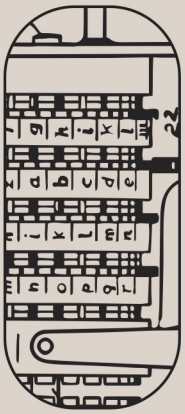
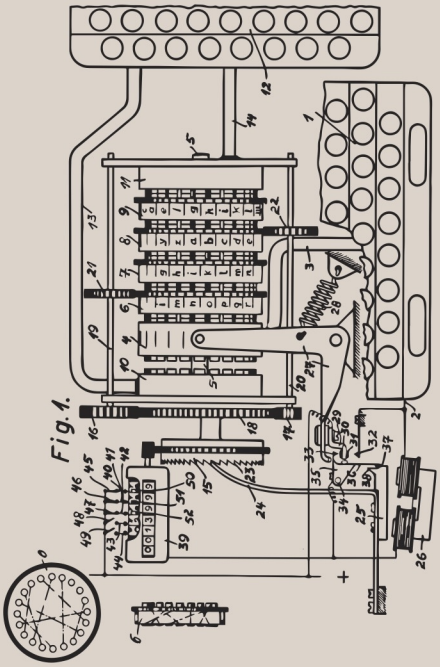
HISTO CRYPT

POZNAŃ 2025

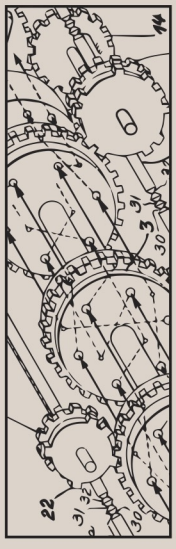
16-18 JUNE



Q W E R
T Z U I O
P A S D



8TH INTERNATIONAL CONFERENCE ON HISTORICAL CRYPTOLOGY



Proceedings of the 8th International Conference on Historical Cryptology

HistoCrypt 2025

Editors
Eugen Antal and Pavol Marák

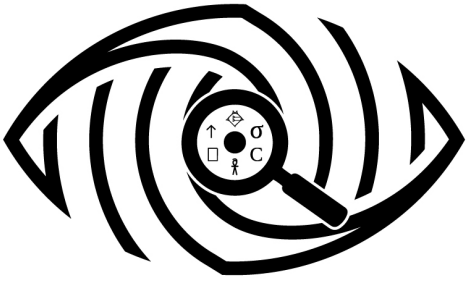
June 16-18, 2025
Poznań, Poland

Published by

NEALT Proceedings Series 58
D-Space at Tartu University Library
ISSN: 1736-8197
eISSN: 1736- 6305
ISBN: 978-99-0853-287-5
https://doi.org/10.58009/aere-perennius0158

SPONSORS

SAMORZĄD
WOJEWÓDZTWA
WIELKOPOLSKIEGO



DESCRYPT

RIKSBANKENS
JUBILEUMSFOND

FRÄMJAR HUMANIORA
OCH SAMHÄLLSVETENSKAP

Preface

The program committee is delighted to present the proceedings of the 8th International Conference on Historical Cryptology (HISTOCRYPT 2025). The conference will be held on June 16-18, 2025 in Poznań, Poland.

Following the tradition of previous HISTOCRYPT conferences, HISTOCRYPT addresses all aspects of historical cryptography and cryptanalysis. It is inherently cross-disciplinary, incorporating work from various fields such as mathematics, history, computer science, AI, computational linguistics, linguistics, and image processing. The conference's topics include, but are not limited to, the use of cryptography in military, diplomacy, business, and other areas, the use of automatic methods including AI in the field of historical cryptology, unsolved historical cryptograms, cipher machines, the history of cryptography, the roots of modern ciphers in historical cryptology, linguistic aspects of cryptology, the influence of cryptography on the course of history, cryptology and its connections to neighbouring fields of study, and teaching and promoting cryptology in schools, universities, and the public.

The scientific program was compiled by an international scientific program committee, consisting of researchers in computer science, cryptology, history, intelligence, language technology, and linguistics. The program committee welcomed submissions in two tracks: regular papers up to 10 pages (including appendices, excluding references) on substantial, original, and unpublished research, including evaluation results, where appropriate; and short papers up to 4 pages (including appendices, excluding references) on smaller, focused contributions, work in progress, negative results, surveys, tutorials, or opinion pieces.

The conference received 24 submissions from all over the world, including from Albania, Australia, Austria, Belgium, Canada, Czechia, France, Germany, Ghana, Hungary, Indonesia, Israel, Italy, Japan, Netherlands, Norway, Slovakia, Spain, Sweden, Switzerland, United Kingdom, United States.

The program committee aimed to compile a high-quality program with a wide variety of topics by conducting a double-blind review process. Each submission was evaluated by at least three expert reviewers in the corresponding field. The reviews were synchronized and, if necessary, thoroughly discussed among the reviewers and area chairs of the program committee. The final selection was based on their recommendations and discussions. In the end, we accepted 15 regular papers and 4 short papers for publication, resulting in a total of 19 papers (79% acceptance rate). One accepted paper was later withdrawn by the author. The remaining 18 accepted submissions are included in this volume, organized alphabetically by the surname of the first author.

For the conference in Poznań, we have invited three keynote speakers who have graciously accepted our invitation. They are:

- *François Desset*,
  European archaeologist and philologist specializing in Ancient Near Eastern history, Université de Liège/Tehran University/ArchéOrient, he led the decipherment of Linear Elamite writing, an ancient script used in southern Iran between the late 3rd and early 2nd millennium BCE;

- *Marek Grajek*,
  freelance researcher, he is a retired consultant in cryptology applications and a historian of the discipline, he is the author of more than a dozen books in this field;

- *Kevin Knight*,
  chief scientist, Threeven Labs, he has co-authored over 150 research papers on natural language processing, as well as the widely adopted textbook "Artificial Intelligence".

Organizing a conference and a peer-review process always relies on the goodwill and support of many colleagues to take their valuable time and contribute to an interesting and fruitful program. First of all, I would like to thank the local organizing committee: *Piotr Bojarski* and *Marcin Słomiński* for the great collaboration and the great conference location, and for managing the ticketing system. My special thanks goes to all area chairs of the program committee: *Carola Dahlke*, *Benedek Láng*, *Beáta Megyesi*, and *Michelle Waldispühl* for their substantial support and for many constructive online-meetings. I also want to thank the 23 reviewers for their time, effort, and constructive feedback during the review process. In addition, I would like to thank all the authors who have made these proceedings possible. Lastly, I would like to express my gratitude to Arno Wacker and Christoph Ruhl for managing the conference website.

Bratislava, May 8, 2025

Eugen Antal
Program Chair of HISTOCRYPT 2025

Program Committee

- Eugen Antal (program chair), Slovak University of Technology in Bratislava, Slovakia
- Carola Dahlke (area chair), Deutsches Museum, Germany
- Benedek Láng (area chair), Budapest University of Technology and Economics, Hungary
- Beáta Megyesi (area chair), Stockholm University, Sweden
- Michelle Waldispühl (area chair), University of Oslo, Norway
- Richard Bean, University of Queensland, Australia
- Paolo Bonavoglia, Mathesis Venezia c/o Convitto “Marco Foscarini”, Italy
- Camille Desenclos, Université de Picardie Jule Verne, France
- Jörgen Dinnissen, Independent researcher, The Netherlands
- John Dooley, Knox College, United States
- Magnus Ekhall, Private Researcher, Sweden
- Joachim von zur Gathen, Bonn-Aachen International Center for Information Technology, Germany
- Otokar Grošek, Slovak University of Technology in Bratislava, Slovakia
- Kevin Knight, Threeven Labs, United States
- Jozef Kollár, Slovak University of Technology in Bratislava, Slovakia
- Nils Kopal, University of Siegen, Germany
- George Lasry, DECRYPT and CrypTool projects, Germany
- Jakub Mírka, The State Regional Archives in Pilsen, Czechia
- Diego Navarro, University Carlos III of Madrid, Spain
- Ingo Niebel, Private Researcher, Germany
- Paul Reuvers, Crypto Museum, Netherlands
- Anne-Simone Rous, State Palaces, Castles and Gardens of Saxony, Germany
- Klaus Schmeh, Codebreaking-Guide.com, Germany
- Gerhard F. Strasser, The Pennsylvania State University, United States
- Satoshi Tomokiyo, Cryptiana, Japan

- Serge Vaudenay, EPFL, Switzerland
- Frode Weierud, Crypto Cellar Research, Norway
- Pavol Zajac, Slovak University of Technology in Bratislava, Slovakia

Local Organizing Committee

- Piotr Bojarski, Enigma Cipher Centre, Poland
- Marcin Słomiński, Enigma Cipher Centre, Poland

Steering Committee

- Carola Dahlke (chair), Deutsches Museum, Germany
- Benedek Láng (vice chair), Budapest University of Technology and Economics, Hungary
- Richard Bean (secretary), University of Queensland, Australia
- Dermot Turing (member), Kellogg College Oxford, UK
- Camille Desenclos (member), Université de Picardie Jules Verne, France

Contents

Enhancing Classical Cipher Type Detection: Prompt Engineering with Common LLMs versus Usage of Custom AI Models

Maik Bastian[1], **Bernhard Esslinger**[2,*], **Eckehard Hermann**[3], **Nils Kopal**[4], **Harald Lampesberger**[3]
[1] CrypTool Project
[2] University of Siegen, Germany
[3] University of Applied Sciences Upper Austria, Austria
[4] Hochschule Niederrhein, Germany
[*] bernhard.esslinger@cryptool.org

Abstract

In the field of cryptography, identifying the type of cipher used in an encrypted message is crucial to effective cryptanalysis. Thus far, from a machine learning perspective, this classification problem has been tackled using specifically designed models, such as the Neural Cipher Identifier (NCID), which require data generation and model training capabilities. The recent advent of Large Language Models (LLMs) raises the following question: Can this classification problem be approached more effectively through prompt engineering? This paper explores various generic strategies for prompt engineering, such as chain-of-thought and in-context learning, by evaluating thousands of generated prompts for classical ciphers using open-source LLMs (on an Nvidia DGX system) and ChatGPT (via a browser interface and API). The classification accuracies achieved through these prompting techniques are compared with those obtained by NCID. Although our findings indicate that NCID still significantly outperforms the use of LLMs for cipher-type detection, the latter offers a more accessible approach to cryptography tasks. Both methods can benefit from domain-specific knowledge in cryptanalysis, highlighting the importance of expert input in improving initial classifications and handling complex cipher types.

1 Introduction

By exploiting weaknesses in a cipher algorithm, a cryptanalyst can attempt to decrypt the ciphertext without knowledge of the key (ciphertext-only attack) or to find the key, when both plaintext and ciphertext are known (known-plaintext attack). For these attacks, the cryptanalyst must first discover the type of cipher used to encrypt the original message. With this knowledge, they can decide on specific techniques appropriate for discovering the message or key of the ciphertext. Since there are a large number of possible cipher types, an automatic tool that discovers the actual cipher type can be very useful. We want to explore prompt engineering with pre-trained large language models (LLMs) as an alternative to established machine learning (ML) models for the task of cipher type detection of classical ciphers.

Training ML algorithms on ciphertexts has proven to be quite successful in identifying the cipher type of a given ciphertext. Neural Cipher Identifier (NCID) (Leierzopf et al., 2022) utilizes ML architectures that employ feature engineering and feature learning approaches, trained with features designed by experts in cryptanalysis and features automatically discovered from ciphertexts, respectively. These models can detect 55 classical cipher types standardized by the American Cryptogram Association (ACA)[1] with an accuracy of 82.78% (Leierzopf et al., 2021). Later additions included the recognition of five World War II-era rotor ciphers, achieving accuracies of 75.82%.[2] Although ML architectures are able to classify ciphertexts accurately, training them requires significant resources and considerable expert knowledge. Although the feature learning approach requires less domain knowledge, significant software engineering expertise is still required to implement the training process. The preparation of data, assembly of models, and implementation of training and evaluation loops all require considerable programming knowledge. Once the models are trained, they need to be stored and made

[1] See: https://www.cryptogram.org/

[2] For details see: https://github.com/cryptool-org/ncid#extended-models-trained-for-recognition-of-aca-and-rotor-ciphers

Proceedings of the 8th International Conference on Historical Cryptology, pages 1-11
Poznań, Poland, 16-18 June, 2025

available to users, which requires dedicated hardware. All these requirements prevent curious, less-technical users from implementing their own cipher identification schemes. Additionally, the trained ML models can only classify the fixed set of cipher types that they were given as input in the training process. Therefore, ciphertexts of an unseen cipher type will be classified as the wrong cipher type.

The advent of LLMs presents new tools for text recognition and processing. This research project evaluates whether they provide new approaches for cryptanalysis and, therefore, for the recognition of cipher types. Our assumption is that the underlying Transformer ML architecture, with its self-attention mechanism (Vaswani et al., 2017), should be able to identify the relations between the characters of the ciphertexts. The presented work explores the performance of several LLMs in classifying ciphertexts using only prompt engineering and compares the resulting classifications to those of NCID. Using the ability of LLMs to learn new tasks without explicit training, known as in-context learning, the process of prompt engineering attempts to improve the output of an LLM solely by altering the textual input to the model (Brown et al., 2020). Enabled by their pattern matching capabilities (Mirchandani et al., 2023), LLMs show potential for the task of ciphertext classification. Using in-context learning of an existing LLM as well as prompt engineering, there is no need to train a specialized ML model, thus lowering the barrier for less technical users. Ideally, they can perform an analysis simply by writing a detailed prompt and submitting it via the interface of an existing LLM. Given the unstructured format of the prompts, we expect the LLMs to be more flexible about the selection of cipher types that they can classify.

Given these considerations, this paper aims to answer the following research questions:

RQ1 How capable are LLMs in classifying classical ciphers using prompt engineering techniques?

RQ2 How do the classifications of LLMs compare to those of specific self-trained ML models?

RQ3 Does the prompt engineering approach enable less-technical users to classify classical ciphers?

To answer these questions, we explore the capabilities of different well-known and performant LLMs in classifying the same 60 classical ciphers (55 ciphers standardized by the ACA as well as five World War II-era ciphers) as examined by NCID, along with a plaintext type for comparison. For modern ciphers like AES or Triple DES, this classification should not work, as their ciphertexts show significantly fewer identifiable features. However, as Gohr showed in (Gohr, 2019), ML-based cryptanalysis of modern ciphers can be effective if they are round-reduced. The explored LLMs include OpenAI's ChatGPT, Meta's Llama, and Mistral's Mistral and Mixtral models.

In Section 2, NCID and its trained ML models for ciphertext classification are described. Afterwards, in Section 3, the generation of the ciphertexts and our developed prompt engineering strategies are presented. Section 4 discusses the accuracies achieved using the different strategies and the tested LLMs. Lastly, in Section 5, the classifications using LLMs are compared to those of NCID, and the usefulness of LLMs for the classification of ciphertexts is assessed.

2 Self-trained ML-based Classification

Self-trained ML models have already been shown to provide useful classifications of classical cipher types. Nuhn and Knight (2014) used neural networks to classify 50 classical ciphers of the ACA. They were able to associate 58.5% of the evaluated ciphertexts with the correct cipher type. Sivagurunathan et al. (2010) classified the three ciphers Playfair, Hill, and Vigenère using neural networks as well. In the following paragraphs, NCID will be presented in more detail.

NCID uses multiple self-trained ML models to implement ciphertext classification. Initially, it was trained to recognize 55 classical ciphers standardized by the ACA (Leierzopf et al., 2022). A further addition enabled the classification of the five World War II-era ciphers: Enigma, M209, Purple, Sigaba, and Typex.[3] NCID can be used through the Web interface at: `https://www.cryptool.org/cto/ncid/`. The models were trained with millions of ciphertexts generated from plaintexts from the Gutenberg library[4].

[3]Details about these additions can be found at: `https://github.com/cryptool-org/ncid#extended-models-trained-for-recognition-of-aca-and-rotor-ciphers`

[4]See: `https://www.gutenberg.org/`

For the classifications of the ACA ciphers, the length of the ciphertexts was set to 100 characters. This number was not determined by systematic research, but because it demonstrated usefulness in previous explorations. Ciphertexts of this length can be identified by manual or computer analysis, while significantly shorter ciphertexts cannot be identified meaningfully by either method, as they lack sufficient information for an unambiguous statistical evaluation. For the further addition of the rotor ciphers into NCID, the length of the ciphertexts was changed to a variable length between 100 and 1000 characters. These longer ciphertexts are needed to differentiate between the rotor ciphers, reducing the uncertainty seen in classifications with shorter ciphertexts. Given the advanced cryptographic complexity of the rotor ciphers, shorter ciphertexts show too few recognizable features for precise identification.

The four ML architectures feedforward neural network (FFNN), random forest (RF), Naive Bayes (NB), and support vector machine (SVM) were trained using a feature engineering approach that takes advantage of statistical features developed beforehand with expert knowledge in the domain of cryptanalysis. These features calculate general statistics of the ciphertexts, like the frequencies of single or multiple letters, the index of coincidence, and specific features targeting individual cipher types. In addition to the feature engineering approach, the two ML architectures Transformer and long short-term memory (LSTM) are trained using a feature learning approach. This approach requires the models to derive the properties of the cipher types without input from expert knowledge, as they are trained solely on raw ciphertexts.

The actual training process was performed iteratively, with a given upper limit of iterations to perform: The models are trained until either this limit is reached, or the process is automatically stopped because the models have not improved for a while. For each iteration, the generated ciphertexts need to be prepared. First, they need to be mapped into a numerical representation to be processable by the ML models. For models using a feature engineering approach, the features of the numerical ciphertext representation are calculated. Models using the feature learning approach will use the numerical representation directly. Once the ciphertexts are processed, the models are trained with the processed ciphertexts and their corresponding cipher types as input. After the training process is completed, the models are evaluated on another previously unseen set of ciphertexts. This time, the models are only provided with the ciphertexts and have to predict the cipher type. The predicted cipher type of each ciphertext is compared to the actual cipher type, and the percentage of correctly predicted cipher types of all evaluated ciphertexts is printed as the model accuracy. The training process must be repeated until all ML models are trained. Finally, the trained models are integrated into an ensemble model, combining their strengths and achieving an accuracy of 75.82% in correctly classifying the 55 ACA and five rotor ciphers.

To achieve these accuracies, many iterations were spent developing the training pipeline, looking for and improving features, and tweaking the hyperparameters of the ML architectures. Depending on the ML architecture, the models were trained between a few hours and up to a week on Nvidia DGX-1 hardware[5]. The resulting models have a combined size of about 2.5 GB. After the training process is completed, the models can classify new ciphertexts on less powerful hardware. Although training can take multiple days, the classification of a new ciphertext takes only a fraction of a second on a normal web server.

3 Prompt Engineering Method

For the evaluation of prompt engineering techniques, the same classical cipher types, as supported by NCID, are selected. These cipher types include the 55 ACA ciphers, the five rotor ciphers, and a plaintext type. The selection allows comparisons with the classifications of NCID and possibly other tools. For simplicity, the plaintext type will not be differentiated from the cipher types in the rest of this text and will simply be referred to as another cipher type. Thus, a total of 61 "cipher types" are evaluated. Approximately one million ciphertexts are created from randomly generated keys and randomly selected English plaintexts from the Gutenberg library, as was done for the training of NCID. The ciphertexts have a fixed length of 100 characters according to the implementation of NCID without the rotor ciphers. Sec-

[5]For more details see: `https://github.com/cryptool-org/ncid#original-models-trained-for-recognition-of-aca-ciphers`

tion 4 provides a discussion of longer ciphertexts. When using variable-length ciphertexts, we observed a tendency of the LLMs to classify ciphertexts based primarily on their length. As a first step, all generated ciphertexts are randomly split into a training set and an evaluation set to prevent leakage of the ciphertext into the context of the classification prompts. Therefore, the evaluation set only contains the ciphertexts that will be classified.

3.1 Naive Approach

First, we tried a naive approach of supplying a simple prompt to the LLM for classification. Although this naive approach does not yield useful results, it demonstrates basic knowledge of the LLMs about some of the ciphers standardized by the ACA and the five rotor cipher machines of the World War II era. Box 1 shows such a naive prompt. The prompt lists the ciphertext and the basic instruction to classify this ciphertext as one of the investigated ciphers.

To improve on the naive approach, the structure and context provided in a prompt are imperative. In an iterative process, many parameters and prompt engineering techniques were tested. Zero-shot, few-shot, and chain-of-thought are examples of such prompt engineering techniques found in literature.

The following paragraphs detail three different strategies, developed by ourselfs for building prompts, using the most promising parameters and techniques. We will refer to these three strategies as the *standard classification strategy*, the *clustered classification strategy*, and the *binary classification strategy*. The results obtained from these strategies are compared in Section 4.

> You are a cryptanalyst. What is the type of cipher of the following ciphertext in quotes "vetntw ornottlyiyettrehotlayneteaofeenfrertrecsheehrrlsm shngoecosehgsaotareoentuitdfrhuidlaotfdinhdxo"? Your options are: "quagmire4", "grandpre", "nihilist_transposition", ...

Box 1: A naive prompt, instructing an LLM to classify the given ciphertext.

3.2 General Considerations

Some general challenges we found while designing prompts for cipher-type detection are the limited input size and the uncommon structure of the prompts. The input text to an LLM is converted into tokens. Depending on the tokenizer, a token can represent a word, a fragment of a word, or a single character (Ali et al., 2024). The size of the prompts is defined by the context window, which specifies the number of tokens an LLM can work with. In the case of ciphertext, single characters are often represented by a single token. Tokenizers based on Byte Pair Encoding (BPE) assign common word fragments or common words to their own tokens but use fallback tokens to represent the individual characters of uncommon character sequences (Sennrich et al., 2016). The random character sequences of the ciphertexts will, therefore, be mostly represented by such character-level tokens. This leads to a very rapidly increasing number of tokens, easily exhausting the context window. Therefore, the amount of additional context that can be given to the LLM is limited, restricting the achievable performance of the model (Huang et al., 2024). Delimiting the ciphertexts explicitly, by providing the samples in an XML-like format, has improved the classifications in our evaluations. An example of this format is shown in Box 2. The ciphertexts of each cipher type are embedded in tags with the name of the cipher type, and each ciphertext is inserted into tags for delimitation.

```
<ciphertext-samples>
  <vigenere>
    <ciphertext>
      MOFOUMZZWVENLQGMULVPR...
    </ciphertext>
    <ciphertext>
      LTIZKTEJCMXYOACRWVLQQ...
    </ciphertext>
  </vigenere>
</ciphertext-samples>
```

Box 2: The format of the ciphertext samples that are part of all prompting strategies.

3.3 Chain-of-Thought Prompting Technique

Each strategy of prompt generation utilizes the chain-of-thought (CoT) prompting technique. Using CoT is an established approach to improve the output of LLMs without the need for fine-tuning (Wei et al., 2022). This technique structures the overall problem into smaller, intermediate steps and provides solutions for each step along with the corresponding reasoning, thus explaining the

overall thought process (Wei et al., 2022). In the context of ciphertext classification, they help the LLMs learn how to analyze ciphertexts and structure their output. A CoT should instruct the LLM on the characteristics of a given cipher type that will be evident in ciphertexts of that type. In this paper, only LLM-generated CoTs are investigated as they have the advantage of not requiring domain knowledge or time-consuming design by experts. To construct such a CoT, an LLM must perform an analysis on its own. Box 3 illustrates an example of a prompt designed to create a LLM-generated CoT. The prompt includes a ciphertext and its corresponding cipher type. The LLM must generate a reasoning for why this ciphertext was encrypted by the cipher type. The ciphertexts for the CoTs are taken from the ciphertext training set. To influence the generated reasoning of the CoT, the prompt includes specific instructions to observe the characteristics of the ciphertexts. These instructions provide general advice, such as to look for patterns in the ciphertext. In addition, they include specific statistical properties useful for cryptanalysis, such as the frequency of letter distributions and the index of coincidence. An example of a CoT can be found at `https://www.cryptool.org/download/research-files/self-generated-cot.png`.

> You are an expert in cryptanalysis. What are the characteristics of the following ciphertext in quotes "OFGXZFSFZEVBUXJOOECOYSXTSJCUROFUHKNUVFRFFELKHFBSQUNQNONJGZHUVBIVMJGFTPDJYNQUSQMJKXZZDJRXZQUHYHDOHMVI" which indicate that it is of type "slidefair"? Compare and contrast it with these other cipher types: "quagmire4", "grandpre", "nihilist_transposition", ...
> To characterize the patterns of the ciphertext use your own language skills as well as cryptanalytical tools, like the IoC, frequency analysis of single letters and bigrams. Limit your output to 2 paragraphs!

Box 3: The structure of CoT prompts for all prompting strategies.

The CoTs are generated by the LLM in a preprocessing step performed before the actual classification. The process of creating the CoTs is illustrated in Listing 1. Before being stored on disk, the CoTs are augmented with a header indicating the key information. This header lists the ciphertext and the corresponding cipher type. Without the header, it cannot be guaranteed that the generated output of the LLM repeats the ciphertext that was given in the initial prompt. Without this ciphertext, the most crucial information of the CoT is missing.

3.4 Standard Classification Strategy

The structure of the prompts for the standard classification strategy is the result of many iterations aimed at improving the classifications. Figure 4 shows the general structure of these prompts, which also serves as the basis for the other classification strategies. Each prompt contains a single ciphertext from the evaluation set that is classified by the LLM as one of the 61 cipher types examined. The prompts start with an initial context for the LLM, which includes the task the LLM has to perform, its role, as well as the structure it is expected to find in the rest of the prompt. Following this initial context, the previously generated CoTs and ciphertexts samples are provided. The ciphertexts for the CoTs and samples are extracted from the training set once more. The prompt is finalized with the actual instruction for the task to be performed. This instruction contains the ciphertext to classify, as well as a list of all 61 possible cipher types that the LLM can choose as a classification result. The instruction references the provided CoTs and samples and it requests a specific output format to simplify processing, evaluation, and debugging of the generated classification.

With ciphertexts of length 100, one CoT per cipher type, and 20 samples per cipher type, the prompts have a length of approximately 104,000 tokens. This number of tokens would, for example, exceed the context window of Mistral's Mixtral 8x7B model.[6] For complete evaluation of the classification accuracy of all 61 cipher types, 61 prompts must be provided to the LLM.

3.5 Clustered Classification Strategy

To improve the standard classification strategy with its large prompt size and numerous cipher types, the clustered classification strategy was conceived. To apply this clustering, expertise in the cryptographic domain was needed, thus limiting the following approaches to users with this expert knowledge. For the clustered classification strategy, the 61 ciphers are divided into 12 distinct

[6]For documentation of Mistral's models see: `https://docs.mistral.ai/getting-started/models/models_overview/`

```
chain_of_thoughts = {}
for cipher_type in cipher_types:
    ciphertext = train_dataset.random_ciphertext(cipher_type)
    prompt = f"Explain why '{ciphertext}' is of type '{cipher_type}'."
    cot = llm.generate(prompt)
    chain_of_thoughts[cipher_type] = cot
```

Listing 1: Pseudocode for generating CoTs for all cipher types.

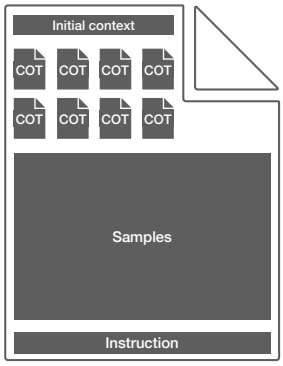


Figure 4: The general structure of the classification prompts used to classify a single ciphertext.

clusters. A table of the clusters and their corresponding cipher types can be found at `https://www.cryptool.org/download/research-files/cipher-clusters.png`. These clusters contain cipher types with comparable characteristics. Therefore, ciphers from different clusters are expected to be more easily distinguishable. For each complete evaluation, a single cipher type is randomly selected as a representation of the cluster. For example, the *Railfence* cipher is selected as a representative of the *Transposition* cluster. The ciphertexts of the selected cipher will be used in the CoTs, samples, and for the final classification. This means that effectively, 12 instead of 61 ciphers are compared by an LLM in each prompt. This strategy is expected to improve the classifications, as the differences in the ciphertexts will be greater than in the standard strategy and there will be less context in each prompt. Using a single CoT per cipher type and 20 samples per cipher type, the number of tokens per prompt is approximately 20,000 tokens. For a complete evaluation of classification accuracy, only 12 prompts must be executed, instead of the 61 prompts required by the standard strategy.

3.6 Binary Classification Strategy

The binary classification strategy is built on top of the clustered strategy, using only the 12 cluster types. The strategy is designed to reduce the amount of context needed in each prompt, helping the LLM to focus on the relevant characteristics of the ciphertext. Instead of augmenting the classification prompt with CoTs and samples of all examined cipher types, the binary classification strategy lists only CoTs and samples of a single cipher type in the classification prompt. The LLM has to determine whether the ciphertext matches this cipher type. For a complete classification of a single ciphertext, at least as many classifications as there are cipher types have to be performed. A complete evaluation of the 12 types of clusters would require at least $12 * 12 = 144$ prompts. Because the classification of a single ciphertext is split into several independent prompting steps, the LLM cannot directly compare the different cipher types and does not know its previous answers for the given ciphertext. Consequently, false positive results for a single binary classification are quite frequent. For better results, a single ciphertext is classified in multiple executions, ideally using different CoTs and samples for each execution. In the end, the LLM has to perform a reflection phase, using the results of the previous classifications to infer a single combined classification of the ciphertext. The prompt for the reflection phase contains the ciphertext, previous answers to classification executions, and samples of the cipher types answered in those previous executions. The LLM then has to select the overall answer. This reflection phase can also be performed after a single classification execution, but using multiple executions can improve accuracy. Listing 2 illustrates the process of the binary classification strategy.

The size of the classification prompts is approximately 3,500 tokens, and thus significantly smaller than in the previous strategies. The length of the prompts of the reflection phase depends on the number of positive classifications of the classi-

```
classifications = {}
for eval_cipher_type in cipher_types:
    eval_ciphertext = eval_dataset.random_ciphertext(eval_cipher_type)
    for tested_cipher_type in cipher_types:
        cot = chain_of_thoughts[tested_cipher_type]
        samples = train_dataset.load_random_samples(tested_cipher_type)
        prompt = f"""
            Analysis of {tested_cipher_type}:
            {cot}
            Samples of {tested_cipher_type}:
            {samples}
            Is '{eval_ciphertext}' of type '{tested_cipher_type}'?
            Answer with YES or NO.
            """
        classification = llm.generate(prompt)
        classifications[eval_cipher_type][tested_cipher] = classification
    if perform_reflection:
        positive_classifications = find_positive_classifications(
            classifications[eval_cipher_type]
            )
        samples = train_dataset.load_random_samples(positive_classifications)
        prompt = f"""
            The previous positive classifications for '{eval_ciphertext}'
            include the following cipher types: {positive_classifications}.
            Here are samples of these cipher types:
            {samples}
            Given these cipher types, what is the actual cipher type of
            the ciphertext '{eval_ciphertext}'?
            """
        final_classification = llm.perform(prompt)
        classifications[eval_cipher_type] = final_classification
```

Listing 2: Pseudocode for generating all classifications of the binary classification strategy.

fication phase and will be approximately 1,500 tokens long. Using a single execution per ciphertext, the total number of prompts that must be executed for a complete evaluation of the classification accuracy is the product of the number of cipher types multiplied by itself for the number of classification prompts, plus the number of cipher types for the number of reflection prompts. Therefore, with n as the number of cipher types, the total number of prompts is given by: $n*(n+1)$. With $n = 12$ clusters, the number of prompts equals 156.

4 Evaluation

The final results of the different prompting strategies described in Section 3 are assessed comparing the accuracies of the classifications of all strategies. The accuracies of the classifications are calculated after the complete execution of each of the three strategies, where the accuracy is the percentage of correctly identified cipher types out of all tested cipher types. The open-source models Meta Llama-3.1-70B-Instruct (quantized to 8-bit precision) and Meta Llama-3.1-8B-Instruct (unquantized), as well as OpenAI's ChatGPT 4o, were evaluated. While working on this research project, OpenAI had updated their ChatGPT 4o model. The specific model we used for the evaluation is ChatGPT 4o-2024-08-06. Additionally to the Llama models, two further open-source LLMs were tested: The quantized versions of Mistral-7B-Instruct-v0.3 and Mixtral-8x7B-Instruct-v0.1 with 16-bit precision. All these models were chosen because of their position on the huggingface leaderboard[7] and because they had already shown promising results for us in other tasks. However, of these four open-source models, only Llama-3.1-70B-Instruct and Llama-3.1-8B-Instruct showed acceptable ciphertext classifications in our tests.

All results were obtained using a Python script and the generated ciphertexts. This Python script simplified prompt generation and interaction with the LLMs. The open source models were executed on an Nvidia DGX H100[8], with eight Nvidia H100 GPUs with a total of 640 GB of VRAM, 2 Intel CPUs with a total of 112 cores, and 2 TB of RAM.

[7] For the current version of the leaderboard see: `https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard`

[8] Details about Nvidia DGX H100: `https://docs.nvidia.com/dgx/dgxh100-user-guide/introduction-to-dgxh100.html`

Each strategy was executed five times. The following results are the average of these five executions.

4.1 Evaluation with Short Ciphertexts

First, we discuss the results for ciphertexts with a length of 100 characters. Table 1 shows the classification accuracies of the Llama-3.1-70B-Instruct model, which was quantized to 8-bit precision. The best results are achieved using the binary classification strategy with an accuracy of 53.3%. Clustered classification shows less accurate results, while the standard classification strategy only achieves poor results. The evaluation of the Llama-3.1-8B-Instruct model is not detailed here for the sake of brevity. The classifications of this model are about 5 to 10 percent worse. However, it generates the results about three times faster than Llama-3.1-70B-Instruct quantized with 8-bit precision. For comparison, the superior classification results of NCID are shown in Table 3.

Table 1: Results of Llama-3.1-70B-Instruct quantized with 8-bit precision.

Strategy	Accuracy
Standard strategy	9.2%
Clustered strategy	43.3%
Binary strategy	53.3%

To compare the performance of open-source models, OpenAI's ChatGPT 4o was used as a commercial alternative. It achieved the best results compared to the open-source models. Table 2 presents the average accuracies for each of the three prompting strategies. Although the results are better than those of the Llama 3.1 models, the trend between the strategies behaves the same way as with Llama-3.1-70B. Only the clustered classification strategy and the binary classification strategy achieve better accuracy than random.

Table 2: Results of ChatGPT 4o-2024-08-06.

Strategy	Accuracy
Standard strategy	19.3%
Clustered strategy	46.7%
Binary strategy	58.3%

The standard classification strategy yields poor results, regardless of the model. In contrast, both the clustered classification strategy and the binary classification strategy significantly improve these results. We assume that the improvements achieved by these strategies in our evaluations can be attributed to the reduced set of cipher types that need to be recognized. Clustering of cipher types also provides the LLMs with ciphertexts with more distinct characteristics. The reduction of superfluous information in the prompts of the binary classification strategy, in comparison to the clustered classification strategy, further improved our results. Consequently, the binary classification strategy appears to be the best strategy, as it can provide all CoTs and ciphertext samples for each cipher type without overloading individual classification prompts. However, a disadvantage of this strategy is the considerable execution time required for both single classifications and complete evaluations of all cipher types.

4.2 Evaluation with Longer Ciphertexts

Experimentation with longer ciphertexts, such as with a length of 200 or 1000 characters, did not yield better classification results. Especially the 1000-character variant showed issues with regard to the increased number of tokens per prompt. Even in the binary classification strategy, the classification prompts have an average size of 22,000 tokens, compared to 3,500 tokens when using ciphertexts with 100 characters. The prompts of the standard classification strategy have an average size of 650,000 tokens when ciphertexts with 1000 characters are used, which exhausts the context window of many LLMs. There is a trade-off between the additional statistical information and patterns available in longer ciphertexts and the overall size of these more extensive prompts. At least with the models tested, the use of ciphertexts with more than 100 characters did not help the LLMs in identifying patterns.

4.3 Comparison with Specific Self-Trained Models

Compared to NCID, even the classifications achieved by ChatGPT 4o are unsatisfactory. In Table 3, the accuracies of the classifications using NCID are presented. The extended NCID is trained and evaluated with variable-length ciphertexts between 100 and 1000 characters, while the LLMs are only evaluated with ciphertexts that are 100 characters long. The standard evaluation uses the ensemble ML model and classifies all cipher types. The clustered evaluation uses the same clusters as in the prompt engineering strategies, and instead of retraining the existing NCID models, the accuracies of this evaluation are de-

termined after the models have made their predictions. In this case, a prediction is marked as correct if the predicted cipher type matches one of the cipher types in the correct cluster. Although the standard classification strategy could not successfully differentiate the 61 cipher types investigated, even when using ChatGPT 4o, NCID provides useful accuracies for classifications of all cipher types.

Table 3: Results of NCID.

Strategy	Accuracy
Standard evaluation	75.82%
Clustered evaluation	93.52%

5 Conclusion and Outlook

The presented work examined strategies for building prompts to detect cipher types and compared the classifications to ML-based NCID. We found that the format of the prompts is very important for these classifications. The CoT technique, together with samples of ciphertexts, provided LLMs with an additional context, improving their output of the classifications and the corresponding reasonings. The following subsections answer the research questions defined in Section 1.

5.1 RQ1: How capable are LLMs in classifying classical ciphers using prompt engineering techniques?

When evaluating the prompting strategies with different LLMs, it is evident that our most basic standard classification strategy using all 61 ACA and World War II-era rotor ciphers does not produce useful classifications. The accuracies achieved even by the most advanced models, classifying all 61 ciphers, only reach 20%. Applying a more distinct and simplified set of ciphers, as well as more complex prompting techniques, improves those results, producing average classifications. The clustered classification strategy, which uses only 12 types, achieves accuracies of 46.7% (see Table 2). The best classifications were achieved by the binary classification strategy. With this strategy, we reached an accuracy of 58.3%. As with the clustered classification strategy, the binary classification strategy considers a smaller set of cipher types compared to the standard classification strategy. Furthermore, the strategy reduces the size of the prompt by embedding only the CoTs and samples of a single cipher type, limiting the amount of superfluous information seen in each prompt.

5.2 RQ2: How do the classifications of LLMs compare to those of specific self-trained ML models?

Compared to classical ML techniques for ciphertext classification, as used by NCID, the classification accuracies achieved with our prompting strategies are significantly worse. Training specific ML models, especially with features developed using domain knowledge, can lead to great results. NCID achieves an accuracy of 75.82% when classifying all 61 cipher types. It only needs around 6 GB of VRAM to execute the trained models and can classify a ciphertext in fractions of a second. A disadvantage of ML models is the time-consuming training process. Although there is no need to train the existing LLMs when using in-context learning, classifying a single ciphertext takes significantly longer using LLMs. Moreover, considerable amounts of computing resources are needed to execute such a LLM. As an example, loading the full Llama-3.1-70B-Instruct model requires around 240 GB of VRAM. Relying on external providers to interface with an LLM removes the need for such hardware but introduces some disadvantages. In addition to the associated cost and potential usage limits, there is no guarantee that a hosted model will not be altered or replaced by the provider. While working on this research project, OpenAI had released a newer version of their ChatGPT 4o model, which generated worse output for the same prompts. Table 4 presents a comparative analysis of the advantages and disadvantages of employing prompt engineering with existing LLMs, as opposed to training specific ML models for the classification of ciphertext.

5.3 RQ3: Does the prompt engineering approach enable less-technical users to classify classical ciphers?

A big advantage of using LLMs for the task of cipher-type classification is the more approachable nature of the tools, decreasing the reliance on domain experts. Using the strategies detailed in this paper with sufficient samples of ciphertexts with known types, anyone can implement a classification scheme simply by providing written instructions to an existing LLM. The textual representation of the output can be helpful in inspecting the thought process of the LLM and for iterating

Table 4: Comparison of self-trained ML models and LLM prompt engineering for ciphertext classification, distinguished by development and production context.

Context	+ / -	ML models	LLM prompt engineering
Development	**Advantages**		• Classifications are interpretable • No need to train a model • Less reliance on domain experts
	Disadvantages	• Long training time • High hardware requirements for training • Domain experts and software engineers required	• Improvements to prompts can be non-obvious
Production	**Advantages**	• Best classification results • Fast classification • No dependency on external providers	• Classifications are interpretable
	Disadvantages	• Results hard to interpret	• Slow classifications • Dependency on external LLM providers, or • very high hardware requirements when self-hosted

the chosen approach. However, as shown by the unsatisfactory results of the standard classification strategy, cryptographic experts are still required to improve upon the initial classifications, especially when inspecting a multitude of possible cipher types.

5.4 Outlook

Future enhancements to LLMs could improve our prompt engineering approaches. Larger context windows would allow for longer prompts with more CoTs and samples with longer ciphertexts. There are further opportunities to optimize detailed prompting strategies. For example, the reflection phase of the binary classification strategy could be enhanced with statistics from previous evaluations, providing the LLMs with insight into their previous false classifications. Until these enhancements are developed, specifically trained ML models remain the better solution for ciphertext classification.

Acknowledgments

This work has been supported by Riksbankens Jubileumsfond, grant M24-0028: Echoes of History: Analysis and Decipherment of Historical Writings (DESCRYPT).

The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

References

[Ali et al.2024] Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Buschhoff, Charvi Jain, Alexander Weber, Lena Jurkschat, Hammam Abdelwahab, Chelsea John, Pedro Ortiz Suarez, Malte Ostendorff, Samuel Weinbach, Rafet Sifa, Stefan Kesselheim, and Nicolas Flores-Herr. 2024. Tokenizer choice for LLM training: Negligible or crucial? In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3907–3924, Mexico City, Mexico, June. Association for Computational Linguistics.

[Brown et al.2020] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

[Gohr2019] Aron Gohr. 2019. Improving attacks on round-reduced speck32/64 using deep learning. Cryptology ePrint Archive, Paper 2019/037.

[Huang et al.2024] Xijie Huang, Li Lyna Zhang, Kwang-Ting Cheng, Fan Yang, and Mao Yang. 2024. Fewer is more: Boosting llm reasoning with reinforced context pruning. In *EMNLP*, February.

[Leierzopf et al.2021] Ernst Leierzopf, Vasily Mikhalev, Nils Kopal, Bernhard Esslinger, Harald Lampesberger, and Eckehard Hermann. 2021. Detection of Classical Cipher Types with Feature-Learning Approaches. In *Data Mining. AusDM 2021. Communications in Computer and Information Science*. Springer Singapore.

[Leierzopf et al.2022] Ernst Leierzopf, Nils Kopal, Bernhard Esslinger, Harald Lampesberger, and Eckehard Hermann. 2022. A Massive Machine-Learning Approach For Classical Cipher Type Detection Using Feature Engineering. In *Proceedings of the 4th International Conference on Historical Cryptology HistoCrypt 2021*, jun.

[Mirchandani et al.2023] Suvir Mirchandani, Fei Xia, Pete Florence, Brian Ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. 2023. Large language models as general pattern machines. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2498–2518. PMLR, 06–09 Nov.

[Nuhn and Knight2014] Malte Nuhn and Kevin Knight. 2014. Cipher type detection. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1769–1773, Doha, Qatar, oct. Association for Computational Linguistics.

[Sennrich et al.2016] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August. Association for Computational Linguistics.

[Sivagurunathan et al.2010] Ganapathi Sivagurunathan, Velayutham Rajendran, and Dr. T. Purusothaman. 2010. Classification of substitution ciphers using neural networks. In *IJCSNS International Journal of Computer Science and Network Security*.

[Vaswani et al.2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

[Wei et al.2022] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903.

New records for Playfair solutions

Richard Bean
School of Electrical Engineering
and Computer Science
University of Queensland
Australia
r.bean1@uq.edu.au

Louie Helm
RockstarResearch.com
louiehelm@protonmail.ch

Abstract

We give solutions to the 24 letter and 22 letter Playfair challenges proposed in Dunin et al. (2022) A number of methods were tried combining successful approaches of previous solvers, introducing new ideas while using letter-level and word-level approaches.

We used vanilla and positional n-gram models for n values from 6 up to 10.

However, these did not greatly assist to distinguish the intended solution from other high-scoring solutions. The most effective discriminative approach involved using a multi-terabyte-scale, unpruned large language model from Buck, Heafield, and Van Ooyen (2014) which moved the solution in each case into the top 5,000 ranked possibilities.

1 Introduction

Beginning in 2019, Schmeh proposed Playfair cipher challenges on his classic cipher blog, Klausi's Krypto Kolumne (Schmeh (2019)). Previous authors such as Deavours (1977) had examined the concept of the "unicity distance" for various ciphers.

Deavours stated: "The *unicity point* of a cipher is the message length beyond which decipherment using a known system becomes a unique process. For messages shorter than the unicity point distance, plural decipherments are the rule and the would-be cryptanalyst has no possible method of selecting the correct decipherment from the many available ones."

He estimated the unicity distance of the Playfair cipher in English as 22.69 letters.

The first challenge cipher was 50 letters of ciphertext. The blog post title was "Playfair cipher: Is it unbreakable, if the message has only 50 letters?". On the same day of the post, George Lasry posted the solution.

The next 40 letter challenge was posted and solved on 8 December 2018 by Nils Kopal (described at Lasry (2019)).

Next, on 15 April 2019, a 30 letter challenge was posted. Magnus Ekhall posted the solution on 3 September 2019.

On 10 September 2019, a 28 letter challenge was posted which Magnus Ekhall solved on 11 November 2019.

Finally, Konstantin Hamidullin solved a 26 letter challenge (22 November 2019) on 17 December 2019.

Most recently, Schmeh posted a 24 letter challenge on 27 January 2020. On 13 August 2021, in a paper explaining previous challenges Dunin et al. (2022) proposed another 22 letter challenge. This length was below Deavours' estimated unicity distance.

Despite previous solvers posting potential solutions, no definitive solution was found.

2 Method

2.1 Cryptanalytic Workflow

We noted that the previous solutions had not used n-gram statistics with $n > 6$. For example, the paper Bean (2020) gave examples of two classical cipher challenges solved in 2019 through the use of 6-gram English statistics. It was expected that 8-gram statistics, in particular, would provide much greater solving power in the case of Playfair ciphers.

Previous successful solver techniques had often been based on the simulated annealing approach published by Cowan (2008) (example C++ code at Cowan (2009) and C code at Lyons (2012). This approach is also used in CrypTool Kopal (2018) with C# code.

Proceedings of the 8th International Conference on Historical Cryptology, pages 12-17
Poznań, Poland, 16-18 June, 2025

Word Count	Score	Text
4	460	WAIT FOR FURTHER INSTRUCTIONS
6	763	STAY WHERE YOU ARE UNTIL THURSDAY
8	1106	TAKE THE LAST TRAIN TO YORK ON SUNDAY
9	1061	MEET YOU TOMORROW AT FOUR TWENTY AT MARKET PLACE
11	1491	WHILE IN PARIS I RECEIVED ORDERS TO REPORT TO GENERAL FOSTER

Table 1: Initial 2-word-gram scores of previous challenges.

We combined previous techniques (simulated annealing based on Cowan (2008), plus tabu search to avoid examining past solutions) with 8-gram statistics. We used C code and "libpthread" for multithreaded execution with up to 96 threads. With this approach, we were able to solve all previous challenges relatively quickly.

This is despite the observation of Lasry (2018) that higher n values are a double-edged sword: more selective, but less resilient to key errors.

We also tested scoring with 2-word-grams, with an initial dictionary of 20,000 common English words. We found the maximum score possible by splitting up each solution into known English words using dynamic programming. Each pair of words was scored by frequency, logged and then scaled to one byte. Table 1 shows the word-gram scores of previous challenges. Linear regression gave a trend line with Score = 141.WordCount - 96. As an initial thresholding technique, we examined all results with a score at least 141.WordCount - 250.

2.2 Higher n-gram models for letters

In the development of the software package "AZdecrypt" Oranchak (2023) two users (Jarlve i.e. Jarl Van Eycke and Beijing House i.e. Louie Helm) had built n-gram statistics for n values up to 10. The statistics were stored using log frequency values and scaled to one byte per n-gram. Thus, the 7-gram table used $26^7 = 8,031,810,176$ bytes or approximately 8 GB of memory.

Tables with $n > 7$ store values for only a subset of the n-grams. Support for 7 and 8-grams was added to the AZdecrypt FreeBASIC source from April 2017 (version 1.03). The 7-gram statistics were published in version 1.14 (March 2019) and 8-gram statistics in version 1.15 (May 2019).

The 8-gram models released in 2019 at first stored $125,000^2$ (Jarlve) or $131,071^2$ (Beijing House) values. The most common 125,000 or 131,071 4-grams index the first and last half of the 8-grams. Later, Beijing House released further refined 8-gram models in 2020 and 2024 that each had improved accuracy while also shrinking index sizes to 120,000 and then 106,290 respectively for an ultimate in-memory size of 10.5 GB.

Similarly, the newly created 9-gram models are also derived from the same 10 TB corpus of English text used to create the newest 8-gram models. Their internal structure is slightly more complex and relies on using independent, variable-sized 4-gram indexes (between 46,952 and 106,290) for the first 4 and last 4 letters of each 9-gram, with 26 unique tables for each possible middle value. However, the RAM requirement for these is 88.8 GB and is only intended for "prosumer" computers with at least 96 GB of memory. These values were published in version 1.24 (January 2024).

Finally, the 10-gram values were developed using the most common 300,000 5-grams. The compressed file size is 14,661,017,462 bytes and uncompressed is 90,000,600,001 bytes. These were privately available from 8 May 2024.

We also calculated 8-gram and 9-gram files in a "positional" form specifically for the purposes of this challenge. That is, the first m letters of a sentence starting at a specified offset, and the last m letters of a sentence starting at a specified offset counted from the end. These were calculated for $1 \leq m \leq 8$, that is 16 different positional m-grams.

In addition to the n-gram letter scoring, we also improved the initial 2-word-gram and 3-word-gram scoring using groups of 2 and 3 words. The new 2-word-grams used the most frequent 65,535 English words and different 3-word-grams models used the most frequent 2,000 or 3,400 words in English.

Then after several sweeps using these positional 8-gram models and then adding in scoring from the modest 2-word and 3-word models, although the highest-scoring potential solutions seemed to qualitatively be turning into more natural or plau-

sible potential solutions, there still wasn't a single, clearly distinguishable solution at the top of our scoring lists.

3 Results

3.1 Top Scoring Candidates

On 20 April 2023 the top scoring initial solutions using 8-gram statistics were these.

```
host emotional after Athens
gesture perhaps we can't do it
offense strategy in the sun
you tended the larger units
never described him it shows
hope might have asked aloud
woman said the Pacific poem
```

For the 22 letter solution, the top-scoring solutions were:

```
Marco publishing heresy
firm developed the brief
lured by some of their food
Guy take some of the dairy
unaware line of the major
```

We noted that the "firm developed the brief" solution was already seen online at `https://www.architectsjournal.co.uk/buildings/sarah-wigglesworth-extends-camden-primary-school-through-creative-collaboration`.

3.2 Extended Solutions

On 17 May 2023, we suggested more:

```
Yes had completion in the MiG
West in terms of the said bank
It's okay in the building shop
I cut something up and admire
Belong to this works for some
HIV positive chances Andrew
I stone people who are not big
Case asked for the girls a boy
No it's list of the best change
On orders of the same to vinyl
Spy out of action winter pays
```

On 6 February 2024 we suggested still another 24-letter solution:

```
we are creating a bold pilot
```

The phrase "creating a bold pilot" was seen in several places on the web.

For 22 letters, we found all the following on the live web or archives of the web.

```
would everyone care for it
during eyebrow services
mum marching the babies
the verdict would be half
tell me which publicity
```

More generic solutions not on the web were also found:

```
I do know that I must point
The weapon may be your God
```

Eventually, we resorted to large language models in English to assist with the scoring of complete sentences. This resulted in close alignment with solutions already proposed by Hamidullin on the Schmeh blog for the 24 letter challenge.

For instance, the Hamidullin solution "And perhaps he's collecting" scored very highly.

3.3 Language Models Employed

The "ultimate" English model available publicly on the net was from Buck, Heafield, and Van Ooyen (2014) based on CommonCrawl data.

The authors explained: "We use the 2012, early 2013, and "winter" 2013 crawls, consisting of 3.8 billion, 2 billion, and 2.3 billion pages, respectively." This was 23.62 TiB of English input with 500,262,522,509 unique n-grams (between 1-grams, 2-grams, 3-grams, 4-grams, and 5-grams)

For the 24 letter challenge, "gifts bestowed to her family", "private sector in making his", and "post from other sites using" were seen on the web. However, none of these could be considered complete sentences.

On 11 June 2024 we submitted the email shown in Figure 1 after scoring all possible solutions with two models. We provided files containing the highest scoring million solutions in each case.

The first was trained on "books3" data from approximately 200,000 English books (see Gao et al. (2020)) restricted to a vocabulary of about 140,000 words from "wlist_match7" from Vertanen (2018).

All instances of the letter "J" replaced by "I" in the vocabulary, and only "a", "I", and "o" were retained as single letter words. The "books3" data were then processed with the I/J replacement, resulting in 100.8 GB of data.

Then, a KenLM model "trie file" was built using 5-word-grams.

The trie file was about 163 GB, with 16,031,566,855 different n-grams.

The other was the Buck, Heafield, and Van Ooyen (2014) "CommonCrawl" model pre-

viously described, with a trie file of size 6.06 TB. (Note that the training data for this model did not have the "J" letters replaced by "I" as traditional Playfair cipher plaintext does.)

Despite its colossal size, this model can be run inside a very small memory envelope, and only needs a large SSD to store the uncompressed trie file at runtime. It contains not just 5-grams, but everything from 1-grams to 5-grams, completely unpruned and with Kneser-Ney smoothing. Both those features are key to it being so especially good at precisely scoring and discriminating between so many short algorithmically-generated English phrases.

3.4 Scoring and Filtering Criteria

While the program gave 22 or 24 letter blocks of output, we used word splitting to split them into all possible output sentences (using the 140,000 word vocabulary above). The solver program itself was run in parallel attempting to generate solutions between 3 and 9 words for word-scoring, and then further splitting was carried out on the initial output.

In total, after splitting, we scored about 4 billion possible 24 letter solutions and 1.8 billion possible 22 letter solutions.

The vast majority of these were grammatically incorrect and could not be considered standalone English sentences. Nor did they have an "imperative" or quasi-military tone as did previous challenge solutions.

Schmeh replied that the CommonCrawl model scoring had placed the intended solution for the 22-letter challenge in the top 5,000, and for the 24-letter challenge in top 2,000. He also stated that for the "books3" model results, the 22-letter challenge was in the top 10,000 and the 24-letter top million did not contain the correct solution. (This was due to an error in our splitting code as it chose "BE LOW" rather than "BELOW"). Further, he stated that "both messages might be relevant for a spy".

After a lengthy manual review of the top solutions, we determined that the 24 letter challenge solution was "FIND DEAD DROPS BELOW BRIDGE" and the 22 letter solution was "MONEY IS HIDDEN BEHIND HUT".

"FIND DEAD DROPS BELOW BRIDGE" was found exactly once in an output file produced using (non-positional) 9-gram scoring. The score was 503 and we had used a cut-off value of 500.

> For the 22 letter challenge, the top two in order in both is WOULD EVERYONE CARE FOR IT and PILLARS OF THE INSANITY, and the top 10 of both include AVOID VISUAL DEFICIENCY (a phrase often seen on web). In the top 20 both have THE VERDICT WOULD BE HALF (seen on web).
>
> For the 24 letter challenge, the top 20 of each both have AFTER THAT ABOUT CLOSING UP, GIFTS BESTOWED TO HER FAMILY, IN EVOLVING THE WORLD MUSIC, ITS HUGE PERFORMANCE MAGIC, THEY SAY THE LIQUOR HAD GONE, and WE ARE CREATING A BOLD PILOT.
>
> The top solutions for 22 and 24 in the books3 model WOULD EVERYONE CARE FOR IT and AND PERHAPS HES COLLECTING by a large margin. (APHC is line 84 in trie 24 solutions)
>
> A bit further down 22 solutions I like WAIT COME BACK TO US FIRST (line 848 in trie 22, line 102 in books3 22). It has a kind of imperative nature the other larger challenge solutions had, plus very simple words.

Figure 1: Email submitted 11 June 2024.

In contrast, "MONEY IS HIDDEN BEHIND HUT" was found 34 times in output files.

We used the CommonCrawl and books3 model, and the various n-gram letter models to assess the relative difficulty of the 22 and 24 letter challenges compared to the 26 letter challenge.

We determined that the 26 letter Playfair challenge solution "WAIT FOR FURTHER INSTRUCTIONS" was the top scoring solution in each KenLM model and occurred in the top 25 solutions of every n-gram letter model with $n \geq 4$. It was the top solution in every n-gram letter model with $n \geq 6$. In contrast, the 22 and 24 letter solutions (considering the highest-scoring million solutions) were never in the top 50,000 highest-scoring solutions. Thus hill-climbing or simulated annealing approaches designed to maximize n-gram letter scoring did not assist much.

3.5 Alternative language models

We had also tested the scoring from another large model we found online (described in Jansen et al. (2022)). This model available as a 10.8 GB binary file. It was trained on harmful and toxic language. The potential solutions were ranked in a quite different way to the other models. For instance, for the 22 letter solution, "I am here to convey the porn" and "I am seeing a lovely cunt" were high

scoring, and for 24 letter solution, "pin up with the naked booty" scored well. Thus, if a model could somehow have been trained on "spycraft" or "military" text only, with an imperative tone, it may have done well with the challenge.

3.6 Assessment

In retrospect, it seems that proposing a challenge with missing articles and connector words made it nearly impossible to distinguish between very common potential solutions and the intended correct solution for such a short cipher. By this we mean that the intended solution with connecting words (i.e. "Find the dead drops below the bridge" and "The money is hidden behind the hut") would have scored very highly indeed, but not met the length constraints.

4 Conclusion

The 22 and 24 letter Playfair challenges proved to be much higher in difficulty level than the previous challenges which ranged in length from 26 to 50 letters.

Using previous techniques such as n-gram letter and word scoring, even with values of n up to 10 and incorporating all additional techniques (tabu search, multithreading, simulated annealing) and new techniques (positional n-grams) was not sufficient to bring the intended solutions anywhere near the highest scoring results.

For the 24 letter challenge, the intended solution was seen only once in more than a year of trying different scoring methods.

The Parker Hitt quote about cryptanalytic success requiring "perseverance, careful methods of analysis, intuition, and luck" (in order) proved true here. This was especially true for the 24 letter solution.

In practical terms, it was not possible to identify the intended unique solution, even though for the 24 letter solution, the length exceeded Deavours' estimated unicity distance for the Playfair cipher in English.

Another practical consideration for challenges of this nature in future is that the models used to assist solution may well now have been trained on AI-generated content, which may render them less suitable for human generated challenges. Projects such as "Paracrawl" of Bañón et al. (2020) attempt to improve upon previous crawls while avoiding these pitfalls. The cost of training a new model with such data would now be substantially lower than for the 2014 CommonCrawl model used here, assuming sufficient compute power was available.

5 Acknowledgements

We would like to thank Klaus Schmeh for constructing the challenges, and the University of Queensland High Performance Computing Resources for use of their "Bunya" supercomputer.

References

Bañón, Marta, Pinzhen Chen, Barry Haddow, Kenneth Heafield, Hieu Hoang, Miquel Esplà-Gomis, Mikel Forcada, et al. 2020. "ParaCrawl: Web-scale acquisition of parallel corpora." Association for Computational Linguistics (ACL).

Bean, Richard. 2020. "The Use of Project Gutenberg and Hexagram Statistics to Help Solve Famous Unsolved Ciphers." In *HistoCrypt*, 31–35.

Buck, Christian, Kenneth Heafield, and Bas Van Ooyen. 2014. "N-gram counts and language models from the common crawl." In *Proceedings of the Language Resources and Evaluation Conference 2014*, 3579–3584.

Cowan, Michael J. 2008. "Breaking short Playfair ciphers with the simulated annealing algorithm." *Cryptologia* 32 (1): 71–83.

Cowan, Michael J. 2009. `http://www.mountainvistasoft.com/cryptoden/index.php/algorithms/churn-algorithm/18-simulated-annealing.html`.

Deavours, Cipher A. 1977. "Unicity points in cryptanalysis." *Cryptologia* 1 (1): 46–68.

Dunin, Elonka, Magnus Ekhall, Konstantin Hamidullin, Nils Kopal, George Lasry, and Klaus Schmeh. 2022. "How we set new world records in breaking Playfair ciphertexts." *Cryptologia* 46 (4): 302–322.

Gao, Leo, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, et al. 2020. "The pile: An 800gb dataset of diverse text for language modeling." *arXiv preprint arXiv:2101.00027* .

Jansen, Tim, Yangling Tong, Victoria Zevallos, and Pedro Ortiz Suarez. 2022. "Perplexed by quality: A perplexity-based method for adult and harmful content detection in multilingual heterogeneous web data." *arXiv preprint arXiv:2212.10440* `https://huggingface.co/oscar-corpus/harmful-kenlms/blob/main/en.binary`.

Kopal, Nils. 2018. "Solving Classical Ciphers with CrypTool 2." In *HistoCrypt*, 149–010.

Lasry, George. 2018. *A methodology for the cryptanalysis of classical ciphers with search metaheuristics*. Kassel university press GmbH.

Lasry, George. 2019. "Solving a 40-letter Playfair challenge with CrypTool 2." In *Proceedings of the 2nd International Conference on Historical Cryptology, HistoCrypt*, 23–26.

Lyons, James. 2012. "Cryptanalysis of the Playfair cipher." `http://practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-playfair/`.

Oranchak, David. 2023. "AZDecrypt." `https://github.com/doranchak/azdecrypt`. [Software].

Schmeh, Klaus. 2019. "Klausi's Krypto Kolumne." `https://scienceblogs.de/klausis-krypto-kolumne/`.

Vertanen, Keith. 2018. "Wordlists." `https://www.keithv.com/software/wlist/`. [Wordlists].

From Statistics to Neural Networks: Enhancing Ciphertext-Plaintext Alignment in Historical Substitution Ciphers for Automatic Key Extraction

Micaella Bruton
Stockholm University
micaella.bruton@ling.su.se

Beáta Megyesi
Stockholm University
beata.megyesi@ling.su.se

Abstract

Ciphertext manuscripts found in archival collections are often intermingled with plaintext manuscripts in various languages, making the manual analysis required to match the documents labour-intensive and complex. Automating the alignment of these texts to reconstruct corresponding cipher keys is therefore highly beneficial, particularly when handling large volumes of documents. This study introduces a novel approach using modern neural networks, specifically Long Short-Term Memory (LSTM) architectures, to develop an automated method for aligning homophonic substitution ciphertexts with plaintext. These neural models are compared to traditional statistical approaches, demonstrating that LSTMs achieve significant accuracy improvements, including perfect alignment for ciphertexts of 50 characters or less. Additionally, to facilitate practical application, a program has been developed to enable the upload of transcribed ciphertext and plaintext documents, using the optimized models to automatically align the texts and extract the substitution key.

1 Introduction

National libraries and archives worldwide house collections containing ciphertexts and cipher keys alongside manuscripts written in plaintext. Cryptanalysts frequently begin their work by attempting to align ciphertext with plaintext to recover the underlying cipher key. This process is inherently complex and time-consuming, underscoring the value of automation. Computational methods, whether statistical or neural, offer significant potential to streamline this process.

One prominent example is the DECODE database, which contains 3,285 ciphertexts and 5,717 keys, yet only 13.9% have been fully decrypted or matched to their respective keys (Héder and Megyesi, 2022; Megyesi et al., 2019; Megyesi et al., 2020). The decryption of such texts remains labor-intensive due to the scarcity of contextual information and the complexity of historical cipher systems. These challenges underscore the necessity for advanced computational tools to complement manual methods and accelerate cryptanalysis.

Modern computational methods, particularly neural networks, offer unprecedented opportunities to address these challenges. By leveraging their ability to process sequential data and identify complex patterns, neural networks can assist in systematically aligning ciphertext with plaintext. Text alignment is a crucial step in systematic decryption, enabling cryptanalysts to reconstruct encrypted messages, extract keys, and gain insights into historical communications.

This paper explores the application of Long Short-Term Memory (LSTM) models for aligning ciphertext and plaintext in historical homophonic substitution ciphers. LSTMs are particularly well-suited for this task due to their ability to capture long-range dependencies in sequential data. By evaluating the efficacy of these models and comparing them to traditional statistical approaches, this study aims to bridge the gap between modern computational advancements and the unique challenges posed by historical cryptanalysis.

Contributions

The primary contributions of this work are as follows:

- to evaluate the performance of LSTM models in character-level alignment for historical ciphertext decryption;

Proceedings of the 8th International Conference on Historical Cryptology, pages 18-29
Poznań, Poland, 16-18 June, 2025

- to investigate the extent to which neural models reduce the reliance on preprocessing techniques, such as chunking, which involves dividing text into smaller segments to facilitate analysis and processing, often at the cost of losing contextual information and introducing potential misalignments in the data;
- to assess the impact of incorporating synthetic data on improving model generalization and performance on historical datasets;
- and, to develop and release a user-friendly program that enables individuals without coding expertise to perform automatic text alignment and key extraction methods on their personal computer.

2 Background

2.1 Text Alignment

Text alignment has long been a foundational task in Natural Language Processing (NLP), commonly used to align tokens and sentences across languages for applications such as machine translation, information retrieval, text entailment, and question answering (Oakes and McEnery, 2000; Véronis and Langlais, 2000; Zha et al., 2024; Bahdanau, 2014; Semmar and Fluhr, 2007). In the context of cryptanalysis, text alignment is equally critical, aiding in the decipherment of unknown or partially understood ciphers. This process involves matching sequences of ciphertext to potential plaintext equivalents by comparing encrypted text to known patterns, linguistic structures, or historical texts.

Unlike typical NLP tasks, where white-space often serves as a natural delimiter for tokens, historical ciphertexts frequently lack such boundaries. The removal of white-space complicates tokenization, as current systems predominantly rely on tokens or words for alignment. Without clear separations, the task shifts from aligning distinct tokens to aligning continuous sequences, a significantly more complex undertaking. This challenge is often further compounded as code groups in the ciphertext symbol sequence may vary in length, often comprising 2-, 3-, or 4-digit codes (Megyesi et al., 2024). Such variability adds another layer of difficulty, as decipherment systems must accurately segment and align symbols of different lengths without prior knowledge of their boundaries.

Table 1 provides an example of aligned text, illustrating multiple ciphertext characters—represented here as double-digit numbers—are mapped to the same plaintext character as an added security measure. This practice, known as homophonic substitution, underscores the importance of text alignment in detecting recurring patterns and facilitating systematic decryption. By addressing these challenges, advanced alignment techniques can enable cryptanalysts to reconstruct cipher keys and derive meaningful insights from encrypted communications.

...	58	**92**	33	**23**	32	96	37	**92**	**28**	91	**19**	...
...	A	**M**	O	**N**	G	W	Y	**M**	**M**	E	**N**	...

Table 1: Aligned Text Showcasing Homophonicity

2.2 Traditional Cryptanalysis & Statistical Approaches

Cryptanalysis has traditionally relied on manual techniques such as frequency analysis, pattern recognition, and linguistic analysis. These methods, while effective in solving simple substitution ciphers, face significant challenges when applied to more complex encryption systems, such as homophonic ciphers or those employing multiple alphabets (Megyesi et al., 2023a). The complexity of such systems significantly increases the effort required for decryption, often making manual methods impractical for large-scale or highly intricate cryptographic challenges.

A notable example is the Zodiac Killer's cipher, whose decryption required over 50 years to complete, even with significant advancements in computational power and collaborative efforts (Oranchak et al., 2024). This highlights the limitations of traditional cryptanalytic methods and underscores the necessity of integrating modern computational techniques to address these challenges more efficiently.

Statistical models have been used to automate parts of the text alignment process. IBM alignment models, originally developed for automatic machine translation, have been widely used used for years and have achieved high accuracy in text alignment tasks, but require significant preprocessing (Brown et al., 1993; Och and Ney, 2003; Koehn et al., 2003; Boglind, 2024). Boglind (2024) evaluated statistical IBM models on ciphertext/plaintext alignment for texts up to character length 2400. Though the best results reported

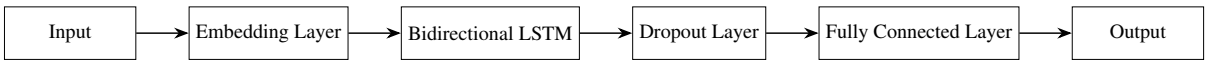


Figure 1: Model Architecture, Illustrating the Flow from Input Sequences to Output Predictions.

near-perfect alignment, this performance relied on segmenting the text into smaller segments, a process known as chunking (Boglind, 2024). While chunking enhances alignment accuracy, it can introduce significant computational overhead, particularly when applied to large-scale datasets or resource-constrained environments, potentially limiting its feasibility for real-time processing. Additionally chunking increases the risk of misalignment by reducing contextual information, which is crucial for accurate alignment. The choice of chunk length also has a substantial impact on performance; for example, increasing the chunk size from 5 to 40 resulted in a more than 70% decrease in the *F1* score; these baseline results can be found in Figure 2 (Boglind, 2024). While these statistical methods are effective when applied to preprocessed and annotated data, their utility is limited in the context of historical ciphers, which often lack tokenized and annotated data.

Despite their utility, these statistical methods face limitations in addressing the unique challenges of historical ciphers. Historical texts often lack token boundaries, exhibit irregular symbol usage, and are prone to transcription errors (Megyesi et al., 2023a). These irregularities complicate alignment tasks, rendering statistical models less effective, as they rely on clearly defined patterns and annotated data. This highlights the need for more adaptive methods to overcome these challenges.

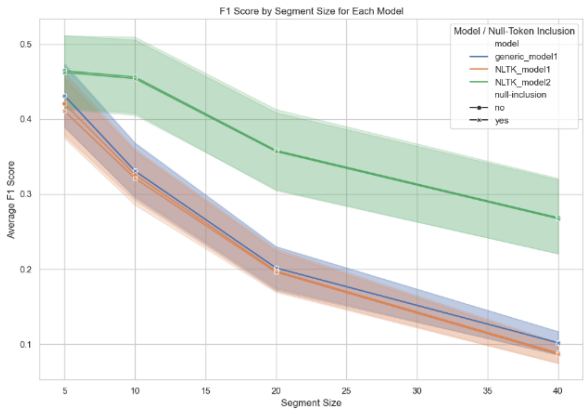


Figure 2: Boglind (2024)'s Results - *F1* Score for Segment (Chunk) Size

2.3 Neural Approaches in Cryptanalysis

Neural networks have demonstrated significant potential addressing the limitations of traditional methods. Sequence-to-sequence models, such as LSTMs, excel at processing sequential data and capturing long-range dependencies, making them well-suited for tasks like text alignment. Applications of neural networks to cryptanalysis have largely focused on image-to-text alignment or transcription tasks, however, their application to ciphertext/plaintext alignment remains underexplored (Torras et al., 2021; Fischer et al., 2011; Torras et al., 2023).

Neural models are well-equipped to address many of the additional challenges presented by historical ciphers (Megyesi et al., 2023a). Unlike traditional statistical models, neural models offer greater flexibility and adaptability in regards to preprocessing and annotated data, particularly when trained on both real-world and synthetic datasets, allowing them to generalize and adapt to complex alignment tasks. Synthetic datasets, which are generated computationally to simulate realistic patterns found in historical ciphers, are particularly valuable in augmenting training data when real-world examples are limited. By incorporating both types of data, neural models can better handle the irregularities and ambiguities inherent in historical cryptanalysis.

Recent studies have highlighted the potential of neural methods for historical cryptology. For example, Megyesi et al. (2023b) demonstrated that language models trained on century-specific historical texts improved the decryption of homophonic substitution ciphers. These findings underscore the importance of incorporating historical context into neural models to enhance their performance on historical datasets.

3 Method

To align each ciphertext character with its plaintext equivalent, models were trained on homophonically encrypted texts using full sequences of fixed lengths—50-, 100-, and 200-

Text Length	Train Data	Embedding Dim	Hidden Size	Layers	Dropout	Learning Rate	Batch Size
50	Real-World	109	196	3	2.97×10^{-1}	5.45×10^{-4}	59 samples
	Synthetic	246	104	3	3.56×10^{-1}	3.77×10^{-4}	49 samples
	Joint	190	225	3	3.36×10^{-1}	6.40×10^{-4}	43 samples
100	Real-World	204	247	3	3.42×10^{-1}	1.34×10^{-3}	19 samples
	Synthetic	149	253	3	3.83×10^{-1}	1.27×10^{-3}	37 samples
	Joint	264	269	3	4.47×10^{-1}	8.04×10^{-4}	24 samples
200	Real-World	117	184	2	3.26×10^{-1}	1.70×10^{-3}	20 samples
	Synthetic	215	291	3	3.33×10^{-1}	6.37×10^{-4}	27 samples
	Joint	149	263	2	3.32×10^{-1}	1.45×10^{-3}	26 samples

Table 2: Hyperparameters for each model, as defined by Optuna (Akiba et al., 2019)

characters—without chunking, in order to maintain the integrity of the input data. This approach was designed to enable the models to learn both local and global patterns within the ciphertext, such as symbol-to-character correspondences, recurring symbol groupings, contextual alignments across sequences, and the statistical distribution of homophonic substitutions.

Three models were trained for each length using distinct datasets: ciphertext/plaintext pairs derived from real-world historical plaintext data, ciphertext/plaintext pairs derived from synthetic historical plaintext data, and a combination of both data types. Synthetic datasets were incorporated to measure their impact on model performance, providing a means to supplement the limited availability of historical data while exposing the models to diverse patterns and structures that replicate the characteristics of historical cipher texts.

The evaluation process involved testing the models across varying ciphertext-to-plaintext mapping ratios, ranging from 1:1 to 5:1. These ratios simulate the complexity of real-world historical ciphers. By including these varied mappings, the study assesses the models' ability to adapt to increasing levels of homophonic complexity and maintain alignment accuracy under diverse cryptographic conditions.

The project repository is available on GitHub[1].

3.1 Data

To train and evaluate the models, datasets were constructed using 35,000 English plaintexts, each with a length of 200 characters. Of these, 1000 texts were reserved for both the validation and test sets, ensuring that training, validation, and testing data remained distinct. For consistency, the same splits were used when training models on the 50- and 100-character sequence lengths, allowing for a direct comparison of performance across different sequence lengths.

Plaintext

Plaintext data was sourced from the HistCorp and Project Gutenberg corpora, restricted to English texts spanning the years 1350–1899 (Pettersson and Megyesi, 2018; Gerlach and Font-Clos, 2020). Synthetic plaintext sequences were generated using 5-gram models trained on these corpora. To prevent overfitting or memorization, all generated sequences were ensured to be unique and distinct from the original data. Post-generation, the datasets were inspected to verify the absence of duplicates. For joint datasets, an equal distribution of real-world and generated plaintexts was maintained to ensure balanced representation.

Ciphertext

The Python library ChronoFidelius[2] was developed as part of this work and used to homophonically encrypt all plaintexts. Each plaintext character was mapped to between 1 and 5 unique four-digit numbers. For the training datasets, the mapping was weighted by historical character frequencies to emulate realistic plaintext distributions, ensuring that the models did not benefit from frequency imbalances. For example, high-frequency characters were mapped to one of five numbers, while low-frequency characters were assigned a single number.

The evaluation datasets were encrypted using flat mapping ratios ranging from 1:1 to 5:1, defined as ciphertext_character:plaintext_character. Spaces were excluded to replicate the characteristics of real-world historical ciphertexts. Additionally, no transcription errors or noise were in-

[1] https://github.com/mbruton0426/ciphertext-plaintext-alignment

[2] https://github.com/mbruton0426/ChronoFidelius

troduced, ensuring the evaluation focused exclusively on the model's alignment capabilities under ideal conditions.

3.2 Model

The model architecture is shown in Figure 1. A bidirectional LSTM captures dependencies from both preceding and succeeding tokens, enhancing its ability to identify alignment patterns. This is followed by a dropout layer to mitigate overfitting, and a fully connected layer that outputs logits for classification. Training was performed using the AdamW optimizer and a Cross-Entropy loss function, with Xavier uniform initialization to promote weight stability (Kingma, 2014; Glorot and Bengio, 2010). Training was conducted on a single NVIDIA A40 GPU until convergence, stopping after five epochs without improvement in validation loss.

Hyperparameter optimization was performed using Optuna, with 50 trials evaluated to determine the best-performing configuration based on validation set performance. The TPESampler and MedianPruner were utilized to improve optimization efficiency (Akiba et al., 2019). Table 2 details the final hyperparameter configurations for each model.

3.3 Evaluation

Performance was assessed using accuracy and the *F1* score to evaluate the model's character-level alignment capabilities. These metrics account for both false positives and false negatives, which are critical in cryptographic applications. They are defined as follows:

- **Accuracy**: The ratio of correct character-level predictions to total character-level predictions;
- ***F1* Score**: The harmonic mean of precision; defined as $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$, where *precision* measures the proportion of correctly predicted positive instances out of all predicted positive instances, and *recall* measures the proportion of correctly predicted positive instances out of all actual positive instances. This metric provides a balanced assessment of model performance.

4 Results

Due to the large number of evaluation sets, complete results are provided in Appendices A, B, and C for models trained on 50-, 100-, and 200-character sequences respectively. Overall, shorter input sequences and lower ciphertext-to-plaintext mapping options led to improved performance. Models trained on a combination of real-world and synthetic data consistently achieved the best results on texts longer than 50 characters.

Key performance highlights are summarized below, with a primary focus on results obtained from real-world test data, as the ultimate goal of this study is to improve alignment and decryption performance on historical ciphers. The highest accuracy and *F1* score achieved for real-world test data were as follows:

- **50-character sequences** 100% accuracy and *F1* score
- **100-character sequences** 67.8% accuracy and 69.24% *F1* score
- **200-character sequences** 52.16% accuracy and 53.36% *F1* score

4.1 Text Length 50

Models trained on 50-character sequences demonstrated exceptional performance across all data types and key configurations. Notably, models trained exclusively on either real-world or synthetic data achieved perfect classification scores on real-world test data, indicating flawless alignment.

Models trained on a combination of real-world and synthetic data showed a marginal decrease in performance, though they maintained an accuracy and *F1* score over 90% in all cases.

4.2 Text Length 100

For models trained on 100-character sequences, performance declined slightly across all conditions compared to shorter sequences. However, real-world test data generally remained the highest-performing category, with models trained on a combination of real-world and synthetic data achieving the best results in this setting.

Models trained exclusively on synthetic data achieved the highest overall scores, with an accuracy of 83.91% and an *F1* score of 84.60%. In contrast, models trained solely on real-world data showed significant limitations, with top accuracy and *F1* scores of 47.28% and 47.46%, respectively. Models trained on both real-world and

synthetic data demonstrated superior generalization compared to real-world-only models, achieving top scores of 67.79% accuracy and 69.24% *F1*.

4.3 Text Length 200

As expected, models trained on sequences of length 200 faced the greatest challenges. Models trained on both real-world and synthetic data achieved performance comparable to real-world-only models trained on 100-character, with a peak accuracy of 52.58% and an *F1* score of 54.14%.

Models trained exclusively on synthetic data struggled to generalize to real-world data, resulting in consistently low scores across all metrics. Among models trained exclusively on real-world data, the highest-performing achieved an accuracy of 35.97% and an *F1* score of 37.11%.

4.4 Program Implementation

A stand-alone program will be released and linked to the project repository, operationalizing the best-performing models. This application allows users to upload two text documents, one representing ciphertext and the other plaintext, after which the program automatically aligns the texts and extracts the corresponding substitution key. Both the aligned text and the extracted key are returned as text documents.

This program offers an open-source, accessible tool for historical cryptanalysts. By providing a user-friendly interface, it eliminates the need for coding or advanced computational skills, making automated text alignment and key extraction more widely accessible.

5 Discussion

This study reveals a consistent pattern: LSTM models trained on shorter input sequences, particularly those incorporating a mix of real-world and synthetic data, achieve superior alignment performance across all configurations. In contrast, longer sequences pose significant challenges, especially for models trained exclusively on real-world data.

The following sections examine the implications of these findings for cipher alignment and decryption tasks and propose recommendations for future research to address the identified limitations.

5.1 Performance Trends by Text Length

Text Length 50

The highest alignment accuracy was consistently achieved by models trained on 50-character sequences. Configurations trained exclusively on either real-world or synthetic data reached perfect alignment scores, suggesting that shorter sequences provide sufficient contextual information for accurate alignment while minimizing the complexity introduced by longer sequences.

This observation aligns with manual cryptanalysis practices, where analysts often examine short text segments to identify alignment patterns efficiently. Furthermore, these results significantly outperformed baseline statistical approaches, as the best reported statistical models required chunking text into segments of five characters to achieve best performance (Boglind, 2024).

Text Length 100

As the input length increased to 100 characters, model performance declined across all configurations. However, models trained on both real-world and synthetic data significantly outperformed those trained exclusively on a single data type when tested on real-world data. This suggests that incorporating synthetic plaintext data enables the model to capture a broader range of alignment patterns, improving generalization as text length increases.

Text Length 200

The most significant challenges were observed with 200-character sequences. While models trained on joint datasets continued to outperform all other configurations, their overall performance was still markedly lower than that of models trained on shorter sequences. This suggests that the complexity of alignment patterns in longer sequences exceeds the capacity of the LSTM architecture to generalize effectively, even with the inclusion of synthetic data.

These findings highlight a fundamental limitation of LSTMs in long-text alignment tasks and suggest the need for more advanced architectures or complementary strategies, such as attention mechanisms, to perform performance such cases.

5.2 Comparative Analysis of Training Data Types

Across longer text sequences, models trained on joint datasets demonstrated superior generaliza-

tion compared to those trained exclusively on either real-world or generated data. The inclusion of synthetic plaintext data in the training set appears to provide necessary diversity to capture a wider array of alignment patterns and develop robustness in variations in text structure.

In contrast, models trained solely on real-world data exhibited limited variability, which constrained their performance on longer sequences. This reinforces the importance of incorporating synthetic data during training to mitigate overfitting and improve alignment accuracy. Joint datasets provide a balanced approach by combining the specificity of real-world data with the broader coverage of synthetic data, offering a particularly effective solution for shorter sequences where alignment challenges are less pronounced.

5.3 Implications for Real-World Contexts & Future Work

The findings of this study highlight the critical role of input length and training data composition in developing neural models for text alignment tasks. Shorter text sequences achieve high alignment accuracy, making them particularly useful for applications involving shorter ciphers in real-world contexts. However, aligning longer texts remains a challenge, suggesting that additional pre- and post-processing techniques or varying architectures may be beneficial. While LSTM-based models demonstrate strong alignment capabilities, their effectiveness diminishes with longer sequences, indicating a need for alternative approaches.

Future research into neural methods for ciphertext/plain text alignment should prioritize the incorporation of synthetic data into training sets, as this approach enables condensed datasets to better reflect real-world text patterns. Efforts to enrich real-world datasets with greater variability should also be pursued to enhance model adaptability. Additionally, exploring architectures beyond LSTMs, such as Transformer-based models, could offer improved performance on longer sequences due to their ability to model long-range dependencies more effectively. While attention mechanisms within LSTMs could enhance performance, fully Transformer-based approaches, including BERT or sequence-to-sequence Transformer models, may be more promising alternatives. While such models can require larger amounts of training data, the inclusion of synthetic data could mitigate this requirement. Furthermore, data augmentation and domain adaptation techniques may improve model generalization to unseen real-world data, while accounting for the complexities of historical ciphertexts.

The present study was restricted to English-language data; future work should extend the approach to other languages in order to assess cross-linguistic robustness. Moreover, introducing errors which are commonly encountered in historical ciphertexts, such as transcription inaccuracies and typographical errors, would increase the practical applicability of these models for real-world cryptanalysis tasks.

6 Conclusion

This study investigated the influence of input sequence length and training data composition on the alignment accuracy of LSTM models designed for ciphertext/plaintext alignment tasks. The results demonstrate that models trained on shorter sequences, particularly those leveraging a combination of real-world and synthetic data, consistently achieve the highest accuracy and *F1* scores.

Synthetic data emerged as a valuable resource, enhancing generalization and enabling models to better handle diverse alignment scenarios, particularly as input length increases. The results also highlight the critical importance of optimizing both training data composition and sequence length when developing alignment models for cryptographic applications.

Despite these advancements, certain limitations remain. The study focused exclusively on English-language data and idealized inputs, and extending this approach to other languages would be necessary to assess cross-linguistic and real-world robustness. Additionally, while the models performed well on shorter sequences, their effectiveness on longer texts remains constrained. Continued exploration of architectures beyond LSTMs, particularly those capable of modeling long-range dependencies, may offer improved scalability to address this issue.

By bridging modern neural approaches with the challenges of historical cryptographic analysis, this work provides a foundation for developing robust, accessible tools that can accelerate decryption tasks and deepen our understanding of past cryptographic practices.

Acknowledgments

The project is financed by the Swedish Research Council, partially by DECRYPT - Decryption of Historical Manuscripts (grant 2018-06074), and partially by The Swedish Graduate School of Digital Philology (grant 2022-06343). The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council (grant 2022-06725).

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Dzmitry Bahdanau. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Fredrik Boglind. 2024. Aligning historical ciphertext and plaintext using statistical machine translation methods. Bachelor's thesis, Department of Linguistics, Stockholm University.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Andreas Fischer, Volkmar Frinken, Alicia Fornés, and Horst Bunke. 2011. Transcription alignment of Latin manuscripts using hidden Markov models. In *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, pages 29–36.

Martin Gerlach and Francesc Font-Clos. 2020. A standardized project Gutenberg corpus for statistical analysis of natural language and quantitative linguistics. *Entropy*, 22(1):126.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

Mihály Héder and Beáta Megyesi. 2022. The Decode database of historical ciphers and keys: Version 2. In *International Conference on Historical Cryptology*, pages 111–114.

Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL 2003)*, pages 48–54. Association for Computational Linguistics.

Beáta Megyesi, Nils Blomqvist, and Eva Pettersson. 2019. The decode database: Collection of historical ciphers and keys. In *The 2nd International Conference on Historical Cryptology, HistoCrypt 2019, June 23-26 2019, Mons, Belgium*, pages 69–78.

Beáta Megyesi, Bernhard Esslinger, Alicia Fornés, Nils Kopal, Benedek Láng, George Lasry, Karl de Leeuw, Eva Pettersson, Arno Wacker, and Michelle Waldispühl. 2020. Decryption of historical manuscripts: The DECRYPT project. *Cryptologia*, 44(6):545–559, November.

Beáta Megyesi, Alicia Fornés, Nils Kopal, Benedek Láng, Michelle Waldispühl, Vasily Mikhalev, and Bernhard Esslinger. 2023a. *Historical Cryptology*. Artech House.

Beáta Megyesi, Justyna Sikora, Filip Fornmark, Michelle Waldispühl, Nils Kopal, and Vasily Mikhalev. 2023b. Historical language models in cryptanalysis: Case studies on English and German. In *Proceedings of the 6th International Conference on Historical Cryptology HistoCrypt 2023*, pages Published on May 30, 2023. Linköping University Electronic Press.

Beáta Megyesi, Crina Tudor, Benedek Láng, Anna Lehofer, Nils Kopal, Karl de Leeuw, and Michelle Waldispühl. 2024. Keys with nomenclatures in the early modern Europe. *Cryptologia*, 48(2):97–139.

Michael Oakes and Tony McEnery. 2000. Bilingual text alignment—an overview. *Multilingual corpora in teaching and research*, pages 1–37.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

David Oranchak, Sam Blake, and Jarl Van Eycke. 2024. The solution of the Zodiac killer's 340-character cipher. *arXiv preprint arXiv:2403.17350*.

Eva Pettersson and Beáta Megyesi. 2018. The HistCorp collection of historical corpora and resources. In *DHN 2018: The Third Conference on Digital Humanities in the Nordic Countries*, pages 306–320, Helsinki, Finland. University of Helsinki.

Nasredine Semmar and Christian Fluhr. 2007. Arabic to French sentence alignment: Exploration of a cross-language information retrieval approach. In *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources*, pages 73–80.

Pau Torras, Mohamed Ali Souibgui, Jialuo Chen, and Alicia Fornés. 2021. A transcription is all you need: Learning to align through attention. In Elisa H. Barney Smith and Umapada Pal, editors, *Document Analysis and Recognition – ICDAR 2021 Workshops*, pages 141–146, Cham. Springer International Publishing.

Pau Torras, Mohamed Ali Souibgui, Jialuo Chen, Sanket Biswas, and Alicia Fornés. 2023. Segmentation-free alignment of arbitrary symbol transcripts to images. In Mickael Coustaty and Alicia Fornés, editors, *Document Analysis and Recognition – ICDAR 2023 Workshops*, pages 83–93, Cham. Springer Nature Switzerland.

Jean Véronis and Philippe Langlais. 2000. Evaluation of parallel text alignment systems: The arcade project. *Parallel text processing: Alignment and use of translation corpora*, pages 369–388.

Yuheng Zha, Yichi Yang, Ruichen Li, and Zhiting Hu. 2024. Text alignment is an efficient unified model for massive nlp tasks. *Advances in Neural Information Processing Systems*, 36.

A Results: Models Trained on Data Length 50

Key Option	Train Data	Test Data	Accuracy	*F1*
1:1	Real-World	Real-World	1.0000	1.0000
		Synthetic	0.9162	0.9169
		Joint	0.9581	0.9582
	Synthetic	Real-World	1.0000	1.0000
		Synthetic	0.9192	0.9190
		Joint	0.9596	0.9595
	Joint	Real-World	0.9891	0.9890
		Synthetic	0.9141	0.9137
		Joint	0.9516	0.9513
2:1	Real-World	Real-World	1.0000	1.0000
		Synthetic	0.9152	0.9159
		Joint	0.9576	0.9577
	Synthetic	Real-World	1.0000	1.0000
		Synthetic	0.9181	0.9179
		Joint	0.9590	0.9589
	Joint	Real-World	0.9897	0.9895
		Synthetic	0.9135	0.9131
		Joint	0.9516	0.9513
3:1	Real-World	Real-World	1.0000	1.0000
		Synthetic	0.9147	0.9153
		Joint	0.9574	0.9575
	Synthetic	Real-World	1.0000	1.0000
		Synthetic	0.9177	0.9175
		Joint	0.9589	0.9587
	Joint	Real-World	0.9899	0.9897
		Synthetic	0.9128	0.9124
		Joint	0.9513	0.9511
4:1	Real-World	Real-World	1.0000	1.0000
		Synthetic	0.9145	0.9152
		Joint	0.9573	0.9574
	Synthetic	Real-World	1.0000	1.0000
		Synthetic	0.9175	0.9173
		Joint	0.9587	0.9586
	Joint	Real-World	0.9895	0.9893
		Synthetic	0.9128	0.9125
		Joint	0.9511	0.9509
5:1	Real-World	Real-World	1.0000	1.0000
		Synthetic	0.9144	0.9150
		Joint	0.9572	0.9573
	Synthetic	Real-World	1.0000	1.0000
		Synthetic	0.9174	0.9172
		Joint	0.9587	0.9586
	Joint	Real-World	0.9894	0.9892
		Synthetic	0.9127	0.9123
		Joint	0.9510	0.9507

Table 3: Performance of models trained on sequences of length 50 — Best results overall and for real-world test data are highlighted

B Results: Models Trained on Data Length 100

Key Option	Train Data	Test Data	Accuracy	*F1*
1:1	Real-World	Real-World	0.4728	0.4755
		Synthetic	0.4724	0.4747
		Joint	0.3642	0.3500
	Synthetic	Real-World	0.2556	0.2401
		Synthetic	0.5768	0.6078
		Joint	0.7078	0.7220
	Joint	Real-World	0.6779	0.6924
		Synthetic	0.6209	0.6330
		Joint	0.7075	0.7211
2:1	Real-World	Real-World	0.4724	0.4746
		Synthetic	0.4723	0.4736
		Joint	0.3640	0.3497
	Synthetic	Real-World	0.2560	0.2415
		Synthetic	0.5761	0.6067
		Joint	0.7073	0.7205
	Joint	Real-World	0.6777	0.6885
		Synthetic	0.6207	0.6314
		Joint	0.7073	0.7202
3:1	Real-World	Real-World	0.4724	0.4738
		Synthetic	0.4724	0.4746
		Joint	0.3642	0.3500
	Synthetic	Real-World	0.2560	0.2406
		Synthetic	0.5760	0.6061
		Joint	0.7069	0.7198
	Joint	Real-World	0.6773	0.6868
		Synthetic	0.5635	0.5794
		Joint	0.7069	0.7198
4:1	Real-World	Real-World	0.4723	0.4736
		Synthetic	0.4724	0.4747
		Joint	0.3641	0.3494
	Synthetic	Real-World	0.2556	0.2403
		Synthetic	0.5767	0.6071
		Joint	0.7069	0.7198
	Joint	Real-World	0.6772	0.6862
		Synthetic	0.6204	0.6303
		Joint	0.7069	0.7198
5:1	Real-World	Real-World	0.4724	0.4738
		Synthetic	0.4724	0.4746
		Joint	0.3641	0.3493
	Synthetic	Real-World	0.2558	0.2407
		Synthetic	0.8391	0.8460
		Joint	0.7073	0.7202
	Joint	Real-World	0.6774	0.6863
		Synthetic	0.6204	0.6304
		Joint	0.7073	0.7202

Table 4: Performance of models trained on sequences of length 100 — Best results overall and for real-world test data are highlighted

C Results: Models Trained on Data Length 200

Key Option	Train Data	Test Data	Accuracy	*F1*
1:1	Real-World	Real-World	0.3597	0.3711
		Synthetic	0.3560	0.3641
		Joint	0.2027	0.2056
	Synthetic	Real-World	0.0457	0.0546
		Synthetic	0.5491	0.5493
		Joint	0.2933	0.2981
	Joint	Real-World	0.5216	0.5336
		Synthetic	0.6105	0.6191
		Joint	0.2933	0.2981
2:1	Real-World	Real-World	0.3548	0.3619
		Synthetic	0.3547	0.3609
		Joint	0.2007	0.2028
	Synthetic	Real-World	0.0453	0.0536
		Synthetic	0.5500	0.5502
		Joint	0.2935	0.2984
	Joint	Real-World	0.5203	0.5310
		Synthetic	0.6094	0.6174
		Joint	0.2935	0.2984
3:1	Real-World	Real-World	0.3544	0.3605
		Synthetic	0.3547	0.3609
		Joint	0.1998	0.2016
	Synthetic	Real-World	0.0449	0.0528
		Synthetic	0.5495	0.5496
		Joint	0.2936	0.2985
	Joint	Real-World	0.5198	0.5291
		Synthetic	0.6087	0.6160
		Joint	0.2936	0.2985
4:1	Real-World	Real-World	0.3547	0.3609
		Synthetic	0.3548	0.3619
		Joint	0.1997	0.2015
	Synthetic	Real-World	0.0448	0.0528
		Synthetic	0.5497	0.5499
		Joint	0.2938	0.2986
	Joint	Real-World	0.5209	0.5306
		Synthetic	0.6095	0.6168
		Joint	0.2938	0.2986
5:1	Real-World	Real-World	0.3544	0.3605
		Synthetic	0.3547	0.3609
		Joint	0.1998	0.2014
	Synthetic	Real-World	0.0453	0.0533
		Synthetic	0.7011	0.7164
		Joint	0.2927	0.2975
	Joint	Real-World	0.5216	0.5336
		Synthetic	0.6125	0.6239
		Joint	0.2927	0.2975

Table 5: Performance of models trained on sequences of length 200 — Best results overall and for real-world test data are highlighted

Practical and Organisational Factors in the Development History of the Typex Cipher Machine and its Use at Bletchley Park

Thomas Cheetham
Bletchley Park Trust
The Mansion, Bletchley Park
Milton Keynes, United Kingdom
tcheetham@bletchleypark.org.uk

Abstract

The Typex was Britain's main cipher machine during the Second World War. The best-described Typex models are the Mark II and the compact Marks III and VI. However, there remain gaps in the Typex 'family tree'. This paper reviews the development history of Typex and describes several previously unknown models of Typex based on documents produced by the British Signals Intelligence agency, the Government Code and Cypher School, a major user of Typex while based at Bletchley Park during the Second World War. Although these models were not brought into widespread service, the documentation sheds useful light on the design process. The design of successive models of Typex, and their adoption or rejection, had less to do with cryptographic considerations than the various mechanical and practical problems involved in designing a reliable cipher machine compatible with the communications systems used by the British state and armed forces.

1 Introduction

The Typex was Britain's main cipher machine during the Second World War, and therefore a significant machine in the history of cryptography. Useful articles have been published on its functioning, development, manufacturing and use by Louis Kruh and CA Deavours and Ralph Erskine.[1] However, the existing literature does not describe all Typex models or provide a full description of Typex development. The main reason for this is that records are extremely scant. Some of the gaps can be filled by papers produced by the British Signals Intelligence agency, the Government Code and Cypher School (GC&CS), a major user of Typex while based at Bletchley Park (BP) during the Second World War. These shed light on several previously unknown models of Typex, and additionally, by indicating why these were unsuccessful, on what made a successful cipher machine.

2 The Development of Typex

All wartime Typex machines were rotor-based cipher machines working on the same essential principle as the well-known Enigma machine. Indeed, the Typex scrambler was a near-copy of a commercial Enigma D (specifically machine A320, which had been purchased by GC&CS for evaluation in 1926)[2] with the addition of two stationary rotors (stators) positioned to the right of the stepping rotors. The rotors were given additional stepping notches, and eventually made reversible. From November 1941 a 'plugboard'

[1] Louis Kruh and CA Deavours, 'The Typex Cryptograph', *Cryptologia*, 7/2 (1983), 145-66; Ralph Erskine, 'The Development of Typex', *The Enigma Bulletin*, 2 (1997), 69-86; Ralph Erskine, 'The Admiralty and Cipher Machines During the Second World War: Not So Stupid After All', *Journal of Intelligence History*, 2/2 (2002), 49-68.

[2] Two machines, A320 and A323, were sold to Britain. A320 was analysed by Hugh Foss of GC&CS and it was almost certainly this machine which was lent to the Typex development team in 1934. A323 was probably purchased separately by the Royal Navy. See UK National Archives (TNA) HW 25/6, 'Early Correspondence Relating to the Enigma Cipher Machine'; HW 25/14, 'The Reciprocal Enigma'; AVIA 8/356, 'A 352: Recommendation for an award to Mr. E.W. Smith in connection with Type "X" Cypher Machines and Accessories', n.d.; Frode Weirud, 'Enigma D versus Zählwerk Enigma', https://www.cryptocellar.org/enigma/e-prod-history/enigma-d-vs-zaehlwerk-enigma.pdf [accessed December, 2024]; David Kenyon and Frode Weirud, 'Enigma G: The Counter Enigma', *Cryptologia*, 44/5 (2020). All subsequent archival references are to files at TNA unless otherwise stated.

Proceedings of the 8th International Conference on Historical Cryptology, pages 30-40
Poznań, Poland, 16-18 June, 2025

was added. This probably allowed the reflector wiring to be altered.[3] The Typex Mark 22, which incorporated both an Enigma-type entry plugboard and a pluggable reflector, did not appear until 1948.[4]

The idea of marrying up the Enigma scrambler with Creed teleprinter parts was first proposed by Wing Commander Oswyn G Lywood, then serving as chief signals officer at Royal Air Force (RAF) Coastal Command, in 1934. When the official Interdepartmental Committee on Cipher Machine Development declined to take the idea forward, Lywood assembled a team including Flight Lieutenant Coulson, responsible for code and cipher security in the RAF Signals Directorate, and a Mr EW Smith, foreman at the RAF wireless workshops at Kidbrooke, to develop the machine for the RAF.[5] Machines were manufactured by the Creed teleprinter company.[6] GC&CS, which had a remit for the security of British cipher systems, has been criticised for failing to contribute sufficiently to machine development,[7] but it did participate in the development of Typex by providing advice and testing prototypes and was regarded by the RAF as a major partner in the process.[8] The responsible officer at GC&CS was Deputy Head Edward Travis, who dealt with cryptographic matters.[9] Lywood later thanked Travis as 'in the early days of my Typex Development Committee when security principles were involved you were good enough to help us out by either attending yourself or sending a suitable representative'.[10]

The Typex Mark I (as it was retrospectively designated) was a large online printing machine which 'enables cyphering to be done direct on to a telegraph line or a perforator for use on a wireless circuit giving a printed copy of both cypher and plain language version for filing'.[11] Pressure of time and funds ensured that 'every effort was made to utilise existing standard telegraph equipment', resulting in a 'somewhat bulky and immobile' apparatus. The machine incorporated two standard commercial teleprinters which 'extend on either side of a scrambler assembly comprising keyboard and drums, and are connected to it by multi-core cables'.[12] No machines or photographs are known to survive, but the arrangement was presumably similar to that of other cipher machines incorporating two electric typewriters and enciphering apparatus mounted to a desk, such as the Kryha *Elektrik* or the well-known US analogue of the Japanese 'Purple' machine. If it was desired to send messages by wireless, the cipher teleprinter could be replaced by a perforator which punched the message into Morse tape.[13] The teleprinter assembly completely replaced the Enigma lampboard; there was no remaining lampboard output.[14] Given the size and complexity of the Mark I, mass production was out of the question and only 29 were produced;[15] these had been installed at Air Ministry and headquarters of RAF Commands by the middle of

[3] ADM 1/27186, 'Review of the Security of Naval Codes and Cyphers', 32; AIR 10/4051, 'Type X Machine Mark VI'. No wartime reference to a Typex with entry plugboard has been found; wartime manuals and 1945 American cryptanalytic reports describe machines with only a pluggable reflector. See the reports at https://cryptocellar.org/rotor/typex/index.html [accessed April 2025].

[4] NK O'Neill and KJ Hughes, *History of CBNRC*, vol. V, https://luxexumbra.blogspot.com/2019/08/history-of-cbnrc.html [accessed April, 2025], paras. 20.7 and 20.15. The Mark 22 'was in fact a Mark II with modified scrambler unit'; later 'the security of Mark 22 was greatly enhanced by the addition of a cross-over plugboard.' The Mark 22 is mentioned in a 1947 British document: see NSA Friedman Collection, A2435939, 'Modifications to Improve the Security of the Combined Cipher Machine', 29/10/47, https://www.nsa.gov/Portals/75/documents/news-features/declassified-documents/friedman-documents/patent-equipment/FOLDER_120/41772139081122.pdf [accessed April 2025]. There was also the Mark 23, which could be converted into the Combined Cipher Machine (CCM).

[5] An additional member of the design team, Sergeant (later Flying Officer) Albert Phillip Lemmon, is mentioned for the first time in relation to the Mark II.

[6] Erskine, 'The Development of Typex', 70-2; Erskine, 'The Admiralty and Cipher Machines, 50-1.

[7] John Ferris, *Behind the Enigma: The Authorised History of GCHQ* (London: Bloomsbury, 2020), 102-4.

[8] AVIA 8/356, memorandum no. 22, 14/05/38.

[9] AG Denniston, 'The Government Code and Cypher School Between the Wars', *Intelligence and National Security*, 1/1 (1986), 53.

[10] HW 62/1, Lywood to Travis, 19/08/43.

[11] AVIA 8/356, memorandum no. 11, 29/12/37.

[12] AVIA 8/356, 'A403: Recommendation of Awards in respect of the development of the Type X Mark II Cypher Machine', n.d.

[13] I.e. using the Wheatstone system, rather than Baudot-Murray teleprinter code.

[14] AVIA 8/356, 'A 352: Recommendation for an award to Mr. E.W. Smith in connection with Type "X" Cypher Machines and Accessories', n.d. However, a lampboard-type device known as a 'shadowgraph' was provided for emergency use.

[15] AVIA 8/356, memorandum no. 68, 12/09/41.

1937 and officially superseded non-machine systems from 17 August that year.[16]

By the end of 1937, Lywood was already working on a simpler offline printing machine intended for use at lower headquarters. The Typex Mark II, which would become the most common variant, appeared in 1938. From a modern perspective, descriptions of the Mark II as simple, cheap and portable appear surprising: compared to its wartime competitors it was notably expensive to manufacture, a hassle to maintain due to its complex mechanics, and not readily man-portable, weighing 55kg.[17] However, compared to the essentially immobile Mark I machines, it was a great improvement. The two models were cryptographically identical but the keyboard, scrambler mechanism and printers all saw major redesign.[18] The Mark II incorporated two printers, which provided a plaintext and ciphertext copy of the message on a paper strip. However it was 'not capable of working direct to line', unless connected to existing Mark I teleprinter and perforator equipment.[19]

The most common other versions were the Mark III and Mark VI.[20] Each largely duplicated the functions of the Mark II in a more compact form, and could be driven electrically or by hand. They also lacked online functionality.[21]

A truly portable version of Typex which output to a lampboard also appeared, in the form of the Mark IA. This very closely replicated the form of

Figure 1: Part of the Cypher Office at Bletchley Park, with Mark II Typex machines in use. Alfred Sidney White, the head of the section, is standing, right (© Crown Copyright, reproduced with the permission of GCHQ under delegated authority from the Keeper of Public Records).

[16] AVIA 8/356, memorandum no. 11, 29/12/37; AVIA 8/356, 'A 352: Recommendation for an award to Mr. E.W. Smith in connection with Type "X" Cypher Machines and Accessories', n.d.
[17] Erskine, 'The Development of Typex', 72.
[18] AVIA 8/356, memorandum no. 54, 06/11/40; AVIA 8/356, memorandum no. 37, 25/09/39.
[19] AVIA 8/356, 'A 403: Recommendation of Awards in respect of the development of the Type X Mark II Cypher Machine', n.d.
[20] For production figures see Erskine, 'The Development of Typex', 72-3 and Erskine, 'The Admiralty and Cipher Machines', 52-3.
[21] Erskine, 'The Development of Typex', 74 and 86 fig. 5; FO 850/134, 'Maintenance of Typex Machines Marks IB, II, III and VI by Code and Cypher Personnel'.

Enigma, down to the wooden box with lid. The Mark IB added a front rail which could be operated with the non-typing hand to help drive the rotors, which was very difficult to do with the keys alone.[22] Little is known about the intended or actual role of the Mark IA/B. One IB machine was supplied to GC&CS, and 20 more were requested in October 1942,[23] but there is no indication that it was produced in numbers rivalling the printing models.

Other models of Typex represented attempts to improve the machine's suitability for large-scale communications, and in particular combine the online functionality of the Mark I with the size and relative portability of the Mark II. Several of these feature in correspondence relating to communications of the Government Code and Cypher School (GC&CS) based at Bletchley Park (BP).

3 The Use of Typex at Bletchley Park

The best-known application of Typex at BP is the use of adapted machines to 'decode' Enigma traffic. However, a more important use was in communications. Hut 6, the section which decrypted German army and air force Enigma traffic, had at most 42 Enigma-Typex machines.[24] Hut 8, which handled a smaller amount of naval Enigma traffic, probably had fewer than this. However, the Cypher Office, which dealt with all encipherment and decipherment required for GC&CS communications, had at least 60-70 Mark II Typexes.[25] By 1945 some 450 staff were processing around 500,000 groups of traffic per day using at least 23 different cipher keys;[26] it was purportedly 'the largest Typex office in the world and GC&CS experience became vastly greater than anyone else's'.[27]

The Cypher Office was a civilian section mostly employing women in their twenties. Many were 'directed' (effectively conscripted) from the local area and few were educated beyond Junior School Certificate level. There were persistent problems with high sickness rates and staff turnover (seen as indicators of poor morale) due to poor working conditions, low pay and the drudgery of the work. Most staff were unaware their work related to Signals Intelligence.[28]

Most traffic handled by the Cypher Office comprised encrypted enemy messages intercepted at overseas Y Stations (radio intercept stations). These were sent to BP via a high-speed Morse network operated by 26 (Signals) Group RAF – commanded by Lywood, who eventually reached the rank of Air Vice-Marshal. At BP messages were received in the 'Auto Room', an RAF-staffed section located adjacent to the Cypher Office in Block E (it also handled return traffic to the Y Stations).[29] Stations working to BP could be

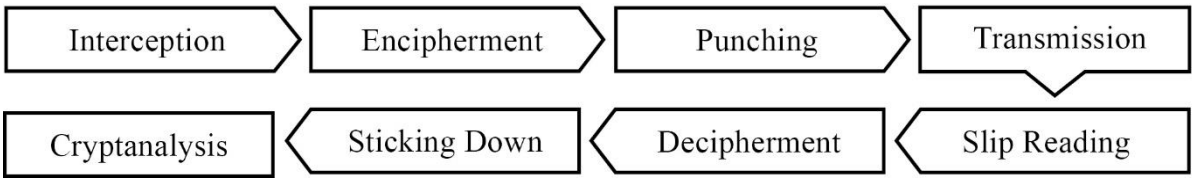


Figure 2: Processes involved in the transmission of intercepted messages to Bletchley Park by high-speed Morse.

[22] Ibid.
[23] HW 62/1, 'Typex Machines for GC&CS', 28/01/43.
[24] HW 43/72, 'History of Hut 6, vol. 3', 21.
[25] HW 14/85, 'Report on E Operations at the GC and CS by WF Friedman', 12/08/43, 106; HW 62/3, memorandum, 18/05/45.
[26] HW 14/145, GC&CS weekly admin returns; HW 62/3, 'Cipher Office: Total Incoming Groups Handled in Each Setting', n.d. but 03/45.
[27] HW 50/50, 'Memorandum by Mr de Grey', 29-30. Earlier in the war, the limited capacity of the Cypher Office was a serious bottleneck in the intelligence-production process: in January 1942 a backlog of over one million groups of intercepted traffic was destroyed as there were not sufficient staff or machines to de-Typex it. See HW 43/2, 'History of British Sigint, 1914-1945; volume II', 528.
[28] HW 64/67, 'Staff Matters'; HW 64/62, 'Welfare'.
[29] RAF Signals Museum, 'The Signals War: A Brief History of No. 26 Group RAF', 15-16; HW 50/50, 'Memorandum by Mr de Grey', 27-8; HW 64/63, 'RAF Personnel at Bletchley Park'. BP was identified to subscribers as 'Church Green'. This was the name of the camp which housed the headquarters of 372 Wireless Unit, which had operational control of the Auto Room and administered most RAF staff at BP, who were accommodated at the camp. It also ran the receiving station at Stoke Hammond (near Bletchley) and transmitting station at Greatworth (Northamptonshire) linked to the Auto Room.

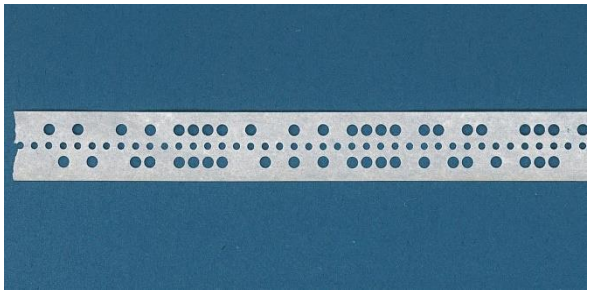


Figure 3: Morse tape (Bletchley Park Trust (BPT)).

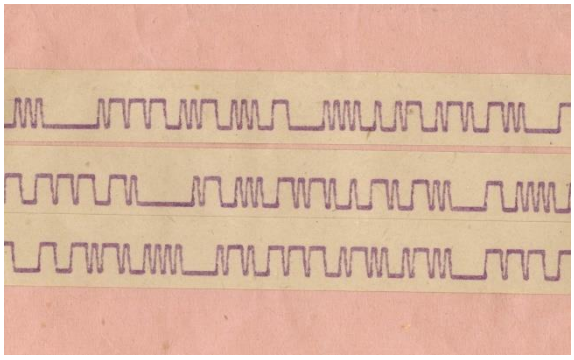


Figure 4: Morse slip (BPT).

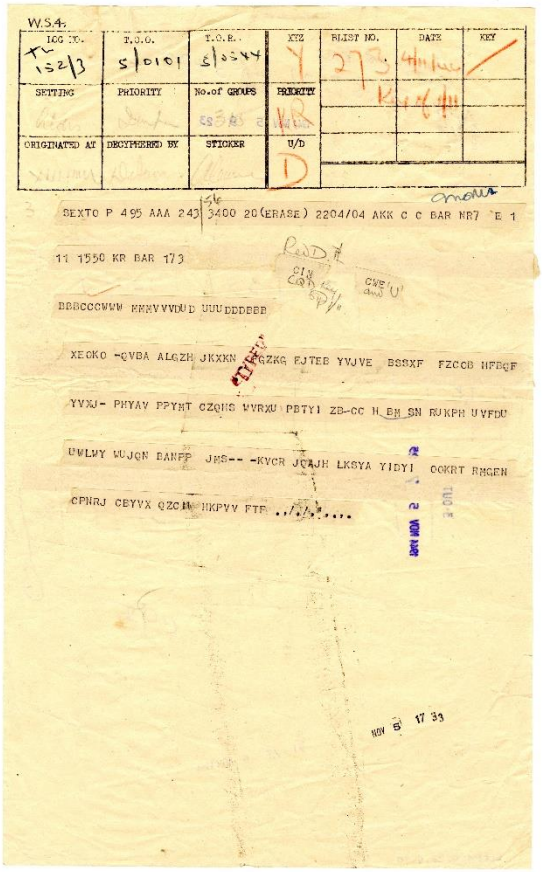


Figure 5: De-Typexed intercept as passed on from the Cypher Office for cryptanalysis (BPT BLEPK:0020.5.128; © Crown Copyright, reproduced with the permission of GCHQ under delegated authority from the Keeper of Public Records).

fixed wireless stations, or mobile stations such as RAF 'Heavy Mobile Units' (HMUs), Army Type M or R Special Wireless Sections and 'Golden Arrow' wireless units.[30] Intercepts were enciphered on Typex (i.e. overlaying the enemy encipherment with an additional layer of British encipherment) before transmission. In March 1943 out of a total requirement of 861 machines throughout the services, 105 were needed by the Y Services for this purpose.[31] On arrival messages had to be deciphered (or 'de-Typexed') in the Cypher Office, restoring the original, singly-enciphered message, before they could be passed to the cryptanalysts. An additional function of the Cypher Office was to encrypt Ultra intelligence before its transmission to Allied commanders in the field.[32]

The process of sending by high-speed Morse was fairly cumbersome (see Figure 2). At the sending end, after encipherment, messages were punched onto Morse (Wheatstone) tape using a perforator (Fig. 3). When the tape was fed through a reader ('autohead') the message was transmitted. At the receiving end, the message was reproduced as a line on tape by an 'undulator'; this was known as 'Morse slip' (Fig. 4). Personnel skilled in 'slip reading' were required to type up the message into readable (albeit still enciphered) text ready for de-Typexing. De-Typexing also involved multiple steps, as the 'plaintext' (singly-enciphered) output tape had to be 'stuck down' onto sheets for practical handling by the cryptanalysts and checked for errors (Fig. 5). Though the whole system could handle vastly more traffic than

[30] HW 43/72, 'History of Hut 6, vol. 2', 49-50; 'The Signals War, 15-16; WO 219/5319, 'Notes on Wireless Interception ("Y") Organisation in the Field', 21/11/43; 'The Golden Arrow Group 1944 to 1945', https://www.goldbeach.org.uk/yservice/Golden%20Arrow.htm [accessed January 2025]; Bletchley Park Trust, material relating to Alan Foot.

[31] HW 62/1, Saunders to Travis, 06/03/43.

[32] Most Ultra was disseminated by cable or via a special MI6 radio station at Windy Ridge near Bletchley, but some went via the Auto Room.

manual encipherment and transmission, a considerable number of personnel were required.

GC&CS therefore explored ways to increase the efficiency of its communications. Some solutions related to organisation and training. The workload and output of individual staff were carefully monitored to identify problems, while maintenance schedules were optimised to minimise the amount of time machines were out of order, and Typex rotors were kept ready made-up to reduce time spent changing settings.[33] Cypher Office staff were given five weeks' training in slip reading so they could de-Typex traffic direct from Morse slip, eliminating the typing-up stage which was normally done in the Auto Room.[34] Solutions were also sought in new machinery.

4 Improved Typex Models

Because of its printed output the Typex was already a much more suitable machine for large-scale communications than Enigma, but some improved models were tested at or designed in consultation with GC&CS. That none of these saw widespread use had to do with organisational factors and the fact that their advanced features did not necessarily lead to improvements in relation to the most important concern: maximising the traffic that could be carried on communications links.

4.1 Typex Mark IV

One attempt to combine the features of the Mark I and Mark II was the Mark IV. This probably involved incorporating the Typex scrambler into the Creed Model 7 teleprinter to create an online teleprinter cipher machine.[35] Development was abandoned for unknown reasons, though it is reasonable to speculate this had to do with technical difficulties, as this was the fate of several other attempts to combine cipher machines with existing typewriters or teleprinters.[36]

4.2 Typex Mark V

The Mark V, known as the 'Teletypex', was a more successful execution of the same concept. It was tested by the War Office on behalf of GC&CS in early 1943, and described as 'quite practicable' on cable links. However, landline links were usually treated as secure so there was little demand for a machine for this purpose. An alternative application was on radio-teleprinter (radio-teletype) links, but here 'the technical irregularities of the average circuit tried was to render the teletypex [sic] Mark V quite ineffective as a method of handling traffic'. There were several serious issues. As soon as a letter was missed due to interference the rotors of the machines at each end of the link got out of step and the rest of the message was unintelligible. The fact the machine did not transmit letters in groups and did not create a record of the message as transmitted – since 'to do so would have caused considerable further increase in the complexity, size, and weight of the apparatus' – made it difficult to correct mistakes.

The Mark V also contravened one of the essential principles of efficient telegraphic communication: keeping the link operating as close to maximum capacity as possible. In a conventional arrangement with separate teleprinter and cipher machines, a message that did not successfully decipher could be worked on separately while the link was kept busy with other traffic. With a combined machine, the valuable and expensive link to which it was connected had to sit idle while any corruptions were resolved; this could seriously reduce its capacity.[37]

In any case, Britain (unlike Germany, the USA and the Soviet Union) did not make significant use of radio-teleprinter, preferring high-speed Morse. Had the demand existed, the Mark V might have been improved to create a machine equivalent to the German SZ40/42 ('Tunny') and T-52 ('Sturgeon'), but in the event it does not appear to have been developed past the prototype stage.

[33] HW 50/50, 'Memorandum by Mr de Grey', 29.
[34] Ibid., 12; HW 14/67, Travis end of year report for 1942, 9.
[35] David Abrutat, 'The Story of TypeX', https://commsmuseum.co.uk/Typex/typex.htm [accessed January 2025].
[36] For example, the early printing models of Enigma and the O'Brien Cipher Typewriter.
[37] HW 62/1, 'Notes on the Teletypex Mark V', n.d. but March 1943; HW 62/1, Lycett to Travis, 23/03/43.

4.3 Typex Mark VII

It was recognised that a machine incorporating a Morse tape perforator and/or tape reader for processing high-speed Morse traffic would be far better suited to requirements. This led to several different projects, which are difficult to disentangle. Two different perforator models were under development by the RAF and Army by early 1943. Development of the RAF machine was begun partly at the instigation of Commander Malcolm Saunders, in charge of GC&CS communications between April 1942 and April 1943.[38] In early 1943 attachments were trialled which permitted both enciphering to Morse tape (using a perforator) or deciphering from tape (using a tape reader).[39] These features were ultimately incorporated into two different models.

The perforator model, designated the Mark VII, was tested by GC&CS in February 1944, but adoption was not recommended. The reasons were essentially organisational. At BP, cipher machines and radio links were operated by different sections (the Cypher Office and Auto Room). If Typex machines were placed in the Auto Room, this would leave two sections dealing with encipherment. There would be inevitable competition for mechanics and spare parts; it was impossible to employ experienced civilian operators in the Auto Room, and the RAF would struggle to find and train sufficient operators from scratch; and with responsibility divided it would be difficult to ensure prompt and reliable processing of traffic.

Alternatively, Mark VIIs could be placed with the existing Typex machines in the Cypher Office. Even though the Mark VII allowed rapid switching between punching plaintext and ciphertext, so that unenciphered message preambles could be typed before an enciphered message, this was incompatible with arrangements at BP: when sending messages, preambles would have to be prepared in the Auto Room, whose staff had the necessary knowledge of signalling arrangements, and joined to the correct ciphertexts produced in the Cypher Office. This was thought too likely to lead to errors.

Other options were to send the preambles at hand speed, which would significantly reduce the capacity of each link; or to prepare messages complete with preambles before they were passed on for encipherment and transmission, which would entail a major reorganisation of communications for which GC&CS had neither the staff, space nor appetite, especially since only a small number of Mark VII machines would initially be available. After consideration by the experts, none of these options was considered practical. It is possible the Mark VII saw service in Army or RAF signals centres where arrangements were simpler.[40]

4.4 Typex Mark IX

The tape reader model appeared as the Mark IX, 'a modified TYPEX machine operating in conjunction with a Morse Printer Head' to decrypt automatically from Morse tape. It was first mentioned in mid-1943 and was tested by GC&CS later in the year. The machine was mechanically reliable, but produced various procedural complications. For example, if the sending cipher operator made a mistake while using the Mark II, they would simply add an 'erasure signal' instructing the receiving operator to disregard the previous group. The Mark IX, however, required tapes without errors or corruptions. Erasures could not be cut out at the receiving station, because when tapes were rejoined this doubled their thickness and they would no longer fit through the Mark IX tape reader. This meant erasures had to be removed at the transmitting station. However, interference on the link could, naturally, reintroduce errors; during testing no more than 40 per cent of messages were received at BP with no errors (recording received messages on 'reperforated' tape was less reliable than using undulator slip). This meant 'any superiority of the Automatic Decoder is practically nullified by virtue of the difficulty experienced in producing a 100% [accurate] tape.'[41]

There were other problems. For collecting the statistics which were necessary to monitor efficiency, it was necessary to time-stamp messages; this was easy when dealing with typed

[38] HW 62/1, Saunders to Travis, 04/02/43.
[39] HW 62/1, 'Report by Mr Heil', 26/01/43; HW 62/1, Lywood to Travis, 17/02/43.
[40] HW 62/1, 'Report by Mr Heil', 26/01/43; HW 62/2, 'Typex Mark VII Machines', 20/02/44; HW 62/2, 'Typex Mark VII Machine', n.d. but 02/44; HW 62/2, 'Typex Mark VII Machines', 22/02/44.
[41] HW 62/1, 'Typex Mark IX Automatic Decoder', 10/12/43.

sheets, but harder when messages arrived as a thin and fragile paper tape. A specialist team decrypted message indicators (using indicator books) before messages were de-Typexed; again, it was unclear how to record the true indicator on the tape. The machine settings still had to be changed manually, so the increased speed of decryption could only be fully exploited where there was a need to process large volumes of traffic on the same cipher key, which was not usually the case at BP.

All these issues were rendered moot, however, by the fact that even under ideal conditions, the Mark IX proved to be no faster or more labour-efficient than the Mark II. A Mark IX, provided with a continuous supply of perfect tapes, processed 23,850 groups in 24 hours. Three experienced operators, each working an 8-hour shift on the Mark II, could process 24,000 groups, irrespective of any corruptions in the messages. Adoption was therefore not recommended.[42]

4.5 Typex Mark VIII

At approximately the same time, the Mark VIII was under development. Internet sources describe it as a Mark II incorporating a perforator but also capable of online use. Some photographs of uncertain provenance are also available.[43] The origins of this model are unclear. The Army had been working on its own high-speed Morse machine (also in collaboration with Creed). According to the sole documentary reference this machine, which had gone into production and was being considered for use by the RAF, 'differs from that produced by [the RAF at] Kidbrooke in that the whole of the operation, either in clear or in cypher is carried out from the Typex keyboard.'[44] This presumably meant it was an online machine, eliminating the need to feed the produced tape through a transmitter. Alternatively the Mark VIII may have been developed from the RAF perforator/tape reader attachments trialled in early 1943.[45]

Unfortunately, nothing else is known about the functions of the Mark VIII. If as described, it had obvious applications on high-speed Morse links. It appears to have been brought into service, as a manual was issued by the RAF.[46] However, the only other information about its use is a remark by Erskine that only 398 were ordered as late as 10 January 1945, which strongly implies it was not brought into widespread service before the end of the war.[47] The Mark VIII should be a subject of further research.

4.6 Unnamed Australian Typex

An online Morse machine was also under development in Australia in early 1943. The machine would encipher direct to line at typing speed. Decryption at the receiving end would be done manually, so there was no problem of synchronisation. However, any errors in encipherment would need to be corrected over the line – rather than being resolved before the message was transmitted – and this seemed to pose security risks, especially if it was necessary to re-encipher the same message on a different setting. GC&CS cryptanalyst Alan Turing was consulted on security aspects, but came to the surprising – if characteristically brusque – conclusion that 'This seems to be a communication problem rather than security, and therefore hardly my concern at all'. It is unclear whether the project came to fruition.[48]

4.7 Typex Mark IIA

By late 1944 an upgrade of the Mark II, designated the Mark IIA, was under development. The right-hand tape printer was replaced by a full page printer which could produce multiple copies of an enciphered message with automatic formatting. This was 'designed to fill the requirements of certain large and busy traffic centres handling signals which have to be dispatched over two or more different communication channels simultaneously' while 'obviating the delay and work of sticking up the tape and of typing out copies'. It was also possible to fit a standard Morse perforator to produce a perforated tape at the same time as the printed

[42] HW 62/1, 'Note of Action', 24/06/43; HW 62/1, 'Type "X" Mark IX Service Trials', 08/12/43; HW 62/1, 'Typex Mark IX Automatic Decoder', 10/12/43.
[43] 'Typex: History/Development', https://www.jproc.ca/crypto/typex.html; 'Typex Mark VIII', https://cryptomuseum.com/crypto/uk/typex/index.htm#mk8; 'Typex', https://en.wikipedia.org/wiki/Typex [accessed January 2025].
[44] HW 62/1, 'Note of Action', 24/06/43.
[45] HW 62/1, 'Report by Mr Heil', 26/01/43; HW 62/1, Lywood to Travis, 17/02/43.
[46] AIR 20/1531, 'Security of RAF Signal Communications 1939-1944', 36.
[47] Erskine, 'The Development of Typex', 86 fig. 5.
[48] HW 62/1, Johnston to Turing, 20/01/43.

message sheets. It is unclear whether the machine went into production and there is no evidence it saw use on intelligence networks.[49]

4.8 Typex Mark X

The most advanced model of Typex, the Mark X, later named Mercury, was under early development by 1944-5. It appears to have been envisioned as a 'universal machine' for post-war general service traffic as well as use on radio teleprinter links.[50] The machine used a set of four control rotors to determine the movement of six ciphering rotors, similar to the principle of the American SIGABA (ECM Mark II). Prototypes were ready by 1948 and the machine saw service between approximately 1950 and 1963.[51]

5 Conclusion

The GC&CS records reveal four Typex models – Marks V, VII and IX, and an unnamed Australian prototype – which have not previously been described. Although some details remain uncertain, a largely complete list of Typex models can be presented for the first time (see Table 1).

Despite the amount of money and engineering ingenuity which must have gone into the development of Typex machines operating online or with perforated tape, none were adopted by GC&CS. Though less advanced, and necessitating more manual processing of traffic by staff, the earlier and simpler Mark II proved a more practical fit for the organisation at BP. The failure of more advanced models highlights how crucial it was that cipher machines, as well as being secure, were fully compatible with the communications system with which they were to be used. Theoretical improvements in efficiency and savings in time and (wo)manpower could evaporate when organisational arrangements and long-standing practices and procedures were taken into account.

It is notable that contemporary comments on the Typex make much more of the great increase in the efficiency of communications the machine permitted, due to the speed and accuracy it offered compared to manual systems, than its security. It was for those reasons, rather than its cryptographic strength, that the RAF Director of Signals predicted that 'general application of the Type X Mark II machine throughout the Services and Dominions will revolutionise secret telegraphic communications'.[52]

[49] FO 850/134, 'Typex Machines Mark IIA', 03/11/1944.
[50] HW 62/3, Lywood to Travis, 30/01/45.
[51] 'Mercury (Cipher Machine)', https://en.wikipedia.org/wiki/Mercury_(cipher_machine) [accessed January 2025].
[52] AVIA 8/356, memorandum no. 69, 12/09/41.

Mark	Year	Features	Adopted?
I	1937	Immobile online printing machine	Yes
IA/B	By 1943[53]	Portable glow-lamp machine	Yes
II	1938	Portable offline printing machine. Upgraded versions designated Mark 22 and Mark 23	Yes
IIA	1944[54]	Portable offline machine with full page printer	Unknown
III	By 1941[55]	Compact hand-operated offline machine	Yes
IV	Unknown	Adaptation of Creed Model 7 Teleprinter	No
V	1943	Online teleprinter machine	No
VI	By 1943[56]	Compact hand-operated offline machine	Yes
n/a	1943?	Unnamed Australian online machine	Unknown
VII	1944	Offline machine with perforator for enciphering to tape	Unknown
VIII	1943?	Online Morse perforator/tape reader machine	Yes
IX	1943	Offline machine with tape reader for deciphering from tape	No
X	1948	Online teleprinter machine	Yes

Table 1: Summary list of Typex models

[53] FO 850/134, 'Maintenance of Typex Machines Marks IB, II, III and VI by Code and Cypher Personnel' (Note the Mark IA/B is described alongside Mark VI).
[54] FO 850/134, 'Typex Machines Mark IIA', 03/11/1944.
[55] Erskine, 'The Admiralty and Cipher Machines', 52. The Mark III was under design by January 1940: see AVIA 8/355, 'Invention Relating to Drum Locking Device', 23/01/40.
[56] AIR 10/4051, 'Type X Machine Mark VI', 01/43; Erskine, 'The Development of Typex', 74.

Acknowledgements

The author would like to thank David Kenyon, Craig Heath, David Abrutat and the reviewers for their helpful comments and suggestions, and Annie Christie for assistance with graphics.

References

David Abrutat, 'The Story of TypeX', https://commsmuseum.co.uk/Typex/typex.htm.

Anon. 'The Signals War: A Brief History of No. 26 Group RAF', RAF Signals Museum, Henlow, Bedfordshire, UK.

Bletchley Park Trust archives (BPT), Bletchley, Buckinghamshire, UK.

AG Denniston, 'The Government Code and Cypher School Between the Wars', *Intelligence and National Security*, 1/1 (1986), 48-70.

Ralph Erskine, 'The Development of Typex', *The Enigma Bulletin*, 2 (1997), 69-86.

Ralph Erskine, 'The Admiralty and Cipher Machines During the Second World War: Not So Stupid After All', *Journal of Intelligence History*, 2/2 (2002), 49-68.

John Ferris, *Behind the Enigma: The Authorised History of GCHQ* (London: Bloomsbury, 2020).

David Kenyon and Frode Weirud, 'Enigma G: The Counter Enigma', *Cryptologia*, 44/5 (2020).

Louis Kruh and CA Deavours, 'The Typex Cryptograph', *Cryptologia*, 7/2 (1983).

NK O'Neill and KJ Hughes, *History of CBNRC*, vol. V (August, 1987), https://luxexumbra.blogspot.com/2019/08/history-of-cbnrc.html.

NSA, 'William F Friedman Collection of Official Papers', https://www.nsa.gov/Helpful-Links/NSA-FOIA/Declassification-Transparency-Initiatives/Historical-Releases/Friedman-Documents/.

Jerry Proc, *Crypto Machines*, https://www.jproc.ca/crypto/index.html.

Paul Reuvers and Marc Simons, *Cryptomuseum*, https://cryptomuseum.com/.

The Y Service, https://www.goldbeach.org.uk/yservice/.

UK National Archives, Kew, Richmond, UK (TNA): files ADM 1/27186, AIR 10/4051, AIR 2/2720, AIR 20/1531, AVIA 8/355, AVIA 8/356, HW 14/145, HW 14/23, HW 14/85, HW 25/6, HW 25/14, HW 43/2, HW 43/72, HW 50/50, HW 62/1, HW 62/2, HW 62/3, HW 64/62, HW 64/63, HW 64/67, WO 219/5319.

Frode Weierud, *Crypto Cellar Research*, https://www.cryptocellar.org/.

Wikipedia, https://www.wikipedia.org/.

Cryptanalytic and historical challenges with unidentified encrypted documents from the early modern era

Camille Desenclos
Université de Picardie Jules Verne,
CHSSC / Université de Lorraine, CNRS,
Inria, LORIA
camille.desenclos@u-picardie.fr

George Lasry
The DECRYPT Project
george.lasry@gmail.com

Abstract

In most cases, historical encrypted documents include some parts in cleartext, such as headers, dates, signatures, or addresses, which allows the origin, date, and language of these documents to be established. An attached decrypted text, similar documents (same encryption, homogeneity of date or origin) in the same volume or box, or the catalog description may assist in that process. However, in a few cases, none of these are available, posing several challenges both from cryptanalytic and historical perspectives. Based on three 16th-century case studies, this paper aims to discuss a multidisciplinary method to proceed from an unidentified encrypted document to a workable transcription – decipherment and identification.

1 Introduction

Performing both a complete decipherment and an accurate historical analysis of a text requires basic identification information: who is writing? To whom? When? Such information can be missing in encrypted documents as well as in clear ones. With clear texts, the content can help determine the origin of the documents. It is the other way with encrypted texts: the origin is crucial to reconstruct the content (especially the names that are encrypted with a nomenclature).

Many encrypted documents from the early modern era are letters: the first words, courtesy formulas, dates, and/or signatures are thus cleartext, as is the last page, which typically contains the address and sometimes the sender's name, the sending and/or reception dates. Each of these cleartext segments offers the required identification information. In a few other cases, the encrypted document does not provide any identification information, but it is preserved alongside other encrypted documents (already decrypted and/or identified), allowing us to link all the documents together because of their similar ciphers. In both cases, not only are the sender, the recipient, and the language identified by this means, but also the date and the nature of the information (diplomatic, military, …) can be predicted. Those are valuable for cryptanalysis as they indicate the vocabulary and names we can reasonably expect.

Things become more challenging when the encrypted document does not provide any such information and cannot be matched with other encrypted documents. As some logic was probably involved in building up the volume or the box (same sender or recipient, same date range, same territorial area, same holder), it is tempting to assume that the documents (encrypted or not) are related, especially when they seem to be homogeneous (same topic, same date, …). However, it is possible that letters belonging to the same correspondence were dispersed and/or inserted into the wrong volume(s) or box(es). While dealing with unidentified encrypted letters, the global content of a volume or a box can serve as a hint, but not as a foundation for cryptanalysis.

These challenges, both for cryptanalysis and historical analysis, are common to every encrypted document, regardless of its origin, date, or content. This paper proposes a method, or at least some potential solutions, to overcome them, from three 16th-century case studies. As centralized archival practices were yet underdeveloped (especially in 16th-century France), Renaissance documents can present additional and specific challenges: they are often scattered between boxes or even archives, and the losses (missing letters within a correspondence, missing decrypted texts) are numerous. In addition, the content of the documents may be less intelligible, especially while cryptanalyzing: non-standardized spelling, titles rather than names for people's designation, …

Proceedings of the 8th International Conference on Historical Cryptology, pages 41-51
Poznań, Poland, 16-18 June, 2025

2 Presentation of the case studies

The three encrypted text sets have been chosen according to their representativeness for Renaissance encrypted documents: scattered letters, scattered pages, and an isolated letter. All three of them are preserved at the Bibliothèque nationale de France (BnF) and were produced during the 16th century. Each case also represents a different stage of the cryptanalysis process: cryptanalysis and identification are completed, cryptanalysis is completed but identification is incomplete, and ongoing cryptanalysis and identification.

2.1 Mary Stuart's scattered letters (1578-1584)

Several papers have recently been published on Mary Stuart's encrypted letters, which were found at the BnF (Lasry/Biermann/Tomokiyo 2023; Biermann/Tomokiyo/Lasry 2023). As the cryptanalytic work has been completed and published, we do not intend, in this paper, to uncover new elements but to examine the methodology that has led to the identification of the letters and see how much it can be replicated.

Before cryptanalysis, none of these letters had been identified except for one. It turned out, after cryptanalysis, that they had been written by Mary Stuart (Queen of Scots) between 1578 and 1584 and were addressed to the French ambassador to Elizabeth I (Queen of England), Michel de Castelnau, seigneur de La Mauvissière. In total, 57 letters are scattered within four volumes at the BnF: Français (fr.) 2988 (26 letters), fr. 3158 (1 letter), fr. 20506 (28 letters), Cinq-Cents de Colbert 470 (2 letters). Only the letter in volume fr. 3158, which has some parts in cipher and the rest in cleartext, had been identified as from Mary Stuart before the work of Lasry *et al.*: Castelnau's name and the date were written in cleartext, and an additional mention on the top margin identified Mary Stuart as the sender. For volume Cinq-Cents de Colbert 470, the connection with Castelnau and Mary Stuart could have been guessed: every document in the volume is related to him and mainly focuses on Mary Stuart; furthermore, numerous letters were written by her. For the letters within the two other volumes (fr. 2988, fr. 20506), as each letter is fully encrypted (even the address), nothing could help connect them with Mary Stuart or Castelnau.[1] Lastly, according to the catalog (the description has been updated since then), the encrypted letters in volume fr. 2988 (on which Lasry *et al.* started working) contained letters unrelated to Mary Stuart or England/Scotland, but rather related to Italy and written at the end of the 1520s.[2]

2.2 Duprat's encrypted reports (1521)

As for Mary Stuart's letters, the second case consists of scattered letters but in three closer volumes. They are preserved at the BnF: fr. 2961 (24 pages, three text sets), fr. 3029 (42 pages, eight text sets), fr. 3092 (six pages, three text sets). These texts are currently being analyzed by the authors (publication forthcoming), but they can already be identified as reports from the French embassy (composed by Antoine Duprat, Jacques de Chabannes, Jean de Selve, and Robert Gedoin) at the French-English conference in Calais in 1521.[3] With one exception,[4] these texts are fully encrypted and are deprived of any address (See Appendix 1). In one encrypted document (fr. 3029, fol. 134), a mention has been added on the top of the first folio: "Lettre en chiffre adressée au roy" (Letter in cipher addressed to the King). However, while this suggests the encrypted text may also be in French, this does not help much, as we do not know which French king this is about. Regardless of this mention, the texts may be encrypted reports attached to letters. However, these reports are not preserved in a way that reflects their initial form. Following cryptanalysis, we have been able to locate historical decryptions (from the early 16th

[1] There are a few remaining seals that display the royal arms of Scotland, but their analysis would have required heraldic knowledge (which is not often mastered by codebreakers or even historians).

[2] Three reports, written in Italian, were mentioning Mary Stuart in volume fr. 2988. As the majority of documents and especially the other encrypted letters were related to Italy, the odds were in favor of 1520s Italy and not of 1570s Scotland.

[3] See about the Calais conference, Russell 1971. As Russell points out, the conference has been well documented by both sides (French/English). The journal of Jean Barillon, Duprat's secretary, in which some of the ambassador's letters were copied, is now preserved at the BnF (NAF 6777) and was published in the 19th century (Barillon 1897-1898).

[4] Fol. 105 (fr. 3029) contains cleartext segments. The Latin text does not offer any precise element for identification (no date, no name, no event). It is the only Latin text; the reports had all been written in French.

century) of some of the reports, allowing us to determine that these 14 texts do not form 14 reports. These plaintext copies helped to distinguish nine reports, one of which was scattered across two volumes, and two of which were incomplete (the missing parts of these two encrypted reports have not yet been found). As for the rest, no plaintext copies matching our decipherments have been found. It has nevertheless been possible to distinguish four other reports, presumably complete, while two other sets have yet to be investigated but presumably form at least three separate reports. Not only are the complete set of Duprat's encrypted reports scattered between three volumes, but their pages are filed in different volumes, which prevents a full understanding of the content when deciphering them. Finally, although the content of the three volumes suggested that the encrypted documents could be diplomatic letters written in the 1510s or the 1520s, no other documents in the volumes were related to the Calais conference. Before the advent of cryptanalysis, nothing could have led to identifying those reports as having been produced by the French embassy in Calais in 1521.

2.3 Fr. 6632 isolated document (1580s)

Although they are scattered in various volumes and sometimes fragmented, Mary Stuart's letters and Duprat's reports each offer numerous encrypted pages, which helps a lot for cryptanalysis. The third and last case study addresses the opposite challenge: a unique specimen. Volume fr. 6632 (BnF) contains a two-page text (possibly a letter) that does not present any identification element and is the only encrypted text of the volume (See Appendix 2). As in the previous examples, the other letters and the catalog description may offer some guidance. Volumes fr. 6628 to 6632 contain letters, mostly sent to King Henri III of France by Queen Mother Catherine de Medici and others. However, the many letter authors in these volumes do not help to formulate any hypotheses, except for the date (probably in the 1580s). At the current stage of the working process, this encrypted text has not been precisely identified.

3 Deciphering unidentified encrypted letters: challenges and methods

There are three different approaches to accessing the content of early modern encrypted documents (which have not been decrypted yet), whether or not they have already been identified:

- Finding the original cipher key and associating it with the encrypted documents
- Recovering the key based on available decrypted text
- Reconstructing the key via cryptanalytic techniques.

3.1 Finding the original cipher key in the archives

The first two approaches can be implemented without cryptanalysis expertise, but they require archival knowledge. Keys and encrypted letters are not systematically preserved in the same volume/box (or even in the same library/archives). If the archival situation varies from country to country, from archives to archives, researchers may face four situations:

1. Encrypted letters and cipher key are preserved in the same volume/box or group of volumes/boxes[5]
2. The cipher key is preserved along with other keys[6]
3. The cipher key is preserved alone in an unrelated volume or box[7]
4. The cipher key has been lost or destroyed.

Scenario 1 is the easiest one. As soon as a cipher key is preserved in a volume or box (or a homogeneous set of volumes or boxes), a comparison with the encrypted letter(s) can be performed. This is done first by comparing the ciphertext characters and secondly by attempting to decrypt the letter using the cipher key. We can know whether the cipher key matches the encrypted letters in a few minutes. In contrast, we cannot state the loss of a cipher key (Situation 4) before conducting extensive archival work.

[5] See for instance, BnF, fr. 4030. The volume contains both the original cipher key and the encrypted letters from Villeroy to Hotman (1609-1612).

[6] Volume fr. 3995 (BnF) contains 68 cipher keys (1580-1595) that were, mainly, used by the Duke of Nevers.

[7] See for instance, BnF, fr. 3349. The volume contains an unidentified cipher key (probably from the 1580s), along with letters from Henri III and Catherine de Medici but none of them is encrypted.

As for Scenarios 2 and 3, Renaissance cipher keys were not systematically labeled by their users. The name of the users (mainly the recipient's name) and/or the date may have been written on the verso.[8] However, many cipher keys remain unidentified or, at the very least, are not sufficiently identified (i.e., lacking a date or name). As a result, it may be impossible to determine the origin of cipher keys without matching them to encrypted letters. This can be a vicious circle: encrypted letters are crucial to identify cipher keys, but cipher keys are crucial to decrypt letters. Moreover, the catalogs often do not list all the documents in the volume/box, regardless of whether they can be identified. Although the precision of catalog descriptions varies from archives to archives and even from collection to collection, we often have to deal with descriptions such as "several cipher keys",[9] with incorrect descriptions,[10] or with descriptions that do not mention the cryptographic nature of the documents.[11]

Linking a cipher key to its encrypted letters would require reviewing all the collections of several libraries and archives to ensure that we have browsed every box and volume that may contain a cipher key, regardless of the description provided by the catalogs.[12] Secondly, as the encrypted letters are not identified, it is impossible to exclude any cipher key based on criteria such as the date, the user, the territorial area, or the language: we would need to compare each cipher key to the encrypted letters manually. It is both time-consuming and ineffective. In contrast, cryptanalysis can help find the original cipher key. Matching letters and original cipher keys is much easier when the key, or part of it, has been reconstructed: it no longer requires trying to decrypt the first lines of the encrypted document with each key whose ciphertext characters seem to match and, because of the newly deciphered content, we can focus on cipher keys (and other encrypted texts) relevant to a specific date and use. A copy of the cipher key that was used by the French ambassadors in Calais, for instance, still exists: Jean Barillon, Duprat's secretary, has copied the key in his "journal" (BnF, NAF 6777), which is a record of the conference (copies of the diplomatic instructions, of the key, of some letters). Because of its nature, this journal is not preserved alongside or near the encrypted reports (which are not preserved with the other letters from the ambassadors). Matching the encrypted reports with the copy of the cipher key was possible only after cryptanalysis, as this copy (not listed in the library catalog) had already been identified and mentioned in Desenclos 2021 as a copy of the key used by the French ambassadors in Calais.[13] Similarly, Kopal and Waldispühl, who worked on a previously unidentified letter from Emperor Maximilian II (1575), were able to find the original cipher key (and two other encrypted letters) in the DECODE database, only after having cryptanalyzed the first letter (Kopal/Waldispühl 2022).

3.2 Recovering the cipher key from available decrypted text

The same issue arises with the second approach (recovering the key based on available decrypted text) as several configurations can be observed, depending on whether the decrypted text has been copied on the original document (margin, between lines) or another folio:

- Encrypted letters are preserved in several but related volumes (groups of manuscripts or boxes dedicated to the same negotiation, the same person, or the same time range), and the decrypted text has been copied on the original letters or on separate folios that can be linked to the

[8] See for example, BnF, fr. 6204, fol. 8: "chiffre avec M. d'Usson, 14 septembre 1701" (cipher with Mr d'Usson, 14 September 1701).

[9] The BnF catalog suggests that volume NAF 8431 contains "diplomatic ciphers". Without inspecting the volume, we cannot know how many cipher keys and whose key the volume really contains.

[10] According to the catalog, volume fr. 3362 (fol. 57) would contain "a cipher". However, it is not a cipher key but part of an encrypted document.

[11] Volume fr. 7129 contains 3 cipher keys; none is mentioned in the catalog.

[12] For French early modern encrypted documents, the keys can be found mainly at the BnF, the Archives nationales (national archives), the Archives diplomatiques (diplomatic archives) or the Service historique de la Défense (military archives), but one can also find isolated keys in local or private archives or in another library.

[13] Another cipher key has been found in volume fr. 2961 (BnF). It does not contain the nomenclature, but only the alphabet symbols. This key could have been easily matched to the encrypted reports: it is held before one encrypted report. However, encrypted letters in volume fr. 2961 have been found only after the first stages of cryptanalysis have been performed on the reports in volumes fr. 3029 and 3092.

encrypted letters (mentions on top or on the back folio, decrypted text which immediately follows the encrypted letter, …).
- Encrypted letters are scattered between volumes of different origins, and the decrypted text has been copied on the original letters
- The decrypted text has been copied onto another folio and preserved in a different volume from the letters, or lost or destroyed over the centuries.

Decrypted texts were preserved, or at least have been identified, only for Duprat's reports (BnF, fr. 2966-2967). However, the connection with the encrypted reports could not have been made as there were no elements available to match a report to its relevant decrypted text: the decrypted texts are contained in another volume, and even if the decrypted texts are partially identified (sender and, sometimes, recipient), the encrypted reports were not. Only recovering the content via cryptanalysis has made this connection possible.

Although it is one of the hardest configurations (unidentified encrypted letters and a separate decrypted text), Duprat's case enhances the two main requirements for recovering a cipher key (without cryptanalysis): a decrypted text that exists and that can be linked to the unidentified encrypted document(s). Matching encrypted and decrypted texts together (as matching encrypted texts and cipher keys), whether to find relevant decrypted texts or enlarge the set of texts encrypted with the same cipher, can only be based on spot checks or an exhaustive inventory of all encrypted texts. The first method cannot be performed when the texts are not identified, and the effort needed for the second method might be prohibitive: matching documents requires both finding other similarly-looking encrypted documents and reconstructing each cipher key manually from these documents.

A third approach could be to look at close volumes or boxes (same territorial area, same period, same writers), but it can be hard to perform without any slight identification element. To overcome this, we can try to determine the origin of the documents and then investigate nearby volumes from a similar origin, rather than similar themes. All three volumes containing Duprat's reports belong to the former collection of Philippe de Béthune, part of which was organized by reigns (Solente 2001). As the content of volume fr. 3029 belongs to the sub-collection related to Francis I's reign, we could simply look into Béthune's volumes that belong to the same sub-collection. However, in the sub-collection related to Francis I's reign, there are more than 170 volumes. Other encrypted reports have certainly been found in volumes that belong to this sub-collection (fr. 2961 and fr. 3092). However, we cannot be absolutely sure that Duprat's reports had not already been scattered before the constitution of Béthune's collection and preserved in distinct collections or even libraries. In fact, one of the decrypted texts does not match any encrypted reports and suggests that at least one report is missing. Even by investigating the volumes only based on their origin, the archival investigation work remains highly time-consuming and represents a significant investment for a few unidentified texts.

In Mary Stuart's case, the pre-existing identification of the letter in volume fr. 3158, the credible identification of Castelnau as the recipient or intermediary of the letters in volume Cinq-Cents de Colbert 470, and the similarities between all the encrypted letters (e.g., "K" at the beginning and in the addresses) could, in theory, have led to identifying all these letters as Mary Stuart's. That would have required prior knowledge of the existence of these four text sets with similar graphical ciphertext characters. Yet, they are scattered in very distinct volumes: two of them were not supposed to contain Mary Stuart's letters, and all four belong to different collections. Volumes fr. 2988 and fr. 3158 come from Béthune's collection,[14] volume fr. 20506 from Gaignières's collection and volume Cinq Cents de Colbert 470 from Colbert's collection. Even if work similar to the work done for Duprat's case (going through all the letters from the same collection) had been done, it would have been unsuccessful. Matching all four volumes was made possible only after successful cryptanalysis. According to Lasry/Biermann/Tomokiyo 2023, the familiarity with the ciphertext characters in

[14] Volume fr. 2988 was part of the ensemble related to Francis I's reign, while volume fr. 3158 was part of the ensemble related to Francis II's reign. Although Mary Stuart was Francis II's wife, the encrypted letters in fr. 3158 were not written during Francis II's reign (nor Francis I's).

volume fr. 2988 made possible a match with volume fr. 20506, on which Tomokiyo was working for another purpose (to add new collections to Cryptiana).[15] While this was partially by luck (looking at this volume at this particular time), this highlights the benefits of systematic surveys but also their limitation: Tomokiyo reconstructs keys but works only on documents that are digitized and put online by the archives and libraries (part of the collections only); the DECODE database is not limited to digitized documents but depends on the contributors intakes (Heder/Megyesi 2022); Desenclos is conducting a systematic survey at the BnF (Desenclos 2021), but it has not been yet published. For Mary Stuart's letters, identifying Castelnau as a possible recipient of the encrypted letters made the search feasible by inspecting only volumes that may contain letters from and to Castelnau (Lasry, Biermann, and Tomokiyo 2023). While some documents' history can be retraced, this is not always the case. As Mary Stuart's letters were addressed to Castelnau, they should have been (at least at first) preserved along with his papers. The fate of Castelnau's papers is unclear, and although a part of them has been preserved as an ensemble (fr. 3158), the other part has been scattered throughout history. Castelnau's papers may have been integrated into Montmorency's papers (through marriage and patronage), from which Philippe de Béthune, François-Roger de Gaignières, and Jean-Baptiste Colbert have built up their own collection. No matter which thread is pulled, there is no way (except for a tedious, exhaustive inventory of thousands of volumes) to match documents, whose consistency and identification are often lost for centuries. Analyzing a volume composition cannot be considered a solid way to identify a document, but rather a hint.

3.3 Reconstructing the key via cryptanalytic techniques

For unidentified encrypted texts, their decipherment through cryptanalysis is often the best, or even the only way to identify the text, to enable finding other encrypted texts (with the same key) and, of course, to access their content. Deciphering historical encrypted documents typically consists of four stages:

1. Computerized cryptanalysis to identify the meaning of most ciphertext characters representing individual letters of the alphabet – the homophones in the common case of homophonic ciphers.
2. Identifying the meaning of the remaining ciphertext characters representing individual letters of the alphabet and those representing doubled letters, nulls, or requiring an action (e.g., to delete the previous character). This phase is typically performed manually, based on the partial results obtained in Stage 1, by analyzing all instances of the given ciphertext characters. The more ciphertext material is available, the higher the chance those characters will be identified in their majority and accurately. Also, by examining which deciphered sequences are plausible and which are not, it is possible to determine which ciphertext characters are likely part of the nomenclature.[16]
3. Identifying the meaning of nomenclature elements representing common words, prepositions, or parts of words. This is also performed by examining the various instances of those elements in their textual and sometimes historical context.
4. Identifying nomenclature elements that represent proper names. Identifying the text and its historical context is critical at this stage (see Section 4).

These four stages are the same, regardless of whether the document has been identified. However, the absence of any identification element (date, nature, context) makes cryptanalysis harder, even impossible in some cases. Identifying the meaning of some nomenclature elements, either names (Stage 4) or infrequently used words (Stage 3), relies on prior knowledge of the context; without it, cryptanalysis will remain incomplete. Identifying, in Stages 1 and 2, the ciphertext characters for the letters of the alphabet (homophones) and frequent nomenclature words (Stage 3), in contrast, will generally still be feasible given enough ciphertext, but may require testing more hypotheses, notably for two reasons: both the language and the cipher

[15] Cryptiana is a website that aims to identify and reconstruct historical ciphers from archives documents in (https://cryptiana.web.fc2.com/code/crypto.htm).

[16] This also allows for determining the structure of nomenclature symbols, such as the addition of certain diacritics, or specific ranges of numerical codes.

features are completely unknown.[17] Cryptanalysis software and algorithms rely on language statistics computed from a corpus of the original language. Even for encrypted letters with the slightest identification, language can be determined or, at the very least, narrowed down to a few possibilities. Assuming the wrong language will cause cryptanalysis algorithms to fail. For Mary Stuart's case, as shown in Lasry/ Biermann/Tomokiyo 2023, a wrong initial assumption based on the catalog and the content of the first studied volume (fr. 2988) – that the letters were originally in Italian – resulted in the project being significantly delayed. If the language is unknown and cannot be inferred from other clues or if the first assumption does not give any conclusive result, all possible languages (and their territorial variations) but also all possible types of ciphers (transposition, homophonic substitution, polyalphabetic substitution, …) and all possible structures of ciphers must be tested for successful cryptanalysis while they are usually guessed from the date, the nature and the author of the document. A French cipher from the 1580s is more likely to be homophonic, with 2-4 homophones per alphabet letter, to utilize nulls, and to feature a nomenclature that incorporates both names and common words. In contrast, a French cipher from the 1520s may be monoalphabetic or have a low number of homophones and a small nomenclature (mostly names). Both structures can be observed in the fr. 6632 letter (homophonic cipher with at two to four homophones per letter of the alphabet) and in Duprat's reports (monoalphabetic substitution, twelve nomenclature elements, and five nulls).

French Renaissance ciphers were less complex and had fewer features, so the cryptanalysis of earlier encrypted texts was usually easier. With Duprat's reports, given the small number of distinct ciphertext characters (about 35), it was hypothesized that the cipher could be a simple monoalphabetic substitution cipher with a few additional elements, such as names (small nomenclature) and nulls. Based on the mention "Lettre en chiffre adressée au roy",[18] in the top margin of one of the encrypted texts (fr. 3029, fol. 134), we rightfully expected the original language to be Middle French. Computerized cryptanalysis easily recovered the ciphertext characters assigned to individual letters of the alphabet; then, six characters representing nulls and four nomenclature elements were also recovered with manual work.[19] In contrast, when the cipher is homophonic and/or uses an important nomenclature, more hypotheses must be formulated, making cryptanalysis harder as these stages are seldom sequential, and the process is mainly iterative. When deciphering Mary Stuart's letters, as shown in Lasry/Biermann/Tomokiyo 2023, the special ciphertext characters (to delete or repeat the last ciphertext character) and the structure of the nomenclature elements – five different meanings for a single ciphertext character based on adding diacritics – were hard to identify. Some of the diacritics were first assumed to be individual ciphertext characters. Only after they identified the origin of the letters were Lasry *et al.* able to locate examples of related cipher keys in archives, similar in structure and contents to the one used in the collection they analyzed,[20] and then to confirm several cryptanalytic hypotheses. Even at this stage, the knowledge of the cryptographic practices (in a specific area, at a specific time) is fundamental for cryptanalysis, and unidentified texts deprive the codebreaker(s) of essential elements for success.

Those challenges are compounded when the cipher is complex (e.g., a large number of homophones) and/or if the ciphertext is short. For the third case study (fr. 6632), we have only obtained partial and tentative results, given the short length of the text (1,100 ciphertext characters) and the high number of distinct ciphertext characters (120). Based on this and the dates of the other letters in the volume (France, 1580s), we hypothesized that this cipher was homophonic with many homophones per letter and included at least a few tens of nomenclature elements. Computerized cryptanalysis (Stage 1)

[17] In the only other documented example of unidentified deciphered documents we have found (Kopal/Waldispühl 2022), the authors had also to base on assumptions about language, date and cipher type to process cryptanalysis.

[18] Letter in cipher, addressed to the King.

[19] Only six of the twelve nomenclature elements are used in the encrypted reports we have identified so far.

[20] Lasry *et al.* have found two ciphers, one used between Mary Stuart and Guillaume de L'Aubespine, baron de Châteauneuf, Castelnau's successor as French ambassador in England, and another preserved along Castelnau's papers. Both share similar features (diacritics, similar nomenclature vocabularies). Their finding helped Lasry *et al.* to confirm several cryptanalysis hypotheses (Lasry/Biermann/Tomokiyo 2023).

was challenging due to the cipher complexity and the short ciphertext, and it required multiple tests with various parameters, leaving many illegible gaps, as expected, because several ciphertext characters were wrongly assumed to be homophones. In Stage 2, all the homophones could be interpreted, corrected, and distinguished from nomenclature elements (which could not be interpreted at this stage), resulting in additional legible segments. Here, the transcription of the first six lines after Stage 2 (the homophones corrected at this stage are marked in bold, and nomenclature codes with a question mark):

"*Madame* ***dair*** *aiant eu* ***? ?****modité en une* ***f****este* ***?*** *se* ***f****eit ché****s*** *elle, entra* ***? ?*** *en propos* ***? ? ?*** *tres p****r****egnantes raisons* ***? ?*** *persuad****? ? ?*** *veoir* ***? ? ?****ntendre au ma****rr****iage sur* ***? ?*** *ses plaintes* ***? ?*** *gran?s protestations de sa"*

Stage 3 was tedious and yielded incomplete results, while Stage 4 could not be performed due to the short ciphertext and the lack of historical context. About ten ciphertext characters could not be identified, including some likely to represent proper names:

"*Madame, Dair aiant eu* ***la com****modité en une feste* ***que*** *se feit ches elle, entra* ***bien ?*** *en propos* ***avec ? avec*** *tres pregnantes raisons* ***pour luy*** *persua****de****r et de veoir* ***?*** *et* ***d'e****ntendre au mariage sur* ***? ?*** *ses plaintes* ***et les*** *gran****de****s protestations de sa"*

4 Identifying unidentified encrypted letters: challenges and methods

For Stages 3 and 4, in particular, knowing the historical context and identifying the document are crucial. Although that depends on the size of the nomenclature and the length of the encrypted document, cryptanalysis can fuel a virtuous circle: reconstructing the plaintext, even without names, can enable the identification of the encrypted document; this identification can, in turn, help identify the words and names (nomenclature); a more complete decipherment can improve the identification of the document, and so on.

Even if the names have not been deciphered yet, the first stages of decipherment may offer several levers for identifying the document, such as names spelled and encrypted in full, or references to an event or a person. As described in Lasry/Biermann/Tomokiyo 2023, Mary Stuart's letters were identified as such based on clues in the deciphered text. There were mentions of a son ("mon fils"), of being in captivity ("ma liberté"), as well as feminine adjectives, and names spelled in full, such as Walsingham (Elizabeth I's principal secretary and her spymaster) or Spain. Similarly, with Duprat's reports, the main clues came from names spelled out in full before encryption. Due to the small size of nomenclature, most of the names were spelled and encrypted in full (Fitzwilliam, bishop of Durham, Duke of Albany, …) and helped a lot to identify the documents as French reports produced during negotiations with England. This has been confirmed by mentions of discussions between "le chancelier" (the chancellor, here Antoine Duprat) and "le cardinal" (the cardinal, here Cardinal Thomas Wolsey) in the context of concurrent discussions with ambassadors from various places (Hungary, Venice, ...). All those pointed toward the negotiations that took place between July and November 1521 in Calais.

It may come as a surprise to see so many names not being encrypted with nomenclature elements. In Duprat's reports, the reason was the simplicity of the cipher: French ciphers in the early 1520s were not as advanced as Italian or Spanish ones, and here, the cipher was designed for a time-limited use in a not-too-far-away embassy. The needs, in terms of security, were not the same. In other cases, such as Mary Stuart's or even the letter from volume fr. 6632, that is not surprising either, and these are no exception. Not all names were intended to be encrypted through a nomenclature (infrequently-mentioned names, people who burst into the political scene after the cipher key was created, etc.). Moreover, encrypting names using homophone elements after fully spelling them out limited the frequency of nomenclature elements assigned to important names. For 21st-century codebreakers, having names encrypted in full facilitates the identification of the origin and context of the text, further assisting in the decipherment process. Certainly, such encryption practices also helped early modern codebreakers, but only in recovering names. When the letters were intercepted and deciphered, the (approximate) date of the letter and its probable origin were already known: the codebreaker knew when and where the letter had been intercepted. He could easily guess the language, nature, and features of the cipher, as well as some frequently used words and names. The decipherment of unidentified documents through codebreaking highlights the gap between early modern codebreakers and 21st-century cryptanalysis. The latter is disadvantaged because

the knowledge of the writing context has been lost, making identifying common words and names more complex, if not impossible. For example, with our third case study (fr. 6632), it is quite possible that an early modern codebreaker would have been able to identify the parties of the marriage mentioned in the text, whether or not the marriage negotiations were public (rumors, intelligence, etc.). Conversely, it is impossible, at our times, to identify those names without any other supporting element. Cryptanalysis may initiate a virtuous circle, but can also lead to a dead end. With the fr. 6632 letter, the challenge is increased by the frequent use of distinct nomenclature elements for names. The reconstructed text segments are short and insufficiently explicit about the people or events mentioned. Here lies a second challenge that 21st-century researchers face: many elements are implicit in historical documents, whether or not they are encrypted. People are never mentioned by their full names but only by shortened names, which can apply to several people from the same bloodline (e.g., the Duke of Albany), or by one of their titles (the cardinal, the Grand Squire, etc.). These made immediate sense to early modern people and can make sense to 21st-century historians when the date is available. Without additional information, these shortened names and titles can remain insufficient to enable the identification of a document.

In many cases, nevertheless, it is possible to identify, at least partially, the sender and recipient and the date. In Mary Stuart's letters (Lasry/Biermann/Tomokiyo 2023), the date and the recipient's name were encrypted and have been recovered via cryptanalysis. In Duprat's reports, no identification elements were written, even in cipher. As a first step, only assumptions can be made about the sender, recipient, or date. Although the sender is the most essential element, two difficulties may arise: the letter's content may not be sufficient to identify the source; there may be several senders. The letters of the French embassy in Calais were written and sent either on behalf of the four ambassadors (joint signature) or by only one ambassador (single signature). The phrase 'moy chancellier' (I, chancellor) might suggest that only Duprat wrote some letters; however, the same phrase is also found in unencrypted letters sent by the whole embassy.

Identifying the addressee can sometimes be easier when the encrypted document appears to be a letter. Letters begin with the addressee's name, or at least with an adapted term ('Mr. Chancellor'), which allows either identifying the addressee or formulating some hypotheses. These can be confirmed by the document's content and/or by additional documents. The cleartext letters from the French embassy in Calais, for example, were mostly addressed to Francis I or Florimont Robertet (who was then in charge of Foreign Affairs); it is easy to assume that reports were also addressed to one of them; but nothing in the report gives a clue about the precise addressee (no words such as "Your Majesty" or "Sir"). Except when the addressee's name is written at the beginning of the letter, identifying the addressee can be particularly complex if the sender is not identified. In the case of fr. 6632 letter, the first word of the letter ("Madame") could indicate that the sender was addressing a woman (possibly Catherine de Medici, given the other documents in the volume), but the text could also begin without addressing anyone. "Madame Dair" – the first two words of the text – could be one of the text's subjects rather than the addressee.

Even if it is often easy to determine the time range of a document, it can be harder to determine a more precise date (day, month, and year). Given the nature of the cipher (number of homophones, type of ciphertext characters, ...) and the content of the document, it may be possible to identify, at least, the decade in which it was written. Although we have been able to date Duprat's reports to 1521 (the date of the embassy), no precise date (day and month) has yet been established for the encrypted reports. Doing so will require reading every unencrypted letter in related volumes to establish a daily or, at least, weekly chronology of the embassy. This work may be successful, but, for now, we have only been able to provide approximate dates (e.g., between 2 October and 10 October) for some reports.

Even a partial or hypothetical identification may enable further work to be carried out: research in the archives can be conducted, as the new identification elements may help limit the investigation's scope. Nevertheless, challenges similar to those outlined in sections 3.1 and 3.2 may persist, depending on the varying accuracy of catalogs and on the information elements that remain missing. Although researchers may get stuck at this stage due to a lack of new identification elements, the deciphering, then the identification, even partial, of the document and,

possibly, the discovery of additional sources (letters, encrypted or not, produced by the same sender and recipient, cipher key, decrypted texts) can lead to valuable results, particularly with recovering nomenclature elements. In the same way that the linguistic context enables the identification of certain common words (by reconstructing the meaning of the sentence), the historical context enables the identification of remaining ciphertext characters, whether they represent common words ("army," "peace," etc.) or proper names (people and places). Even before linking it to the cipher key copied by Jean Barillon, the identification of Duprat's letters had enabled the identification of four nomenclature elements. Only two causes can prevent the nomenclature from being completely reconstructed: the frequency of the ciphertext characters is too low, and they cannot be attributed with certainty to a word or name; the encrypted text is too short and contains numerous nomenclature elements.

5. Conclusion

Regardless of their date or origin, unidentified encrypted documents are less prevalent than identified ones. Researchers often do not address them, and these documents pose multiple historical, archival, and cryptanalytic challenges. Being deprived of any identification element complicates every step of the document analysis, especially for early encrypted documents. When facing an encrypted document, the first reaction is to search for the original cipher key, or encrypted and possibly decrypted texts. This is most often not feasible with unidentified documents, requiring exhaustive and time-consuming work in archives and libraries or a fair amount of luck. Our three case studies suggest that cryptanalysis may be the most effective way to achieve, quicker and better, the identification (and analysis) of early modern documents, even though cryptanalysis itself faces additional challenges as neither the language, the historical context, nor the cipher type is known a priori. However, cryptanalysis can enable the identification of the document thanks to the deciphered content.

While the three cases have varying success (depending not only on the absence of identification but also on the size of the encrypted texts), they all emphasize the dual nature of codebreaking for early modern documents: it does not rely only on cryptanalytic techniques but also on historical understanding. Reconstructing the nomenclature, for instance, depends on the document's identification and historical knowledge of the context. With unidentified documents, even if it is easier to describe the cryptanalysis process as distinct stages, it looks more like an iterative process: multiple tests are made based on hypotheses, cryptanalysis leads to better identification, which leads to better cryptanalysis… This representation of the process as iterative highlights the benefits of continuous collaboration between codebreakers and historians: historical and cryptanalytic inputs are relevant and essential throughout the entire process, not just at the beginning (cryptanalytic techniques) or at the end (historical knowledge).

Funding

The Swedish Research Council has supported the work of one author, grant 2018-06074, DECRYPT – Decryption of Historical Manuscripts.

6. References

Barillon, J., 1897-1899. *Journal de Jean Barillon, secrétaire du chancelier Duprat, 1515-1521*, ed. Pierre de Vaissière, 2 vol.

Biermann, N., Tomokiyo, S., and Lasry, G., 2023. *What encryption errors can reveal: cross-cipher errors in Mary Queen of Scots' letters*. Proceedings of the 7th International Conference on Historical Cryptology HistoCrypt2024, 17-26.

Desenclos, C., 2021. *Écrire le secret quotidien. Pratiques de la cryptographie au sein de la diplomatie française (XVIe-premier XVIIe siècle).* Spies, Espionage, and Secret Diplomacy in the Early Modern Period, 85–103.

Lasry, G., Biermann, N., and Tomokiyo, S., 2023. *Deciphering Mary Stuart's lost letters from 1578-1584*. Crypotologia, 47, 101-202.

Héder, M., Megyesi, B., 2022. *The DECODE Database of Historical Ciphers and Keys: Version 2.* Proceedings of the 5th International Conference on Historical Cryptology HistoCrypt2022, 111-114.

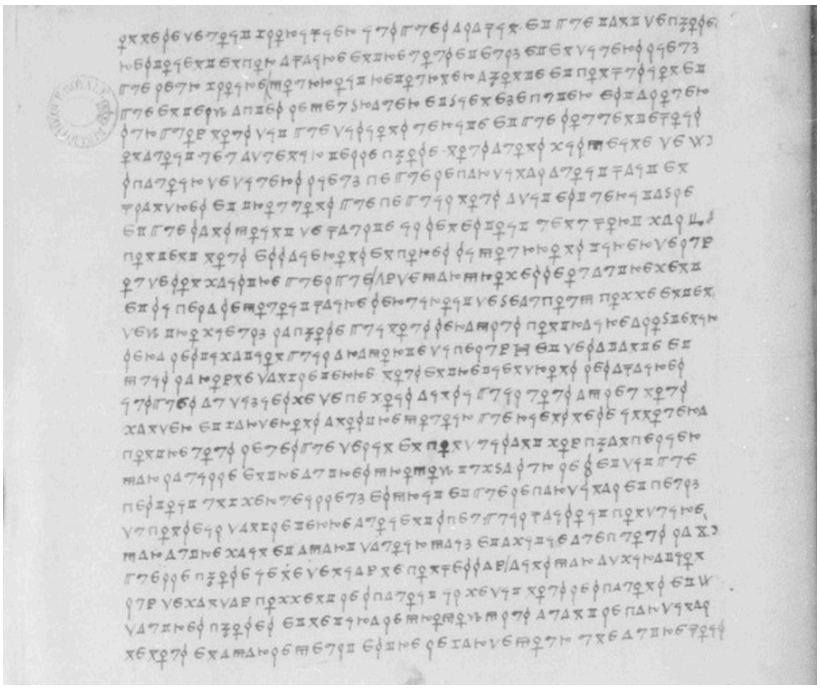
Kopal, N., Waldispühl, M., 2022. *Deciphering three diplomatic letters sent by Maximilian II in 1575.* Cryptologia, 46, 103-127.

Russel, J. G., 1971. *The Search for Universal Peace: the Conferences at Calais and Bruges in 1521*. Historical Research, 44, 162.

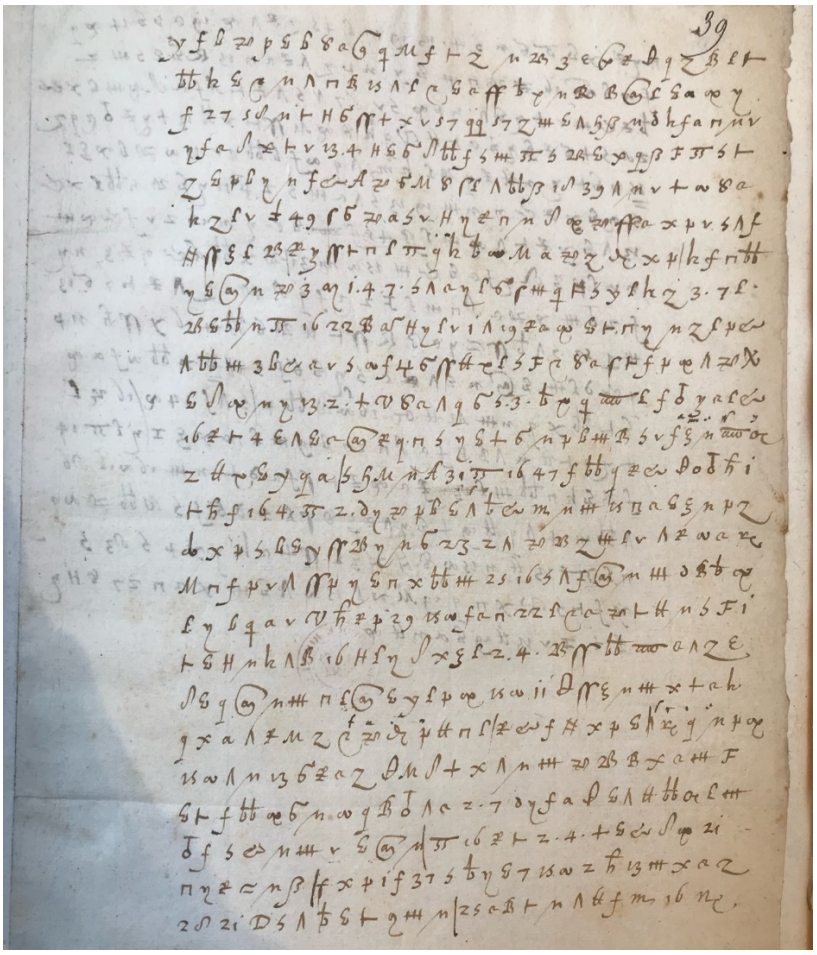
Solente, S., 1980-2001. *Les manuscrits des Béthune à la Bibliothèque nationale*.

7. Appendix

7.1. End of a scattered Duprat's report (BnF, fr. 3092, fol. 101 ©Gallica-BnF)



7.2. First page of the fr. 6632 letter (BnF, fr. 6632, fol. 39 ©BnF)



Playfair crib validation as a constraint satisfaction problem

Magnus Ekhall
Private researcher
magnus.ekhall@gmail.com

Abstract

This paper shows how a crib[1] for a Playfair enciphered message can be seen as a constraint satisfaction problem. This problem can then be solved with standard constraint programming tools. The solution will tell whether the crib is possible, and if so, can list all possible Playfair keys that would result in the given crib.

1 Introduction

The Playfair cipher, invented by the physicist and inventor Sir Charles Whetstone in the 19th century, is a bigram substitution cipher. As a key, 25 out of the 26 letters of the alphabet are written into a matrix of size 5×5. Usually the letter J is the omitted letter, with I used as a substitute (Kahn, 1996).

Playfair is a bigram cipher which means that it always enciphers or deciphers pairs of letters. Also, a letter pair cannot consist of the same letter. If needed, a padding letter (usually X) can be used to work around these situations. Example: If the plaintext that shall be enciphered is `MOONCHEESE` the padded plaintext letter pairs would be `MO ON CH EX ES EX`.

1.1 The three Playfair rules

The actual encipherment is carried out using three rules which will be referred to as the Playfair rules. The letter pair to be enciphered are looked up in the key matrix. If the letters happen to be on the same row in the key matrix, rule 1 applies. If the letters instead are in the same column in the key matrix, rule 2 applies. If the letters are neither on the same row nor in the same column, rule 3 applies.

[1] A crib is a known or guessed plaintext for (part of) an enciphered message.

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

Table 1: A Playfair key

Rule 1: For each of the two plaintext letters, take the letter to the right in the key matrix. This is done in a cyclic manner.

Rule 2: For each of the two plaintext letters, take the letter directly under in the key matrix.

Rule 3: For each of the two plaintext letters, take the letter on the same line of the key matrix that also is on the same column as the other plaintext letter.

Using the key in table 1 the example plaintext above would be enciphered to:

`MO` $\rightarrow$ `XY` (Rule 1)

`ON` $\rightarrow$ `ZA` (Rule 2)

`CH` $\rightarrow$ `UW` (Rule 3)

`EX` $\rightarrow$ `LY` (Rule 3)

`ES` $\rightarrow$ `CP` (Rule 3)

`EX` $\rightarrow$ `LY` (Rule 3)

Deciphering is done in a similar fashion. When deciphering, rule 1 selects the letter to the left, and rule 2 the letter above. Rule 3 works just as when enciphering.

The three Playfair rules together with the nature of the key matrix imposes some constraints on what a plaintext letter can be enciphered into. For example, a plaintext letter can only be enciphered to five different cipher text letters: the four others on the same row as the plaintext letter and the letter directly under it. The constraints will be in focus in this paper.

Proceedings of the 8th International Conference on Historical Cryptology, pages 52-59
Poznań, Poland, 16-18 June, 2025

-	-	A	-	-
M	X	O	Y	-
-	-	Z	-	-
-	-	-	-	-
-	-	N	-	-

Table 2: A partial Playfair key

-	Z	-	-	-
-	-	-	-	-
-	N	-	-	-
-	A	-	-	-
X	O	Y	-	M

Table 3: An equivalent Playfair key

1.2 Related work

Modern cryptanalysis of Playfair enciphered messages involve computer algorithms such as hill climbing. This works most reliably with long messages and can be used as a ciphertext-only attack. For shorter messages other methods can be tried for example by testing whether certain words would be possible as the plaintext in different manners (Dunin et al., 2022).

When performing cryptanalysis with pen-and-paper methods, both cribs and taking advantage of the constraints imposed by the Playfair rules have been commonplace (US Government, 1990; Friedman, 1938).

1.3 Problem statement

This paper aims to show how the common pen-and-paper method for breaking Playfair messages (US Government, 1990) can be viewed as a constraint satisfaction problem and, as such, can be computerized using standard constraint programming tools. More specifically this applies to the tedious work related to calculating possible Playfair keys given a certain crib for an enciphered message.

2 Playfair crib validation

Whether a crib is possible for a Playfair enciphered text is determined by if there is at least one Playfair key that would encrypt the crib to the ciphertext. Such a key can be "partial" in the sense that only those letters used in the crib and its corresponding ciphertext need to be present in the key matrix.

For example, if the crib `MOON` is to be tested at the beginning of the ciphertext `XYZAUWLYCPLY`, then the partial key shown in table 2 would show that the crib is possible.

When a partial key derived from a crib is used to decrypt a message it might not be possible to decrypt the whole message. Only those bigrams where the plaintext letters and the corresponding ciphertext letters are present in the key matrix will be possible to decrypt. The crib itself, however, will always be possible to decrypt.

How many different keys exist for a given crib? In order to answer that question certain properties of the Playfair key should be considered.

A Playfair key matrix shifted horizontally or vertically results in a new matrix that encrypts or deciphers exactly the same as the original. All such keys are therefore equivalent with respect to decryption or encryption. As an example, table 3 shows a key which is equivalent to that shown in table 2, shifted to the left once and up twice.

The pen-and-paper method for deriving partial Playfair keys from a crib described in (US Government, 1990) is an iterative process.

It can be explained as follows:

Algorithm 1: Keys from Playfair crib

An empty key, k_0, is created as the only member of a set, K, of current keys.
The crib is processed bigram by bigram. For each bigram, all keys in the set K are examined, one by one. For each of the keys it is tested whether one or more of the three Playfair rules can be used to produce the current bigram from the crib. This may involve adding new letters to the key and can result in up to three new keys: one for each of the three Playfair rules that could successfully be used. The conclusion could also be that none of the three Playfair rules can be used to produce the bigram. In that case that particular key is removed. Otherwise, any new keys are kept as the new members of the key set K to be used in the next iteration.
In the first iteration, only k_0 is present but the number of possible keys can grow exponentially.
The algorithm ends when either the key set K becomes empty or all bigrams of the crib

have been successfully used to construct prospective keys.
If the key set ends up empty, this indicates that it is impossible to construct a key that decrypts the ciphertext into the crib, proving the crib incorrect.
Conversely, if there are keys in the key set at the end of the last iteration that shows that it is possible to decrypt the ciphertext into the crib at the given position, but there is no guarantee that the crib corresponds to the original plaintext.

A difficulty with algorithm 1 is that it is challenging to say exactly where in the key matrix the attempted use of the different Playfair rules shall be placed. In (US Government, 1990) this decision is left to the reader to handle. This problem can be handled by introducing the constraint satisfaction problem that stems from the Playfair rules.

3 Constraints from Playfair rules

The three Playfair rules imposes three different sets of constraints on the letters they operate on. All constraints are cyclic or modular with respect to the five-by-five size of the Playfair key matrix.

3.1 Rule 1 constraints

Rule 1 applies when both letters of plaintext are on the same line in the key matrix. When enciphering, the ciphertext letters are the letters immediately to the right of the plaintext letters.

For this to be possible the following must be true:

- The two ciphertext letters must be on the same row in the matrix.
- The first ciphertext letter must be directly to the right of the first plaintext letter.
- The second ciphertext letter must be directly to the right of the second plaintext letter.

3.2 Rule 2 constraints

Rule 2 is similar to rule 1, but here the two plaintext letters are in the same column in the matrix.

For this to be possible the following must be true:

- The two ciphertext letters must be in the same column in the matrix.
- The first ciphertext letter must be directly below the first plaintext letter.
- The second ciphertext letter must be directly below the second plaintext letter.

3.3 Rule 3 constraints

If the two plaintext letters are not on the same line nor in the same column, this produces the following constraints:

- The ciphertext letters must be on different rows in the matrix.
- The ciphertext letters must be in different columns in the matrix.
- The first plaintext letter must be on the same row as the first ciphertext letter.
- The second plaintext letter must be on the same row as the second ciphertext letter.
- The first plaintext letter must be in the same column as the second ciphertext letter.
- The second plaintext letter must be in the same column as the first ciphertext letter.

3.4 Stragglers

Sometimes a crib does not produce complete bigrams. The beginning or end of the crib may end up so that only one plaintext letter is present in a bigram. Take for example the crib `FOR` with the enciphered message from the example above. The two bigrams involved are then:

```
MR BY ...
FO R
```

These “stragglers” produce what can be seen as half bigrams. In this case, since only one plaintext letter is available, the constraints listed above do not fully apply to stragglers. Instead a reduced set of constraints can be used in this case: the constraints that involves the missing plaintext letter are simply ignored.

4 Constraint satisfaction problem

When implementing algorithm 1 on a computer, the constraints described in section 3 can be used to handle the fact that the precise positioning of the letters of the bigrams in the prospective Playfair key are not known. Even if the position itself is not known, the constraints imposed by the Playfair rules still apply and must hold true.

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

Table 4: Integers representing the positions in the key matrix.

This is a constraint satisfaction problem, which is a well-studied field. There are several computer implementations of tools to solve this kind of problem which go under the name "constraint programming".

4.1 Constraint satisfaction problem example

In order to illustrate what a constraint satisfaction problem can look like here follows a short (and unrelated) example:

Consider the task of assigning integers to three variables under specific constraints:

- **Variables:** X, Y, Z
- **Domains:** Each variable takes a value from $\{1,2,3,4,5\}$.
- **Constraints:**
  - $X + Y = Z$
  - $X < Y$
  - $X \neq Y, Y \neq Z, X \neq Z$

The goal is to find integer assignments to X, Y, and Z that satisfy all the given constraints simultaneously.

This problem has several solutions, for example $X = 2, Y = 3, Z = 5$.

4.2 Playfair crib constraints

To efficiently be able to describe the constraints in section 3 with a constraint programming tool it is helpful to use integers as references to specific places in the key matrix.

With the representation as in table 4 the various constraint types used in section 3 can be described mathematically in a way that many constraint programming tools use:

- x is on the same row as y: $\lfloor x/5 \rfloor = \lfloor y/5 \rfloor$
- x is in the same column as y: $x \bmod 5 = y \bmod 5$
- x is directly to the right of y: $\lfloor x/5 \rfloor = \lfloor y/5 \rfloor \wedge x \bmod 5 = ((y \bmod 5) + 1) \bmod 5$
- x is directly under y: $x \bmod 5 = y \bmod 5 \wedge \lfloor x/5 \rfloor = (\lfloor y/5 \rfloor + 1) \bmod 5$

5 Implementation

An implementation was made in the Python programming language. The Python library CPMpy (Guns, 2019) was used as an interface to a OR-Tools, a free and open source software suite developed by Google. OR-Tools includes an award winning SAT-solver for solving constraint programming problems (Perron and Furnon, 2024).

In CPMpy, a "model" consists of a set of decision variables and a set of constraints. With the representation of table 4 all decision variables used are integers $v \in [0,24]$, where each v is unique. A model can then be solved by the SAT-solver, which means that the SAT-solver will try to find specific values for the decision variables that fulfills all the constraints.

As input, the program takes a Playfair enciphered message and a crib for a part of the message. The program implements algorithm 1 using the mathematical constraints as described in section 4.

The program start with an empty set of models. For the first bigram of the crib three new models are created: one for each of the three Playfair rules. The newly created models are given decision variables corresponding to the letters involved in the plaintext and ciphertext bigram, as well as the constraints stemming from the Playfair rules.

For each of the remaining bigrams all models are tested with new decision variables and constraints from each of the three Playfair rules. If the SAT-solver can find at least one solution, the model is kept in the set of models. If not, the model is thrown away.

If this intermediate testing was not done, the size of the model set would grow exponentially which would make it difficult to handle cribs with more than a few bigrams.

When the last bigram of the crib has added, the program instructs the SAT-solver to retrieve all possible solutions for every model in the model set.

This results in the output of the program: a list of all possible partial keys for which the ciphertext deciphers to the crib. The program will also decrypt as much as possible of the ciphertext with

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

Table 5: Key used in the example

E	U	R	Y	T
M	.	N	X	B
.	.	D	G	.
C	.	L	A	.
.	.	.	.	.

Table 6: One of the partial keys output from the example

every partial key. From this output it is sometimes possible to draw conclusions on plaintext letters *outside* the crib as will be shown in an example below.

As an alternative, the program can be instructed to stop as soon as one solution has been found. In that case the program will only answer the question whether it is possible for the crib to exist and present one example of a partial key that satisfies the crib.

5.1 Optimizations

The implementation also takes advantage of the fact that a Playfair key matrix can be shifted vertically and horizontally without affecting the result, as showed in section 2. This is done by fixing one of the letter variables, l, to a fixed position in the key matrix: $l = 0$. It does not matter which letter that is selected, or what position in the letter matrix it is fixed to. The fixing of one letter reduces the key space with a factor of 25.

One further optimization can be done. The order in which the various bigrams from the crib are tested matters. To reach a conclusion as quickly as possible it is advantageous to try to get a contradiction as soon as possible. Therefore the program first looks at all the available bigrams and starts with those that have many letters in common. This increases the probability to get contradicting constraints which in turn will reduce the number of models that the SAT-solver will need to investigate.

6 Examples

Below are two examples, the first with a plausible (and in fact correct) crib which results in a number of partial keys. The second example is with a crib that turns out to be impossible.

Both example use the plaintext message "Next telegram must be enciphered":

`NEXTTELEGRAMMUSTBEENCIPHERED`.

Enciphered with the key in table 5 the enciphered message becomes:

`MRBYEUCRDYCXZEFEMTRMKPQSUYRS`.

6.1 Plausible crib

If the (correct) crib `NE XT TE LE GR AM` is tested, the following sequence of events occur:

1. The first bigram chosen from the crib is `TE` which is matched against ciphertext bigram `EU`. After this step, the number of models in the model set is 2. That means that there are two different sets of constraints that both independently can produce the crib `TE` from the ciphertext `EU`.
2. Second bigram `NE` $\rightarrow$ `MR`. Model set size: 4.
3. `LE` $\rightarrow$ `CR`. Model set size: 4.
4. `XT` $\rightarrow$ `BY`. Model set size: 7.
5. `GR` $\rightarrow$ `DY`. Model set size: 1.
6. `AM` $\rightarrow$ `CX`. Model set size: 1.

As can be seen, the number of plausible models first increases, but later decreases when more bigrams and therefore more constraints are added.

The execution time on a normal PC with this example is about 0.5 seconds, and the output consists of 48 partial keys. The first of the partial keys is shown in table 6.

The partial decrypt with the key from table 6 is `NEXTTELEGRAM....BEEN....ER...` As can be seen three bigrams outside the crib have been deciphered with this particular key. The "extra" bigrams can vary with the various partial keys. If all 48 partial keys are considered it can be noted that the following plaintext is common for all keys: `NEXTTELEGRAM....BEEN....E....` That means that if the crib `NEXTTELEGRAM` is correct, then it is certain that the other letters are also correct. Given the "extra" letters, new words can be guessed and the crib thus expanded until the whole message is deciphered. If unlikely letter combinations appears as "extra" letters it can be suspected that

even if the crib was possible, it may be the wrong solution.

6.2 Impossible crib

For this example the crib "next message": `NE XT ME SX SA GE` is used with the same ciphertext as in section 6.1.

The sequence of events becomes as follows:

1. The first bigram chosen from the crib is `ME` which is matched against ciphertext bigram `EU`. After this step, the number of models in the model set is 2.
2. Second bigram `GE` → `CX`. Model set size: 4.
3. `NE` → `MR`. Model set size: 2.
4. `SX` → `CR`. **Model set size: 0.**

After four of the six bigrams have been added there are no models left that satisfy the constraints. The crib can be proven to be impossible and the program is terminated. Execution time about 0.35 seconds.

7 Results

To test the efficiency of the method with different cribs, a list of 1000 common English words with a length of at least two letters was created (Fletcher, 2003 2010). The words were then used as a crib at the beginning of the example message from section 6, one word at a time. The time it took for the program to come to a conclusion (either positive or negative) was recorded and is shown in figure 1. It can be seen that the median time to reach a conclusion is below half a second. The worst case is 2.45 seconds and the word in this case, `TALKING`, has no repeated letters, nor does it have any letters in common with the corresponding ciphertext. Of the 1000 words, 678 were found to be possible cribs with at least one partial key. The word length varied from two to thirteen letters, with the median being five.

The results from figure 1 can be partitioned into two parts: the possible cribs that were tested and the impossible cribs. This is shown in figures 2 and 3 respectively. It can be noted that the cribs with the longest processing time were the possible ones. The reason is that in the impossible case the algorithm will stop as soon as a contradiction is encountered, while in the possible case all bigrams need to be processed.

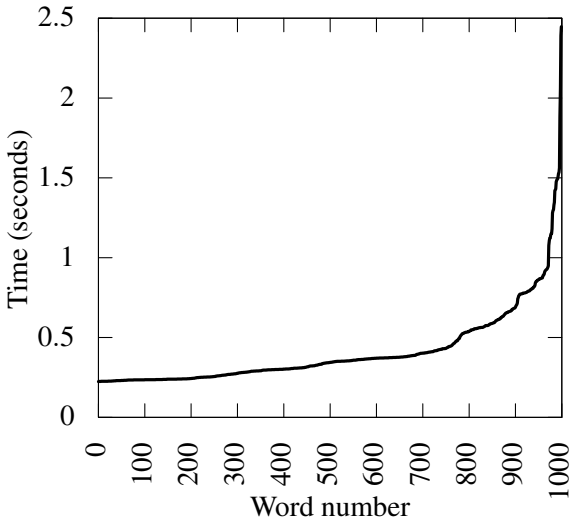


Figure 1: Runtime for 1000 common words, sorted by runtime.

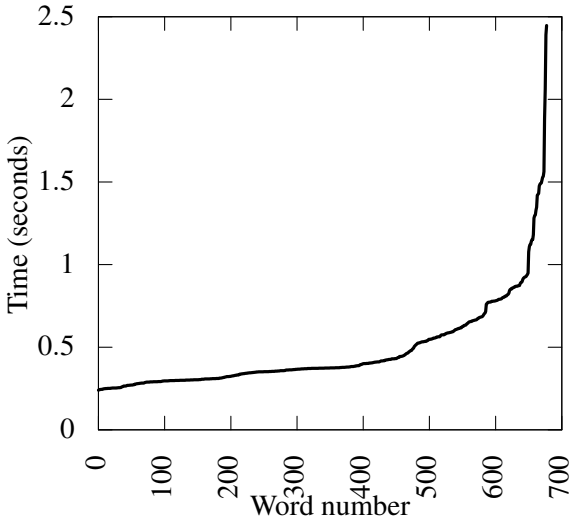


Figure 2: Runtime for the 678 of the 1000 words that were concluded to be possible cribs. Sorted by runtime.

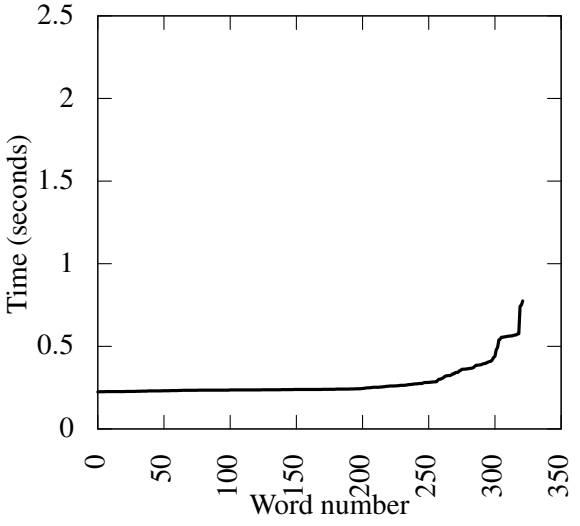


Figure 3: Runtime for the 322 of the 1000 words that were concluded to be impossible cribs. Sorted by runtime.

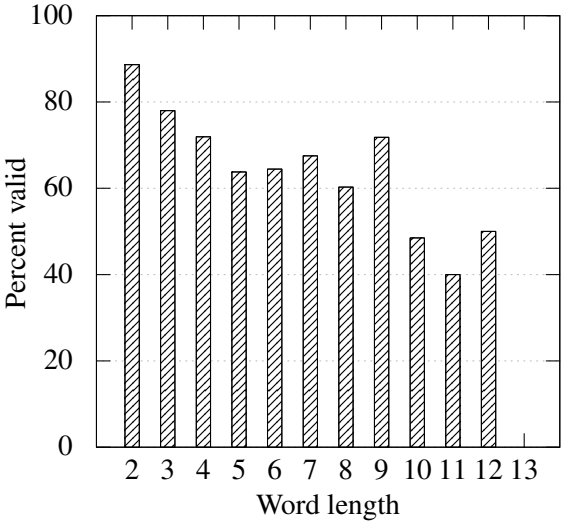


Figure 4: Histogram showing the percentage of possible cribs for all different word lengths in the 1000 common word line.

It might also be interesting to see the relationship between crib length and the chances of the crib being deemed possible. This is visualized in a histogram in figure 4 using data from the run of the 1000 common words. As can be seen, there is a clear tendency for shorter words to be possible cribs.

The longer the crib, the higher is the chance to quickly find a contradiction since the program generally has more letters to pick from. This can be seen in figure 5 where a list of 1000 long English words were used as cribs, one word at a time. The shortest word was fourteen letters and the longest twenty-two. The median word length was 15 letters. The word that took the longest time to test was `PHANTASMAGORIA`, which took 1.36 seconds to process. This word was found to be impossible as a crib, but only after processing all the bigrams of the word. Of the 1000 long words processed, only eight were found to be possible cribs.

If the whole, correct plaintext is used as a crib the program runs for 0.57 seconds before presenting a correct partial key with only three empty places in it.

In general it can be concluded that the Playfair cipher is very well suited to be used with constraint logic programming due to the nature of the key and the consequences of the three Playfair rules. The method to gradually build a set of prospective Playfair keys has been known since

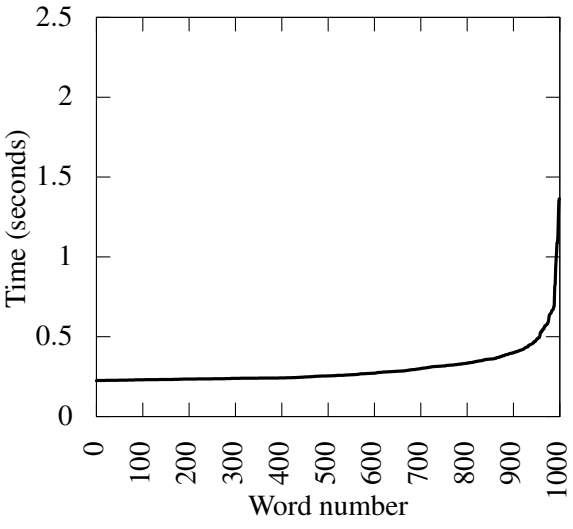


Figure 5: Runtime for 1000 long words, sorted by runtime.

before the advent of the computer. For example, this method is partially described in the novel "Have His Carcase" by Dorothy L. Sayers, published in 1932. By using state-of-the-art constraint programming software tools together with the old pen-and-paper method, things can be significantly accelerated.

Acknowledgments

The author would like to thank Henning Gideskog for his valuable comments on an earlier draft of this paper. Thanks are also due to Lisa Hallingström for drawing attention to the parallels between the method used in this study and the approach to the Playfair cipher in the novel *Have His Carcase*.

References

Elonka Dunin, Magnus Ekhall, Konstantin Hamidullin, Nils Kopal, George Lasry, and Klaus Schmeh. 2022. How we set new world records in breaking playfair ciphertexts. *Cryptologia*, 46(4):302–322.

William H. Fletcher. 2003-2010. Phrases in English. http://phrasesinenglish.org. Accessed 2025-01-04.

William F. Friedman. 1938. *Military Cryptanalysis, Part I*. War Department, Washington, Office of the Chief Signal Officer.

Tias Guns. 2019. Increasing modeling language convenience with a universal n-dimensional array, cppy as python-embedded example. In *Proceedings of the 18th workshop on Constraint Modelling and Reformulation at CP (Modref 2019)*, volume 19.

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

Laurent Perron and Vincent Furnon. 2024. OR-Tools. https://developers.google.com/optimization/. v9.10, Google.

US Government. 1990. Basic cryptanalysis, field manual no. 34-40-2. http://www.contestcen.com/ArmyFieldManual.pdf. Headquarters, Department of the Army, Washington, DC. Accessed 2024-12-29.

Machine learning for text classification in classical cryptography

Floe Foxon
University of Leeds
United Kingdom
floefoxon@protonmail.com

Abstract

This study furthers previous work on text classification to distinguish between ciphertext and gibberish. The statistical/linguistic properties of four text types were studied: meaningful English text, and three gibberish types ($n = 1,250$ each; total $N = 5,000$). Dimension reduction techniques (PCA, t-SNE, and UMAP) were used to reduce the statistical/linguistic feature space of the texts to two dimensions, revealing distinct regions of (lower dimensional) feature space occupied by each text, with some overlap. Machine learning models including random forests, neural networks (NNs), and support vector machines (SVMs) were used to classify the four text types based on their statistical/linguistic properties. Nested cross-validation revealed better generalization performance for the NNs and SVMs, classifying texts with $> 90\%$ accuracy. Applied to the Dorabella cryptogram, the models suggest that this text resembles meaningful English text more closely than gibberish types, which comports with the Dorabella cryptogram as a monoalphabetic substitution cipher, but this classification should be interpreted with caution. Features that better separate meaningful English from English-like gibberish are needed, and other encryption schemes/cryptograms should be explored with these methods.

1 Introduction

Previous work (Foxon, 2024) explored the the possibility of distinguishing between English text and 'gibberish' text generated at random from a uniform probability distribution in the context of classical cryptography, namely for the identification of 'hoax' cryptograms. The statistical and linguistic characteristics of those two datasets (English and uniform gibberish) were calculated, and a simple neural network (no hidden layers) was trained to distinguish between the two. The previous model had a high cross-validated mean accuracy (99.8%), and classified the Dorabella cryptogram as most likely representing English text (probability 0.9999) as opposed to uniform gibberish (probability 0.0001).

While taking a necessary first step toward successful meaningful/gibberish classification for putative cryptograms, the previous work had significant limitations in its scope. First, uniform gibberish is not the only alternative to meaningful English text that may be represented by a putative cryptogram. Second, the actual differences between the English and uniform gibberish texts were obscured by the high dimensionality of the data; no attempt was made to visualize the feature space. Third, the model in the previous work was selected arbitrarily with no optimization.

This work expands upon the previous work in three significant ways. First, the previous binary dataset is expanded to include two additional categories of text besides English and uniform random to potentially improve classification. Second, these texts are visualised in lower dimensional feature space using dimension reduction techniques to assess their separability. Third, a comprehensive approach to model selection is taken with nested cross-validation, comparing different types of machine learning models across different model hyperparameters.

Proceedings of the 8th International Conference on Historical Cryptology, pages 60-64
Poznań, Poland, 16-18 June, 2025

2 Methods

The methods used in this study are described in detail in the online supplemental materials, which contain a fully documented Jupyter notebook with statistical analysis code and text description along with the data:
`https://doi.org/10.17605/OSF.IO/ZYMTD`.
These methods are summarised only succinctly in the following. Readers are directed to the supplement for elaboration.

2.1 Data

As in the previous work, a training and testing set of meaningful English-language plaintexts was generated from Wikipedia articles (Davies, 2015), and a training and testing set of uniform gibberish was generated from a uniform probability distribution in which each character (lowercase English alphabet only) had equal probability $\left(\frac{1}{26}\right)$ of being selected next. These two types of text are referred to as 'English Corpus' and 'Uniform Random' hereafter.

Two new types of text were defined for the present study as follows.

- 'English-Like Random' text was generated by taking the frequency distribution of the English Corpus text and generating gibberish in which the letter 'e' had the highest probability of being selected next ($p \approx 0.116$), and 'q' the lowest ($p \approx 0.001$), as in the English Corpus.

- 'Typewriter-Like Random' text was generated, in theory, by a human typing 'at random' on a typewriter. The text used to represent Typewriter-Like Random text was the lowercase English alphabet characters of the 'Rilke cryptogram', for which there is some statistical evidence that the 'cryptogram' is a random sequence created with a typewriter (Foxon, 2022). While this identification is not certain, it was hypothesised that the Rilke text effectively represents a distinct category from the other three text categories in this study (later confirmed in the results). Readers may think of this category as 'Unknown' if preferred.

For each of the four text categories, $n = 1,250$ strings were generated for a well-balanced total of $N = 5,000$ strings, each 87-characters in length to match the Dorabella cryptogram.

As before, the values for 15 statistical and linguistic properties, including alphabet length, frequency, distance, entropy, index of coincidence, Zipf's exponent, and mean associated contact count were calculated for each string in each of the four text types.

2.2 Dimension Reduction and Data Visualisation

Since it is not possible to visualize the differences between each text in 15-dimensional space, dimension reduction techniques were used to reduce the number of dimensions of the feature space while retaining much of the variance. Three dimension reduction techniques were applied to the four text types in this study, namely Principal Component Analysis (PCA; Hotelling (1933); implemented with scikit-learn 1.3.0 (Pedregosa et al., 2011)), t-distributed stochastic neighbor embedding (t-SNE; van der Maaten and Hinton (2008); implemented with scikit-learn) and Uniform Manifold Approximation and Projection (UMAP; implemented with the UMAP Python package; see McInnes and Healy (2018)). By reducing the original 15-dimensional feature space to two effective dimensions, it was possible to visualize the separability of the data and provide an initial prediction for the classification of the Dorabella cryptogram prior to rigorous modelling. Figures were generated with matplotlib 3.7.2 (Hunter, 2007).

2.3 Machine Learning Models

Multi-class classification of the four-text dataset was achieved with three machine learning approaches. Namely, random forests (RFs; Breiman (2001)), neural networks (NNs; Rosenblatt (1958)), and support-vector machines (SVMs; Cortes & Vapnik (1995)), all implemented with sci-kit learn. To estimate the generalization performance of these models, 3×3 nested cross validation (NCV) was implemented. Grid searching was used over the hyperparameter space of each model, including a range of maximum tree depths, impurity criteria, and number of estimators for the RFs; solvers, activation functions, learning rates, and hidden layer sizes for the NNs; and kernel types, polynomial degrees, and regularization parameter values for the SVMs, described in the supplemental materials. These models were applied to the full 15-dimensional dataset.

Finally, the most accurate models were applied to the Dorabella cryptogram to obtain a possible

identification of its text type.

Analyses were conducted in Python version 3.8.16 with the additional packages Numpy 1.24.3, Pandas 2.0.3, Scipy 1.10.1, and Natural Language Toolkit 3.9.1.

3 Results

Figure 1 shows the letter/unigram frequency distributions for each type of text. Notably, the English-Like Random and English Corpus types have identical distributions (by design), the Uniform Random distribution is flat, and the Typewriter-Like Random distribution lies somewhere in-between.

Figure 2 shows the results of the dimension reduction techniques, plotting each string in two-dimensional feature space, where each of the reduced dimensions is a function of the original 15 dimensions of the data. These plots suggest that each type of text occupies a relatively distinct area of the reduced feature space (or 'text space'), with some overlap, particularly between the English Corpus and English-Like Random sets. The Dorabella cryptogram lies somewhere in the boundary between the clusters of English Corpus and English-Like Random text, and is quite distinct in its properties from Uniform Random and Typewriter-Like Random text. Note however that this is only a projection of the feature space onto two dimensions; there may be significant difference between English Corpus and English-Like Random text obscured by the loss of variance in dimension reduction. These plots at least suggest that classification of the four text types is feasible.

Table 1 shows the testing and training accuracies for the machine learning models. NNs and SVMs had $\sim 94\%$ NCV testing accuracy, with little difference between testing and training accuracies, indicating little-to-no overfitting. RFs performed somewhat worse with $\sim 90\%$ NCV testing accuracy and much higher training accuracy, suggesting that the RF models were overfitted. NNs and SVMs were carried forward for Dorabella classification.

Predictions for the classification of the Dorabella cryptogram were similar for the final NN and SVM (Table 2), with both assigning the most probable classification to English Corpus text (probability ≥ 0.75), and the least probable classification to the English-Like Random text (probability ≤ 0.01).

Statistic	**RF**	**NN**	**SVM**
NCV Testing Accuracy % (SD)	90.4 (3.6)	93.8 (1.8)	94.0 (1.5)
NCV Training Accuracy % (SD)	99.1 (1.2)	95.0 (0.4)	95.3 (0.3)
Difference (pp)	8.7	1.2	1.3

Table 1: Nested cross-validation results for machine learning models.

Type	**NN**	**SVM**
English Corpus	0.75	0.81
Typewriter-Like Random	0.24	0.07
Uniform Random	0.01	0.11
English-Like Random	0.00	0.01

Table 2: Predicted probabilities for the Dorabella cryptogram by machine learning models.

4 Discussion

Comporting with the previous study, the present study reifies the use of machine learning models to classify text in historical cryptography, and found that the Dorabella cryptogram resembles meaningful English text more closely than gibberish types of text.

However, this result should be interpreted with caution. First, despite being more comprehensive than the previous binary dataset, the four text categories explored in this study do not capture all possible explanations, for example, other encryption schemes. Second, of the four text types in the present study, the final NN and SVM models were least accurate in classifying English Corpus text (though still with accuracies $> 90\%$).

This study also explored the concept of a 'textspace', i.e. a two-dimensional feature space for representing the characteristics of types of text related to historical cryptography. Results suggest that the textspace is helpful for determining the separability of the space, though the location of the Dorabella cryptogram so close to the boundary between English Corpus and English-Like Random - despite the machine learning models assigning very low probabilities to an English-Like Random classification - is not intuitive. This highlights the limitations of dimension reduction techniques.

Future work should find features that better separate meaningful English from English-like gibberish, apply these methods to other putative cryptograms, and explore other encryption schemes.

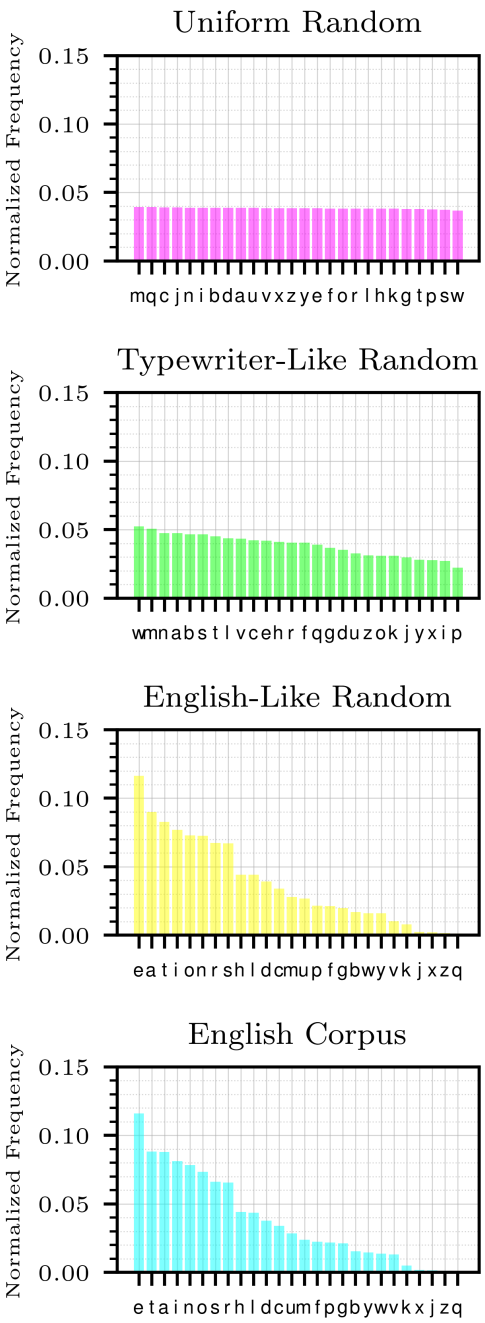


Figure 1: Letter/unigram frequency distributions for each text type.

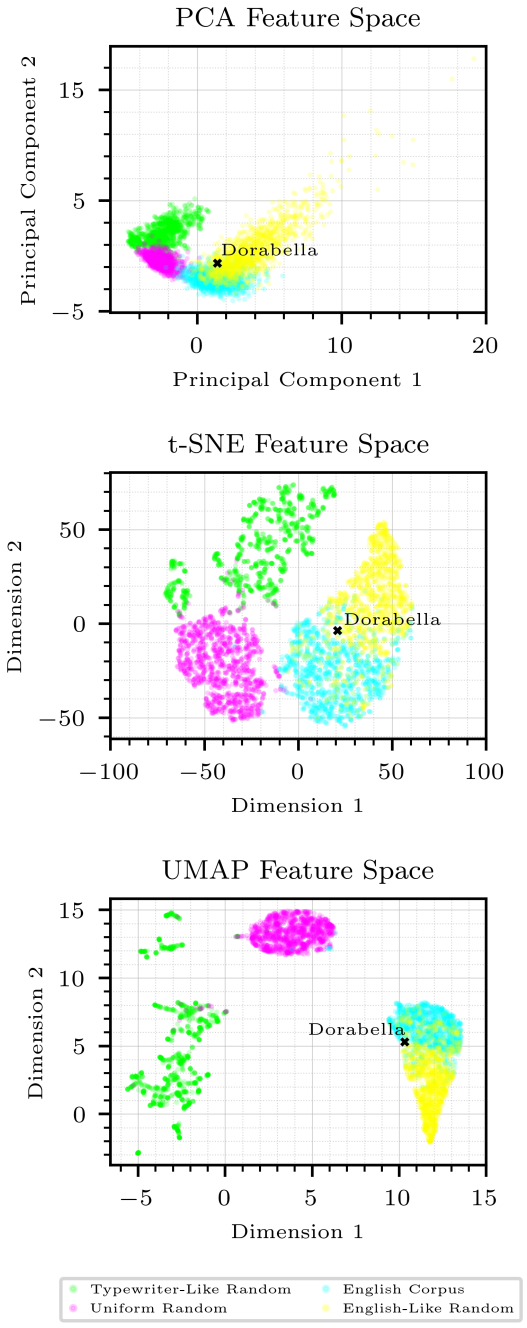


Figure 2: Dimension reduction of the original 15 features to two-dimensional feature space.

References

L. Breiman. 2001. Random Forests. *Machine Learning*, 45:5–32. https://doi.org/10.1023/A:1010933404324

C. Cortes and V. Vapnik. 1995. Support-Vector Networks. *Machine Learning*, 20:273–297. https://doi.org/10.1007/BF00994018

M. Davies. 2015. The Wikipedia Corpus. https://www.english-corpora.org/wiki/

F. Foxon. 2022. A Treatise on the Rilke Cryptogram. *Cryptologia*, 47(6):493–510. https://doi.org/10.1080/01611194.2022.2092784

F. Foxon. 2024. Artificial Neural Network for Hoax Cryptogram Identification. *Proceedings of the 7th International Conference on Historical Cryptology, HistoCrypt*, pages 86–90. https://doi.org/10.58009/aere-perennius0094

H. Hotelling. 1933. Analysis of a Complex of Statistical Variables into Principal Components. *Journal of Educational Psychology*, 24:417–441. https://psycnet.apa.org/doi/10.1037/h0071325

J. D. Hunter. 2007. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3):90–95. https://doi.org/10.1109/MCSE.2007.55

L. McInnes and J. Healy. 2018. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv*, 1802.03426. https://doi.org/10.48550/arXiv.1802.03426

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. https://dl.acm.org/doi/10.5555/1953048.2078195

F. Rosenblatt. 1958. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological review*, 65(6):386–408. https://psycnet.apa.org/doi/10.1037/h0042519

L. van der Maaten and G. Hinton. 2008. Visualizing Data Using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605. http://jmlr.org/papers/v9/vandermaaten08a.html

Dutch Cryptanalysis of Four American Diplomatic Codes in World War I

Florentijn van Kampen
iHub, Radboud University
Nijmegen – The Netherlands
florentijn.vankampen@ru.nl

Abstract

During the First World War, the Netherlands carefully maintained a neutral position. To guard this neutrality, the Dutch authorities monitored the activities of the belligerent surrounding countries. International telecommunications via telephone and telegraph were closely monitored and censored by censor bureaus.

In 2019, the Dutch intelligence and security service released a dossier about these censor bureaus to the Dutch National Archive. In that dossier, a previously unknown history of two groups of pioneering codebreakers based at the censor bureaus in Amsterdam and Rotterdam, was uncovered. In 2024, a first publication appeared about this dossier, with particular emphasis on how the local staff successfully broke German codes.

Additionally, the Dutch codebreakers successfully broke four American diplomatic codes between June and December 1918. This breakthrough enabled Dutch intelligence to monitor secret diplomatic traffic between American officials in the Netherlands and Washington during and after World War I.

This paper examines the systematic cryptanalysis of U.S. Department of State communications by Dutch codebreakers. Through analysis of original documents and surviving codebooks, it identifies the compromised diplomatic codes and places these findings in a broader historical perspective.

1 Introduction

During World War I (WWI) the Netherlands served as a hub for spies, smugglers, and diplomats. As a neutral country with a stable government, it provided an environment where people could discreetly conduct their activities. The Dutch government sought to maintain an overview of these activities and installed so-called censor bureaus at various telephone and telegraph offices to monitor and censor unwelcome communications.

The censor bureaus in Amsterdam and Rotterdam were particularly well placed because they served as international telecommunication hubs. The local military staff had full access to multiple international diplomatic communication channels, including coded communications. These two groups started analysing and breaking German codes and soon they were achieving cryptanalytical successes on a par with their more widely known and much larger counterparts in France, England and the United States (Jacobs and van Kampen, 2024b).

While it is well documented that British cryptanalysts successfully decoded American diplomatic telegrams during this period (Larsen, 2017) (Zegart, 2022), the fact that at the end of WWI, Dutch codebreakers had fully compromised American diplomatic traffic is as yet unknown. This article will show how the Dutch codebreakers were able - without international cooperation - to intercept, analyse and solve the American diplomatic codes in use at that time.

This article begins with an introduction to American diplomatic codes in WWI, followed by the methodology section. Section 4 provides an overview of the recently disclosed Dutch archive material on American Diplomatic Codes. Section 5 will successively present the four American diplomatic codes, how they were used, how the Dutch analysed and broke them and which historical sources were used to verify this. Section 6 will conclude this article with a number of observations and conclusions.

Proceedings of the 8th International Conference on Historical Cryptology, pages 65-75
Poznań, Poland, 16-18 June, 2025

2 American Diplomatic Codes in World War I

The United States has a long history of codes and ciphers protecting sensitive military and diplomatic communications. The standard work on United States Diplomatic Codes and Ciphers (Weber, 1979), traces this history all the way back to 1775 when Charles William Frederic Dumas, one of America's first secret agents, designed and dispatched the first revolutionary secret diplomatic cipher to Benjamin Franklin. It masked the correspondence between the Continental Congress and its foreign agents in Europe.

2.1 Codes and Ciphers

The U.S. Department of State protected its diplomatic communications using a system called "The Cipher of the Department of State". In cryptology, however, it is common to make a distinction between "codes" and "ciphers" because they refer to different cryptographic techniques. It is therefore important to explore these terms in a bit more detail. For this, we use the terminology used by David Kahn (Kahn, 1996).

A *cipher* operates on one, or sometimes more, plaintext letters. A so-called "substitution cipher" replaces these plaintext letters with enciphered equivalents. In the case of a "transposition cipher", the order of the plaintext letters is changed in some way. In both cases, this process is performed in a pre-arranged way, referred to as the cipher key.

In the case of a *code*, words, expressions or names are replaced by *code equivalents*, in most cases, a series of digits. These series of digits are called codegroups. The system that describes which word or expression should be replaced by which series of digits is called a a codebook. Codebooks may also include elements like syllables, full sentences, punctuation, or grammatical instructions.

So, the "cipher" of the U.S. Department of State is really a code using a one-part codebook. It is called a one-part codebook because both encoding and decoding are performed using a single alphabetical ordered list, an example of which can been seen Figure 2. Note the characteristic property that plaintext words and phrases that are alphabetically close to each also have codenumbers and codewords that are close to each other. This is unlike a two-part codebook where the words and expressions are not in alphabetical order, but in random order. Therefore, these codebooks need two parts, one part for encoding with the plaintext entries in alphabetical order and one part for decoding with the codegroups in numerical order. This article will focus on the various editions of this cipher that were in use by the U.S. Department of State between 1914 and 1918. The origin of these books go back to 1876 and start with the adoption of standardised communication rules of the telegraph.

2.2 The Cipher of the U.S. Department of State

The adoption in 1870 of the five character group, numbers and letters, as the standard "word" to be used in telegraph communications served as a basis for the diplomatic code of 1876, designed by by John. H. Haswell, Chief of the Bureau of Indexes and Archives (Hove, 2011). Cost and efficiency for telegraph transmission was an important design criterion for this new code, and now, every English word or phrase would translate to a five digit codenumber, or one standard "word" in telegraph terms.

The first version of this new "Cipher of the Department of State" was introduced in 1876 and was commonly referred to as "The Red Cipher" because of the color of its cover. Updated versions of this codebook would be introduced in the years ahead, also designated by colors: The Blue Cipher in 1899, The Green Cipher in 1910, the Gray Cipher in 1918 and the Brown Cipher in 1938.

This article is about the four diplomatic codes that were intercepted and solved by the Dutch codebreakers: the Cipher of U.S. Department of State in the editions of 1876, 1899, 1910 and 1918.

3 Methodology

As a neutral nation, the Dutch codebreakers worked in international isolation and had no knowledge of American codebooks, colors or systems. They created their own classification system for organizing the codes that they intercepted and analysed. However, to understand the historical significance of their work, it is necessary to determine which American codes the Dutch actually broke and therefore to link the Dutch classification system to the official American code-naming system. For each of the four American codes, this paper will present the results in two steps.

The first step consists of the relevant information from the Dutch archive material with a focus on key insights into cryptanalytic methods, breakthrough moments and technical observations.

The second step involves the validation of all available observations against original, historical American codebooks. For each of these validations, a specific example will be presented alongside its corresponding entry from the original codebook.

Each section on a specific code will provided a reference to the specific historical sources and their location. This is both to provide historical evidence for the the Dutch codebreaking results but also to facilitate possible further research.

4 Dutch cryptanalysis of American codes

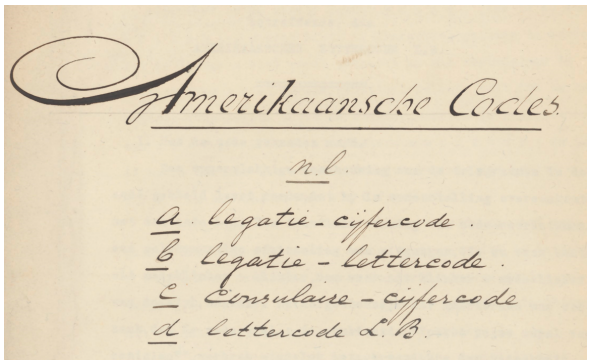


Figure 1: Title page of the Dutch collection of "notes" on American Codes (*Amerikaansche Codes* in Dutch). Source: Dutch National Archives.

The archive material of the Dutch codebreakers consists of two kinds of documents: "Reports" describing the day-to-day business of the censorship bureaus and "Notes" summarising the final cryptanalytical results of the codebreakers. The report[1] from Captain Nyweide, station chief in the Rotterdam telegraph office, describes the work on American codes. This report covers the timeframe between September 6, 1917 and September 30, 1918. On page 9 captain Nyweide writes[2]:

> Now that an English code had finally been broken[3], the American cipher telegrams were examined with even greater attention, and Lieutenant Vis of the Amsterdam bureau succeeded on June 14, 1918, in uncovering the American legation code, and on August 7 of the same year, he discovered the consular number code. Meanwhile, at the beginning of July, Lieutenant Berenschot managed to break an American letter code, to which Reserve Lieutenant de Vries subsequently dedicated all his attention and investigative skills. This code has also progressed to such a stage that the first telegrams could be submitted.[4]

The Dutch archive material includes a file[5] that contains four separate notes that deal with the topic of American codes. The title page of this file is shown in Figure 1. Not all of its contents has survived. The notes that are present are, in order of the American codebook versions that they cover.

- *"The Consular Number Code"*, Amsterdam, August 9, 1918, two pages. This note turns out to be about the solution of the Red Cipher of 1876.
- *"How the code was found"*, Amsterdam, June 18, 1918, two pages. As we will see below, this note is about the solution of the Blue Cipher of 1899 .
- *"Concerning the American Word Code"*, Rotterdam July 19, 1918, five pages. This note is about the solution of the Green Cipher of 1910.
- *"Concerning the American Letter Code B"*, Rotterdam , December 4 1918, five pages. This note is about the solution of the Gray Cipher of 1918.

In the section below, we will now explore the Dutch observations and analysis and link their results to the canonical names of U.S. Department of State codebooks.

5 The solution of four diplomatic codes

5.1 The Red Cipher of 1876

The codebook's central component consisted of almost 1000 pages of alphabetically ordered

[1] Dutch National Archives, 2.13.70 - 1939

[2] This translation from Dutch to English and all the subsequent translations from Dutch are by the author.

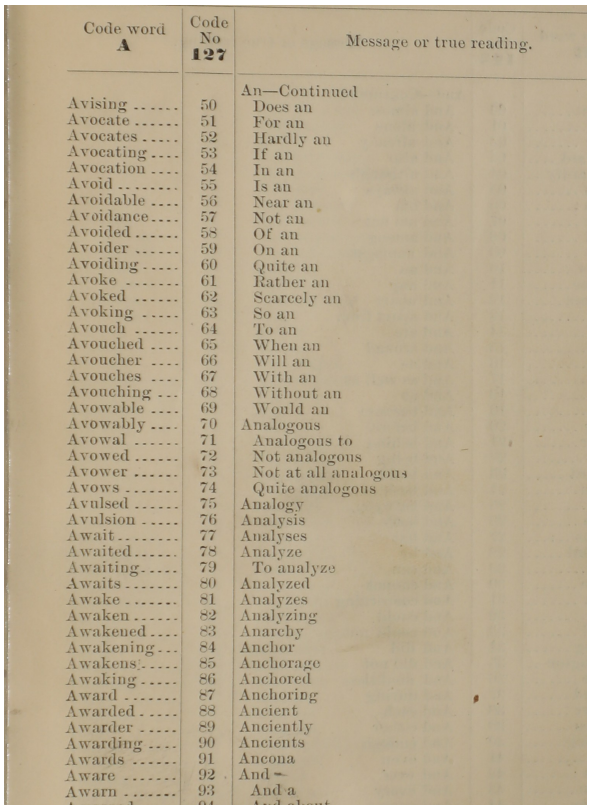
[3] Unfortunately, no information has survived about these results on English codes.

[4] Submitted here means: sent to the headquarters in The Hague to be used for intelligence purposes.

[5] Dutch National Archives, 2.13.70 - 1943

words and phrases. Each entry provided a choice between two encoding methods: A plaintext word or phrase could either be replaced with a code number or with a code word.

For example, the word "And" that is used in one of the archived diplomatic messages[6] is found on page 127 with number 92, as shown in Figure 2. The information from line 92 shows that the plaintext word "And" could be encoded with the codenumber "12792" or with the codeword "Aware". From the Dutch archives we can observe that most diplomatic communications that the Dutch intercepted from U.S. diplomats using this codebook was encoded with *codenumbers*.



Code word A	Code No 127	Message or true reading.
		An—Continued
Avising	50	Does an
Avocate	51	For an
Avocates	52	Hardly an
Avocating	53	If an
Avocation	54	In an
Avoid	55	Is an
Avoidable	56	Near an
Avoidance	57	Not an
Avoided	58	Of an
Avoider	59	On an
Avoiding	60	Quite an
Avoke	61	Rather an
Avoked	62	Scarcely an
Avoking	63	So an
Avouch	64	To an
Avouched	65	When an
Avoucher	66	Will an
Avouches	67	With an
Avouching	68	Without an
Avowable	69	Would an
Avowably	70	Analogous
Avowal	71	Analogous to
Avowed	72	Not analogous
Avower	73	Not at all analogous
Avows	74	Quite analogous
Avulsed	75	Analogy
Avulsion	76	Analysis
Await	77	Analyses
Awaited	78	Analyze
Awaiting	79	To analyze
Awaits	80	Analyzed
Awake	81	Analyzes
Awaken	82	Analyzing
Awakened	83	Anarchy
Awakening	84	Anchor
Awakens	85	Anchorage
Awaking	86	Anchored
Award	87	Anchoring
Awarded	88	Ancient
Awarder	89	Anciently
Awarding	90	Ancients
Awards	91	Ancona
Aware	92	And —
Awarn	93	And a

Figure 2: Original Red Cipher codebook, page 127. Entry 92 is "And". Source: the National Cryptologic Museum.

5.1.1 Dutch solution: Consulary Number Code

The Dutch codebreaker Lieutenant Vis wrote about this codebook in his note "about the Consular Number Code" dated, Amsterdam, August 9, 1918. His observations were brief and concise:

- This consular number code is alphabetical, ranging from 10000 "A" until 57371 "your".
- This code is used for the correspondence between the consulates

The Dutch considered the solution of this American code important, because of "America's increasing influence in European affairs". They noted that the representatives of the United States both in the Netherlands and in the Dutch Colonies found it impossible to maintain regular contact[7] with their principles in Washington. For time critical communication, the Americans had no choice but to use telegram or telephone communications, which were inherently more susceptible to interception.

The note includes four examples of intercepted encoded telegrams labelled "A, B, C, D" shown in Figure 3. The handwritten words above the codenumbers show some of the words that were recovered by the Dutch cryptanalyst: 12792 = "and", 31008 = "from", 57015 = "work" and 57371 = "your". The result for the word "And" can be verified in Figure 2.

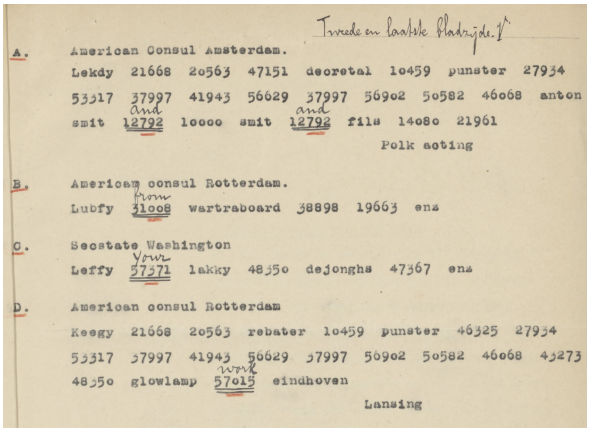


Figure 3: Analysis of the Consulary number code with Dutch handwritten annotations. Source: Dutch National Archive.

Careful examination of these telegrams show that each of these messages starts with a character sequence that is not related to the Red Cipher of 1876 (`Ledky`, `Lubfy`, `Leffy`, `Kecgy`, `Lakky`). It appears that these telegrams use an encoding system for dating telegrams which was designed in September 1900 and was first included in the Green Cipher of 1910 (Weber, 1979, page 247). This mix of codenumbers from 1876 and timestamps from 1910 is possible because these message were sent in 1918 after both editions were

[6] Telegram A in section 5.1.1

[7] This means by postal services.

Month	Day		Hour
	Tens	Units	
B January	O 0	B 1	B 1
C February	A 1	C 2	C 2
D March	E 2	D 3	D 3
F April	U 3	F 4	F 4
G May		G 5	G 5
K June		K 6	K 6
L July		L 7	L 7
M August		M 8	M 8
P September		P 9	P 9
R October		R 0	R 10
S November			S 11
V December			V 12
			i = forenoon noon y = afternoon midnight

Table 1: Date encoding system.

published. Apparently, these systems could be combined in daily usage. This date encoding system is shown in Table 1.

Since these four telegrams are the only examples of intercepted coded American diplomatic messages in the Dutch archives, it is worthwhile to demonstrate the decoding process and present the underlying plaintext messages. Using the date system from Table 1 for decoding the dates and the original American Red Cipher codebook from 1876 for the codenumbers and a few codewords, we can now decode the messages[8]:

Telegram A
American Consul Amsterdam `LEKDY 21668 20563 47151 DECRETAL 10459 PUNSTER 27934 53317 37997 41943 56629 37997 56902 50582 46068` *Anton Smit* `12792 10000` *Smit* `12792` *fils* `14080 21961 Polk acting`
July 26 3 pm: Confer colleagues report connections may not be accepted recommendations enemy Trading List or white list with specific reasons Anton Smit and A Smit and fils are they connected Polk[9] acting

Telegram B
American consul Rotterdam `LUBFY 31008 wartraboard`[10] `38898 19663` enz[11]
July 31 4pm: from War Trade Board you may certify etc.

Telegram C
Secstate Washington `LEFFY 57371 LAKKY 48350` *dejonghs* `47367` enz
July 24 4pm: your July 16 6pm : S de Jonghs reputation etc.

Telegram D
American consul Rotterdam `KECGY 21668 20563 REBATER 10459 PUNSTER 46325 27934 53317 37997 41943 56629 37997 56902 50582 46068 43273 48350` *glowlamp* `57015` *eindhoven* Lansing[12]
June 22 5pm: confer colleagues report recommendations enemy trading list or white list with specific reasons Philips[13] glowlamp works Eindhoven Lansing

5.1.2 Sources

Two sources for the original Red Cipher codebook of 1876 were used for this section. There is one copy in the collection of the National Cryptologic Museum. That version was used as the source for the image in Figure 2 and to decode the telegram messages. Another copy is available on microfilm in the U.S. National Archives and on-line[14].

5.2 The Blue Cipher of 1899

After more than 20 years of use, Haswell thought it was time to update the Red Cipher of 1876. He wrote that "(...) since European Black Chambers[15] operated freely, and probably possessed copies of the U.S. Department of State 1876 Cipher, it would seem good policy, after a service of twenty-five years to establish a new and improved system of telegraphic communication between the Department and its diplomatic Agents" (Weber, 2013).

This new version of codebook saw the light in 1899 and became known as the Blue Book[16]. The design and format was very much the same as the previous version, although the new version was expanded from 1,200 to 1,500 pages (Weber, 2013).

5.2.1 Dutch solution: Legation Number Code

The Dutch name for this system can only be deduced from the title page of the Dutch archive material of "notes" on American codes as seen in Figure 1. There are two number codes, the consulary

[8] The codenumbers and codewords are in `typewriter` font and uppercase, the inline plaintext is *Italic*.

[9] Frank Lyon Polk was counsellor of the U.S. Department of State. `https://en.wikipedia.org/wiki/Frank_Polk`

[10] Wartraboard = abbreviation for War Trade Board.

[11] enz is the Dutch version of etc (*enzovoort*)

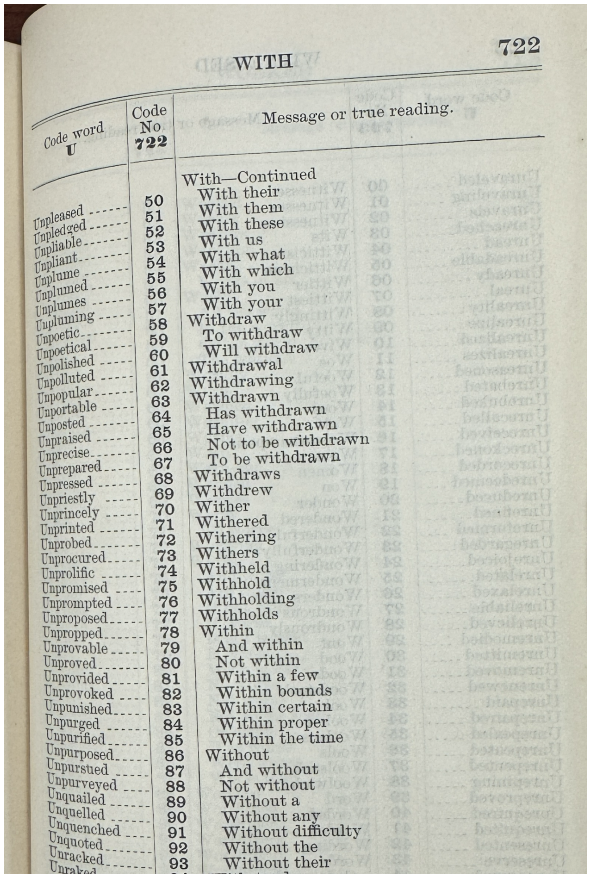
[12] Robert Lansing was the Secretary of State `https://en.wikipedia.org/wiki/Robert_Lansing`

[13] Philips is a well known Dutch electronics company that used to produce Lightbulbs

[14] For this article, a scan of the microfilm was requested. As a result, this scan is now available on-line for everybody to use: `https://catalog.archives.gov/id/417267710`

[15] See (de Leeuw, 2014) on the history of European codebreaking efforts organized in so called *Black Chambers*

[16] Although this codebook is often referred to as "The Blue Book", this paragraph was called "Blue Cipher" for the sake of consistency



WITH 722

Code word U	Code No. 722	Message or true reading.
		With—Continued
Unpleased	50	With their
Unpledged	51	With them
Unpliable	52	With these
Unpliant	53	With us
Unplume	54	With what
Unplumed	55	With which
Unplumes	56	With you
Unpluming	57	With your
Unpoetic	58	Withdraw
Unpoetical	59	To withdraw
Unpolished	60	Will withdraw
Unpolluted	61	Withdrawal
Unpopular	62	Withdrawing
Unportable	63	Withdrawn
Unposted	64	Has withdrawn
Unpraised	65	Have withdrawn
Unprecise	66	Not to be withdrawn
Unprepared	67	To be withdrawn
Unpressed	68	Withdraws
Unpriestly	69	Withdrew
Unprincely	70	Wither
Unprinted	71	Withered
Unprobed	72	Withering
Unprocured	73	Withers
Unprolific	74	Withheld
Unpromised	75	Withhold
Unprompted	76	Withholding
Unproposed	77	Withholds
Unpropped	78	Within
Unprovable	79	And within
Unproved	80	Not within
Unprovided	81	Within a few
Unprovoked	82	Within bounds
Unpunished	83	Within certain
Unpurged	84	Within proper
Unpurified	85	Within the time
Unpurposed	86	Without
Unpursued	87	And without
Unpurveyed	88	Not without
Unquailed	89	Without a
Unquelled	90	Without any
Unquenched	91	Without difficulty
Unquoted	92	Without the
Unracked	93	Without their

Figure 4: Original Blue Cipher codebook, page 722. Entry 86 is "without". Source: the National Cryptologic Museum.

number code and the legation number code. Since the consulary number code is the Red Cipher of 1876 and since the Blue Cipher of 1899 is the only other number code and since this code is in use by the American legation in The Hague, the Dutch probably referred to the Blue Cipher as the "Legation Number Code".

The proof that the Dutch codebreakers solved the Blue Cipher can be deduced from Lieutenant Vis's note: "How the code was found". His text takes us back to June 12, 1918. He explains how a sharp observation about two separate transmissions initiated the solution of this codebook. The first transmission, encoded in the Blue Cipher, is from Robert Lansing, the U.S. Secretary of State, to the American Legation in the Hague. The second transmission, in plain text, is from the Dutch representative in Washington, sent to the Dutch ministry of Foreign Affairs (a channel the Dutch codebreakers apparently also monitored!).

> On Wednesday afternoon, June 12, around 2 p.m., a long coded telegram addressed to "Amlegation[17] the Hague" from Lansing passed through the telegraph office in Amsterdam. In it, a few ship names appeared in plain language. From this it was possible to deduce the traditional word "Dampfer", this time "steamship".
>
> Two hours later, an approximately equally long telegram to "Celer[18] Haag" from the Dutch representative in Washington passed through. The same names also appeared in this telegram. Upon comparison after transmission, both telegrams turned out to be almost identical[19], allowing more than three hundred words to be recorded.
>
> The code found is in five-digit form and appears to be purely alphabetical: between 10425 "A" and 72286 "without" are the alphabetically lowest and highest groups found. Geographical names lie within this range. Nothing can yet be reported regarding numbers, letter groups, and personal names.
>
> Work on deciphering earlier telegrams can begin immediately, and due to the regularity and the experience gained with irregular codes, results can be obtained relatively quickly.

The notes of Lieutenant Vis show that this new code started with the number 10425 for the letter "A" and went all the way up to 72286 for "Without". Both these observation can be confirmed with the original codebook from 1899. The latter one can be seen in Figure 4. It shows page 722 of the Blue Cipher codebook with entry 86 for "without". This leads to a codenumber of 72286 in accordance with the analysis of the Dutch codebreakers.

5.2.2 Sources

There is one copy of the original Blue Cipher codebook of 1899 in the collection of the National Cryptologic Museum. That version was used as

[17] Short for "American Legation"

[18] The word "Celer" in the signature of telegrams represents the postal address of the Dutch Ministry of Foreign Affairs.

[19] This somewhat unusual situation could have been caused by American and Dutch officials each quoting the same newspaper.

the source for the image in Figure 4 and to verify the results obtained from the Dutch archive.

5.3 The Green Cipher of 1910

The U.S. Department of State published its next code, the Green Cipher, in 1910. This 1418 page volume introduced a completely new code design for enhanced security. In the absence of original codebook scans, this example relies on the information provided by (Weber, 1979, page 246). The pages of the Green Cipher codebook are divided into 5 columns:

Bab 102	Bab	102	Bab	A
Aa	ba	00	ab	a
Aachen	ca	01	ac	–bad
Aal	da	02	ad	–badge
Aalborg	fa	03	af	–band
Aam	ga	04	ag	– – of

In this example the page number is 102 and the header of this page shows the stem "Bab". That means that all the codewords for this page start with "Bab". The coding system makes a distinction between plaintext words and plaintext phrases. The codeword for *Aalborg* would be "Babfa" while the codeword for *A band* would be "Babaf". Careful observation shows that plaintext words are encoded with *stem* + *consonant* + *vowel* and plaintext phrases are encoded with *stem* + *vowel* + *consonant*. The notes left in the archive show that the Dutch codebreakers took particular professional pride to have noticed this structure.

5.3.1 Dutch solution: Legation Letter Code A

Lieutenant Berenschot published his analysis of the Green Cipher on July 19, 1918. In his note from the Rotterdam bureau he identified this code as "The American Word Code" as can been seen in Figure 5. However, the December 4, 1918, note discussing Letter Code B (examined in the next section on the Gray Cipher) refers to this code as "Letter Code A". This shows how the Dutch sometimes struggled with their own nomenclature of intercepted codes. With hindsight, the choice of "Legation Letter Code A", in combination with the subsequent "Letter Code B", actually makes much more sense because of the similarity between the Green code (Letter Code A) and the Gray code (Letter Code B).

Berenschot writes that the code consists of pronounceable words of five letters where every code-

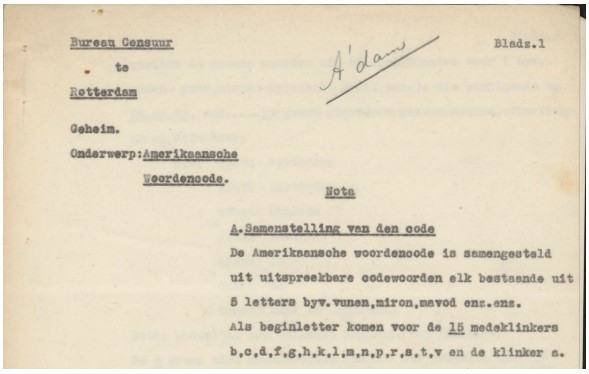

Bureau Censuur Bladz.1
te
Rotterdam
A'dam
Geheim.
Onderwerp:Amerikaansche
Woordencode.
Nota
A.Samenstelling van den code
De Amerikaansche woordencode is samengesteld
uit uitspreekbare codewoorden elk bestaande uit
5 letters byv.vunen,miron,mavod enz.enz.
Als beginletter komen voor de 15 medeklinkers
b,c,d,f,g,h,k,l,m,n,p,r,s,t,v en de klinker a.

Figure 5: Heading of the note *"Concering the American Word Code*. This copy was written in Rotterdam and was probably intended to inform and instruct the Amsterdam ("A'dam") bureau. Source: Dutch National Archives.

word is made of a "stem" of three letters and a suffix of two letters. Here we clearly see the basic structure of the codebook as described above. Berenschot gives a description of the methods and techniques he used to solve this code. The Dutch archival materials hold particular value because they preserve the codebreakers' actual thoughts and analytical processes. Therefore, the analysis is presented below as much as possible in Berenschot's words.

Analysis of message start

All telegrams sent on the same day begin with codewords that share the same root but have varying endings. These roots progress alphabetically from one day to the next. It could therefore be safely assumed that the first codeword represents the date of the telegram, particularly because American plaintext telegrams always start this way.

In American telegrams, the second codeword immediately following the date often includes the codewords `miron` and `vunen`, which are always followed by two new words starting with the letter "a". Since the "a" only appears as an initial letter in this situation, and since the construction of these codewords starting with the "a" deviates from the rest, it was deduced that these codewords represent numbers, especially because telegrams often begin with:

"Your / My (telegram) `number number`".

Assuming the simplest interpretation, the code is alphabetical, where `miron` means "My" and `vunen` means "Your" and the two codewords start-

ing with "a" represents "number number".

Telegrams Between The Hague and London

Many telegrams between The Hague and the American Embassy in London had similar message headers. The start of these messages, the message headers, were very similar in structure:

From	To	Header
The Hague	London	`fakaw rospo fibof fucku`
The Hague	London	`fakaw godmy fibof fucku`
London	The Hague	`fakaw fucku fibof rospo`

Analysis of known American number codes revealed that Sheldon and Frothingham were American envoys in London and that Edwards served in The Hague. This led to the following conclusion:

`fakaw`	for
`fibof`	from
`fucku`	Edwards
`godmy`	Frothingham
`rospo`	Sheldon

It was concluded that codewords ending with a vowel (like the "o", the "y" and the "u" in this example), represent personal names, while those ending with consonants represent common words (e.g., `miron` and `vunen`). Later, it was also discovered that these codes represent place names and abbreviations, for example: `pystu = Rotterdam`. (See table 2 for more examples).

Washington Messages

Plaintext telegrams were sent almost daily by Garrett, the American envoy in The Hague, to Washington to request visas for specific individuals. These telegrams always began with "Asks visa", followed by a name, and often include the phrase "nationality parentage Dutch".

The opening lines of the coded messages sent to Washington were now carefully compared with these plaintext telegrams in the hope that such visa application matters were also submitted in code. Having similar messages in both plaintext and code would greatly facilitate the solution of codewords. This hope was indeed fulfilled. There were quite a few telegrams intended for Washington that began with `cudam tisuk` followed by some words from the supplementary code (personal names) and in which the expression `mitel nipaf gigep` appeared. So that led to:

`cudam`	asks
`gigep`	Dutch
`mitel`	nationality
`nipaf`	parentage
`tisuk`	visa

This analysis revealed that the code for common words was indeed alphabetical.

Frequently Used Words

Common words like "for, from, to, and" had multiple codewords that were spread throughout the codebook:

`hugys, fakaw, filuc`	for
`hekyd, fibof, fadin`	from
`degyl, nuvog, somet`	to
`bykef, myses`	and

And with this, First Lieutenant Berenschot ended his report on the analysis of the Green Cipher. An entry from August 9, 1918 in the report indicates that the Dutch codebreakers were able to decode all Green Cipher messages.

5.3.2 Sources

The information on the Green Cipher of 1910 used in this sections comes from a special source. The U.S. National Archives have in their collection a file of recaptured material from the Germans after World War II. Apparently, the Germans had in their possession a copy of the Green Cipher as reconstructed by English codebreakers during WWI[20].

Table 2 shows a comparison of letter codes A (Green Cipher) and B (Gray Cipher). We can see in this table that for the Green Cipher the word "bank" corresponds to the codeword "cylub". This particular example can be seen in figure 6 that shows the page for codewords starting with "cyl".

5.4 The Gray Cipher of 1918

The U.S. Department of State introduced the Gray Cipher on March 22, 1918. At 1,173 pages, this volume was more compact than its 1910 predecessor. The Department designated this code for all consular telegrams, with specific instructions to exclude verbatim quotations from newspapers or government sources. This last instruction makes

[20] NR 3922 ZEMA121 36655A 19410000 Cryptographic Codes and Ciphers: United Kingdom Code Books for B3, Green Cipher, T and Sonder List B 1 Systems Recaptured from Germans in WWII. `https://catalog.archives.gov/id/2810671`

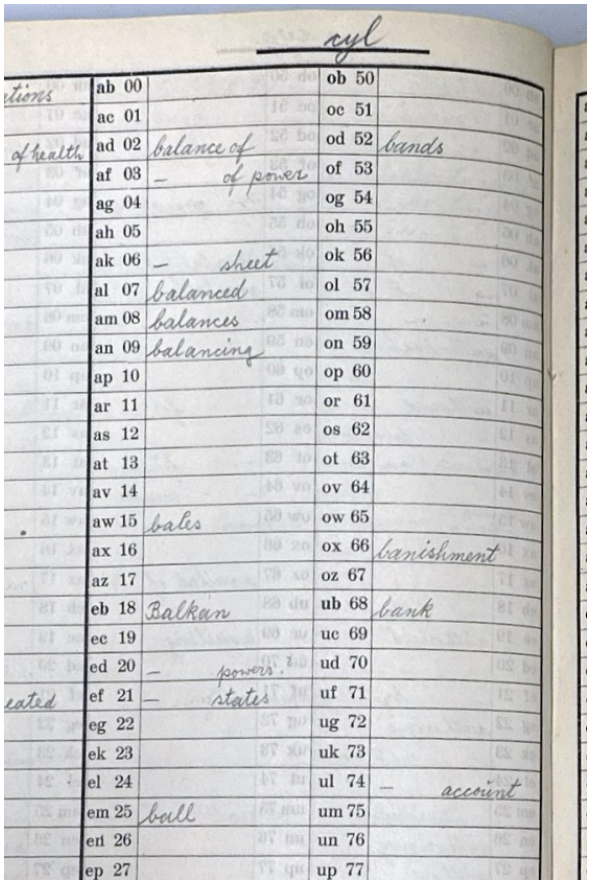


Figure 6: Inside page of the reconstructed version of the Green Cipher by the British codebreakers with “cylub” as “bank”. Source: U.S. National Archives.

a lot of sense because if the interceptor would know the verbatim underlying plaintext, for example from the latest newspaper, the solution of the codewords would be trivial. The design of the Gray Cipher was very similar to that of the Green Cipher (Weber, 1979). The well known U.S. Cryptographer William F. Friedman describes, somewhere between 1940 and 1946, the properties of the Gray Code as follows: ”This system has no confidential character whatever [sic] and is used for economy purposes” (Friedman, 1940).

5.4.1 Dutch solution: Legation Letter Code B

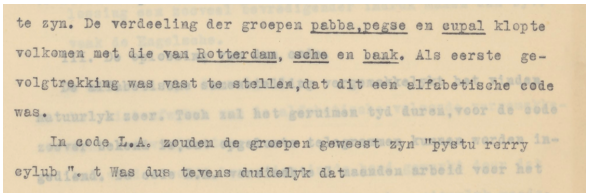
Reserve First Lieutenant de Vries from the Rotterdam bureau published his note on Letter Code B, the Gray Cipher, on December 4, 1918, only nine months after this new codebook was published. This date, falling several weeks after the November 11, 1918, armistice that ended WWI, demonstrates that Dutch intelligence continued monitoring and decoding American diplomatic traffic even after the war’s conclusion. This might very well be related to the fact that the Dutch were keen to follow the developments after the war in preparation for the peace negotiations in Versaille (Moeyes, 2014). Historical sources from the same time show that the Dutch codebreakers were breaking German codes to follow specifically the communications between the Americans and the Germans. (Jacobs and van Kampen, 2024a)

In the following section, First Lieutenant de Vries describes how he solved Letter Code B.

How the solution was found

A superficial examination of the telegrams written in this code already reveals that its structure matches that of Letter Code A (the Green Cipher). The identical formation of codeword stems and the similar alternation of endings — sometimes consonant / vowel, and at other times vowel / consonant — are immediately noticeable. The carelessness of the Americans was the reason the code was broken: the usage of the predictable sequence `pabba pegse cupal` was the breakthrough that would unravel the otherwise meaningless jumble of letters.

Captain Boomsma’s speculative remark that this sequence might mean “Rotterdamsche Bank” (Bank of Rotterdam) turned out to be spot on. The division of the groups `pabba`, `pegse`, and `cupal` perfectly aligned with the pattern of Rotterdam’s bank names. The first conclusion drawn was that this was an alphabetical code.


te zyn. De verdeeling der groepen pabba,pegse en cupal klopte volkomen met die van Rotterdam, sche en bank. Als eerste gevolgtrekking was vast te stellen,dat dit een alfabetische code was.

In code L.A. zouden de groepen geweest zyn "pystu rorry cylub ". t Was dus tevens duidelyk dat

Figure 7: Excerpt from the note about Legation Letter Code B showing the “De Rotterdamsche Bank” hypothesis in both Letter Code A and B. Source: Dutch National Archives.

Table 2 shows the Dutch phrase “Rotterdamsche Bank” or “Bank of Rotterdam” in both Letter Code A and B. This comparison shows that Letter Code B uses fewer letter groups than Letter Code A. The stem `pys` in Letter Code A corresponds with `pab` in Letter Code B and `cyl` corresponds to `cup`, and so on. The note contains a complete mapping between the stems of both let-

Word	L'code A	L'code B
Rotterdam	**pys**tu	**pab**ba[21]
sche	**rer**ry	**peg**se
bank	**cyl**ub[22]	**cup**al

Table 2: Words in Letter Code A and B

ter codes, part of which can be seen in Figure 8.

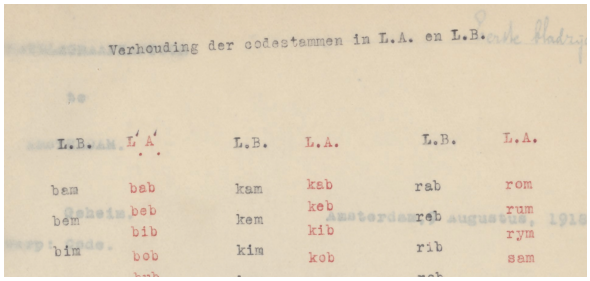


Figure 8: Mapping between stems for Letter Code A and B ("L.A." and "L.B."). Source: Dutch National Archive

5.4.2 Sources

Access to the original 1918 codebook that was used in this section was obtained through the Georgetown University Library. This library is home to the Russel J. Bowen collection with more than 16,000 titles including a large number of photocopied articles. The Bowen books cover a range of topics from military intelligence and spycraft all the way through spy fiction. Part of this collection is a unique original copy of the 1918 Gray Cipher codebook[23], as can be seen Figure 9.

6 Conclusions

Between June and December 1918, the Dutch codebreakers from Amsterdam and Rotterdam successfully broke all four diplomatic codes from the U.S. Department of State that were in use at that time: The Red Cipher of 1876, The Blue Cipher of 1899, The Green Cipher of 1910 and the Gray Cipher of 1918.

The fact that these different codes, with a similar design, were in use at the same time was actually an advantage for the Dutch. It was a deliberate choice by the U.S. Department of state as can be seen in introduction of the 1918 Gray Cipher

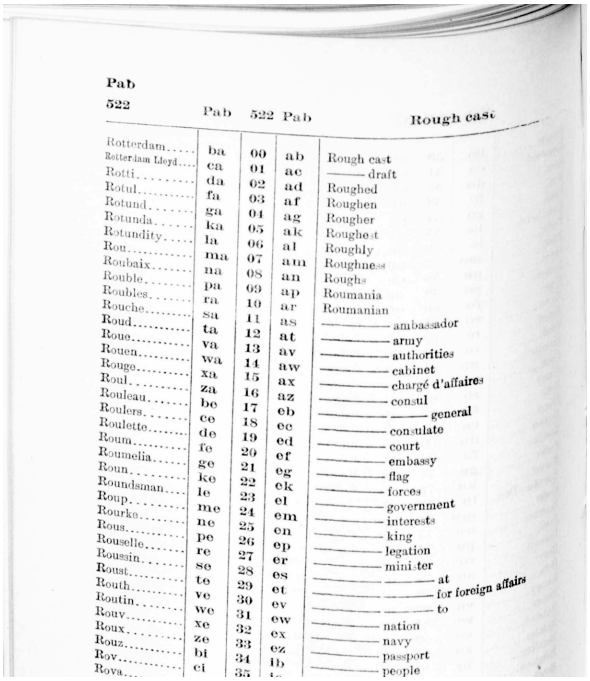


Figure 9: The original Gray Cipher codeook with the page for "Pabba" Rotterdam. Source: Georgetown University Library

codebook. Page XII reads: "This cipher (meaning the Gray Cipher) should never be used for the transmission of verbatim messages from individuals or from officials of a foreign Government. (...) Verbatim messages will be transmitted only in the Consular Red Code, The Diplomatic Blue, or such other codes as may be designated for this purpose by departmental instructions."

But the biggest advantage for the Dutch was the fact that the Americans did not use any advanced security measures on the channels the Dutch monitored, like super encipherment or similar techniques that were known at the time. The American officials were using straightforward, alphabetical one-part codebooks, greatly facilitating the Dutch in their codebreaking efforts.

In the note about the Blue Cipher, written on June 18, 1918, the author mentions a request for copies of coded telegrams dating all the way back to January 1, 1914. This is a striking historic example of the modern signals intelligence strategy of "store now, decrypt later".

As a result, by 1918, the Dutch had accumulated substantial knowledge of confidential and secret American diplomatic communications, both during and immediately after the war.

[22] See Figure 9 for source example

[22] See Figure 6 for source example

[23] https://wrlc-gu.primo.exlibrisgroup.com/permalink/01WRLC_GUNIV/1ok4lbs/alma991001599509704111

Acknowledgements

This research would not have been possible without access to original American historical codebooks. Special thanks are due to the following individuals who made extraordinary efforts to retrieve historical codebooks from special storage facilities and provide scans and images: Molly Kamph, Research Rooms & Augmented Processing Branch, National Archives at College Park; Jay Silvestre, Curator of Rare Books, Georgetown University Library and Robert J. Simpson, Museum Archivist and Librarian, National Cryptologic Museum, Maryland. Their efforts are highly appreciated. Finally, I would like to thank Prof. Dr. Bart Jacobs for his valuable ideas and comments on this article.

References

Karl de Leeuw. 2014. Books, Science, and the Rise of the Black Chambers in Early Modern Europe. *Geheime Post: Kryptologie und Steganographie der diplomatischen Korrespondenz europaïscher Höfe während der Frühen Neuzeit*, pages 87–89.

William F. Friedman. 1940. Communications Systems in Use by the Department of State. *NSA Friedman documents - Folder 467 - nr A67345*.

Mark T. Hove. 2011. *History of the Bureau of Diplomatic Security of the United States Department of State*. U.S. Dept. of State, Bureau of Diplomatic Security, Global Publishing Solutions.

Bart Jacobs and Florentijn van Kampen. 2024a. Geheimschrijverij bij de Meijer. In *Uiterst vertrouwelijk, Achter de schermen van de Nederlandse geheime diensten*, pages 127–135. Querido Facto.

Bart Jacobs and Florentijn van Kampen. 2024b. A new perspective on Dutch WWI codebreaking with its international ramifications. *Proceedings of the 7th International Conference on Historical Cryptology (HistoCrypt 2024)*.

David Kahn. 1996. *The Codebreakers : The Story of Secret Writing. Revised edition*. New York: Scribner.

Daniel Larsen. 2017. British codebreaking and American diplomatic telegrams, 1914–1915. *Intelligence and National Security*, 32(2):256–263.

Paul Moeyes. 2014. *Buiten Schot, Nederland Tijdens de Eerste Wereldoorlog*. de Arbeiderspers.

Ralph E. Weber. 1979. *United States Diplomatic Codes & Ciphers 1775-1938*. Transaction Publishers.

Ralph E. Weber. 2013. *Masked Dispatches: Cryptograms and Cryptology in American History, 1775–1900*. Center for Cryptologic History, National Security Agency.

Amy B. Zegart. 2022. *Spies, Lies, and Algorithms: The History and Future of American Intelligence*. Princeton University Press.

Overview of Ciphers Used by the Czechoslovak "Maffie"

Jozef Krajčovič
Ministry of Justice
Bratislava
Slovakia
kryptosvet@gmail.com

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

Abstract

This paper provides an overview of encryption systems and steganographic techniques used by the Czechoslovak Maffie, which was an anti-Austrian underground resistance organization in 1914-1918.

1 Introduction

Austria-Hungary (1867 – 1918) was a constitutional monarchy and great power in Central Europe and basically a multi-ethnic state entity. Czechs and Slovaks were two of the nations that made up Austria-Hungary. They felt oppressed and their voice were often not heard in the administration of public affairs. The *Maffie* was an underground organization of Czech politicians and leaders formed in Prague during World War I (1914 – 18) and an anti-Austrian resistance organization that engaged in intelligence activities.

In this work, we studied the cryptographic and steganographic techniques used by the representatives of Czechoslovak Maffie. In section 2 we shortly describe the historical background of the organization. Next, in Section 3 we shortly describe the used ciphers and steganographic techniques, and in Section 4 we bring insight into the biggest affair of the resistance called "Button Affair", which is also related to cryptology.

2 Secret Political Organisation Called "Maffie"

Maffie[1] was founded after the emigration of Tomáš Garrigue Masaryk in 1914 by Czech politician Edvard Beneš, who later became the second president of Czechoslovakia, and other (mainly) anti-royalists[2] (Hlaváč, 2008). During the years of the Great World War, it was a central part of the First Czechoslovak Resistance. Its main objective was to overthrow the Emperor of Austria and to cause the disintegration of his country. Their goal was to tirelessly seek all kinds of information of a political, economic, and military nature, and deliver it to Masaryk abroad in a conspiratorial way.

The *Military Maffia*[3] was an armed component of the Czech domestic anti-Austrian resistance in 1914-1918 and was part of a broader movement that led to the establishment of the Czechoslovak Republic. Its main task was to undermine the morale of the Austro-Hungarian army, organize sabotage actions, and prepare for an armed uprising. The Military Maffia played a key role in the preparations for the coup of 28 October 1918, which led to the establishment of the Czechoslovak Republic (Hálek, 2018).

3 Overview of Used Ciphers and Steganographic Techniques

The Czechoslovak Maffie, as well as other secret societies, used various steganographic techniques and a wide range of cryptographic methods for their communication as: hand ciphers, codes, and encryption devices.

One of the simplest methods used[4] was a simple substitution using a field postcard (the so-called "feldpostkarta") with irregularly cut out windows. This card served as an encryption table, also a second card was used with the alphabet. The encryption process was based on changing the relative position of the window part and the alphabet part (Valenta, 1993).

A classical Vigenère cipher (adapted to the

[1] The name 'Maffie' was only a nickname of the group, and has no real connection or reference to the Sicilian crime group called Mafia (Paulová, 1937; Hlaváč, 2008).

[2] Karel Kramář, Alois Rašín, Josef Scheiner and their "boss", Přemysl Šámal.

[3] The main figures of the Military Mafia were František Sis, Vladimír Slavík, Jindřich Čapek and Jaroslav Rošický.

[4] For example used by Karel Locher, a member of the Maffie resistance organization in charge of liaison with the Poles and later served in various political and journalistic positions.

Proceedings of the 8th International Conference on Historical Cryptology, pages 76-79
Poznań, Poland, 16-18 June, 2025

Czech alphabet) was the main cipher used by the organization (Werstadt, 1931), also see Figure 1. This encryption method was, however, easily broken by the Austrian secret service Evidenzburo, which allowed them to monitor communications between the domestic resistance and the emigration (Paulová, 1939). After discovering that the modified Vigenère cipher had apparently been broken, a new, more robust encryption method was selected, based on a reworded alphabet, longer passwords, and shorter blocks of text. The cipher was available in two forms: a ciphering device named *Kryptograf*[5] and its paper version called the "autoclave ruler" according to Josef J. Frič (Kučera, 2003). The cipher was based on the Vigenére autokey.

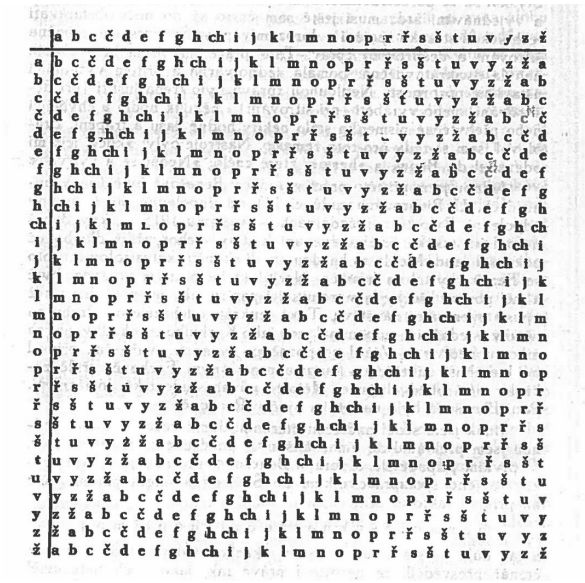


Figure 1: Modified Vigenère / Tabula Recta (Werstadt, 1931)

A different (simple) encryption device was found in (Butyrskij et. al., 2018), however, we were unable to identify the machine (see Figure 2).

At least two different codes were used, one to hide the names of the resistance members in the communication (cover names, see Figure 3) and one classical code book for encryption. The second one encrypts words and phrases to another words and phrases and was marked as *Allegoric Code* (orig. Allegorický kod Československé zahraniční revoluce, see Figure 4).

[5]Created by P. J. Baráček. This ciphering device could be a predecessor of the Condenser PBJ cipher machine, see (Antal et. al., 2024).

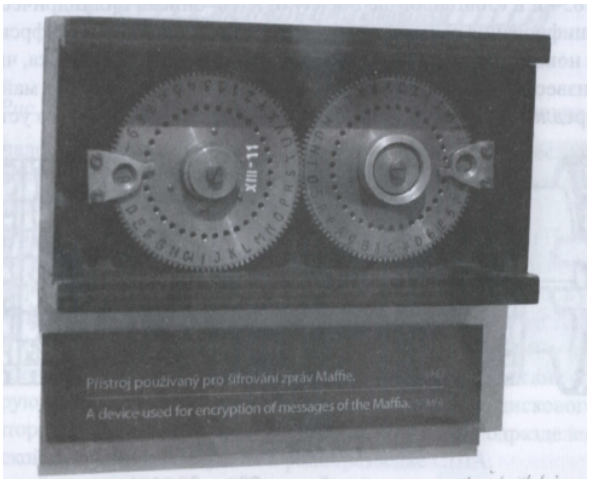


Figure 2: Unidentified encryption device allegedly used by Maffie's spies (Butyrskij et. al., 2018)

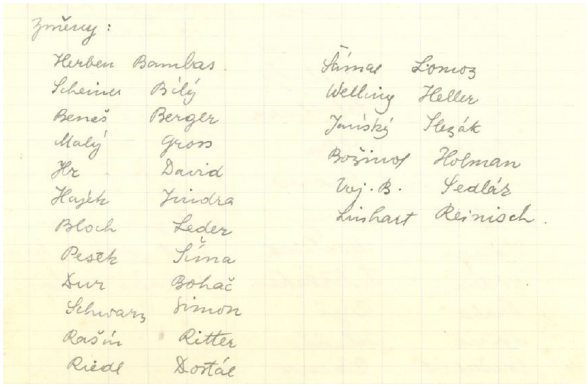


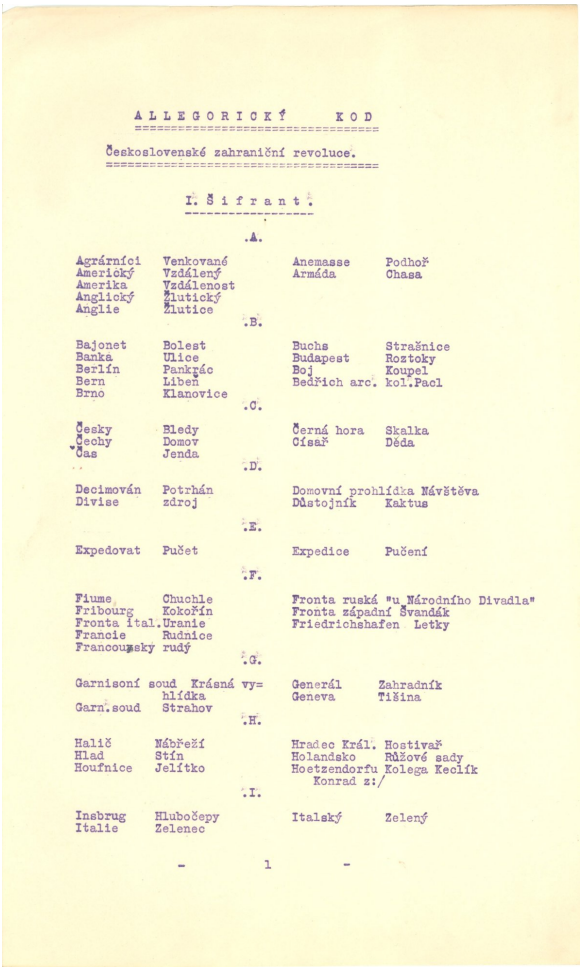
Figure 3: List of cover names of the members of Maffie

Various steganographic techniques were used by the resistance (Stanek, 1931; Očenášek, 1991; Kučera, 2003):

- Invisible ink - developed by Dr. Vladimír Staněk a prominent Czech chemist and scientist. He was the administrator of the chemical department of the Sugar Research Station (see Figure 5).

- Based on a geographical map of Switzerland, different cantons meant different fronts. The order of the letters in the text of the postcard signified an assessment of the overall situation (politically and militarily).

- Two identical metal plates with holes that looked like a sieve. Each hole corresponds to one letter of the alphabet according to a secret key.



ALLEGORICKÝ KOD

Československé zahraniční revoluce.

I. Šifrant.

.A.

Agrárníci	Venkovaný	Anemasse	Podhoř
Americký	Vzdálený	Armáda	Chasa
Amerika	Vzdálenost		
Anglický	Žlutický		
Anglie	Žlutice		

.B.

Bajonet	Bolest	Buchs	Strašnice
Banka	Ulice	Budapest	Roztoky
Berlín	Pankrác	Boj	Koupel
Bern	Liben	Bedřich arc.	kol.Pacl
Brno	Klanovice		

.C.

Česky	Bledy	Černá hora	Skalka
Čechy	Domov	Císař	Děda
Čas	Jenda		

.D.

Decimován	Potrhán	Domovní prohlídka	Návštěva
Divise	zdroj	Důstojník	Kaktus

.E.

Expedovat	Pučet	Expedice	Pučení

.F.

Fiume	Chuchle	Fronta ruská	"u Národního Divadla"
Fribourg	Kokořín	Fronta západní	Švandák
Fronta ital.	Uranie	Friedrichshafen	Letky
Francie	Rudnice		
Francouzsky	rudý		

.G.

Garnisoní soud	Krásná vyhlídka	Generál	Zahradník
		Geneva	Tišina
Garn.soud	Strahov		

.H.

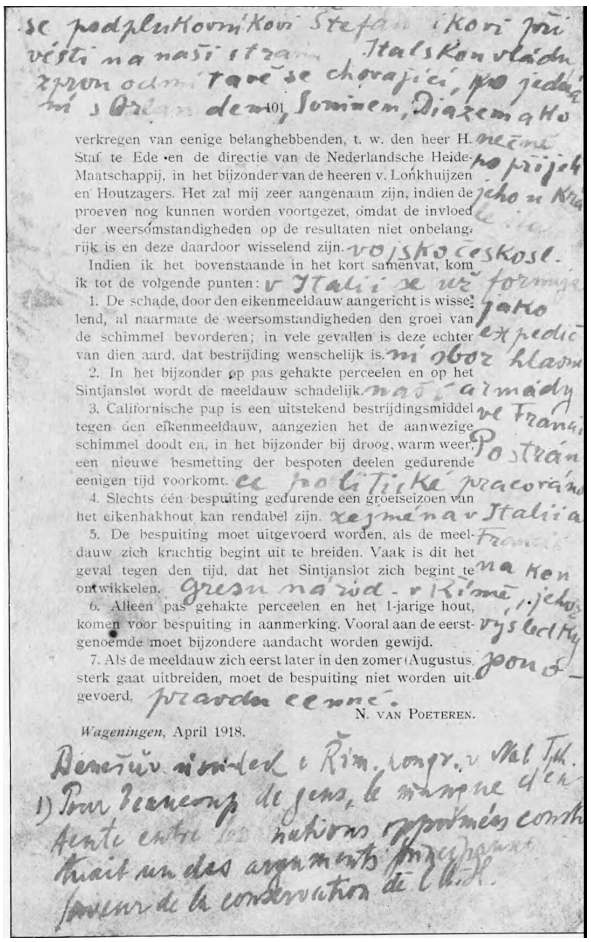
Halič	Nábřeží	Hradec Král.	Hostivař
Hlad	Stín	Holandsko	Růžové sady
Houfnice	Jelítko	Hoetzendorfu	Kolega Kecalík
		Konrad z:/	

.I.

Insbrug	Hlubočepy	Italský	Zelený
Italie	Zelenec		

- 1 -

Figure 4: First page of the Allegoric Code (VHA, 1915)



401

verkregen van eenige belanghebbenden, t. w. den heer H. Staf te Ede en de directie van de Nederlandsche Heide-Maatschappij, in het bijzonder van de heeren v. Lonkhuijzen en Houtzagers. Het zal mij zeer aangenaam zijn, indien de proeven nog kunnen worden voortgezet, omdat de invloed der weersomstandigheden op de resultaten niet onbelangrijk is en deze daardoor wisselend zijn.

Indien ik het bovenstaande in het kort samenvat, kom ik tot de volgende punten:

1. De schade, door den eikenmeeldauw aangericht is wisselend, al naarmate de weersomstandigheden den groei van de schimmel bevorderen; in vele gevallen is deze echter van dien aard, dat bestrijding wenschelijk is.
2. In het bijzonder op pas gehakte perceelen en op het Sintjanslot wordt de meeldauw schadelijk.
3. Californische pap is een uitstekend bestrijdingsmiddel tegen den eikenmeeldauw, aangezien het de aanwezige schimmel doodt en, in het bijzonder bij droog, warm weer, een nieuwe besmetting der bespoten deelen gedurende eenigen tijd voorkomt.
4. Slechts één bespuiting gedurende een groeiseizoen van het eikenhakhout kan rendabel zijn.
5. De bespuiting moet uitgevoerd worden, als de meeldauw zich krachtig begint uit te breiden. Vaak is dit het geval tegen den tijd, dat het Sintjanslot zich begint te ontwikkelen.
6. Alleen pas gehakte perceelen en het 1-jarige hout, komen voor bespuiting in aanmerking. Vooral aan de eerstgenoemde moet bijzondere aandacht worden gewijd.
7. Als de meeldauw zich eerst later in den zomer (Augustus, sterk gaat uitbreiden, moet de bespuiting niet worden uitgevoerd.

N. VAN POETEREN.

Wageningen, April 1918.

Figure 5: Magazine "Prager Zuckermarkt" with a steganographic message written with Dr. Staněk's original invisible ink (Stanek, 1931)

- Newspaper advertisements were used to confirm the successful delivery of encrypted messages.

- Furthermore, a special case/luggage/shoes with double walls and hidden compartments were used to carry the messages.

A combination of encryption and steganography was used in a network in Genoa, Italy, to transmit confidential information (Kučera, 2003). It consisted of three layers. The message was first encrypted, then written with invisible ink, and concealed in scrolls or other inconspicuous objects. The sent messages passed through the hands of the Austrian censors, but thanks to the precision of the encryption, their contents remained undetected (Paulová, 1939).

4 The "Button Affair"

A very famous (espionage) affair of the Czechoslovak resistance was the so-called *Button Affair*. It was one of the biggest blows during the war when the Czechoslovak resistance was fighting against Austria-Hungary. The affair had serious consequences for the Czechoslovak resistance, including the arrest of many prominent figures. The case became well-known and was discussed after the establishment of Czechoslovakia, becoming the subject of political controversy.

We shortly describe the events of the affair. In September 1915, Edvard Beneš and Jan Kyjovský decided to send Aloysia (Lola) Linhart to a specific mission. As a courier, she was responsible to deliver a secret message (see Figure 6) from Switzerland to Prague. She hid the message in a button on her dress, which was considered a safe way of transporting the information. Lin-

hart successfully crossed the border and arrived in Prague, where she was to deliver the message. However, due to a conspiratorial error and the incompetence[6] of some members of the Resistance, and the report fell into the hands of the wrong people, leading to the exposure and arrest of several people. (Paulová, 1937; Paulová, 1939; Lukeš, 2016)

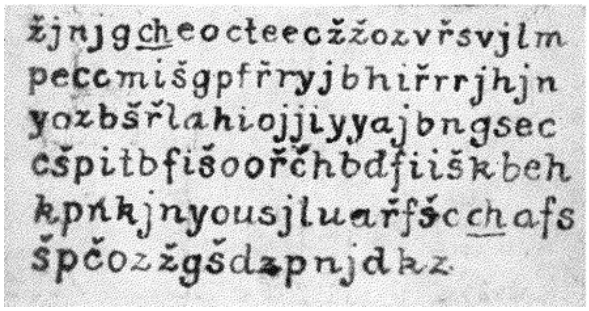


Figure 6: Encrypted message from Aloysia (Lola) Linhart's button (Pichlík, 1991)

The message was encrypted using the modified Vigenère cipher (adapted to the Czech alphabet). The Austrian military authorities had already deciphered the key to the Maffian alphabet during the affair, which gave them access to sensitive information and allowed them to intervene more effectively against the resistance (Kárník, 1995).

Acknowledgments

This work was supported by grant VEGA 2/0054/24.

References

Eugen Antal, Paul Reuvers and Pavol Zajac. 2024. The Condenser PBJ cipher machine. Cryptologia. https://doi.org/10.1080/01611194.2024.2372254.

L.S. Butyrsky, D.A. Larin, G.P. Shankin 2018. *Kriptograficheskiy front Velikoy Otechestvennoy.*. Moskva, Gelios ISBN: 978-5-85438-220-5.

Jan Hálek. 2018. Vojenská Maffie. K pramenum a interpretaci ozbrojené složky české domácí protirakouské rezistence. In *Historie a vojenství*, pages 4 - 13. Vojenský historický ústav, 2018.

[6]In Prague, Aloysia (Lola) Linhart contacted Bohumil Mareš, who was to deliver the message onwards. Mareš tried to pass the message on to the editors of Čas, but he was unsuccessful. Then he contacted the deputy František Soukup, who knew nothing about the mission and refused to accept the report. Mareš's suspicious behavior led Soukup to inform the police. The police arrested Mareš, Linhart, and others involved, including the editors of Čas.

Miroslav Hlaváč. 2008. *Čeští mafiáni 1914-1918*. Paris, Karviná ISBN 978-80-87173-00-8.

Zdeněk Kárník. 1995. Edvard Beneš a Maffie v nejtěžším období Velké války. Řídila politická emigrace domácí odboj? (červenec 1917 – říjen 1918). In *Historie a vojenství*, pages 3 - 22. Vojenský historický ústav, 1995.

Martin Kučera. 2003. *Zprávy tajným inkoustem: kapitola z dějin českého zpravodajství za první světové války (1914-1918)*. Masarykův ústav AV ČR, Praha ISBN 80-86495-18-3.

Michal Lukeš. 2016. *Špionážní případy. První republika*. Academia, Praha ISBN 978-80-200-2529-6.

Ludvík Očenášek. 1919. *Na pomoc dohodě: tajná činnost několika českých vlastenců za války*. Tiskem Pražské akciové tiskárny, nákl. vlastním, 1919.

Karel Pacner. 2002. *Československo ve zvláštních službách: pohledy do historie československých výzvědných služeb 1914-1989*. Themis, Praha ISBN 80-7312-002-X.

Milada Paulová. 1937. *Dějiny Maffie: odboj Čechů a Jihoslovanů za světové války 1914-1918, I.díl*. Československá grafická Unie, Praha.

Milada Paulová. 1939. *Dějiny Maffie: odboj Čechů a Jihoslovanů za světové války 1914-1918, I.díl, 2.část*. Československá grafická Unie, Praha.

Karel Pichlík. 1991. *Bez legend.*. Praha, Panorama ISBN 80-7038-217-1.

Vladimír Staněk. 1931. Z maffiánské laboratoře. *Naše revoluce VII. pp. 59-79.*

Jaroslav Valenta. 1934. Český Wallenrod. (Politický důvěrník české Maffie Karel Locher v Krakově v r. 1918.). *Slovanské historické studie. 19, (1993,) pp. 50-70.*

Jaroslav Werstadt. 1931. Ze vzpomínek Starckových. *Naše revoluce VII. pp. 210-222.*

VHA 1915. Alegorický kod československé zahraniční revoluce. Spojovací klíče. *ČSNR V PAŘÍŽI III-5 – (1915) 1916-1918.* Inv. n. 5464 Karton varia

Antonio Elio "Cipher" and his Polyphonic-Syllabic Cipher

George Lasry
The DECRYPT Project
george.lasry@gmail.com

Marcello Simonetta
The Medici Archive Project
marcello.simonetta@gmail.com

Norbert Biermann
Universität der Künste Berlin
mail@norbertbiermann.de

Abstract

Antonio Elio (Helius) (1506–1576) was a Roman Catholic prelate who served as Bishop of Capodistria and Pola and Titular Patriarch of Jerusalem. Also a prolific cryptographer in the service of Pope Paul III, he is credited for the invention of polyphonic ciphers. In this article, we provide an overview of his career and work in cryptography and describe an ingenious polyphonic-syllabic cipher he designed. Although several matching plaintext-ciphertext segments were available, reconstructing the cipher key required a significant and unusual amount of time, underscoring the cipher's high level of security. Ciphertext-only cryptanalysis for such a cipher would be extremely difficult and nearly impossible, even with modern computing, without prior knowledge of the principles of its complex design.

1. Biography

In April 1535, Bernardo Boccarini, the secretary to the papal nuncio in France, Rodolfo Pio da Carpi, wrote to a colleague a facetious letter in which he complained about the fatigue of his own profession and, while sketching the portrait of the "perfect secretary," he asked him to send nice words to "Antonio" so that he would not perceive all the curses ("cancari") he had in store against him for having created his own set of ciphers that made him sweat so much.[1] By then, this nearly twenty-year-old man had already created a reputation for himself so that he was universally known as "Antonio delle Zifre."

Antonio was born in Capodistria[2] in the first decade of the 16th century in a noble family.[3] He gave early signs of his brilliance and was recommended to his compatriot Aurelio Vergerio, the resident cryptographer ("a secretis") in Clement VII's curia. After Aurelio's tragic death in 1532, Antonio took over his job and started his own career as a prelate and diplomat. His first known letter of May 1535 was addressed to Aurelio's brother, Pier Paolo Vergerio, then a papal nuncio in Vienna at the court of Ferdinand I Habsburg, and by 1536, bishop of Capodistria. Over the next few years, the relationship between the two competing curial officers became sour over some unpaid debts, and eventually, their breakup became very public and violent.

By 1549, Vergerio left Italy and his bishopric, turning on the Protestant side, while Elio became bishop of Pola[4] in 1548. This shift reflected the trust that he had earned working loyally for the Farnese family under Pope Paul III. After the pope's death, for a few months, Antonio entertained the idea of moving to Florence at the service of Duke Cosimo I de' Medici, but he decided to stay loyal to the Farnese, even when the new pope, Julius III del Monte, waged war against Parma. Elio remained in the papal curia also under Paul IV Carafa and the following pontiffs, who always highly valued his special services.

One particularly appreciated aspect of Elio's skills was his ability to break enemies' codes. For instance, we learn from Medici correspondence that the Carafas systematically intercepted all outgoing diplomatic dispatches and submitted the ciphered samples to Elio. Bongianni Gianfigliazzi, the Florentine ambassador in Rome, wrote to his duke in June 1558 that one informant told him that "*the aforementioned monsignor had deciphered many letters of*

[1] Bernardo Boccarini to Trifon Benci, Rouen, 4 April 1535 (*Lettere facete*, D. Atanagi ed., 1561, I, 349).
[2] Koper, in today's Slovenia.
[3] For an introductory biography, see L. Byatt, https://www.treccani.it/enciclopedia/antonio-elio_(Dizionario-Biografico)/
[4] Pula, Croatia.

Proceedings of the 8th International Conference on Historical Cryptology, pages 80-89
Poznań, Poland, 16-18 June, 2025

ambassadors, and in this genre it is an awesome thing that he does, since he saw him sometimes be handed a cipher, shut himself in a room alone, and in a short time break it. This achievement appears miraculous."[5]

However, it is unclear whether anyone managed to break Elio's ciphers. Antonio invented a new kind of encoding that, even when known, still requires a high degree of concentration and imagination to be fully decoded.

Antonio Elio, during his long service in the papal curia (1532-1572), laid the ground for the establishment of a professional cipher service, and his work significantly influenced further generations of cipher secretaries. He trained Giovanni Battista Argenti before he became the pope's cipher secretary (Meister 1906, p.55), and in his treatise on cryptography, Matteo Argenti, Giovanni's nephew and successor, recognizes Elio's contribution (Meister 1906, p.50, p.161).

2. Antonio Elio's innovative cipher - the Challenge

We found a document partially in cipher that exemplifies Antonio Elio's innovative contributions to cryptography. This is a batch of letters sent by Filiberto Ferrerio, Bishop of Ivrea and Apostolic Nuncio in France, between 1537 and 1540. The letters we consider were written from Paris in late October 1540, and they all bear the unmistakable cryptographic marks of Elio's work for the Farnese chancery.

This document from the Vatican Archives (AA.Arm. XVIII. 6532) contains several segments of interlinear deciphered text. As part of the analysis of this letter and the cipher employed to encode it, we needed to reconstruct the original key. As shown in Figure 1, the enciphered parts consist of a mix of graphical cipher symbols with Latin letters. Our initial assumption was that the cipher was homophonic, with each alphabet letter represented by one or more cipher symbols and other cipher symbols representing elements of a nomenclature, such as words or proper names. We identified approximately 40 distinct cipher symbols, which usually hints at a simple cipher with a small number of homophones per letter of the alphabet. Some symbols have dots on top, and we also needed to determine their role in the enciphering and deciphering process.

Based on our experience extracting a cipher key from matching ciphertext-plaintext symbols, we expected that this analysis would take no more than an hour or two. However, it took two full days of intensive work to make initial inroads into the cipher and begin to understand its principles. Numerous additional hours were required to identify most of the elements of the cipher key. Before proceeding to our detailed analysis of the cipher and its principles in Section 4, we invite the curious reader to experiment and attempt to recover the key from the samples in Figures 1, 2, and 3 below. The ciphered passages are underlined.

[5] "*detto monsignor di quelle delli Imbasciatori ne ha deciferate assai*, et che in questo genere è cosa grande quello che fa che l'ha visto qualche volta havere una cifra, et riserarsi in camera da per sè, et in poco tempo ritrovarla et la par' cosa miracolosa." (Bongianni Gianfigliazzi to duke Cosimo, Rome, 10 June 1558 (Archivio di Stato di Firenze, Mediceo del Principato, f. 3278, 90)

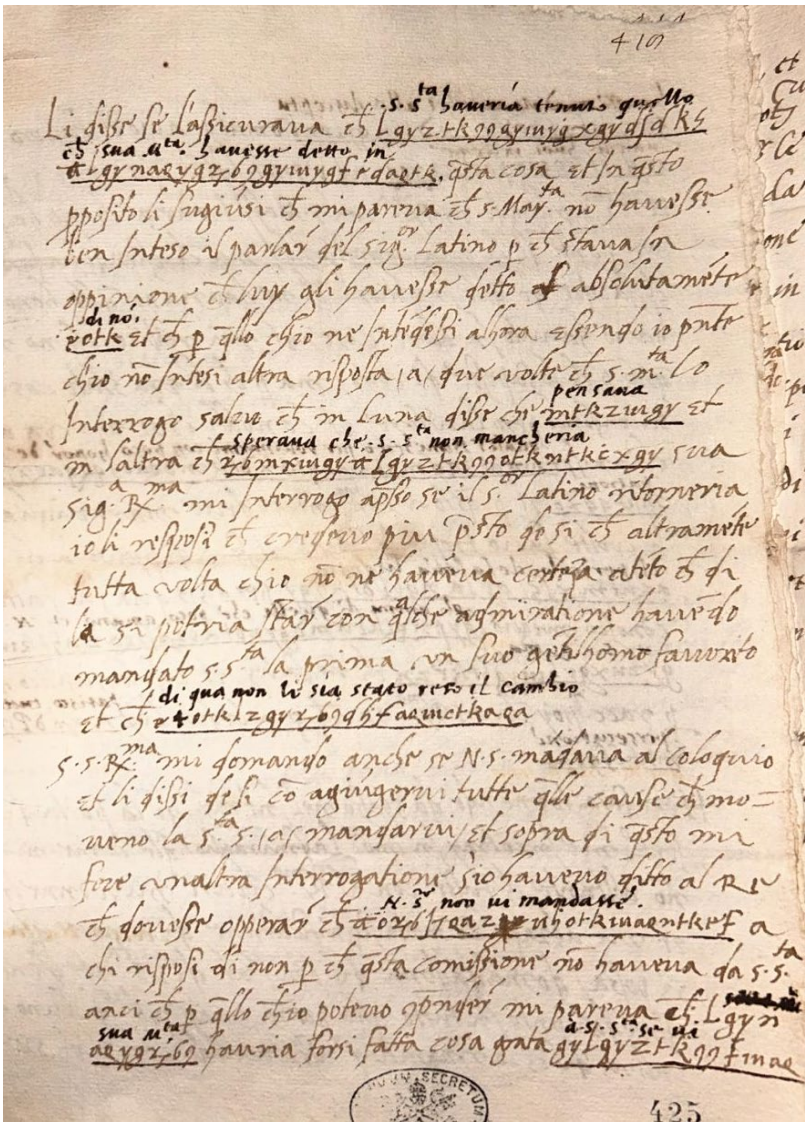


Figure 1 - First part

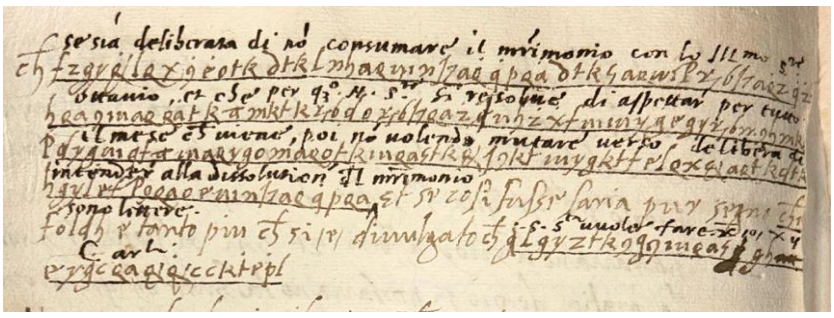


Figure 2 - Second part

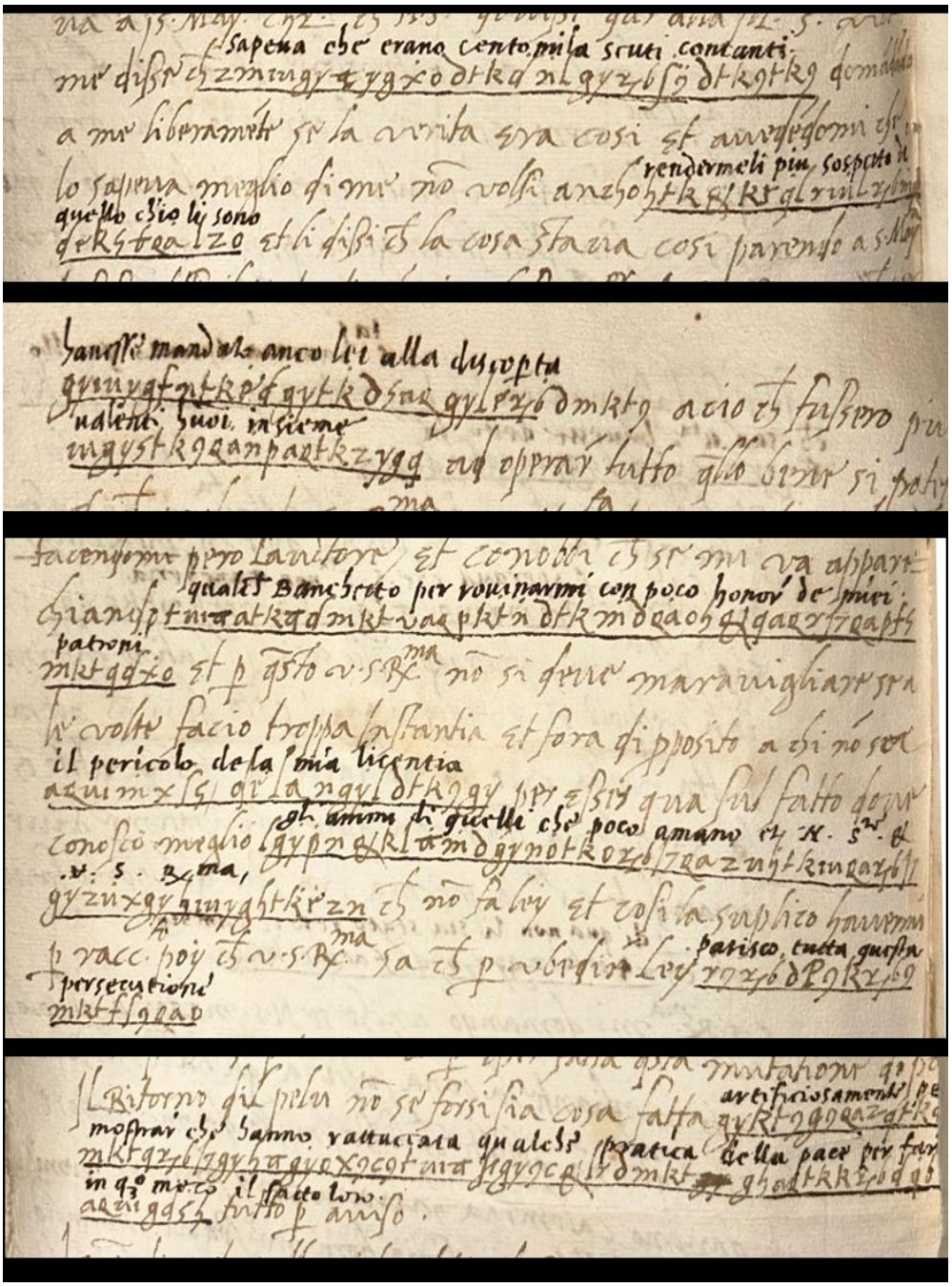


Figure 3 - Third part

We provide here a transcription of the interlinear decrypted text:[6]

First part:
Sua Santità averia tenuto quello / che Sua Maiestà havesse ditto in / di non / pensava / sperava che Sua Santità non mancaria / di qua non li sia stato reso il canbio / Nostro Signore non vi mandasse/ Sua Maiestà … a Sua Santità se vi

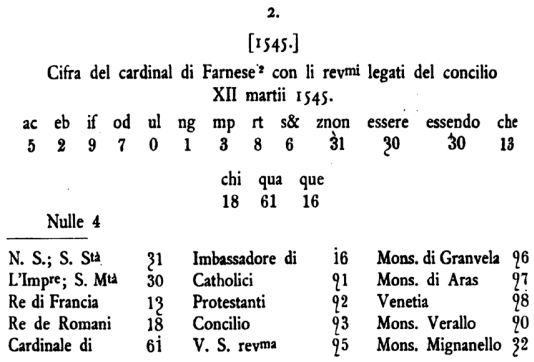
Second part:
se sia deliberata di non consumare il matrimonio con lo illustrissimo signore /
Ottavio, et che per questo Nostro Signore si risolve di aspettare per tutto / el mese che viene, poi non volendo mutar verso, delibera de /
intendere alla dissolutione del matrimonio / sono lettere … Sua Santità vuole fare dieci o dodeci / cardinali

Third part:
sapeva che erano centomilia scuti contanti / rendermeli più suspetto di / quello ch'io li sono
…
havesse mandato anco lei alla discoperta /
valenti homini insieme
…
qualche banchetto per ruinarmi con poco honore de miei / patroni [se lo permetteranno] / il periculo de la mia licentia / li animi de quelli che poco amano et Nostro Signore et / Vostra Signoria reverendissima / patisco tutta questa / persecutione
…
artifitiosamente per / mostrare che hanno rattuccata qualche pratica della pace per fare /
in questo mezo il fatto loro

3. Antonio Elio's Work on Cryptography

The invention of polyphonic ciphers has been attributed to Antonio Elio (Sacco, 1958, p. 15). In the 16th Century, papal cipher secretaries widely used those ciphers (Meister, 1906; Lasry et al., 2020). Those were employed sporadically elsewhere (Lasry, 2023). In Figure 4, we show the earliest known published polyphonic cipher, used by Cardinal Farnese. It is highly likely that Antonio Elio, his cipher secretary, was behind the design of this cipher. Dozens of similar polyphonic papal ciphers from 1545 to 1583 are documented in (Meister, 1906). Other examples of Elio's ciphers are illustrated in (Lestocquoy 1966, pp. 42-45).

With polyphonic ciphers, a cipher symbol (e.g., a certain digit) may have multiple meanings, usually two possible meanings. First, some letters of the Latin alphabet (typically h and z) are removed and ignored, and only the remaining 17 or 18 letters of the alphabet must be encoded. For that purpose, only 9 of the 10 digits are often used, and the 10th is assigned to be a null.



2.

[1545.]

Cifra del cardinal di Farnese[2] con li revmi legati del concilio XII martii 1545.

ac	eb	if	od	ul	ng	mp	rt	s&	znon	essere	essendo	che
5	2	9	7	0	1	3	8	6	31	30	30	18

chi	qua	que
18	61	16

Nulle 4

N. S.; S. Stà	31	Imbassadore di	16	Mons. di Granvela	96
L'Impre; S. Mtà	30	Catholici	91	Mons. di Aras	97
Re di Francia	13	Protestanti	92	Venetia	98
Re de Romani	18	Concilio	93	Mons. Verallo	90
Cardinale di	61	V. S. revma	95	Mons. Mignanello	32

Figure 4 – Example of polyphonic cipher used by Cardinal Farnese – 1545 (Meister, 1906, p. 222)

In the example shown in Figure 4, only the letters a,b,c,d,e,f,g,i,l,m,n,o,p,r,s,t, and u can be encoded. For example, the digit 5 is used to encode both the letters "a" and "c," and the digit 2 is used for the letters "e" and "b." The digit 4 represents a null. Note that "et" ("and," in Italian, shorthanded with "&") is encoded with the digit 6, which also encodes the letter "s." Additional words or proper names are encoded using two-digit numbers (e.g., 18 = "chi"). In this cipher example, one of these numbers may also

[6] Note that in this transcript, we have made some adjustments to the interlinear decryption to match the actual encrypted text better and that the line breaks do not necessarily correspond to those of the ciphertext.

represent a special symbol or a digit with a dot above it.

As a result, the key of a polyphonic cipher is usually shorter and more compact than the key of a homophonic cipher. The polyphonic key can be memorized easily if it does not include a large nomenclature for words or names. Furthermore, as most entities are encoded with a single digit, ciphertexts tend to be shorter than ciphertexts with a homophonic digit cipher, where most entries are encoded with two-digit or three-digit codes.

While innovative, with more compact keys and ciphertexts of shorter lengths, polyphonic ciphers have several drawbacks. Firstly, polyphonic ciphers, such as the example above, are not difficult to break. For more details on a modern technique for the cryptanalysis of polyphonic ciphers, see (Lasry et al., 2020). Secondly, the work of the decipherer, even of a secretary who knows the key, may not be trivial. Because there is ambiguity in deciphering most digits, multiple options exist to decipher a segment of ciphertext, some of which may be plausible, and decipherment, even if the key is known, is not deterministic. Cipher secretaries in the 16th Century were expected to determine which decipherment options were correct based on context and their experience. To remove ambiguity in deciphering, diacritics such as dots were sometimes added (above or below the current digit or the preceding one) to indicate to the decipherer which of the two options should be chosen. Examples of deterministic and non-deterministic polyphonic ciphers are given in (Meister, 1906) and (Lasry et al., 2020).

4. Antonio Elio's Innovative Cipher - our Analysis

In this section, we present our analysis of the cipher introduced in Section 2. Following days of intensive teamwork testing numerous hypotheses, we were able to understand how the cipher works, recover most of the key, and decipher those parts for which there was no matching plaintext. The cipher features several characteristics that make it unique, innovative, and hard, if not impossible, to break.

- **Syllabic**: Most of the enciphered text consists of symbols that encode consonant-vowel syllables, such as "ca" or "ni." A few other symbols encode other types of syllables or letter combinations, or short words, such as "gno", "nd", or "che".

- **Polyphonic**: Each symbol used to encode a syllable has a dual meaning. For example, the symbol  used to encode "de" also encodes the syllable "do". The second choice ("do") is indicated by adding a dot above the previous cipher symbol. Without the dot on the previous cipher symbol, represents "de". Similarly,  represents "di" if a dot is above the previous symbol and "da" if there is no such dot. In theory, decipherment is deterministic. However, those dots are not consistently inserted; in other cases, they might not be visible. As a result, deciphering those symbols is often non-deterministic.

- **Compound symbols**: Using two digits or more (e.g., 23, 123) to encode a specific alphabet letter was customary with digit ciphers. Such compound cipher symbols were rare with non-digit ciphers, which employ graphical symbols or letters to encode plaintext entities. In the present cipher, compound symbols (pairs of graphical or letter symbols) encode some plaintext elements, making cryptanalysis significantly more challenging, as any symbol may be either a stand-alone symbol or part of a pair of symbols used in conjunction. For example,  represents "r" and  represents both "n" and "&", while  and alone represent "qua" and "que". As a result, deciphering segments with such dual-use symbols may not be deterministic.

Each attribute adds to the cipher's complexity, but their combination is unique and unseen in other contemporary ciphers. Generally, consonant-vowel syllables and short words are encoded using single symbols. Compound symbols (using pairs of letters of the Latin alphabet) are used to encode individual letters of the alphabet (e.g., "a", "r") or pairs of consonants such as "nt" or "cr". As with most contemporary Italian ciphers, the letter "h", and any doubled letter are omitted before encryption. For example, "hoggi" is first reduced to "ogi" before being encrypted.

In Figure 5, we show the reconstructed key, where a dot (if present) indicates that the previous symbol had a dot on top. In Figure 6, we show a decipherment example, highlighting the correspondence between the cipher symbols and the elements of the deciphered plaintext. Note: In the colored boxes, a dot on the left indicated that the previous symbol had a dot on top (or was supposed to have one).

At the time we were about to finalize the paper, we discovered another ciphered document sent by the apostolic nuncio in Spain, Giovanni Guidiccioni, to Pope Paul III and the Farnese Curia, from 1535 to 1537, held in the Institute of History in Petersburg. Interestingly, recovering the key, based on matching plaintext sequences, of the cipher employed to encode this document was also quite challenging. The recovered key for this cipher is shown in Figure 7. This cipher has several features in common with the cipher from 1540, such as encoding syllables and using polyphonic symbols and compound symbols. It is also more complex than the cipher from 1540, as it is also a homophonic cipher, unlike the 1540 cipher, which has only one symbol per letter of the alphabet. Furthermore, the 1535-1537 cipher does not employ diacritics to disambiguate polyphones. This cipher further exemplifies the sophistication of cryptographic techniques employed during Elio's service at the papal curia.

5. Conclusion

The present work illustrates Antonio Elio's creative and original approach as a cryptographer. His work also exemplifies the advantage of expertise in both code-making and codebreaking, as only a deep understanding of potential weaknesses in codes can lead to the design of secure ciphers that would withstand cryptanalysis even today.

Funding

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

References

Byatt, L., 1993. "Antonio Elio", in *Dizionario Biografico degli Italiani*, vol. 42, Rome, Instituto dell'Enciclopedia Italiana.

Meister, A., 1906. *Die Geheimschrift im Dienste der Päpstlichen Kurie von Ihren Anfängen bis zum Ende des XVI. Jahrhunderts*, vol. 11. Paderborn: F. Schoningh.

Lasry, G., B. Megyesi, and N. Kopal, 2020. "*Deciphering Papal Ciphers from the 16th to the 18th Century*," Cryptologia, 2020, pp. 479–540.

Lasry, G., 2023. "*Armand de Bourbon's Poly-Homophonic Cipher – 1649*," in Proceedings of the 6th International Conference on Historical Cryptology, 2023, pp. 105–112.

Lestocquoy, J., 1966. *Correspondance des nonces en France. Dandino, Della Torre et Trivultio (1546-1551)*, Roma-Paris, Gregoriana- Boccard.

Sacco, L., 1958. *Un primato italiano: la crittografia nei secoli XV e XVI*. Instituto storico e di cultura dell'arma del Genio.

Figure 5 - Reconstructed key - 1540 cipher

Figure 6 - Sample decipherment

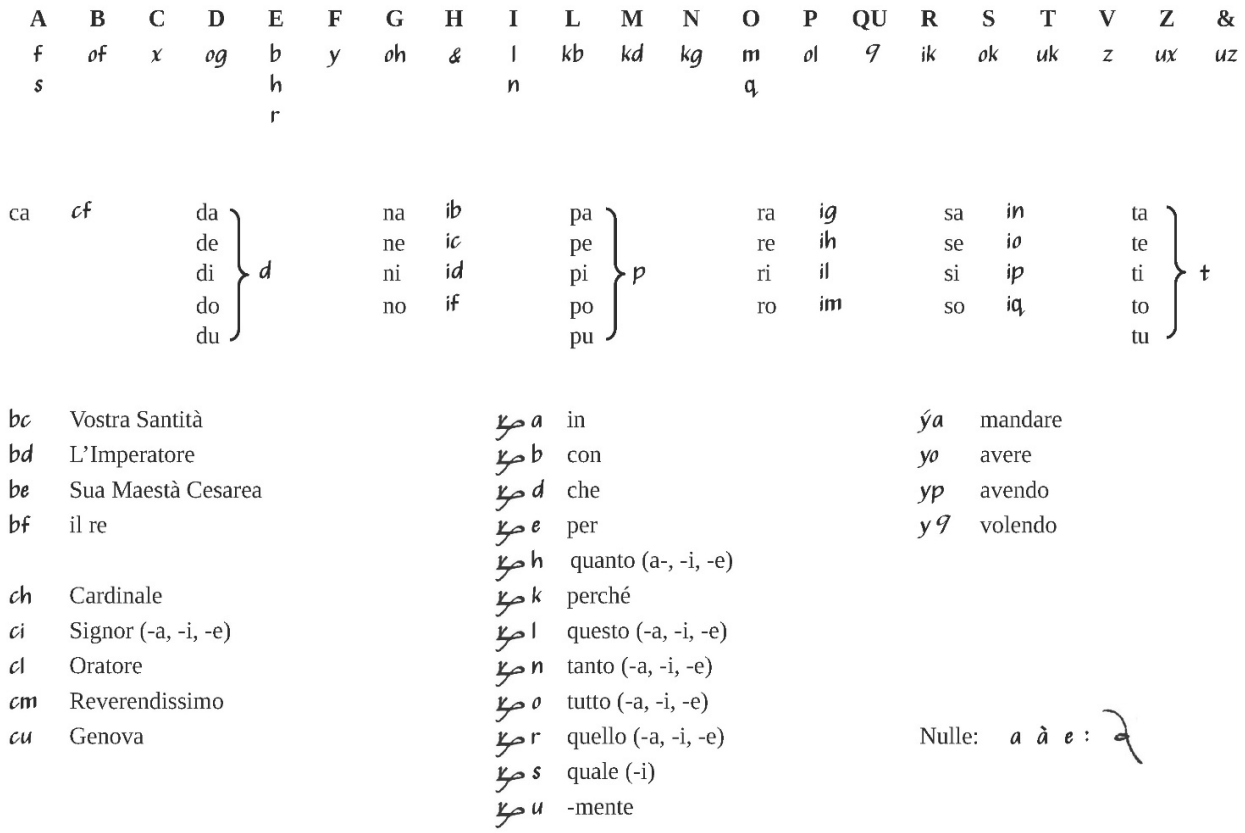


Figure 7 – Reconstructed key for a similar cipher from 1535-1537

A Typology of Pseudo-Cryptology

Benedek Láng
Eötvös Loránd University, Hungary
benedeklang@gmail.com

Abstract

Cipher and code systems can be classified in many ways, with numerous typologies available for organizing both modern and historical cryptographic systems based on their structure. In this article, I propose a different type of typology. I organize various ciphers and codes into a system based on the confirmability of their alleged or actual solutions. This approach places side by side ciphers (e.g., monoalphabetic and polyalphabetic) that would otherwise seem far apart in terms of encoding techniques, and it highlights methods (e.g., book ciphers) that typically do not play a central role in cryptology classifications. This typology becomes useful when attempting to navigate the flood of sensational new cipher-breaking claims that surface weekly in popular media, helping to form a preliminary opinion on whether a proposed solution is arbitrary and unfounded or well-grounded and deserving of professional trust.

1 Introduction

European (and worldwide) archives and manuscript collections house thousands of encrypted messages, estimated to constitute about one percent of the surviving handwritten source material. The DECODE database (Héder-Megyesi, 2022) contains a representative collection of more than ten thousand items, including letters in unknown languages, secret messages, deciphered diaries, unreadable texts, and artificial linguistic systems.

While cipher texts are often easy to read because their solutions were written above the cipher characters by the original recipient, this is not always the case. Many texts remain unsolved within these source collections. Hundreds of mysteries await resolution, with passionate codebreakers proposing numerous—often mutually exclusive—solutions to the most famous examples. A common experience among crypto scholars is encountering—or even directly receiving—cipher and code solutions that are either blatantly false or too complex to verify. What tools can we use to determine whether a proposed solution, presented as the correct decryption, is indeed valid? To what extent do ciphers and codes differ in terms of confirmability? When is the disconfirmation of a given "solution" a straightforward technical matter, and when does accepting a solution proposal become a question of trust? How can we determine whether a proposed solution is correct or arbitrary? Is the provided key *the* solution, or is it something that only the self-proclaimed codebreaker can apply to the text? To what extent, and in what ways, can the validity of a decryption be proven or refuted? The article establishes a typology of solution verifiability based on the attempts to decipher several well-known (e.g., the Beale cipher, the Zodiac killer's letters, and the Rohonc Codex) and some less well-known mysteries.

A casual observer might assume that distinguishing between genuine and pseudo-cryptography is a far easier task than differentiating between reliable and less reliable answers in other fields of science. After all, topics such as the mechanisms of gravitational waves, proper epidemiological methods, the feasibility of cold fusion, or diets for a healthy thyroid often present the layperson with many contradictory proposals. Without specialized knowledge, they may struggle to assess the validity of these claims based on the remnants of their high school science education. While it is generally wise to prioritize and follow the answers provided by institutions that produce scientific knowledge, it is not uncommon for contradictory responses to come from equally reputable scientific sources. However, the aforementioned casual observer might think that in the science of cryptography, the solutions to mysteries rest on mathematical, statistical, and linguistic foundations—fields that, at least at this level, tend to offer relatively uncontroversial answers.

Proceedings of the 8th International Conference on Historical Cryptology, pages 90-100
Poznań, Poland, 16-18 June, 2025

However, distinguishing between genuine and pseudo-cryptology is far from straightforward. Cipher solutions often promise to unveil historical secrets, reveal the locations of hidden treasures, or decode mysteries known to only a select few. These topics are inherently captivating to the general public, and those presenting such solutions can attract significant attention—even when offering baseless decryptions that merely add to the already extensive catalog of popular historical conspiracy theories.

The ability to distinguish between baseless and well-founded decryption attempts—and whether this is always possible—depends on the type of cipher or code in question. The typology of cryptographic methods, ranging from monoalphabetic substitution to polyalphabetic ciphers, as well as homophonic and transposition techniques, is extensive and detailed, even when confined to so-called classical methods. In the following discussion, however, we will focus only on the most fundamental types, as these encompass the problematic cases where the dilemma of confirmation versus disconfirmation—or, in other words, the challenge of establishing or refuting validity—most frequently arises. Rather than systematically analyzing the full range of conceivable historical ciphers—for instance, by surveying all classical ciphers of historical relevance up to the Second World War as standardized by the American Cryptogram Association (ACA)—my selection of cipher types will be somewhat selective and informed by personal experience. I shall focus exclusively on those classical ciphers that have been particularly widespread in history and in which pseudo-cryptology is especially likely to emerge.

Before delving into the task, it is worth briefly addressing the two terms “confirmation” and “disconfirmation”, as they will be used frequently in the discussion that follows. These are scientific concepts employed to support or refute theories. The reason we do not use more familiar expressions from everyday language is that “confirmation” inherently implies long-term uncertainty: what seems validated today may be disproven tomorrow. Newtonian physics, for example, was strongly confirmed for several centuries by scientific observations—many of which are still taught in schools today. During the period between Newton and Einstein, it was more rational to follow Newtonian physics than the earlier Aristotelian framework. Contemporary physicists had good reason to accept it as a reliable theory. However, confirmation is never final; in the history of science, it cannot be. For this reason, we avoid using the term "justification." With the advent of Einsteinian physics, our understanding of mass and energy shifted so significantly that we can no longer speak of Newtonian physics as confirmed.

The concept of “disconfirmation” is even more intriguing. At times, it appears to be a straightforward and final refutation, such as when evidence reveals a conspiracy theory to be nothing more than a simple falsehood, making it unreasonable to believe in it. However, in other cases, it involves the rejection of a scientific theory that might later prove to be useful, relevant, or even supported after all—a phenomenon well-documented in the history of experimental science.

I am going to employ the terms *verification* and *validation* in a restricted sense. While Karl Popper famously argued that the definitive verification of scientific theories—namely, universal statements—is impossible, and that only *falsification* can be conclusive, it nonetheless remains appropriate to speak of the *verification* of non-universal statements, such as the confirmation of a mathematical equation or the solution of a problem with a limited scope. Since cipher solutions constitute non-universal theories, I therefore consider it legitimate to use the term *verification* in this—very limited—context. However, in most cases, instead of employing the terms *verification* and *falsification*—which suggest finality and conclusiveness—I prefer to use *confirmation* and *disconfirmation*, as these imply a provisional and temporally bound status.

The history of cryptanalysis illustrates the gradual and provisional nature of both concepts—*confirmation* and *disconfirmation*. There are solutions where a simple and local verification can determine their reliability, leaving little room to imagine this judgment changing in the future. In other cases, uncertainty remains: a solution may be strongly disconfirmed yet could still, in whole or in part, eventually integrate into our body of established knowledge. Conversely, a solution may seem confirmed only to be partially or entirely overturned later.

Despite their gradual and provisional nature, these categories are well worth exploring. In fact, the essence of scientific inquiry across all fields lies in making informed decisions about them.

2 Verification of the decryption of monoalphabetic and polyalphabetic ciphers

Monoalphabetic ciphers offer relatively little room for pseudo-cryptology. Following the work of Arabic scientists who introduced frequency analysis over 1,200 years ago (Mrayati et al, 2003), numerous additional methods have been developed—such as the probable word method based on relative letter frequencies, as well as bigram and n-gram analysis. These techniques render the decryption of monoalphabetic ciphers quick, definitive, and straightforward to confirm.

However, when dealing with historical sources, several factors significantly complicate the process of decryption. The handwriting of early modern scribes was often unreliable. They did not adhere to consistent spelling conventions and frequently made mistakes. A common inference used by modern codebreakers is to assume that no single word contains the same letter repeated three times in succession. Thus, if a symbol appears three times consecutively in the ciphertext, it must represent two consecutive words. However, this reasoning does not always hold for genuine historical ciphers. Scribes and letter writers often repeated or omitted characters purely out of carelessness or error.

Even in cases where the scribe worked meticulously, they faced numerous choices. Should the same cipher character be used to encode both u and v? Should j and i be differentiated? Should separate symbols be applied for accented letters and other special characters (e.g., in French: é, è, ê, à, ç, etc.), or should their unaccented counterparts be used instead? Surviving cipher keys provide examples of all these choices.

How, then, should the codebreaker determine which text to use as a basis for calculating the statistical data to compare with the ciphertext's statistics? What should be considered the linguistic analogue to the encoded text? Modern text editions are unsuitable for this purpose. While the editorial changes they incorporate are often well-justified by professional considerations, they alter the original text's statistical properties, making them ultimately unfit as analogues for the ciphertext.

Even if sufficient historical texts faithfully preserving the original—unstable—spelling are available in electronic form, the historian-codebreaker still faces significant challenges. Which text should they use to aid in decryption? One that includes accented letters and distinguishes between u and v, as well as i and j? Or one without accents but still differentiating these characters? Or perhaps one that uses accents inconsistently, distinguishes i from j, but applies only one form for u and v? The possibilities are overwhelming!

Thus, while the encryption and decryption techniques may be remarkably simple, the linguistic challenges can become so complex that many arbitrary elements inevitably find their way into the decryption process.

All of these difficulties, however, are ultimately technical in nature. With a robust codebreaking tool, such as Cryptool 2, which can calculate relative frequencies from a sufficiently large corpus of historical texts—not only for individual letters but also for bigrams (pairs of consecutive letters), trigrams, and even word patterns—even monoalphabetic systems originating from unconventional linguistic contexts can be reliably decrypted (Megyesi, 2020). Once a monoalphabetic cipher has been cracked, the decryption is considered definitive, as the key provided by the codebreaker can be verified by anyone using a different portion of the text.

And yet, ungrounded monoalphabetic solutions are occasionally proposed for historical texts, relying heavily on real or imagined linguistic contingencies. An early and swiftly disconfirmed solution of the Rohonc Codex serves as an example, that proposed reading the characters of the codex as simple letters (see more about this attempt in Láng, 2019). However, for the reasons outlined above, there is rarely any substantial debate surrounding the decryption of such ciphers.

Polyalphabetic ciphers, which change the cipher alphabet according to a specific system, appear much more complex but are similarly straightforward when it comes to confirmability.

Since the 19th-century work of codebreaker Charles Babbage, this method has been solvable by exploiting its periodicity (the cipher alphabets always follow the same sequence). While some polyalphabetic ciphers are easier to crack and others require extraordinary effort, they remain straightforward from a confirmation perspective: once solved, the key allows anyone to verify the solution.

Consequently, neither monoalphabetic nor polyalphabetic ciphers do give rise to long lasting debates, conspiracy theories, or pseudo-solutions.

3 Confirmation of Homophonic Cipher Decryptions

The homophonic method combines elements of ciphers and codes. It emerged in the 15th century with techniques that assigned more than one cipher character to more frequently used letters, thereby obscuring frequency differences in the ciphertext. To make the method even more resistant to decryption, special symbols were often used for double letters, as these are characteristic of specific languages and offer a starting point for codebreakers. Additionally, syllables frequently had their own symbols, and the system often included meaningless characters, or nulls, to further complicate analysis. Symbols also replaced common words, preventing their characteristic patterns from aiding the codebreaker. These dictionary-like structures, called nomenclators, represent the encoding aspect of the method. By the 19th century, codebooks had become highly elaborate, but even in early homophonic systems, the number of code symbols often reached into the hundreds.

Codes and ciphers differ significantly from the perspective of decryption. For ciphers, the main tools include frequency analysis of letters and groups of symbols, vowel identification tests, clustering, and the analysis of repetitions and word patterns. In the case of codes, however, progress often depends on locating the codebook and performing context analysis. Fewer computational tools are available for solving codes and nomenclators, making their decryption more reliant on human intelligence and historical knowledge rather than computational algorithms. While ciphers can often be fully decrypted if solved, codes may not be entirely deciphered, as there are often one or two symbols that the codebreaker simply cannot determine.

Consequently, codes and ciphers differ similarly when it comes to confirming their solutions. As mentioned earlier, cipher solutions can be validated by decoding a randomly selected portion of the text using the key provided by the codebreaker. In the case of codes, however, the process is more akin to a hermeneutic circle. The question becomes how well other parts of the encoded message and the historical context support the codebreaker's assumptions.

It is no coincidence that historical homophonic systems are sometimes extremely difficult—if not impossible—to decrypt completely. While the alphabet of the cipher eventually yields (even if simple frequency analysis is insufficient to break it) and nulls can be uncovered with sufficient effort, the syllable symbols and the code elements—the words from the nomenclator table—pose far greater challenges.

But sometimes even the decryption of a homophonic alphabet is not straightforward, particularly when the available text is too short. Consider a real and intriguing example: the Zodiac letters.

The story of the Zodiac Killer is well-known, thus here we will focus solely on his encrypted letters. The first cipher (the so called Zodiac 408) was cracked by amateur codebreakers using a simple probable word method, assuming that the distinctive pattern of the word “kill” would appear in the text. Afterward, the killer devised a new cipher that stumped the codebreakers. Many suspected it involved homophonic substitution, but solutions only emerged in recent years.

A solution for the letter referred to as Zodiac 340—or more specifically, for its first half—was proposed by a noted cryptography historian. However, this was not accepted by the professional community. The main issue is that the text is so short that, when reconstructed as a homophonic cipher, the same character is rarely encoded in the same way twice. In short, if a text of around 170 characters is revealed to use a homophonic cipher with nearly 100 cipher symbols, it becomes extremely difficult to demonstrate that the solution is indeed the correct one.

However, it is not impossible, and this is where the new solution by David Oranchak and his collaborators comes into play, presented in late 2020. Oranchak and his team also reconstructed the Zodiac 340 as a homophonic cipher, but they tackled the entire text (Oranchak et al 2024). The solution was tested by many others, who arrived at similar results, and the FBI confirmed its accuracy. By the night following the announcement, the solution had achieved near-complete confirmation.

While the earlier solution's issue—that decoding a short text as a homophonic cipher can be arbitrary—could also be raised here, it can fortunately be ruled out using mathematical tools. Without delving into technical details, it is worth mentioning that the American mathematician, electrical engineer, and cryptographer Claude Elwood Shannon (1916–2001) introduced the concept of unicity distance in two seminal studies published in 1948 and 1949 (von zur Gathen, 2023). This is a complex formula that, when applied to variables such as the length of the text, the entropy of the language, and other factors, determines definitively whether the solution to a classical cipher is arbitrary or well-founded. Shannon's formula can also be applied to the decryption of homophonic ciphers, providing an entirely objective measure of the reliability of the codebreaking process.

Therefore, it is no surprise that the true domain of pseudo-cryptography is not homophonic ciphers.

4 Understanding Transpositions

For those seeking to excite their audience with entirely unfounded yet flashy codebreaking claims, transposition ciphers provide a far better playground. This is ironic because there is nothing inherently wrong with transposition ciphers; alongside substitution ciphers, they represent one of the two major and highly respectable branches of cryptography. Transposition ciphers do not replace the letters of the original message with symbols, numbers, or other letters. Instead, they rearrange the letters according to a specific system, systematically altering their order.

However, this method also lends itself to the greatest potential for misuse, as exemplified most clearly by the famous "Bible Code" theory. It is important to emphasize upfront that in this and similar cases, the so-called "decoders" are not reconstructing a proper transposition cipher. Instead, they merely claim that by arranging a text in a certain way, one can discern a meaningful sequence of letters, which, of course, is said to carry a deeper significance.

It is not immediately obvious where the deception lies in the Bible Code theory, popularized through the works of Michael Drosnin. Drosnin, an American journalist, published several books on the Bible Code, and at one point, there were even plans to adapt the topic into a Hollywood film.

The foundational idea, however—that hidden messages or codes are embedded in the Hebrew text of the Torah, the first five books of the Bible—did not originate with Drosnin but with Eliyahu Rips, an Israeli mathematician. Using various statistical analyses, Rips concluded that if you remove the spaces from the Torah's text, you can, intriguingly, find the names of numerous important rabbis from the two millennia following the Bible's composition (Witztum, Rips, Rosenberg, 1994.). These names are not found randomly but through the use of a computer program capable of identifying sequences of letters spaced at equal intervals within the lengthy text. For example, the first letter of a rabbi’s name might appear in the Book of Genesis, the second letter 1,606 characters later, the third at twice that distance, and so on. The final letters of the name might even appear in the Book of Exodus.

To illustrate the concept on a smaller scale, imagine this: take the English word GENERALIZATION and read every third letter. You will find the word NAZI hidden within it. From this, you could immediately draw all sorts of conclusions! Of course, what Rips and his collaborators did was not exactly this. First, because the letters they found were much farther apart from one another, and second, because they genuinely believed that the names of rabbis who lived after the composition of the Bible were intentionally embedded in the revealed text of the Torah.

Michael Drosnin took Rips's foundational idea and developed it in a highly dramatic way. Using

a sophisticated computer program, he searched for instances where the four Hebrew letters corresponding to the name Obama appeared in the Bible, evenly spaced. When he found such a sequence—for example, with the letters spaced 1,200 characters apart—he formatted the Bible's text into rows of 1,200 characters each. This caused the letters spelling Obama to align vertically. Drosnin then examined the surrounding text in this newly formatted grid. Since the horizontal rows contained the original text of the Bible, it was quite likely that he would find something suggestive. Vertically and diagonally, he identified additional words whose letters were also evenly spaced, reinforcing the ominous interpretation. The result was striking for the average reader: around the letters spelling Obama, words hinting at presidential election emerged. Even more impactful was Drosnin's claim that using this same method, he had predicted the assassination of former Israeli Prime Minister Yitzhak Rabin.

Mathematicians and more knowledgeable experts found themselves at a loss as to where to begin explaining why this is nonsense. First, biblical Hebrew is a consonantal language, meaning that words are generally much shorter than in most languages. As a result, it is relatively easy to find short words formed by letters spaced at equal intervals. Moreover, whether something feels like a "discovery" is often a matter of interpretation. Second—and this is the more critical argument—with this method, meaningful words can always be found in sufficiently long texts. If you work with a large enough text file, you will inevitably locate the evenly spaced letters of many different words.

Drosnin rejected the criticisms and promised that if assassinations could be demonstrated in the text of Moby Dick, he would consider it a serious counterargument. According to tradition, the Hebrew text of the Bible's first books is no ordinary text—it was dictated by God to Moses, letter by letter. Therefore, it's not surprising, he argued, that superhuman knowledge or, if you prefer, divine foresight can be discerned within it. Moby Dick, on the other hand, is not regarded as a revealed text—except perhaps by a few ardent fans of Herman Melville. Thus, there should be no information pointing to assassinations within its text.

That was all the critics, led by mathematician Brendan McKay and his colleagues, needed to spring into action (McKay 1999, 1997, Bar-Hillel 1998). They quickly pulled the story of the white whale, Moby Dick, off the shelf. The challenge, of course, was inherently unfair because Moby Dick was written in English, a language that is not purely consonantal, and names and words referring to assassinations typically consist of more letters. Nevertheless, the critics embraced the task, and using their own computer program, they swiftly found the names of Indira Gandhi, Leon Trotsky, Abraham Lincoln, Martin Luther King, John F. Kennedy, and Princess Diana. Around these names, they also identified intriguing details about the circumstances of their deaths. For instance, near Trotsky's name, the word hammer appeared, while intersecting with Diana's name were Dodi al-Fayed and the name of her chauffeur.

McKay and his colleagues didn't stop there. They became so engrossed in their puzzle-solving—or rather, puzzle-creating—that they even "predicted" Drosnin's death using a similar method. According to their mock prophecy, Drosnin's death would be violent, occur in either Cairo or Athens, and the perpetrators would be two fellow code researchers. Of course, they quickly clarified—since some readers had previously taken their results seriously—that this was all just a joke.

The final blow came when they examined the chapter in Drosnin's book discussing the September 11 attacks and its supposed biblical codes. They noticed, to their amusement, that Drosnin's own text—without the author's apparent awareness—also contained codes. By connecting letters spaced at equal intervals, they found that Drosnin's text "predicted" the bloody terrorist attack in Kuta, a town in Bali.

In summary: while the "regular" transposition decryptions, which provide an explanation for every character in the encrypted message, can be easily confirmed (such as the decryption of the 3rd message of Kryptos), those transposition decryptions that identify certain equidistant character patterns within a large text corpus—i.e., decrypt only a small portion of the available text while neglecting the rest from the perspective of decryption—are often arbitrary. At the same time, disproving such solutions typically requires considerable effort.

5 Book ciphers

The book cipher is one of the most resilient cryptographic methods. In essence, it involves a series of numbers that correspond to the positions of words or letters in a specific book—one previously agreed upon by the sender and recipient of the message. As long as both parties use the same edition and count accurately, they can exchange messages with a very high degree of security. How can this method be misused to create an unfounded solution that falsely claims to be a book cipher? The best example to illustrate this is the Beale Cipher (Kruh 1982, 1988).

The story of the Wild West treasure hunter is well known, so we can skip its details. From a codebreaking perspective, the key point is that three letters contain three numerical sequences, the second one of which, according to the story, was identified and deciphered as a book cipher by a certain James B. Ward. The second letter's numbers correspond to the numbered words of the United States Declaration of Independence, with each number standing for the first letter of the corresponding word. By their nature, book ciphers are nearly impossible to crack unless you know the specific edition of the book to which the numbers refer.

The decryption of this letter revealed the exact quantity of the supposed gold treasure and indicated that the first letter contains the treasure's precise location. The third letter reportedly provides information about Beale's expedition partners and their relatives, who were to inherit the treasure.

Following this, hordes of gold prospectors, codebreakers, adventurers, and mathematicians turned their attention to the first and third undeciphered codes (unsurprisingly, focusing more on the first, as the list of beneficiaries in the third letter seemed far less thrilling than the treasure's location). Over the past two centuries, secret excavations have been carried out under the cover of night in numerous locations across Bedford County, much to the frustration of the locals. In the late 1960s, the Beale Cypher Association was founded and organized several academic conferences in the following years to study the Beale codes. Many have claimed to have deciphered the letters, while others have cast doubt on these claims. To date, however, no one has succeeded in identifying the book that serves as the key for the first and third letters.

Many have noted that the overly romantic treasure-hunting backstory raises serious suspicions. Why does the deciphered second letter state that the first letter will contain the treasure's location and the third will list the beneficiaries? How could the author have known that the second letter would be deciphered first? Why write the second letter at all if the first already reveals the treasure's location? Furthermore, why does the second letter refer to the others as the "first" and "third," assuming it would be solved first? Why not call the other letters the "second" and "third" instead? Another puzzling detail: how is it that the third letter, which supposedly contains the names and addresses of thirty treasure hunters and their families, is only 618 characters long? Such a brief text would be insufficient to encode at least sixty names, let alone their addresses. These oddities strongly suggest that the entire story might be a hoax.

But how can it be proven that a sequence of numbers is a hoax and not decodable? The answer, as all codebreakers agree, is: it cannot! Interestingly, the same evidence is interpreted by some as proof of a hoax and by others as confirmation of authenticity. If you align the numbers from the first letter with the Declaration of Independence, you do not get a meaningful text. However, at one point, the sequence produces the alphabet in order, from A to P. According to statistical rules, this is so improbable that it must be deliberate. Some believe this pattern indicates the hoaxer grew tired and, instead of generating more random numbers, resorted to following a pattern. Others suspect it is a deliberate clue—a nod from the encoder, encouraging the decoder that they are on the right track but that the text is doubly encoded and the real solution is yet to be uncovered.

Recently, several developments have come to light that cast new perspective on the debate about the authenticity of the Beale ciphers. Several cryptologists have attempted to replicate the work Ward claimed to have done when decrypting the second letter. They took the Declaration of Independence, numbered its words, and read the letters corresponding to the

numbers. However, the resulting text was far from the clear message Ward published. This discrepancy arose for two reasons: first, multiple copies of the Declaration of Independence were in circulation, and while the differences between them were minor (such as whether certain words were written as one or two), these small variations significantly impacted the numbering of the words and, consequently, the final decryption. Second, Beale himself made numerous mistakes during the encoding process, incorrectly numbering the words.

Curiously, Ward in the 1880s used the exact same version of the Declaration of Independence as Beale supposedly did in the 1820s. Even more striking, he made exactly the same counting errors during decryption that Beale did during encryption. And yet, in his published decryption, Ward made no mention of these complications, presenting the process as smooth and straightforward. Could it be because it was straightforward for him? Could it be because the encoder and decoder were one and the same—Ward himself?

Did the publisher of the Beale Papers deceive everyone? Is stronger evidence needed to support the hoax theory?

Seeing these peculiar complications, it becomes clear what the renowned American codebreaker William Friedman—who cracked numerous wartime ciphers—might have meant many years ago when he was asked whether he believed the suspicious and undeciphered Beale ciphers were genuine:

> ‘‘On Mondays, Wednesdays and Fridays, I think it is real, on Tuesdays, Thursdays and Saturdays I think it is a hoax.” (quoted: Clark 1977, 126.)

In recent years, a range of new statistical analyses has been conducted, most of which lend support to the long-standing suspicion that the Beale ciphers are a hoax (Campanelli 2022, Wase 2020a and b).

6 The Difficulties of Decoding Artificial Languages

In our typology, we have finally reached the area where it is the most difficult to prove, understand, or even refute the correctness of a decryption, and this is the field of artificial languages and writings. In such languages, the author encodes entire words and concepts, making artificial languages technically related to codes. However, the decryption of codes follows the same principle we stated earlier: even in the best case, decryption is never 100% ready, because there will always be one or two code groups whose meaning cannot be identified based on the text's context. What is even worse is that even a correct decryption is extremely difficult to confirm. It is not surprising that it is assumed most commonly in the case of artificial languages, that they are actually bluffs, and their decryption is impossible.

To illustrate this case, let us look at a famous cryptographic example, the Rohonc Codex, which held a prominent place for a long time among unsolved manuscripts (Láng, 2021). After numerous unsuccessful, and often amateur, decryption attempts—claiming that the codex was written in Old Hungarian, vulgar Latin, ancient Romanian, or Sanskrit—the breakthrough came with the collaboration of two Hungarian researchers, Gábor Tokai and Levente Király. The two researchers, by examining the repetitions, were able to determine the correct order of the torn out pages. Then, based on the biblical-themed images in the codex and the chapter-starting repeating patterns, they identified the names of the four evangelists in the symbolic system. Using the analogy of the biblical stories summarized in the codex, they identified the digits, and then, through trial and error, gradually assigned meaning to the individual symbol groups. This procedure was supported by an online codebreaking software developed by Király.

In the case of artificial languages and codes, confirming or refuting a solution is far more complex and circular than with ciphers. Suppose, for example, that I claim to have deciphered a text and assert that it consists of a finite number of code groups referring to nouns, which are written sequentially without following any known grammatical rules. In such a scenario, my theory would be nearly impossible to disprove. Using my key, any encoded text could be interpreted as a sequence of nouns. I might believe that I have found the correct solution to the text, even though anyone else could generate an entirely different sequence of nouns that

"perfectly maps" onto the encoded text as well. Thus, one could just as easily "prove" that the text is actually a list of demon names or alchemical terms.

For this reason, this approach should be avoided when attempting to decipher unknown languages—not because it doesn't work, but precisely because it works too easily. If words are skillfully paired with symbols, it is possible to produce a prayer text that seems somewhat plausible, even if it lacks structure or grammar. But similarly, well-chosen profanities could just as easily fit the symbols, yielding a coherent, though vulgar, text.

According to Tokai and Király's solution, the Rohonc Codex was written in a script following some form of artificial linguistic scheme. This means that instead of searching for phonemes or individual letters, one must look for words or at least word fragments. They argue that the text is religious in nature, containing gospel stories, biblical paraphrases, and prayers.

They were, of course, fully aware of the risks. Let us quote their methodology at length:

"The principles of our criteria and method of codebreaking may seem banal to the reader, but we must emphasize them because of the bad reputation gained by the amateur researchers of the codex. Furthermore, as many examples in our next paper on the "wobbliness" of the code will show, the writing system is far from being simple and clean. We must affirm that these results are not due to methodically deficient research but to the writing itself, which was analyzed with painstaking care and strictness.

We demand that one symbol signify one thing, and whenever there is any digression from this principle—either by more symbols signifying one thing or one symbol signifying more things—it must be sufficiently supported by argument. Our case is difficult because the codex has codes signifying words of a language, and words behave less regularly than letters. In every natural language the presence of homonyms and synonyms creates ambiguity. Yet we demand that even this amount of ambivalence in our proposed solution be supported by evidence." (Király and Tokai 2018, 293)

To determine whether Tokai and Király's solution is correct, we must first become as familiar as possible with the vocabulary they propose. As a second step, we must accept the specific linguistic features they attribute to the codex's language. Temporarily, we must place trust in their theory while studying and essentially learning this newly proposed language in order to make an informed judgment about whether it functions as they claim. This process requires considerable time and effort, but eventually, the first signs of confirmation begin to emerge—just as they did in Tokai and Király's case.

The vocabulary primarily consists of words with a single meaning, which they consistently retain whenever they appear in the text. However, confusingly, some symbol combinations sometimes represent different concepts—for instance, the words we, man, and you are all represented by the same symbol combination. Similarly, another symbol combination can mean either I am or you are, while yet another can stand for both yours and ours. Despite such ambiguities, the authors remain confident that they are on the right track: "The core of our reading has such strong inner and outer evidence that we may affirm that it stands beyond doubt."

Once the lexicon is learned and the unusual grammar is accepted, more and more of the codex's content becomes accessible, bringing clear confirmations. The vocabulary derived from one part of the codex enables the reading of other sections. It becomes evident that the biblical references in the codex correspond to passages from the New Testament: the same numerical figures appear in the codex as in the Gospels—five barley loaves and two fish, twelve baskets, ten thousand talents, a hundred denarii, and so on.

But it is not only the numbers that can be identified. Tokai and Király also find the Pater Noster prayer in the codex, along with the texts of *Ave Sanctissima Maria* and *Ave Maria Gratia Plena*. Additionally, they uncover verbatim quotes from gospel passages, such as Matthew 7:17–18, the parable of good and bad fruit.

The first publication of the decryption received mixed and ambivalent reactions (Pelling, 2018) However, with full access to the complete dictionary (Der Rechnitzer Kodex: https://www.rechnitzer-kodex.hu/), the entire digitized codex is now "readable". This online tool, as the authors explain, was created to

facilitate the study of this extraordinary manuscript. The glossary—or dictionary—is searchable. Users can enter any Rohonc word into the search field and retrieve not only its meaning but also its occurrences in the codex and its various grammatical contexts.

The solution is further confirmed by an indirect historical argument. Similar structured artificial languages were created one after another in the 16th and 17th centuries by figures such as Johannes Trithemius, John Wilkins, Athanasius Kircher, René Descartes, Isaac Newton, Gottfried Wilhelm Leibniz, Marin Mersenne, and Cave Beck. As anyone can test, for example, with Wilkins's well-developed system, these languages become nearly indecipherable code systems if their vocabulary and grammar are unavailable to the reader. Essentially, they function as ciphers—even if their creators did not intend to encrypt their content.

Given the nature of such code systems, we should not expect a solution to be 100% complete (98% suffices), nor should we expect it to be quickly or easily confirmed. The nature of the linguistic system itself prevents the kind of definitive proof that, for example, Shannon's formula can provide with a quantifiable value.

7 Summary

Ciphers and codes can be classified in many different ways. In this article, I have attempted to apply a very indirect criterion, organizing them based on the nature of the confirmation or disconfirmation of their alleged or actual solutions. This approach places side by side ciphers (such as monoalphabetic and polyalphabetic) that would otherwise seem distant from each other in terms of encoding techniques, and gives particular emphasis to those (such as the book cipher) that do not play a central role in the science of cryptology. However, this typology will become especially significant when we try to navigate through the flood of sensational codebreaking claims that appear week after week in popular news, and form an initial opinion on whether a given solution is arbitrary and unsupported, or well-founded and worthy of the professional community's trust.

What can we learn from distinguishing between real and pseudo-cryptography? Above all, we learn that many solutions appear to be definitively confirmed or disconfirmed. In other words, it can be definitively stated whether they offer the real solution or whether we are faced with a simple bluff. The method of judgment is provided by science, through mathematical tests, statistical analyses, and linguistic insights. This process is similar to what happens in other scientific fields when a new theory is either accepted by the scientific community or dismissed as baseless.

However, in the field of cryptography—just as in other scientific areas—there are more complex situations: solutions derived through systematic scientific tools may not be accepted by the community, or may be reluctantly accepted, with the confirmation/disconfirmation process being more gradual and uncertain. This is particularly true in the case of homophonic ciphers, nomenclators, book ciphers, transpositions, and code or artificial languages. This uncertainty is as much a part of scientific research as final acceptance or rejection.

Acknowledgments

This work has been supported by Riksbankens Jubileumsfond, grant M24-0028: Echoes of History: Analysis and Decipherment of Historical Writings (DESCRYPT).

References

Bar-Hillel, Maya & Bar-Natan, Dror & McKay, Brendan. 1998. "The Torah Codes: Puzzle and Solution." CHANCE. 11.

Héder, Mihály and Beáta Megyesi. 2022. The DE-CODE Database of Historical Ciphers and Keys: Version 2. In *Proceedings of the 5th International Conference on Historical Cryptology*, HistoCrypt22, 111-114.

Campanelli, Leonardo. 2022. "A statistical cryptanalysis of the Beale ciphers." *Cryptologia*, 47(5), 466–473.

Clark, R. *The Man Who Broke Purple*. Boston, MA: Little, Brown and Co., 1977.

Király, Levente Zoltán, Tokai Gábor, 2018. "Cracking the code of the Rohonc Codex", *Cryptologia* 42: 285-315.

Kruh, Louis. 1982. “A Basic Probe of the Beale Cipher as a Bamboozlement.” *Cryptologia*, 6: 378-382.

Kruh, Louis. 1988. “The Beale Cipher as a Bamboozlement Part II.” Cryptologia, 12: 241-246.

Láng, Benedek. 2019. Dead Ends in Breaking an Unknown Cipher: Experiences in the Historiography of the Rohonc Codex. In *2nd International Conference on Historical Cryptology* HistoCrypt 2019, 51-54.

Láng, Benedek. 2021. *The Rohonc Code*, Penn State University Press.

McKay, Brendan. 1997. Assassinations Foretold in Moby Dick! http://users.cecs.anu.edu.au/~bdm/dilugim/moby.html (accessed: 2025.02.07)

McKay, Brendan, Bar-Natan, Dror Bar-Hillel, Maya, Kalai, Gil. 1999. “Solving the Bible Code Puzzle. Statistical Science.” 14.

Megyesi, Beáta, et al. 2020. "Decryption of historical manuscripts: the DECRYPT project." *Cryptologia* 44.6: 545-559.

Mrayati, Mohamad, Yahya Meer Alam, and M.Hassan at-Tayyan. *al-Kindi's Treatise on Cryptanalysis.* Riyadh: King Faisal Center for Research and Islamic Studies, 2003.

Oranchak, D., Blake, S., & Van Eycke, J. (2024). “The Solution of the Zodiac Killer's 340-Character Cipher.” *arXiv preprint arXiv:2403.17350*.

Pelling, Nick. 2018. Király and Tokai’s Rohonc Codex decryption… http://ciphermysteries.com/2018/06/03/kiraly-and-tokais-rohonc-codex-decryption (accessed: 2025.02.07)

von zur Gathen, Joachim. 2023. "Unicity distance of the Zodiac-340 cipher." *Cryptologia* 47.5: 474-488.

Witztum, Doron, Eliyahu Rips, Yoav Rosenberg, 1994. "Equidistant Letter Sequences in the Book of Genesis" *Statistical Science* 9 (3)

Wase, Viktor. 2020a. “Benford’s law in the Beale ciphers.” *Cryptologia*, 45(3), 282–286.

Wase, Viktor. 2020b. “The Role of Base 10 in the Beale Papers.” In *International Conference on Historical Cryptology*, HistoCrypt, 153-157.

Decipherment of Historical Manuscripts with Unknown or Rare Writings: The DESCRYPT Project

Beáta Megyesi[1], Alicia Fornés[2], Mihály Héder[3], Raphaela Heil[1], Benedek Láng[4], Nils Kopal[5], Rune Rattenborg[6] and Michelle Waldispühl[7]
[1]Stockholm University, Sweden [2]Universitat Autònoma de Barcelona, Spain
[3]Technical University of Budapest, Hungary [4]Eötvös Loránd University, Hungary
[5]Hochschule Niederrhein, Germany, [6]Lund University, Sweden, [7]University of Oslo, Norway

Abstract

We present a newly funded research program, DESCRYPT, aimed at deciphering and analyzing historical texts with rare or unknown scripts. The project leverages advancements in computational linguistics, artificial intelligence (AI), and image processing, alongside traditional philological methods, to develop innovative tools for transcription, recognition, and interpretation of historical writings with rare/unknown scripts, including ciphertexts. By integrating interdisciplinary expertise, DESCRYPT addresses the challenges posed by complex and undeciphered texts, preserving and unlocking the secrets of our shared cultural heritage.

1 Introduction

Historical texts have long served as windows into the cultural, intellectual, and linguistic landscapes of human civilization. These writings provide unique perspectives on the evolution of languages, belief systems, and societal structures. However, analyzing such texts—particularly those written in rare or undeciphered scripts—poses significant challenges. Traditional philological methods, relying on meticulous manual examination and contextual expertise, are often insufficient when faced with obscure linguistic systems or fragmented historical records.

In recent decades, digital humanities have introduced new opportunities for text analysis through computational tools like 3D modeling, optical character recognition (OCR), hand-written text recognition, topic modeling and other content-based text analytics based on machine learning algorithms. Despite their promise, these technologies struggle with the irregularities that characterize many historical texts, such as degraded materials, non-standard scripts, and irregular and cryptic linguistic structures. Unknown writings share striking similarities with ciphers in that they encode meaning through complex symbols or systems that require decipherment to be understood.

Building on the success of the DECRYPT project (Megyesi et al., 2020), which focused on European early modern cryptographic texts, DESCRYPT expands its scope to embrace a wider range of historical writings. By integrating computational linguistics, AI, cryptanalysis, and traditional linguistic, historical and archeological expertise, DESCRYPT seeks to overcome these challenges and establish innovative methodologies for transcribing, deciphering, and contextualizing rare scripts. This endeavor represents a vital step toward preserving and interpreting the hidden layers of human history.

2 Previous Work

The decipherment of historical writing systems with rare or unknown scripts has evolved significantly, from early breakthroughs to modern computational approaches. Pioneers such as Jean-François Champollion (Champollion, 1822) and Michael Ventris with John Chadwick (Chadwick, 1958) laid the foundation for understanding ancient texts, demonstrating key methodologies that remain influential today.

The field has since integrated computational linguistics, enhancing traditional methods through digital tools. The expansion of digital humanities has led to extensive online repositories cataloging inscriptions across various civilizations, including Egyptian hieroglyphs, Linear B, Runic inscriptions, Ancient South Arabian scripts, Greek and Latin epigraphy, and cuneiform texts. However, challenges persist in data standardization, digitization, and large-scale comparative analysis.

In addition, encrypted manuscripts constitute another category of historical texts that remain

Proceedings of the 8th International Conference on Historical Cryptology, pages 101-105
Poznań, Poland, 16-18 June, 2025

largely undeciphered. Historical cryptology, outlined by David Kahn (Kahn, 1967), has progressed significantly through computational techniques.

Advancements in computational cryptanalysis have enabled breakthroughs in deciphering various encoded texts. Algorithmic approaches have been successfully applied to historical ciphers such as the Zodiac ciphers, the Copiale cipher (Knight et al., 2012), the papal ciphers of the Vatican (Lasry et al., 2020) and the recently deciphered Mary Stuart letters (Lasry et al., 2023). Machine learning has further expanded the potential for decoding unknown scripts, as demonstrated in studies on the Borg cipher (Aldarrab, 2017), anagram-based sources (Hauer and Kondrak, 2016), infilling text in ancient tablets (Papavassileiou et al., 2023) and restoring text along with attributing geographic area and dating of ancient Greek inscriptions (Assael et al., 2022). However, despite progress, computational methods remain limited in fully automating the decipherment of unknown scripts. Instead, interdisciplinary collaboration continues to play a crucial role, as seen in the recent decipherments of Linear Elamite (Desset et al., 2022), the Maya code (Coe, 2011), Amorite vocabulary (George and Krebernik, 2022), and Kushan script (Bonmann et al., 2023).

Automatic script recognition through artificial intelligence has further transformed the field. Techniques such as cluster-based script identification and digital image processing have enhanced manuscript analysis. However, current Handwritten Text Recognition (HTR) tools, such as Transkribus [1] and eScriptorium [2], require extensive labeled training data, rendering them ineffective for rare scripts with limited textual records. Additionally, uncommon scripts often lack Unicode representation, complicating their digital processing.

The importance of interdisciplinary collaboration has gained recognition, particularly in projects such as DECRYPT (Megyesi et al. 2020), which integrates expertise from computational linguists, cryptologists, computer vision specialists, philologists and historians to digitize encrypted documents and develop transcription and cryptanalysis tools. These efforts underscore the necessity of combining traditional scholarship with digital advancements to unravel historical texts.

[1] https://www.transkribus.org

[2] https://escriptorium.inria.fr

Despite persistent challenges, the integration of computational techniques with historical expertise continues to advance the field of decipherment, offering new opportunities for understanding ancient and encoded texts.

3 Objectives

The overarching goal of the DESCRYPT program is to bridge the gap between traditional philological methods and state-of-the-art computational tools. The program is guided by several key objectives:

- **Developing Decipherment Techniques:** Innovate methods for analyzing and interpreting rare scripts, moving beyond manual or semi-automatic approaches.
- **Creating a Digital Corpus:** Assemble a repository of digitized rare scripts, enriched with standardized metadata for preservation and accessibility.
- **Designing Recognition Models:** Develop algorithms capable of transcription and analysis of undeciphered writing systems, ensuring scalability and adaptability.
- **Enhancing User-Centric Research Tools:** Build robust, user-friendly AI tools that integrate feedback from experts, enabling continuous improvement in transcription and analysis accuracy.
- **Fostering Interdisciplinary Collaboration:** Unify expertise from linguistics, cryptanalysis, computer vision, history, and archaeology to create a comprehensive research framework.
- **Preserving Cultural Heritage:** Make rare and historically significant texts accessible and comprehensible, contributing to the understanding of human civilization.

4 Research Questions

The research is framed around questions that address both methodological and technological challenges:

- How can innovative methodologies enhance the analysis and decipherment of rare and unknown scripts?

- What image processing techniques can be adapted to automate the transcription of historical writings across diverse writing systems?
- What strategies can ensure systematic and consistent transcription of symbols from varying notational styles?
- How can interactive platforms incorporate user feedback to refine transcription and decipherment tools?
- What improvements can be made to historical language models to facilitate accurate script identification and interpretation?

These questions guide the program's research design, ensuring that theoretical insights and practical applications are seamlessly integrated.

5 Methodology

The DESCRYPT framework incorporates advanced technologies and traditional expertise to systematically address rare scripts. Its key methodological components include collection, transcription, decipherment and historical and linguistic analysis, as illustrated in Figure 1.

The DESCRYPT program is structured into five work packages (WPs), following a pipeline from the collection of historical sources with rare or unknown scripts to their transcription, decipherment and linguistic and historical analysis. Each component plays an essential role in advancing decipherment techniques through interdisciplinary collaboration.

WP1: Collection and Digitization The first phase focuses on identifying, gathering, and digitizing historical sources from archives, libraries, catalogs, and private collections. High-resolution imaging techniques are used to create digital copies ensuring the best possible preservation and analysis. The sources are then annotated with metadata using the TEI XML standard. To make them accessible for further study, metadata is added in compliance with TEI XML (Consortium, 2023) standards and Linked Open Data (LOD) (Gaitanou et al., 2022), enabling structured documentation and retrieval.

The aim is to create a comprehensive, searchable database integrating rare and undeciphered scripts, including ancient writing systems (e.g., Linear A and B, the Phaistos Disk), historical shorthand systems, early modern artificial language schemes, and encrypted texts. The collection ensures systematic access to digital reproductions of rare scripts while linking to existing scholarly resources.

WP2: Historical Language Models This phase develops historical language models for script recognition, transcription, and interpretation. It involves collecting and standardizing diplomatic transcriptions of historical languages, training models for language identification, and refining them through integration with image processing and cryptanalysis techniques. The corpus includes languages such as Akkadian, Ancient Greek, Classical Arabic, Latin, Old Norse, and Sanskrit. Addressing challenges like non-standard spellings, dialectal variations, and code-switching, the project aims to optimize algorithms for linguistic analysis and transcription correction. One of the key challenges in this process is handling non-standard spellings, dialectal variations, and instances of code-switching, which complicate linguistic modeling and text interpretation.

WP3: Transcription and Image Processing Transcription and image processing play a crucial role in decipherment. Digital images undergo preprocessing techniques to enhance readability, making inscriptions and scripts more legible. Additionally, layout analysis, text segmentation, and symbol recognition algorithms are implemented to automate transcription and streamline the decoding process. To improve readability and transcription accuracy, this phase develops advanced image processing techniques. Methods for preprocessing, document layout analysis, text segmentation, and symbol recognition are implemented to automate transcription for various writing systems. Given the limitations of current handwritten text recognition (HTR) models for rare scripts, a hybrid approach combining computational recognition with user input is developed based on the CTTS tool (Lasry et al., 2023). The outputs include standardized transcription methods, enhanced image analysis tools, and a user-friendly transcription system that can adapt to new sources.

WP4: Interactive Decipherment Platform A critical component of DESCRYPT is the development of an interactive platform that facilitates user engagement in decipherment. The platform integrates AI-driven cryptanalysis, allowing users

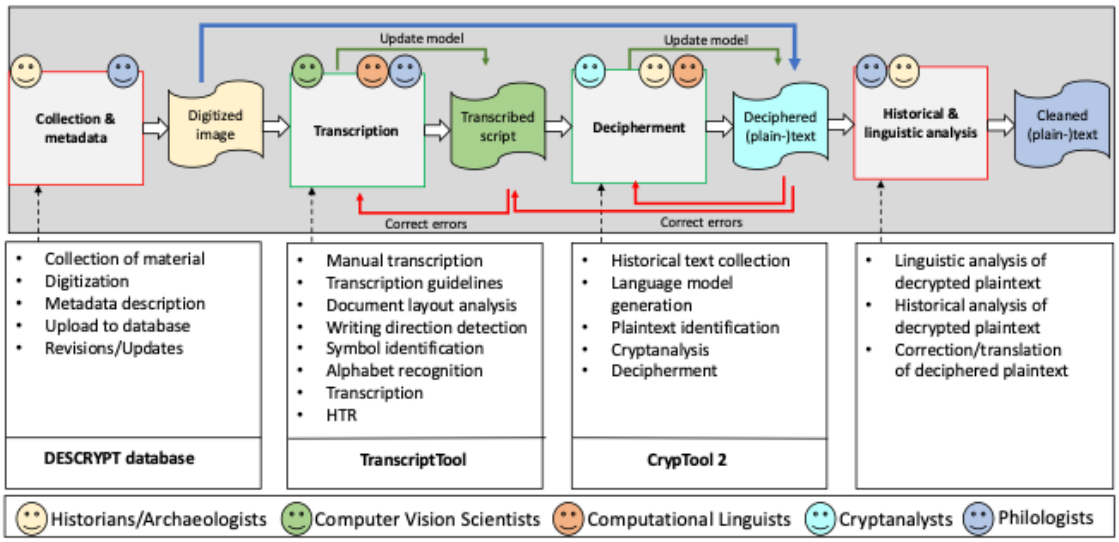


Figure 1: DESCRYPT: methodological overview.

to input corrections, enhancing transcription and decipherment accuracy. Drawing on historical language models and cryptographic algorithms, the platform balances automated analysis with expert insights, allowing users to contribute corrections, thereby improving the accuracy and efficiency of decipherment. The goal is to create a semi-automatic system that combines computational tools with human expertise to decipher rare and encrypted scripts more effectively.

WP5: Testing, Validation, and Refinement The final phase involves testing the developed tools on diverse scripts, evaluating their effectiveness, and refining them based on user feedback. A user-friendly interface and a structured framework ensure accessibility and adaptability. Large-scale experiments validate the reliability of transcription and decipherment models, while continuous improvements enhance the overall system. Large-scale experiments are conducted to assess the scalability and effectiveness of the approaches, ensuring that they can be applied to a wide range of historical texts and encoded scripts. Diverse scripts are used to validate the methodologies, and user feedback is collected to refine processes. Evaluations of interdisciplinary collaboration help identify best practices for integrating computational methods with traditional philological research.

DESCRYPT's success hinges on collaboration across multiple disciplines. Computational linguistics plays a key role in developing historical language models, ensuring accurate representation of rare scripts. Experts in history and philology contribute contextual knowledge and linguistic expertise, aiding in the interpretation of texts. Paleographers provide critical insights into historical handwriting styles, script evolution, and scribal practices, which are essential for the accurate reading and dating of manuscripts. Computer vision specialists enhance image processing and text recognition capabilities, making inscriptions and manuscripts more accessible for study. Cryptologists apply advanced cryptographic techniques to assist in deciphering encoded and unknown scripts. By bringing together leading researchers in archeology, history, linguistics, paleography, computer science, and cryptanalysis, the program fosters a balanced interdisciplinary approach, enabling more effective analysis and interpretation of historical texts to develop a holistic framework for deciphering rare and unknown writing systems.

6 Conclusion

DESCRYPT pioneers a transformative approach to historical text analysis by combining computational methods with traditional expertise. Its interdisciplinary framework aims to contribute to the study of rare and unknown scripts, preserving cultural heritage and enhancing our understanding of human history. By advancing methodologies for transcription, recognition, and interpretation, DESCRYPT paves the way for future innovations in the humanities and beyond. The program's outcomes will benefit multiple disciplines, including

linguistics, history, and computer science, while providing tools for the broader academic community. Ultimately, DESCRYPT demonstrates the power of interdisciplinary collaboration in uncovering the secrets of the past and safeguarding them for future generations.

Acknowledgments

This work has been supported by Riksbankens Jubileumsfond, grant M24-0028: Echoes of History: Analysis and Decipherment of Historical Writings (DESCRYPT).

References

Nada Aldarrab. 2017. Decipherment of historical manuscripts. Master's thesis, University of Southern California.

Yannis Assael, Thea Sommerschield, Brendan Shillingford, Mahyar Bordbar, John Pavlopoulos, Marita Chatzipanagiotou, Ion Androutsopoulos, Jonathan Prag, and Nando de Freitas. 2022. Restoring and attributing ancient texts using deep neural networks. *Nature*, 603:280–283.

Svenja Bonmann, Jakob Halfmann, Natalie Korobzow, and Bobomullo Bobomulloev. 2023. A partial decipherment of the unknown kushan script. *Transactions of the Philological Society*, 121:293–329.

John Chadwick. 1958. *The Decipherment of Linear B*. Cambridge University Press.

Jean-Francois Champollion. 1822. *Lettre à M. Dacier relative à l'alphabet des hiéroglyphes phonétiques*.

Michael D. Coe. 2011. *Breaking the Maya Code*. Thames Hudson Ltd, 3rd edition.

The TEI Consortium, 2023. *Guidelines for Electronic Text Encoding and Interchange P5 Version 4.7.0*. Last updated on 16th November 2023.

Francois Desset, Kambiz Tabibzadeh, Matthieu Kervran, Gian Pietro Basello, and Gianni Marchesi. 2022. The decipherment of linear elamite writing. *Zeitschrift für Assyriologie und vorderasiatische Archäologie*, 112(1):11–60.

Panorea Gaitanou, Ioanna Andreou, Miguel-Ángel Sicilia, and Emmanouel Garoufallou. 2022. Linked data for libraries: Creating a global knowledge space, a systematic literature review. *Journal of Information Science*, 50:204 – 244.

Andrew George and Manfred Krebernik. 2022. Two remarkable vocabularies: Amorite-Akkadian bilinguals! *Revue d'assyriologie et d'archéologie orientale*, 116:113–166.

Bradley Hauer and Grzegorz Kondrak. 2016. Decoding anagrammed texts written in an unknown language and script. In *Transactions of the Association for Computational Linguistics*, volume 4, pages 75–86. MIT Press.

David Kahn. 1967. *The Codebreakers*. New York, 2nd edition.

Kevin Knight, Beáta Megyesi, and Christiane Schaefer. 2012. The copiale cipher. In *ACL Workshop on Building and Using Comparable Corpora (BUCC)*.

George Lasry, Beáta Megyesi, and Nils Kopal. 2020. Deciphering papal ciphers from the 16th to the 18th century. *Cryptologia*, pages 479–540.

George Lasry, Norbert Biermann, and Satoshi Tomokiyo. 2023. Deciphering Mary Stuart's lost letters from 1578–1584. *Cryptologia*.

Beáta Megyesi, Bernhard Esslinger, Alicia Fornés, Nils Kopal, Benedek Láng, Georg Lasry, Karl de Leeuw, Eva Pettersson, Arno Wacker, and Michelle Waldispühl. 2020. Decryption of historical manuscripts: the decrypt project. *Cryptologia*.

Katerina Papavassileiou, Dimitrios Kosmopoulos, and Gareth Owens. 2023. A generative model for the mycenaean linear b script and its application in infilling text from ancient tablets. *Journal on Computing and Cultural Heritage*, 16.

A new attack on the mysterious inscription of Santa Maria La Nova

Cosimo Palma
University of Naples "L'Orientale"
University of Pisa, Italy
cosimo.palma@phd.unipi.it

Yll Rugova
University of Prishtina
Kosovo
yll.rugova@trembelat.com

Paolo Bonavoglia
Mathesis Venezia c/o Liceo Foscarini
Italy
paolo.bonavoglia@liceofoscarini.it

Abstract

Expanding upon the established hypothesis of monoalphabetic substitution with potential transposition and polyalphabetic elements, this analysis of the Santa Maria la Nova epigraph incorporates Ancient Greek, Old Church Slavonic, Old Romanian and Old Albanian, thus exploring the possibility that the cipher's plaintext derives from historically under-examined languages, particularly those with cultural and historical ties to medieval Naples and its Eastern Mediterranean networks. Special attention is given to aligning the analyzed corpora with the epigraph's actual textual rendering and to the evaluation of multilingualism.

1 Introduction

After publishing the preliminary cryptoanalysis of the ciphered epigraph in the Turbolo Chapel at Santa Maria la Nova (Palma, 2023), the public interest in this fascinating artifact has been further stirred by the production of a novel documentary that revolves around the same places of the inscription (Repubblica, 2024). Despite the failed decryption attempt, the attention reclaimed from the scientific community, whose main outcome is represented by this contribution, has been positively responded to.

In the "Future Work" section of (Palma, 2023), checking for magical/esoteric words or conlangs (such as Hildegard's *Lingua Ignota*, magical papyri, gnostic libraries and other books about occultism and alchemy) was proposed as a possible approach to carry on the investigation. Format constraints necessitate postponing this approach in favor of a more promising research direction. The present work focuses on monoalphabetic substitution attacks in relevant languages, prioritizing initial decryption stages:

1. Obtaining the clearest possible transcription based on the most recognizable inscription segments;
2. Selecting historically grounded candidate languages;
3. Performing attacks on smaller epigraph portions before attempting alternative methods, considering potential multilingualism.

2 Data Preprocessing and preliminary evaluation

To comply with the first listed principle, a manual transcription has been newly performed, as verifiable in Figure 2.

In addition, the CTTS transcription tool[1] has been used to provide an aided transcription (see Figure 1).

One of its output, Figure 3, helped us notice that some glyphs cluster overwhelmingly in one side (top or bottom) of the inscription. For instance, the sixth stripe from right, representing the frequency and position of the glyph  in the epigraph, clearly shows it only appears in the first half of the inscription.

This observation strengthens the hypotesis that some glyphs can be homophones, an hypotesis that is difficult to reconcile with the character's frequency distribution, which fully aligns with a pure monoalphabetic substitution cipher.

An analysis of digrams composed of most frequent characters (as shown in Figure 4) opens a new avenue for investigation, as every language must display a costant behavior for digrams as well as for their opposite (obtained by switching the letters).

In the following sections, four languages have been more thoroughly investigated because of their promising features.

[1] https://github.com/CrypToolProject/CTTS/tree/main.

Proceedings of the 8th International Conference on Historical Cryptology, pages 106-110
Poznań, Poland, 16-18 June, 2025

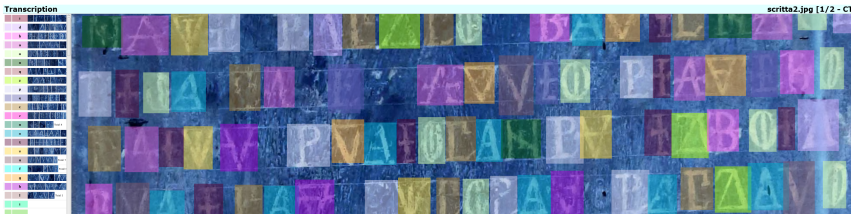


Figure 1: A snapshot of the encrypted epigraph of the Turbolo Chapel in the Neapolitan Church of Santa Maria La Nova from the transcription environment CTTS.

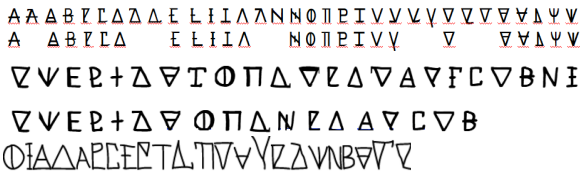


Figure 2: Comparison of characters transcription from the epigraph performed independently by the authors (the first two providing a minimal and extended character's interpretation; the latter a single one.)

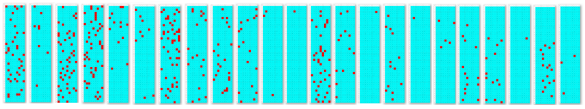


Figure 3: A visualization capturing both position and frequency of each inscription's glyph.

3 In the footsteps of Kostantin Castrioti: the Old Albanian lead

After the death of Scanderberg, many Albanians migrated to Italy; part of them, to Naples (Schmitt, 2014). The church of Santa Maria la Nova in Naples is important for Albanological studies precisely for this reason. Within its premises, for example, is located the grave of Konstantin Kastrioti (d. 1500), nephew of Gjon Kastrioti (Skenderberg) — Albanian national hero (Ciarlanti, 1644; Padiglione, 1879). The dedication below the sculpture on the sarcophagus clearly states that it was raised by Donika Kastrioti, Skenderbeg's wife, in 1500, for commemorating the untimely death of her nephew (Amarelli, 2012). When in the summer of 2024 a delegation of the Academy

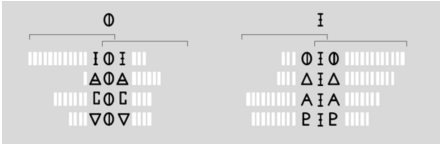


Figure 4: A diagrammatic sample of most frequent digrams.

of Science of Kosovo visited Napoli, they were informed that the grave of Konstantine Kastrioti was at one point somewhere in the Turbolo chapel, where the undeciphered inscription takes up a corner of a wall. Such changes within the church seem to have been a common occurrence (Forgione, 2023). In any scenario, this would mean that the mysterious inscription could belong to a period before the Turbolo family restructured the chapel (1582), maybe even to the year 1500 when Konstantine's grave was placed there.[2]

Until the late 17th century, Albanian was mainly written using the Latin script. The oldest Albanian written text is an inscription with Latin letters, found on a manuscript dated 1462. It is a well known short (Elsie, 2017) word baptism formula originally from Durrës in Albania. The oldest book written in Albanian is from 1555, known as *Meshari* (Mass book), a book printed using Venice typographic materials surviving in a single copy kept in the Vatican Library in Rome. This book was also written in Latin script, with 4 additional symbols mainly borrowed from cyrillic *bosancica* script. Though a manuscript from 1400 mentions an original alphabet for Albanian texts (Schmitt, 2014), no traces of such a script were witnessed until the end of the 17th century (Rugova, 2022). A certain manuscript called *Anonimi i Elbasanit*, dated to be from the end of the 17th, or the middle of the 18th century is the first known text in Albanian with an original alphabet (Elsie, 2017). In subsequent decades several other such original scripts emerged, mainly in what is known as the orthodox milieu of the Albanian speaking communities [ibid.]. One of them seems to have been shared by the Arumanian speaker of Albania as well (Demiraj, 2019). No similarities can be spotted between these original scripts and the one of the Napoli inscription. However, given the

[2] A work from Simone Papa (Il Vecchio) in 1488 is one example of works finished there before the Turbolo family made their changes in 1580s (Bryan, 1849).

plurality of different Albanian scripts mentioned above, another one from 1500 would indeed be a delightful discovery for Albanian studies, but not a surprising one. That, together with statistical cues–as shown in (Palma, 2024), was the reason we used an Albanian corpus to try to decipher the Napoli inscription. For this purpose, given the closeness to Konstantin's period, the Meshari of Gjon Buzuku was used as the main Albanian corpus.

It has been harvested from the Titus-Project (Gippert, 1996)[3] and processed by means of the cryptology suite *Malanova*[4] previously developed as codebase for (Palma, 2023). Enlarging the corpus with further documents has been momentarily avoided, given the exploratory nature of the analysis. Moreover, every corpus in Old Albanian presents slightly different scripts and even vocabularies, which would have posed an issue of alignment that cannot be tackled in the limited scope of this survey. Using the software *AZDecrypt*[5], loaded with the 5-grams generated from the preprocessed corpus, an attack has been run using the *Substitution* as well as the *Substitution Skips and Nulls* functions, respectively loading all different epigraphy transcriptions as described in section 2. Every test has been run setting different levels of entropy: 1, 5 and 10, each time for the whole epigraph, then the upper, and then the lower half.

4 Tracking down the Balkan way: failed attempts with Old Romanian and Old Church Slavonic

Previous statistical analysis have also shown a good fit between the character distribution of the inscription with Old Church Slavonic and Old Romanian. This correlation has been further confirmed by calculating the *Entropy* and the *Index of Coincidence* (Friedman, 1922). The brevity of the text does not allow to be overly optimistic about such metrics, that can be reliable only on larger portions of text. However, it cannot be neglected that these encouraging values have been measured only for these languages, which by chance corresponds to the language whose alphabets feature in the inscription, and are in the closest geographical area surrounding the one where Old Albanian

Carattere	Frequenza	
	Assoluta	Relativa.
𐌰	49	13.0 %
𐌴	45	11.9 %
𐌱	41	10.8 %
𐌹	39	10.3 %
𐌿	28	7.4 %
𐌻	25	6.6 %
𐍅	18	4.8 %
𐌽	17	4.5 %
𐌳	16	4.2 %

- *lettere*	
A	11.94 %
O	9.87 %
I	9.61 %
T	8.77 %
E	8.58 %
Σ	7.84 %
N	5.90 %
H	4.96 %
P	4.82 %

Figure 5: Comparison of the frequencies of the cipher and those of ancient Greek

was spoken. The very same approach as described in section 3 has been applied for these languages. The related corpora have been harvested from the Language Bank of Finland *Kielipankki*[6] and from (Niculescu and Dimitrescu, 2000) for Old Romanian, by means of OCR digitization[7].

5 It may be Greek, after all

As mentioned in (Rocco, 1928), the author assesses that the inscription is the Greek translation of the adjacent Latin inscription. Although even a profane would easily acknowledge that the first epigraph is too long, compared to the second one, there are indeed many signs similar to Greek letters (Α Β Δ Ε Ι Λ Ν Ο Π Ρ Τ Υ ψ Φ) appearing in the epigraph. If we also consider that father Rocco was esteemed as a cultivated literate, this kind of mistake would sound excessively superficial. What if he knew something he could not openly disclose? Another element worth considering is the final sign of each line. Of the 20 counted lines, 9 ended with Δ, 6 with A and 5 with I. This is quite unexpected; most languages have words ending with a large number of different letters. Conversely, in Italian almost all words end with the four vowels A O I E. And in Greek they end in a very large part with A, H or Σ.

In classical cryptanalysis, the first step after frequency analysis, is identifying the vowels. In addition to the individual letters, the frequency of digrams is very useful. The vowels agree with almost all consonants, while the consonants are more quarrelsome; calculating the entropy[8] of the

[3] https://titus.uni-frankfurt.de/texte/etcs/alban/buzuku/buzuk.htm

[4] https://github.com/Glottocrisio/MaLaNova

[5] https://github.com/doranchak/azdecrypt

[6] https://www.kielipankki.fi/corpora/ccmh/

[7] We thankfully acknowledge Francesco Pastore for providing this resource.

[8] About entropy and cryptanalysis see (Bauer, 2007)

digrams of each letter, the vowels have a value of about 4, while the consonants have values under 3.5 except for R (P in Greek) which has values slightly lower than the vowels. Hence, it turns out that the signs similar to O, I, A, are almost certainly vowels. The Δ sign also seems to be a vowel. However, even not knowing the original language, these data may be useful, because most frequencies of the same letter are not so different: this case is a good example. Assuming Latin as the language and the highest frequencies, the word BASILEIA, which does not exist in Latin, turned up in the first complete line, while ΒΑΣΙΛΕΙΑ exists in ancient and modern Greek. Putting aside the Latin language and moving on to Greek, something very interesting emerges: a whole line can be deciphered according to this hypothesis . After ΒΑΣΙΛΕΙΑ there are four signs ;A;Σ and at the end of the line, there are four more characters: E;IA , resulting in this partially decrypted line:

ΒΑΣΙΛΕΙΑ ΝΑΥΣ..Α Ι ..ΕΡΙΑ

Later, a name from the most remote school memories (the Odyssey) emerges: ΝΑΥΣΙΚΑ Nausicaa the daughter of the king of the Phaeacians who welcomes the shipwrecked Ulysses. In reality between Σ and A there is a practically empty space, two or three letters erased. If we assume that they are IK it comes out precisely . Later we found that the island of the Phaeacians, today identified with Corfu, was called in ancient times ΣΧΕΡΙΑ![9] giving:

ΒΑΣΙΛΕΙΑ ΝΑΥΣΙΚΑ.ΣΧΕΡΙΑ

Three semantically related concepts appearing in the same row strongly suggest a pattern. However, this hypothesis faces these challenges:

1. Up to now trials based on this same substitution pattern to the remaining text, did not yield meaningful texts;
2. Another row contains a seemingly meaningful Greek phrase, but with inconsistent character mappings for at least two glyphs.

A plausible explication of these problems is the use of multiple languages. While this would significantly complicate decryption, it's supported by the emergence of plausible Greek syllables throughout the text, consistent with typical Greek letter distribution patterns.

[9] An homophone was used here for Σ; not so strange, because Σ has also the ς variant and the sum of frequencies is almost the same as in Greek.

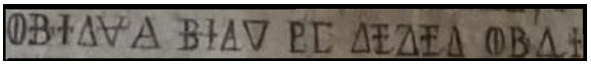


Figure 6: A detail of row 22, showing the only palindrome word of the whole inscription.

The twenty-second row of the inscription contains the only palindrome in the text (see Figure 6). Among all possible five-letter palindromes in Koiné Greek, only ρητηρ (rhetor) is consistent with the character frequencies shown in Figure 5. This line is particularly notable for its repeated glyphs. Manual decryption through character substitution yields: **ἀγορεύ(-) γονυ δέ ῥητήρ ἀγνό**, meaning "... to harangue on one's knees, but rather as a venerable rhetor".

6 Final remarks and Future Work

In cryptanalysis, the successful decryption of consecutive meaningful words typically enables complete reconstruction of the cipher through systematic steps. However, this case presents unusual challenges. Several critical questions emerge: is the current decryption attempt entirely incorrect? Is this particular line accurately deciphered while other sections follow different patterns? Does the inscription employ multiple languages? Automated analyses using Old Romanian and Old Albanian corpora have not yielded meaningful results[10]. Given the epigraph's manageable size, manual cryptanalysis appears to be a more promising approach. While definitive solutions remain elusive, several avenues for future investigation present themselves. The most promising directions include:

1. Systematic testing of Old Albanian dialectal variations;
2. Deepening the Greek lead, trying to exploit the "crib grid" function of AZDecrypt to produce constrained output based on partial decryptions achieved from manual attempts;
3. Involving in the quest experts in the candidate languages.

[10] The output files of the automatic attacks using AZDecrypt are stored in the codebase: `https://github.com/Glottocrisio/AZDecryptMariaLaNova/tree/main/main/Output`. Results from the present work are stored in the folders "gre", "rom", "alb" and "slav".

References

Marco Amarelli. 2012. Costantino e La Casa Castriota. Nuovi Contributi Sulla Biografia e Gli Scritti Di «Filonico Alicarnasseo». *Critica Letteraria*, 40(154):121.

Friedrich Ludwig Bauer. 2007. *Decrypted secrets: Methods and Maxims of Cryptology*. Springer, Berlin.

Michael Bryan. 1849. *A Biographical and Critical Dictionary of Painters and Engravers, from the Revival of the Art... to the Present Time...* H. G. Bohn, London.

Giovanni Vincenzo Ciarlanti. 1644. *Memorie Historiche Del Sannio Chiamato Hoggi Principato Vltra, Contado Di Molisi, e Parte Di Terra Di Lauoro, Prouincie Del Regno Di Napoli*. Camillo Cauallo, Isernia.

Bardhyl Demiraj. 2019. Enigma e Gravurës Së Ardenicës. *Shejzat-Pleiades*, (1–2):121–125.

Robert Elsie. 2017. *Albanian Alphabets: Borrowed and Invented*. Centre for Albanian Studies, London.

Alessandro Forgione. 2023. Dracula é sepolto a Napoli? *Fenix*.

William F. Friedman. 1922. The index of coincidence and its applications in cryptanalysis. *Riverbank Publication*, 22.

Jost Gippert. 1996. Titus - alte und neue perspektiven eines indogermanistischen thesaurus. *Studia Iranica, Mesopotamica et Anatolica*, 2:49–76. Published in 1997.

A. Niculescu and F. Dimitrescu, editors. 2000. *Testi romeni antichi (secoli XVI-XVIII)*. Vulgares eloquentes. Antenore.

Carlo Padiglione. 1879. *Di Giorgio Castriota Scanderbech e de' Suoi Discendenti: Narrazione*. Stab. Tipografico del Cav. Francesco Giannini, Napoli.

Cosimo Palma. 2023. Encrypted epigraphy - the case of a mysterious inscription in the neapolitan church of santa maria la nova. In *International Conference on Historical Cryptology*.

C. Palma. 2024. Epigrafi cifrate nelle chiese antiche. In *ERUDITORUM ACTA 2024*, volume 3 of *De Cifris Koine*, pages 35–51. De Cifris Press, September. Cosimo Palma, 0000-0002-8161-9782.

La Repubblica. 2024. Vampiri per caso, la vera storia di dracula a napoli: la prima del film di syusy blady. *la Repubblica*, October. Online newspaper article.

G. Rocco. 1928. *Il convento e la chiesa di Santa Maria La Nova*. Tipografia Pontificia degli Artigianelli, Napoli.

Yll Rugova. 2022. Dëshmi Mbi Një Alfabet Origjinal Të Shqipes Nga Shekulli XVII: Kozmai i Durrësit Si Shpikës i Mundshëm i Alfabetit Të Parë Origjinal Të Shqipes. *Shejzat-Pleiades*, 7(3–4):21–37.

O. J. Schmitt. 2014. *Skënderbeu, Rishtypje e zgjeruar*. Fjala Publishing, Tiranë.

DECODE2LOD: Connecting the DECODE Database with the Linked Open Data Cloud

Cosimo Palma
University of Naples "L'Orientale"
University of Pisa, Italy
cosimo.palma@phd.unipi.it

Beáta Megyesi
Department of Linguistics
Stockholm University, Sweden
beata.megyesi@ling.su.se

Abstract

This paper presents a novel approach to enhancing the analytical power and interoperability of historical cryptology data by transforming the DECODE database into a Linked Open Data (LOD) resource. We introduce a methodology for modeling encrypted historical documents and cipher keys as a knowledge graph, encompassing ontology development, data transformation, and SPARQL-based querying. This integration enables complex queries across domains, encourages collaboration beyond cryptology, and aligns DECODE with broader efforts in digital humanities and open science. By bridging historical cryptology with LOD principles, we offer a scalable framework for enriching specialized research databases through semantic technologies.

1 Introduction

Historical cryptology, like all scientific fields, depends on robust research infrastructure. This infrastructure typically includes systematic, well-curated collections of diverse data sources described through metadata, alongside tools designed to process and analyze the data. The DECODE database (Megyesi et al., 2019; Héder and Megyesi, 2022), with its repository of nearly 10,000 records at the time of writing, has become the de facto standard resource for historical cryptology research. Each record in DECODE—whether a ciphertext, cipher key, or manual—is enriched with extensive metadata describing the source's current location, origin, content, and additional analyses. As a searchable, evolving monitor corpus, DECODE continuously incorporates new data, facilitating numerous studies in historical cryptology.

Beyond its use by historical cryptologists, the DECODE database holds significant potential for researchers in adjacent fields, such as Digital History. However, to fully leverage the database's richness and to enable more sophisticated, granular analysis, a technological advancement in its integration is necessary. This paper addresses this need by proposing the integration of the DECODE database with Linked Open Data (LOD), enabling advanced querying, fostering interdisciplinary research, and supporting interoperability with external datasets.

2 Digital History and Linked Open Data

The decipherment of encrypted manuscripts has often provided new insights into history, reshaping our understanding of the past. A recent example is the decryption of Mary Stuart's encrypted letters (Lasry et al., 2023), which offer remarkable glimpses into her life as a prisoner under the watch of the Earl of Shrewsbury. Furthermore, each deciphered text must be examined within its historical and cultural framework, taking into account the time period, geographic location, and sociopolitical context.

Interdisciplinary collaboration among scholars from different fields enriches the decipherment process in multiple ways. Historians and linguists, for instance, can provide cryptologists with crucial contextual insights and help interpret the paratextual elements of ciphers (Megyesi et al., 2020). This is particularly important when encrypted texts go beyond a simple sequence of characters and take the form of complex artifacts embedded within a specific historical and cultural setting, as discussed in (Palma, 2023).

Digital history is increasingly dependent on computational tools and methodologies for analyzing historical data, often utilizing structured databases, text mining, and topic modeling to generate informative visualizations. Among these

Proceedings of the 8th International Conference on Historical Cryptology, pages 111-121
Poznań, Poland, 16-18 June, 2025

tools, LOD has emerged as a key approach for connecting disparate datasets through semantic relationships, thereby enhancing both discoverability and interoperability.

One of LOD's strengths is its support for federated queries, which allow users to retrieve and integrate information from multiple distributed databases in a single query, eliminating the need for centralized data storage. This capability improves data granularity and accessibility, enabling researchers to explore interconnected historical sources more effectively. By facilitating seamless cross-database exploration, LOD makes historical research more interactive, integrative, and comprehensive.

The concept of Linked Data refers to a set of best practices for publishing structured data on the Web. These principles were introduced by Tim Berners-Lee in his Linked Data design issue note[1], and reformulated through the FAIR principles– *Findability*, *Accessibility*[2], *Interoperability*, and *Reusability*—-which aim to ensure that data is structured, discoverable, and reusable. To align with these principles, the dataset has been modeled using Uniform Resource Identifiers (URIs) for each record, ensuring unique identification and, where possible, aligning properties with existing ontologies to promote interoperability.

Despite DECODE's extensive metadata framework and its proven utility, the database had yet to be integrated into the LOD cloud, thus enabling for more complex and scalable analyses. By aligning DECODE with LOD principles, researchers can unlock deeper insights into encrypted sources and connect historical cryptology with broader datasets in cultural heritage and other domains. To address this gap, this study pursues the following objectives:

- Creating an **ontology** for DECODE that mirrors its existing metadata.
- **Populating** the ontology with DECODE's current dataset, ensuring semantic alignment.
- Designing and demonstrating example **SPARQL queries** that showcase the enhanced analytical capabilities enabled by LOD integration.

The following questions guide this research in demonstrating the relevance and added value of the proposed approach:

- How does this data representation improve upon the existing metadata framework?
- What new insights and research opportunities can be derived from SPARQL queries applied to the LOD-integrated DECODE database?

By addressing these objectives and research questions, this study seeks to advance the field of historical cryptology and establish a model for integrating domain-specific databases, particularly those that do not conform to tabular or relational data structures—into the broader LOD ecosystem.

3 From Relational to Graph Databases: Modeling DECODE for the Semantic Web

A brief account on how to model a database containing a collection of ciphers has been already provided by Bonavoglia (Bonavoglia, 2023).

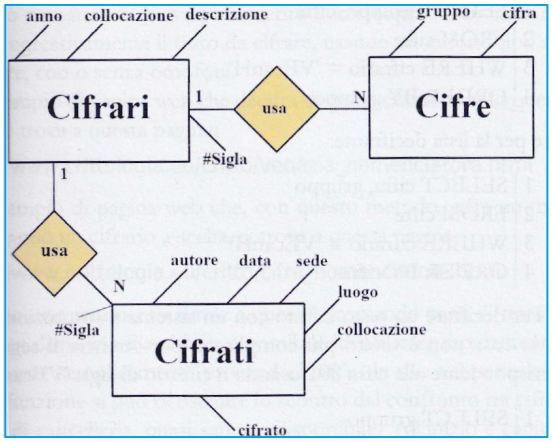


Figure 1: An overview of tables and properties in the relational database proposed by Bonavoglia (Bonavoglia, 2023).

His model is based on a relational database in which entries are stored in tables linked by specific keys. The typical query language for relational databases is SQL, which retrieves information by manipulating the tables in which information is stored. For example, in Figure 1, "Encryption method" (*Cifrari*), "Ciphers" (*Cifre*) and "Cyphertexts"/records (*Cifrati*) would be represented as *tables*, and their properties as columns. Querying common elements between encryptions (top-right box) and encrypted texts (bottom box) would require running a query that performs a union over the encryption methods table.

[1] `https://www.w3.org/wiki/LinkedData`

[2] Currently, the Knowledge Graph developed by the authors is not published online, i.e. the related SPARQL query endpoint is not hosted on any web domain.

Relational databases have traditionally been favored for efficiently capturing relationships between entries, but they fall short when it comes to representing the detailed properties of those entries. In contrast, graph-based databases not only capture these relationships, but also define and store properties for each node. This allows them to represent complex, interconnected data more naturally and flexibly than relational databases, leading to higher performance and a more intuitive data model for many modern applications.

Although graph databases offer advantages in performance, flexibility, and scalability, these benefits are minimal for a medium-to-small database like DECODE. However, their ability to support interoperability with other knowledge graphs—an essential feature in the rapidly evolving field of Digital Humanities—is uniquely facilitated within a LOD framework.

The literature presents various methods for converting datasets into Resource Description Framework (RDF)[3] format (Klyne and Carroll, 2004), regardless of whether the source is a structured spreadsheet (Fiorelli et al., 2015), a natural language description (di Buono et al., 2014), or a more heterogeneous data collection (Dimou et al., 2014).

To ensure compatibility and semantic integration, the DECODE Knowledge Graph is aligned with CIDOC-CRM (Doerr, 2003), the de facto standard ontology for Cultural Heritage, as well as with more general ontologies such as Schema.org (Guha et al., 2016) and Dublin Core (Baker, 2000; Weibel and Koch, 1997).

4 Building DECODE2LOD

To make the DECODE database LOD-compatible[4], we present a workflow and describe the data collection process along with the steps involved in ontology development and semantic transformation. The workflow is illustrated in Figure 2.

Data acquisition from the current DECODE database consists of several interconnected functions that systematically retrieve and process records from a REST API[5]. The ontology population process, which structures the acquired data within a defined schema, includes three key components: 1) ontology initialization, 2) data harvesting and ingestion, and 3) semantic transformation. Each of these components are described in detail below.

4.1 DECODE to LOD Conversion

The conversion of the DECODE2LOD begins by initializing an RDF graph structure with a predefined name space (https://de-crypt.org/r/), incorporating an ontology from a Turtle (TTL) file. The Ontology has been previously crafted manually to both mirror the structure and metadata of a DECODE record shown in the record view of the database, as illustrated in Figure 3.

Initially, the system establishes a base URL for API communication. The primary function, ”fetch records”, constructs API endpoint URLs by combining the base URL with specific table identifiers and handles HTTP GET requests with pagination parameters. This function returns JSON-formatted responses or raises appropriate exceptions upon failure.

To gather the complete dataset, the ”fetch all records” function implements an iterative pagination mechanism, collecting all available records page by page.

Individual record details are accessed through the ”fetch specific record” function, which constructs endpoints for specific record IDs and retrieves detailed information for each record. The ”fetch all record details” function extends this capability by processing collections of records, systematically retrieving detailed information for each valid record ID while gracefully handling missing or invalid identifiers.

Data persistence is managed through the ”save to json” function, which serializes the collected data into JSON[6] format and writes it to specified

[3]As described in `http://w3.org/RDF/`, RDF is a standard model for data interchange on the Web. It facilitates data integration even when underlying schemas differ and supports schema evolution over time without requiring modifications from all data consumers.

[4]The codebase for this operation can be retrieved at `https://github.com/Glottocrisio/Decode2LOD/tree/main`

[5]REST provides a set of architectural constraints that, when applied as a whole, emphasizes scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems (Fielding and Taylor, 2002). The DECODE API can be accessed at: `https://de-crypt.org/decrypt-web/api`

[6]JSON (JavaScript Object Notation, pronounced /desn/ or /desn/) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of name–value pairs and arrays (or other serializable values). It is a commonly used data format

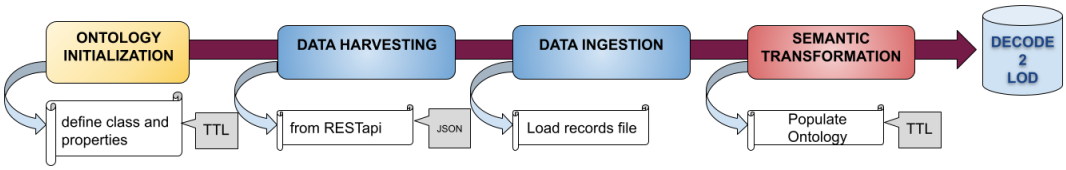


Figure 2: The workflow of converting DECODE to LOD.

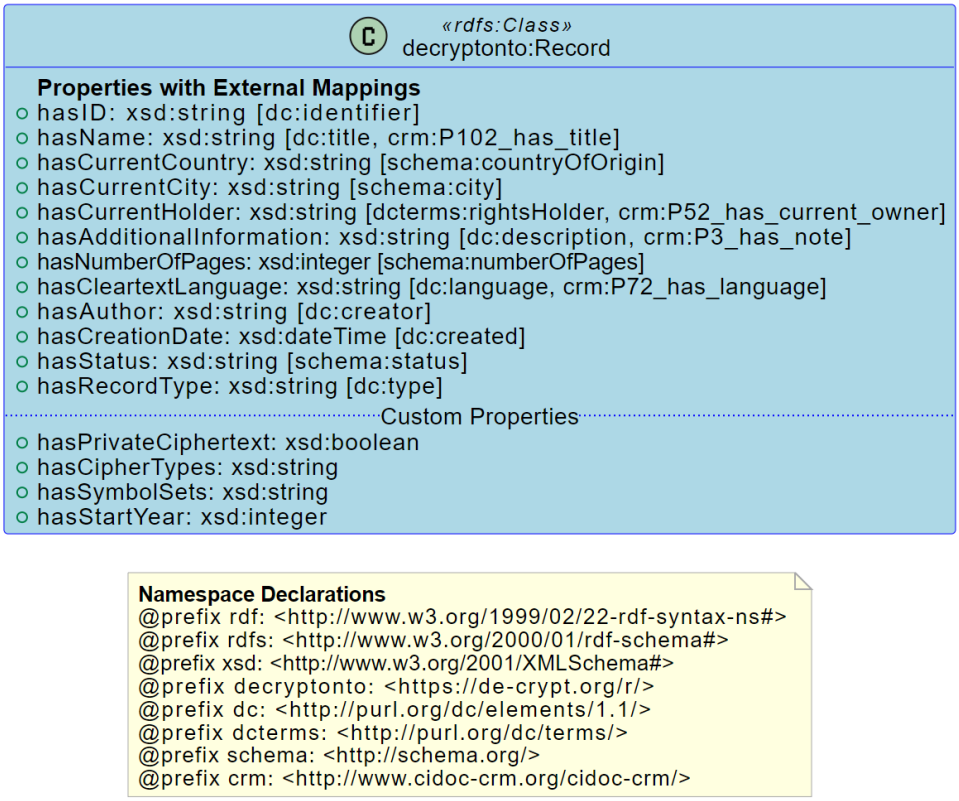


Figure 3: A specification of ”Record”, the only class in the Decrypt Ontology.

files. The main routine orchestrates these components by defining target tables and output files, coordinating the collection of both summary and detailed records, and ensuring proper error handling throughout the execution process.

The entire system implements error handling at multiple levels to manage potential failures in API communication, data processing, or file operations, ensuring robust operation in production environments.

The data acquisition phase implements a robust JSON parsing mechanism with built-in encoding fallback. This component initially attempts to parse the input file using UTF-8 encoding, and if unsuccessful, automatically retries with UTF-8-SIG encoding, ensuring compatibility with various text encodings that may be present in real-world datasets.

The core semantic transformation process maps structured data to ontological concepts through a comprehensive property mapping system.

Properties within the ontology are defined as *rdf:Property* types and are constrained to operate exclusively within the *decryptonto:Record* domain. Each property must explicitly declare its range, which is restricted to the following XML Schema Definition (XSD) data types: *string, integer, dateTime*, or *boolean*. Documentation

with diverse uses in electronic data interchange, including that of web applications with servers. (*Source: Wikipedia*)

for all properties is eventually provided through *rdfs:comment* annotations. Additionally, properties may be semantically linked to external ontologies through *rdfs:sameAs* mappings, enabling interoperability across different knowledge systems.

This mapping framework establishes precise correspondences between source data attributes and ontological predicates, while simultaneously defining appropriate XML Schema data types for literal values. The transformation process iterates through each record, creating unique URIs for individual entities and establishing their type relationships within the ontological framework. For each property in the mapping dictionary, the system performs datatype-specific transformations, handling special cases such as Boolean value normalization, Integer conversion, and DateTime parsing with ISO 8601 compliance.

Upon completion, the populated ontology is serialized in Turtle (.ttl) format, producing a semantically enriched representation of the original dataset.

4.2 Dataset Blueprint

The resulting DECODE Knowledge Graph comprises 25,970 nodes and 115,303 edges. The nodes include both the record themselves and the (both *literal-* and *object-*) values of every property, in this case displayed as edges (or relationships).

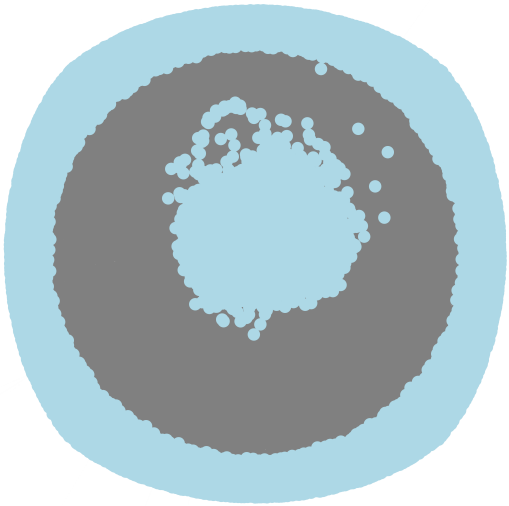


Figure 4: A distant view of the DECODE-LOD knowledge graph.

The average node centrality, calculated using the formula

$$\text{Average Centrality} = \frac{2 \times \text{Total Relationships}}{\text{Total Entities}}$$

is 8.88, meaning that each entity is, on average, connected to 9 other entities. Given that the dataset contains 9,390 records, an additional 16,580 nodes represent property values. As illustrated in Figure 4, individual records (represented by blue dots in the center) are not directly linked to one another but are instead connected to various properties (the dense cluster of blue dots at the periphery). The dark-grey zone represents the extensive network of relationships originating from these records—though visually indistinguishable due to graphical limitations. To enhance interconnections among record-entities, several strategies can be implemented, as it is discussed in Section 6.

5 Querying the Knowledge Graph

Visualizing and querying a knowledge graph are two distinct yet complementary processes. While visualization helps users explore the structure and relationships within the graph, querying enables the extraction of specific insights through a structured search.

5.1 Executing Queries

The Turtle file generated in the previous workflow cannot be used directly for querying. Although various online tools facilitate knowledge graph visualization[7], executing queries requires downloading and running appropriate software locally. One of the most popular choices for this purpose is Apache Jena Fuseki[8], known for its balance between user-friendliness and performance. As illustrated in Figure 5, setting up Apache Jena Fuseki involves downloading and extracting the package into a local directory, then running the ”fuseki-server” Windows batch file.

Once the server is running, the URL displayed in the command prompt window (highlighted in red in the figure) must be copied and pasted into a web browser. From there, the Turtle file containing the Knowledge Graph can be loaded, allowing users to execute SPARQL queries through the built-in SPARQL endpoint . Once the Turtle file is successfully loaded, users can dynamically explore and analyze the Knowledge Graph, lever-

[7]Figure 3 was generated using `https://semantechs.co.uk/turtle-editor-viewer/`, while Figure 4 was produced using `https://issemantic.net/rdf-visualizer`.

[8]Available for download at `https://jena.apache.org/download/index.cgi`.

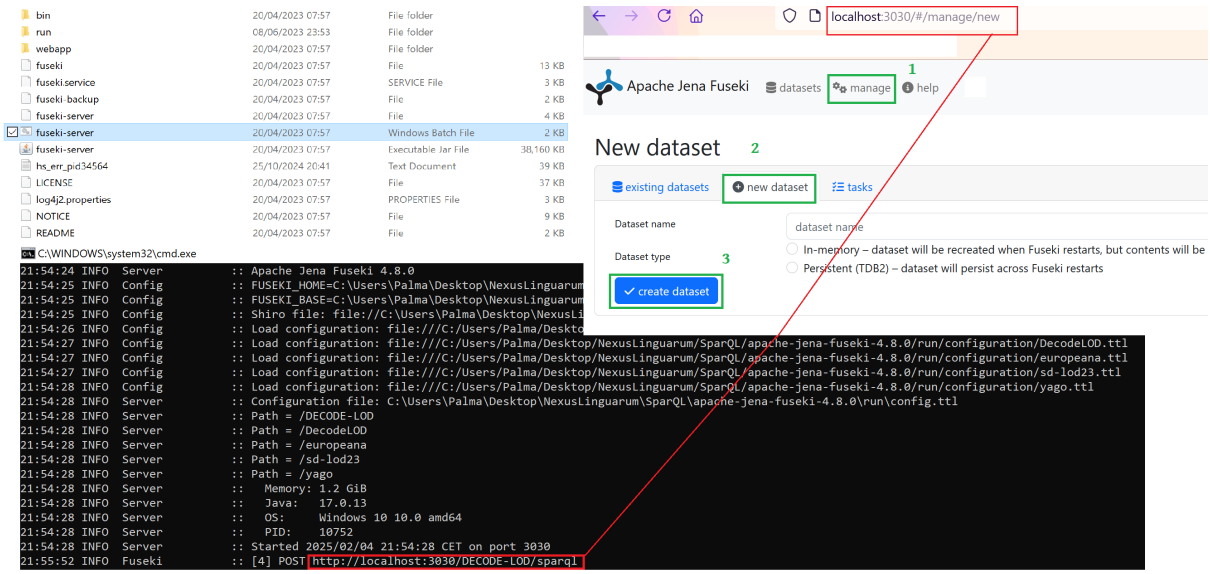


Figure 5: An overview of the Apache Jena Fuseki environment. The selection of the Windows Batch File will open a prompt featuring the number of port that needs to be added to ”localhost” in the browser search bar.

aging SPARQL queries to extract meaningful insights.

5.2 Example Queries

We illustrate the capabilities of the DECODE knowledge graph by presenting a series of advanced queries designed to explore historical cryptographic records. The primary user persona considered here is the *Digital Historian*, an historian using digital tools for exploring and gathering data about historical events. By leveraging SPARQL—the query language for RDF databases—we demonstrate how DECODE2LOD can facilitate nuanced investigations that would be challenging or impossible using traditional database search functionalities.

Some queries require federated search, where information from DECODE2LOD is cross-referenced with external knowledge bases such as DBpedia[9], a structured knowledge base derived from Wikipedia. This approach enables researchers to enrich their analyses with additional historical and contextual information. For example, the historian might want to *retrieve all unresolved ciphers from any European country that was also involved in a war during the seventeenth century.* The answer to the research question requires a federated query where the encrypted records are retrieved from the DECODE2LOD database, and are cross-referenced with DBpedia to gather additional historical context, such as wars in which a country was involved. The query is shown below for this particular example:

```
PREFIX rdf:
<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:
    <http://www.w3.org/2000/01/rdf-schema#>
PREFIX decryptonto: <https://de-crypt.org/r/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX dbc:
    <http://dbpedia.org/resource/Category:>
PREFIX xsd:
    <http://www.w3.org/2001/XMLSchema#>

SELECT DISTINCT ?cipherName ?country ?warName
    ?warYear
WHERE {
  # Query the DECRYPT ontology for unresolved
    ciphers
  ?cipher rdf:type decryptonto:Record ;
          decryptonto:hasName ?cipherName ;
          decryptonto:hasCurrentCountry
    ?country ;
          decryptonto:hasStatus ?status .

  # Filter for unresolved ciphers
  FILTER(?status != "4" && ?status != 4)

  # Prepare country variables for matching
```

[9] https://www.dbpedia.org/

```
  BIND(LCASE(?country) AS ?countryLower)
  BIND(REPLACE(STR(?country), " ", "_") AS
    ?countryNormalized)
  BIND(CONCAT("http://dbpedia.org/resource/",
    ?countryNormalized) AS ?countryResource)
  BIND(CONCAT("http://dbpedia.org/resource/
  Category:Battles_in_", ?countryNormalized)
    AS ?battleInCategory)
  BIND(CONCAT("http://dbpedia.org/resource/
  Category:Battles_involving_",
    ?countryNormalized) AS
    ?battleInvolvingCategory)

  # Federated query to DBpedia for wars
  SERVICE <https://dbpedia.org/sparql> {
    # Get the war information
    ?war rdf:type dbo:MilitaryConflict ;
         rdfs:label ?warName .

    # Try multiple date formats
    OPTIONAL {
      ?war dbo:date ?date .
      BIND(YEAR(?date) AS ?warYear)
    }

    # If no date directly available, try
    start date
    OPTIONAL {
      ?war dbo:startDate ?startDate .
      BIND(YEAR(?startDate) AS ?warYear)
    }

    # Only use date if available and in 17th
    century
    FILTER(BOUND(?warYear) && ?warYear >=
    1600 && ?warYear < 1700)
    FILTER(LANG(?warName) = "en")

    # Connection to the country - using
    multiple approaches
    {
      # Option 1: Direct link to country
      ?war dbo:wikiPageWikiLink
    ?countryResource
    } UNION {
      # Option 2: Combatant contains country
    name
      ?war dbo:combatant ?combatant .
      FILTER(CONTAINS(LCASE(STR(?combatant)),
    ?countryLower))
    } UNION {
      # Option 3: Property combatant contains
    country name
      ?war dbp:combatant ?propCombatant .

    FILTER(CONTAINS(LCASE(STR(?propCombatant)),
    ?countryLower))
    } UNION {
      # Option 4: Place in country
      ?war dbo:place ?place .
      ?place rdfs:label ?placeLabel .
      FILTER(LANG(?placeLabel) = "en")
      FILTER(CONTAINS(LCASE(?placeLabel),
    ?countryLower))
    } UNION {
      # Option 5: Battles in country category
      ?war dcterms:subject ?battleInCategory
    } UNION {
      # Option 6: Battles involving country
    category
      ?war dcterms:subject
    ?battleInvolvingCategory
    }
  }
}
ORDER BY ?country ?warYear
LIMIT 100
```

The output of the query is shown in Figure 6, listing war names and starting years to illustrate the descriptive capabilities of the executed query. Additional information, such as the locations and key participants in each event, could also be included. The answer to the research question posed by the query appears in the first column, *cipherName*. This column was chosen over displaying full URIs due to space constraints; the URIs, however, remain accessible via clickable links that redirect to the corresponding DECODE record pages.

The DECODE database includes two built-in search functionalities: the *Advanced Search* and the *Query Builder*. The Advanced Search, shown at the left of Figure 7, corresponds to a natural language query such as: ”Retrieve all records that have...”—with specific properties and values inserted by the user. The Query Builder, displayed at the right of Figure 7, offers a more refined and flexible search experience. Unlike Advanced Search, which implicitly uses the AND operator and assumes equality conditions, the Query Builder allows users to define more complex queries using logical operators such as OR, as well as a broader set of conditions like ”contains,” ”ends with,” and their respective negations.

In terms of expressive power, the Query Builder more closely aligns with the SPARQL query-endpoint, which not only allows for precise data retrieval but also provides control over the output format. SPARQL enables users to refine result sets by defining specific entities to be displayed, incorporating optional statements, and setting limits on the number of rows. This is achieved using constructs such as DISTINCT, REDUCED, ORDER BY, LIMIT, OFFSET, and GROUP BY. Moreover, SPARQL supports data manipulation functions (e.g., LANG, DATATYPE, BOUND, IRI, URI, STRDT, STRLANG) and conditional expressions (e.g., IF, COALESCE, BNODE, CASE WHEN, THEN, ELSE). Some queries involve them alongside graph pattern matching (using BIND), making them more complex than those that can be constructed with the Query Builder. These queries

Table | Response | 100 results in 15.314 seconds | Simple view | Ellipse | Filter query results | Page size: 50

	cipherName	warName	warYear
1	BL_Add_MS_18983_001	"Battle of Nieuwpoort"@en	"1600"^^<http://www.w3.org/2001/XMLSchema#integer>
2	BL_Add_MS_18983_001	"Siege of San Andreas (1600)"@en	"1600"^^<http://www.w3.org/2001/XMLSchema#integer>
3	BL_Add_MS_18983_002	"Battle of Nieuwpoort"@en	"1600"^^<http://www.w3.org/2001/XMLSchema#integer>
4	BL_Add_MS_18983_002	"Siege of San Andreas (1600)"@en	"1600"^^<http://www.w3.org/2001/XMLSchema#integer>
5	BL_Add_MS_18983_003	"Battle of Nieuwpoort"@en	"1600"^^<http://www.w3.org/2001/XMLSchema#integer>
6	BL_Add_MS_18983_003	"Siege of San Andreas (1600)"@en	"1600"^^<http://www.w3.org/2001/XMLSchema#integer>

Figure 6: First six rows of query results for Query 3, showing battles in England from the 17th century and DECODE records of unsolved ciphers from the same period.

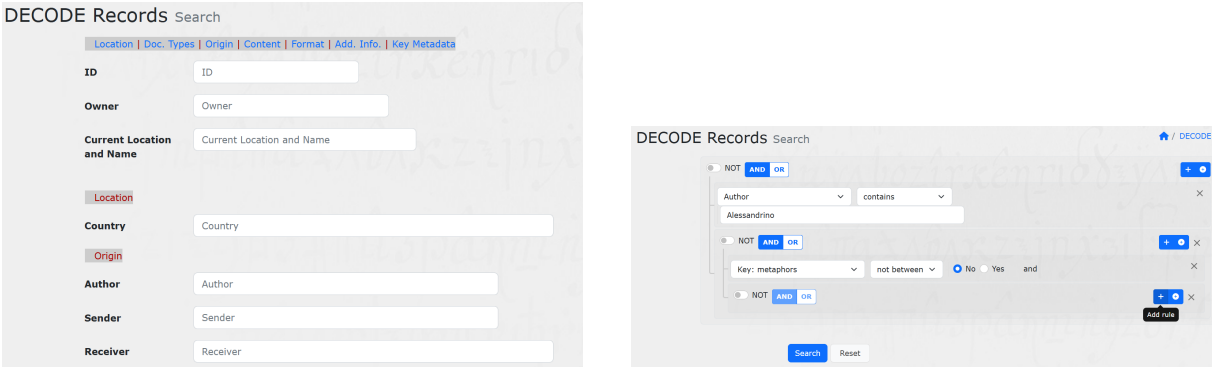


Figure 7: A snapshot of advanced search (left) and query builder (right) in the DECODE database.

support more advanced analyses, such as tracing the geographical spread of cipher usage over time or performing cross-cultural comparisons of cipher complexity—both of which require enhanced querying capabilities. Examples of these two query scenarios are provided in Appendix (7).

6 Limitations and Challenges

While the Query Builder offers a major advantage by allowing users—particularly those without experience in SPARQL or other query languages—to construct complex queries in an intuitive and user-friendly way, the dataset and underlying model presented here still face several limitations. These challenges currently hinder the system's immediate deployment and highlight the need for further development to improve its usability, interoperability, and completeness. To address these issues, several key enhancements are planned:

1. Enhancing Graph Density by Automated Inference.
2. Improving Interoperability and Wikidata Integration.
3. Integrating Large Language Models (LLMs) with Linked Open Data (LOD).

Enhancing Graph Density through Automated Inference This task focuses on creating new connections between existing nodes in the knowledge graph. When two records share similarities, relationships can be established using CIDOC-CRM properties such as P67 (refers to) or P130 (shows features of). Alternatively, a semantic rule layer can be manually defined to enable automated inference, enriching the graph with additional entities and relationships.

SPARQL serves not only as a query language for retrieving data but also as a means to manipulate the knowledge base by updating, deleting, or inserting nodes and relationships. Automated approaches can be employed to infer missing links or align the *decryptonto:hasAuthor* property with external sources like Wikipedia. Although many authors appear in both DECODE and Wikipedia, inconsistencies in naming conventions—such as "Cardinal Mazzarino" versus "Giulio Mazzarino"—necessitate further normalization before integration into DECODE2LOD can be achieved.

Improving Interoperability and Wikidata Integration Continuously aligning DECODE with Wikipedia and Wikidata for entities such as au-

thors, countries, years, and languages is essential not only for interoperability but also for enhancing its usability within the Digital History domain. For example, the author of the record shown in Figure 8 corresponds to the historical figure depicted in Figure 9.

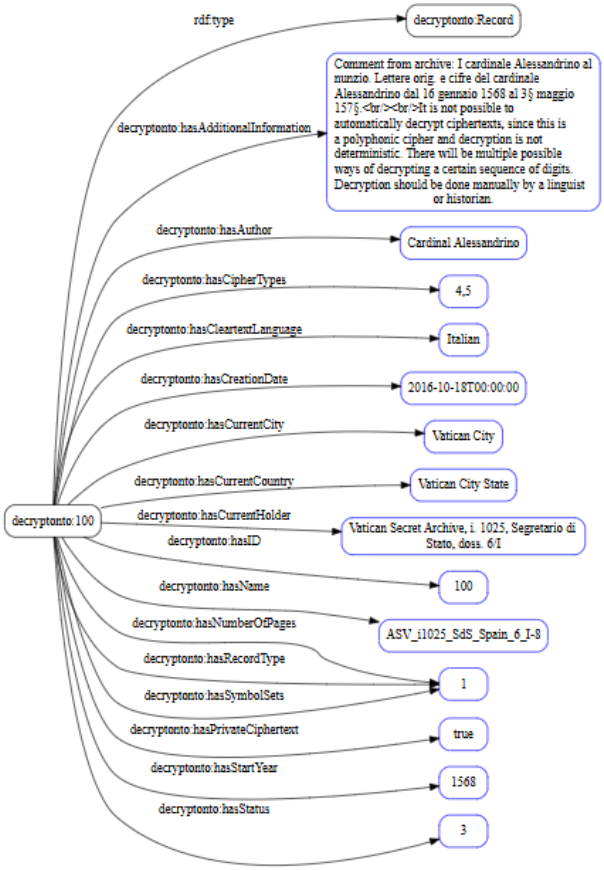


Figure 8: Visualization of a ciphertext from the DECODE database, whose author is Cardinal "Alessandrino", its cleartext language is Italian, and the cipher is stored at the Vatican city.

Figure 9: A portrait of Cardinal Michele Bonelli, known as "Alessandrino" (Source: *Wikipedia*)[11].

Wikipedia searches have also proven useful for disambiguating authors' nicknames. To improve clarity, a new property such as *decryptonto:alternativeName* could be introduced to link alternative names used for ciphertext authors.

Additionally, several key properties require alignment with their Wikidata equivalents, including:

- decryptonto:hasStartYear
- decryptonto:hasAuthor
- decryptonto:hasCleartextLanguage
- decryptonto:hasCreationDate
- decryptonto:hasCurrentCity
- decryptonto:hasCurrentCountry
- decryptonto:hasCurrentHolder

Integrating Large Language Models (LLMs) with Linked Open Data (LOD) While Pretrained Language Models (PLMs) often face challenges with accuracy and factual consistency—particularly due to their tendency to hallucinate information—they are remarkably effective at automatically generating SPARQL queries. This makes them especially valuable for improving accessibility and simplifying interaction with Linked Open Data (LOD) systems. Rather than viewing LLMs and LOD as competing technologies, they should be seen as complementary: LLMs enhance query formulation and user engagement, while LOD ensures structured, verifiable, and precise knowledge retrieval.

7 Concluding Remarks

In this paper, we presented the transformation of the DECODE database into a graph-based structure using Linked Open Data (LOD) to enhance its analytical potential. Unlike traditional relational databases, which rely on predefined table structures, a knowledge graph offers a more flexible and interconnected model, enabling complex queries that span multiple dimensions of historical cryptology. This transition not only improves data storage and retrieval but also facilitates advanced search techniques, such as semantic reasoning and federated queries. The improvements discussed in this paper aim to enhance both the technical robustness and practical accessibility of the historical cryptology ontology, making it a more powerful resource for digital humanities researchers.

However, deploying and maintaining a SPARQL endpoint on a web domain presents several challenges, particularly in terms of long-term maintenance. While these concerns may be minimal for the current dataset, they become

more significant as the dataset scales or faces high query loads. As more historical cryptographic records are digitized and integrated, the system must efficiently manage not only increased data volume but also complex relationship patterns and inference requirements.

Beyond storage capacity, scalability challenges also involve query optimization, cache management, and concurrent user access patterns. These technical constraints must be carefully balanced against the needs of digital humanities scholars, who often conduct exploratory analyses that generate resource-intensive queries. To ensure sustainable long-term operation, implementing strategies such as caching mechanisms, query rate limiting, and load balancing will be essential. These measures will help maintain system performance while preserving the analytical capabilities that make the graph-based approach valuable for historical cryptology research.

Acknowledgments

We gratefully acknowledge Mihály Héder for his invaluable advice and support, particularly during the technical implementation of the work presented here. We sincerely thank the anonymous reviewers for their insightful comments and constructive suggestions, which greatly improved the final version of this paper.

We also extend our thanks to the Department of Digital History (C²DH) at the University of Luxembourg, the Swedish Research Council (grant 2018-06074) for supporting the DECRYPT – Decryption of Historical Manuscripts project, and Riksbankens Jubileumsfond (grant M24-0028) for funding the DESCRYPT – Echoes of History: Analysis and Decipherment of Historical Writings project.

References

Thomas Baker. 2000. The Dublin core metadata initiative. *D-lib magazine*, 6(12):1082–9873.

Paolo Bonavoglia. 2023. *La crittografia della Repubblica di Venezia*. Aracne.

Maria Pia di Buono, Mario Monteleone, and Annibale Elia. 2014. How to populate ontologies: Computational linguistics applied to the cultural heritage domain. In Elisabeth Métais, Mathieu Roche, and Maguelonne Teisseire, editors, *Natural Language Processing and Information Systems*, volume 8455 of *Lecture Notes in Computer Science*, pages 55–66. Springer.

Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. 2014. RML: A generic language for integrated rdf mappings of heterogeneous data. In *Proceedings of the 7th Workshop on Linked Data on the Web*, LDOW '14. CEUR Workshop Proceedings.

Martin Doerr. 2003. The CIDOC CRM - An ontological approach to semantic interoperability of metadata. *AI Magazine*, 24(3).

Roy T. Fielding and Richard N. Taylor. 2002. Principled design of the modern Web architecture. *ACM Transactions on Internet Technology*, 2(2):115–150.

Manuel Fiorelli, Tiziano Lorenzetti, Maria Teresa Pazienza, Armando Stellato, and Andrea Turbati. 2015. Sheet2rdf: a flexible and dynamic spreadsheet import&lifting framework for rdf. In *Current Approaches in Applied Artificial Intelligence*, pages 131–140. Springer International Publishing.

Ramanathan V Guha, Dan Brickley, and Steve Macbeth. 2016. Schema. org: Evolution of structured data on the web. *Communications of the ACM*, 59(2):44–51.

Mihály Héder and Beáta Megyesi. 2022. The DECODE Database of Historical Ciphers and Keys: Version 2. In *Proceedings of the 5th International Conference on Historical Cryptology, HistoCrypt22*.

Graham Klyne and Jeremy J. Carroll. 2004. Resource Description Framework (RDF): Concepts and Abstract Syntax. *Journal of Web Semantics*, 2(1):9–29.

George Lasry, Norbert Biermann, and Satoshi Tomokiyo. 2023. Deciphering Mary Stuart's lost letters from 1578-1584. *Cryptologia*, 47(2):101–202.

Beáta Megyesi, Nils Blomqvist, and Eva Pettersson. 2019. The DECODE Database: Collection of Ciphers and Keys. In *Proceedings of the 2nd International Conference on Historical Cryptology, HistoCrypt19*, Mons, Belgium, June.

Beáta Megyesi, Bernhard Esslinger, Alicia Fornés, Nils Kopal, Benedek Láng, George Lasry, Karl de Leeuw, Eva Pettersson, Arno Wacker, and Michelle Waldispühl. 2020. Decryption of historical manuscripts: the DECRYPT project. *Cryptologia*, 44(6):545–559.

Cosimo Palma. 2023. Encrypted epigraphy - the case of a mysterious inscription in the Neapolitan church of Santa Maria La Nova. In *International Conference on Historical Cryptology*.

Stuart L Weibel and Traugott Koch. 1997. The Dublin core: a simple content description model for electronic resources. *Bulletin of the American Society for Information Science and Technology*, 24(1):9–11.

Appendix

Query 1. Geographical Spread of Cipher Usage Over Time

```
SELECT ?period ?region (COUNT(DISTINCT
    ?cipher) AS ?cipherCount)
WHERE {
  ?cipher decryptonto:hasStartYear ?year ;
          decryptonto:hasCurrentCountry
    ?country .

  # Define historical periods
  BIND(
    IF(?year < 1500, "Medieval",
      IF(?year < 1700, "Early Modern",
        IF(?year < 1900, "Modern",
    "Contemporary")
      )
    ) AS ?period
  )

  # Group countries into regions (simplified)
  BIND(
    IF(REGEX(?country,
    "France|Spain|Italy|Germany", "i"),
    "Western Europe",
      IF(REGEX(?country,
    "England|Scotland|Ireland", "i"),
    "British Isles",
        IF(REGEX(?country,
    "Russia|Poland|Hungary", "i"), "Eastern
    Europe",
          "Other"
        )
      )
    ) AS ?region
  )
}
GROUP BY ?period ?region
ORDER BY ?period ?region
```

Query 2. Cross-Cultural Comparison of Cipher Complexity

```
SELECT ?culture ?avgSymbolSetSize
    ?maxSymbolSetSize
WHERE {
  {
    SELECT ?culture (AVG(?symbolSetSize) AS
    ?avgSymbolSetSize) (MAX(?symbolSetSize)
    AS ?maxSymbolSetSize)
    WHERE {
      ?cipher decryptonto:hasCurrentCountry
    ?country ;
              decryptonto:hasSymbolSets
    ?symbolSets .

      # Simplified culture grouping
      BIND(
        IF(REGEX(?country,
    "China|Japan|Korea", "i"), "East Asian",
          IF(REGEX(?country,
    "Arabia|Persia|Ottoman", "i"), "Islamic",
            IF(REGEX(?country,
    "Italy|France|Spain", "i"), "Latin",
              IF(REGEX(?country,
    "England|Germany|Netherlands", "i"),
    "Germanic",
                "Other"
              )
            )
          )
        ) AS ?culture
      )

      # Assuming symbolSets is a
    comma-separated list, count the items
      BIND(1 + STRLEN(?symbolSets) -
    STRLEN(REPLACE(?symbolSets, ",", "")) AS
    ?symbolSetSize)
    }
    GROUP BY ?culture
  }
}
ORDER BY DESC(?avgSymbolSetSize)
```

	period	region	cipherCount
1	Contemporary	British Isles	"6"^^<http://www.w3.org/2001/XMLSchema#integer>
5	Early Modern	Eastern Europe	"114"^^<http://www.w3.org/2001/XMLSchema#integer>
10	Medieval	Western Europe	"699"^^<http://www.w3.org/2001/XMLSchema#integer>
11	Modern	British Isles	"670"^^<http://www.w3.org/2001/XMLSchema#integer>

Figure 10: A partial view of Query 1 results.

	culture	avgSymbolSetSize	maxSymbolSetSize
1	Latin	"1.819563780568407138136153"^^<http://www.w3.org/2001/XMLSchema#decimal>	"4"^^<http://www.w3.org/2001/XMLSchema#integer>
2	Other	"1.536078845476944737768391"^^<http://www.w3.org/2001/XMLSchema#decimal>	"4"^^<http://www.w3.org/2001/XMLSchema#integer>
3	Germanic	"1.264308681672025723472669"^^<http://www.w3.org/2001/XMLSchema#decimal>	"4"^^<http://www.w3.org/2001/XMLSchema#integer>

Figure 11: Results for Query 2.

A Caribbean Directory-based Encryption during the American War of Independence

Bellecombe, governor of Saint-Domingue, 1782

Cécile Pierrot
CNRS, Inria, LORIA
Université de Lorraine
54000 Nancy, France
cecile.pierrot@inria.fr

Olivier Chaline
FED 4124 Fédération d'histoire
et d'archéologie maritimes
Sorbonne Université, Paris, France
olivier.chaline@sorbonne-universite.fr

Gaspard Damoiseau-Malraux
Sorbonne Université
Paris, France
gaspard.dama@gmail.com

Paul Mekhail
Sorbonne Université
Paris, France
paul.mekhail4@pm.me

Ludovic Perret
EPITA, EPITA Research Lab (LRE)
Le Kremlin-Bicêtre, France
Sorbonne Université
CNRS, LIP6, F-75005 Paris, France
ludovic.perret@epita.fr

Abstract

The corpus of letters we are studying is located at the *Archives Nationales d'Outre-Mer* in Aix-en-Provence, France. These late 18th-century letters come from Saint Domingue (now Haiti), a French colony in the Caribbean Sea of which Bellecombe, the author, was governor. They were written in the context of the American War of Independence, in which France took part on the side of the Americans. We have reconstructed Bellecombe's correspondence with the Secretary of State for the Navy, in Versailles: the archives contain hundreds of letters in clear and three encrypted letters, including some clear/cipher pages that were our lever for reconstructing part of the key, and 96% of the encrypted letter that was opaque at first. From a cryptanalytical point of view, Bellecombe used a directory-based encryption. The common use of this type of cipher in the 17th and 18th-century European countries raises the question of the method to be used (then as now!) to decode such messages.

1 Introduction

Occasionally, in a box of archives, the historian suddenly comes across an encrypted document that remains just as it was sent. It is not surprising to find this type of missive: precautions were taken to ensure that a confidential document could not be read if it fell into enemy hands. What could surprise the historian is that the official addressee either did not decrypt it, or did not keep and enclose it. As it is rare for the cipher key to be included in archives, some documents remain unexploited because they are incomprehensible.

While studying the American War of Independence, particularly in the West Indies, one of us came across a number of encrypted letters, the contents of which have remained inaccessible to historians. These documents are kept at the *Archives Nationales d'Outre-mer* "French National Oversea Archives", in Aix-en-Provence, which holds all the archives relating to French colonization, from the 17th to the 20th century. The discovery was made while exploring correspondence from the governor of Saint-Domingue, now the Republic of Haiti, at the time the largest of the French sugar islands, and therefore a major economic stake. The historian, powerless in the face of such a succession of numbers, turned to the code breakers.

At first glance, the cipher is already visually very typical: the presence of pages of numbers between 1 and 999 (see Figure 1) suggests that this is a *directory-based encryption*. We do not know if there is already an English word for this type of cipher. If not, we would like to suggest this name for ciphers halfway between homophonic ciphers (with a small nomenclature of around 20/30 words) and code-book ciphers that use entire dictionaries. The name is based on the French term *chiffrement à répertoire*. Whereas the cryptanalysis of ciphers from the 16th century, such as homophonic ciphers, starts to be well understood (Pierrot et al., 2023; Lasry et al., 2023) this type of system used in the 17th and 18th-centuries in Europe

Proceedings of the 8th International Conference on Historical Cryptology, pages 122-132
Poznań, Poland, 16-18 June, 2025

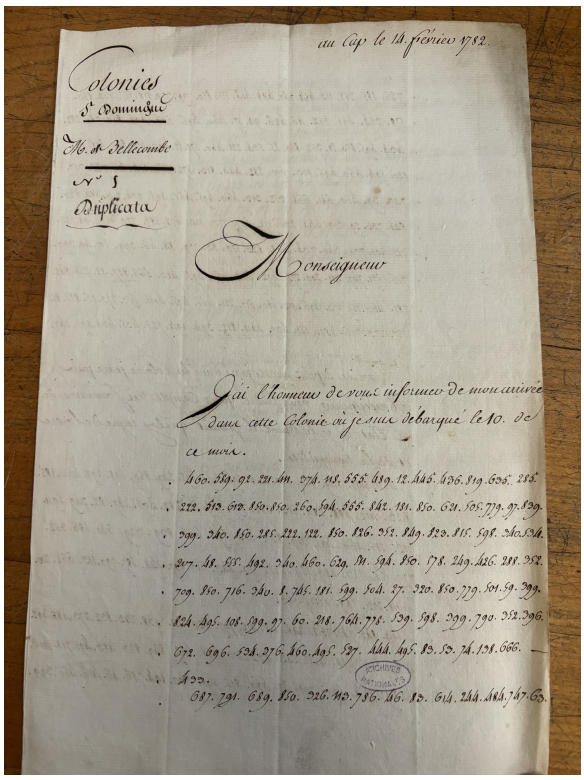


Figure 1: First page of the encrypted letter 1

deserves a closer look.

2 Historical Context

The American Revolutionary War, which pitted the Thirteen Colonies, a group of frontier colonies in North America, against the Kingdom of Great Britain from 1775 to 1783, was one of the processes of the American Revolution that saw the United States emancipate itself from the British Empire and gain independence. In 1777, following the Battle of Saratoga, other European powers joined the American people, notably France which provided soldiers, equipment, donations, and loans to the insurgents. France officially committed itself in 1778. French naval and land aid, and the support of its allies, contributed to the American victory, notably at the Battle of Yorktown, which led the British to surrender their arms and accept the independence of the United States in 1783. Contrary to its name, most of the fighting in the American War of Independence took place in the Caribbean, where the involved countries all had colonies. The main islands were: Cuba and Santo Domingo (the eastern part of the island of Hispaniola) for the Spanish; Jamaica and the Bahamas for the English; Saint-Domingue (the western part of Hispaniola) for the French.

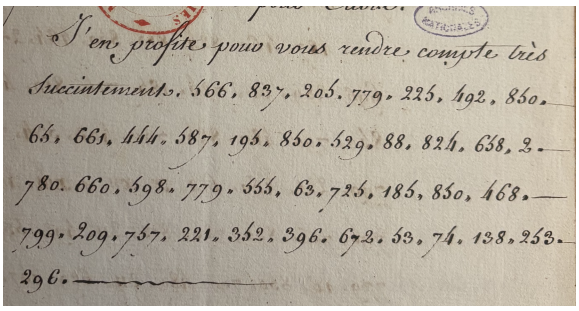


Figure 2: Beginning of the ciphertext in letter 57

Several governors took over the French colony of Saint Domingue during this war: D'Argoût until 1780, Lilancour on an interim basis, Villeverd from 1780 to 1781, Lilancour again, then Bellecombe in 1782. More precisely, Guillaume de Bellecombe, who is the author of all the encrypted letters this article deals with, remained in post from February 14, 1782 until July 3, 1785, i.e. after the end of the war.

3 Bellecombe's letters

All Bellecombe's ciphertexts found are in the COL C9 B 32 collection at the French National Oversea Archives[1]. This collection corresponds to all documents sent from Saint Domingue to the Secretary of State for the Navy during the year 1782. A large number of clear letters have been preserved in COL C9 A 153 and COL C9 A 154 (year 1783) and then in COL C9 A 155 and COL C9 B 34 (year 1784), but none of them are encrypted. The number of pages from Bellecombe can be quite large: for instance, there are 360 clear pages in Bellecombe's hand in the COL C9 A 155 collection.

In all, there are several hundred pages of Bellecombe's clear letters, and 11 encrypted pages spread over 3 different letters numbered 1, 57 and 87, all dated 1782. Every single letter Bellecombe sent has a number and a date, so organization is easy. In COL C9 B 32 we found many non-encrypted letters, numbered: 2, 5, 6, 7, 8, 9, 10, 11, 14, 26, 30, 31, 32, 38, 47, 48, 49, 50, **57**, 58, 59, 61, 62, 76, 77, 71, 80, 83, **87**, 131, 132, 163, 168, 186, 216, 248, 272, 276, 277, 287, 288, 291, 317, 321, 361. Table 1 highlights the existence of two pairs of clear/cipher with the same number. It was a prolific correspondence, with almost a letter per day. Only one eighth has survived to this day.

[1] Archives Nationales d'Outre Mer, 29, ch. du moulin de Testas, 13182 AIX-EN-PROVENCE, France. `http://www.archivesnationales.culture.gouv.fr/anom/fr/`

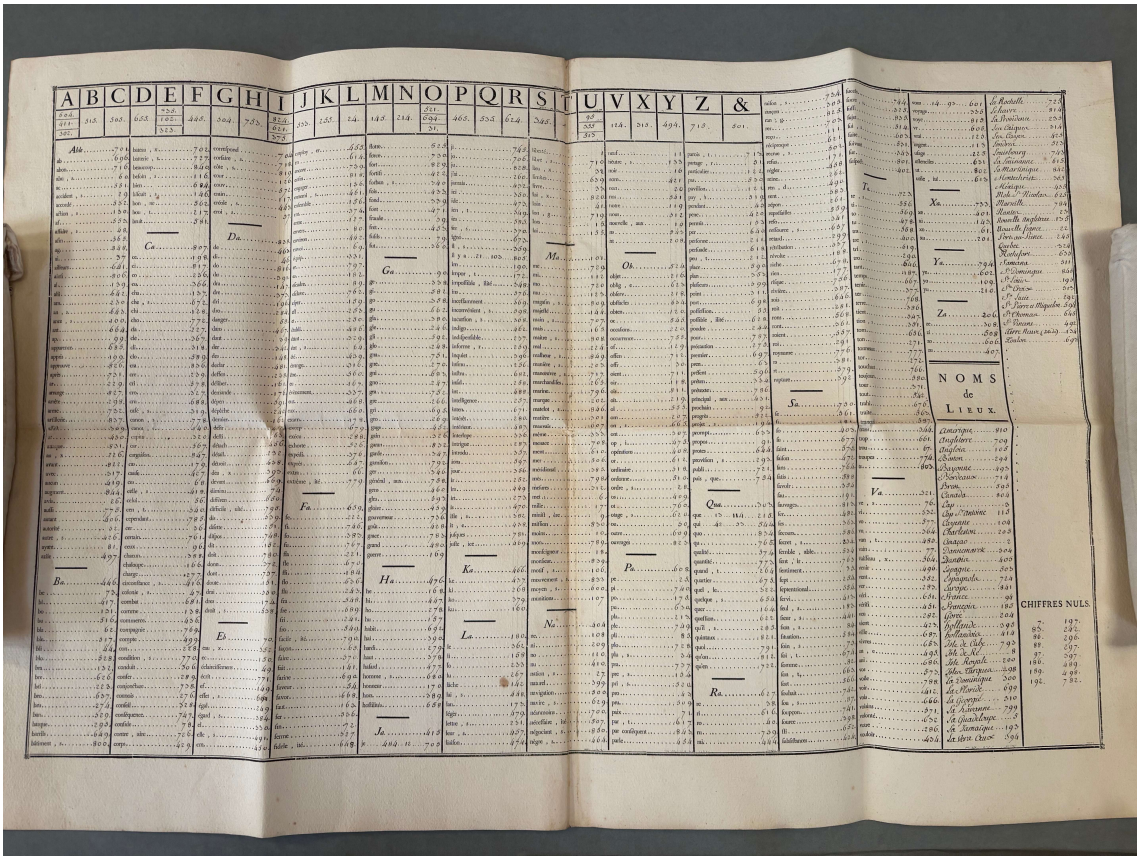


Figure 3: The gigantic cipher table of D'Argoût, governor of Saint Domingue, 1778.

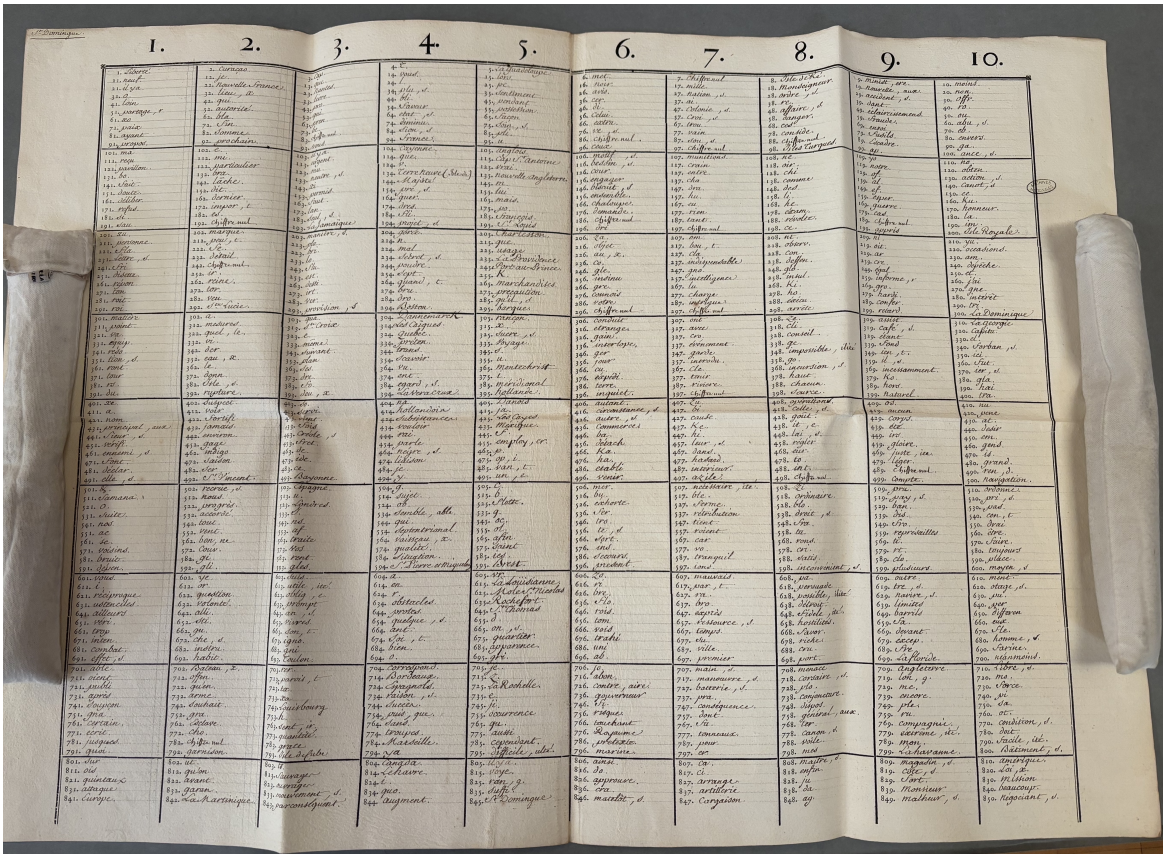


Figure 4: The corresponding table to decipher D'Argoût's cipher, 1778.

Letter number	Date	Length in pages
Cleartexts		
57 (primata)	4 July 1782	2
87 (copie)	17 September 1782	3
Ciphertexts		
1 (duplicata)	14 February 1782	4
57 (triplicata)	4 July 1782	4
87 (duplicata)	17 September 1782	3

Table 1: Bellecombe's ciphertexts together with their cleartexts, when they exist, in COL C9 B 32.

4 Decryption methods

4.1 Cryptographic method of the time

All the ciphertexts we have consist of long sequences of numbers between 1 and 999, which is common in the 17th and 18th centuries. See Figure 2 for instance.

Given this visual aspect, the century, and the fact that this is military correspondence during a war period, everything suggests that we are dealing with a directory-based encryption: in this type of cryptosystem, each symbol (or number) always carries the same value throughout the message. The difficulty of cryptanalysis lies in the very large number of these symbols. Some have the value of a letter or syllable, others are nulls (around ten or twenty) or proper nouns, but the vast majority are ciphers of complete and common words (e.g. in French *contre* "against", *pour* "for", *aucun* "none"), prepositions and endings (e.g. *ment*, *ance*), or verb radicals (e.g. *donne* "give", *diminu* "decreas"). We found in COL C9 B 28 collection a valuable cipher table established in the same context: that of Governor Robert, comte d'Argoût, represented in Figures 3 and 4. D'Argoût was governor of the same colony from 22 May 1777 to 7 Mars 1780. His cipher table from 1778 predates the beginning of the war and is accompanied by a decryption table that is easier for the recipient to read and use. Figure 3 is not Bellecombe's key – it is likely that the keys changed with each governor – but it does indicate the maritime (e.g. *navire* "ship", *voile* "sail", *chaloupe* "rowboat") and military (e.g. *paix* "peace", *munition* "ammunition") vocabulary we can imagine finding there too. Even better! A careful eye will notice that the vocabulary of the table is printed, only the place names are left to the secretary's choice. This suggests that Bellecombe uses the same basic words in his nomenclature.

While the process of encoding and decoding using a directory-based encryption table is well understood, we are not aware of any document detailing the cryptanalysis methods employed at the time.

4.2 Transcription

We give in the Appendix the transcriptions of the cipher parts of letters 1, 57, and 87. It was not always easy due to ink deterioration and the difficulty of recognizing certain numbers caused by the writing style. A common technique in the field of historical cryptanalysis is to use machine learning tools to automate transcriptions. Yet, the transcript of Bellecombe's letters with online tools such as (Escriptorium, 2019) or (Transkribus, 2013) was not sufficiently good to rely on it. The consequences of a bad transcription can be important as it will influence our analysis a lot. For this reason, we manually transcribed a total of almost 1500 symbols and double-checked everything.

4.3 Reconstructing part of the key

A quick statistical analysis – see Figure 5 – gave us the meaning of the number 850, which is the word *de* "of". We tried to decrypt in priority the most common numbers to clear as much of the letter as possible. In fact, this is not an easy task. Because the slicing of a plaintext word follows no apparent rule, it is very difficult to guess from nothing what the first numbers of a sentence could be. The fact that letter 57 had multiple cleartext zones in the cipher was a huge help, so we focused on this letter first. Those cleartext zones, as well as the initial observation that $850 = de$ gave us a clue to match the numbers and corresponding plaintext parts.

Other obstacles emerged. For example, we stumbled across a letter which the cleartext did not perfectly match our decryption. In the cipher letter 57 we deciphered the sentence 657 389 114 399 756 113 736 580 185 273 254 572 614 344 194 850 716 610 530 233 92 495 408 779 664 571 185 759 489 12 745 672 850 285 181 493 623 495 505 779 97 839 399 340 as *je me borne a vous annoncer quil partira de ce port quelque jours après lequinoxe un convoy de quante voiles sous lescorte...*[2]. However, in the clear letter 57

[2]I can only announce that a convoy of "fty" sails will leave

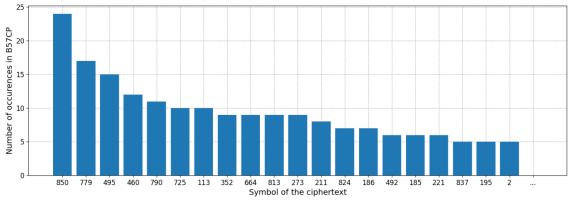


Figure 5: Most common numbers in letter 57 and their respective frequencies.

we read *Je me borne à vous annoncer qu'il partira de ce port, quelques jours après l'équinoxe, un convoi de quarante-cinq à cinquantes voiles sous l'escorte*[3]. As we can see, a crucial information here is the number of sails, and it does not appear clearly in the ciphertext.

A step-by-step comparison of the plaintext and ciphertext in letters 57 and 87 enabled us to establish an initial correspondence of values. With this partial table, 73% of the total different numbers we see in letters 1, 57, 87 can be decrypted.

4.4 Guessing last words with ciphertext only

Alas, most of the important information in letter 1 was missing, including city names, islands, or specific words that we had not encountered before. The extended part of the key we propose permits to read 96% of the numbers that appear. To extend this cipher table, we first decrypted letter 1 with the intermediate table obtained in Section 4.3, to get a letter with holes in it. First, we filled in the smallest holes (one or even two missing numbers), as well as those with numbers appearing elsewhere in the same letter. "Guessing" the value of the numbers is not done blindly: we always choose words that are in the vocabulary of Figure 3.

4.5 The key

The whole reconstructed key is in the Appendix. We report the confidence we have in the values attributed to the number: by default we are sure of the values but we mark with * those we doubt slightly. Typically, words guessed at the end, using the D'Argoût's table, are marked with *. The dots ... mean that the value is not known yet. Besides, we found 8 nulls. Note that we expect the real Bellecombe's table to be bigger than the one we reconstruct. If all the numbers between 1 and 850 are assigned as in D'Argoût's table, then we can estimate that we have reconstructed 36% of Bellecombe's real table.

this port a few days after the equinox, under the escort of...

[3]I can only announce that a convoy of forty-five to fifty sails will leave this port a few days after the equinox, under the escort of...

5 Results

Here is the first letter of Bellecombe, reconstructed as best we could.

5.1 Beginning in cleartext.

Au Cap, le 14 février 1782. Monseigneur, J'ai l'honneur de vous informer de mon arrivée dans cette colonie où je suis débarqué le 10 de ce mois.

"Cape Town, February 14, 1782. Sire, I have the honor of informing you of my arrival in this colony, where I disembarked on the 10th of this month."

5.2 The decryption of letter 1

*Le même jour, il est arrivé un convoi espagnol por*tant* quatre *cent* *barques* de débarquement venant de 621 sous l'escorte de quatre vaisseaux de *guerre* et trois frégates ou corvettes. Le logement de nos troupes en garnison et celles de cette *flotte* vont nous créer bien de l'embarras, mais, nous espérons, Monsieur Bongard et moi, pouvoir les satisfaire. Je 791,e de *suite* à La Havane pour faire part à Dom Solano et à Dom Galvé de l'arrivée de nos forces et pour en*voy*er les dépêches de messieurs de 21 et de Bouillé qui contient probablement les *différentes* positions qu'ils ont *choisies* pour la campagne *marin*e et sur les *terre*s. Je ne dois pas vous laisser *penser* que je n'ai nullement été consulté. Je vais *par conséquent* faire l*impossible* pour remplir ce qui m'est pre(s)cri par les ordres du *roy* concernant le *cla*n arrêté a *Versailles* *dont* je n'ai *connaissance* suivant *les* *excep*tions *qu* à la *lu*eur des 147,or,es. Arrivé depuis quatre jours dans la colonie, je ne puis, Monseigneur, vous rendre qu'un compte très sommaire de l'état où je la trouve ; il paraît qu'il y règne de l'ordre et de la tranquilité. Les trois régiments des troupes de 508 Gatinois en g,h,553 et Cambresis actuellement à 215, les deux régiments coloniaux et l' 14,153,un,550,ta,l,843,540,c,ti,f d'à peu près cinq *cent* barques. Nous n'avons 448,ve encaissé *que* *cent* vingt-et-un 513,*douze* *cent* vingt-trois 483,s,718. Allons-nous *envoyer* une frégate à La Havane pour aller chercher des *ustenciles* en argent, ils sont*

*d'autant plus pressants qu'il y a peu de *marchandise* en ce genre dans cette *colonie* à ce que l'on m'a assuré. Nous avons en *provision* *douze* *mille* trois *cent* *dix* neuf *livres* de *farine*, trois *mille* trente *sacs* de *sucre*, trente cinq *sacs* de riz, treize *cent* six *livres* de salaison et *dix* barriques de *mauvais* vin *rouge*. L'arrivée des espagnols crée une augmentation de consomation qui va absorber bien vite nous approvisionnements. Le *seul* *bateau* du convoi de 385 et de celui de 167 qui *est* arrive est le Senault. La Marquise de Galart qui port*ait* les *hommes* de Monsieur Bongard et les miens, poursuivi par deux frégates anglaises à l'entrée de cette rade où il a voulu se réfugier près du port de 712, où il a péri et on *croit* que *tout* est *per*du. La Néréide et la Railleuse *sont* les *seuls* *bateaux* de *guerre* *que* nous ayons pour protéger le commerce sur la côte *méridionale*. On ne *doit* nullement compter sur les vaisseaux espagnols. Il ne partiront prochainement pas de la rade à moins qu'ils n'en reçoivent l'ordre de Dom Solano. *Aussi* le commandant de cette escadre a refusé de porter par mer ses troupes de débarquement près du fort du u,p,h,i,ou,je,au(x),544,517,e, *terre*,s,324,s,se,nt ac,807,pe une campagne à celle *qu'il* *attendoit* de La Havane.*

A translation could be: "On the same day, a Spanish convoy arrived, carrying four *hundred* landing *boats* from 621, escorted by four *war* vessels and three frigates or corvettes. The accommodation of our garrison troops and those of this *fleet* will cause us a great deal of embarrassment, but we hope, Monsieur Bongard and I, to be able to satisfy them.

I 791-e right away to Havana to inform Dom Solano and Dom Galvé of the arrival of our forces and to *send* the dispatches of Sire 21 and De Bouillé which probably contain the *different* positions they have *chosen* for the military campaign on land and at sea. I must not let you *think* that I was in no way consulted. I will *therefore* do my *possible* to fulfill what is prescribed to me by the orders of the *roy* concerning the clan arrested at *Versailles* *of which* I only have knowledge about exception (... something not readable there).

Having arrived in the colony four days ago, Monseigneur, I can only give you a very brief account of the state I find it in; it seems that there is order and tranquility. The three regiments of 508 Gatinois troops in g-h-553 and Cambresis currently at 215, the two colonial regiments and the 14,153,un,550,ta,l,843,540,c,ti,f of about five *hundred* boats. We only 448,ve cashed *hundred* twenty-one 513, *twelve* *hundred* twenty-three 483,s,718. Are we going to *send* a frigate to Havana to fetch silver *ustencils*, they are all the more pressing as there is little merchandise of this kind in this *colony* as I have been assured. We have in *provision* *twelve* *thousand* three *hundred* *nineteen* *pounds* of *flour*, three *thousand* thirty *sacks* of *sugar*, thirty-five *sacks* of rice, thirteen *hundred* six *pounds* of cured meats and *ten* barrels of *bad* *red* wine. The arrival of the Spaniards creates an increase in consumption that will soon absorb our supplies.

The only boat of the convoy of 385 and that of 167 which has arrived is The Senault. The Marquise de Galart which carries the *men* of Monsieur Bongard and mine, pursued by two English frigates at the entrances of this roadstead where he wanted to take refuge near the port of 712, where he perished and we think all is lost. La Néréide and La Railleuse are the only war boats we have to protect trade on the *southern* coast. We *must* by no means rely on Spanish vessels. They will not leave the roadstead unless ordered to do so by Dom Solano. *Also* the commander of this squadron has refused to bring his landing troops by sea near the flat of the u,p,h,i,or, (something not readable...) a campaign he expected from Havana."

5.3 End in cleartext.

La voie dont je me sers pour vous faire parvenir cette lettre avec le duplicata de celle que j'ai eu l'honneur de vous écrire du camp de Monsieur de Bouillé à Saint Christophe est peu sûre, mais comme c'est la première qui se présente je crois devoir en profiter. C'est un bâtiment américain qui doit se rendre en Hollande. Je suis avec respect, monseigneur, votre très humble très obéissant serviteur Bellecombe.

A translation could be: "The means I am using to send you this letter with the duplicate of the one of the one I had the honor of writing to you from Monsieur de Bouillé's camp in Saint Christophe is not very safe, but as it is the first one I think I should take advantage of it. It is an American

ship bound to Holland. I am respectfully, monseigneur, your most humble and obedient servant Bellecombe."

5.4 Content of the letters

The letter we have decrypted is the first letter that Bellecombe sent right after his arrival on Saint Domingue Island. We learn of the landing of large numbers of Spaniards on the island of Saint Domingue. Alas, this is causing concern for the governor, who is worried about how to feed all these men, in addition to those garrisoned there. In this letter, he gives a precise breakdown of the colony's food supplies. He also points out that the Spanish troops stationed on the island will be of no help in defending it in the event of an English attack. We also discover the reason for the letter's encryption: the author of the letter considers that the means of transport he uses is not very secure (it is an American ship), but he takes the risk of entrusting his mail to it, as it is the first ship to leave the island to Europe.

Conclusion: further reflection on directory-based encryptions

The preservation and completeness of the correspondence that reaches us remains uncertain: with only one eighth of the letters sent from Bellecombe preserved, it is fortunate that we were able to find identical passages that were sent (and preserved!) in both clear and encrypted form. A natural question that arose, and to which we have no answer at the moment, concerns the disappearance of encrypted letters. In wartime, when messages cross the ocean, the letters preserved in clear (e.g. letters 58, 59, 61, 62, 76...) may have been originally encrypted at least partially as well, especially given that the preceding and following ones have partially encrypted copies in other files. When and why did their encrypted counterparts disappear? From a cryptographic point of view, it was the presence of a clear/encrypted pair that enabled us to reconstitute the cipher table; but what would we have to do in the presence of a cipher-only attack? What methods were used at the time to attempt to penetrate the contents of letters protected by a directory-based encryption? How far can today's hill-climbing and simulated annealing algorithms, which enable cryptanalysis of homophonic ciphers (Lasry, 2018), help us when we are dealing with ciphers with nomenclatures of the order of a thousand values? Since those systems were used for more than two centuries, we encourage all code breakers to look into the matter.

About the name *directory-based encryption*. We suggested this name because we feel that the current option *nomenclator* is not clear enough. Indeed, here is a quote from David Kahn's *Codebreakers* that a kind reviewer pointed out to us. In the preface there is a small text on nomenclators: "For 450 years, from about 1400 to about 1850, a system that was half a code and half a cipher dominated cryptography. It usually had a separate cipher alphabet with homophones and a codelike list of names, words, and syllables. This list, originally just of names, gave the system its name: nomenclator. Even though late in its life some nomenclators grew larger than some modern codes, such systems are still called nomenclators if they fall within this historical period." The issues with the name *nomenclator* are the following:

- The word has several meanings (both the cryptosystem and the list of names) and this introduces confusion.
- As in current cryptography, systems must be named according to their mechanism and not the year or century in which they were used.
- Naming objects and ideas allows us to work with them, and is the first step towards understanding a concept. For example, the lack of distinction in the vocabulary (naming everything homophonic cipher) makes it impossible to distinguish between systems with very small nomenclators and those described here. The former are within the reach of modern algorithms for breaking them. The latter are not. If the attack algorithms are different, it's because the inherent security of the defence systems is different. Wouldn't it be worth naming these systems differently? Or do we want to keep all the cryptography that has been used for 450 years under the same name? Isn't that a bit reductive?

Aknowledgment

The authors thank the anonymous reviewers for their useful comments. Part of this work is founded by the Back In Time project.

References

project Escriptorium. 2019. Escriptorium: A platform for digital paleography. `https://escriptorium.inria.fr`. Accessed: 2025-02-10.

George Lasry, Norbert Biermann, and Satoshi Tomokiyo. 2023. Deciphering Mary Stuart's lost letters from 1578-1584. *Cryptologia*, 47(2):101–202.

George Lasry. 2018. *A Methodology for the Cryptanalysis of Classical Ciphers with Search Metaheuristics*. Kassel University Press GmbH.

Cécile Pierrot, Camille Desenclos, Pierrick Gaudry, and Paul Zimmermann. 2023. Deciphering Charles Quint (A diplomatic letter from 1547). *Histocrypt*, 195:148–158.

Transkribus. 2013. Transkribus: AI-powered text recognition and handwritten text analysis. `https://www.transkribus.org/en`. Accessed: 2025-02-10.

Appendix

Letter 57, cipher part: 566 837 205 779 225 492 850 65 661 444 587 195 850 529 88 824 658 2 780 660 598 779 555 63 725 185 850 468 799 209 757 221 352 396 672 53 74 138 253 296.779 664 610 175 352 460 196 850 725 837 359 813 195 354 495 842 468 460 495 195 310 548 625 523 29 211 113 725 850 604 282 813 195 420 850 677 745 278 604 396 492 46 747 831 779 496 572 599 560 412 445 221 244 523 12 555 187 2 599 72 779 113 495 669 664 566 832 273 488 492 282 683 juin 1782 172 460 495 354 495 842 468 779 97 850 856 352 790 542 527 669 77 72 460 132 404 850 185 784 501 51 186 819 797 88 779 629 313 725 1 850 779 133 186 140 790 664 852 282 450 484 273 761 211 696 289 63 725 185 352 468 757 209 765 221.487 460 824 819 850 484 273 761 211 696 289 63 725 185 716 819 411 200 716 850 790 489 824 790 493 534 779 813 396 799 221 460 614 x xii et xiii 44 495 360 790 706 850 725 813 211 492 534 664 113 779 16 850 790 489 311 629 757 60 725 677 273 735 738 316 689 185 273 115 172 460 140 813 145 714 340 113 492 534 272 399 195 310 1 779 2 664 566 819 352 779 813 672 747 354 221 460 614 v vi vii viii 44 495 360 790 706 207 411 489 466 850 viii 92 495 186 115 46 71 850 725 186 857 113 489 196 664 186 140 850 716 819 813 194 514 830 850 166 399 166 824 225 571 113 460 589 44 495 211 672 309 664 352 571 662 12 361 455 850 122.779 45 850 778 790 493 468 757 209 765 221 839 747 610 211 664 850 222 278 191 122 352 849 823 815 598 340 411 614 278 493 716 686 444 273 46 779 133 348 664 749 837 495 340 824 359 689 113 709 850 484 273 63 725 185 790 493 484 765 191 122 571 837 291 756 113 786 476 244 779 2 689 566 262 850 59 728 686.813 29 211 670 46 818 340 568 460 401 849 823 122 460 495 604 619 813 460 39 779 199 27 352 489 436 762 849 823 815 598 340 697 97.

La frégate marchande Le Maréchal de Mouchy 614 344 113 388 832 273 488 492 186 784 173 604 273 789 716 211 824 604 ce bâtiment 172 687 186 824 672 779 552 850 736 113 790 837 495 259 780 546 495 352 790 2 399 72 850 115 500 216 495 607 495.

Letter 87, cipher part: 657 389 114 399 756 113 736 580 185 273 254 572 614 344 194 850 716 610 530 233 92 495 408 779 664 571 185 759 489 12 745 672 850 285 181 493 623 495 505 779 97 839 399 340 566 39 779 199 27 352 850 849 823 815 598 340 778 850 686 399 492 808 529 211 818 610 211 790 54 365 344 313 53 74 296 666 838 592 48 340 598 779 730 499 666.

Letter 1, encrypted: 460 589 92 221 411 374 118 555 489 12 445 436 819 635 285 222 513 613 850 850 260 594 555 842 181 850 621 505 779 97 839 399 340 850 285 222 122 850 826 352 849 823 815 598 340 534 207 48 555 492 340 460 629 511 594 850 578 249 426 288 352 709 850 716 340 8 745 181 599 504 27 320 850 779 501 59 399 824 495 108 599 97 60 218 764 778 539 598 399 790 352 396 672 696 534 376 460 495 527 444 495 83 53 74 138 666 433.

850 578 249 426 288 352 709 850 716 340 8 745 181 599 504 27 320 850 779 501 59 399 824 495 108 599 97 60 218 764 778 539 598 399 790 352 396 672 696 534 376 460 495 687 791 689 850 326 113 786 46 83 614 244 484 747 63 725 185 352 113 484 747 488 416 555 850 779 374 118 555 664 850 578 354 495 352 46 426 94 27 460 495 546 495 850 780 282 434 495 850 21 352 850 25 534 221 460 571 12 444 412 800 818 59 120 594 460 495 103 610 282 444 813 572 560 104 46 725 797 747 39 450 641 664 352 313 460 495 465 495 734 756 484 823 844 736 29 259 644 236 172 657 273 113 672 187 779 460 594 523 12 205 779 340 734 468 823 179 83 779 84 46 399 501 49 399 716 571 747 411 200 275 492 614 460 495 423 566 827 12 254 842 181 460 55 273 496 113 480 527 221 460 495 406 657 273 16 825 85 527 273 716 487 551 437 444 813 394 194 725

435 434 195 147 48 97.

Arrivé depuis quatre jours dans la colonie je ne puis Monseigneur vous rendre qu'un compte très sommaire de l'état où je la trouve. Il, paraît qu'il y regne de l'ordre et de la tranquilité 460 495 849 823 837 453 594 495 195 249 850 508 598 444 185 823 412 488 289 553 352 797 747 51 282 495 147 82 749 594 113 215 460 495 310 837 453 594 495 839 629 661 186 352 779 14 153 489 550 225 779 843 540 207 444 587 790 113 662 200 604 273 789 513 613 599 842 745 764 448 555 412 797 823 211 538 243 832 273 488 492 489 513 378 425 243 832 273 488 492 849 823 483 495 718 416 629 764 599 791 27 489 815 598 340 194 786 46 416 460 399 272 399 272 399 195 526 412 374 511 181 221 63 181 790 763 115 200 527 181 495 572 672 113 662 850 364 412 716 488 412 837 72 716 492 340 675 113 716 175 779 813 686 113 495 205 837.

599 244 745 764 412 413 378 425 513 849 823 243 403 565 134 850 277 849 823 513 492 774 664 258 850 398 492 774 689 604 273 789 258 850 118 495 492 837 278 191 243 282 683 134 850 527 29 813 352 403 683 59 118 172 495 850 170 832 273 108 779 374 118 555 689 195 436 504 664 489 805 113 658 850 12 63 686 492 324 639 571 468 497 63 399 88 399 320 832 340 578 19 818 832 282 813 756 594 495 53 74 138 253 296 433.

460 729 463 566 12 745 672 850 385 352 850 830 850 167 571 336 374 118 555 411 460 211 842 799 779 492 725 686 399 595 211 850 598 779 374 492 571 819 637 460 495 475 850 778 114 273 598 399 790 352 460 495 199 412 495 46 358 832 614 310 815 598 340 851 97 113 779 2 664 850 716 492 340 824 850 221 194 745 799 809 211 837 395 453 27 200 566 819 850 712 534 221 244 60 118 352 813 302 175 516 411 263 566.725 756 837 278 850 352 725 824 684 779 460 799 211 847 460 495 729 463 850 826 538 599 113 672 813 46 818 340 515 399 460 401 313 725 1 627 813 756 405 187 779 460 594 663 399 313 460 495 122 436 221 756 812 344 560 66 756 594 844 850 725 824 850 113 622 572 273 412 837 839 684 140 779 423 850 484 747 63 725 185 280 460 839 747 686 273 142 850 716 492 340 45 113 837 395 211 850 819 27 614 677 360 249 850 790 664 260 594 200 566 316 388 757 696 289 278 534 687 186 544 517 493 465 495 324 495 211 181 145 807 60 489 797 747 696 605 756 113 709 750 54 492 412 405 850 986.

N°	Cleartext	N°	Cleartext
1	*côte*	103	**différent(es)*
2	*entre*	104	**choisi(es)*
8	**flotte*	108	**mais*
10	**rouge or *blanc*	113	*a*
12	*con*	114	*bo*
14	*...*	115	*plus*
16	**ai*	118	**ri*
19	**ap*	120	**ble*
21	a name	122	*vaisseau*
25	*b*	132	*cou(r)*
27	*er*	133	*Amérique*
29	*lais*	134	**livre(s)*
39	*pa*	140	*vent*
44	*bra*	142	*dant*
45	*escadre*	145	*ac*
46	*pour*	147	*ac*
48	*or*	153	*...*
49	**pli*	166	*bu*
51	*bre*	170	**mauvais*
54	*at*	172	*que*
55	**cla*	175	*que*
59	*ba*	179	**par conséquent* or **inces-samment*
60	*pe*	181	*nt*
63	*so*	185	*no*
65	*fi*	186	*au(x)*
66	**prochain*	187	*nu*
71	*venir*	191	*ze*
72	*dans*	194	*a*
77	*ront*	195	*des*
82	*tu*	196	*lieu*
83	**faire*	199	*mi*
84	**impossible*		
85	**s*		
88	*be*		
92	*jour*		
94	**voy*		
97	*es*		

Table 2: (Partial) reconstitution of Bellecombe's cipher table for the numbers between 1 and 199.

The key.

N°	Cleartext	N°	Cleartext
200	*pre(s)*	302	**croit*
205	*su*	309	*ag*
207	*c*	310	*deux*
209	*dre*	311	*cha*
211	*se*	313	*sur*
215	a location	316	**fort*
218	*ro*	320	**bien*
221	*il*	324	*...*
222	*tre*	326	**suite*
225	*ta*	336	**est*
233	*quelque*	340	*te*
236	**ser*	344	*ir*
243	**cent*	348	*septentrional*
244	*a*	352	*et*
249	*troupe*	354	*force*
254	*cer*	358	*sui*
258	a quantity like **sacs*	359	*uni*
259	*ser*	360	*ses*
260	*barque*	361	*tenir*
262	*détroit*	364	**marchandise*
263	*per*	365	*ter*
272	*che*	374	**ar*
273	*n*	376	**voir*
275	*cri*	378	**dou*
277	**farine*	387	**re*
278	*i*	388	*du*
280	**aussi*	389	*me*
282	*si*	394	*qu*
285	*qua*	395	**fu*
288	*garnison*	396	*mo*
289	*h*	398	**sucre* or **farine*
291	*tour*	399	*r*

Table 3: (Partial) reconstitution of Bellecombe's cipher table for the numbers between 200 and 399.

N°	Cleartext	N°	Cleartext
401	*commerce*	501	*em*
403	**di(x)*	504	**cre*
404	*(r)ant*	505	*sous*
405	**doit*	511	**ge*
406	**dont*	513	**mille*
408	*après*	514	*encore*
411	*est*	516	*tout*
412	*en*	517	*...*
413	**provision*	523	*été*
416	**al*	526	**ustenciles*
420	*général*	527	*sa*
423	*ordre*	529	*li*
425	**ze*	530	*rt*
434	**eur*	534	*ou*
435	**lu*	538	**que*
436	*espagnol*	539	**bon* or **que*
437	**excep*	540	*...*
444	*ti*	542	*Espagne*
445	*voye*	544	*...*
450	*gne*	546	*dépêche*
455	*beaucoup*	548	*nation*
460	*le*	550	*...*
463	**bateau(x)*	551	**les*
465	**terre*	552	*honneur*
466	*affaire*	555	*ve*
475	*homme*	556	*France*
476	*jusque*	565	*neuf*
480	*ver*	566	*du*
484	*do*	568	*ger*
487	*suivant*	571	*qui*
488	*g*	572	*qu'il*
489	*un(e)*	578	**nos*
492	*t*	580	*an*
493	*e*	587	*f*
495	*s*	589	*même*
496	*arrêté*	592	*cap*
		594	*ment*
		595	*qui*
		598	*ga*
		599	*nous*

Table 4: (Partial) reconstitution of Bellecombe's cipher table for the numbers between 400 and 599.

N°	Cleartext	N°	Cleartext
604	*ci*	697	*français*
605	*ag*	706	*eau*
607	*détail*	709	*celle*
610	*po*	714	*os*
613	**barque*	716	*ce*
614	*par(t)*	725	*la*
619	*pi*	728	*ha*
621	a (likely) Spanish location	729	**seul*
622	**moins*	734	*je*
623	*voile*	735	*étant*
625	*ayant*	736	*vous*
627	**méridional(e)*	738	*jamais*
629	*lo*	745	*vo*
635	**ant*	747	*m*
637	**ait*	749	*elle*
639	*on*	750	*qu'il*
641	*marin*	756	*ne*
644	**pen*	757	*u*
657	*je*	759	*xe*
658	*tion*	761	*jo*
660	*de*	762	*avec*
661	*ni*	763	**autant*
662	*peu(t)*	764	*ns*
663	*compte*	765	*u*
664	*e*	774	**rent*
669	*semble*	778	*monsieur*
670	*ici*	779	*l*
672	*y*	780	*mes*
675	**colonie*	784	*v*
677	*mer*	786	*La Havane*
683	*x*	789	*q*
684	*i*	790	*d*
686	*ma*	791	**envoye(r)*
687	*je*	797	*ca*
689	*e*	799	*u*
696	*p*		

Table 5: (Partial) reconstitution of Bellecombe's cipher table for the numbers between 600 and 799.

N°	Cleartext	Null
800	*t*	53
805	*augment*	74
807	...	138
808	*el*	253
809	**lu*	296
812	**par(t)*	499
813	*on(s)*	666
815	*fre*	730
818	*pro*	
819	*port*	
823	*is*	
824	*ra*	
825	**connai*	
826	**guerre* or **combat*	
827	**roy*	
830	*celui*	
831	*ot*	
832	*vi*	
837	*re*	
838	*le*	
839	*co*	
842	*na*	
843	...	
844	**pas*	
847	**sont*	
849	*tro*	
850	*de*	
851	**anglais*	
852	*curacao*	
856	*France*	
857	*cap*	

Table 6: (Partial) reconstitution of Bellecombe's cipher table for the numbers between 800 and 899, together with null symbols.

Solving a 750-Letter General Bigram Substitution Challenge

Elonka Dunin
Codebreaking-guide.com
elonka@gmail.com

Jarl Van Eycke
Independent Researcher
jarlve@yahoo.com

Louie Helm
RockstarResearch.com
louiehelm@protonmail.ch

Klaus Schmeh
Codebreaking-guide.com
klaus@schmeh.org

Abstract

The general bigram substitution cipher is an encryption method originating in the Renaissance. It operates using a substitution table that maps each possible letter pair (bigram) to a unique replacement. While conceptually straightforward, this cipher is notably challenging to break, particularly when dealing with short ciphertexts. To inspire further research, one of the authors initiated a bigram substitution challenge featuring a 750-character ciphertext. In this paper, we present the solution to that challenge, achieved by two other authors using a hill climbing algorithm combined with a scoring function based on 8-gram (eight-letter sequence) frequencies. Since no prior 8-gram frequency statistics existed for the English language, one of the authors developed a comprehensive dataset by analyzing 2 terabytes of text, including 5.8 million books and the entire content of Wikipedia. This achievement, to our knowledge, marks the shortest bigram substitution ciphertext ever successfully decrypted. Furthermore, we propose a new challenge based on a 600-character ciphertext and invite readers to tackle it, setting the stage for future advancements in this field.

1 Introduction

A bigram, or digraph, is a pair of letters, such as PC, QE, IE, or WW. In the commonly used 26-letter Latin alphabet, there are 676 possible bigrams, calculated as 26×26. The general bigram substitution cipher is an encryption technique that replaces each bigram with another bigram, a symbol, or a sequence of characters, following a predefined substitution table. This table can be constructed using a password or another mnemonic method. However, for the purposes of this paper, we assume that the substitution table is always generated randomly.

The earliest known general bigram substitution cipher was introduced by Giovanni Battista Porta in his 1563 book “De Furtivus Literarum Notis” (Kahn, 1996). Porta's system utilized a 20-letter alphabet, resulting in a substitution table with 400 entries (Figure 1). To implement this, Porta employed 400 distinct glyphs in his substitution table. In contrast, the bigram substitutions discussed in this paper involve replacing letter pairs directly with other letter pairs, adhering to a more modern approach.

2 The Playfair cipher

A variety of specialized versions of the general bigram substitution can be designed by replacing an exhaustive substitution table with a more concise set of rules. A well-known method of this type is the Playfair cipher, which operates on a 25-letter alphabet (Dunin and Schmeh, 2023). This cipher employs a simple yet effective set of four substitution rules applied to a 5×5 letter matrix, making it significantly more practical than a general bigram substitution system. The matrix, which serves as the cipher's key, can either be generated from a keyword or selected at random—the latter offering greater security.

As with all substitution ciphers, cryptanalysis of the Playfair cipher becomes more difficult if there is less ciphertext to analyze. The shortest

Proceedings of the 8th International Conference on Historical Cryptology, pages 133-140
Poznań, Poland, 16-18 June, 2025

random-matrix Playfair ciphertext that has ever been broken consists of 26 letters (Dunin et al. 2021).

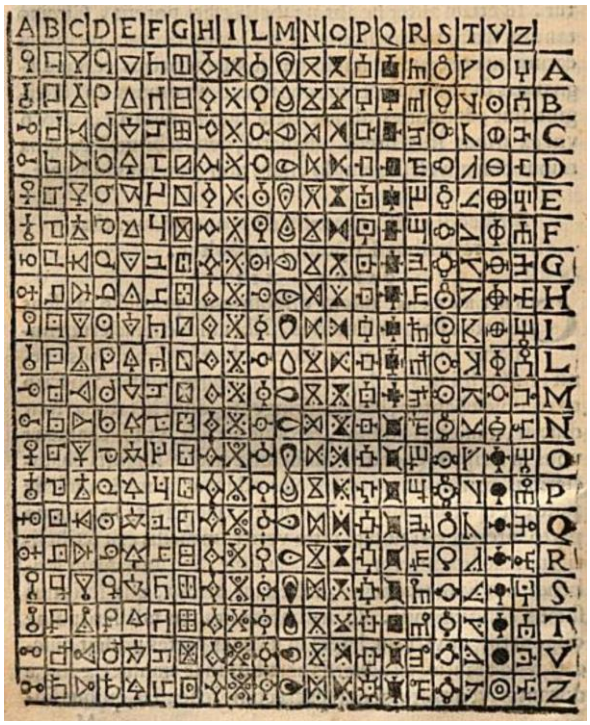


Figure 1. The earliest known general bigram substitution, published in the 1563 book “De Furtivus Literarum Notis” by Giovanni Battista Porta.

The general bigram substitution has never attained the widespread popularity of the Playfair cipher, likely due to its greater complexity. However, it is evident that the general bigram substitution offers a higher level of security compared to the Playfair cipher, though the exact extent of this advantage remains uncertain. Until recently, detailed information regarding the cryptanalysis of the general bigram substitution was notably lacking.

3 The first three challenges

To inspire research on this question, in 2017 one of the authors (Klaus) published two challenges (Schmeh, 2017): a 5000-letter message and a 2500-letter message, both encrypted with a general bigram substitution. Norbert Bierman solved these challenges within a few days. Other interesting comments came from Thomas Bosbach, Thomas Ernst, George Lasry, and Armin Krauß. The method used to break the two ciphertexts was hill climbing (including simulated annealing) with a scoring function based on 4-gram and 5-gram frequencies. To our knowledge, the 2500-letter cryptogram became the shortest one of its kind ever solved.

Building on this success, Klaus introduced a third general bigram challenge ciphertext, comprising 1,346 letters (Schmeh, 2019a). Norbert Biermann deciphered it within a few weeks (Schmeh, 2019b), setting another new world record. Once again, the breakthrough was achieved using hill climbing. This time, the scoring function was based on 6-gram frequencies.

4 The 1000-letter challenge

Next, Klaus created a fourth challenge. This time, he took a plaintext consisting of exactly 1000 letters and encrypted it with a general bigram substitution. In October 2019, he published the resulting cryptogram online (Schmeh, 2019c). Again, the challenge was broken within days, which meant that a new world record was set (Schmeh, 2019d). This time, the solution came from Jarl and Louie, who are among the authors of this paper. Richard Bean, George Lasry, Dave Oranchak, and Christoph Tenzer provided helpful comments.

Jarl is the developer of the software package AZdecrypt (Van Eycke, 2016). Initially conceived in late 2014 as an evolution of a similar program, AZdecrypt underwent significant enhancements. By early 2016, it was officially released with a Windows graphical user interface (GUI), becoming instrumental in research on the Z340 cipher associated with the Zodiac Killer (Kopal, 2019; Oranchak, 2020). During this project, the software's capabilities expanded with additional solvers and features. Over the years, AZdecrypt has played a key role in decrypting several previously unsolved ciphers (Schmeh, 2021; Vierra, 2023). The software, along with its source code and n-gram data, is freely accessible online.

In order to solve the 1000-letter bigram challenge, Jarl optimized the code of AZdecrypt for deciphering bigram substitutions. The main technique to be applied was hill climbing. The

program's scoring function relied on the frequency analysis of 8-grams (eight-letter blocks). To our knowledge, this marked the first ever application of 8-gram statistics in cryptanalysis.

Since no 8-gram statistics were available, Louie took the initiative to create one. Working with 8-grams presents significant challenges due to their sheer quantity; in a 26-letter alphabet, there are 26^8 possible combinations – approximately 200 billion. Compiling meaningful reference statistics requires an immense volume of text. Louie gathered approximately 2 terabytes of English text sourced from the following materials:

- 1.3 million English public domain books from Project Gutenberg
- 4.5 million potentially copyrighted English books
- All English Reddit comments and submissions
- All of the English Wikipedia from a dump as of 2019-02-25.
- All English Project Gutenberg books
- All of the English subtitles of every movie ever released
- All lyrics of all available English songs ever produced
- 7 billion words from Usenet posts
- 34 million sentences from online news stories
- 135 million online reviews from Amazon, Tripadvisor, and Yelp
- 4.4 million Yahoo Answers exchanges
- Louie's own re-creation of the OpenAI GPT-2 data model

As handling such enormous quantities of data requires a large amount of memory, Louie developed several quirks in how 8-grams are compiled, encoded, and loaded into memory. First, he cleaned up the data with a chi-square filter in order to remove text from other languages. Using Index of Coicidence (IoC) and entropy filters, he removed overly redundant and sparse information. All these methods to clean up the input data are simple and were easy to implement.

Next, the 8-grams were restricted to encodings where both the initial and final four-letter subgrams were valid English language combinations (this included about a quarter of all possible combinations). Each one was given 8-bit capped-log-frequency scores and stored in a compressed pointer table and compressed with the Gzip algorithm.

In addition, Louie developed a novel n-gram format designed to efficiently handle large datasets. The process begins by calculating 4-gram frequencies from the corpus. From these, the top 100,000 4-grams (as an example, based on a chosen metric) are selected. A new table is then created, structured as a two-dimensional array (100,000 × 100,000) with 8-bit values (log probabilities), requiring approximately 10 GB of storage.

This table is indexed using a 4-gram lookup table, which maps each 4-gram to the corresponding values in the main table. The main table is structured as an array ($4g(x_1, x_2, x_3, x_4)$, $4g(x_5, x_6, x_7, x_8)$). If a given 4-gram has a frequency of zero, its associated 8-gram value is automatically set to zero. This acts as a low-level filter, ensuring that higher n-gram sizes, which are inherently more sparse, are managed effectively.

Another benefit of this system was leveraged by Jarl in AZdecrypt. If the target computer does not have enough RAM available, the table size (100,000 for example) can be adjusted on load time. This allows the n-gram format to work on less powerful machines, albeit with diminished capacities.

Louie's n-gram system is also valuable for lower n-gram sizes. For example, 6-grams can now be made to be much more cache-friendly and therefore run much faster on mainstream computers where the letter n-gram table does not entirely fit in the L3 cache (the L3 cache is the largest cache in current CPU architectures).

The resulting body of text provided 2,062,507,743,806 samples of 8,178,871,377 unique 8-grams. The subgram structure reduced the final file to 3,631,818,052 "valid" English 8-grams, which could be comfortably loaded into 14 GB of memory (instead of 195 GB for a naïve implementation). Jarl and Louie also tried larger models, close to 64 GB, but they observed reduced convergence and slower performance. This happened because many of the $26^4 = 456976$ existing 4-grams (such as XZQQ) don't appear at all, while others (such as XAIB or LMAO) make their way into the raw text data via foreign languages or internet slang. This distorts English 4-gram frequencies if one begins including much beyond 100,000 4-grams. As a consequence, 14 GB turned out to be the best memory size.

To apply hill climbing, AZdecrypt first created two new symbols for every unique bigram based on the following coding method: JOININGTHEJOINTS = 1 2 3 4 3 4 5 6 7 8 1 2 3 4 6 9. Then it changed the key (i.e., the substitution table) one symbol at a time. Bigram homophones (i.e., several ciphertext bigrams decrypting to the same plaintext bigram) were allowed but punished (i.e., they led to a lower result of the scoring function). After a few days of optimization, the program was able to consistently solve the 1346-letter challenge (the one already solved) in less than five minutes even without requiring a crib.

After this success, Jarl felt confident and applied AZdecrypt on the 1000-letter challenge. He started it running before going to work. When he came back, the software had returned a 70-80% accurate decryption in about 4 hours. He shared this result with Louie. They cleaned it up and submitted it to Klaus. Even though it wasn't a 100% solution, it was close enough that it was aceepted.

5 The 750-letter challenge

The next bigram challenge Klaus created was based on a plaintext consisting of 750 letters. The ciphertext is provided in the following (Schmeh, 2019e):

YYXFTVUJKXMYWODAWFZPSAPPVDW
NEXAJXFPPRXKCMFBZIXDLTCVIBSKLZO
XIUKPEMUXFEMDUOGPCRRMWZSVBNM
YYSHLWCIAJJWORCFCHKYRXYYJVUPAG
JHBZAJZPCJSEWZSEWZCJLFWOFHSAEMX
ZZUJHLNGNNMYYIXUVNMYYIXBWAOK
YJRYCHUBMNOQTXAPCRRMWPPWZAML
LPCXFEMWFITKYPGISZEKJMOMUXAERE
KWGQTEOXILBUGGNTCYOYAHUUQZNK
YBJADXIAFICRWCRFPPGZIEEBZHUIWKR
KERRLZWFGQNAJRJQNTPYKBPEKBDLNG
DYXPVAZSSKUVHUBBDLXAWFZUPNHZC
CRXGOLFZUHUGNVWDYRRSAJHTRZUXA
XPKMYYYCHRXZDUQSLFDYKJIAZIDLGQ
NAQXBVWRSWGXPPAJMPDUPPVWAVNA
ORHUUWNBLNFMBSSAPPDVGCGCWFWY
DYZEWOPETHDLMUZURXKJHMKJYUBV
OJWYDYUGCYZPZIDLXFLWPCFSEXZRWF
ERWFIXDTYYWUVJPNAJZURXTFHZOAX
ALZXITHDLBSKLZOXIUKJPYYSHLWCIAJ
XFZURLVWUNPCHUPTXZHCAJANBWLPK
MHUVCWRKXKMBVCXCTHUHMNCQXVB
TCNGADRHPCKWUGRRKBRQXFPGWAMU
DYIXDLKJJSDUOGQTRRKBXILBUGBBIPD
LXZZUWAOSDLYYZPYAZSVBKBGCJPUJX
LLHDYIAKBVBZENMVCRWFA

Again, Jarl and Louie solved the challenge (Schmeh, 2019f). They needed about a week. Here is the plaintext they found:

IN THE FOLLOWING YEAR FOSTER
ACQUIRED THE RIGHT TO USE THE NAME
MARLBOROUGH AND THE MODEL
DESIGNATION NINE HUNDRED WHICH
HAD ORIGINALLY BEEN USED FOR AN
OPEN OPERATING SYSTEM PRESENTED
IN NINETEEN NINETY BY NORWEGIAN
SOFTWARE DESIGNER PETER IDE THE
MARLBOROUGH NINE HUNDRED WHICH

WAS PRODUCED IN A GERMAN FACTORY IN PROBABLY OVER FOUR THOUSAND STYLES IS FAR LESS WELL KNOWN THAN THE LION GAMMA THREE AND THERE ARE SOME AMBIGUITIES AND INCONSISTENCIES REGARDING ITS PRODUCTION NEVERTHELESS IT IS CONSIDERED A MODERN CLASSIC AND STATE OF MASTERPIECE CARS CONNINGHAM THEN DESIGNED A NEW DISCUS CONCEPT FOR THE THUNDERBIRD HARDWARE TRYING TO SOME DESIGN FEATURES FROM THE MARLBOROUGH NINE HUNDRED THESE INCLUDE AN IMPROVED KEYBOARD A NEW COLOR DISPLAY AND ADDITIONAL INPUT DEVICES IN TWO THOUSAND ONE THE NEW MODEL WAS INTRODUCED TO THE PRESS AT THE INFORMATION TECHNOLOGY CONVENTION IN NEW YORK

The plaintext originates from a non-English Wikipedia article, translated into English using DeepL. During the process, Klaus altered all names, locations, technical terms, and numerical data. As a result, the plaintext became entirely unique and had never been published before, rendering it untraceable through online searches.

6 Solution

The program first produced a number of pretty convincing "phantom solves", including one that began: "AT THE FESTIVAL OF THE BULLS…", which scored nearly as well as the eventual solution. Louie spent a day fruitlessly trying to solve the challenge using various strategies along with this false crib.

After several days of no progress, Louie switched to a development version of his 8-gram file that now additionally included all the Twitter (now known as X) data and also some other new extremely large data sources. This model was generally a few percent better in solve accuracy given enough computing time.

After six hours, the program found a solution that was 55.93% letter-accurate, which Louie only noticed after it had been running for about 14 hours. At this point it had become fairly clear that the message the program found was fairly single-topic, so he emailed it to Jarl, believing it was possible that at least some portion of it was correct. Then two hours later, the program refined the result to a 69.49% letter-accurate solution.

Louie helped Jarl crib several portions that were completely unconstrained (with no repeated symbols anywhere in the ciphertext). After a few messages back and forth, the two were able to clean up the solution to a satisfactory result they felt confident in. Nevertheless, there were still a few portions they could not be completely sure of. The "Peter Ide" portion could easily be "Peak Code" or something similar. There was not enough symbol reuse to know for sure.

Jarl and Louie contacted Klaus with their work thus far, and Klaus confirmed the correctness of the solution, which meant that the world record in solving short general bigram substitution ciphertexts had been improved to a ciphertext length of 750 letters.

The main differences between their approach to the 750 versus the 1000 bigram challenge was a slightly more powerful 8-gram model, using more aggressive annealing parameters, some generalized convergence improvements that Jarl added to his program, and a new version of Peter Norvig's word-gram model for automated word division that Louie recently rewrote.

7 The 600-letter challenge

In March 2020, Klaus presented a new general bigram challenge, which as of this writing remains unsolved. It consists of 600 letters (Schmeh, 2020):

UGBZAEHINYQLBPZLNFTLUEBMULTLSL
ZPBZPZKPOVUGYSQPNYHLRYFHATQKR
HTZEHPDQUUGYSUJOVYTUGYVRHAJNF
TLUEXFRUEOOJTZOSLUPZEICVADYMYL
CRBZXOUGSVDJOIDYRHTZOSWZROYNKJ
RMEIXOREOVNFTLUESAMNDJHIIWJGKR
YFUBTIQPULBPRMJORECJCYWZZPQRXX
VNOSZLBLNYJMPLYNOVLCLKIOGUKUKF
SAKAQRSVQXUJIOANYSWZSDKUKFLNR
MEIRJYVEOLXLKMEYKERHXZPBZXOZX

QPCRKSYOSVHNTLIXKRYFUBTIMGWIZL
OSONRMIDKYNYLCFFOMTTLLJHWTADH
LYNRHMZADOGMUKBWZZPPQBZBZNOC
RHINYNFTLUEYNOVBZNOQPGCQMRHTZI
DKYNYCRBZXOUGSVTTQPOSDYXOMQK
KVNEALUYVRMUFPYNXZAVLRHTZNYQ
XMFYVUCMZSAJMBZZPXPBZMNVFUCJT
NYQXGHEITPPYFWKUZFPZQUDEVLDBO
MGRUEKFSCYTVNANLDRMNBYVUTFNUJ
MUMMEOIXIISDVNZPMNRYRCTFUGZPD
NUTLXJNSSVNCRJC

The plaintext is in English. Can a reader break this challenge? If so, they will set a new world record.

8 Conclusion and outlook

The main conclusion of our paper is that it is possible to solve a 750-character general substitution cryptogram using hill climbing. To prove this required significant effort from two of the authors. Their work suggests that solving shorter cryptograms of this kind is likely going to be even more challenging.

It is worth noting that previous challenges of this nature were solved with hill climbing with scoring functions based on 4-gram or 5-gram frequencies. In contrast, the cryptanalysis of the 1000-character and the 750-character ciphertexts relied on 8-gram frequencies. This indicates that hill climbing becomes increasingly effective as the value of *n* in n-gram statistics rises. However, while 9-grams (nine-character blocks) instead of 8-grams might offer even greater potential, finding sufficient amounts of text to produce meaningful statistics remains a significant challenge.

The authors of this work have no proof that the conjecture "the greater *n*, the more powerful are n-gram frequency statistics" is correct for hill climbing and in case it is, why. If a reader knows more about a visible proof, we would be interested to know.

Jarl states "There are practical reasons why greater n-gram sizes may not continue to scale or work well. For example with 10-grams or 12-grams, a unique unseen plaintext may have many 12-grams that are entirely unique to the plaintext. The hill-climber will then cause havoc within this zero space and the 12-grams will lose their strength since they cannot properly discriminate."

The difficulty of breaking general bigram substitutions has to be contrasted with the Playfair cipher, which is a special case of the general bigram substitution (Dunin and Schmeh, 2023). As mentioned, a Playfair ciphertext with only 26 letters of ciphertext can be solved. Comparing that with the difficulty of solving the general bigram substitution, we can conclude that the latter is by two orders of magnitude more secure than the Playfair. It is noteworthy that an even shorter Playfair ciphertext is solvable with other techniques such as a dictionary attack, but this is not applicable on the general bigram substitution, provided that the substitution table is generated at random. On the other hand, a general bigram substitution requires a large table and is therefore considerably more difficult to handle than a Playfair.

While the 750-letter challenge was solved, as of this writing in 2025, the 600-letter variant has remained unbroken for four years. This suggests that the limit may have been reached. A detailed evaluation of this question based on confidence limits (Kubáček 1994) is not within the scope of this work.

Jarl and Louie state that they are working on the 600-letter challenge. Louie has proposed a more efficient hill-climbing approach by replacing the current approach that includes homophones with a true one-to-one bigram substitution solver. Following this suggestion, Jarl developed a proper one-to-one bigram substitution solver, which has shown significantly better performance compared to the original method. Additionally, Louie has compiled new 8-gram statistics using a considerably larger dataset. The amount of text to produce these statistics has grown from 2 terabytes to 10 terabytes. The additional 8 terabytes of data came primarily from more Reddit data (produced from 2020-2024) and ParaCrawl (English ParaCrawl data is also a processed form of CommonCrawl) (Figure 2).

In addition, Louie improved the pre-filtering, cross-weighting between different corpora, and the use of dynamic range in model representation. With these improvements, data like Bitcoin addresses or long strings of non-English text can still be used, which will help lead to more accurate frequency counts.

Meanwhile, Jarl for his part has been focusing on compressing n-grams using neural networks with binary weights optimized for fast CPU retrieval. He hopes this approach will complement Louie's n-gram system, particularly for larger n-grams, as Louie's system already effectively handles unseen n-grams.

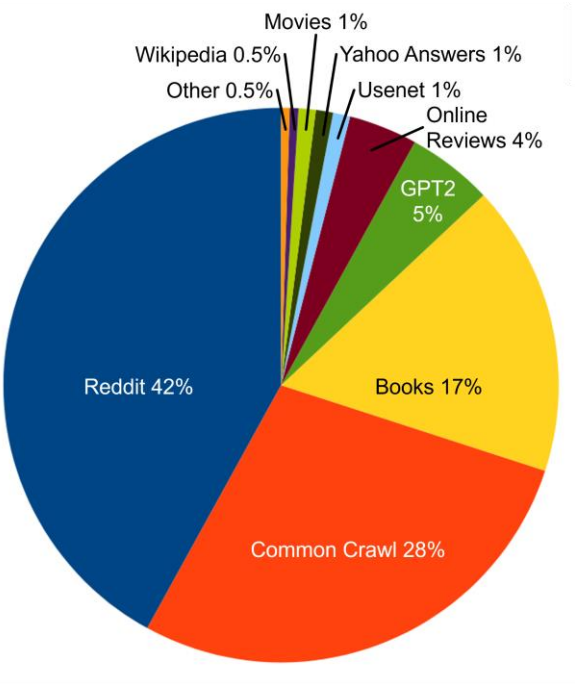


Figure 2. The amount of text used by Louie to generate 8-gram statistics has grown to 10 terabytes. The sources are shown in this diagram.

The enhancements currently being made by Jarl and Louie to their software tools may ultimately pave the way for solving the 600-letter challenge.

Acknowledgments

The authors would like to thank Richard Bean, Norbert Biermann, Thomas Bosbach, Thomas Ernst, Nils Kopal, Armin Krauß, George Lasry, Dave Oranchak, and Christoph Tenzer.

References

Elonka Dunin, Magnus Ekhall, Konstantin Hamidullin, Nils Kopal, George Lasry, Klaus Schmeh. 2021. *How we set new world records in breaking Playfair ciphertexts*. Cryptologia Volume 46, 2021 (4): 302-322

Jarl Van Eycke. 2016. *AZdecrypt 1.22*. https://forum.zodiackillerciphers.com/community/zodiac-cipher-mailings-discussion/azdecrypt-1-19b/

Elonka Dunin, Klaus Schmeh. 2022. *Codebreaking: A Practical Guide*. No Starch Press, San Francisco: 245-247

David Kahn. 1996. *The Codebreakers*. Scribner, New York: 138

Nils Kopal. 2019. *Cryptanalysis of Homophonic Substitution Ciphers Using Simulated Annealing with Fixed Temperature*. https://ep.liu.se/ecp/158/012/ecp19158012.pdf

Lubomir Kubáček. 1994. *Confidence limits for proportions of linguistic entities*. Journal of Quantitative Linguistic Volume 1. 1994: 56-61

David Oranchak. 2020. *The 340 Is Solved!* https://www.youtube.com/watch?v=-1oQLPRE21o

Klaus Schmeh. 2017. *Bigram substitution: An old and simple encryption algorithm that is hard to break*. https://scienceblogs.de/klausis-krypto-kolumne/2017/02/13/bigram-substitution-an-old-and-simple-encryption-algorithm-that-is-hard-to-break/

Klaus Schmeh. 2019a. *Can you solve this bigram challenge and set a new world record?* https://scienceblogs.de/klausis-krypto-kolumne/2019/07/13/can-you-solve-this-bigram-challenge-and-set-a-new-world-record/

Klaus Schmeh. 2019b. *Norbert Biermann solves bigram challenge and sets a new world record*. https://scienceblogs.de/klausis-krypto-kolumne/2019/08/13/norbert-biermann-solves-bigram-challenge-and-sets-a-new-world-record/

Klaus Schmeh. 2019c. *Solve this bigram challenge and set a new world record*. https://scienceblogs.de/klausis-krypto-kolumne/2019/10/07/solve-this-bigram-challenge-and-set-a-new-world-record/

Klaus Schmeh. 2019d. *Bigram 1000 challenge solved, new world record set*. https://scienceblogs.de/klausis-krypto-kolumne/2019/10/27/bigram-1000-challenge-solved-new-world-record-set/

Klaus Schmeh. 2019e. *Solve this bigram challenge and set a new world record*. https://scienceblogs.de/klausis-krypto-kolumne/2019/12/12/solve-this-bigram-challenge-and-set-a-new-world-record-2/

Klaus Schmeh. 2019f. *Bigram 750 challenge solved, new world record set*. https://scienceblogs.de/

klausis-krypto-kolumne/2019/12/19/bigram-750-challenge-solved-new-world-record-set/

Klaus Schmeh. 2020. *Solve the Bigram 600 challenge and set a new world record*. https://scienceblogs.de/klausis-krypto-kolumne/2020/03/03/solve-the-bigram-600-challenge-and-set-a-new-world-record/

Klaus Schmeh. 2021. *Jarl Van Eycke löst 400 Jahre alte Längengrad-Botschaft*. https://scienceblogs.de/klausis-krypto-kolumne/2021/02/18/jarl-van-eycke-loest-400-jahre-alte-laengengrad-botschaft/

David Vierra. 2023. *Solutions to Feynman Ciphers #2 and #3*. https://codewarrior0.github.io/cipher-blog/2023/05/27/feynman-solved.html

A Florentine 'polyalphabetic' cipher in the 15th century

Marco Vito
marcovito001@gmail.com

Abstract

The 15th century in Italy was a period of revolution in cryptography. Leon Battista Alberti developed the first western polyalphabetic cipher, while the monoalphabetic system spread throughout the peninsula. The aim of this study is to present a never before published late medieval 15th-century Florentine polyalphabetic cipher, explain its functioning, and shed light on a system—specifically the polyalphabetic cipher—that, although seemingly unused during the 15th century, was in fact employed in Florentine diplomacy.

1 Introduction

The 15th century is the period in which traditional cryptography develops and is enriched by various forms of ciphering. The use of monoalphabetic substitution ciphers, homophones to reduce frequency, nulls to make decryption more difficult, as well as alchemical, astral, planetary and numerical signs, are all examples of a revolution in the field of cryptography. The most important of these innovations, although the least adopted by contemporaries in the 15th century, was the polyalphabetic system. The first Western inventor of this form of secret writing was Leon Battista Alberti in his *De Componendis Cyfris*. In his 1466 work, he devised a system that no longer relied on a single alphabet, but on multiple alphabets that could be used within a single encrypted message, thus giving birth to the Alberti cipher.

Alberti's work, which remained unpublished for some time, was followed a century later by Giovan Battista Della Porta in *De furtivis literarum notis vulgo de ziferis libri IIII*, who, following Alberti's model (but without mentioning it), described the use of cipher discs to write polyalphabetic ciphers[1].

Cipher disc systems were considered extremely useful and secure by Alberti himself:

> "Nunc de scribendi ratione a nobis inventa dicendum sequitur. Habet ea quidem has commoditates, nulla omnium qua qui suti possit cyfra expeditior, nulla scribitur commodius, nulla ubi ex instituto modum teneas, promptius apertiusque legitur, nulla (indices constitutos inter me atque alium, ad quem, scribo, si ignoraris) excogitari ptest obscurior"[2].

The importance of polyalphabetic cryptography (primarily in its theoretical formulation) had a greater impact in modern times, while in the late Middle Ages, apart from Alberti's treatise, there are no sources on the subject. With this in mind, it is interesting to note that among the Italian ciphers, there are mainly monoalphabetic substitution ciphers with homophones, followed in very small numbers by ciphers using only nomenclators and jargon ciphers. However, there is a cipher in the State Archives of Florence that could be considered as polyalphabetic, the only existing unpublished example currently known, demonstrating the intention to create a cipher with a mobile alphabet rather than a static one.

The cipher, located within a miscellany (miscellanea repubblicana),[3] appears in isolation and is not part of a homogeneous corpus. The file also separately includes a second, distinct cipher, this one written in jargon.

The difficulty in its identification stems from its archival placement, which is detached from

[1] See (Della Porta 1563).

[2] (Alberti 1994) P. 41. English translation: *We must now consider the method of writing which we have discovered. It has these advantages, each of which is more convenient for writing in cipher, Nothing is written more conveniently, nothing is written more easily and openly, where one sticks to the established manner, nothing (Indices established between myself and another, to whom I write, if you do not know) is more obscure to solve.* A revised and reprinted edition of the volume was issued in 1998 (Alberti 1998).

[3] Archivio di Stato di Firenze (ASFI). *Miscellanea Repubblicana 11*. fasc.292, I. All images are reproduced by permission of the Italian Ministry of Culture and the State Archives of Florence.

Proceedings of the 8th International Conference on Historical Cryptology, pages 141-148
Poznań, Poland, 16-18 June, 2025

the original context in which the cipher was conceived and used. The author is unknown, most likely a chancellor for whom no certain information is available. The recipient, however, is known. Moreover, compared to many other ciphers currently under study, this cipher represents the only example of a 'polyalphabetic'[4] cipher.

This cipher allows us to see how Florence, even in the time of Lorenzo de' Medici, could have been at the forefront of cryptography and, above all, provides a point of comparison with other Italian contexts[5]. The Sforza case, for example, is one of the most analysed examples from the 15th century, showing that among the extensive series of ciphers, monoalphabetic ciphers with homophones were by far the most widely used, almost exclusively[6].

The Florentine testimony allows us to observe how, during the times of Lorenzo the Magnificent, there was a constant attempt to improve existing ciphers by experimenting with different and new ways of writing in code.

2 Context of the cipher

The fifteenth century in Italy was a turbulent period for all the forces involved. Lorenzo de' Medici, for his part, sought to be at the centre of these various forces, positioning himself as a mediator between competing territorial claims[7]. Especially in the 1480s, there were several destabilising factors, such as the War of Ferrara, which ended within two years with the Peace of Bagnolo (1482-1484), and the conspiracy of the Neapolitan barons (1485-1486), which took place at a time when Florence was allied to King Ferrante of Naples. There was also the internal front, with Florence concentrating on expanding and countering the Genoese in the Lunigiana[8].

The cipher provides no historical evidence of its use, other than the intended recipient of the cipher itself. Indeed, on the back of the sheet we can read 'Cifera di Livorno Conposta per la buona aromoria[9] di Piero Capponi'[10] (Cipher of Livorno, composed for the good harmony of Piero Capponi). Moreover, the word 'Ciffra' is written vertically on the verso.

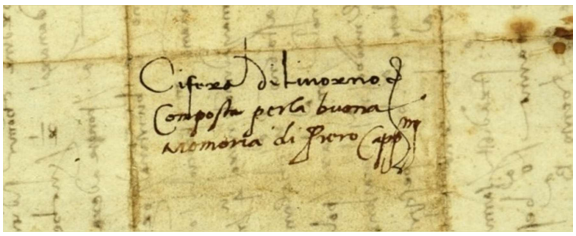


Figure 1. Detail of the back of the cipher, with reference to the holder

Piero Capponi was a loyal supporter and close associate of Lorenzo the Magnificent, serving as his ambassador and military commander on several occasions[11].

Capponi visited Livorno on two separate occasions. The first time was in December 1484, when he was appointed commissioner for the war against Genoa - a mission that ended in failure due to a storm during the expedition against the Republic.

In January 1486, Capponi was sent to Naples to support the Aragonese, who were then at war with the Papacy, allied with the mercenary captain Roberto Sanseverino. Capponi remained with the Aragonese allies until September of that year.

Following these events, Capponi returned to Livorno a second time in 1486-1487, this time as commissioner for the war against Sarzana.

Since Livorno and Piero Capponi are the only references given by the document, it is possible to date the cipher to the period between 1484-1485, corresponding to his mission against Genoa, and 1486-1487, corresponding to his mission against Sarzana.

Considering the events in Naples and likely his need to return to Livorno, it is reasonable to suggest that the cipher should be dated to 1487.

[4] The term is placed in quotation marks, as it was not employed by the cipher's creator or scribe, but came into scholarly use only at a later phase. Example (Sacco 1947) and (Zanotti 1928).

[5] The case of the Papal States, beginning in the Middle Ages, offers significant evidence of a sustained and systematic effort to refine and develop the secret writing practices in use, reflecting an evolving awareness of cryptographic needs within the administrative and diplomatic apparatus (Meister 1906).

[6] (Cerioni 1970).

[7] On Florentine diplomacy, see (Santini 1922) and (Lang 2014); for a general overview, see (Queller 1967).

[8] For further insights into the Italian context, with a specific focus on the Florentine setting, see (Fubini 1994).

[9] Uncertain transcription, as the text has an apparently incorrect spelling, most likely to be interpreted as 'armonia' for harmony.

[10] The italicised letters indicate the expansion of abbreviated words, in which one or more letters have been omitted through the use of abbreviation marks.

[11] (Mallet 1976).

3 The ‘polyalphabetic’ cipher

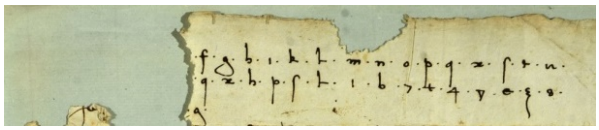


Figure 2. Cipher, detail of the alphabet (at the top the one in clear, at the bottom the one in figures)

The Florentine example makes it possible to analyse the cipher system and to understand it thanks to the work of the chancellor who composed it.

The document is part of the Miscellanea Repubblicana 11 fasc. 292 Cifra I and is in poor condition, with some tears around the cipher alphabet. However, it is still possible to understand how it works.

The cipher system in question consists of a single cipher alphabet which corresponds to the regular alphabet in a one-to-one relationship, meaning that for each letter of the alphabet there is a corresponding cipher symbol or letter, without the use of homophones. On this premise, it would appear to be a monoalphabetic substitution cipher with no distinguishing features. However, the writer provides a detailed explanation of how it works below the alphabet.

The writer offers an example word for coding: ‘*papa’* (pope)[12].

The first letter corresponding to the cipher to be used for encryption is taken (pigliasi i choxi che la prima lettera cum*que* a pro primo poi si ch*on*ta insino alla lettera che tu vuoi trovare). Since ‘*papa’* is the word to be encrypted, the corresponding cipher symbol for the letter ‘p’ is read from the cipher alphabet, which is ‘t’ (Chomincia alla prima lettera ad *esempio* a. b. c. *etc*. e troverrà che al p viene un t).

At this point the polyalphabetic factor comes into play, as the alphabet changes the cipher symbols for each letter, effectively creating a new cipher alphabet. This happens because once the single letter is ciphered, the next cipher symbol becomes the equivalent of the letter ‘a’ (the first letter of the alphabet). Specifically:

> ‘e poi rechomincia alla . a . che un 4. e poi rechomincia alla . a. un’altra volta de la ‘Ȳ’

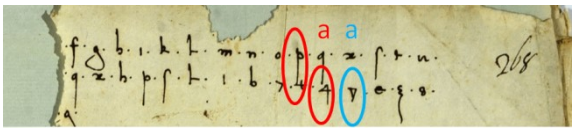


Figure 3. Cipher system: shift of the cipher sign and new alphabet. After ‘p’ the sign corresponding to ‘q’ becomes the sign to indicate ‘a’, the second letter being the second letter for papa, the next sign becomes the new sign to indicate the letter ‘a’

The cipher changes and the sign corresponding to the following letter becomes the new sign for ‘a’ so for the second letter of papa, ‘a’ now corresponds to ‘4’ and is the second letter in the cipher. For the third letter is repeated, the cipher following ‘4’ that is ‘ȳ’, now corresponds to ‘a.’ Scrolling through the letters, ‘p’ now corresponds to ‘l,’ which becomes the third letter of the word papa. Finally, the cipher symbol following ‘l’ is ‘i’ which now corresponds to ‘a’. Therefore, to write ‘*papa*’, one has to write ‘t4li’ (ntruovo a il p che .L. e poi ntruova e rechomincia a dare . a . al . i . adunque adire papa co*n* questa regola sia t4li).

Furthermore, the use of null letters is possible in order to complicate the cipher, although they must be added arbitrarily:

> “p*er* ischauvare il numero delle lettere vo messo . s. false *che*sono nel secondo merso a queste non rompano l’ordine ne se chontano chome s’elle no*n* vi fussino […] tu vuoi scrivere papa chomincia a una falsa . d. e poi per p. ti viene t. e poi una falsa . x . e poi seguente alla a. un 4. e poi Li e dirai papa. dt4xli ec*ho*si si *con*ti semp*er*e”[13]

Thus, following the cipher and its example, two letters are inserted: one before the word ‘*papa’* (which is ‘d’) and one in the middle, which is a ‘false,’ meaning a null ‘x.’ (queste non rompano l’ordine ne se chontano chome

[12] ‘tu vuoi scrivere papa’ in the text. English Translation: *you want to write pope.*

[13] English translation: *To hide the number of letters, I put fake letters between the real letters, which are used for encryption. The fake letters are not counted and do not follow the rule, as if they did not exist. If you want to write pope (papa) you start with a false ‘d’ and then, following the rule, for ‘p’ you write ‘t’ and then a false ‘x’ and then to write ‘a’ you use the sign ‘4’ and ‘li’ (as before) and you will write pope dt4xli and this is how it must always be done*. Note: Although the rule is written, the writer seems to have made a mistake, as he states that the ‘x’ should be placed after the first letter ‘p’, but still in the same period, he states that the letter null should be placed in the middle, as he then writes.

s'elle no*n* vi fussino). Following the same principle, the word '*papa*' can be ciphered by applying the principle of the null letters; thus the result is the word 'dt4xli'.

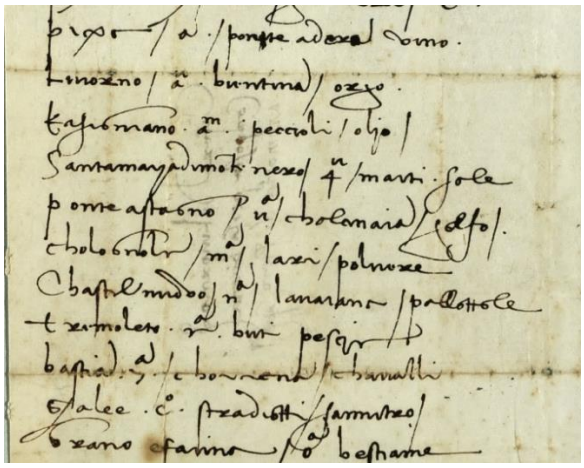


Figure 4. Nomenclator: on the left, the names to be encrypted; after the slash '/', the different possible substitutions

Finally, the cipher also includes a nomenclator to hide strategic words with substituted letters and words. It consists of thirteen words, each of which has three different possible substitutions, some of which concern places such as Livorno, Santamaria di Montenero in Liguria or Tremoleto in Tuscany.[14] Other elements of the nomenclator have to do with grain, provisions and galleys. The purpose of the second section of the cipher was twofold: to improve security by using a second encryption system, and to speed up and better conceal references to strategic information, such as locations and supplies.

4 Cipher utility and analysis

The cipher presented here is particularly effective for its time because, with the exception of Alberti's cipher discs, no other ciphers modified the cipher alphabet. The presence of a late medieval Florentine 'polyalphabetic' cipher is certainly an uncommon feature compared to other medieval Italian ciphers used on the peninsula and beyond. Furthermore, similarities with Alberti's work can be found in the substitution mechanism. Alberti, in explaining the functioning of his cipher discs, explains how one must change the alphabet:

> Cum autem tres quottuorve dictiones exscripsero mutabo nostra in formula situm indicis versione circuli[15].

The Florentine cipher operates in the same way for Piero Capponi, with the difference that the alphabet must be changed after each letter, rather than after every three or four words. The reason for this is the absence of cipher discs and the fact that there was no method for using the Florentine cipher outside the established rule. If this rule had not been followed, each letter in the cipher would no longer have corresponded to the plaintext. For this reason, the writer emphasizes the necessity of applying the rule only to the letters to be ciphered and using the nulls without following the rule, in order to avoid errors.

In fact, the error would have been irreparable: if one had made a mistake in writing a letter in cipher, the entire system would have shifted, changing the sign corresponding to each letter in the text and making it extremely difficult to decipher[16].

There was also the problem that if the rule for ciphering fell into the wrong hands, the entire cipher would become useless, a risk common to all existing ciphers.

The Florentine cipher, following the polyalphabetic system, decreases the frequency since, as seen for the word '*papa*' formed by only two repeated letters, it has formed four different signs to write in cipher, as if they were homophones, but with the characteristic that they can be used with several letters and not exclusively with one. The resulting frequency does not allow an intuitive association between a cipher symbol and a letter of the alphabet, as this association constantly changes. The result is a modified frequency in which the number of usable symbols matches the number of alphabetic letters, but they are used in a manner that appears confusing to one who needs to decrypt it, yet is extremely clear to the person using the cipher key.

Another defining feature of the cipher was the nomenclator, which enhanced the system by combining two ciphering methods.

[14] For the regions present in the cipher, see. (Repetti 2005).

[15] English Translation: *when I have written three or four words, I will change our formula by rotating the disk.*

[16] For example, if the character corresponding to the letter 'a' is used to write the character for 'b', the entire cipher would be incorrectly shifted by one position for each successive letter encrypted.

In this case, the nomenclator enabled the rapid insertion of keywords that, while not encrypted through the polyalphabetic system, introduced both an alternative method of encryption and a definition that, as with any nomenclator, conveyed a distinct and specific meaning.

As a result, in certain passages of text encoded with the polyalphabetic cipher, some words appeared that were not encrypted in the same way. Nevertheless, they still concealed meaning by employing the nomenclator, operating alongside the primary system.

5 Conclusions

The Florentine 'polyalphabetic' cipher is a very important example of how in the same century as Leon Battista Alberti and before Giovan Battista Della Porta, there was an example of a polyalphabetic cipher, used in diplomatic activity by a Florentine envoy closely linked to the figure of Lorenzo de' Medici.

The presence of this cipher makes it possible to analyse Italian secret diplomacy from a different critical perspective, showing how Florence tried to find better methods of ciphering that surpassed those of other Italian realities.

References

Alberti Leon Battista. 1994. *Dello Scrivere in cifra,* Galimberti, Torino.

Alberti Leon Battista. 1998. *De Componendis Cyfris,* Galimberti, Torino.

Archivio di Stato di Firenze (ASFI). *Miscellanea Repubblicana 11*. fasc.292, I.

Bullard Melissa Meriam. 1993. The Language of Diplomacy in the Renaissance. In *Lorenzo de'Medici, new perspectives: proceedings of the international conference held at Brooklyn College and the Graduate Center of the City University of New York, April 30-May 2, 1992*. Toscani B. (a cura di), San Francisco, Berna ed altri, Peter Lang, New York: 263-278.

Cerioni Lydia. 1970. *La diplomazia sforzesca nella seconda metà del Quattrocento e i suoi cifrari segreti*. Vol. I (di II), Fonti e studi del Corpus membranarum italicarum VII, Centro di Ricerca, Roma.

Della Porta Giovan Battista. 1563. *De furtivis literarum notis vulgo de ziferis libri IIII*. Vol. I-IV, Napoli.

Fubini Riccardo. 1994. *Italia quattrocentesca. Politica e diplomazia nell'età di Lorenzo il Magnifico*. Franco Angeli, Milano.

Lang Heinrich. 2014. Power in Letters. Political Communication and Writing in the Medici Letters. In *Medien der Macht und des Entscheidens. Schrift und Druck im politischen Raum der europäischen Vormoderne (14.-17. Jahrhundert)*, Wehrhahn, Hannover: 83-102.

Mallett Michael. 1976. Capponi, Piero. In *Dizionario Biografico degli Italiani (DBI)*, 19. Url: https://www.treccani.it/enciclopedia/piero-capponi_(Dizionario-Biografico)/ (Link active to the 30/01/2025).

Meister Aloys. 1906. *Die Geheimschrift im Dienste der päpstlichen Kurie*. Druck und Verlag von Ferdinand Schöningh, Paderborn.

Queller Donald Edward. 1967. *The Office of Ambassador in the Middle Ages*. Princeton University Press, New Jersey.

Repetti Emanuele. 2005. *Dizionario geografico, fisico, storico della Toscana*. Firenzelibri, vol. 5, Reggello.

Sacco Luigi. 1947. *Manuale di Crittografia*. Ist. Poligr. Dello Stato, Roma.

Santini Emilio. 1922. *Firenze ei suoi "Oratori" nel Quattrocento*. R. Sandron, Milano.

Zanotti Mario. 1928. *Crittografia, le scritture segrete*. Hoepli, Milano.

Appendix A Miscellanea Repubblicana 11 fasc.292 Cifra I Text.

Legend: [ ] The square brackets identify additions not present in the original text, provided for a clearer understanding of the content.

[…] Square brackets with ellipsis indicate a portion of the text that is illegible due to a tear in the paper, resulting in the loss of the written content.

"." The dots between the letters, as well as those evidently placed by the writer, have been preserved to maintain consistency with the original text.

Italics indicate abbreviations used by the writer of the 15th century document.

-Recto

[…] f g h I k L m n o p q r s t u
[…] q r h p s L I b 7 t 4 Ȳ e z 8
[…] g.
Con q[torn, perhaps questo] […] enna resta e pigliasi i choxi che la prima lettera cum*que* a pro primo poi si ch*on*ta[17] insino alla lettera che tu vuoi trovare e quella relieva quella lettera. Verbi *gratia*[18]*,* tu vuoi scrivere papa. Chomincia alla prima lettera ad *esempio* a. b. c. *etc*. e troverrà che al p viene un t e poi rechomincia alla . a . che [è] un 4. e poi rechomincia alla . a. un'altra volta de la Ȳ e vo ntruovo a il p che [è] .L. e poi ntruova e rechomincia a dare . a . al . i . adunque adire papa co*n* questa regola sia t4li e p*er* ischauvare il numero delle lettere vo messo . s. false *che*sono nel secondo merso a queste non[19] rompano l'ordine ne se chontano chome s'elle no*n* vi fussino, verbi gratia, tu vuoi scrivere papa chomincia a una falsa . d. e poi per p. ti viene t. e poi una falsa . x . e poi seguente alla a. un 4. e poi Li e dirai papa. dt4xli ec*ho*si si *con*ti semp*er*e.
Ce lego ogni chosa a . *et*. lo ze certi annota che cierti[20] no*n* rompano l'ordine chome se dice alle false.
[pixe][21] danare 1/r | suchero | e^{a}
Pixe[22] / a/ ponete adara vino
Livorno / a^{n} . bientina / orzo
Kasigriano a^{m} . peccioli / olio /
Santamarhia di Mo*n*tenero / 4^{n} / marti, sale
Ponte astagno / u^{a} / cholonaia / zolfo /
Cholognole / m^{a} / lari / polvere
Chastel Nuovo / n^{a} / Ianaiane / pallottole
Tremolete[23] r^{a} . buti peseri
Bastia 7^{a} chorrende chavalli
Zalee . c^{o} . stradiotti / Sarnitro[24]
Grano e farina / o^{a} bestiame
Vettovaglie / vento . B^{a} .

-Verso
Cifera di Livorno[25]
Composta per la buona aromoria di Piero Cappo^{ni}
Ciffra[26]

[17] 'ta' overwrite su sich*on*ta.
[18] As *exempli gratia,* that means for example.
[19] Added on line spacing.
[20] Uncertain transcription.
[21] deleted.
[22] Pisa.
[23] Tremoleto, (Repetti 2005).
[24] Salnitro, *saltpetre.*
[25] followed by an interruption sign.
[26] written vertically.

Appendix B. Cipher key recto-verso

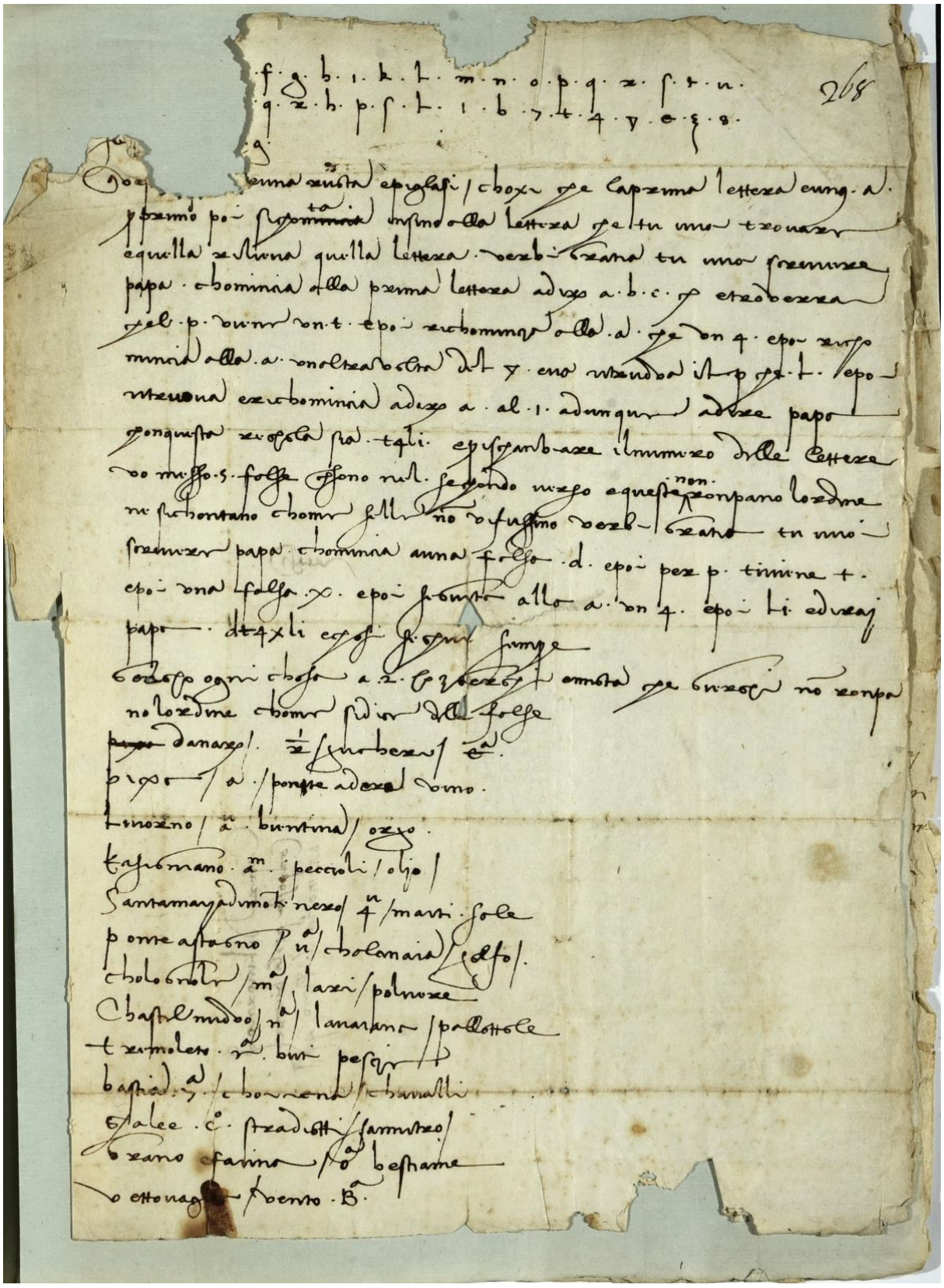


Figure 5. Polyalphabetic cipher of Piero Capponi, recto

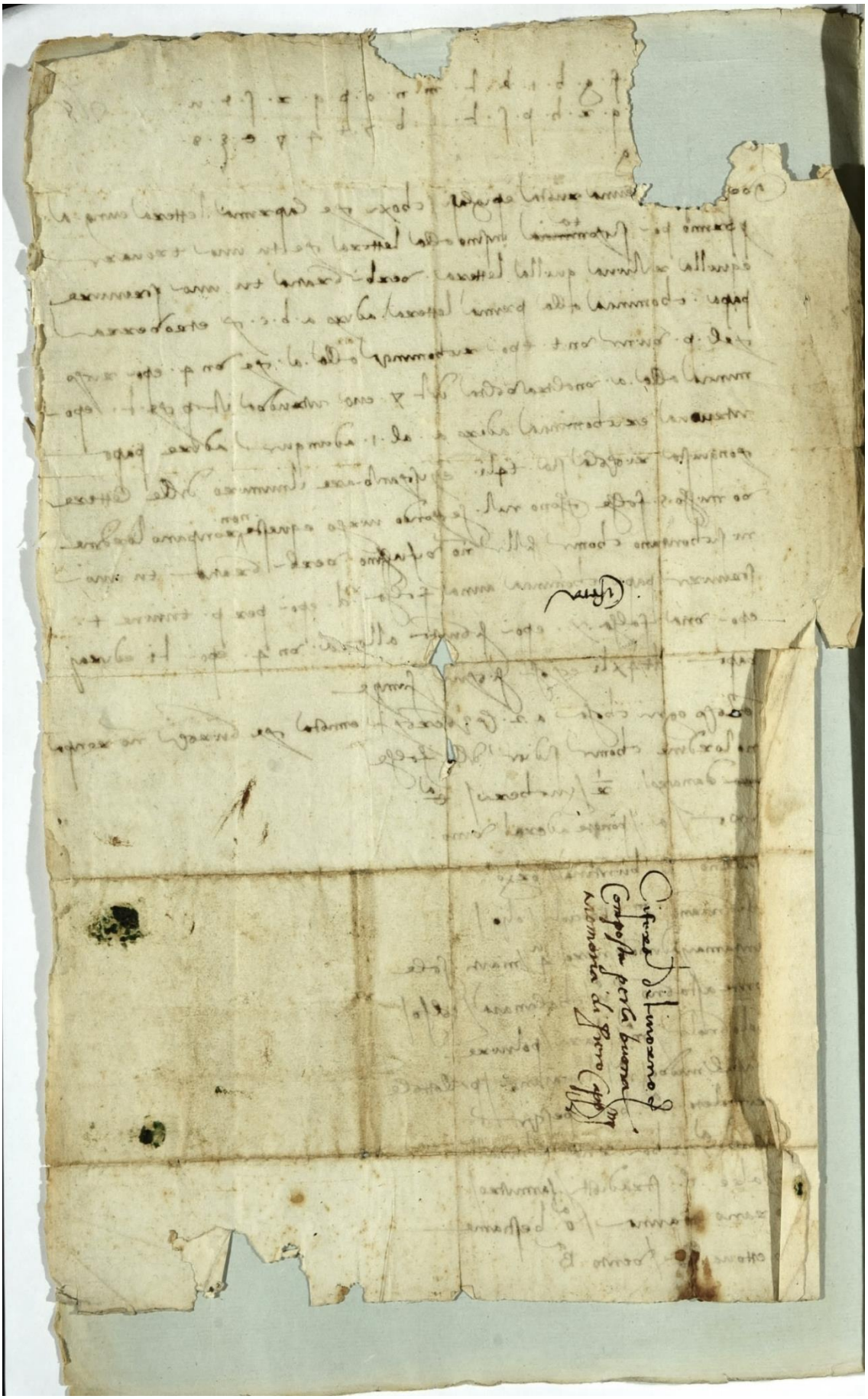


Figure 6. Polyalphabetic cipher of Piero Capponi, verso

Solving Anagrams with Integer Linear Programming

Pavol Zajac **Tomáš Selep** **Eugen Antal**
Slovak University of Technology in Bratislava
Ilkovičova 3, 841 04 Bratislava
Slovakia
pavol.zajac@stuba.sk xselep@stuba.sk eugen.antal@stuba.sk

Abstract

Given some sequence of letters, an anagram is formed by changing their order to create a different text. In a historical context, anagrams were popular mainly as puzzles, but they are also connected to classical transposition ciphers. To solve an anagram means to rearrange the letter sequence to a form that is acceptable as a word or sentence in some language.

In this article, we formalize the anagram solving problem. We focus on anagrams based on a simplified language model based on fixed dictionaries. We study the applicability of known methods for this problem. We propose a method of anagram solving based on integer linear programming. The new method is not strictly superior to existing methods but provides new tools to tackle the problem. The new representation shows potential for integration with Word2Vec representation of words for finding potentially meaningful anagrams in natural languages.

1 Introduction

One of the main categories of historical ciphers is transpositions, the basic principle of which consists in the rearrangement of letters. Transposition leaves the characters of the plain text invariant (Bauer, 2002). These ciphers are closely related to the term anagram — in a traditional sense of rearranging one (meaningful) text to spell another (meaningful) one (Kahn, 1996). Anagrams have a rich history. They were popular in the past, used in puzzles by amateurs, but also used by scientists to establish a priority of inventions (Bauer, 2002). Anagrams were used in the old inscriptions, and many of them are still unsolved, such as the mysterious anagram from the church of the Poor Clares in Bratislava (Antal and Zajac, 2024).

Anagrams are also related to the cryptanalysis of transposition ciphers: the anagram problem is a generalization of the problem of reconstructing the plain text from the letters of the cipher text (Bauer, 2002). However, it has to be noted that from a specific group of letters, multiple meaningful messages can be constructed (Bauer, 2002). On the other hand, if two or more cipher texts of the same length (encrypted with the same transposition using the same key) are available, the *multiple anagramming* method (Gaines, 1956; Barker, 1994; Kahn, 1996; Dunin and Schmeh, 2020) can be used to solve[1] the cipher texts. This method is considered as a general solution for all transposition ciphers (Kahn, 1996). It can also be used to solve complex transposition ciphers such as double or triple columnar transposition ciphers, for which either there are no efficient methods of solving (Antal et al., 2020), or they only work with some limitations (Barker, 1994; Lasry et al., 2014).

Anagram solving is essentially a process that tries to reconstruct meaningful sentences out of the input letters. Note that all rearrangements of a set of words are anagrams of each other. Thus, a sufficient goal in anagram solving is to arrange the input letters into words. For a given input set of letters, there are typically many possible anagram solutions. For example:

Input: `aadehooortuwyy`

Output:

- `who,are,you,today`
- `the,door,you,away`
- `you,heat,door,way`
- `how,are,you,today`

[1] Solving does not necessarily mean obtaining the original cipher key of the transposition cipher, rather obtaining the plain text and the global permutation applied to the whole plain text.

Proceedings of the 8th International Conference on Historical Cryptology, pages 149-154
Poznań, Poland, 16-18 June, 2025

- `you,year,wood,hat`
- `way,outdoor,yeah`
- ...

Various methods of anagram solving were studied in the past, from testing simple rearrangements of letters, to more advanced combinatorial and dictionary based methods (such as presented by Knuth (2011)). Modern approaches, such as Nishino et al. (2019) proposed, try to incorporate the methods from artificial intelligence and large language models to generate anagrams that have a meaning in some specified language. In this article, we try to formalize anagram solving (over some specific dictionary) as a difficult computational problem. Furthermore, we provide a new mathematical formalization of the problem that presents the anagram solving as a specific instance of the integer linear programming (ILP) problem. We experimentally explore whether this representation is suitable for anagram solving and provide some new ideas on how this new representation can be extended to account for solving anagrams in natural language using modern large language models.

2 Preliminaries

Let A be an alphabet. We will call $a \in A$ a letter, and a sequence of letters $s \in A^*$ a string. Let D be a finite set of strings. We will call D a dictionary. Each $w \in D$ will be called a word. A finite sequence of words $s \in D^*$ will be called a sentence. Each sentence can also be considered a string (of letters), but some strings are not sentences. Note that some strings can represent multiple sequences, depending on the dictionary D, because without separators, we can not guarantee the unique parsing of strings into words.

A language that consists of all strings over A will be denoted by $\mathscr{A} = A^*$. Language that contains all sentences (concatenation of strings from dictionary D) will be denoted by $\mathscr{D} = D^*$. A language that is a proper subset of $\mathscr{D}$ will be denoted by $\mathscr{L} \subset \mathscr{D}$, $\mathscr{L} \neq \mathscr{D}$. In practice, we will suppose that $D \subset \mathscr{L}$, that is, every word in the dictionary D taken separately belongs to the language $\mathscr{L}$, but the techniques can be adapted for any subset of $\mathscr{D}$.

Let $s \in \mathscr{A}$ be a string of length n, and let $\{\{s\}\}$ denote a multiset of letters of string s. We say that s_1 is an anagram of s_2 if and only if $\{\{s_1\}\} = \{\{s_2\}\}$. In other words, s_2 contains all letters from s_1 in any (different) order. Let n_i denote the frequency of letter i in $\{\{s\}\}$, and $|s| = n = \sum_{i \in A} n_i$. Then there exists

$$C(s) = \frac{n!}{\prod_{i \in A}(n_i!)}$$

different anagrams of s in $\mathscr{A}$. As $\mathscr{L}$ is a (proper) subset of $\mathscr{A}$, we can naturally ask how many anagrams of s are in $\mathscr{L}$. This question is difficult in general, and the decision version of this problem is NP-complete (it can be formulated as a subset sum problem, which is NP complete (Karp, 2009)).

3 Anagram solving

Under *solving an anagram* for the given string s, we mean rearranging the letters of s to form a string s' that belongs to language $\mathscr{L}$. For natural languages, this is a difficult task due to complex grammar rules, thus, we have to simplify this process slightly in computer assisted anagram solving.

Let D be a dictionary of individual words of the language $\mathscr{L}$, and let $\mathscr{D} = D^* \supset \mathscr{L}$. We can solve anagrams in $\mathscr{L}$ with a two-step process:

1. Enumerate anagrams of s that belongs to (dictionary based) language $\mathscr{D}$. That is, we only construct a concatenation of words from dictionary D, ignoring grammar rules of $\mathscr{L}$.
2. Filter potential candidates (anagrams in $\mathscr{D}$) by checking the grammar rules of $\mathscr{L}$.

An alternative approach is to combine these two steps by using statistical hints from a language model, such as the method presented by Nishino et al. (2019). However, in this article, we only want to focus on step 1 in the process, the construction of anagrams in $\mathscr{D}$.

Furthermore, observe that language $\mathscr{D}$ contains all possible sequences of words from dictionary D, regardless of their order. Therefore, if sentence $w_1|w_2|\cdots w_n \in \mathscr{D}$ is an anagram of s, any sentence that is a permutation of these words will also be an anagram of s. Thus, for the given input string s, the anagram solving method based on dictionary D will output a multiset of dictionary words $W = \{\{w_1, w_2, \ldots, w_n\}\}$, $w_i \in D$, such that $\{\{s\}\} = \{\{w_1|w_2|\ldots|w_n\}\}$.

3.1 Anagram solving based on letter permutations

A naïve method of anagram solving involves a generate and test approach: for the given input

string s, create all letter permutations, and test whether the result can be parsed into dictionary words.

The complexity of this method is given by $C(s) \cdot T_1$, where $C(s)$ is the number of letter-anagrams of s (or in a simple implementation by $n!$, where $n = |s|$). Time T_1 is the time required to check whether the letter sequence can be parsed into dictionary words[2].

We expect that this method can be useful in scenarios where the dictionary size is large and the input size is small.

3.2 Anagram solving based on dictionary search

Let D be a dictionary of size d. Given input s, we can compute the multiset $S = \{\{s\}\}$. For each word in dictionary $w_i \in D$, we can also compute multiset $W_i = \{\{w_i\}\}$. To find anagrams in $\mathscr{D}$, we can use the following steps:

1. Filter dictionary: remove words from D, that cannot be formed from letters of s.
2. Search: for each remaining word w_i in D, remove letters of w_i from s, and recursively try to construct an anagram of the remaining letters.

Note that our goal is to enumerate all multisets of words that are anagrams to s. To avoid different word permutations, we need to use a sorted dictionary (in any order). In the recursive step, we only search for sub-anagrams in a dictionary that consists of word w_i, and words higher in the dictionary order (because words lower in the order were already explored).

During the algorithm, we need to check whether some dictionary word w can be a sub-anagram of string s. This means that we need to check if $\{\{w\}\} \subset \{\{s\}\}$. This multiset comparison can be done using vectors with letter frequencies, which requires $|A|$ comparisons. A well-known optimization[3] is to associate the letters of the alphabet with primes. Strings over the alphabet are then associated with products of the letter primes (with multiplicities), as their numerical fingerprints. That is, we use some function $\pi : A^* \to \mathbb{Z}$, with $\pi(x) = p_x$ with $x \in A$, and p_x a prime number associated with x. For longer strings, $\pi(x_1|x_2|\ldots|x_n) = \prod p_{x_i}$. It is easy to see that $\{\{w\}\} \subset \{\{s\}\}$ holds if and only if $\pi(s) = 0 \mod \pi(w)$ (numerical fingerprint of w is a divisor of the fingerprint of s).

The complexity of the method depends on the size of the dictionary d, on the contents of the dictionary, and on the length of the anagram n. If the average word length in the dictionary is l, we expect that we need n/l words to form an anagram of s. We thus have to search through the space of (approximately) $d^{n/l}$ strings.

4 Integer linear programming representation of the anagram solving

When considering a dictionary based anagram solver, we can notice that the only criteria for some string being a (sub-)anagram of another string are the involved letter frequencies. In this section, we present our new approach, similar to the approach used in algebraic cryptanalysis: we represent the problem in algebraic way and use specialized tools to solve the problem in its algebraic form. Our chosen representation is in the form of an integer linear programming instance.

Let $m = |A|$, and let $d = |D|$. We will represent dictionary word $w_i \in D$ with m-dimensional vector $\mathbf{w}_i \in \mathbb{Z}^m$, with $w_{i,j}$ coordinate equal to multiplicity of letter j in word w_i. Let $x_i \in \mathbb{Z}$ denote multiplicity of word w_i in multiset W, and let $y_j \in \mathbb{Z}$ denote multiplicity of letter j in $\{\{s\}\}$. Then it must hold that $y_j = \sum_{i=1}^{d} x_i w_{i,j}$, for each $j = 1, 2, \ldots, m$. Thus, we can write a system of linear equations

$$\left(\mathbf{w}_1^T \mathbf{w}_2^T \cdots \mathbf{w}_d^T\right) \mathbf{x}^T = \mathbf{y}^T,$$

with d unknowns x_i, and m equations. We are looking for an integer solution ($x_i \in \mathbb{Z}$) of this linear equation system with $x_i \geq 0$. This is known as an integer linear programming problem, ILP in short. Any valid solution $\mathbf{x}$ of the ILP problem represents a single multiset $W = \{\{w_1, w_2, \ldots, w_n\}\}$ that can be used to construct anagrams of s that are in $\mathscr{D}$.

Note that some words w_i in the dictionary might be an anagram of another word w_k. In this case $\mathbf{w}_i = \mathbf{w}_k$. In practice, we can represent both words with a single vector, and construct multiple anagram solutions by substituting $\mathbf{w}_i$ with each anagram word from the original dictionary.

[2] This is known as a Word-break problem, and has complexity $O(n^2)$ (Cormen et al., 2009).

[3] This method might be very old, but it is not clear to us who invented the method. We can refer to `https://stackoverflow.com/questions/13215789/comparing-anagrams-using-prime-numbers`, where the method is discussed.

Note that in terms of complexity, the linear programming approach is, in practical terms, equivalent to a dictionary search. This is because typically there is a significantly larger number of dictionary words (unknowns in the system) than alphabet letters (equations in the system). This means that the ILP solver needs to enumerate a large number of potential solutions word by word in a similar manner to a dictionary search.

5 Comparison of the methods

We have implemented both the dictionary based method and the ILP based method to test their behavior and efficiency on (relatively) small anagrams. The ILP method implementation uses ILP solver Gurobi[4], version 12.0.0. We focused on finding any feasible solutions with a systematic search, without improving the search with heuristics, and optimization through an objective function. The full settings of the solver are summarized in Table 1.

We have tested the methods on a specific dataset of n input sentences of increasing length k. The original input sentences were picked from the OANC corpus[5]. Thus, the sentences can be considered meaningful in the English language. All words in these sentences were present in the custom dictionary[6] of 89000 words.

For anagram solving, we reduced the dictionary size to a set of 10000 most frequent words with a length of at least 3. The dictionary reduction is necessary for performance reasons, and moreover it can simulate the case where we do not always have access to the full original dictionary. The similar reasoning holds for using short and special words, such as 'a', 'an', 'the': they have a detrimental effect on anagram solving, and can be preprocessed separately (guessed and removed from the input sequence).

During the experiment, we took the input sentence and ordered the letters in this sentence alphabetically. Both solvers (dictionary based, and ILP based) tried to find anagrams of this scrambled input.

A specific input sequence can produce multiple anagrams valid in the language given by the reduced test dictionary (see Figure 1) and can fail to produce the original sentence from the larger input dictionary. In the test settings, the probability of finding the original sentence was approximately 81%. To objectivize the comparison of methods, the efficiency of anagram solving was measured by the time required to find the original input sentence as an anagram of the input. The results of the experiments are summarized in Figure 2.

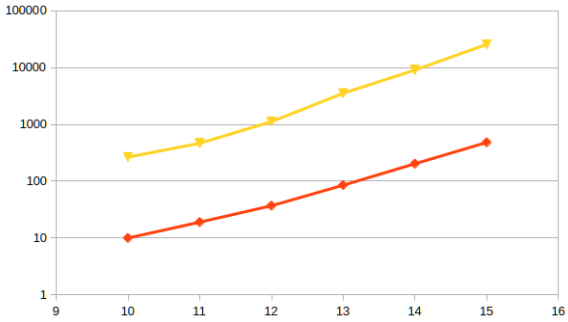


Figure 1: Number of anagrams obtained from an input of length k (x-axis). The bottom line represents the median value, and the top line the maximum value attained (dataset size 10000 inputs).

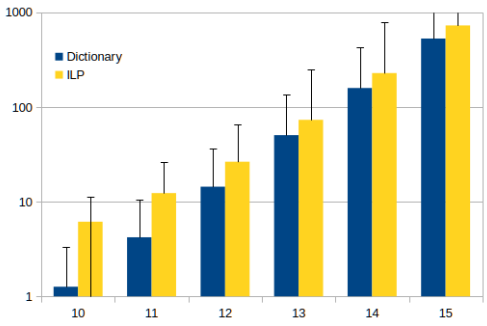


Figure 2: Average runtime of dictionary based vs. ILP based method (as a function of input of length k). Error bars are based on standard deviation.

As expected, we can see that the complexity of the anagram search increases exponentially with the length of the input k. The effect of the search method contributes only to a constant term. The dictionary based method is faster, but its advantage seems to be decreasing with increased input size k (see Figure 3). This is mainly a contribution of the more complex initialization of the ILP method. We have not tested potential advantages that could be obtained by ILP heuristics, and whether they can outperform dictionary based methods (on average).

[4] https://www.gurobi.com/downloads/gurobi-software/

[5] https://anc.org/data/oanc/download/

[6] https://people.sc.fsu.edu/~jburkardt/datasets/words/anagram_dictionary.txt

Parameter	Value	Description
Objective fn.	None	Solver searches for any feasible solution.
Heuristics	0	Heuristics turned off.
MIPFocus	1	Focus on feasible solutions.
PoolSearchMode	2	Systematic search.
PoolSolutions	MAX_INT	(No) Limit on the number of solutions.
Others	Default	

Table 1: The settings of the ILP solver Gurobi 12.0.0 used in the experiments.

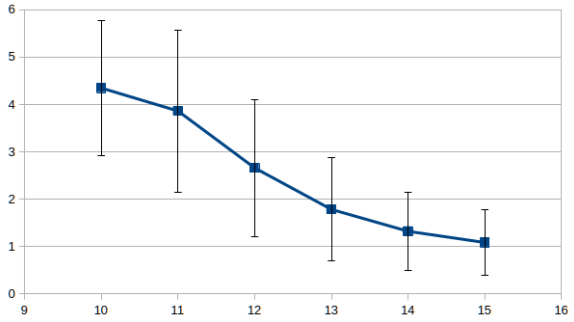


Figure 3: Average ratio of ILP solving time vs. dictionary based solving time (as a function of input of length k). Error bars are based on standard deviation. Values above 1 mean ILP method was slower by this constant factor.

6 Conclusions

In this paper, we have summarized some known methods that can be used for automated anagram solving, focusing on dictionary based anagrams. Anagram solving can be used to study old inscriptions, such as the mysterious anagram from the church of the Poor Clares in Bratislava (Antal and Zajac, 2024). Other potential applications are in the analysis of classical transposition ciphers and various transposition puzzles.

We have also introduced a new method of anagram solving that uses integer linear programming (ILP). Initial experiments show that this method is less suitable for solving short anagrams than dictionary method, due to the higher cost of ILP initialization. The ILP method compares more favorably with the dictionary based method with sufficiently long anagrams. Furthermore, we have only tested a base setting of the ILP solver, and have not used any heuristics or optimizations (that would require a specification of some objective function, and it is not clear what the function should look like).

A main difference between ILP based approach and dictionary based approach is the potential to extend the algebraic model with additional information that can be processed in the same way (using ILP solver). A potential extension can be provided by representing words with higher-dimensional Word2Vec (Mikolov et al., 2013) coefficients. A linear combination of words in a potential anagram would create a point in this vector space that can express a joint information about the whole collection of candidate words. The resulting coefficients (and/or their combinations) can then be targets of ILP constraints, and provide additional information improving the ILP efficiency. This approach, however, requires further research, as it is not clear how exactly these constraints should be constructed and used in anagram solving.

Acknowledgments

This research was supported by the project ”Artificial intelligence for encrypted handwritten document processing”, ”09I05-03-V02-00031” of call ”Support of research projects aimed at digitization of the economy in TRL levels 1-3”, Call. No. 09I05-03-V02, managed by Research Agency and funded by The Recovery and Resilience Facility of Slovak Republic.

References

Eugen Antal and Pavol Zajac. 2024. Can artificial intelligence solve the mysterious anagram from the church of the Poor Clares in Bratislava? In *Proceedings of the 7th International Conference on Historical Cryptology (HistoCrypt 2024)*.

Eugen Antal, Pavol Zajac, and Otokar Grošek. 2020. Diplomatic ciphers used by Slovak Attaché during the WW2. In *International Conference on Historical Cryptology*.

W.G. Barker. 1994. *Cryptanalysis of the Double Transposition Cipher*. A cryptographic series. Aegean Park Press.

F.L. Bauer. 2002. *Decrypted Secrets: Methods and Maxims of Cryptology*. Decrypted Secrets: Methods and Maxims of Cryptology. Springer.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2009. Introduction to algorithms (3-rd edition). *MIT Press and McGraw-Hill*.

Elonka Dunin and Klaus Schmeh. 2020. *Codebreaking: A Practical Guide*. Little, Brown Book Group.

H.F. Gaines. 1956. *Cryptanalysis: A Study of Ciphers and Their Solution*. Dover Brain Games & Puzzles. Dover Publications.

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

Richard M Karp. 2009. Reducibility among combinatorial problems. In *50 Years of Integer Programming 1958-2008: from the Early Years to the State-of-the-Art*, pages 219–241. Springer.

Donald E Knuth. 2011. *The art of computer programming, volume 4A: combinatorial algorithms, part 1*. Addison-Wesley Professional.

George Lasry, Nils Kopal, and Arno Wacker. 2014. Solving the double transposition challenge with a divide-and-conquer approach. *Cryptologia*, 38(3):197–214.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2013)*, pages 746–751.

Masaaki Nishino, Sho Takase, Tsutomu Hirao, and Masaaki Nagata. 2019. Generating natural anagrams: Towards language generation under hard combinatorial constraints. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6408–6412.

Published by

NEALT Proceedings Series 58
D-Space at Tartu University Library
ISSN: 1736-8197
eISSN: 1736- 6305
ISBN: 978-99-0853-287-5
https://doi.org/10.58009/aere-perennius0158