).
   d. If \( M_{y,c} < M_{\text{min}} \) for all clusters, create a new cluster and assign ciphertext \( y \) to it.

4. Repeat process steps (2) and (3), stopping if no ciphertexts have moved from one cluster to another and no new clusters have been created.

Figure 7. Screenshot of the DECRYPT Decipherer component in CrypTool 2.
The clustering process is illustrated in Figure 8 using a theoretical simplified example with three types of ciphers that use only the digits 1, 2, and 3. In hindsight, after recovering the keys for most of the ciphertexts in the clusters, those clusters were found to be fully accurate, that is, all the ciphertexts in a given cluster can be deciphered with the same key, and the recovered keys are different for any two distinct clusters.

6.3.1. A DECRYPT key clusterer component in CrypTool 2
This component takes as input ciphertexts that are transcribed using the DECRYPT transcription guidelines. It clusters the given input texts based on the aforementioned algorithm. The threshold $M_{min}$ can be configured before clustering. Figure 9 shows a screenshot of the DECRYPT key clusterer component in CrypTool 2. The first column shows the name of the cluster, which is by default the name of the first transcription file assigned to the cluster. The second column shows the number of transcription files currently assigned to the cluster. The third column shows the total number of symbols in the cluster transcripts. The fourth column shows the number of distinct symbols in the cluster.

The last column shows the frequencies of the symbols in the cluster. For example $0 = 24$ means that 24% of all symbols in the cluster are zeros.

6.4. Parsing Tests
Parsing Tests can be applied after one or more schemes have been identified (e.g., from collections for which there was a matching plaintext) to check whether a cluster of ciphertexts was encrypted with a key using one of the known schemes.
Two examples of schemes that were identified early on are as follows:

- Fixed-length homophonic with 2-digit regular codes, 2 digits reserved as nulls, and nomenclature codes with 3 digits, always followed by a null. This scheme is employed in B6956 and F64.
- Fixed-length homophonic with 2-digit regular codes, 1 digit reserved as the null, and nomenclature codes with 4 digits, starting with a designated digit. This scheme is employed in S364 and S304.

The hypothesis was that other keys might use one of those known schemes, possibly with different parameters, e.g., the digit(s) used as nulls. To validate the hypothesis, Parsing Tests, as described below, were applied to the clusters for which the key was still unknown.

The Parsing Tests work as follows:

- For each known parsing scheme $S$ (this applies only to deterministic parsing schemes):
  - For each combination of values for the scheme parameters (e.g., the specific digits assigned for nulls or as prefix for nomenclature codes):

![Figure 8. Visualized simplified example of clustering.](image-url)
Parse the ciphertext, then discard the nulls and the nomenclature codes, keeping only regular codes.

Compute the index of coincidence (IC) for the regular codes.

Keep the IC result together with the values of the scheme parameters.

Select the values for the scheme parameters that resulted in the highest IC for the regular codes, \( IC_{\text{max}} \), as well as those parameter values resulting in the second highest IC, \( IC_{\text{max}2} \).

If the relative difference between \( IC_{\text{max}} \) and \( IC_{\text{max}2} \) is at least 5%, there is a reasonable chance that the ciphertexts are texts enciphered with the scheme S and the parameter values corresponding to \( IC_{\text{max}} \).

In this case, an additional method such as the automated homophonic key recovery algorithm (see Section 6.5) can be applied to recover the mapping of the regular codes to their corresponding (regular) elements.

If the difference between \( IC_{\text{max}} \) and \( IC_{\text{max}2} \) is smaller than 5%, it is unlikely that the original key uses the scheme S being tested.

If none of the schemes produces any significant result, it is likely that the original key uses a scheme different from those being tested.

With this process, P117 and S423 were found to use the same scheme as S364.
6.4.1. A DECRYPT parser tester component in CrypTool 2

The DECRYPT parser tester is mainly based on the aforementioned parsing test algorithm, but uses Shannon’s entropy instead of the index of coincidence. The component tests all parsers implemented in CrypTool 2 to tokenize the given ciphertext into cipher elements. It tests all possible values for nulls as well as for the prefix for nomenclature codes (if relevant). It computes a significance value based on the entropy, which shows whether the tested parser may be the correct one for the ciphertext. Figure 10 shows a screenshot of the DECRYPT parser tester analyzing ciphertext transcripts from the B6956 collection. The first column shows the name of the tested parsers. The second column indicates whether the result of the test with a parser is significant. The third column shows the computed entropy of the tokens generated by the parser. The fourth column shows the assumed codes for nulls during the tests. The fifth column shows the nomenclature code prefixes, e.g., for those schemes that specify a prefix for nomenclature codes. In the shown case, the parser “Nomenclature3Digits EndingWithNullDigits” performed “best” during the tests (indicated by the “True” in the Significant column), when the nulls were assumed to be the digits 8 and 1, as expected for B6956.

The DECRYPT parser tester can be used to get a first idea whether a given ciphertext can be parsed using one of the parsers implemented in CT2. If no parser seems suitable, it is likely that the cipher scheme is a new and unknown one, and a new parser needs to be implemented in CT2 to support that new scheme, after the scheme has been determined via other cryptanalytic tools.

6.5. Automated homophonic key recovery

This algorithm is intended to recover the codes (or homophones) assigned to regular elements in homophonic ciphers. A prerequisite is that the sequences of digits of the ciphertexts have been correctly parsed and decomposed into their original regular plaintext elements. Nulls and nomenclature elements are discarded and ignored.

We assume that the set of (plaintext) regular elements (possible letters, common syllables, and common prepositions) is known, or at least most of them. The set of plaintext regular elements in B6956 is relatively large and it is assumed to be a likely superset of most plaintext regular elements used in other ciphers.

Next, a language model for sequences of regular elements is computed, based on $n$-grams of consecutive regular elements, from previously deciphered material. This model reflects not only the language of the original plaintext (Old Italian) but also the specific spelling and enciphering
conventions applied. Those include the removal of double letters, the letter v being interchangeable with the letter u, the letter h most often omitted, the u in qu being omitted, and the encoding of common syllables and preposition using a single regular element (instead of spelling out the letters that compose the syllable or the preposition). The model also reflects the habits of the secretaries enciphering the messages in the period those messages were enciphered. It is important to note that this model is different from conventional n-gram language models, which are based either on sequences of consecutive letters or on sequences of consecutive words.

For this project, as a reference corpus, we employ the set of decoded regular elements obtained after deciphering collections for which the key is known, e.g., B6956, and compute the frequencies $Ref_{i,j,k}$ of all possible trigrams of consecutive regular elements $i$, $j$, and $k$.

We apply a simulated annealing algorithm to recover the mapping of regular codes to regular elements. This algorithm uses a fitness score $Score(K)$ that is computed for a candidate key $K$, as follows:

- Parse the ciphertexts to obtain sequences of regular codes, discarding nulls and nomenclature codes.
- Decode the sequences of regular codes using the candidate key $K$. 
• Compute the frequencies of the trigrams of regular elements $F_{i,j,k}$ in the resulting sequence of regular elements.
• Compute the score for $K$, that is, $Score(K) = \sum_{i,j,k} (F_{i,j,k} \cdot \log(Ref_{i,j,k}))$

We use that score in a simulated annealing search, in which the following transformations are tested at each iteration:

• Swap the assignments of any two homophones. For example, if 00 maps to ‘con’ and 23 maps to e, then after swapping, 23 will map to ‘con’ and 00 will map to e.\(^{40}\)
• Change the assignment of a single homophone, e.g., 01 → ‘che’ instead of 01 → ‘a’.\(^{41}\)
• Given long enough sequences (in terms of cumulative length) of parsed regular codes, this algorithm is likely to correctly recover most of the key mappings between the homophones and the regular elements. However, homophones for rarely used regular elements might often be wrongly mapped.

To improve a candidate key, an additional process can be applied. First, a corpus of Old Italian is parsed, and all distinct words entered into a dictionary, after removing doubled letters, omitting h, and replacing v with u. In addition, all subsequences of 6 letters inside words longer than 6 letters are also added into the dictionary. For example, the word *heretici* is first converted to *eretici*, and added to the dictionary, and the subsequences *eretic* and *retici* are also added.

When the dictionary is ready, the following process is then applied separately for each regular code, e.g., the pair of digits 37, and for each possible assignment of 37 to a regular element (e.g., a letter, syllable, preposition):

• All the ciphertexts are decoded.
• Then, for each occurrence of 37, we check whether the decoding of 37, together with the decoding of regular codes before or after it, generates a plaintext sequence that appears in the dictionary.
• We count the number of such occurrences for each possible assignment.

\(^{40}\)Note that this operation does not change the number of homophones mapped to e or ‘con’.

\(^{41}\)Note that this transformation increases the number of homophones assigned to ‘che’, and decreases the number of homophones assigned to a. To ensure that the key is well balanced in terms of distribution of homophone assignments, a certain maximum number of homophones per regular element is specified when running the algorithm.
If one of the counts is significantly higher than all the others, there is a high probability that the assignment is correct. This process is repeated until there are no more changes in the key.

There might be cases where there are still several possible interpretations, and resolving such ambiguities may require the help of a linguist, as well as a careful review of the transcriptions.

6.6. Digit Clustering

Digit Clustering is different from key clustering. It is applied to a single ciphertext or a set of ciphertexts assumed to have been enciphered using the same key, as opposed to key clustering, which is applied at once to all the ciphertexts in all collections.

The purpose of Digit Clustering is to find a subset of the digits 0 to 9 that behave similarly. For example, in the case of a cipher in which two digits are reserved as nulls (e.g., 1 and 8), the digit cluster consisting of those two values is expected to behave similarly (and display similar statistics) in similar contexts. Another possible cluster, showing similar statistical attributes, should also be expected if only some of the digits are allowed as the second digit of 2-digit regular codes.

Digit Clustering works as follows:

- For each possible partitioning of the 10 digits into 2 groups (or digit clusters) $i$ and $j$, compute the partition score, as follows:
  - For each digit in any cluster:
    - Count the number of times the digit right before it belongs to the same cluster or to the other cluster. Compute the corresponding relative frequencies, denoted as $L_{1i}$ and $L_{1j}$.
    - Count the number of times the digit right after it belongs to the same cluster or to the other cluster. Compute the corresponding relative frequencies, denoted as $R_{1i}$ and $R_{1j}$.
    - Count the number of times the second digit on its left belongs to the same cluster or to the other cluster. Compute the corresponding relative frequencies, denoted as $L_{2i}$ and $L_{2j}$.
    - Count the number of times the second digit on its right belongs to the same cluster or to the other cluster. Compute the corresponding relative frequencies, denoted as $R_{2i}$ and $R_{2j}$.
  - The partition score is the sum of the squares of all the frequencies, that is,
    - $(L_{1i})^2 + (L_{1j})^2 + (R_{1i})^2 + (R_{1j})^2 + (L_{2i})^2 + (L_{2j})^2 + (R_{2i})^2 + (R_{2j})^2$
  - The higher the score, the more homogeneous the clusters are.
Select the partition with the highest score. This top partition score has to be statistically significant. To validate significance:

- The relative difference between the scores of the two partitions with the highest score has to be at least 5%.

A similar process can be performed for possible partitions to more than 2 clusters. The number of digits in each partition can also be set as a parameter (e.g., a partition with 2 clusters of 5 digits, or a partition with 2 clusters of 2 and 8 digits, or a partition with 3 clusters of 2, 4, and 4 digits, respectively). Also, the algorithm can be extended to include special symbols (such as dots), as additional standalone (fixed) clusters.

This algorithm, when applied to F283, with 2 clusters of 5 digits, produced the top partitions and the corresponding scores as shown in Table 8.

It is clear that, with F283, the even digits (0, 2, 4, 6, and 8) play a similar role and that the odd digits (1, 3, 5, 7, and 9) also play a similar role, but different from the role of the even digits. It was later found that the second digit of a 2-digit regular code in F283 can only be an even digit, which explains the Digit Clustering results.

### 6.7. Interval Analysis

This tool computes the distribution of the distance between two occurrences of a given digit (e.g., 0 or 1) or, more accurately, the number of other digits between one occurrence of that certain digit and the next one. For example, in the sequence...2378453903356..., there are 4 digits between the first and second occurrences of the digit 3, and two digits between the second and third occurrences of the digit 3. There are no other digits between the third and fourth occurrences of the digit 3.
In fixed-length homophonic ciphers, this analysis often helps in identifying the digits likely to be nulls. If the regular codes have 2 digits, the intervals between nulls are expected to be mostly even, as nulls enclose sequences of 2-digit codes.

Figure 11 shows a histogram for the output of the tool, for F18, of the interval lengths for the digit 5. Similarly, Figure 12 shows a histogram of the intervals for the digit 8. The $y$-axis is proportional to the number of

![Figure 11. Distribution of intervals between occurrences of digit 5.](image1)

![Figure 12. Distribution of intervals between occurrences of digit 8.](image2)
occurrences at a certain interval (the x-axis), starting with an interval of 0 length (two adjacent 5’s or 8’s, respectively).

It can be seen that, for the digit 8, there is a clear propensity of intervals with an even length (86%), compared to the digit 5, where only 57% of the intervals are even. The hypothesis for F18, later validated, was that 8 is a null and that the cipher elements between nulls are mostly 2-digit regular codes, consisting of pairs of digits other than 8.

6.8. Homophone Detection

The Homophone Detection tool is based on the assumption that if two different regular codes (e.g., 26 and 62 in the case of 2-digit regular codes) appear in the same context, they are likely to be homophones. To illustrate the concept, we search F22 ciphertexts for occurrences of any two regular codes with one, two, or three digits that appear right after the same \( n \) preceding digits, and before the same \( n \) following digits. For example, the search produces the following pair of sequences (with \( n = 8 \)):

- 00202151250212429
- 0020215121250212429

It is likely that 2 and 212 are homophones. The probability that 2 and 212 are homophones is higher if several pairs with similar preceding and following sequences can be found. In F22, 30 such pairs exist. When looking for all pairs \( (n = 8) \), the following list is produced (‘count’ is the number of occurrences, H1 is the first likely homophone, H2 is the second one):

<table>
<thead>
<tr>
<th>Count</th>
<th>H1</th>
<th>H2</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>2</td>
<td>212</td>
</tr>
<tr>
<td>28</td>
<td>171</td>
<td>7</td>
</tr>
<tr>
<td>13</td>
<td>7</td>
<td>721</td>
</tr>
<tr>
<td>12</td>
<td>191</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>121</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>82</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>84</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>521</td>
</tr>
<tr>
<td>6</td>
<td>122</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>26</td>
<td>62</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td>123</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>124</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>125</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>19</td>
<td>76</td>
</tr>
</tbody>
</table>
After having recovered the key for F22, we were able to evaluate the accuracy (and helpfulness) of the output of the tool.

The following entries correctly indicate homophones:

<table>
<thead>
<tr>
<th>[Count]</th>
<th>H1</th>
<th>H2</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>2</td>
<td>212</td>
</tr>
<tr>
<td>28</td>
<td>171</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>191</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>26</td>
<td>62</td>
</tr>
<tr>
<td>5</td>
<td>19</td>
<td>76</td>
</tr>
</tbody>
</table>

The following entries can be explained by the presence of the null digit 8, rather than the presence of homophones:

<table>
<thead>
<tr>
<th>[Count]</th>
<th>H1</th>
<th>H2</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>2</td>
<td>82</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>84</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>28</td>
</tr>
</tbody>
</table>

For example, if 8 is a null, then \( 2 = 28 = 82 \).

The following entries are more tricky:

<table>
<thead>
<tr>
<th>[Count]</th>
<th>H1</th>
<th>H2</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>7</td>
<td>721</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>121</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>521</td>
</tr>
<tr>
<td>6</td>
<td>122</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>123</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>124</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>125</td>
<td>5</td>
</tr>
</tbody>
</table>

It may look as if 21 and 12 are also nulls, or that 7 and 721 are homophones representing the same letter. But this is not the case, as in F22, \( 21 = a \), and \( 12 = z \).

Further investigation shows that those equivalences are the result of 21, 212, and 2 being homophones for the same letter \( a \). As a result, \( 72 = 7212 \), which explains \( 7 = 721 \) in the list (if we remove the trailing 2). Similarly, \( 2123 = 23 \), which explains \( 123 = 3 \) in the list.

Note that the tool may produce also only unreliable results, or no results at all, if the overall length of the ciphertexts is limited. Usually, more than 10,000 symbols are required for this tool to be effective and useful.

### 6.9. Nomenclature code analysis

In many of the ciphers mentioned in this article, nomenclature codes are marked with a dot on top of one of its digits (or on top of the digit right before the digits of the nomenclature codes). In other cases, one of the digits of the nomenclature codes is a special symbol. The purpose of the tool...
described in this section is to identify the likely format of the nomenclature codes.

First, the special signs or the presence of dots can be identified visually, by examining the ciphertext images or transcripts. For illustration, we assume here that, for a given cipher, e.g., F22, those are marked with a dot on top of a digit, but the number of digits in each nomenclature code is unknown. The purpose is to find the length of the nomenclature code (how many digits) and which of the digits is marked. For that purpose, we test several possible lengths (e.g., 2, 3, or 4) and several positions for the digit marked with a dot on top (e.g., in the case of nomenclature codes with 3 digits, either the first digit, the middle digit, the last digit, or the digit right before the nomenclature code). For the hypothesis of nomenclature codes with 3 digits and a dot on the first, we obtain the following top 10 most frequent nomenclature codes:

<table>
<thead>
<tr>
<th>Count</th>
<th>Nomenclature Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>149</td>
<td>6^ .55</td>
</tr>
<tr>
<td>89</td>
<td>6^ .62</td>
</tr>
<tr>
<td>79</td>
<td>6^ .61</td>
</tr>
<tr>
<td>76</td>
<td>1^ .32</td>
</tr>
<tr>
<td>70</td>
<td>6^ .52</td>
</tr>
<tr>
<td>62</td>
<td>1^ .31</td>
</tr>
<tr>
<td>51</td>
<td>6^ .67</td>
</tr>
<tr>
<td>50</td>
<td>1^ .35</td>
</tr>
<tr>
<td>39</td>
<td>6^ .64</td>
</tr>
<tr>
<td>35</td>
<td>1^ .12</td>
</tr>
</tbody>
</table>

This might indicate that nomenclature codes start (mainly) with either 1 or 6. However, if we now assume that the second digit of a nomenclature code has a dot on top and run the tool again, we obtain the following top nomenclature codes:

<table>
<thead>
<tr>
<th>Count</th>
<th>Nomenclature Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>328</td>
<td>16^ .6</td>
</tr>
<tr>
<td>282</td>
<td>16^ .5</td>
</tr>
<tr>
<td>243</td>
<td>11^ .3</td>
</tr>
<tr>
<td>188</td>
<td>11^ .1</td>
</tr>
<tr>
<td>94</td>
<td>16^ .7</td>
</tr>
<tr>
<td>80</td>
<td>12^ .9</td>
</tr>
<tr>
<td>45</td>
<td>16^ .1</td>
</tr>
<tr>
<td>41</td>
<td>13^ .1</td>
</tr>
<tr>
<td>39</td>
<td>22^ .6</td>
</tr>
<tr>
<td>34</td>
<td>21^ .3</td>
</tr>
</tbody>
</table>

This is even more convincing, as 4 nomenclature codes occur more than 100 times vs. only 1 in the previous case.

Finally, if we assume the last digit has a dot, we obtain

<table>
<thead>
<tr>
<th>Count</th>
<th>Nomenclature Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>175</td>
<td>211^ .</td>
</tr>
<tr>
<td>171</td>
<td>216^ .</td>
</tr>
</tbody>
</table>
While the results are better than when assuming that the first digit has a
dot, they are worse than when assuming that the second digit has a dot. Therefore, nomenclature codes in this cipher have 3 digits, the middle one
having a dot on top.

It is also possible to validate (or estimate) the likely length of the nomenclature codes by looking at adjacent nomenclature codes. In the case of F22, it was found that there are 10s of cases of digits with a dot on top, followed by two other digits, and another digit with a dot on top. For example, in the (frequent) sequence 11.316.5, the second 1 has a dot on top, followed by 31, and by 6 with a dot on top. This validates the assumption that the nomenclature codes of F22 have 3 digits.42

Those assumptions about the nomenclature codes of F22 (in addition to some homophone candidate pairs) were useful in order to search for a matching key in the sources. Such a key was eventually found in Meister (1906, 393/40), and the assumptions were found to be correct after conducting a sample decipherment.

6.10. Probabilistic Decoding of Variable-Length Homophones

Some variable-length homophonic ciphers can be deciphered (parsed and decoded) in a deterministic manner, provided that there is a way to differentiate between 1-digit, 2-digit, and 3-digit regular codes, as in F6 and F17.43 However, in most cases, parsing for such ciphers is not deterministic. At every step, there might be more than one possibility to decompose a subsequence of digits in the ciphertext (enclosed by nulls or by nomenclature codes) into valid regular codes. And in the case that the subsequence is long, the number of parsing (and decoding) options rises exponentially and may be very high.

The algorithm described in this section is intended to assist a historian or linguist familiar with Old Italian in deciphering such a ciphertext for which the key is known, in a semi-automatic way, by suggesting several options for parsing and decoding, at each step.

42It was later found that 11.3 = Re di Francia (King of France), and 16.5 = quelli.
43In F17, dots are employed to mark regular codes, but not always consistently.
In some cipher collections, nulls are consistently used at the beginning and at the end of each word (alternatively, a word may be enclosed inside two nomenclature elements). In such a case, possible parsing and decoding options can be automatically generated and checked against a dictionary. Only those decipherments that result in a word that can be found in this dictionary of Old Italian words are displayed.

The disadvantages of this method are:

- In the case of historical enciphering errors or of (modern) transcription errors, the word might not be correctly reproduced and found in the dictionary.
- A preposition may precede a word that follows right after without a null in-between. The combination of the preposition and the word following it might not appear in the dictionary.

This method also does not work if there are no nulls between two words or if nulls were not added consistently, which is the case for several ciphers. For those cases, an extended probabilistic decoding tool was developed as follows:

- Building a database of \( n \)-grams of letters in a corpus of Italian text (with the spelling adapted, e.g., removing doubled letters). For the project, 5-grams of letters were used.
- Computing the relative frequencies of the \( n \)-grams in the corpus and scaling those frequencies to a logarithmic scale.
- When deciphering a ciphertext, evaluating possible parsing and decoding options for each subsequence, against the reference \( n \)-grams statistics.
- Showing a sorted list of the most likely decipherments. Also, if one of the options appears in the dictionary, it is always shown first, regardless of its score.

For example, for a sequence taken from a ciphertext from F22, the tool produces the following options:

\[
700035453531171
\]

\[
\text{presenti* presentir* presentir presentir inresentir inresenti preseuri presenti preseenti preseuentir preseuir preseuenti inresentir inresenti preseuir inresentir pinesentir pinesentir pinesentir pinesentir pinesentir pinesentir pinesentir pinesentir pinesentir}
\]

The words marked with * indicate that they appear in the dictionary.

We illustrate here two of the possible parsing and decoding options and how the composing digits are interpreted:
The word ‘presentir’ is more likely than the second choice. But both are valid.

This enhanced method better handles transcriptions with errors or sequences of words not separated by nulls rather than only relying on dictionary words. Still, manual intervention is needed to select the most plausible option. The process might be slow and tedious.

6.11. Probabilistic Decoding of Polyphones

Probabilistic Decoding of Polyphones is similar in concept to the Probabilistic Decoding of Variable-Length Homophones except that, with polyphonic ciphers, parsing is deterministic but decoding is not (whereas in variable-length homophonic ciphers, parsing is not deterministic but decoding is deterministic). Note that, with some polyphonic ciphers, decoding may also be deterministic, for example, if the correct choice for the polyphone is indicated by the presence or absence of a dot on top (or bottom) of the digit, as in F4.

For other polyphonic ciphers such as S6, decoding is not deterministic and it involves multiple choices for most of the digits of the ciphertext and, therefore, the need for the tool described here.

The tool uses a dictionary and the frequencies of 5-grams of letters. It produces and ranks all possible decoding options for a subsequence of digits between nulls (or nomenclature codes). For each digit, there are 2 decoding options, so the number of options raises exponentially w.r.t. the length of the subsequence. Therefore, the tool is effective only for sequences of 20 or fewer digits.

For example, for a subsequence in a ciphertext of S6, the top suggestions are:

```
6686521415

necesitato* necesitats necesiuauo necesiuuusu necesiuuato necesiuuats
necesitauso necesitausu necesitlauo necesitlausu necesitolto necesitolts
encesitato encesitats
```

Only the first option is valid.

6.12. Automated Polyphonic Key Recovery

This tool is needed when a cipher has been identified to be polyphonic, but the key is unknown. The polyphones (the original regular elements,
usually, single letters) assigned to each regular code (a single digit in most cases) can be automatically recovered using the algorithm described here.

The algorithm requires a database of \( n \)-grams of letters from a corpus, in this case, 5-grams. The frequencies of the 5-grams are converted to a logarithmic scale. Also, the structure of the nomenclature codes must be known so that the nomenclature codes can be removed from the ciphertext sequences.

The algorithm employs **simulated annealing**, as follows:

- The **candidate key** consists of the selection of the digit assigned as the null, and for the remaining 9 digits, of the assignment of a pair of letters. The 9 pairs of letters are selected from a set of 18 letters, which only contains the most frequent letters (but not h, which is also omitted).
- To compute the **score** for a candidate key,
  - Set the key score to an initial value of 0.
  - For each subsequence of 5 consecutive digits in the ciphertext (excluding nomenclature codes, and null digits):
    - Generate all options (there are \( 2^5 = 32 \) options) for decoding the 5 digits, and select the option, which is a 5-gram of letters, with the highest frequency in the corpus. Convert this frequency to a logarithmic scale and add this value to the key score.
  - Repeat for all following subsequences, possibly overlapping with the last 4 digits of the previous subsequence.
- The **transformations** to be applied on a candidate key at each step of simulated annealing, consist of
  - Swapping the assignments of any 2 of the 18 letters (provided they are not assigned to the same digit). For example, if \( 2 = \{a, l\} \) (that is, 2 represents either the letter a or the letter l), and \( 7 = \{q, z\} \), then, after swapping letters a and z, we obtain \( 2 = \{z, l\} \) and \( 7 = \{q, a\} \).
  - Swapping the polyphones of a digit with the polyphones of another digit. For example, if \( 2 = \{a, l\} \), and \( 7 = \{q, z\} \), then, after swapping the polyphones of 2 with the polyphones of 7, we obtain \( 2 = \{q, z\} \) and \( 7 = \{a, l\} \);
  - Changing the digit assigned as null. For example, if 0 is the null digit and \( 7 = \{q, z\} \), then after swapping the assignments of 0 and 7, we obtain \( 0 = \{q, z\} \) and 7 as the null digit.

This algorithm reliably recovers polyphones and nulls, for ciphertexts with as little as 1,000 digits.
### 6.13. A **DECRIPT** symbol heatmap component in CrypTool 2

To visualize the bigram frequencies of digit ciphers we implemented the **DECRIPT** Symbol Heatmap component in CrypTool 2. Also, trigram frequencies can be visualized. Less frequent combinations are shown in blue color (“cold”) and more frequent combinations are shown in red color (“hot”). Combinations that are close to the average frequency are shown in green. **Figure 13** shows a screenshot of the **DECRIPT** Symbol Heatmap component with a heatmap of C13 ciphertext bigrams. The X-axis specifies the first digit of the bigram, the Y-axis specifies the second digit of the bigram. The number within a cell shows the total number of the specific digit bigram. In **Figure 13**, “22” is the most frequent bigram.

![Figure 13: Screenshot of the **DECRIPT** Symbol Heatmap in CrypTool 2 showing a heatmap of C13.](image)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><strong>0</strong></td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td>9</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>17</td>
<td>20</td>
<td>15</td>
<td>14</td>
<td>3</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td>5</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>9</td>
<td><strong>35</strong></td>
<td>25</td>
<td>7</td>
<td>28</td>
<td>18</td>
<td>7</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>27</td>
<td>17</td>
<td>18</td>
<td>12</td>
<td>8</td>
<td>14</td>
<td>9</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>7</td>
<td>18</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>2</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>11</td>
<td>21</td>
<td>11</td>
<td>13</td>
<td><strong>29</strong></td>
<td>14</td>
<td>9</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>13</td>
<td>14</td>
<td>13</td>
<td>9</td>
<td><strong>26</strong></td>
<td>23</td>
<td>10</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>15</td>
<td>5</td>
<td>5</td>
<td>11</td>
<td>9</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>7</td>
<td>8</td>
<td>12</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>15</td>
<td>10</td>
<td>8</td>
<td>5</td>
<td>1</td>
<td>9</td>
<td>9</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>
7. Conclusion

Based on our study and the large set of decipherments and recovered keys from the 16th, 17th, and 18th Centuries originated from the papal correspondence in European countries, important conclusions can be drawn.

In the 16th Century, and in accordance with Meister, there is strong evidence for diversity, innovation, and sophistication in the development and use of (papal) cipher methods, as illustrated in Table 9.

Cipher types for that period include monoalphabetic, homophonic, and polyphonic ciphers. Most of the homophonic ciphers use variable-length homophones. Interestingly, variable-length homophonic ciphers are highly secure even against modern computing and algorithms. The authors could only recover the keys for such ciphers if a matching plaintext was

<table>
<thead>
<tr>
<th>Key Ref.</th>
<th>Period</th>
<th>Type</th>
<th>Deterministic</th>
</tr>
</thead>
<tbody>
<tr>
<td>K_P1</td>
<td>1535-1536</td>
<td>Monoalphabetic</td>
<td>No</td>
</tr>
<tr>
<td>K_S1</td>
<td>1538-1548</td>
<td>Variable length</td>
<td>No?</td>
</tr>
<tr>
<td>K_F3</td>
<td>1552-1554</td>
<td>Variable length</td>
<td>No</td>
</tr>
<tr>
<td>K_S6</td>
<td>1566-1572</td>
<td>Polyphonic</td>
<td>No</td>
</tr>
<tr>
<td>K_F6</td>
<td>1573</td>
<td>Variable length</td>
<td>Yes</td>
</tr>
<tr>
<td>K_F283</td>
<td>1580</td>
<td>Fixed length</td>
<td>Yes</td>
</tr>
<tr>
<td>K_P8</td>
<td>1579-1581</td>
<td>Fixed length</td>
<td>Yes</td>
</tr>
<tr>
<td>K_F17</td>
<td>1583-1585</td>
<td>Variable length</td>
<td>Yes</td>
</tr>
<tr>
<td>K_F18</td>
<td>1585-1586</td>
<td>Fixed length</td>
<td>Yes</td>
</tr>
<tr>
<td>K_F4</td>
<td>1587</td>
<td>Polyphonic</td>
<td>Yes</td>
</tr>
<tr>
<td>K_F22</td>
<td>1585-1589</td>
<td>Variable length</td>
<td>No</td>
</tr>
<tr>
<td>K_F22b</td>
<td>1585-1589</td>
<td>Variable length</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 9. Diversity of papal ciphers in the second half of the 16th Century.
available or if the key could be found in Meister, based on external information or clues obtained with the help of computerized tools. Those ciphers for which parsing is not deterministic are not only difficult to break, but they were also harder to use, historically. This supports the hypothesis of a level of sophistication by the cipher creators and users, driven by the Argentis and their predecessors, giving precedence to security over simplicity.

An additional level of flexibility is demonstrated by the use of polyphonic ciphers. In general, there is a trade off between the simplicity of usage and the security of papal ciphers:

- Polyphonic ciphers are very simple to use but are not secure. Due to their simplicity, polyphonic ciphers were a convenient option for communicating with envoys in countries without strong codebreaking capabilities.
- Variable-length homophonic ciphers are complex and more difficult to use, but much more secure.
- Fixed-length homophonic ciphers are in the middle of the two aforementioned regarding simplicity and security.

The recovered keys and decipherments also highlight additional measures to increase security, such as the removal of redundant letters (e.g., double letters), the judicious use of nulls, as well as the inclusion of a nomenclator.

<table>
<thead>
<tr>
<th>Key Ref.</th>
<th>Period</th>
<th>Type</th>
<th>Deterministic</th>
</tr>
</thead>
<tbody>
<tr>
<td>17th Century</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K_F64</td>
<td>1625</td>
<td>Fixed length</td>
<td>Yes</td>
</tr>
<tr>
<td>K_B6956</td>
<td>1628</td>
<td>Fixed length</td>
<td>Yes</td>
</tr>
<tr>
<td>K_F346</td>
<td>1632-1634</td>
<td>Fixed length</td>
<td>Yes</td>
</tr>
<tr>
<td>K_C13</td>
<td>1656</td>
<td>Fixed length</td>
<td>Yes</td>
</tr>
<tr>
<td>18th Century</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K_5364</td>
<td>1717-1720</td>
<td>Fixed length</td>
<td>Yes</td>
</tr>
<tr>
<td>K_C5</td>
<td>1721</td>
<td>Fixed length?</td>
<td>Yes?</td>
</tr>
<tr>
<td>K_5423</td>
<td>1736</td>
<td>Fixed length</td>
<td>Yes</td>
</tr>
<tr>
<td>K_P117</td>
<td>1757-1761</td>
<td>Fixed length</td>
<td>Yes</td>
</tr>
<tr>
<td>K_5304</td>
<td>1767</td>
<td>Fixed length</td>
<td>Yes</td>
</tr>
</tbody>
</table>
A completely different picture emerges from the recovered keys and decipherments of ciphertexts from the 17th and 18th Centuries, as listed in Table 10.

The ciphers are exclusively fixed-length homophonic, with deterministic parsing and decoding. As a result, their level of security is in general lower than with 16th Century’s ciphers, but they are also more convenient for encryption and decryption. While some of the 16th Century ciphers did not include nomenclatures, all ciphers in this later period have some nomenclature and are of a typically larger size.

To complete the picture, Alvarez’s findings on the 19th Century reflect an ongoing decline, due to a lack of innovation, a reliance on outdated methods, and a cipher organization being marginalized, with the transferring of cryptographic duties to junior clerks instead of professional cryptographers.

The main trends in papal cryptography, from the 14th to the 19th Centuries, are summarized in Table 11.

We conclude here the primary contributions of our study:

- The largest collection of deciphered papal ciphertexts. Some of the material was not available in plaintext form before and can now be read for the first time.
- Insights into cipher methods and practices that complement Meister and other sources on the 16th Century.
- Unique source on papal cryptography for the 17th and 18th Centuries, exposing some clear evidence for a steady decline.
A new set of tools for cryptanalysis, several of which are integrated into CrypTool 2.

Due to the vast historical, linguistic, technical, and cryptanalytic scope of the subject, the results presented in this article only represent work in progress and cannot claim to bring definitive answers yet. In particular, additional research is required to study the historical context for the evolution of papal ciphers during that period, as well as comparing papal cryptography with the cryptography of other European countries over time.

Appendix A

In the following sections, we present examples of papal ciphers from the 16th Century to the 18th Century. In the decryption screenshots, regular codes are shown in blue, nomenclature codes are shown in green, and nulls are shown in gray. The plaintext/ciphertext mappings are generated using the DECRYPT Decipherer component of CrypTool 2 (CT2). We also present the keys.

F283 (Homophonic, Fixed-Length, 16th Century)
Fixed-length homophonic, only the vowels have two homophones. This cipher does not employ nulls. Nomenclature codes end with a special character.

![Figure 14. F283 – Ciphertext/plaintext shown in the CT2 DECRYPT Decipherer.](image-url)
**Table 12.** F283 – Key.

| a | b | c | d | e | f | g | h | i | l | m | n | o | p | r | s | t | u | v | z |
| 50| 42| 14| 16| 18| 20| 12| 24| 26| 28| 30| 22| 34| 36| 40| 32| 44| 46| 48| |
| 92| 52| 62| 72| 82|  |

**S.S.** 124+  .  284+  Regina  744+

**M.S.** 144+   questa  288+   mente  800+

quanto 188+   re  344+   V.S.  822+

non  200+   per  366+   N.S.  824+

S.M.Ta 224+   quello  388+   tanto  966+

che  266+   qua  488^   essere  988^.

---

**C13 (Homophonic, Fixed-Length, 17th Century)**

Fixed-length homophonic. Nomenclature codes have 3 digits and are followed by a null. Nulls have two digits.

---

**Figure 15.** C13 – Ciphertext/plaintext shown in the CT2 DECRYPT Decipherer.
### Table 13. C13 – Key.

<table>
<thead>
<tr>
<th>Null</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
<th>l</th>
<th>m</th>
<th>n</th>
<th>o</th>
<th>p</th>
<th>r</th>
<th>s</th>
<th>t</th>
<th>u/v</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>55</td>
<td>77</td>
<td>88</td>
<td>15</td>
<td>25</td>
<td>35</td>
<td>45</td>
<td>51</td>
<td>17</td>
<td>11</td>
<td>165</td>
<td>329</td>
<td>22</td>
<td>23</td>
<td>33</td>
<td>28</td>
<td>61</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>66</td>
<td>99</td>
<td>21</td>
<td>31</td>
<td>41</td>
<td>51</td>
<td>17</td>
<td>65</td>
<td>32</td>
<td>39</td>
<td>42</td>
<td>34</td>
<td>72</td>
<td>91</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>68</td>
<td>71</td>
<td>43</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
<th>l</th>
<th>m</th>
<th>n</th>
<th>o</th>
<th>p</th>
<th>r</th>
<th>s</th>
<th>t</th>
<th>u/v</th>
</tr>
</thead>
<tbody>
<tr>
<td>59</td>
<td>56</td>
<td>48</td>
<td>64</td>
<td>46</td>
<td>67</td>
<td>47</td>
<td>63</td>
<td>54</td>
<td>58</td>
<td>75</td>
<td>74</td>
<td>62</td>
<td>81</td>
<td>78</td>
<td>79</td>
<td>49</td>
<td>52</td>
</tr>
</tbody>
</table>

- **Pace?** 115 essere 722 qualo 746
- **N.S.** 121 havuto 731 quanto/quando? 749
- **Re** 157 mente 734 quello 751
- **Sig.re** 189 menare? 735 tanto 753
- **trattato** 237 molto 736 avviso 787
- **Mo.re.?** 264 qualih 745 Elettore? 882
- **essere** 722 qualo 746 neghio 986

---

**S304 (Homophonic, Fixed-Length, 18th Century)**

Fixed-length homophonic. Nomenclature codes have 4 digits and start with 8.
Figure 16. S304 – Ciphertext/plaintext shown in the CT2 DECRYPT Decipherer.

Table 14. S304 – Key.

|   | a | b | c | d | e | f | g | i | l | m | n | o | p | r | s | t | u/v | z |
|5  | 17| 14| 09| 11| 13| 20| 29| 42| 44| 10| 12| 16| 19| 66| 76| 79| 37|   |
|   | 71| 41| 90| 49| 92| 99|   |   |   |   |   |   |   |   |   |   |   |   |

<table>
<thead>
<tr>
<th>che</th>
<th>con</th>
<th>da</th>
<th>dal</th>
<th>de</th>
<th>del</th>
<th>gli</th>
<th>il</th>
<th>in</th>
<th>io</th>
<th>ia</th>
<th>ma</th>
<th>mi</th>
<th>ne</th>
<th>nel</th>
<th>no</th>
<th>non</th>
<th>qu</th>
<th>si</th>
<th>st</th>
<th>ta</th>
<th>te</th>
<th>ti</th>
<th>to</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>04</td>
<td>06</td>
<td>07</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>26</td>
<td>27</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>36</td>
<td>43</td>
<td>46</td>
<td>47</td>
<td>60</td>
<td>61</td>
<td>64</td>
<td>67</td>
<td>69</td>
<td>70</td>
<td>72</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ancora 8010  mente  8037  Confesore  8171
aveve  8014  molto  8040  Conte  8190
bene  8017  perche  8046  Ducas  8220
benche  8019  quando  8063  libero  8312
contunocio  8021  ultime  8090  Madrid  8317
fame  8030  Camera  8140  secondo  8493
mente  8037  Confesore  8171  giorni  8706
F17 (Homophonic, Variable-Length, Deterministic, 16th Century)
  Variable-length homophonic cipher. Two-digit codes (usually) start with a digit with a
dot on top.

Figure 17. F17 – Ciphertext/plaintext shown in the CT2 DECRYPT Decipherer.

Table 15. F17 – Key.

|    | a | b | c | d | e | f | g | h | i | l | m | n | o | p | q | r | s | t | u | et |
| 1  | 3 | 5 | 9 | 7 | 03| 25| 35| 45| 07| 23| 33| 43| 05| 29| 39| 49| 59| 69| 01| 37|
| 09 | 15| 13| 19|   |

<table>
<thead>
<tr>
<th></th>
<th>che</th>
<th>266+</th>
<th>heretici</th>
<th>344</th>
<th>Mons.</th>
<th>600</th>
</tr>
</thead>
<tbody>
<tr>
<td>cho</td>
<td>344</td>
<td>Mons.</td>
<td>600</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
F22b (Homophonic, Variable-Length, Nondeterministic, 16th Century)
Variable-length homophonic cipher. There are no nulls.

Figure 18. F22b – Ciphertext/plaintext shown in the CT2 DECRYPT Decipherer.
Table 16. F22b – Key.

| a | d | e | g | h | i | l | m | n | o | p | r | s | t | u |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 91| 5 | 12| 9 | 6 | 14| 10| 23| 15| 80| 34| 19| 30| 21|   |
| 7 | 8 | 17|    |   |   | 18|    |   |   | 54|   |   |   |   |   |

<table>
<thead>
<tr>
<th>b</th>
<th>a</th>
<th>b</th>
<th>i</th>
<th>c</th>
<th>o</th>
<th>c</th>
<th>d</th>
<th>a</th>
<th>d</th>
<th>e</th>
<th>i</th>
<th>d</th>
<th>o</th>
<th>f</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>86</td>
<td>55</td>
<td>64</td>
<td>73</td>
<td>51</td>
<td>40</td>
<td>16</td>
<td>20</td>
<td>76</td>
<td>99</td>
<td>34</td>
<td>44</td>
<td>79</td>
<td>97</td>
<td>67</td>
<td>37</td>
<td>22</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>m</th>
<th>o</th>
<th>n</th>
<th>a</th>
<th>ne</th>
<th>ni</th>
<th>no</th>
<th>pa</th>
<th>pe</th>
<th>pi</th>
<th>qua</th>
<th>que</th>
<th>ra</th>
<th>re</th>
<th>ri</th>
<th>ro</th>
<th>sa</th>
</tr>
</thead>
<tbody>
<tr>
<td>92</td>
<td>74</td>
<td>70</td>
<td>31</td>
<td>36</td>
<td>49</td>
<td>52</td>
<td>33</td>
<td>32</td>
<td>35</td>
<td>41</td>
<td>43</td>
<td>48</td>
<td>63</td>
<td>78</td>
<td>83</td>
<td>96</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>mpo</th>
<th>102</th>
<th>ma</th>
<th>130</th>
<th>me</th>
<th>216</th>
</tr>
</thead>
<tbody>
<tr>
<td>mi</td>
<td>106</td>
<td>al</td>
<td>1(^.^{37})</td>
<td>re</td>
<td>2(^.^{16})</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>me</th>
<th>116</th>
</tr>
</thead>
<tbody>
<tr>
<td>re</td>
<td>1(^.^{16})</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>umena</th>
<th>1(^.^{24})</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ma</th>
<th>130</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>me</td>
</tr>
</tbody>
</table>
F4 (Polyphonic, Deterministic, 16th Century)
Polyphonic cipher. Digit 3 is a null. Nomenclature codes have two or three digits, preceded by a digit with a dot on top.

S6 (Polyphonic, Nondeterministic, 16th Century)
Polyphonic cipher. Most nomenclature codes have 3 digits, the middle one being 0 (0 also serves as null).
Acknowledgments

We are grateful to Eszter Száraz and the librarians for assisting in the secret archives to collect the ciphers, to all students who helped out in transcribing the manuscripts, and to Paolo Bonavoglia for his assistance with Italian texts.

About the authors

George Lasry is a Computer Scientist in the High-Tech Industry in Israel, and a member of the DECRYPT and the CrypTool projects. He obtained his PhD in 2017 with the research group “Applied Information Security” (AIS) at the University of Kassel. His primary interest in cryptographic research is the computerized cryptanalysis of classical ciphers and cipher machines.

Beáta Megyesi is an Associate Professor in Computational Linguistics working at the Department of Linguistics and Philology, Uppsala University, Sweden. She is specialized in digital philology and natural language processing of non-standard language data. She serves as the PI of the DECODE and DECRYPT projects, aiming at the development of infrastructural resources and tools for historical cryptology.

Nils Kopal is a Computer Scientist and Cryptanalyst working as a postdoc at the University of Siegen, Germany. He specializes in cryptanalysis of classical ciphers and

Table 18. S6 – Key.

<table>
<thead>
<tr>
<th></th>
<th>0+</th>
<th>non/no</th>
<th>206</th>
<th>qui</th>
<th>509</th>
</tr>
</thead>
<tbody>
<tr>
<td>qua</td>
<td>30+9</td>
<td>che</td>
<td>209</td>
<td>quel</td>
<td>609</td>
</tr>
<tr>
<td>N.S.</td>
<td>101</td>
<td>Re</td>
<td>301</td>
<td>quello/a</td>
<td>709</td>
</tr>
<tr>
<td>H.S.re.</td>
<td>102</td>
<td>Frandra?</td>
<td>304</td>
<td>Re di Francia</td>
<td>801</td>
</tr>
<tr>
<td>Regina di Francia?</td>
<td>104</td>
<td>H.S.re.</td>
<td>401</td>
<td>Sauer?</td>
<td>809</td>
</tr>
<tr>
<td>Francia</td>
<td>105</td>
<td>S.M.</td>
<td>403</td>
<td>H.S.re.</td>
<td>901</td>
</tr>
<tr>
<td>V.S.</td>
<td>109</td>
<td>questo</td>
<td>406</td>
<td>quelle</td>
<td>902</td>
</tr>
<tr>
<td>con?/per?</td>
<td>202</td>
<td>Germania</td>
<td>409</td>
<td>Germania</td>
<td>903</td>
</tr>
<tr>
<td>Con.o.</td>
<td>204</td>
<td>per</td>
<td>502</td>
<td>questa</td>
<td>906</td>
</tr>
<tr>
<td>non/no</td>
<td>206</td>
<td>qui</td>
<td>509</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 20. S6 – Ciphertext/plaintext shown in the CT2 DECRYPT Decipherer.
distributed cryptanalysis. He is leading the development of the open-source software CrypTool 2. In the DECRYPT project he is responsible for developing tools for cryptanalysis of historical and classical ciphers and integrating these in the DECRYPT pipeline and CT2.

**Funding**

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

**References**


Archivio Segreto Vaticano. 2016a. Segr.di.Stato/Francia/17/030r@2016 Archivio Apostolico Vaticano. The picture has been reproduced by the permission of the AAV, all rights reserved.

Archivio Segreto Vaticano. 2016b. Segr.di.Stato/Francia/18/007r@2016 Archivio Apostolico Vaticano. The picture has been reproduced by the permission of the AAV, all rights reserved.

Archivio Segreto Vaticano. 2016c. Segr.di.Stato/Francia/283C/050v@2016 Archivio Apostolico Vaticano. The picture has been reproduced by the permission of the AAV, all rights reserved.

Archivio Segreto Vaticano. 2016d. Segr.di.Stato/Francia/283C/052r@2016 Archivio Apostolico Vaticano. The picture has been reproduced by the permission of the AAV, all rights reserved.


