Cryptology Part I: Classic Ciphers (Mathematical Version)

Klaus Pommerening
Fachbereich Physik, Mathematik, Informatik
der Johannes-Gutenberg-Universität
Saarstraße 21
D-55099 Mainz

October 25, 1999—English version October 5, 2013—last change January 19, 2021

Preliminary Note

This text is somewhat incomplete. It focusses on the mathematical background of Cryptology. People without mathematical ambitions may browse the HTML pages—these are informal and hopefully self-contained. Also for historical or motivational stuff I often refer to the accompanying web pages

`http://www.staff.uni-mainz.de/pommeren/Cryptology/`

Motivational Hints

Classical cryptography considers ciphers in use up to the 1970's, that is, in the precomputer era. Today no one seriously uses these ciphers. Why does it make sense dealing with them?

- We get a feeling for the security of the basic encryption steps that are in use as components of the more complex ciphers of today.
- The complexity of modern techniques becomes perspicuous.
- Most of the mathematical foundations are relevant also for modern cryptologic techniques.
- We may learn a lot from the failures of the past—many of the commonly accepted principles of cryptology arose a long time ago. In short: *The algorithms are out-of-date, the methods and principles are up-to-date.*
- Classical cryptology makes a good part of general education, not only for mathematicians or computer scientists. In particular it provides many interesting project ideas for undergraduates or even school children.
- Classical cryptology provides intellectual challenges—better than chess, poker, or war games [:-)]. The puzzle corners of journals often contain puzzles whose cryptological background shines through.
- And last but not least: occupation with classical cryptology is fun.

Elonka Dunin's web site "Famous Unsolved Codes and Ciphers" has an overview over unsolved "historic" cryptograms:

`http://www.elonka.com/UnsolvedCodes.html`

"The Secret Code Breaker" (Bob Reynard) has a lot of elementary material that's also great for kids:

`http://www.secretcodebreaker.com/`

CrypTool also contains a lot of educational material and challenges:

`http://www.cryptool.org/en/`

CrypTool online contains lots of classic ciphers which are explained and executable in a browser or on a smartphone:

`http://www.cryptool-online.org/en/`

MysteryTwister C3, abbreviated MTC3, is a crypto cipher contest with currently more than 180 challenges created by more than 40 authors and used by more than 5000 solvers. The website has a moderated forum. The challenges are distributed in 4 different levels:

`http://www.mysterytwisterc3.org/`

Klaus Schmeh has a blog with the latest news in classic cryptology and many unsolved ciphers (German mostly):

`http://scienceblogs.de/klausis-krypto-kolumne/`

Conventions

In order to not get lost in less relevant and nasty details most examples in this chapter follow the model:

- Ciphertexts are written in uppercase letters without word boundaries, employing the 26 letter alphabet `A...Z`.
- Plaintexts are written in upper-, lower-, or mixed-case letters, with or without word boundaries and punctuation.

The mathematical considerations try to be as general as possible with respect to the used alphabet.

Gender Mainstreaming: It is common use in modern cryptology to staff the scenarios with men and women alternately. Alice and Bob are communicating partners, Eve is the eavesdropper, and Mallory, the "man in the middle". In classical cryptology the role of the cryptanalyst corresponds to the eavesdropper. For this reason in the following we consider the cryptanalyst as female in honor of the famous cryptanalysts Elizebeth FRIEDMAN, Joan CLARKE, and Mavis LEVER.

Chapter 0

Cryptology as Entertainment—Literature and Puzzles

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/0_Entertain/`

Chapter 1

Monoalphabetic Substitutions

1.1 Mathematical Model of Cryptography

We want to give a formal definition of the following two concepts:

- *An encryption function transforms arbitrary character strings into other character strings.* (Where the strings are from a given alphabet.)

- *A cipher is a parametrized family of encryption functions. The parameter is called the key.* It determines the choice of a function from the family.

The purpose of this construct is that nobody can invert the encryption function except people who know the key. That is, an encrypted message (or a text, a file ...) is kept secret from third parties. These can see that there is a message, but they cannot read the contents of the message because they don't have the key and therefore don't know which of the functions from the family to invert.

Alphabets and Texts

Let Σ be a finite set, and call it **alphabet**. Call its elements **letters** (or **symbols**, or **characters**).

Examples. Here are some alphabets of cryptographic relevance:

- {A, B, ..., Z}, the standard 26 letter alphabet of classical cryptography.

- The 95 character alphabet of printable ASCII characters from "blank" to "tilde", including punctuation marks, numbers, lowercase, and uppercase letters.

- $\{0,1\} = \mathbb{F}_2$, the alphabet of bits, or the field of two elements. The earliest appearence (after Bauer[1]) is Bacon 1605.
- $\mathbb{F}_2^5$, the alphabet used for telegraphy code since Baudot (1874). It has 32 different symbols and also goes back to Bacon (after Bauer[1]).
- $\mathbb{F}_2^8$, the alphabet of bytes (correctly: octets, because in early computers bytes did not necessarily consist of exactly 8 bits). The earliest appearance seems to be at IBM around 1964.
- More generally $\mathbb{F}_2^l$, the alphabet of l-bit blocks. Often $l = 64$ (for example in DES or IDEA), or $l = 128$ (for example in AES). See Part II (on Bitblock Ciphers).

Often the alphabet Σ is equipped with a group structure, for example:

- $\mathcal{Z}_n$, the cyclic group of order $n = \#\Sigma$. Often we interpret the calculations in this group as arithmetic $\bmod n$, as in elementary number theory, and denote $\mathcal{Z}_n$ by $\mathbb{Z}/n\mathbb{Z}$, the residue class ring of integers $\bmod n$.
- $\mathbb{F}_2$ with the field addition $+$, as Boolean operator often denoted by XOR or $\oplus$. (Algebraists like to reserve the symbol $\oplus$ for direct sums. For this reason we'll rarely use it in the Boolean context.)
- $\mathbb{F}_2^l$ as l-dimensional vector space over $\mathbb{F}_2$ with vector addition, denoted by $+$, XOR, or $\oplus$.

For an alphabet Σ we denote by Σ^* the set of all finite sequences from Σ. These sequences are called **texts** (over Σ). A subset $M \subseteq \Sigma^*$ is called a **language** or **plaintext space**, and the texts in M are called meaningful texts or **plaintexts**.

Note that the extreme case $M = \Sigma^*$ is not excluded.

Ciphers

Let K be a set (finite or infinite), and call its elements **keys**.

Definition (i) An **encryption function** over Σ is an injective map $f\colon \Sigma^* \longrightarrow \Sigma^*$.

(ii) A **cipher** (also called encryption system or cryptosystem) over Σ with key space K is a family $F = (f_k)_{k\in K}$ of encryption functions over Σ.

(iii) Let F be a cipher over Σ, and $\tilde{F} = \{f_k | k \in K\} \subseteq \operatorname{Map}(\Sigma^*, \Sigma^*)$ be the corresponding set of different encryption functions. Then $\log_2(\#K)$ is called the **key length**, and $d(F) = \log_2(\#\tilde{F})$, the **effective key length** of the cipher F.

Remarks

1. This is not the most general definition of an encryption function. One could also consider non-injective functions, or even relations that are not functions, or are not defined on all of Σ^*.
2. Strictly speaking, the encryption functions need to be defined only on the plaintext space M, however we almost always consider encryption functions that are defined on all of Σ^*.
3. The encryption functions f_k, $k \in K$, need not be pairwise different. Therefore in general $\#\tilde{F} \leq \#K$, and effective key length $\leq$ key length. If K is infinite, then $\tilde{F}$ can be finite or infinite. In general the key length is easier to determine than the effective key length, however it is less useful.
4. The elements in the ranges $f_k(M)$ depend on the key k. They are called **ciphertexts**.
5. Note that the identification of the alphabet Σ with the integers $\bmod n$, $\mathbb{Z}/n\mathbb{Z}$, also defines a linear order on Σ. We often implicitely use this order. In some cases for clarity we must make it explicit.

1.2 Shift Ciphers

Assume that the alphabet is linearly ordered. A shift cipher replaces each letter of the plaintext by the letter that follows a certain number k of positions in the alphabet. If the end of the alphabet is reached, restart at the beginning. That means, we consider cyclic shifts. The number k is the key.

Decryption works in the reverse direction: Count backwards from the ciphertext letter.

Example 1: Original Caesar

Here $\Sigma = \{A,...,Z\} = \mathcal{Z}_{26}$, hence $A \leftrightarrow 0, B \leftrightarrow 1, ..., Z \leftrightarrow 25$. Caesar used the fixed key $k = 3$. Encryption looks like follows

```
C A E S A R  | +3  (plaintext)
-----------
F D H V D U        (ciphertext)
```

Note that the original Roman alphabet had only 23 letters without J, U, W. However in this part of the lecture we (almost) always use the 26 letter alphabet.

As key space we could also take $K = \mathbb{Z}$. Then the key length is ∞. But effectively we only have 26 different encryption functions, one of them being trivial. Therefore the effective key length is only $\log_2(26) \approx 4.7$.

Example 2: ROT13

ROT13 is a shift cipher over the alphabet {A, ..., Z} that shifts each letter by 13 positions ahead in the alphabet. As mnemonic take the table

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

As encryption function this is almost useless. Its purpose is hiding some texts, say of offensive content, from immediate recognition. The reader of the message can figure it out only by a conscious act.

Because $13 + 13 = 26$, double encryption restores the plaintext. That is, ROT13 is an involution. Or in other words: encryption = decryption as functions.

Example 3: XOR

This example extends the notion of shift cipher towards the more general version given in the mathematical description below. In this sense XOR is a shift cipher on the space of l-bit blocks. Thus our alphabet is the l-dimensional vector space $\mathbb{F}_2^l$ over the two element field $\mathbb{F}_2$. The operation XOR is the addition of vectors in this space (because XOR of bits is the addition in the field $\mathbb{F}_2$). The key is a fixed block k. Each plaintext block a is XORed with k bitwise, that is, "shifted" (or translated) by k.

Mathematical Description

Let the alphabet Σ be a finite group G with n elements and with group composition $*$. As key space also take $K = G$. For $k \in K$ let

$$f_k : \Sigma^* \longrightarrow \Sigma^*$$

be the continuation of the right translation $f_k(s) = s * k$ for $s \in \Sigma$, that is

$$f_k(a_1, \ldots, a_r) = (a_1 * k, \ldots, a_r * k) \quad \text{for } a = (a_1, \ldots, a_r) \in \Sigma^r.$$

The effective key length is $d(F) = \log_2(n)$. Thus the key space is quite small and is easily completely searched except when n is VERY LARGE. An example will follow in the next section.

1.3 Cryptanalysis of Shift Ciphers by Exhaustion

General Approach

The most primitive of all cryptanalytic attacks is **exhaustion**, also known as **brute force attack**. It consists of a complete key search—run through the complete key space K, and try key after key until you get a valid decryption.

Assume that K is finite (as it is in all practical situations). Then the attacker needs $\#K$ steps in the worst case, and $\#K/2$ steps in the mean. *This method applies to all ciphers.* A precondition for the success is the redundancy of the plaintext language that allows distinguishing between meaningful text and nonsense character sequences. In general the solution is unique as soon as the length of the text exceeds the "unicity distance" of the cipher, see Chapter 10.

For distinguishing between meaningful and meaningless texts, algorithms that compute language statistics may be used, see Chapter 3.

Solving Shift Ciphers

```
FDHVDU
GEIWEV
HFJXFW
IGKYGX
JHLZHY
KIMAIZ
LJNBJA
MKOCKB
NLPDLC
OMQEMD
PNRFNE
QOSGOF
RPTHPG
SQUIQH
TRVJRI
USWKSJ
VTXLTK
WUYMUL
XVZNVM
YWAOWN
ZXBPXO
AYCQYP
BZDRZQ
CAESAR
DBFTBS
ECGUCT
```

This is an example for solving a shift cipher by exhaustion. The first row is the ciphertext from the last section. The following rows contain the candidate plaintexts for each possible key one after the other.

Only the row `CAESAR` makes sense as plaintext. Hence the ciphertext is decrypted and the key is 3.

Note that each column contains the standard alphabet, cyclically continued. From this observation a purely mechanical approach derives: Produce some vertical strips containing the alphabet twice, and arrange them beneath each other in such a way that one row contains the ciphertext. Then scan the other rows for meaningful plaintext.

> Because of this scheme the exhaustion method is sometimes called "generatrix method". This notation comes from an analogy with cipher cylinders, see Chapter 4.

Lessons Learned

1. Shift ciphers are solvable as soon as the attacker has some small amount of ciphertext, at least when the alphabet is not too large and the language is only a small part of all character sequences. (Later we'll express this as "high redundancy" or "low entropy", see Chapter 10.)
2. A cipher should use a large key space (or rather a large effective key length). But bear in mind:

   The effective key length measures the complexity of the exhaustion attack. But in general it is an insufficient measure of the complexity of the cryptanalysis of a cipher.

In other words: In many cases there are more efficient attacks against a cipher than exhaustion.

1.4 Monoalphabetic Substitution

Introductory Example

The key of a monoalphabetic substition is a permutation of the alphabet, for example:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
UNIVERSTABCDFGHJKLMOPQWXYZ
```

For encryption locate each letter of the plaintext in the first row of this table, and replace it by the letter below it. In our example this becomes:

```
ENGLI SHAST RONOM ERWIL LIAML ASSEL LDISC OVERE DTRIT ON
EGSDA MTUMO LHGHF ELWAD DAUFD UMMED DVAMI HQELE VOLAO HG
```

For decryption we use the inverse permutation, given by the table

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
IJKLEMNOCPQRSBTUVFGHADWXYZ
```

Mathematical Description

Let $\mathcal{S}(\Sigma)$ be the group of permutations of the alphabet Σ, that is the full symmetric group. See Appendix A for an introduction to permutations.

A monoalphabetic substitution consists of the elementwise application of a permutation $\sigma \in \mathcal{S}(\Sigma)$ to texts:

$$f_\sigma(a_1, \ldots, a_r) := (\sigma a_1, \ldots, \sigma a_r) \quad \text{for } (a_1, \ldots, a_r) \in \Sigma^r.$$

Definition A **monoalphabetic cipher** over the alphabet Σ with keyspace $K \subseteq \mathcal{S}(\Sigma)$ is a family $(f_\sigma)_{\sigma \in K}$ of monoalphabetic substitutions.

Examples 1. The shift cipher where $K =$ the set of right translations.

2. The general monoalphabetic cipher where $K = \mathcal{S}(\Sigma)$. Here $\#K = n!$ with $n = \#\Sigma$.

The Effective Key Length

The general monoalphabetic cipher F defeats the exhaustion attack, even with computer help. The $n!$ different keys define $n!$ different encryption functions. Therefore

$$d(F) = \log_2(n!) \geq n \cdot [\log_2(n) - \log_2(e)] \approx n \cdot \log_2(n)$$

by STIRLING's formula, see Appendix B. For $n = 26$ we have for example

$$n! \approx 4 \cdot 10^{26}, \quad d(F) \approx \log_2(26!) \approx 88.38.$$

Note that for a ciphertext that doesn't contain all letters of the alphabet the search is somewhat faster because the attacker doesn't need to determine the entire key.

1.5 Algorithms and Programming in Perl

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/1_Monoalph/MonoPerl.html`

1.6 Cryptanalysis of Monoalphabetic Substitution

General Approach

The cryptanalysis of the monoalphabetic substitution makes use of its *invariants*, that is of properties of a text that remain unchanged under encryption:

1. The distribution of the frequencies of single characters is invariant.

    - This means that a letter in the ciphertext occurs exactly as many times as the corresponding letter in the plaintext.
    - The same is true for bigrams (= pairs of letters), trigrams, ..., n-grams.

2. Repeated patterns in the plaintext show up also in the ciphertext.

Both of these invariant properties suggest cryptanalytic approaches:

1. Statistical analysis
2. Pattern recognition (for example matching with the words of a dictionary)

Often the cryptanalyst combines both of these approaches, and supplements them with systematic guesses:

- *Cryptography is Mathematics.*
- *Cryptanalysis is struggling, using all available aids.*

Only in rare situations cryptanalysis is completely algorithmic. But no matter which method applies and how clean its theoretical basis is, the successful solution legitimates the cryptanalyst.

1.7 Statistical Analysis of Ciphertext

Character Frequencies

Natural languages such as German, English, Russian, ..., and also artificial languages such as MS-DOS-EXE, ..., Pascal, ..., MS-Word, ..., show typical character frequencies that are

- nonuniformly distributed,
- characteristic for the language.

Texts of about 500 or 1000 letters in a natural language rareley show a significant deviation from the typical frequencies.

This allows automating the cryptanalysis based on letter frequencies to a large extent. The web offers several such programs, for example see the ACA Crypto Dropbox [`http://www.und.nodak.edu/org/crypto/crypto/`].

Mathematical Model

The simplest mathematical model for statistical analysis of ciphertext is a probability distribution on the underlying (finite) alphabet Σ with atomic probabilities $p(s)$ for all letters $s \in \Sigma$. Thus we assume that plaintexts are streams of independent (but not uniformly distributed) random letters.

A closer approximation to the truth would account for dependencies of letters from their predecessors according to the typical bigram distribution.

There are further possible refinements, for example the most frequent initial letter of a word in English is T, in German, D.

Example: Byte Frequencies in MS-Word Files

Byte	**Frequency**
`00`	ca 7-70%
`01`	ca 0.8-17%
`20` = space	ca 0.8-12%
`65` = e	ca 1-10%
`FF`	ca 1-10%

Observations

- The variability is rather large, unexpected peaks occur frequently.
- The distribution depends on the software version.
- All bytes `00`-`FF` occur.
- We see long sequences of zero bytes. If the file is encrypted by XOR, large parts of the key shine through.

The last remark yields an efficient method for analysis of the XOR encryption of a WORD file with periodically repeated key. This not exactly a statistical cryptanalysis, it only uses the frequency of a single byte. To start with, pairwise add the blocks. If one of the plaintext blocks essentially consists of zeroes, then the sum is readable plaintext:

$$\begin{array}{lcccccc} \textbf{Plaintext} & \ldots & a_1 \ldots a_s & \ldots & 0 \ldots 0 & \ldots \\ \textbf{Key}\text{ (repeated)} & \ldots & k_1 \ldots k_s & \ldots & k_1 \ldots k_s & \ldots \\ \textbf{Ciphertext} & \ldots & c_1 \ldots c_s & \ldots & c'_1 \ldots c'_s & \ldots \end{array}$$

where $c_i = a_i + k_i$ in the first block, and $c'_i = 0 + k_i$ in the second block for $i = 1, ..., s$ (s the blocksize).

Therefore $c_i + c'_i = a_i + k_i + k_i = a_i$,—one block of plaintext revealed and identified—; and $k_i = c'_i$—the key revealed.

If the addition of two cipher text blocks yields a zero block, then with high probability both plaintext blocks are zero blocks (or with small probability are identical nonzero blocks). Also in this case the key is revealed.

1.8 Example of a Statistical Cryptanalysis

See web pages `http://www.staff.uni-mainz.de/pommeren/Kryptologie /Klassisch/1_Monoalph/Beispiel.html` (in German) or `http://www.staff.uni-mainz.de/pommeren/Kryptologie/Klassisch /0_Unterhaltung/Lit/Goldbug_Crypto.html` (in English)

1.9 Pattern Search

Word Lists

The second basic approach to cryptanalysis of the monoalphabetic substitution is the search for patterns in the ciphertext that correspond to the patterns of

- supposed words (probable words),
- words from a list.

This method is cumbersome if done by hand but easy with computer support that completely searches lists of several 100000 words in a few seconds.

Searching for a probable word is a variant of pattern search. We search for the pattern of a word that we suspect from knowledge of the context as occuring in the plaintext.

Numerical Patterns for Strings

To normalize letter patterns we describe them by numbers. Here is an example: The word "`statistics`" defines the pattern `1232412451`. The general procedure is: Replace the first letter by 1. Then replace each following letter by

- the number that was assigned to this letter before,
- the next unused number, if the letter occurs for the first time.

Here is a formal definition:

Definition Let Σ be an alphabet. Let $a_1, \ldots, a_q$ be letters from Σ. The **pattern** belonging to the string $(a_1, \ldots, a_q)$ ist the q-tuple $(n_1, \ldots, n_q) \in \mathbb{N}^q$ of numbers that is defined recursively by

- $n_1 := 1$.
- For $k = 2, \ldots, q$:
  If there is an i with $1 \leq i < k$ and $a_k = a_i$, then $n_k := n_i$,
  else $n_k := 1 + \max\{n_i \mid 1 \leq i < k\}$.

Remarks

1. $n_i = n_j \Longleftrightarrow a_i = a_j$ for $1 \leq i \leq j \leq q$.
2. $\{n_1, \ldots, n_q\} = [1 \ldots m]$ where $m = \#\{a_1, \ldots, a_q\}$ (= number of different letters in $(a_1, \ldots, a_q)$).

Algorithmic Description

Goal: Determine the numerical pattern of a string.

Input: The string as a list `string =` $(a_1, \ldots, a_q)$.

Output: The numerical pattern as a list `pattern =` $(n_1, \ldots, n_q)$.

Initial value: `pattern = empty list`.

Auxiliary variables:

- n = current number, initial value = 0.
- `assoc =` list of processed letters.
  The index i belongs to the letter `assoc[i]`.
  Initial value: `assoc = empty list`.

Procedure: Loop over the letters in `string`. The current letter is `x`.

```
If there is an $i$ with x = assoc[i], then append $i$ to pattern,
else increment $n$, append $n$ to pattern, append x to assoc.
```

For a Perl program that implements this algorithm see the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/1_Monoalph/PattPerl.html`

1.10 Example of Cryptanalysis by Pattern Search

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/1_Monoalph/Puzzle.html`

1.11 Known Plaintext Attack

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/1_Monoalph/knownplain.html`

1.12 Early History of Cryptology

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/1_Monoalph/EarlyHist.html`

1.13 Variants of Cryptographic Procedures

Some Definitions

Substitution: Letters or groups of letters are replaced by other ones.

Monoalphabetic substitution: Each letter is replaced by another letter that is always the same.

Polyalphabetic substitution: Each letter is replaced—depending on its position in the text—by another letter. (The most important method of classical cryptography in the 20th century up to the sixties)

Monographic substitution: Letters are replaced by symbols one at a time.

Polygraphic substitution: In each step one or more letters are replaced by several symbols.

Homophonic substitution: For some plaintext letters or groups there are several choices of ciphertext symbols.

A mathematical model uses a probability space Ω and considers encryption functions of the type

$$f_k : M \times \Omega \longrightarrow \Sigma^*.$$

This is called **probabilistic encryption**.

Transposition: The letters of the plaintext are permuted.

Codebook: Letter groups of various lengths (for example entire words) are replaced by other ones according to a list. Since the Renaissance this was in use under the denomination **Nomenclator**. It was the most used encryption method even in the 20th Century, especially by diplomats.

Source coding (superencrypted code): The plaintext is transformed with a codebook, and the resulting "intermediate text" is encrypted by some kind of substitution.

Book cipher: Plaintext words or letters are looked up in a certain book. As ciphertext one takes the position of the word or letter in the book, for example page number, line number, number of the word (or number of the letter).

Block cipher: In each step a fixed number of letters is substituted at once.

Stream cipher: In each step a single letter is substituted, each time in another way, depending on its position in the plaintext.

Product cipher: A sequence of several transpositions and block substitutions is applied one after the other (also called cipher cascade).

Polygraphic Substitution

For a fixed l in each step an l-gram (block of l letters) is encrypted at once.

As simplest nontrivial example we consider **bigraphic substitution**. Here pairs of letters are encrypted together. The easiest description of the cipher is by a large square of sidelength $n = \#\Sigma$. An example for the standard alphabet:

	a	**b**	**c**	**d**	**...**
a	`CA`	`FN`	`BL`	...	...
b	`SK`	`WM`	...	...	...
c	`HP`	...	...	...	...
d	...	...	...	...	...
...	...	...	...	...	...

With this table `BA` is encrypted as `SK` .

The earliest historical example was given by PORTA in 1563. His bigram table however contained strange symbols meeting the spirit of the time. A picture is on the web page `http://www.staff.uni-mainz.de/pommeren /Cryptology/Classic/1_Monoalph/PortaBi.gif`

Properties of the Polygraphic Substitution

1. The key space of a bigraphic substitution is the set $\mathcal{S}(\Sigma^2)$ of all permutations of the Cartesian product $\Sigma \times \Sigma$. It contains the huge number of $n^2!$ keys. (Of course one also could restrict the keys to a subspace.) The effective keylength is

$$d(F) = \log_2(n^2!) \approx n^2 \cdot \log_2(n^2) = 2 \cdot n^2 \cdot \log_2(n).$$

   For $n = 26$ this amounts to about 4500. Exhaustion surpasses all present or future computer capacity.

2. Compared with a monoalphabetic (and monographic) substitution the frequency distribution of single letters is flattened down. A statistical analysis therefore must resort to bigram frequencies and is a lot harder. Pattern recognition and search for probable words also is harder, but not so much. Also more general attacks with known plaintext are feasible.

3. We may interpret a polygraphic substitution of l-grams as a monographic substitution over the alphabet $\tilde{\Sigma} = \Sigma^l$ of l-grams. The larger

l, the more complicated is the cryptanalysis. However for the *general* polygraphic substitution also the complexity of specifying the key grows with n^l, that is exponentially with l. Therefore this encryption method is useful only with a restricted keyspace. That means we need to fix a class of substitutions $\Sigma^l \longrightarrow \Sigma^l$ whose description is much shorter than the complete value table of n^l entries.

A bigraphic example from history is the PLAYFAIR cipher, invented by WHEATSTONE.

4. Polygraphic substitutions are the predecessors of modern block ciphers.

Codebooks

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/1_Monoalph/Codebook.html`

Chapter 2

Polyalphabetic Substitutions

2.1 Key Alphabets

The Idea of Polyalphabetic Cipher

A polyalphabetic cipher—like a monoalphabetic one—encrypts each letter by a substitution that is defined by a permuted alphabet. However for each letter another alphabet is used, depending on its position in the plaintext.

Thus polyalphabetic encryption breaks the invariants that led to successful cryptanalysis of monoalphabetic substitutions:

- Letter frequencies
- l-gram frequencies
- Patterns

This method was considered unbreakable until the 19th Century, its variants that used cipher machines even until the begin of the computer era. Nevertheless before cipher machines became available polyalphabetic substitution was rarely used because it requires concentrated attention by the operator, and the ciphertext often is irreparably spoiled by encryption errors.

The Key of a Monoalphabetic Substitution

The key of a monoalphabetic substitution over the alphabet Σ is a permutation $\sigma \in \mathcal{S}(\Sigma)$. It has a unique description by the sequence of substituted letters in the order of the alphabet, that is by the family $(\sigma(s))_{s\in\Sigma}$.

Example for the standard alphabet $\Sigma = \{\texttt{A, ..., Z}\}$

1. representation by the permutation table:

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

2. or representation by the permuted alphabet alone:

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

The term "monoalphabetic" reflects that this one (permuted) alphabet defines the complete encryption function.

The Key of a Polyalphabetic Substitution

Now let us write several permuted alphabets below each other and apply them in order: the first alphabet for the first plaintext letter, the second alphabet for the second letter and so on. In this way we perform a **polyalphabetic substitution**. If the list of alphabets is exhausted before reaching the end of the plaintext, then we restart with the first alphabet. This method is called **periodic polyalphabetic substitution**.

Example for the standard alphabet with 5 permuted alphabets

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

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

Using these alphabets we encrypt

```
UNIVERSITAETMAINZ           = plaintext
S    J    W    X            alphabet from line 1
 Y    N    Q    I           alphabet from line 2
  Y    Y    K               alphabet from line 3
   F    Z    U              alphabet from line 4
    E    S    Q             alphabet from line 5
-----------------
SYYFEJNYZSWQKUQXI           = ciphertext
```

Classification of Polyalphabetic Substitutions

We classify polyalphabetic substitutions by four independent binary properties:

- Periodic (or repeated key)

- Aperiodic (or running key)

depending on whether the alphabets repeat cyclically or irregularly.

- Independent alphabets
- Primary alphabet and accompanying secondary alphabets

where secondary alphabets derive from the primary alphabet by a fixed recipe. In the example above we took simple cyclical shifts. A closer inspection reveals that the definition of the shifts is given by the keyword `KLAUS`.

- Progressive alphabet change
- Alphabet choice controlled by a key

depending on whether the alphabets are used one after the other in their original order, or the order is changed by a key.

- Contextfree
- Contextsensitive

depending on whether the alphabets depend only on the position in the text, or also on some adjacent plaintext or ciphertext letters.

In general we take a set of alphabets (only $n!$ different alphabets are possible at all), and use them in a certain order, periodically repeated or not. Often one takes exactly n alphabets, each one beginning with a different letter. Then one can control the alphabet choice by a keyword that is cyclically repeated, or by a long keytext that is at least as long as the plaintext.

2.2 The Invention of Polyalphabetic Substitution

Polyalphabetic Encryption in Renaissance

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/2_Polyalph/Renaissance.html`

The Trithemius Table (aka Vigenère Table)

This table is used for polyalphabetic substitution with the standard alphabet and its cyclically shifted secondary alphabets. It has n rows. The first row consists of the alphabet Σ. Each of the following rows has the alphabet cyclically shifted one position further to the left. For the standard alphabet this looks like this:

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

TRITHEMIUS used it progressively, that is he used the n alphabets from top to down one after the other for the single plaintext letters, with cyclic repetition.

> Note that this procedure involves no key and therefore is not an encryption in the proper sense. Its security is only by obscurity.

Notwithstanding this weakness even TRITHEMIUS's method results in a crucial improvement over the monoalphabetic substitution: Each letter is encrypted to each other the same number of times in the mean. The frequency distribution of the ciphertext is perfectly uniform.

The BELLASO Cipher (aka VIGENÈRE Cipher)

Even VIGENÈRE himself attributes this cipher to BELLASO. It uses the TRITHEMIUS table but with the alphabet choice controlled by a keyword: for each plaintext letter choose the row that begins with this letter. This method uses a key and therefore is a cipher in the proper sense.

As an **example** take the keyword MAINZ. Then the 1st, 6th, 11th, ... plaintext letter is encrypted with the "M row", the 2nd, 7th, 12th, ... with the "A row" and so on. Note that this results in a periodic CAESAR addition of the keyword:

```
p o l y a l p h a b e t i c
M A I N Z M A I N Z M A I N
---------------------------
B O T L Z X P P N A Q T Q P
```

In general the BELLASO cipher uses a group structure on the alphabet Σ. For the key $k = (k_0, \ldots, k_{l-1}) \in \Sigma^l$ we have

Encryption: $c_i = a_i * k_{i \bmod l}$

Decryption: $a_i = c_i * k_{i \bmod l}^{-1}$

The first one who described this cipher algebraically as an addition apparently was the French scholar Claude COMIERS in his 1690 book using a 18 letter alphabet. Lacking a suitable formal notation his description is somewhat long-winded. Source:

> Joachim von zur Gathen: *Claude Comiers: The first arithmetical cryptography.* Cryptologia 27 (2003), 339 - 349.

2.3 Tools for Polyalphabetic Substitution

See the web page http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/2_Polyalph/Tools.html

2.4 Mathematical Description of Periodic Polyalphabetic Substitution

The General Case

In general a periodic polyalphabetic cipher has a key space $K \subseteq \mathcal{S}(\Sigma)^l$, consisting of sequences of l permutations of the alphabet Σ. The key $k = (\sigma_0, \ldots, \sigma_{l-1})$ defines the encryption function f_k: $\Sigma^r \longrightarrow \Sigma^r$ given by

$$\begin{array}{llllllll}
a_0 & a_1 & \ldots & a_{l-1} & a_l & \ldots & a_i & \ldots & a_{r-1} \\
\downarrow & \downarrow & & \downarrow & \downarrow & & \downarrow & & \\
\sigma_0 a_0 & \sigma_1 a_1 & \ldots & \sigma_{l-1} a_{l-1} & \sigma_0 a_l & \ldots & \sigma_{i \bmod l} a_i & \ldots & \ldots
\end{array}$$

The componentwise encryption formula for $c = f_k(a) \in \Sigma^r$ is

$$c_i = \sigma_{i \bmod l}(a_i),$$

and the formula for decryption

$$a_i = \sigma_{i \bmod l}^{-1}(c_i).$$

Effective Key Length

Bellaso Cipher

The primary alphabet is the standard alphabet, and we assume the cryptanalyst knows it. The key is chosen as word (or passphrase) $\in \Sigma^l$. Therefore

$$\begin{aligned} \#K &= n^l, \\ d(F) &= l \cdot \log_2(n). \end{aligned}$$

For $n = 26$ this amounts to $\approx 4.70 \cdot l$. To avoid exhaustion l should be about 10 (pre-computer age), or about 20 (computer age). However there are far more efficient attacks against this cipher than exhaustion, making these proposals for the key lengths obsolete.

Disk Cipher

The key consists of two parts: a permutation $\in \mathcal{S}(\Sigma)$ as primary alphabet, and a keyword $\in \Sigma^l$. Therefore

$$\begin{aligned} \#K &= n! \cdot n^l, \\ d(F) &= \log_2(n!) + l \cdot \log_2(n) \approx (n + l) \cdot \log_2(n) \end{aligned}$$

For $n = 26$ this amounts to $\approx 4.70 \cdot l + 88.38$.

If the enemy knows the primary alphabet, say be capturing a cipher disk, the effective key length reduces to that of the Bellaso cipher.

A More General Case

For a periodic polyalphabetic cipher that uses l independent alphabets,

$$\begin{aligned} K &= \mathcal{S}(\Sigma)^l, \\ d(F) &= \log_2((n!)^l) \approx nl \cdot \log_2(n). \end{aligned}$$

For $n = 26$ this is about $88.38 \cdot l$.

Another View

An l-periodic polyalphabetic substitution is an l-gram substitution, or block cipher of length l, given by the product map

$$(\sigma_0, \ldots, \sigma_{l-1})\colon \Sigma^l = \Sigma \times \cdots \times \Sigma \longrightarrow \Sigma \times \cdots \times \Sigma = \Sigma^l,$$

that is, a monoalphabetic substitution over the alphabet Σ^l. In particular the Bellaso cipher is the shift cipher over Σ^l, identified with $(\mathbb{Z}/n\mathbb{Z})^l$.

For $\Sigma = \mathbb{F}_2$ the Bellaso cipher degenerates to the simple XOR on $\mathbb{F}_2^l$.

2.5 The Cipher Disk Algorithm

Mathematical Notation

Take the alphabet $\Sigma = \{s_0, \ldots, s_{n-1}\}$, and interpret (or code) it as the additive group of the ring $\mathbb{Z}/n\mathbb{Z}$. The key $(\sigma, k) \in \mathcal{S}(\Sigma) \times \Sigma^l$ of a disk cipher consists of a primary alphabet (represented by the permutation σ) and a keyword $k = (k_0, \ldots, k_{l-1}) \in \Sigma^l$. Our notation for the corresponding encryption function is

$$f_{\sigma,k}\colon \Sigma^* \longrightarrow \Sigma^*$$

Special case: The BELLASO cipher with keyword k is $f_{\varepsilon,k}$ where $\varepsilon \in \mathcal{S}(\Sigma)$ denotes the identity permutation.

The Alphabet Table

We arrange the alphabets for the polyalphabetic substitution in form of the usual table:

s_0	s_1	s_2	$\ldots$	s_{n-1}
t_0	t_1	t_2	$\ldots$	t_{n-1}
t_1	t_2	t_3	$\ldots$	t_0
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
t_{n-1}	t_0	t_1	$\ldots$	t_{n-2}

where $t_i = \sigma s_i$ for $0 \le i \le n-1$.

Note that whenever we refer to an alphabet table we implicitely use an order on the alphabet Σ. This order manifests itself by indexing the letters as $s_0, \ldots, s_{n-1}$.

The Encryption Function

Now we encrypt a text $a = (a_0, a_1, a_2, \ldots) \in \Sigma^r$ using this notation. Let $a_i = s_q$ and $k_i = t_p$ as letters of the alphabet. Then we read the ciphertext letter c_i off from row p and column q of the table:

$$c_i = t_{p+q} = \sigma s_{p+q} = \sigma(s_p + s_q) \quad \text{[sums in } \mathbb{Z}/n\mathbb{Z}\text{]}.$$

We have

$$k_i = t_p = \sigma(s_p), \quad s_p = \sigma^{-1}(k_i), \quad \text{hence } c_i = \sigma(a_i + \sigma^{-1}(k_i)).$$

If we denote by f_σ the monoalphabetic substitution corresponding to σ, then this derivation proves:

Theorem 1 *The disk cipher $f_{\sigma,k}$ is the composition (or "superencryption") of the* BELLASO *encryption $f_{\varepsilon,k'}$, where $k' = f_\sigma^{-1}(k)$, with the monoalphabetic substitution f_σ,*

$$f_{\sigma,k} = f_\sigma \circ f_{\varepsilon,k'}$$

Algorithm

The naive straightforward algorithm for the disk cipher is

- Take the next plaintext letter.
- Take the next alphabet.
- Get the next ciphertext letter.

From Theorem 1 we derive an algorithm that is a bit more efficient:

1. Take $k' = f_\sigma^{-1}(k)$, in coordinates $k'_i = \sigma^{-1}(k_i)$ for $0 \leq i < l$.
2. Add a and (the periodically extended) k' over $\mathbb{Z}/n\mathbb{Z}$, and get b, in coordinates $b_j = a_j + k'_{j \bmod l}$
3. Take $c = f_\sigma(b) \in \Sigma^r$, in coordinates $c_j = \sigma(b_j)$.

A Perl program implementing this algorithm is on the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Perl/porta.pl`, the corresponding program for decryption on `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Perl/portadec.pl`. They can be called online from the pages `http://www.staff.uni-mainz.de/pommeren/Kryptologie/Klassisch/2_Polyalph/portaenc.html` and `http://www.staff.uni-mainz.de/pommeren/Kryptologie/Klassisch/2_Polyalph/portadec.html`

2.6 Analysis of Periods

Kasiski's approach

Already in the 16th Century Porta and the Argentis occasionally broke polyalphabetic encryptions by guessing the key or a probable word. For some more historical bits see the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/2_Polyalph/AnaPer.html`

An attack with known plaintext is easy against a disk cipher as soon as the primary alphabet is compromised, for example by a lost cipher disk. It is trivial against the Bellaso cipher that uses the standard alphabet. In contrast it is quite difficult against ciphers that use independent alphabets.

In 1863 the Prussian Major F. W. Kasiski published a solution that immediately demolished the belief in the security of periodic polyalphabetic ciphers. In fact Babbage had found this method ten years before but never published it. Therefore it is appropriate to credit the method to Kasiski.

The solution proceeds in three steps:

1. Determine the period l.

2. Arrange the ciphertext in rows of length l. Then the columns each are encrypted by a (different) monoalphabetic substitution.

3. Break the monoalphabetic columns.

Step 3, that is cryptanalyzing the monoalphabetically encrypted columns, faces the complication that the columns don't represent connected meaningful texts. Pattern search is pointless. However frequency analysis makes sense.

There are some simplifications for dependent alphabets:

- Adjusting the frequency curves. This works when the primary alphabet is known, see Sections 2.7 and 2.8.

- Symmetry of position when the primary alphabet is unknown (not treated here, but see Chapter 5). This method, proposed by KERCKHOFFS, uses regularities in the alphabet table to infer further entries from already known entries, for example by completing the diagonals in the alphabet table of a disk cipher.

Especially simple is the situation with BELLASO's cipher, as soon as the period is known: Each column is CAESAR encrypted. Therefore we need to identify only one plaintext letter in each column.

How to Determine the Period

Three approaches to determining the period of a periodic polyalphabetic cipher are

1. Exhaustion: Try $l = 1, 2, 3, \ldots$ one after each other. The correct l reveals itself by the appropriate frequency distribution of the letters in each column. As tools use some statistical "goodness of fit" tests. We'll study appropriate methods in Chapter 3.

2. Search for repetitions, see next subsection. This is an instance of the general method "pattern search".

3. Coincidence analysis after FRIEDMAN, KULLBACK, and SINKOV. This is also a subject of Chapter 3, and is an instance of the general method "statistical analysis".

In contrast to the exhaustion approach the other two methods immediately identify the situation where there is *no period*.

Search for Repetitions

We start with three observations:

1. If a plaintext is encrypted using l alphabets in cyclic order, and if a sequence of letters occurs k times in the plaintext, than this sequence occurs in the ciphertext about k/l times encrypted with the same sequence of alphabets.
2. In each of these occurrences where the sequence is encrypted the same way the ciphertext contains a repeated pattern in a distance that is a multiple of l, see Figure 2.1.
3. Not every repeated pattern in the ciphertext necessarily arises in this way. It could be by accident, see Figure 2.2. However the probability of this event is noticeably smaller.

An assessment of this probability is related to the birthday paradox of probability theory, and is contained in Appendix C. It was published in

> K. Pommerening: *Kasiski's Test: Couldn't the repetitions be by accident?* Cryptologia 30 (2006), 346-352.

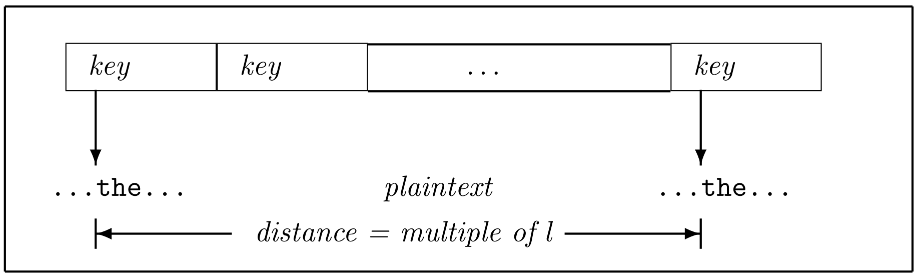


Figure 2.1: Repetition in ciphertext

A Perl program that searches for repetitions is on the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Perl/kasiski.pl`

For online use see the web form `http://www.staff.uni-mainz.de/pommeren/Kryptologie/Klassisch/2_Polyalph/kasiski1.html`

2.7 Cryptanalysis of a Polyalphabetic Ciphertext

(for a German plaintext)

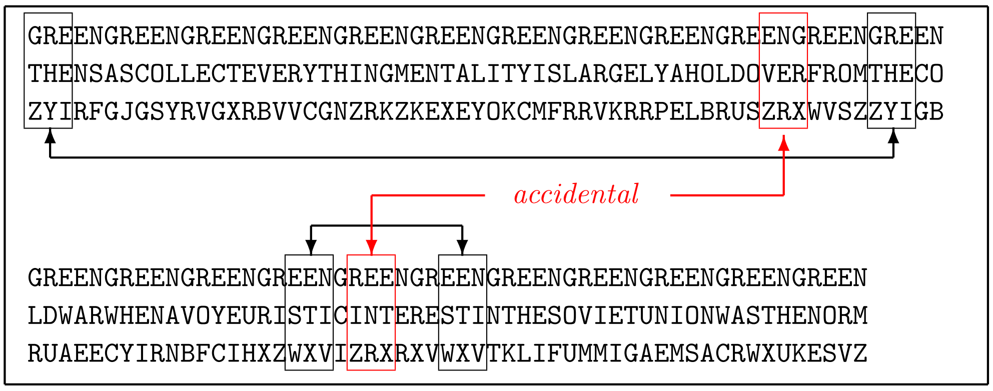

GREENGREENGREENGREENGREENGREENGREENGREENGREENGREENGREENGREEN
THENSASCOLLECTEVERYTHINGMENTALITYISLARGELYAHOLDOVERFROMTHECO
ZYIRFGJGSYRVGXRBVVCGNZRKZKEXEYOKCMFRRVKRRPELBRUSZRXWVSZZYIGB
accidental
GREENGREENGREENGREENGREENGREENGREENGREENGREENGREENGREEN
LDWARWHENAVOYEURISTICINTERESTINTHESOVIETUNIONWASTHENORM
RUAEECYIRNBFCIHXZWXVIZRXRXVWXVTKLIFUMMIGAEMSACRWXUKESVZ

Figure 2.2: True and accidental repetitions

Finding the Period by Searching Repetitions

`http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/2_Polyalph/Kasiski.html`

Column Analysis and Rearrangement

`http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/2_Polyalph/Columns.html` and `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/2_Polyalph/Rearrang.html`

2.8 Rearranging the Columns

The Problem

The formula for the disk cipher from Theorem 1 was $f_{\sigma,k} = f_\sigma \circ f_{\varepsilon,k'}$ where $k' = f_\sigma^{-1}(k)$. However we didn't use this formula in our analysis but rather a similar one of the type $f_{\sigma,k} = g \circ f_\sigma$ where g should describe the shifts in the alphabets and g^{-1} the rearrangement. What we did was first rearrange the shifts in the different columns, and then solve the resulting monoalphabetic ciphertext. Note that for this method to work in general the primary alphabet must be known. Unfortunately there is no useful general interpretation of the formula $g = f_\sigma \circ f_{\varepsilon,k'} \circ f_\sigma^{-1}$ when σ is unknown.

We'll analyze the situation, first for an example.

Example

We take the standard alphabet $\Sigma =$ `A...Z`, and consider an alphabet table.

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

Phrased in terms of permutations the top row, Row 0, the standard alphabet, corresponds to the identical permutation $\varepsilon \in \mathcal{S}(\Sigma)$. The next row, Row 1, the primary alphabet, corresponds to the permutation $\sigma \in \mathcal{S}(\Sigma)$. Row 2 corresponds to $\sigma \circ \tau$, where τ is the alphabet shift

$$\tau(\mathtt{A}) = \mathtt{B}, \quad \tau(\mathtt{B}) = \mathtt{C}, \quad \ldots, \quad \tau(\mathtt{Z}) = \mathtt{A}$$

Row i corresponds to $\sigma \circ \tau^{i-1}$. For the concrete example we have

$$\sigma(\mathtt{A}) = \mathtt{Q}, \quad \sigma(\mathtt{B}) = \mathtt{W}, \quad \ldots$$

and thus

$$\sigma \circ \tau(\mathtt{A}) = \sigma(\mathtt{B}) = \mathtt{W}, \quad \sigma \circ \tau(\mathtt{B}) = \sigma(\mathtt{C}) = \mathtt{E}, \quad \ldots$$

On the other hand

$$\tau \circ \sigma(\mathtt{A}) = \tau(\mathtt{Q}) = \mathtt{R}, \quad \tau \circ \sigma(\mathtt{B}) = \tau(\mathtt{W}) = \mathtt{X}, \quad \ldots$$

Shifts in the Primary Alphabet

Recall the alphabet table in the general case

s_0	s_1	s_2	$\ldots$	s_{n-1}
t_0	t_1	t_2	$\ldots$	t_{n-1}
t_1	t_2	t_3	$\ldots$	t_0
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
t_{n-1}	t_0	t_1	$\ldots$	t_{n-2}

where $t_i = \sigma s_i$ for $0 \leq i \leq n-1$, and σ is the permutation that defines the primary alphabet.

Identify as usual the alphabet $\Sigma = \{s_0, \ldots, s_{n-1}\}$ with $\mathbb{Z}/n\mathbb{Z}$, the integers $\bmod n$, via $i \mapsto \sigma_i$ and take indices $\bmod n$. Mathematical expressions for the shifts in the original and primary alphabets are

- τ = shift by 1 in the original alphabet, $\tau(s_i) = s_{i+1}$.
- τ^k = shift by k in the original alphabet, $\tau^k(s_i) = s_{i+k}$.
- $\sigma\tau\sigma^{-1}$ = shift by 1 in the primary alphabet,

$$t_i \overset{\sigma^{-1}}{\mapsto} s_i \overset{\tau}{\mapsto} s_{i+1} \overset{\sigma}{\mapsto} t_{i+1}$$

- $\sigma\tau^k\sigma^{-1} = (\sigma\tau\sigma^{-1})^k =$ shift by k in the primary alphabet.

The alphabet table, interpreted as list of permutations, is the orbit of $\sigma \in \mathcal{S}(\Sigma)$ under iterated right translation by τ (or under the cyclic subgroup $\langle\tau\rangle \subseteq \mathcal{S}(\Sigma)$ generated by τ).

The "naive" shift that we performed in Section 2.7 shifted the single letters of the primary alphabet by a certain number of positions in the *standard* alphabet—we performed $\tau^i \circ \sigma$ for some value i. Why was this successful? Under what conditions are the naively shifted primary alphabets again rows of the alphabet table?

Decimated alphabets

We take the ordering of the alphabets into account and let $T_1 = (t_0, \ldots, t_{n-1})$ be the ordered primary alphabet where $t_i = \sigma s_i$. The secondary alphabets then are $T_i = (t_{i-1}, \ldots, t_{n-1}, t_0, \ldots, t_{i-2})$ for $i = 2, \ldots, n$. They correspond to the permutations $\sigma \circ \tau^{i-1}$, that is $T_i = (\sigma s_{i-1}, \sigma s_i, \ldots)$.

The primary alphabet used in the example of Section 2.7 was of a special kind: It had $t_i = s_{3i \bmod 26}$. The corresponding formula for the general case is

$$t_i = s_{ki \bmod n},$$

and t_i for $i = 0, \ldots, n-1$ runs through all elements of Σ if and only if k and n are relative prime.

Definition. Let the alphabet Σ be linearly ordered as $(s_0, \ldots, s_{n-1})$, and let $\gcd(k, n) = 1$. The (ordered) alphabet $T = (t_0, \ldots, t_{n-1})$ is called **decimated alphabet** of order k (of Σ with the given linear order relation) if there is an index $p \in \{0, \ldots, n-1\}$ such that $t_{p+i} = s_{ki \bmod n}$ for $i = 0, \ldots, n-1$.

That means, beginning with $t_p = s_0$ we take each k-th letter from Σ.

If the primary alphabet is decimated, so are all the secondary alphabets; we get them all by varying the index p.

Now when we apply the shift τ to the (ordered) primary and secondary alphabets $T_1, \ldots, T_n$ we get new alphabets $f_\tau(T_1), \ldots, f_\tau(T_n)$; note that we interpret the n-tuples T_i as texts and apply τ elementwise. The question we want to answer is whether the $f_\tau(T_i)$ belong to the collection of the T_i. The answer involves the normalizer $N(\langle\tau\rangle)$ of the subgroup $\langle\tau\rangle \leq \mathcal{S}(\Sigma)$.

Theorem 2 (Decimated alphabets) *Let the alphabet Σ be linearly ordered as $(s_0, \ldots, s_{n-1})$. Let the (ordered) primary alphabet $T_1 = (t_0, \ldots, t_{n-1})$ be defined by $t_i = \sigma s_i$ where $\sigma \in \mathcal{S}(\Sigma)$, and let $T_2, \ldots, T_n$ be the corresponding ordered secondary alphabets. Then the following statements are equivalent:*

(i) *There is a $j \in \{1, \ldots, n\}$ with $f_\tau(T_1) = T_j$.*

(ii) *f_τ permutes the $\{T_1, \ldots, T_n\}$.*
(iii) *T_1 is a decimated alphabet of Σ.*
(iv) *$\sigma \in N(\langle\tau\rangle)$.*

Proof. "(i) $\Longrightarrow$ (iv)": $f_\tau(T_1) = T_j$ means that $\tau \circ \sigma = \sigma \circ \tau^j$. Then $\sigma^{-1} \circ \tau \circ \sigma \in \langle\tau\rangle$ or $\sigma \in N(\langle\tau\rangle)$.

"(iv) $\Longrightarrow$ (iii)": By conjugation σ defines an automorphism of the cyclic group $\langle\tau\rangle$. These automorphisms are known, the following Lemma 1 gives $\sigma \circ \tau \circ \sigma^{-1} = \tau^k$ for some k, relative prime with n. The letter s_0 occurs somewhere in T_1, so let $s_0 = t_p$. Then $\sigma s_p = t_p = s_0$ and

$$t_{j+p} = \sigma s_{j+p} = \sigma\tau^j s_p = \tau^{jk}(\sigma s_p) = \tau^{jk} s_0 = s_{jk} \quad \text{for } j = 0, \ldots, n-1,$$

where as usual we take the indices mod n.

"(iii) $\Longrightarrow$ (iv)": Let p and k as in the definition. For any i we have

$$\tau^k \sigma s_{p+i} = \tau^k t_{p+i} = \tau^k s_{ki} = s_{ki+k} = s_{k(i+1)} = t_{p+i+1} = \sigma s_{p+i+1} = \sigma\tau s_{p+i}.$$

From this we conclude $\sigma \circ \tau = \tau^k \circ \sigma$ or $\sigma \circ \tau \circ \sigma^{-1} \in \langle\tau\rangle$.

"(iv) $\Longrightarrow$ (ii)": We have $\sigma^{-1} \circ \tau \circ \sigma = \tau^{k'}$ where $k'k \equiv 1 \pmod n$ whence $\tau \circ \sigma = \sigma \circ \tau^{k'}$. The permuted alphabet T_i corresponds to the permutation $\sigma \circ \tau^{i-1}$. Therefore $f_\tau T_i$ corresponds to $\tau \circ \sigma \circ \tau^{i-1} = \sigma \circ \tau^{k'+i-1}$. We conclude $f_\tau T_i = T_{k'+i}$.

"(ii) $\Longrightarrow$ (i)" is the restriction to a special case. $\diamond$

Lemma 1 *Let $G = \langle g\rangle$ be a finite cyclic group of order m. Then the automorphisms of G are the power maps $g \mapsto g^k$ where k is relatively prime to m. In other words, the automorphism group* $\operatorname{Aut} G$ *is isomorphic with the multiplicative group $(\mathbb{Z}/m\mathbb{Z})^\times$.*

Proof. Let h be an automorphism of G. Then $h(g) = g^k$ for some $k \in \mathbb{Z}$. This k uniquely defines h on all of G, and k is uniquely determined by h up to multiples of $\operatorname{Ord}(g) = m$. The power map $g \mapsto g^k$ is bijective if and only if k is relatively prime to m. $\diamond$

2.9 Summary

The canonical method of cryptanalyzing the disk cipher $f_{\sigma,k}$ proceeds in three steps:

1. Determine the period l.

2. Rearrange the ciphertext in rows of length l.

3. Reconstruct the monoalphabets of the columns.

Note that the effort is essentially independent of the key length. However the success probability decreases with the period length, because

- The probability of finding non-accidental repetitions decreases.
- Finding useful frequency distributions in the columns becomes harder.

Some special cases have special facilities:

- For a BELLASO cipher or more generally for a disk cipher with a decimated alphabet or even more generally for a disk cipher with a known primary alphabet we may rearrange the monoalphabets of the columns and are left with a large monoalphabetic ciphertext.
- Known plaintext gives the plaintext equivalents of single letters in a few columns that may be extended to other columns by symmetry of position when the alphabets are related, for example for a disk cipher (not treated here, but see Chapter 5).

These findings result in two recommendations for the use of polyalphabetic ciphers:

- The larger the period, the better the security.
- Independent alphabets more reliably protect from attacks.

Both of these recommendations make polyalphabetic ciphers more cumbersome in routine use, and therefore in history were adopted only after many failures.

Chapter 3

Some Statistical Properties of Languages

In this chapter we study certain statistical properties of texts and languages. These help to answer questions such as:

- Does a given text belong to a certain language? Can we derive an algorithm for automatically distinguishing valid plaintext from random noise? This is one of the central problems of cryptanalysis.
- Do two given texts belong to the same language?
- Can we decide these questions also for encrypted texts? Which properties of texts are invariant under certain encryption procedures? Can we distinguish encrypted plaintext from random noise?
- Is a given ciphertext monoalphabetically encrypted? Or polyalphabetically with periodic repetition of alphabets? If so, what is the period?
- How to adjust the alphabets in the columns of a periodic cipher? Or of several ciphertexts encrypted with the same key and correctly aligned in depth?

To get useful information on these questions we define some statistical reference numbers and analyze the distributions of these numbers. The main methods for determining reference values are:

- **Exact calculation.** This works for artificial languages with exact descriptions and for simple distributions, but for natural languages it is hopeless.
- **Modelling.** We try to build a simplified model of a language, based on letter frequencies etc. and hope that the model on the one hand approximates the statistical properties of the language closely enough,

and on the other hand is simple enough that it allows the calculation of the relevant statistics. The two most important models are:

  - the computer scientific model that regards a language as a fixed set of strings with certain statistical properties,
  - the stochastic model that regards a language as a finite stationary Markov process. This essentially goes back to Shannon in the 1940s after at least 20 years of naive but successful use by the Friedman school.

- **Simulation.** We take a large sample of texts from a language and determine the characteristic reference numbers by counting. In this way we find empirical approximations to the distributions and their characteristic properties.

The main results of this section go back to Friedman, Kullback, and Sinkov in the 1920s and 1930s. However the statistical methodology has since developed and now provides a uniform conceptual framework for statistical tests and decisions.

For a systematic treatment of the first two questions above a good reference is [8, 9]. An elementary but mathematically sound introduction to probability and statistics is [10], whereas [16] and [25] use an elementary "naive" approach to probability theory.

3.1 Recognizing Plaintext: Friedman's Most-Frequent-Letters Test

We begin with the first question: *Does a given text belong to a certain language?* Friedman gave a quite simple procedure for distinguishing valid text from random noise that works surprisingly well, even for short texts. Besides it makes a smooth introduction to statistical test theory.

Friedman's Procedure

Assume we are given a string of letters and want to decide whether it is a part of a meaningful text (in a given language, say English), or whether it is random gibberish. Our first contact with this problem was the exhaustion attack against the simple shift cipher that produced 26 strings, exactly one of which represented the correct solution. Cherry-picking it was easy by visual inspection. But for automating this decision procedure we would prefer a quantitative criterion.

Such a criterion was proposed by Friedman in Riverbank Publication No. 16 from 1918 [7]. The procedure is

1. Identify a set of most frequent letters from the target language. For English take `ETOANIRSHD` that make up 73.9% of an average English text but only $10/26 \approx 38.5\%$ of a random text.
2. Count the cumulative frequencies of these most-frequent letters for each of the candidate strings.
3. Pick the string with the highest score. If this doesn't work, also consider the next highest scores.

Example. For the Caesar example in Section 1.3 the scores are in Table 3.1. We immediately see that the correct solution `CAESAR` has the highest score (even if this is not a genuine English word).

Table 3.1: Friedman *scores for the exhausion of a shift cipher*

FDHVDU	3	OMQEMD	3		XVZNVM	1	
GEIWEV	3	PNRFNE	4	<---	YWAOWN	3	
HFJXFW	1	QOSGOF	3		ZXBPXO	1	
IGKYGX	1	RPTHPG	3		AYCQYP	1	
JHLZHY	2	SQUIQH	3		BZDRZQ	2	
KIMAIZ	3	TRVJRI	4	<---	CAESAR	5	<===
LJNBJA	2	USWKSJ	2		DBFTBS	3	
MKOCKB	1	VTXLTK	2		ECGUCT	2	
NLPDLC	2	WUYMUL	0				

The example shows that Friedman's procedure seems to work well even for quite short strings. To confirm this observation we analyze the distribution of the Most-Frequent-Letters scores—in short **MFL scores**—for strings of natural languages and for random strings. First we consider this task from a theoretic viewpoint, then we also perform some empirical evaluations.

The distribution of MFL Scores

Consider strings of length r over an alphabet Σ whose letters are independently drawn with certain probabilities, the letter $s \in \Sigma$ with probability p_s. Let $\mathcal{M} \subseteq \Sigma$ be a subset and $p = \sum_{s\in\mathcal{M}} p_s$ be the cumulative probability of the letters in $\mathcal{M}$. The **MFL score** of a string $a = (a_1, \ldots, a_r) \in \Sigma^r$ with respect to $\mathcal{M}$ is

$$N_{\mathcal{M}}(a) = \#\{i \mid a_i \in \mathcal{M}\}.$$

To make the scores for different lengths comparable we also introduce the **MFL rate**

$$\nu_{\mathcal{M}}(a) = \frac{N_{\mathcal{M}}(a)}{r}.$$

The MFL rate defines a function

$$\nu_{\mathcal{M}} \colon \Sigma^* \longrightarrow \mathbb{Q}.$$

(Set $\nu_{\mathcal{M}}(\emptyset) = 0$ for the empty string $\emptyset$ of length 0.)

The distribution of scores is binomial, that is the probability that a string $a \in \Sigma^r$ contains exactly k letters from $\mathcal{M}$ is given by the binomial distribution

$$P(a \in \Sigma^r \mid N_{\mathcal{M}}(a) = k) = B_{r,p}(k) = \binom{r}{k} \cdot p^k \cdot (1-p)^{r-k}.$$

Random strings. We take the 26 letter alphabet `A...Z` and pick a subset $\mathcal{M}$ of 10 elements. Then $p = 10/26 \approx 0.385$, and this is also the expected value of the MFL rate $\nu_{\mathcal{M}}(a)$ for $a \in \Sigma^*$. For strings of length 10 we get the two middle columns of Table 3.2.

English strings. Assuming that the letters of an English string are independent is certainly only a rough approximation to the truth, but the best we can do for the moment, and, as it turns out, not too bad. Then we take $\mathcal{M} = \{$`ETOANIRSHD`$\}$ and $p = 0.739$ and get the rightmost two columns of Table 3.2.

Table 3.2: *Binomial distribution for $r = 10$. The columns headed "Total" contain the accumulated probabilities.*

		$p = 0.385$ **(Random)**		$p = 0.739$ **(English)**	
Score	Coefficient	Probability	Total	Probability	Total
0	$B_{10,p}(0)$	0.008	0.008	0.000	0.000
1	$B_{10,p}(1)$	0.049	0.056	0.000	0.000
2	$B_{10,p}(2)$	0.137	0.193	0.001	0.001
3	$B_{10,p}(3)$	0.228	0.422	0.004	0.005
4	$B_{10,p}(4)$	0.250	**0.671**	0.020	**0.024**
5	$B_{10,p}(5)$	0.187	0.858	0.067	0.092
6	$B_{10,p}(6)$	0.097	0.956	0.159	0.250
7	$B_{10,p}(7)$	0.035	0.991	0.257	0.507
8	$B_{10,p}(8)$	0.008	0.999	0.273	0.780
9	$B_{10,p}(9)$	0.001	1.000	0.172	0.951
10	$B_{10,p}(10)$	0.000	1.000	0.049	1.000

A Statistical Decision Procedure

What does this table tell us? Let us interpret the cryptanalytic task as a decision problem: We set a threshold value T and decide:

- A string with score $\leq T$ is probably random. We discard it.
- A string with score $> T$ could be true plaintext. We keep it for further examination.

There are two kinds of possible errors in this decision:

1. A true plaintext has a low score. We miss it.
2. A random string has a high score. We keep it.

Example. Looking at Table 3.2 we are tempted to set the threshold value at $T = 4$. Then (in the long run) we'll miss 2.4% of all true plaintexts because the probability for an English 10 letter text string having an MFL score ≤ 4 is 0.024. On the other hand we'll discard only 67.1% of all random strings and erroneously keep 32.9% of them.

The lower the threshold T, the more unwanted random strings will be selected. But the higher the threshold, the more true plaintext strings will be missed. Because the distributions of the MFL scores for "Random" and "English" overlap there is no clear cutpoint that always gives the correct decision.

This is a typical situation for statistical decision problems (or **tests**). The statistician usually bounds one of the two errors by a fixed amount, usually 5% or 1%, and calls this the **error of the first kind**, denoted by α. (The complementary value $1 - \alpha$ is called the sensitivity of the test.) Then she tries to minimize the other error, the **error of the second kind**, denoted by β. The complementary value $1 - \beta$ is called the **power** (or specifity) of the test. FRIEDMAN's MFL-method, interpreted as a statistical test (for the "null hypothesis" of English text against the "alternative hypothesis" of random text), has a power of $\approx 67\%$ for English textstrings of length 10 and $\alpha = 2.4\%$. This α-value was chosen because it is the largest one below 5% that really occurs in the sixth column of Table 3.2.

To set up a test the statistician faces two choices. First she has to choose between "first" and "second" kind depending on the severity of the errors in the actual context. In our case she wants to bound the number of missed true plaintexts at a very low level—a missed plaintext renders the complete cryptanalysis obsolete. On the other hand keeping too many random strings increases the effort of the analysis, but this of somewhat less concern.

The second choice is the error level α. By these two choices the statistician adapts the test to the context of the decision problem.

Remark. We won't discuss the trick of raising the power by exhausting the α-level, randomizing the decision at the threshold value.

Note. There is another ("BAYESian") way to look at the decision problem. The **predictive values** give the probabilities that texts are actually what we decide them to be. If we decide "random" for texts with MFL score ≤ 4, we'll be correct for about 671 of 1000 random texts and err for 24 of 1000 English texts. This makes 695 decisions for random of which 671 are correct. The predictive value of our "random" decision is $96.5\% \approx 671/695$. The decision "English" for an MFL score > 4 will be correct for 976 of 1000 English texts and false for 329 of 1000 random texts. Hence the predictive value of the decision "English" is about $75\% \approx 976/1305$. That means that if we pick up texts (of length 10) with a score of at least 5, then (in the long run) one out of four selected texts will be random.

Other Languages: German and French

Table 3.3: *Distribution of MFL scores for $r = 10$*

	$p = 0.751$ **(German)**		$p = 0.791$ **(French)**	
Score	Probability	Total	Probability	Total
0	0.000	0.000	0.000	0.000
1	0.000	0.000	0.000	0.000
2	0.000	0.000	0.000	0.000
3	0.003	0.003	0.001	0.001
4	0.016	**0.019**	0.007	0.008
5	0.058	0.077	0.031	**0.039**
6	0.145	0.222	0.098	0.137
7	0.250	0.471	0.212	0.350
8	0.282	0.754	0.301	0.651
9	0.189	0.943	0.253	0.904
10	0.057	1.000	0.096	1.000

German: The ten most frequent letters are `ENIRSATDHU`. They make up 75.1% of an average German text.

French: The ten most frequent letters are `EASNTIRULO`. They make up 79.1% of an average French text.

With these values we supplement Table 3.2 by Table 3.3.

As before for English we get as conclusions for textstrings of length 10:

German: With a threshold of $T = 4$ and $\alpha = 1.9\%$ the MFL-test has a power of 67%. The predictive value for "German" is $75\% \approx 981/1310$.

French: With a threshold of $T = 5$ and $\alpha = 3.9\%$ the MFL-test has a power of 86%. The predictive value for "French" is 87% $\approx 961/1103$.

Textstrings of length 20

Table 3.4: *Distribution of MFL scores for* $r = 20$

	Random		**English**		**German**		**French**	
Score	Prob	Total	Prob	Total	Prob	Total	Prob	Total
0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	0.001	0.001	0.000	0.000	0.000	0.000	0.000	0.000
2	0.005	0.005	0.000	0.000	0.000	0.000	0.000	0.000
3	0.017	0.022	0.000	0.000	0.000	0.000	0.000	0.000
4	0.045	0.067	0.000	0.000	0.000	0.000	0.000	0.000
5	0.090	0.157	0.000	0.000	0.000	0.000	0.000	0.000
6	0.140	0.297	0.000	0.000	0.000	0.000	0.000	0.000
7	0.175	0.472	0.000	0.000	0.000	0.000	0.000	0.000
8	0.178	0.650	0.001	0.001	0.001	0.001	0.000	0.000
9	0.148	0.798	0.004	0.006	0.003	0.004	0.001	0.001
10	0.102	**0.900**	0.013	**0.019**	0.010	0.013	0.003	0.004
11	0.058	**0.958**	0.034	0.053	0.026	**0.040**	0.010	0.013
12	0.027	**0.985**	0.072	0.125	0.060	0.100	0.028	**0.041**
13	0.010	0.996	0.125	0.250	0.111	0.211	0.064	0.105
14	0.003	0.999	0.178	0.428	0.168	0.379	0.121	0.226
15	0.001	1.000	0.201	0.629	0.202	0.581	0.184	0.410
16	0.000	1.000	0.178	0.807	0.191	0.772	0.217	0.627
17	0.000	1.000	0.119	0.925	0.135	0.907	0.193	0.820
18	0.000	1.000	0.056	0.981	0.068	0.975	0.122	0.942
19	0.000	1.000	0.017	0.998	0.022	0.997	0.049	0.991
20	0.000	1.000	0.002	1.000	0.003	1.000	0.009	1.000

The distribution is given in Table 3.4. We conclude:

English: With a threshold of $T = 10$ and $\alpha = 1.9\%$ the MFL-test has a power of 90% and a predictive value of 91% $\approx 981/1081$.

German: With a threshold of $T = 11$ and $\alpha = 4.0\%$ the MFL-test has a power of 96% and a predictive value of 96% $\approx 960/1002$.

French: With a threshold of $T = 12$ and $\alpha = 4.1\%$ the MFL-test has a power of 98.5% and a predictive value of 98.5% $\approx 959/974$.

3.2 Empirical Results on MFL Scores

The power calculations for the tests—not the tests themselves!—relied on the independency of the letters in a string. This assumption is clearly false for natural languages. Therefore getting experimental results for the distributions of the MFL scores makes sense. The description of the experiments is in Appendix D

The empirical values for English amount to a power of 68% (instead of 67%) and a predictive value of 75% (75%), a power of 63% (theory: 67%) and a predictive value of 75% (75%) for German, and a power of 87% (86%) and a predictive value of 88% (87%) for French.

3.3 Application to the Cryptanalysis of the Bellaso Cipher

The Friedman procedure doesn't need contiguous plaintext. It also works when we pick out isolated letters from a meaningful text. In particular it works in a (semi-) automated approach to adjusting the columns of a Bellaso ciphertext.

As an example we consider the ciphertext

```
UMHOD BLRHT SCWWJ NHZWB UWJCP ICOLB AWSWK CLJDO WWJOD L
```

We assume a Bellaso cipher with period 4. (The Kasiski analysis yields a single significant repetition `WWJ` at a distance of 28.) The four columns (written horizontally) are

```
UDHWHUPLSLWD  MBTWZWIBWJWL  HLSJWJCAKDJ  ORCNBCOWCOO
```

For an exhaustion attack we complete the alphabets (i. e. we increment the letters step by step) and count the MFL scores for letter combinations in each row, see Table 3.5.

We pick up the most promising result for each column:

```
Column 1: RAETERMIPITA
Column 2: ETLOROATOBOD
Column 3: PTARERKISLR
Column 4: ADOZNOAIOAA or EHSDRSEMSEE
```

Only for column 4 we have more than one choice. However the first choice yields an ugly "plaintext". We drop it and keep

```
Col 1: RAETERMIPITA
Col 2: ETLOROATOBOD
Col 3: PTARERKISLR
Col 4: EHSDRSEMSEE
```

Table 3.5: *MFL scores for the example*

UDHWHUPLSLWD	5	MBTWZWIBWJWL	2	HLSJWJCAKDJ	4	ORCNBCOWCOO	6
VEIXIVQMTMXE	5	NCUXAXJCXKXM	2	IMTKXKDBLEK	4	PSDOCDPXDPP	5
WFJYJWRNUNYF	3	ODVYBYKDYLYN	4	JNULYLECMFL	2	QTEPDEQYEQQ	5
XGKZKXSOVOZG	3	PEWZCZLEZMZO	3	KOVMZMFDNGM	3	RUFQEFRZFRR	5
YHLALYTPWPAH	5	QFXADAMFANAP	6	LPWNANGEOHN	7	SVGRFGSAGSS	6
ZIMBMZUQXQBI	2	RGYBEBNGBOBQ	4	MQXOBOHFPIO	5	TWHSGHTBHTT	8*
AJNCNAVRYRCJ	6	SHZCFCOHCPCR	5	NRYPCPIGQJP	3	UXITHIUCIUU	5
BKODOBWSZSDK	6	TIADGDPIDQDS	9*	OSZQDQJHRKQ	5	VYJUIJVDJVV	2
CLPEPCXTATEL	5	UJBEHEQJERET	7	PTARERKISLR	8*	WZKVJKWEKWW	1
DMQFQDYUBUFM	2	VKCFIFRKFSFU	3	QUBSFSLJTMS	4	XALWKLXFLXX	1
ENRGREZVCVGN	6	WLDGJGSLGTGV	3	RVCTGTMKUNT	5	YBMXLMYGMYY	0
FOSHSFAWDWHO	8*	XMEHKHTMHUHW	6	SWDUHUNLVOU	5	ZCNYMNZHNZZ	4
GPTITGBXEXIP	5	YNFILIUNIVIX	6	TXEVIVOMWPV	4	ADOZNOAIOAA	10*
HQUJUHCYFYJQ	2	ZOGJMJVOJWJY	2	UYFWJWPNXQW	1	BEPAOPBJPBB	3
IRVKVIDZGZKR	5	APHKNKWPKXKZ	3	VZGXKXQOYRX	2	CFQBPQCKQCC	0
JSWLWJEAHALS	6	BQILOLXQLYLA	3	WAHYLYRPZSY	4	DGRCQRDLRDD	7
KTXMXKFBIBMT	3	CRJMPMYRMZMB	2	XBIZMZSQATZ	4	EHSDRSEMSEE	10*
LUYNYLGCJCNU	2	DSKNQNZSNANC	8*	YCJANATRBUA	6	FITESTFNTFF	7
MVZOZMHDKDOV	5	ETLOROATOBOD	10*	ZDKBOBUSCVB	3	GJUFTUGOUGG	2
NWAPANIELEPW	7	FUMPSPBUPCPE	2	AELCPCVTDWC	4	HKVGUVHPVHH	4
OXBQBOJFMFQX	2	GVNQTQCVQDQF	3	BFMDQDWUEXD	4	ILWHVWIQWII	5
PYCRCPKGNGRY	3	HWORURDWRERG	8*	CGNEREXVFYE	5	JMXIWXJRXJJ	2
QZDSDQLHOHSZ	7	IXPSVSEXSFSH	7	DHOFSFYWGZF	4	KNYJXYKSYKK	2
RAETERMIPITA	10*	JYQTWTFYTGTI	5	EIPGTGZXHAG	5	LOZKYZLTZLL	2
SBFUFSNJQJUB	3	KZRUXUGZUHUJ	2	FJQHUHAYIBH	5	MPALZAMUAMM	3
TCGVGTOKRKVC	4	LASVYVHAVIVK	5	GKRIVIBZJCI	4	NQBMABNVBNN	5

From this scheme we read the solution columnwise:

> Repeat the last order. Errors make it impossible to read.

Exercise. What was the encryption key used in this example?

Remark. Friedman in his Riverbank Publication No. 16 [7] uses the MLF method also for polyalphabetic ciphers with non-standard, but known, primary alphabets.

3.4 Recognizing Plaintext: Sinkov's Log-Weight Test

The MFL-test is simple and efficient. Sinkov in [25] proposed a more refined test that uses the information given by all single letter frequencies, not just by separating the letters into two classes. We won't explore the power of this method but treat it only as a motivation for Section 3.5.

As in Section 3.1 we assign a probability p_s to each letter s of the alphabet Σ. We enumerate the alphabet as $(s_1, \ldots, s_n)$ and write $p_i := p_{s_i}$. For a string $a = (a_1, \ldots, a_r) \in \Sigma^r$ we denote by $N_i(a) = \#\{j \mid a_j = s_i\}$ the multiplicity of the letter s_i in a. Then for an n-tuple $k = (k_1, \ldots, k_n) \in \mathbb{N}^n$ of natural numbers the probability for a string a to have multiplicities exactly given by k follows the multinomial distribution:

$$P(a \in \Sigma^r \mid N_i(a) = k_i \text{ for all } i = 1, \ldots, n) = \frac{r!}{k_1! \cdots k_n!} \cdot p_1^{k_1} \cdots p_n^{k_n}.$$

The Log-Weight (LW) Score

A heuristic derivation of the LW-score of a string $a \in \Sigma^r$ considers the "null hypothesis" (H_0): *a belongs to a given language with letter probabilities p_i*, and the "alternative hypothesis" (H_1): *a is a random string.* The probabilities for a having k as its set of multiplicities if (H_1) or (H_0) is true, are (in a somewhat sloppy notation)

$$P(k \mid \mathrm{H}_1) = \frac{r!}{k_1! \cdots k_n!} \cdot \frac{1}{n^r}, \quad P(k \mid \mathrm{H}_0) = \frac{r!}{k_1! \cdots k_n!} \cdot p_1^{k_1} \cdots p_n^{k_n}.$$

The quotient of these two values, the "likelihood ratio"

$$\lambda(k) = \frac{P(k \mid \mathrm{H}_0)}{P(k \mid \mathrm{H}_1)} = n^r \cdot p_1^{k_1} \cdots p_n^{k_n},$$

makes a good score for deciding between (H_0) and (H_1).

Usually one takes the reciprocal value, that is H_1 in the numerator, and H_0 in the denominator. We deviate from this convention because we want to have the score large for true texts and small for random texts.

For convenience one considers the logarithm (to any base) of this number:

$$\log \lambda(k) = r \log n + \sum_{i=1}^{n} k_i \cdot \log p_i.$$

(We assume all $p_i > 0$, otherwise we would omit s_i from our alphabet.) Noting that the summand $r \log n$ is the same for all $a \in \Sigma^r$ one considers

$$\log \lambda(k) - r \log n = \sum_{i=1}^{n} k_i \cdot \log p_i = \sum_{j=1}^{r} \log p_{a_j}.$$

Because $0 < p_i < 1$ the summands are negative. Adding a constant doesn't affect the use of this score, so finally we define SINKOV's **Log-Weight (LW) score** as

$$S_1(a) := \sum_{i=1}^{n} k_i \cdot \log(1000 \cdot p_i) = \sum_{j=1}^{r} \log(1000 \cdot p_{a_j}) = r \cdot \log 1000 + \sum_{j=1}^{r} \log p_{a_j}.$$

The numbers $\log(1000 \cdot p_i)$ are the "log weights". More frequent letters have higher weights. Table 3.6 gives the weights for the English alphabet with base-10 logarithms (so $\log 1000 = 3$). The MFL-method in contrast uses the weights 1 for `ETOANIRSHD`, and 0 else.

Note that the definition of the LW score doesn't depend on its heuristic motivation. Just take the weights given in Table 3.6 and use them for the definition of S_1.

Examples

We won't analyze the LW-method in detail, but rework the examples from Section 3.1. The LW scores for the CAESAR example are in Table 3.7.

The correct solution stands out clearly, the order of the non-solutions is somewhat permuted compared with the MFL score.

For the period-4 example the LW scores are in Tables 3.8 to 3.11. The method unambiguously picks the correct solution except for column 3 where the top score occurs twice.

In summary the examples show no clear advantage of the LW-method over the MFL-method, notwithstanding the higher granularity of the information used to compute the scores.

As for MFL scores we might define the LW rate as the quotient of the LW score be the length of the string. This makes the values for strings of different lengths comparable.

Table 3.6: *Log weights of the letters for English (base-10 logarithms)*

s	A	B	C	D	E	F	G
$1000p_s$	82	15	28	43	127	22	20
Log weight	1.9	1.2	1.4	1.6	2.1	1.3	1.3

s	H	I	J	K	L	M	N
$1000p_s$	61	70	2	8	40	24	67
Log weight	1.8	1.8	0.3	0.9	1.6	1.4	1.8

s	O	P	Q	R	S	T	U
$1000p_s$	75	19	1	60	63	91	28
Log weight	1.9	1.3	0.0	1.8	1.8	1.9	1.4

s	V	W	X	Y	Z		
$1000p_s$	10	23	1	20	1		
Log weight	1.0	1.4	0.0	1.3	0.0		

Table 3.7: *LW scores for the exhausion of a shift cipher*

```
FDHVDU   8.7     OMQEMD   8.4        XVZNVM   5.2
GEIWEV   9.7     PNRFNE  10.1 <---   YWAOWN   9.7
HFJXFW   6.1     QOSGOF   8.2        ZXBPXO   4.4
IGKYGX   6.6     RPTHPG   9.4        AYCQYP   7.2
JHLZHY   6.8     SQUIQH   6.8        BZDRZQ   4.6
KIMAIZ   7.8     TRVJRI   8.6        CAESAR  10.9 <===
LJNBJA   7.1     USWKSJ   7.6        DBFTBS   9.0
MKOCKB   7.7     VTXLTK   7.3        ECGUCT   9.5
NLPDLC   9.3     WUYMUL   8.5
```

Exercise. Give a more detailed analysis of the distribution of the LW scores for English and for random texts (with "English" weights). You may use the Perl script `LWscore.pl` in the directory `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Perl/`.

Table 3.12 gives log weights for German and French.

Table 3.8: *LW scores for column 1 of a period 4 cipher*

UDHWHUPLSLWD	18.7		DMQFQDYUBUFM	13.9		MVZOZMHDKDOV	14.5	
VEIXIVQMTMXE	14.5		ENRGREZVCVGN	17.4		NWAPANIELEPW	20.4	<--
WFJYJWRNUNYF	15.4		FOSHSFAWDWHO	19.9		OXBQBOJFMFQX	10.5	
XGKZKXSOVOZG	11.0		GPTITGBXEXIP	15.9		PYCRCPKGNGRY	16.9	
YHLALYTPWPAH	19.1		HQUJUHCYFYJQ	12.3		QZDSDQLHOHSZ	13.9	
ZIMBMZUQXQBI	10.2		IRVKVIDZGZKR	13.9		RAETERMIPITA	21.7	<==
AJNCNAVRYRCJ	16.7		JSWLWJEAHALS	17.9		SBFUFSNJQJUB	13.8	
BKODOBWSZSDK	16.2		KTXMXKFBIBMT	13.9		TCGVGTOKRKVC	16.7	
CLPEPCXTATEL	18.5		LUYNYLGCJCNU	16.6				

Table 3.9: *LW scores for column 2 of a period 4 cipher*

MBTWZWIBWJWL	15.0		VKCFIFRKFSFU	16.2		ETLOROATOBOD	21.6	<==
NCUXAXJCXKXM	10.5		WLDGJGSLGTGV	16.4		FUMPSPBUPCPE	17.2	
ODVYBYKDYLYN	16.8		XMEHKHTMHUHW	17.7		GVNQTQCVQDQF	11.3	
PEWZCZLEZMZO	13.2		YNFILIUNIVIX	17.4		HWORURDWRERG	20.1	<--
QFXADAMFANAP	16.3		ZOGJMJVOJWJY	11.4		IXPSVSEXSFSH	16.5	
RGYBEBNGBOBQ	16.3		APHKNKWPKXKZ	13.1		JYQTWTFYTGTI	16.3	
SHZCFCOHCPCR	17.3		BQILOLXQLYLA	14.5		KZRUXUGZUHUJ	11.7	
TIADGDPIDQDS	18.2		CRJMPMYRMZMB	14.7		LASVYVHAVIVK	17.0	
UJBEHEQJERET	17.1		DSKNQNZSNANC	16.6				

Table 3.10: *LW scores for column 3 of a period 4 cipher*

HLSJWJCAKDJ	13.3		QUBSFSLJTMS	14.5		ZDKBOBUSCVB	13.6	
IMTKXKDBLEK	14.3		RVCTGTMKUNT	16.7		AELCPCVTDWC	17.0	
JNULYLECMFL	15.8		SWDUHUNLVOU	17.1		BFMDQDWUEXD	13.6	
KOVMZMFDNGM	14.0		TXEVIVOMWPV	14.8		CGNEREXVFYE	16.2	
LPWNANGEOHN	18.7	<-	UYFWJWPNXQW	11.6		DHOFSFYWGZF	15.0	
MQXOBOHFPIO	14.5		VZGXKXQOYRX	8.2		EIPGTGZXHAG	14.7	
NRYPCPIGQJP	13.6		WAHYLYRPZSY	15.5		FJQHUHAYIBH	14.6	
OSZQDQJHRKQ	10.1		XBIZMZSQATZ	10.0		GKRIVIBZJCI	13.3	
PTARERKISLR	18.7	<-	YCJANATRBUA	16.8				

Table 3.11: *LW scores for column 4 of a period 4 cipher*

```
ORCNBCOWCOO 18.0    XALWKLXFLXX 10.3    GJUFTUGOUGG 14.8
PSDOCDPXDPP 15.1    YBMXLMYGMYY 13.5    HKVGUVHPVHH 15.1
QTEPDEQYEQQ 12.4    ZCNYMNZHNZZ 11.3    ILWHVWIQWII 15.8
RUFQEFRZFRR 14.6    ADOZNOAIOAA 18.5    JMXIWXJRXJJ  7.6
SVGRFGSAGSS 17.1    BEPAOPBJPBB 14.9    KNYJXYKSYKK 11.4
TWHSGHTBHTT 18.7 <- CFQBPQCKQCC 10.3    LOZKYZLTZLL 12.4
UXITHIUCIUU 16.1    DGRCQRDLRDD 16.1    MPALZAMUAMM 15.6
VYJUIJVDJVV 11.0    EHSDRSEMSEE 20.4 <= NQBMABNVBNN 15.1
WZKVJKWEKWW 11.7    FITESTFNTFF 18.4
```

Table 3.12: *Log weights of the letters for German and French (base-10 logarithms)*

s	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
German	1.8	1.3	1.4	1.7	2.2	1.2	1.5	1.6	1.9	0.5	1.2	1.5	1.4	2.0	1.5	1.0	0.0	1.9	1.8	1.8	1.6	1.0	1.2	0.0	0.0	1.0
French	1.9	1.0	1.5	1.6	2.2	1.1	1.0	0.8	1.8	0.5	0.0	1.8	1.4	1.9	1.7	1.4	1.0	1.8	1.9	1.9	1.8	1.2	0.0	0.6	0.3	0.0

3.5 Recognizing Plaintext: The Log-Weight Method for Bigrams

In the last four sections we used only the single letter frequencies of a natural language. In other words, we treated texts as sequences of independent letters. But a characteristic aspect of every natural language is how letters are combined as bigrams (letter pairs). We may hope to get good criteria for recognizing a language by evaluating the bigrams in a text. Of course this applies to contiguous text only, in particular it is useless for the polyalphabetic example of Sections 3.3 and 3.4.

In analogy with the LW score we define a **Bigram Log-Weight (BLW) score** for a string. Let p_{ij} be the probability (or average relative frequency) of the bigram $s_i s_j$ in the base language. Because these numbers are small we multiply them by 10000.

Tables containing these bigram frequencies for English, German, and French are in `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/8_Transpos/Bigrams.html`

In contrast to the single letter case we cannot avoid the case $p_{ij} = 0$: some letter pairs never occur as bigrams in a meaningful text. Therefore we count the frequencies k_{ij} of the bigrams $s_i s_j$ in a string $a \in \Sigma^r$, and define the BLW-score by the formula

$$S_2(a) := \sum_{i,j=1}^{n} k_{ij} \cdot w_{ij} \quad \text{where } w_{ij} = \begin{cases} \log(10000 \cdot p_{ij}) & \text{if } 10000 \cdot p_{ij} > 1, \\ 0 & \text{otherwise.} \end{cases}$$

Note. We implicitly set $\log 0 = 0$. This convention is not as strange as it may look at first sight: For $p_{ij} = 0$ we'll certainly have $k_{ij} = 0$, and setting $0 \cdot \log 0 = 0$ is widespread practice.

To calculate the BLW score we go through the bigrams $a_t a_{t+1}$ for $t = 1, \ldots, r-1$ and add the log weight $w_{ij} = \log(10000 \cdot p_{ij})$ of each bigram. This approach is somewhat naive because it implicitly considers the bigrams—even the overlapping ones!—as independent. This criticism doesn't mean that we are doing something mathematically wrong, but only that the usefulness of the score might be smaller than expected.

We prepare matrices for English, German, and French that contain the relative frequencies of the bigrams in the respective language. These are in the files `eng_rel.csv`, `ger_rel.csv`, `fra_rel.csv` in the directory `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Files/` as comma-separated tables. The corresponding bigram log-weights are in the files `eng_blw.csv`, `ger_blw.csv`, `fra_blw.csv`. Programs that compute BLW scores for English, German, or French are `BLWscE.pl`, `BLWscD.pl`, and `BLWscF.pl` in the Perl directory.

As an example we compute the scores for the CAESAR example, see Table 3.13. The correct solution is evident in all three languages.

Table 3.13: *BLW scores for the exhaustion of a* CAESAR *cipher*

```
BLW scores English    German      French
  FDHVDU    1.4        3.1         2.2
  GEIWEV    5.8 <---   7.3 <===    4.3
  HFJXFW    0.9        0.3         0.0
  IGKYGX    2.2        2.1         1.3
  JHLZHY    0.5        1.9         0.3
  KIMAIZ    5.9 <---   5.2         4.9
  LJNBJA    1.1        2.4         0.9
  MKOCKB    2.7        4.2         0.8
  NLPDLC    3.0        2.8         1.4
  OMQEMD    3.5        3.8         3.6
  PNRFNE    3.6        4.7         3.6
  QOSGOF    5.8 <---   4.0         3.4
  RPTHPG    4.5        2.6         2.7
  SQUIQH    2.3        0.6         6.3 <---
  TRVJRI    4.1        4.3         4.9
  USWKSJ    3.3        3.7         2.0
  VTXLTK    1.3        2.0         1.1
  WUYMUL    3.1        2.9         2.7
  XVZNVM    0.6        1.3         1.0
  YWAOWN    5.5        2.3         0.0
  ZXBPXO    0.0        0.0         0.0
  AYCQYP    3.2        0.0         0.3
  BZDRZQ    1.0        2.1         1.1
  CAESAR    7.7 <===   7.5 <===    8.4 <===
  DBFTBS    4.7        3.5         0.6
  ECGUCT    5.5        3.6         5.5
```

3.6 Empirical Results on BLW Scores

The heuristic motivation of the BLW score, like for all the scores in this chapter, relies on independence assumptions that are clearly violated by natural languages. Therefore again it makes sense to get empirical results by analyzing a large sample of concrete texts.

The empirical results for the 5%-level of the error of the first kind are as follows, see Appendix D.

English. We take the threshold value $T = 11$ for English texts. Then 86 of 2000 English scores are $\leq T$, the error of the first kind is $\alpha = 86/2000 = 4.2\%$. For random texts 1964 of 2000 scores are $\leq T$, the power is $1964/2000 = 99.5\%$. There are 36 random scores and 1914 English scores $> T$, the predictive value for English is $1914/1950 = 98.2\%$.

German. We take the threshold value $T = 12$ for German texts. Then 84 of 2000 German scores are $\leq T$, the error of the first kind is $\alpha = 84/2000 = 4.2\%$. For random texts 1991 of 2000 scores are $\leq T$, the power is $1991/2000 = 99.6\%$. There are 9 random scores and 1916 German scores $> T$, the predictive value for German is $1916/1925 = 99.5\%$.

French. We take the threshold value $T = 11$ for French texts. Then 58 of 2000 French scores are $\leq T$, the error of the first kind is $\alpha = 58/2000 = 2.9\%$. For random texts 1967 of 2000 scores are $\leq T$, the power is $1967/2000 = 98.3\%$. There are 33 random scores and 1942 French scores $> T$, the predictive value for French is $1942/1975 = 98.3\%$.

The BLW score is significantly stronger than the MFL score.

3.7 Coincidences of Two Texts

The first six sections of this chapter introduced efficient methods for recognizing plaintext in comparison with noise. These methods break down for encrypted texts because they ignore properties that remain invariant under encryption. One such invariant property—at least for monoalphabetic substitution—is the equality of two letters, no matter what the concrete value of these letters is.

This is the main idea that we work out in the next sections: Look for identical letters in one or more texts, or in other words, for coincidences.

Definition

Let Σ be a finite alphabet. Let $a = (a_0, \ldots, a_{r-1})$ and $b = (b_0, \ldots, b_{r-1}) \in \Sigma^r$ be two texts of the same length $r \geq 1$. Then

$$\kappa(a,b) := \frac{1}{r} \cdot \#\{j \mid a_j = b_j\} = \frac{1}{r} \cdot \sum_{j=0}^{r-1} \delta_{a_j b_j}$$

is called **coincidence index** of a and b (where $\delta =$ Kronecker symbol).

For each $r \in \mathbb{N}_1$ this defines a map

$$\kappa \colon \Sigma^r \times \Sigma^r \longrightarrow \mathbb{Q} \subseteq \mathbb{R}.$$

The scaling factor $\frac{1}{r}$ makes results for different lengths comparable.

A Perl program is in the Web: `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Perl/kappa.pl`.

Remarks

1. Always $0 \leq \kappa(a,b) \leq 1$.
2. $\kappa(a,b) = 1 \Longleftrightarrow a = b$.
3. By convention $\kappa(\emptyset,\emptyset) = 1$ (where $\emptyset$ denotes the empty string by abuse of notation).
4. Note that up to scaling the coincidence index is a converse of the Hamming distance that counts non-coincidences.

Example 1: Two English Texts

We compare the first four verses (text 1) of the poem “If ...” by Rudyard Kipling and the next four verses (text 2). (The lengths differ, so we crop the longer one.)

```
IFYOU CANKE EPYOU RHEAD WHENA LLABO UTYOU ARELO OSING THEIR
IFYOU CANMA KEONE HEAPO FALLY OURWI NNING SANDR ISKIT ONONE
||||| |||                                        |
SANDB LAMIN GITON YOUIF YOUCA NTRUS TYOUR SELFW HENAL LMEND
TURNO FPITC HANDT OSSAN DLOOS EANDS TARTA GAINA TYOUR BEGIN
                                  | |
OUBTY OUBUT MAKEA LLOWA NCEFO RTHEI RDOUB TINGT OOIFY OUCAN
NINGS ANDNE VERBR EATHE AWORD ABOUT YOURL OSSIF YOUCA NFORC
                                                 |

WAITA NDNOT BETIR EDBYW AITIN GORBE INGLI EDABO UTDON TDEAL
```

```
EYOUR HEART ANDNE RVEAN DSINE WTOSE RVEYO URTUR NLONG AFTER
          |                       |
INLIE SORBE INGHA TEDDO NTGIV EWAYT OHATI NGAND YETDO NTLOO
THEYA REGON EANDS OHOLD ONWHE NTHER EISNO THING INYOU EXCEP
                                             |
KTOOG OODNO RTALK TOOWI SEIFY OUCAN DREAM ANDNO TMAKE DREAM
TTHEW ILLWH ICHSA YSTOT HEMHO LDONI FYOUC ANTAL KWITH CROWD
 |                       |                ||           |
SYOUR MASTE RIFYO UCANT HINKA NDNOT MAKET HOUGH TSYOU RAIMI
SANDK EEPYO URVIR TUEOR WALKW ITHKI NGSNO RLOOS ETHEC OMMON
|                          |
FYOUC ANMEE TWITH TRIUM PHAND DISAS TERAN DTREA TTHOS ETWOI
TOUCH IFNEI THERF OESNO RLOVI NGFRI ENDSC ANHUR TYOUI FALLM
         |  |                                   |
MPOST ORSAS THESA MEIFY OUCAN BEART OHEAR THETR UTHYO UVESP
ENCOU NTWOR THYOU BUTNO NETOO MUCHI FYOUC ANFIL LTHEU NFORG
            ||                                   ||
OKENT WISTE DBYKN AVEST OMAKE ATRAP FORFO OLSOR WATCH THETH
IVING MINUT EWITH SIXTY SECON DSWOR THOFD ISTAN CERUN YOURS
   |   |                               |
INGSY OUGAV EYOUR LIFEF ORBRO KENAN DSTOO PANDB UILDE MUPWI
ISTHE EARTH ANDEV ERYTH INGTH ATSIN ITAND WHICH ISMOR EYOUL
|                                 |
THWOR NOUTT OOLS
LBEAM ANMYS ON
            |
```

In these texts of length 562 we find 35 coincidences, the coincidence index is $\frac{35}{562} = 0.0623$.

Invariance

The coincidence index of two texts is an invariant of polyalphabetic substitution (the keys being equal):

Proposition 1 (Invariance) *Let $f\colon \Sigma^* \longrightarrow \Sigma^*$ be a polyalphabetic encryption function. Then*

$$\kappa(f(a), f(b)) = \kappa(a, b)$$

for all $a, b \in \Sigma^*$ of the same length.

Note that Proposition 1 doesn't need any assumptions on periodicity or on relations between the alphabets used. It only assumes that the encryption function uses the same alphabets at the corresponding positions in the texts.

Mean Values

For a fixed $a \in \Sigma^r$ we determine the mean value of $\kappa(a,b)$ taken over all $b \in \Sigma^r$:

$$\begin{aligned}
\frac{1}{n^r} \cdot \sum_{b\in\Sigma^r} \kappa(a,b) &= \frac{1}{n^r} \cdot \sum_{b\in\Sigma^r} \left[\frac{1}{r} \cdot \sum_{j=0}^{r-1} \delta_{a_j b_j} \right] \\
&= \frac{1}{rn^r} \cdot \sum_{j=0}^{r-1} \underbrace{\left[\sum_{b\in\Sigma^r} \delta_{a_j b_j} \right]}_{n^{r-1}} \\
&= \frac{1}{rn^r} \cdot r \cdot n^{r-1} = \frac{1}{n},
\end{aligned}$$

because, if $b_j = a_j$ is fixed, there remain n^{r-1} possible values for b.

In an analogous way we determine the mean value of $\kappa(a, f_\sigma(b)$ for fixed $a, b \in \Sigma^r$ over all permutations $\sigma \in \mathcal{S}(\Sigma)$:

$$\begin{aligned}
\frac{1}{n!} \cdot \sum_{\sigma\in\mathcal{S}(\Sigma)} \kappa(a, f_\sigma(b)) &= \frac{1}{n!} \cdot \frac{1}{r} \sum_{\sigma\in\mathcal{S}(\Sigma)} \#\{j \mid \sigma b_j = a_j\} \\
&= \frac{1}{rn!} \cdot \#\{(j,\sigma) \mid \sigma b_j = a_j\} \\
&= \frac{1}{rn!} \cdot \sum_{j=0}^{r-1} \#\{\sigma \mid \sigma b_j = a_j\} \\
&= \frac{1}{rn!} \cdot r \cdot (n-1)! = \frac{1}{n},
\end{aligned}$$

because exactly $(n-1)!$ permutations map a_j to b_j.

Note that this conclusion also works for $a = b$.

This derivation shows:

Proposition 2 (i) *The mean value of $\kappa(a,b)$ over all texts $b \in \Sigma^*$ of equal length is $\frac{1}{n}$ for all $a \in \Sigma^*$.*

(ii) *The mean value of $\kappa(a,b)$ over all $a, b \in \Sigma^r$ is $\frac{1}{n}$ for all $r \in \mathbb{N}_1$.*

(iii) *The mean value of $\kappa(a, f_\sigma(b))$ over all monoalphabetic substitutions with $\sigma \in \mathcal{S}(\Sigma)$ is $\frac{1}{n}$ for each pair $a, b \in \Sigma^*$ of texts of equal length.*

(iv) *The mean value of $\kappa(f_\sigma(a), f_\tau(b))$ over all pairs of monoalphabetic substitutions, with $\sigma, \tau \in \mathcal{S}(\Sigma)$, is $\frac{1}{n}$ for each pair $a, b \in \Sigma^*$ of texts of equal length.*

Interpretation

- For a given text a and a "random" text b of the same length $\kappa(a,b) \approx \frac{1}{n}$.

- For "random" texts a and b of the same length $\kappa(a,b) \approx \frac{1}{n}$.
- For given texts a and b of the same length and a "random" monoalphabetic substitution f_σ we have $\kappa(a, f_\sigma(b)) \approx \frac{1}{n}$. This remark justifies treating a nontrivially monoalphabetically encrypted text as random with respect to κ and plaintexts.
- For given texts a and b of the same length and two "random" monoalphabetic substitutions f_σ, f_τ we have $\kappa(f_\sigma(a), f_\tau(b)) \approx \frac{1}{n}$.
- The same holds for "random" polyalphabetic substitutions because counting the coincidences is additive with respect to arbitrary decompositions of texts.

Values that significantly differ from these mean values are suspicious for the cryptanalyst, they could have a *non-random cause.* For more precise statements we should assess the variances (or standard deviations) or, more generally, the distribution of κ-values in certain "populations" of texts.

Variance

First fix $a \in \Sigma^r$ and vary b over all of Σ^r. Using the mean value $\frac{1}{n}$ we calculate the variance:

$$\begin{aligned} V_{\Sigma^r}(\kappa, a) &= \frac{1}{n^r} \cdot \sum_{b \in \Sigma^r} \kappa(a,b)^2 - \frac{1}{n^2} \\ &= \frac{1}{n^r} \cdot \sum_{b \in \Sigma^r} \left[\frac{1}{r} \cdot \sum_{j=0}^{r-1} \delta_{a_j b_j} \right]^2 - \frac{1}{n^2} \end{aligned}$$

Evaluating the square of the sum in brackets we get the quadratic terms

$$\sum_{j=0}^{r-1} \delta_{a_j b_j}^2 = \sum_{j=0}^{r-1} \delta_{a_j b_j} = r \cdot \kappa(a,b) \quad \text{because} \quad \delta_{a_j b_j} = 0 \text{ or } 1$$

$$\sum_{b \in \Sigma^r} \sum_{j=0}^{r-1} \delta_{a_j b_j}^2 = r \cdot \sum_{b \in \Sigma^r} \kappa(a,b) = r \cdot n^r \cdot \frac{1}{n} = r \cdot n^{r-1}$$

and the mixed terms

$$2 \cdot \sum_{j=0}^{r-1} \sum_{k=j+1}^{r-1} \delta_{a_j b_j} \delta_{a_k b_k} \quad \text{where} \quad \delta_{a_j b_j} \delta_{a_k b_k} = \begin{cases} 1 & \text{if } a_j = b_j \text{ and } a_k = b_k \\ 0 & \text{else} \end{cases}$$

If we fix two letters b_j and b_k, we are left with n^{r-2} different b's that give the value 1. The total sum over the mixed terms evaluates as

$$\sum_{b\in\Sigma^r}\left(2\cdot\sum_{j=0}^{r-1}\sum_{k=j+1}^{r-1}\delta_{a_jb_j}\delta_{a_kb_k}\right) = 2\cdot\sum_{j=0}^{r-1}\sum_{k=j+1}^{r-1}\underbrace{\sum_{b\in\Sigma^r}\delta_{a_jb_j}\delta_{a_kb_k}}_{n^{r-2}}$$

Substituting our intermediary results we get

$$\begin{aligned} V_{\Sigma^r}(\kappa,a) &= \frac{1}{n^r r^2}\left(r\cdot n^{r-1}+r\cdot(r-1)\cdot n^{r-2}\right)-\frac{1}{n^2}\\ &= \frac{1}{rn}+\frac{r-1}{rn^2}-\frac{1}{n^2}=\frac{1}{rn}-\frac{1}{rn^2}=\frac{1}{r}\left(\frac{1}{n}-\frac{1}{n^2}\right)\end{aligned}$$

Next we let a and b vary and calculate the variance of κ:

$$\begin{aligned} V_{\Sigma^r}(\kappa) &= \frac{1}{n^{2r}}\sum_{a,b\in\Sigma^r}\kappa(a,b)^2-\frac{1}{n^2}\\ &= \frac{1}{n^r}\sum_{a\in\Sigma^r}\underbrace{\left(\frac{1}{n^r}\sum_{b\in\Sigma^r}\kappa(a,b)^2\right)}_{\frac{1}{r}\left(\frac{1}{n}-\frac{1}{n^2}\right)+\frac{1}{n^2}}-\frac{1}{n^2}\\ &= \frac{1}{r}\left(\frac{1}{n}-\frac{1}{n^2}\right)+\frac{1}{n^2}-\frac{1}{n^2}=\frac{1}{r}\left(\frac{1}{n}-\frac{1}{n^2}\right)\end{aligned}$$

We have shown:

Proposition 3 (i) *The mean value of $\kappa(a,b)$ over all texts b of equal length $r\in\mathbb{N}_1$ is $\frac{1}{n}$ with variance $\frac{1}{r}\left(\frac{1}{n}-\frac{1}{n^2}\right)$ for all $a\in\Sigma^r$.*

(ii) *The mean value of $\kappa(a,b)$ over all $a,b\in\Sigma^r$ is $\frac{1}{n}$ with variance $\frac{1}{r}\left(\frac{1}{n}-\frac{1}{n^2}\right)$ for all $r\in\mathbb{N}_1$.*

For the 26 letter alphabet A...Z we have the mean value $\frac{1}{26}\approx 0.0385$, independently from the text length r. The variance is $\approx\frac{0.03370}{r}$, the standard deviation $\approx\frac{0.19231}{\sqrt{r}}$. From this we get the second row of Table 3.14.

Table 3.14: *Standard deviations and 95% quantiles of κ for random text pairs of length r*

r	10	40	100	400	1000	10000
Std dev	0.0608	0.0304	0.0192	0.0096	0.0061	0.0019
95% quantile	0.1385	0.0885	0.0700	0.0543	0.0485	0.0416

For statistical tests (one-sided in this case) we would like to know the 95% quantiles. If we take the values for a normal distribution as approximations, that is "mean value + 1.645 times standard deviation", we get the values in the third row of Table 3.14. These raw estimates show that the κ-statistic in this form is weak in distinguishing "meaningful" texts from random texts, even for text lengths of 100 letters, and strong only for texts of several thousand letters.

Distinguishing meaningful plaintext from random noise is evidently not the main application of the κ-statistic. The next section will show the true relevancy of the coincidence index.

3.8 Empirical Values for Natural Languages

Empirical Observations

Some additional explicit examples are on the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/3_Coincid/EmpNat.html` These examples show some tendencies that will be empirically or mathematically founded later in this section:

- The typical coincidence index of two German texts is about 0.08.
- The typical coincidence index of two English texts is about 0.06.
- The typical coincidence index of a German and an English text is about 0.06 to 0.07.
- The typical coincidence index of a plaintext and ciphertext is about 0.03 to 0.05, that is near the "random" value $\frac{1}{26} \approx 0.0385$. The same is true for two independent ciphertexts.
- *If the same key is used for two polyalphabetic ciphertexts this fact reveals itself by a coincidence index that resembles that of two plaintexts.*

This latter statement is the first application of coincidence counts. No matter whether the encryption is periodic or not—if we get several ciphertexts encrypted in the same way, we can arrange them in parallel rows ("in depth") and get monoalphabetically encrypted columns that eventually can be decrypted.

Historical Example

The Polish cryptanalyst REJEWSKI was the first who successfully broke early military versions of the German cipher machine Enigma, see Chapter 6. He detected that ciphertexts were "in phase" by coincidence counts. It is unknown whether he knew FRIEDMAN's approach, or whether he found it for himself. Friedman's early publications were not classified and published even in France.

For example Rejewski noted that the two ciphertexts

```
RFOWL DOCAI HWBGX EMPTO BTVGG INFGR OJVDD ZLUWS JURNK KTEHM
RFOWL DNWEL SCAPX OAZYB BYZRG GCJDX NGDFE MJUPI MJVPI TKELY
```

besides having the initial six letters identical also had a suspicious number of coincidences between the remaining 44 letters ($5/44 \approx 0.114$).

Exercise. How many coincidences among 44 letters would you expect for independently encrypted texts?

Rejewski assumed that the first six letters denoted a "message key" that was identical for the two messages, and from this, that the Enigma operators prefixed their messages by a six letter message key. (Later on he even detected that in fact they used a repeated three letter key.)

Source: F. L. Bauer: Mathematik besiegte in Polen die unvernünftig gebrauchte ENIGMA. Informatik Spektrum 1. Dezember 2005, 493–497.]

Empirical Results on the Kappa Distribution

We want to learn more about the distribution of coincidence indices $\kappa(a, b)$ for English texts (or text chunks) a and b. To this end we again perform experiments whose results are in Appendix D.

Applications

To test whether a text a belongs to a certain language we would take one (or maybe several) fixed texts of the language and would test a against them. Because the values for natural languages are quite similar this test would only make sense for testing against random. This test is much weaker then the MFL, LW and BLW tests.

Also adjusting the columns of a disk cipher could be tested this way: If two alphabets are relatively shifted, the corresponding columns behave like random texts with respect to each other. If the alphabets are properly adjusted, the columns represent meaningful texts encrypted by the same monoalphabetic substitution, therefore they belong to the same language and show the typical coincidence index—up to statistical noise. Note that we need quite long columns for this test to work in a sensible way!

In the following sections we'll see some better tests for these problems. The main application of the coincidence index in its pure form is detecting identically encrypted polyalphabetic ciphertexts. Moreover it is the basis of some refined methods.

3.9 Autoincidence of a Text

Introduction

For the cryptanalysis of periodic polyalphabetic ciphers the following construction is of special importance: Let $a \in \Sigma^*$, and let $a_{(q)}$ and $a_{(-q)}$ be the cyclic shifts of a by q positions to the right resp. to the left. That is

$$\begin{array}{lclllllllll}
a & = & a_0 & a_1 & a_2 & \dots & a_{q-1} & a_q & a_{q+1} & \dots & a_{r-1} \\
a_{(q)} & = & a_{r-q} & a_{r-q+1} & a_{r-q+2} & \dots & a_{r-1} & a_0 & a_1 & \dots & a_{r-q-1} \\
a_{(-q)} & = & a_q & a_{q+1} & a_{q+2} & \dots & a_{2q-1} & a_{2q} & a_{2q+1} & \dots & a_{q-1}
\end{array}$$

Clearly $\kappa(a, a_{(q)}) = \kappa(a, a_{(-q)})$.

Definition. For a text $a \in \Sigma^*$ and a natural number $q \in \mathbb{N}$ the number $\kappa_q(a) := \kappa(a, a_{(q)})$ is called the q-th **autocoincidence index** of a.

Note. This is not a common notation. Usually this concept is not given an explicit name.

Example. We shift a text by 6 positions to the right:

```
COINCIDENCESBETWEENTHETEXTANDTHESHIFTEDTEXT <-- original text
EDTEXTCOINCIDENCESBETWEENTHETEXTANDTHESHIFT <-- shifted by 6
           |  |      | |          |    | <-- 6 coincidences
```

Properties

The q-th autocoincidence index κ_q defines a map

$$\kappa_q \colon \Sigma^* \longrightarrow \mathbb{Q}.$$

Clearly $\kappa_q(a) = \kappa_{r-q}(a)$ for $a \in \Sigma^r$ and $0 < q < r$, and κ_0 is a constant map.

Application

Take a ciphertext c that is generated by a periodic polyalphabetic substitution. If we determine $\kappa_q(c)$, we encounter two different situations: In the general case q is not a multiple of the period l. Counting the coincidences we encounter letter pairs that come from independent monoalphabetic substitutions. By the results of Section 3.7 we expect an index $\kappa_q(c) \approx \frac{1}{n}$.

In the special case where $l|q$ however we encounter the situation

$$\begin{array}{llllll}
\sigma_0 a_0 & \sigma_1 a_1 & \dots & \sigma_0 a_q & \sigma_1 a_{q+1} & \dots \\
 & & & \sigma_0 a_0 & \sigma_1 a_1 & \dots
\end{array}$$

where the letters below each other come from the same monoalphabetic substitution. Therefore they coincide if and only if the corresponding plaintext letters coincide. Therefore we expect an index $\kappa_q(c)$ near the coincidence index κ_M that is typical for the plaintext language M.

More precisely for a polyalphabetic substitution f of period l, plaintext a, and ciphertext $c = f(a)$:

1. For l not a divisor of q or $r - q$ we expect $\kappa_q(c) \approx \frac{1}{n}$.
2. For $l | q$ and q small compared with r we expect $\kappa_q(c) \approx \kappa_q(a)$, and this value should be near the typical coincidence index κ_M.

This is the second application of coincidence counts, detecting the period of a polyalphabetic substitution by looking at the autocoincidence indices of the ciphertext. Compared with the search for repetitions after Kasiski this method also takes account of repetitions of length 1 or 2. In this way we make much more economical use of the traces that the period leaves in the ciphertext.

Example

We want to apply these considerations to the autocoincidence analysis of a polyalphabetic ciphertext using the Perl program `coinc.pl` from `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Perl/`. We start with the cryptogram that we already have solved in Chapter 2 by repetition analysis:

```
        00    05    10    15    20    25    30    35    40    45

0000    AOWBK NLRMG EAMYC ZSFJO IYYVS HYQPY KSONE MDUKE MVEMP JBBOA
0050    YUHCB HZPYW MOOKQ VZEAH RMVVP JOWHR JRMWK MHCMM OHFSE GOWZK
0100    IKCRV LAQDX MWRMH XGTHX MXNBY RTAHJ UALRA PCOBJ TCYJA BBMDU
0150    HCQNY NGKLA WYNRJ BRVRZ IDXTV LPUEL AIMIK MKAQT MVBCB WVYUX
0200    KQXYZ NFPGL CHOSO NTMCM JPMLR JIKPO RBSIA OZZZC YPOBJ ZNNJP
0250    UBKCO WAHOO JUWOB CLQAW CYTKM HFPGL KMGKH AHTYG VKBSK LRVOQ
0300    VOEQW EALTM HKOBN CMVKO BJUPA XFAVK NKJAB VKNXX IJVOP YWMWQ
0350    MZRFB UEVYU ZOORB SIAOV VLNUK EMVYY VMSNT UHIWZ WSYPG KAAIY
0400    NQKLZ ZZMGK OYXAO KJBZV LAQZQ AIRMV UKVJO CUKCW YEALJ ZCVKJ
0450    GJOVV WMVCO ZZZPY WMWQM ZUKRE IWIPX BAHZV NHJSJ ZNSXP YHRMG
0500    KUOMY PUELA IZAMC AEWOD QCHEW OAQZQ OETHG ZHAWU NRIAA QYKWX
0550    EJVUF UZSBL RNYDX QZMNY AONYT AUDXA WYHUH OBOYN QJFVH SVGZH
0600    RVOFQ JISVZ JGJME VEHGD XSVKF UKXMV LXQEO NWYNK VOMWV YUZON
0650    JUPAX FANYN VJPOR BSIAO XIYYA JETJT FQKUZ ZZMGK UOMYK IZGAW
0700    KNRJP AIOFU KFAHV MVXKD BMDUK XOMYN KVOXH YPYWM WQMZU EOYVZ
0750    FUJAB YMGDV BGVZJ WNCWY VMHZO MOYVU WKYLR MDJPV JOCUK QELKM
```

```
0800   AJBOS YXQMC AQTYA SABBY ZICOB XMZUK POOUM HEAUE WQUDX TVZCG
0850   JJMVP MHJAB VZSUM CAQTY AJPRV ZINUO NYLMQ KLVHS VUKCW YPAQJ
0900   ABVLM GKUOM YKIZG AVLZU VIJVZ OGJMO WVAKH CUEYN MXPBQ YZVJP
0950   QHYVG JBORB SIAOZ HYZUV PASMF UKFOW QKIZG ASMMK ZAUEW YNJAB
1000   VWEYK GNVRM VUAAQ XQHXK GVZHU VIJOY ZPJBB OOQPE OBLKM DVONV
1050   KNUJA BBMDU HCQNY PQJBA HZMIB HWVTH UGCTV ZDIKG OWAMV GKBBK
1100   KMEAB HQISG ODHZY UWOBR ZJAJE TJTFU K
```

The Autocoincidence Indices

This is the sequence of autocoincidence indices of our cryptogram

κ_1	κ_2	κ_3	κ_4	κ_5	κ_6	κ_7	κ_8
0.0301	0.0345	0.0469	0.0354	0.0371	0.0354	**0.0822**	0.0416
κ_9	κ_{10}	κ_{11}	κ_{12}	κ_{13}	κ_{14}	κ_{15}	κ_{16}
0.0265	0.0309	0.0416	0.0389	0.0327	**0.0787**	0.0460	0.0345
κ_{17}	κ_{18}	κ_{19}	κ_{20}	κ_{21}	κ_{22}	κ_{23}	κ_{24}
0.0460	0.0309	0.0327	0.0309	**0.0769**	0.0318	0.0309	0.0327
κ_{25}	κ_{26}	κ_{27}	κ_{28}	κ_{29}	κ_{30}	κ_{31}	κ_{32}
0.0318	0.0309	0.0416	**0.0875**	0.0477	0.0416	0.0442	0.0354
κ_{33}	κ_{34}	κ_{35}	κ_{36}				
0.0318	0.0389	**0.0610**	0.0371				

The period 7 stands out, as it did with the period analysis after KASISKI in the last chapter. This is also clearly seen in the graphical representation, see Figure 3.1.

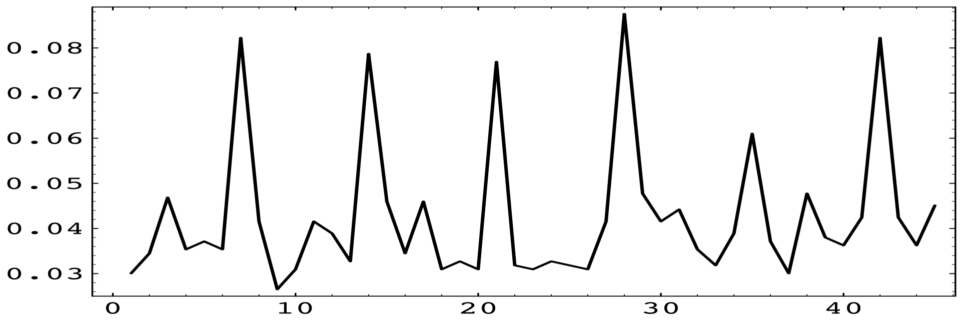


Figure 3.1: *Autocoincidence spectrum of a sample ciphertext*

The values other than at multiples of 7 fluctuate around the "random" value $\frac{1}{26} \approx 0.0385$ as expected. The values in the peaks fluctuate around the typical coincidence index near 0.08 of the plaintext language German, for which we gave empirical evidence in the last section. This effect has an easy explanation.

The Autocoincidence Spectrum

To analyze the effect seen in Figure 3.1, let c be the ciphertext from a polyalphabetic encryption of a text $a \in M$ with period l. What values can we expect for the $\kappa_q(c)$?

$$\begin{array}{rllll|llll}
c = & c_0 & \dots & c_{q-1} & & c_q & & \dots & c_{r-1} \\
c_{(q)} = & c_{r-q} & \dots & c_{r-1} & & c_0 & & \dots & c_{r-q-1} \\
\hline
\text{expected coinc.:} & q \cdot \kappa_M & \text{if} & l | r-q, & & (r-q) \cdot \kappa_M & \text{if} & l | q, & \\
 & q \cdot \kappa_{\Sigma^*} & \text{else} & & & (r-q) \cdot \kappa_{\Sigma^*} & \text{else} & &
\end{array}$$

Adding these up we get the following expected values for the autocoincidence spectrum:

1. case, $l | r$

$$\kappa_q(c) \approx \begin{cases} \frac{q \cdot \kappa_M + (r-q) \cdot \kappa_M}{r} = \kappa_M & \text{if } l | q, \\ \frac{q \cdot \kappa_{\Sigma^*} + (r-q) \cdot \kappa_{\Sigma^*}}{r} = \kappa_{\Sigma^*} & \text{else.} \end{cases}$$

2. case, $l \not| r$

$$\kappa_q(c) \approx \begin{cases} \frac{q \cdot \kappa_{\Sigma^*} + (r-q) \cdot \kappa_M}{r} & \text{if } l | q, \\ \frac{q \cdot \kappa_M + (r-q) \cdot \kappa_{\Sigma^*}}{r} & \text{if } l | r-q, \\ \kappa_{\Sigma^*} & \text{else.} \end{cases}$$

In particular for $q << r$

$$\kappa_q(c) \approx \begin{cases} \kappa_M & \text{if } l | q, \\ \kappa_{\Sigma^*} & \text{else.} \end{cases}$$

This explains the autocoincidence spectrum that we observed in the example. Typical autocoincidence spectra are shown in Figures 3.2 and 3.3.

Since in the second case the resulting image may be somewhat blurred, one could try to calculate autocoincidence indices not by shifting the text cyclically around but by simply cutting off the ends.

Definition. The sequence $(\kappa_1(a), \dots, \kappa_{r-1}(a))$ of autocoincidence indices of a text $a \in \Sigma^r$ of length r is called the **autocoincidence spectrum** of a.

Note. that this notation too is not common in the literature, but seems adequate for its evident cryptanalytical importance.

Exercise 1. Determine the autocoincidence spectrum of the ciphertext that you already broke by a KASISKI analysis. Create a graphical representation of it using graphic software of your choice.

Exercise 2. Cryptanalyze the ciphertext

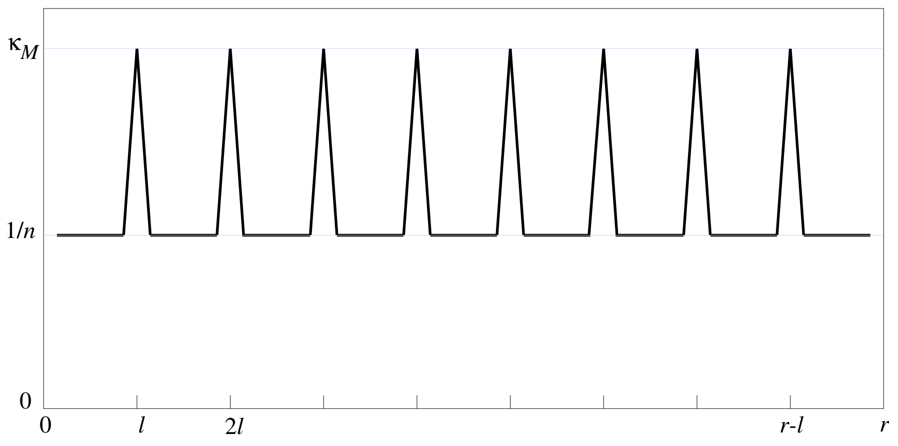


Figure 3.2: *Text length is multiple of period*

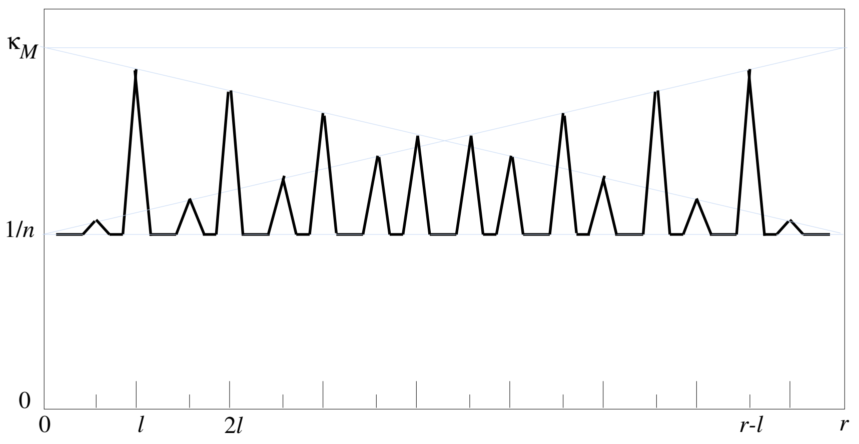


Figure 3.3: *Text length not multiple of period*

```
ECWUL MVKVR SCLKR IULXP FFXWL SMAEO HYKGA ANVGU GUDNP DBLCK
MYEKJ IMGJH CCUJL SMLGU TXWPN FQAPU EUKUP DBKQO VYTUJ IVWUJ
IYAFL OVAPG VGRYL JNWPK FHCGU TCUJK JYDGB UXWTT BHFKZ UFSWA
FLJGK MCUJR FCLCB DBKEO OUHRP DBVTP UNWPZ ECWUL OVAUZ FHNQY
XYYFL OUFFL SHCTP UCCWL TMWPB OXNKL SNWPZ IIXHP DBSWZ TYJFL
NUMHD JXWTZ QLMEO EYJOP SAWPL IGKQR PGEVL TXWPU AODGA ANZGY
BOKFH TMAEO FCFIH OTXCT PMWUO BOK
```

3.10 The Inner Coincidence Index of a Text

Definition

Let $a \in \Sigma^r$ $(r \geq 2)$ be a text, and $(\kappa_1(a), \ldots, \kappa_{r-1}(a))$ be its autocoincidence spectrum. Then the mean value

$$\varphi(a) := \frac{1}{r-1}\,[\kappa_1(a) + \cdots + \kappa_{r-1}(a)]$$

is called the **(inner) coincidence index** of a.

It defines a map

$$\varphi\colon \Sigma^{(\geq 2)} \longrightarrow \mathbb{Q}.$$

See the Perl program `phi.pl` from `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Perl/`.

Another description

Pick up the letters from two random positions of a text a. How many "twins" will you find? That means the same letter $s \in \Sigma$ at the two positions, or a "coincidence"?

Let $m_s = m_s(a) = \#\{j \mid a_j = s\}$ be the number of occurrences of s in a. Then the answer is

$$\frac{m_s \cdot (m_s - 1)}{2}$$

times. Therefore the total number of coincidences is

$$\sum_{s\in\Sigma} \frac{m_s \cdot (m_s-1)}{2} = \frac{1}{2} \cdot \sum_{s\in\Sigma} m_s^2 - \frac{1}{2} \cdot \sum_{s\in\Sigma} m_s = \frac{1}{2} \cdot \sum_{s\in\Sigma} m_s^2 - \frac{r}{2}$$

We count these coincidences in another way by the following algorithm: Let z_q be the number of already found coincidences with a distance of q for $q = 1, \ldots, r-1$, and initialize it as $z_q := 0$. Then execute the nested loops

```
for $i = 0, \ldots, r-2$                 [loop through the text $a$]
    for $j = i+1, \ldots, r-1$           [loop through the remaining text]
        if $a_i = a_j$                   [coincidence detected]
            increment $z_{j-i}$          [with distance $j - i$]
            increment $z_{r+i-j}$        [and with distance $r + i - j$]
```

After running through these loops the variables $z_1, \ldots, z_{r-1}$ have values such that

Lemma 2 (i) $z_1 + \cdots + z_{r-1} = \sum_{s\in\Sigma} m_s \cdot (m_s - 1)$.
(ii) $\kappa_q(a) = \frac{z_q}{r}$ *for* $q = 1, \ldots, r-1$.

Proof. (i) We count all coincidences twice.
(ii) $\kappa_q(a) = \frac{1}{r} \cdot \#\{j \,|\, a_{j+q} = a_j\}$ by definition (where the indices are taken $\bmod r$). $\diamond$

The Kappa-Phi Theorem

Theorem 3 (Kappa-Phi Theorem) *The inner coincidence index of a text* $a \in \Sigma^*$ *of length* $r \geq 2$ *is the proportion of coincidences among all letter pairs of* a*.*

Proof. The last term of the equation

$$\begin{aligned}\varphi(a) &= \frac{\kappa_1(a) + \cdots \kappa_{r-1}(a)}{r-1} = \frac{z_1 + \cdots + z_{r-1}}{r\cdot(r-1)} \\ &= \frac{\sum_{s\in\Sigma} m_s \cdot (m_s-1)}{r\cdot(r-1)} = \frac{\sum_{s\in\Sigma} \frac{m_s\cdot(m_s-1)}{2}}{\frac{r\cdot(r-1)}{2}}\end{aligned}$$

has the total number of coincidences in its numerator, and the total number of letter pairs in its denominator. $\diamond$

Corollary 1 *The inner coincidence index may be expressed as*

$$\varphi(a) = \frac{r}{r-1} \cdot \sum_{s\in\Sigma} \left(\frac{m_s}{r}\right)^2 - \frac{1}{r-1}$$

Proof. This follows via the intermediate step

$$\varphi(a) = \frac{\sum_{s\in\Sigma} m_s^2 - r}{r\cdot(r-1)}$$

$\diamond$

Note that this corollary provides a much faster algorithm for determining $\varphi(a)$. The definition formula needs $r-1$ runs through a text of length r, making $r\cdot(r-1)$ comparisons. The above algorithm reduces the costs to $\frac{r\cdot(r-1)}{2}$ comparisons. Using the formula of the corollary we need only one pass through the text, the complexity is linear in r. For a Perl program implementing this algorithm see the Perl script `coinc.pl` on the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Perl/`

Corollary 2 *The inner coincidence index of a text is invariant under* **mono***alphabetic substitution.*

Proof. The number of letter pairs is unchanged. ◇

3.11 The Distribution of the Inner Coincidence Index

First we calculate the exact mean value of the inner coincidence index $\varphi(a)$ for $a \in \Sigma^r$. Then we determine empirical values for mean value and variance for English, German, and random texts by simulation, as we did for κ.

The exact value of the variance leads to a somewhat more complicated calculation. We omit it.

Mean Value

We calculate the mean value of the letter frequencies $m_s(a)$ over $a \in \Sigma^r$ for each $s \in \Sigma$. Because of the symmetry in s all these values are identical, therefore we have

$$n \cdot \sum_{a\in\Sigma^r} m_s(a) = \sum_{s\in\Sigma}\sum_{a\in\Sigma^r} m_s(a) = \sum_{a\in\Sigma^r}\underbrace{\sum_{s\in\Sigma} m_s(a)}_{r} = r \cdot n^r$$

This gives the mean value

$$\frac{1}{n^r}\sum_{a\in\Sigma^r} m_s(a) = \frac{r}{n}$$

for each letter $s \in \Sigma$.

Next we calculate the mean value of $\kappa_q(a)$ over $a \in \Sigma^r$. We treat the indices of the letters of the texts a as elements of the cyclic additive group $\mathbb{Z}/n\mathbb{Z}$. Then we have

$$\begin{aligned}
\sum_{a\in\Sigma^r} \kappa_q(a) &= \sum_{a\in\Sigma^r} \frac{1}{r}\#\{j \in \mathbb{Z}/n\mathbb{Z} \mid a_{j+q} = a_j\} \\
&= \frac{1}{r}\sum_{j\in\mathbb{Z}/n\mathbb{Z}}\sum_{a\in\Sigma^r} \delta_{a_{j+q},a_j} \\
&= \frac{1}{r}\sum_{j\in\mathbb{Z}/n\mathbb{Z}} \underbrace{\#\{a \in \Sigma^r \mid a_{j+q} = a_j\}}_{n^{r-1}} \\
&= n^{r-1}
\end{aligned}$$

because in the underbraced count for a we may choose $r-1$ letters freely, and then the remaining letter is fixed. This gives the mean value

$$\frac{1}{n^r}\sum_{a\in\Sigma^r}\kappa_q(a)=\frac{1}{n}$$

for each $q=1,\ldots,r-1$.

Now for φ. We use the additivity of the mean value.

$$\begin{aligned}\frac{1}{n^r}\sum_{a\in\Sigma^r}\varphi(a) &= \frac{1}{r-1}\left[\frac{1}{n^r}\sum_{a\in\Sigma^r}\kappa_1(a)+\cdots+\frac{1}{n^r}\sum_{a\in\Sigma^r}\kappa_{r-1}(a)\right]\\ &= \frac{1}{r-1}\cdot(r-1)\cdot\frac{1}{n}=\frac{1}{n}\end{aligned}$$

We have shown:

Proposition 4 *The mean values of the q-th autocoincidence index for $q=1,\ldots,r-1$ and of the inner coincidence index over $a\in\Sigma^r$ each are $\frac{1}{n}$.*

And for the letter frequencies we have:

Corollary 3 *The sum of the letter frequencies $m_s(a)$ over $a\in\Sigma^r$ is*

$$\sum_{a\in\Sigma^r}m_s(a)=r\cdot n^{r-1}$$

for all letters $s\in\Sigma$.

Corollary 4 *The sum of the squares $m_s(a)^2$ of the letter frequencies over $a\in\Sigma^r$ is*

$$\sum_{a\in\Sigma^r}m_s(a)^2=r\cdot(n+r-1)\cdot n^{r-2}$$

for all letters $s\in\Sigma$.

Proof. By the Kappa-Phi Theorem we have

$$\sum_{t\in\Sigma}\left[\sum_{a\in\Sigma^r}m_s(a)^2-\sum_{a\in\Sigma^r}m_s(a)\right]=r\cdot(r-1)\cdot\sum_{a\in\Sigma^r}\varphi(a)=r\cdot(r-1)\cdot n^{r-1}$$

Substituting the result of the previous corollary and using the symmetry of the sum of squares with respect to s we get

$$n\cdot\sum_{a\in\Sigma^r}m_s(a)^2=\sum_{t\in\Sigma}\sum_{a\in\Sigma^r}m_s(a)^2=r\cdot(r-1)\cdot n^{r-1}+rn\cdot n^{r-1}=r\cdot n^{r-1}\cdot(r-1+n)$$

Dividing by n we get the above formula. $\diamond$

Empirical Results

For empirical results on the distribution of the inner coincidence index φ for English, German, and random texts we again refer to Appendix D.

Applications

To which questions from the introduction do these results apply?

We can decide whether a text is from a certain language. This includes texts that are monoalphabetically encrypted because φ is invariant under monoalphabetic substitution. And *we can recognize a monoalphabetically encrypted ciphertext.*

For both of these decision problems we calculate the coincidence index $\varphi(a)$ of our text a and decide "belongs to language" or "is monoalphabetically encrypted"—depending on our hypothesis—if $\varphi(a)$ reaches or surpasses the 95% quantile of φ for random texts of the same length—if we are willing to accept an error rate of the first kind of 5%.

For a text of 100 letters the threshold for φ is about 0.0451 by Table D.12. Tables D.10 and D.11 show that English or German texts surpass this threshold with high probability: For both languages the test has a power of nearly 100%.

It makes sense to work with the more ambitious "significance level" of 1% = bound for the error of the first kind. For this we set the threshold to the 99% quantile of the φ distribution for random texts. Our experiment for texts of length 100 gives the empirical value of 0.0473, failing the empirical minimum for our 2000 English 100 letter texts, and sitting far below the empirical minimum for German. Therefore even at the 1%-level the test has a power of nearly 100%.

The Phi Distribution for 26 Letter Texts

Since the φ test performs so excellently for 100 letter texts we dare to look at 26 letter texts—a text length that occurs in the Meet-in-the-Middle attack against rotor machines.

The results are in Appendix D.

The decision threshold on the 5%-level is 0.0585. For English texts the test has a power of only 50%, for German, near 75%. So we have a method to recognize monoalphabetic ciphertext that works fairly well for texts as short as 26 letters.

3.12 Sinkov's Formula

Let's apply the approximative formulas for $\kappa_q(c)$ from Section 3.9 to the coincidence index of a periodically polyalphabetically encrypted text $c =$

$f(a)$ with $a \in M$ of length r. In the case $l|r$ we get:

$$\begin{aligned}
\varphi(c) &= \frac{1}{r-1} \cdot [\kappa_1(c) + \cdots + \kappa_{r-1}(c)] \\
&\approx \frac{1}{r-1} \cdot \left[(\frac{r}{l} - 1) \cdot \kappa_M + (r - \frac{r}{l}) \cdot \kappa_{\Sigma^*} \right] \\
&= \frac{r-l}{r-1} \cdot \frac{1}{l} \cdot \kappa_M + \frac{r(l-1)}{l(r-1)} \cdot \kappa_{\Sigma^*} \\
&\approx \frac{1}{l} \cdot \kappa_M + \frac{l-1}{l} \cdot \kappa_{\Sigma^*},
\end{aligned}$$

since $\frac{r}{l} - 1$ summands scatter around κ_M, the other $r - \frac{r}{l}$ ones around κ_{Σ^*}.

In the same way for $l \not| r$ we get:

$$\begin{aligned}
\varphi(c) \approx\ & \frac{1}{r-1} \cdot \left[\frac{r-1}{l} \cdot \frac{q \cdot \kappa_{\Sigma^*} + (r-q) \cdot \kappa_M}{r} \right. \\
& \left. + \frac{r-1}{l} \cdot \frac{q \cdot \kappa_M + (r-q) \cdot \kappa_{\Sigma^*}}{r} + (r-1) \cdot (1 - \frac{2}{l}) \cdot \kappa_{\Sigma^*} \right] \\
=\ & \frac{1}{l} \cdot \frac{r \cdot \kappa_{\Sigma^*} + r \cdot \kappa_M}{r} + (1 - \frac{2}{l}) \cdot \kappa_{\Sigma^*} \\
=\ & \frac{1}{l} \cdot \kappa_M + \frac{l-1}{l} \cdot \kappa_{\Sigma^*},
\end{aligned}$$

that is the same approximative formula in both cases. Note that this is a weighted mean.

$$\boxed{\varphi(c) \approx \frac{1}{l} \cdot \kappa_M + \frac{l-1}{l} \cdot \kappa_{\Sigma^*}}$$

For the example $M =$ "German" and $l = 7$ we therefore expect

$$\varphi(c) \approx \frac{1}{7} \cdot 0.0762 + \frac{6}{7} \cdot 0.0385 \approx 0.0439,$$

and this is in accordance with the empirical value from the former example. In general Table 3.15 and Figure 3.4 show the connection between period and expected coincidence index for a polyalphabetically encrypted German text. The situation for English is even worse.

If we solve the above formula for the period length l, we get SINKOV's formula:

$$l \cdot \varphi(c) \approx \kappa_M + (l-1) \cdot \kappa_{\Sigma^*},$$

$$l \cdot [\varphi(c) - \kappa_{\Sigma^*}] \approx \kappa_M - \kappa_{\Sigma^*},$$

$$\boxed{l \approx \frac{\kappa_M - \kappa_{\Sigma^*}}{\varphi(c) - \kappa_{\Sigma^*}}.}$$

Table 3.15: *Coincidence index and period length (for German)*

period	1	2	3	4	5
Coinc. index	0.0762	0.0574	0.0511	0.0479	0.0460
	6	7	8	9	10
	0.0448	0.0439	0.0432	0.0427	0.0423

period	10	20	30	40	50
Coinc index	0.0423	0.0404	0.0398	0.0394	0.0393
	60	70	80	90	100
	0.0391	0.0390	0.0390	0.0389	0.0389

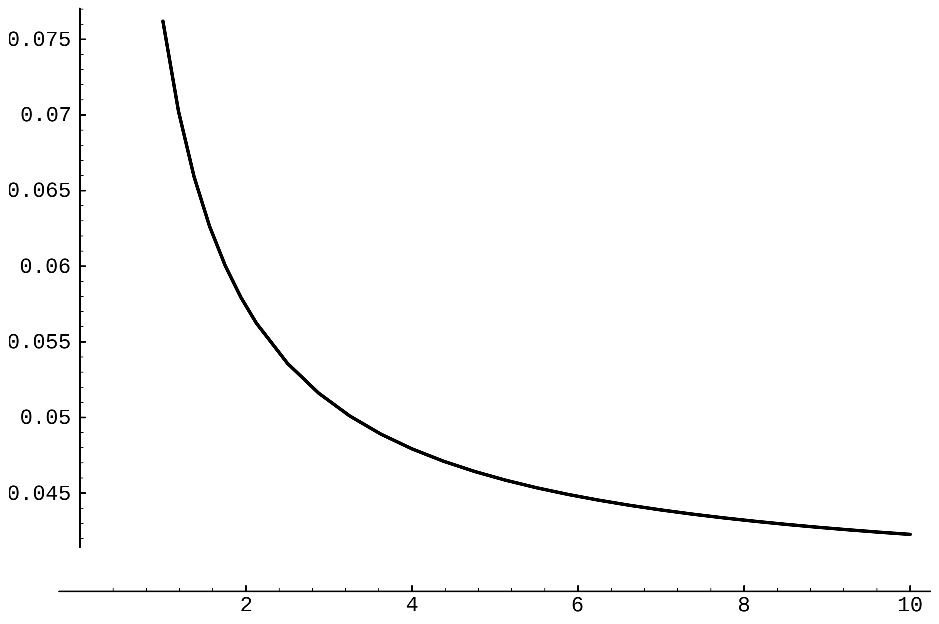


Figure 3.4: *Coincidence index and period length (for German)*

Remark. There are “more exact” versions of this formula. But these don’t give better results due to the variation of $\varphi(c)$ and the numerical instability of the small denominator.

For our sample cryptanalysis we get

$$l \approx \frac{0.0762 - 0.0385}{0.0440 - 0.0385} \approx 6.85.$$

This is also evidence for 7 being the length of the period.

The problem with SINKOV’s formula is the lack of numerical stability: the larger the period, the closer the coincidence index is to the value for random texts, as the table shows, that is, the closer the denominator in the formula is to 0.

Therefore the autocoincidence spectrum usually yields a better guess of the period. In fact SINKOV himself in his book [25] uses “his” formula—or rather the English equivalents of Table 3.15 and Figure 3.4—only for distinguishing between monoalphabetic and polyalphabetic ciphertexts. For determining the period he gives a very powerful test, see Section 3.13.

3.13 SINKOV’s Test for the Period

We want to test a pretended period l whether it is the real period. We write the text in rows of width l and consider the columns.

- If l is the correct period, each column is monoalphabetically encrypted and has its coincidence index near the coincidence index of the plaintext language.
- Otherwise the columns are random garbage and have coincidence indices near the random value $\frac{1}{n}$. Or rather near the value for a polyalphabetic ciphertext of period (the true) l.

Maybe the columns are quite short, thus their coincidence indices are diffuse and give no clear impression. However we can put all the indices together without bothering about the different monoalphabets, and get a much more precise value, based on all the letters of the text.

Definition For a text $a \in \Sigma^*$ and $l \in \mathbb{N}_1$ the mean value

$$\bar{\varphi}_l(a) := \frac{1}{l} \cdot \sum_{i=0}^{l-1} \varphi(a_i a_{i+l} a_{i+2l} \dots)$$

is called the SINKOV **statistic** of a of order l.

Note that $\bar{\varphi}_1 = \varphi$.

A Perl program, `phibar.pl`, is in `http://www.staff.uni-mainz.de/Cryptology/Classic/Perl/`.

Example

Let us again examine the ciphertext from Section 3.9. We get the values:

$\bar{\varphi}_1(a)$	0.0442	$\bar{\varphi}_7(a)$	**0.0829**	$\bar{\varphi}_{13}(a)$	0.0444
$\bar{\varphi}_2(a)$	0.0439	$\bar{\varphi}_8(a)$	0.0443	$\bar{\varphi}_{14}(a)$	**0.0839**
$\bar{\varphi}_3(a)$	0.0440	$\bar{\varphi}_9(a)$	0.0427	$\bar{\varphi}_{15}(a)$	0.0432
$\bar{\varphi}_4(a)$	0.0438	$\bar{\varphi}_{10}(a)$	0.0421	$\bar{\varphi}_{16}(a)$	0.0439
$\bar{\varphi}_5(a)$	0.0430	$\bar{\varphi}_{11}(a)$	0.0426	$\bar{\varphi}_{17}(a)$	0.0444
$\bar{\varphi}_6(a)$	0.0435	$\bar{\varphi}_{12}(a)$	0.0432	$\bar{\varphi}_{18}(a)$	0.0419

The period 7 is overwhelmingly evident. The values other than at the multiples of 7 are in almost perfect compliance with a (German) ciphertext of period around 7.

A Short Ciphertext

Our example ciphertext was quite long, and it is no surprise that the statistical methods perform very well. To get a more realistic picture let us examine the following ciphertext of length 148:

```
MDJJL DSKQB GYMZC YKBYT ZVRYU PJTZN WPZXS KCHFG EFYFS ENVFW
KORMX ZQGYT KEDIQ WRVPM OYMQV DQWDN UBQQM XEQCA CXYLP VUOSG
EJYDS PYYNA XOREC YJAFA MFCOF DQKTA CBAHW FYJUI LXBYA DTT
```

The Kasiski test finds no reptitions of length 3 or more. It finds 16 repetitions of length 2 and no eye-catching pattern. The common factors 10 or 20 could be a hint at the correct period, but repetitions of length 2 are not overly convincing.

Repetition:	DS	SK	GY	YM	CY	BY	YT	TZ
Distance:	98	28	47	60	100	125	40	8
Repetition:	GE	FY	OR	MX	QW	DQ	AC	YJ
Distance:	60	94	60	31	12	50	40	21

The coincidence index of the text is 0.0386 and doesn't distinguish the ciphertext from random text. The first 40 values of the autocoincidence spectrum are

κ_1	κ_2	κ_3	κ_4	κ_5	κ_6	κ_7	κ_8
0.0270	0.0203	0.0541	0.0405	0.0405	0.0338	0.0405	**0.0676**
κ_9	κ_{10}	κ_{11}	κ_{12}	κ_{13}	κ_{14}	κ_{15}	κ_{16}
0.0270	0.0473	0.0270	**0.0676**	0.0405	0.0473	0.0541	0.0541
κ_{17}	κ_{18}	κ_{19}	κ_{20}	κ_{21}	κ_{22}	κ_{23}	κ_{24}
0.0203	0.0203	**0.0608**	0.0473	0.0473	0.0135	0.0541	0.0270
κ_{25}	κ_{26}	κ_{27}	κ_{28}	κ_{29}	κ_{30}	κ_{31}	κ_{32}
0.0338	0.0405	0.0541	**0.0811**	0.0338	0.0338	0.0405	0.0203
κ_{33}	κ_{34}	κ_{35}	κ_{36}	κ_{37}	κ_{38}	κ_{39}	κ_{40}
0.0068	0.0473	0.0473	0.0270	0.0405	0.0066	0.0203	0.0473

Values above 0.06 occur for shifts of 8, 12, 19, 28, the latter being the largest one. This makes a diffuse picture, giving slight evidence for a period of 28. Finally let's try SINKOV's test. It gives as its first 40 values:

$\bar{\varphi}_1$	$\bar{\varphi}_2$	$\bar{\varphi}_3$	$\bar{\varphi}_4$	$\bar{\varphi}_5$	$\bar{\varphi}_6$	$\bar{\varphi}_7$	$\bar{\varphi}_8$
0.0386	0.0413	0.0386	0.0492	0.0421	0.0441	0.0433	0.0471
$\bar{\varphi}_9$	$\bar{\varphi}_{10}$	$\bar{\varphi}_{11}$	$\bar{\varphi}_{12}$	$\bar{\varphi}_{13}$	$\bar{\varphi}_{14}$	$\bar{\varphi}_{15}$	$\bar{\varphi}_{16}$
0.0330	0.0505	0.0265	**0.0591**	0.0333	0.0486	0.0444	0.0410
$\bar{\varphi}_{17}$	$\bar{\varphi}_{18}$	$\bar{\varphi}_{19}$	$\bar{\varphi}_{20}$	$\bar{\varphi}_{21}$	$\bar{\varphi}_{22}$	$\bar{\varphi}_{23}$	$\bar{\varphi}_{24}$
0.0280	0.0395	0.0439	**0.0589**	0.0357	0.0264	0.0476	0.0548
$\bar{\varphi}_{25}$	$\bar{\varphi}_{26}$	$\bar{\varphi}_{27}$	$\bar{\varphi}_{28}$	$\bar{\varphi}_{29}$	$\bar{\varphi}_{30}$	$\bar{\varphi}_{31}$	$\bar{\varphi}_{32}$
0.0507	0.0359	0.0444	0.0488	0.0368	**0.0622**	0.0312	0.0323
$\bar{\varphi}_{33}$	$\bar{\varphi}_{34}$	$\bar{\varphi}_{35}$	$\bar{\varphi}_{36}$	$\bar{\varphi}_{37}$	$\bar{\varphi}_{38}$	$\bar{\varphi}_{39}$	$\bar{\varphi}_{40}$
0.0091	0.0294	0.0429	**0.0611**	0.0541	0.0307	0.0256	0.0542

The values for 12, 20, 30, and 36 stand somewhat out, followed by the values for 24, 37, and 40, then 10 and 25—again there is no clear favorite. Let's discuss the candidate values for the period and rate each criterion as "good", "weak", or "prohibitive".

Period?	Pros and cons
8	$\varphi(c)$ should be slightly larger (weak). Only 3 repetition distances are multiples of 8 (weak). κ_8 and κ_{16} are good, κ_{40} is weak, κ_{24} and κ_{32} are prohibitive. $\bar{\varphi}_8$ is weak, $\bar{\varphi}_{16}$ and $\bar{\varphi}_{32}$ are prohibitive, $\bar{\varphi}_{24}$ and $\bar{\varphi}_{40}$ are good.
10	$\varphi(c)$ should be slightly larger (weak). 7 repetition distances are multiples of 10 (good). κ_{10}, κ_{20}, and κ_{40} are weak, κ_{30} is prohibitive. $\bar{\varphi}_{10}$, $\bar{\varphi}_{20}$, $\bar{\varphi}_{30}$, and $\bar{\varphi}_{40}$ are good.
12	$\varphi(c)$ should be slightly larger (weak). 4 repetition distances are multiples of 12 (good). κ_{12} is good, κ_{24} and κ_{36} are prohibitive. $\bar{\varphi}_{12}$, $\bar{\varphi}_{24}$, and $\bar{\varphi}_{36}$ are good.
19	0 repetition distances are multiples of 19 (prohibitive). κ_{19} is good, κ_{38} is prohibitive. $\bar{\varphi}_{19}$ and $\bar{\varphi}_{38}$ are prohibitive.
20	6 repetition distances are multiples of 20 (good). κ_{20} and κ_{40} are weak. $\bar{\varphi}_{20}$ and $\bar{\varphi}_{40}$ are good.
24	0 repetition distances are multiples of 24 (prohibitive). κ_{24} is prohibitive. $\bar{\varphi}_{24}$ is good.
28	Only 1 repetition distance is a multiple of 28 (weak). κ_{28} is good. $\bar{\varphi}_{28}$ is weak.
30	3 repetition distances are multiples of 30 (good). κ_{30} is prohibitive. $\bar{\varphi}_{30}$ is good.
36	0 repetition distances are multiples of 36 (prohibitive). κ_{36} is prohibitive. $\bar{\varphi}_{36}$ is good.
37	0 repetition distances are multiples of 37 (prohibitive). κ_{37} is prohibitive. $\bar{\varphi}_{37}$ is good.

To assess these findings let us score the values “good” as +1, “weak” as 0, and “prohibitive” as −1. Note that 3 repetitions for period 8 are weaker than 3 repetitions for period 30. The candidates 19, 24, 36, and 37 have negative weights, the candidates 8 and 28, zero weights. We skip them in the first round. Positive weights have 10 (3 of 9), 12 (3 of 8), 20 (3 of 5), and 30 (1 of 3). We rank them by their relative weights: 20 with score $0.6 = 3/5$, then 12 with score 0.375, then 10 and 30 with scores 0.333.

The most promising approach to further cryptanalysis starts from the hypothetical period 20, see Section 3.15.

3.14 Kullback's Cross-Product Sum Statistic

For a decision whether two texts $a \in \Sigma^r$, $b \in \Sigma^q$ belong to the same language we could consider $\varphi(a||b)$, the coincidence index of the concatenated string $a||b$. It should approximately equal the coincidence index of the language, or—in the negative case—be significantly smaller. This index evaluates as

$$(q+r)(q+r-1)\cdot\varphi(a||b) = \sum_{s\in\Sigma}[m_s(a)+m_s(b)]\,[m_s(a)+m_s(b)-1]$$

$$= \sum_{s\in\Sigma} m_s(a)^2 + \sum_{s\in\Sigma} m_s(b)^2 + 2\cdot\sum_{s\in\Sigma} m_s(a)m_s(b) - r - q$$

In this expression we consider terms depending on only one of the texts as irrelevant for the decision problem. Omitting them we are left with the "cross-product sum"

$$\sum_{s\in\Sigma} m_s(a)m_s(b)$$

From another viewpoint we could consider the "Euclidean distance" of a and b in the n-dimensional space of single letter frequencies

$$d(a,b) = \sum_{s\in\Sigma}[m_s(a)-m_s(b)]^2 = \sum_{s\in\Sigma} m_s(a)^2 + \sum_{s\in\Sigma} m_s(b)^2 - 2\cdot\sum_{s\in\Sigma} m_s(a)m_s(b)$$

and this also motivates considering the cross-product sum. It should be large for texts from the same language, and small otherwise.

Definition

Let Σ be a finite alphabet. Let $a \in \Sigma^r$ and $b \in \Sigma^q$ be two texts of lengths $r, q \geq 1$. Then

$$\chi(a,b) := \frac{1}{rq}\cdot\sum_{s\in\Sigma} m_s(a)m_s(b),$$

where m_s denotes the frequency of the letter s in a text, is called **cross-product sum** of a and b.

For each pair $r, q \in \mathbb{N}_1$ this defines a map

$$\chi\colon \Sigma^r \times \Sigma^q \longrightarrow \mathbb{Q}.$$

A Perl program, `chi.pl`, is in `http://www.staff.uni-mainz.de/Cryptology/Classic/Perl/`.

Transforming a and b by the same monoalphabetic substitution permutes the summands of $\chi(a,b)$. Therefore χ is invariant under monoalphabetic substitution.

Lemma 3 *Always $\chi(a,b) \leq 1$. Equality holds if and only if a and b consist of repetitions of the same single letter.*

Proof. We use the CAUCHY-SCHWARTZ inequality:

$$\begin{aligned}\chi(a,b)^2 = \left(\sum_{s\in\Sigma} \frac{m_s(a)}{r}\frac{m_s(b)}{q}\right)^2 \quad &\leq \quad \sum_{s\in\Sigma}\left(\frac{m_s(a)}{r}\right)^2 \cdot \sum_{s\in\Sigma}\left(\frac{m_s(b)}{q}\right)^2 \\ &\leq \quad \sum_{s\in\Sigma}\frac{m_s(a)}{r} \cdot \sum_{s\in\Sigma}\frac{m_s(b)}{q} = 1\end{aligned}$$

Equality holds if and only if

- $m_s(a) = c \cdot m_s(b)$ for all $s \in \Sigma$ with a fixed $c \in \mathbb{R}$,
- *and* all $\frac{m_s(a)}{r}$ and $\frac{m_s(b)}{q}$ are 0 or 1.

These two conditions together are equivalent with both of a and b consisting of only one—the same—repeated letter. $\diamond$

Considering the quantity $\psi(a) := \chi(a,a) = \sum_s m_s(a)^2/r^2$ doesn't make much sense for Corollary 1 of the Kappa-Phi-Theorem gives a linear (more exactly: affine) relation between ψ and φ:

Lemma 4 *For all $a \in \Sigma^r$, $r \geq 2$,*

$$\varphi(a) = \frac{r}{r-1} \cdot \psi(a) - \frac{1}{r-1}$$

Side Remark: COHEN's Kappa

In statistical texts one often encounters a related measure of coincidence between two series of observations: COHEN's kappa. It combines FRIEDMAN's kappa and KULLBACK's chi. Let $a = (a_1, \ldots, a_r)$, $b = (b_1, \ldots, b_r) \in \Sigma^r$ be two texts over the alphabet Σ (or two series of observations of data of some type). Then consider the matrix of frequencies

$$m_{st}(a,b) = \#\{i \mid a_i = s, b_i = t\} \quad \text{for } s,t \in \Sigma.$$

Its row sums are

$$m_s(a) = \#\{i \mid a_i = s\} = \sum_{t\in\Sigma} m_{st}(a,b),$$

its column sums are

$$m_t(b) = \#\{i \mid b_i = t\} = \sum_{s\in\Sigma} m_{st}(a,b),$$

its diagonal sum is

$$\sum_{s\in\Sigma} m_{ss}(a,b) = \sum_{s\in\Sigma} \#\{i \mid a_i = b_i = s\} = \#\{i \mid a_i = b_i\}.$$

The intermediate values from which COHEN's kappa is calculated are

$$p_0 = \frac{1}{r} \cdot \sum_{s\in\Sigma} m_{ss}(a,b) = \kappa(a,b) \quad \text{and} \quad p_e = \frac{1}{r^2} \cdot \sum_{s\in\Sigma} m_s(a)m_s(b) = \chi(a,b)$$

COHEN's kappa is defined for $a \neq b$ by

$$\mathsf{K}(a,b) := \frac{p_0 - p_e}{1-p_e} = \frac{\kappa(a,b)-\chi(a,b)}{1-\chi(a,b)}$$

If a and b are random strings with not necessarily uniform letter probabilities p_s, then K is asymptotically normally distributed with expectation 0 and variance

$$\frac{p_0 \cdot (1-p_0)}{r \cdot (1-p_0)^2}$$

Therefore its use is convenient for large series of observations—or large strings—but in cryptanalysis we mostly have to deal with short strings, and considering κ and χ separately may retain more information.

Mean Values

For a fixed $a \in \Sigma^r$ we determine the mean value of $\kappa(a,b)$ taken over all $b \in \Sigma^q$:

$$\begin{aligned}
\frac{1}{n^q} \cdot \sum_{b\in\Sigma^q} \chi(a,b) &= \frac{1}{n^q} \cdot \sum_{b\in\Sigma^q} \left[\frac{1}{rq} \cdot \sum_{s\in\Sigma} m_s(a)m_s(b)\right] \\
&= \frac{1}{rqn^q} \cdot \sum_{s\in\Sigma} m_s(a) \underbrace{\sum_{b\in\Sigma^q} m_s(b)}_{q\cdot n^{q-1}} \\
&= \frac{1}{rqn^q} \cdot r \cdot q \cdot n^{q-1} = \frac{1}{n}
\end{aligned}$$

where we used the corollary of Proposition 4.

In an analogous way we determine the mean value of $\chi(a, f_\sigma(b))$ for fixed $a, b \in \Sigma^r$ over all permutations $\sigma \in \mathcal{S}(\Sigma)$:

$$\frac{1}{n!} \cdot \sum_{\sigma\in\mathcal{S}(\Sigma)} \chi(a, f_\sigma(b)) = \frac{1}{rqn!} \cdot \sum_{\sigma\in\mathcal{S}(\Sigma)} \sum_{s\in\Sigma} m_s(a) m_s(f_\sigma(b))$$

As usual we interchange the order of summation, and evaluate the sum

$$\begin{aligned}
\sum_{\sigma\in\mathcal{S}(\Sigma)} m_s(f_\sigma(b)) &= \frac{1}{n} \cdot \sum_{t\in\Sigma} \sum_{\sigma\in\mathcal{S}(\Sigma)} m_t(f_\sigma(b)) \\
&= \frac{1}{n} \cdot \sum_{\sigma\in\mathcal{S}(\Sigma)} \underbrace{\sum_{t\in\Sigma} m_t(f_\sigma(b))}_{q} = \frac{1}{n} \cdot n! \cdot q = (n-1)! \cdot q
\end{aligned}$$

using the symmetry with respect to s. Therefore

$$\begin{aligned}\frac{1}{n!}\cdot\sum_{\sigma\in\mathcal{S}(\Sigma)}\chi(a,f_\sigma(b)) &= \frac{1}{rqn!}\cdot\sum_{s\in\Sigma}m_s(a)\cdot\sum_{\sigma\in\mathcal{S}(\Sigma)}m_s(f_\sigma(b))\\ &= \frac{1}{rqn!}\cdot r\cdot(n-1)!\cdot q=\frac{1}{n}\end{aligned}$$

Note that this conclusion also holds for $a = b$.

This derivation shows:

Proposition 5 (i) *The mean value of $\chi(a,b)$ over all texts $b \in \Sigma^*$ of a fixed length q is $\frac{1}{n}$ for all $a \in \Sigma^*$.*

(ii) *The mean value of $\chi(a,b)$ over all $a \in \Sigma^r$ and $b \in \Sigma^q$ is $\frac{1}{n}$ for all $r, q \in \mathbb{N}_1$.*

(iii) *The mean value of $\chi(a, f_\sigma(b))$ over all monoalphabetic substitutions with $\sigma \in \mathcal{S}(\Sigma)$ is $\frac{1}{n}$ for each pair $a, b \in \Sigma^*$.*

(iv) *The mean value of $\chi(f_\sigma(a), f_\tau(b))$ over all pairs of monoalphabetic substitutions, with $\sigma, \tau \in \mathcal{S}(\Sigma)$, is $\frac{1}{n}$ for each pair $a, b \in \Sigma^*$.*

Interpretation

- For a given text a and a "random" text b we have $\chi(a,b) \approx \frac{1}{n}$.
- For "random" texts a and b we have $\chi(a,b) \approx \frac{1}{n}$.
- For given texts a and b and a "random" monoalphabetic substitution f_σ we have $\chi(a, f_\sigma(b)) \approx \frac{1}{n}$. This remark justifies treating a nontrivially monoalphabetically encrypted text as random with respect to χ and plaintext.
- For given texts a and b and two "random" monoalphabetic substitutions f_σ, f_τ we have $\chi(f_\sigma(a), f_\tau(b)) \approx \frac{1}{n}$.

Empirical Results

The results are in Tables D.16, D.17, and D.18. We see that χ—in contrast with the coincidence index κ—performs extremely well, in fact in our experiments it even completely separates English and German texts from random texts of length 100. It is a test with power near 100% and error probability near 0%. The χ test even distinguishes between English and German texts at the 5% error level with a power of almost 75%. For this assertion compare the 95% quantile for English with the first quartile for German.

The results for 26 letter texts are in Tables D.19, D.20, and D.21. The χ-test is quite strong even for 26 letters: At the 5% error level its power is around 91% for English, 98% for German.

3.15 Adjusting the Columns of a Disk Cipher

As a last application in this chapter we look at the problem: How to adjust the alphabets in the columns of a disk cipher? From Chapter 2 we know that this works only when the primary alphabet is known.

Imagine a ciphertext from a disk cipher whose period l we know already. Write the ciphertext in rows of length l. Then the columns are monoalphabetically encrypted, each with (in most cases) another alphabet. By Proposition 5 (iv) we expect a χ-value of about $\frac{1}{n}$ for each pair of columns. Since the alphabets for the columns are secondary alphabets of a disk cipher they differ only by a relative shift in the alphabet. There are 26 different possible shifts. These can be checked by exhaustion: We try all 26 possibilities (including the trivial one, bearing in mind that two columns can have the same alphabet). The perfect outcome would be 25 values near $\frac{1}{n}$, and one outcome around the coincidence index of the plaintext language, clearly indicating the true alphabet shift. The experimental results of Section D.5 give hope that real outcome should approximate the ideal one in a great number of cases.

Example 1

Let us try out this idea for the ciphertext from Section 3.9. We are pretty sure that the period is 7. (And we also adjusted the columns by visual inspection in Chapter 2.) The first two columns are

```
ARCYPMEAZKRWKHZLRXTRTMYYRLMTVYCMRBZZKOLKKTKOTCUKKOMVBLYUYYZALR
  OEKWZMWZZRYZOOTUYURMTYYSOZEKLYVUYBYTZYKOVMYYMZMZVYROKYTYMUWZ
  PZTZLSPLYLZVYYYBYMQMWWRXZYOKKMYZTZAKQZZT
OMZYYDMYPQMHMFKAMMAACDNNZPIMYZHCJSCNCJQMMYLEMMPNNPZYSNYHPNMOAM
  CAJMPZIVNMPADAHNKFNNAHNVFJHFXNYPNSYFMKNFMDNPZFGJMVMCMXYZZMQC
  MSYIMVAMKZOANZVSZFKMYEMQHZQMNDPMHDMKIYJF
```

Using the Perl script `adjust.pl` we get the results

Shift:	0	1	2	3	4	5	6
χ:	0.0499	0.0365	0.0348	0.0285	0.0320	0.0341	0.0298
7	8	9	10	11	12	13	14
0.0416	0.0307	0.0421	0.0402	0.0448	**0.0799**	0.0495	0.0373
15	16	17	18	19	20	21	22
0.0375	0.0293	0.0330	0.0276	0.0307	0.0306	0.0316	0.0352
23	24	25					
0.0338	0.0461	0.0529					

The result is clear without ambiguity: The correct shift is 12. Going through all $7 \times 6/2 = 21$ pairs of columns we use the Perl script `coladj.pl` and get results in Table 3.16 that are consistent with each other and with the results of Chapter 2.

Table 3.16: *The optimal alphabet shifts for 7 columns*

Column:	0	1	2	3	4	5
1	12					
2	4	18				
3	15	3	11			
4	10	24	6	21		
5	24	12	20	9	14	
6	3	17	25	14	19	5

Example 2

The best guess for the period of the short ciphertext of Section 3.13 was $l = 20$. Therefore we consider 20 columns of lengths 8 or 7:

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

We have to assume the primary alphabet as known in order to know how to shift the columns, that is, how to identify the distance of the secondary alphabets of two columns relative to each other. The primary alphabet is `QWERTZUABCDFGHIJKLMNOPSVXY`, the complete alphabet table is in Table 3.17.

The method from Example 1 gives $20 \times 19/2 = 190$ proposals for optimal shifts between columns. However even for the first columns we already get inconsistent results. We face a complex optimization problem. Instead of continuing with the next columns we better would follow a proposal by SINKOV: Pick up the highest χ-values and try to build clusters of fitting columns. But also this approach fails. After several hours off the track we try to understand why.

Let us imagine a plaintext of the same length, written in rows of length 20, columns of length 7 or 8. Take two columns that each have one letter twice and five or six single letters. Shifting the alphabets in such a way that the "twins" become identical letters, they contribute a summand of

$$\frac{4}{49} \approx 0.0818 \text{ for lengths } 7/7, \ \frac{4}{56} \approx 0.0714 \text{ for } 8/7, \ \frac{4}{64} \approx 0.0625 \text{ for } 8/8,$$

Table 3.17: *The alphabet table used in the example*

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

to the χ-value. If accidentally there is another common letter, these values rise to

$$\frac{5}{49} \approx 0.1020 \text{ for lengths } 7/7,\ \frac{5}{56} \approx 0.0893 \text{ for } 8/7,\ \frac{5}{64} \approx 0.0781 \text{ for } 8/8.$$

And therefore we'll get many false alarms that will make the task of finding the correct solution very time-consuming. An experiment with plaintext comfirms this. Here all shifts should be 0, however we found the maximal χ-value for a shift of 0 in less then 20% of all cases.

To get better chances for success we need some known plaintext or more ciphertext or luck. We had luck and got more ciphertext. The following two messages b and c,

```
AWYFN DHZPE PENES YGAVO YHGAD VTNLL TFKKH FHGYT DOGJI HJHHB
OOYFV EWDSJ MOIFY DRTLA BRRFE ZQGYQ AVYCH BQZPR RZTTH IONZE
SCEFH EFJBJ RNRWE TGVZR EYIIQ IZRWT OLGOC ICLFS EMYAH E

LIGJC KTNLF KBMZH XYWFB UWVPC RNYAJ WEVKV BRVPN PXYOT KVGLE
MBVHE WFZSM UOWFI EYXLB XRRKC XKGPT YONFY DKZLU CXRDC YJWZT
UWPDS VZWNU KORLK WUXUO WVHFL IEGXJ ZUKGC YJVDN EFYDK GJZON
BYXEV EWQSD MMHSS GJ
```

could be encrypted with the same key. Number 1 and 2 have a coincidence index $\kappa(a, b) \approx 0.0411$ only. But $\kappa(a, c) \approx 0.0811$, $\kappa(b, c) \approx 0.1027$. For both b and c the period 20 is confirmed by the SINKOV statistic and also by the autocoincidence spectrum. Therefore we align all three messages below each other with rows of length 20. From bad experience we know we should proceed very thoughtfully. Therefore we first look at the letter frequencies in the single columns (of lengths 22 to 25). The columns 2, 3, and 12 contain a letter in 7 exemplars. We try to adjust these columns in such a way that the most frequent letters match. For column 3 relative to column 2 we get a χ-value of 0.1072 for a shift of 14, the next χ-value being 0.0608. If we write the columns as rows, the result looks like this

```
Column 02: JRYDQYCBYGGIYEIYGVYWNPHYH
Column 03: JYFIQDOYFAJFCFIAJPOFFDFDS
shifted:   RHYEIXBHYPRYVYEPRCBYYXYXD
```

In both cases the most frequent letter with 7 occurrences is `Y`. For column 12 we get the optimal shift 22 relative to column 2 with a χ-value of 0.1273, the next χ-value being 0.0836. This also looks good and gives the result

```
Column 02: JRYDQYCBYGGIYEIYGVYWNPHYH
Column 12: MZRMYRANKYRTRGMVVRRRKX
shifted:   IWYIPYRJGPYQYBINNYYYGO
```

Also in the shifted column 12 the letter Y occurs 7 times. If we are right, comparing columns 3 and 12 should lead to the same result. Indeed the optimal shift is 8 with $\chi \approx 0.1109$, the next χ-value being 0.0727.

This makes us confident that we are on the right track, and encourages us to set Y it to plaintext e. We continue our task under the hypothesis that columns 2, 3, and 12 match with the given shifts as

```
...
JRYDQYCBYGGIYEIYGVYWNPHYH
RHYEIXBHYPRYVYEPRCBYYXYXD
...
IWYIPYRJGPYQYBINNYYYGO
...
```

We take this text fragment as cluster "A" and try to match further columns. First we take columns where the most frequent letters occur 6 or 5 times.

```
A vs  5: Optimal shift is 15 with chi = 0.0906 (next is 0.0683)
A vs  8: Optimal shift is  8 with chi = 0.1092 (next is 0.0758)
A vs 14: Optimal shift is 16 with chi = 0.1092 (next is 0.0859)

A vs  0: Optimal shift is 23 with chi = 0.0878 (next is 0.0817)
A vs  5: Optimal shift is  0 with chi = 0.0809 (next is 0.0619)
A vs  9: Optimal shift is 21 with chi = 0.0966 (next is 0.0663)
```

The most convincing match is with column 8, therefore we join it to our cluster, forming cluster "B":

```
...
JRYDQYCBYGGIYEIYGVYWNPHYH
RHYEIXBHYPRYVYEPRCBYYXYXD
...
BHNRLWGRYPYRKCYJYYYWUE
...
IWYIPYRJGPYQYBINNYYYGO
...
```

Looking at the distribution of letters the Y stands out by far—that is no surprise because we picked columns with the most frequent letters and matched these. As a more meaningful check we transform our cluster to (presumed) plaintext; this means decrypting the fragments with the secondary alphabet that transforms e to Y, that is PSVXYQWERTZUABCDFGHIJKLMNO. This gives the supposed plaintext fragment (to be read top down):

```
...
uiepfeonerrtehtercegyases
```

```
isehtdnseaiecehaioneededp
...
nsyiwgrieaeivoeueeeglh
...
tgetaeiuraefentyyeeerz
...
```

This looks promising. Trying to extend this cluster by a formal procedure is dangerous because there could be columns with a most frequent (plaintext) letter other then `e`. Instead we look at neighboring columns, say at column 4 that should give a readable continuation of columns 2 and 3, in particular extending the digraph `th` in a meaningful way. The proposed shift should have a `Y` (for `e`) as 15th letter, or maybe a `P` (for `a`), or an `R` (for `i`).

Cluster B versus column 4 yields the optimal shift 3 with $\chi \approx 0.0753$, the 15th letter being `R` (for `i`). The next best values are $\chi \approx 0.0664$ for a shift of 12, the 15th letter then being `G` (for `r`), and $\chi \approx 0.0604$ for a shift of 25, the 15th letter being `Y` (for `e`). To decide between these possible solutions we decrypt the shifted columns and get the proposed cleartext columns

```
zoeiaetpbswhvvivrrmwhezye
ixnrjncykbfqeereaavfqnihn
vkaewaplxosdrrernnisdavua
```

Joining them to columns 3 and 4 the first one looks somewhat inauspicuous but possible, the second one looks awkward, the third one looks best and is our first choice. This gives the three adjacent columns

```
uiepfeonerrtehtercegyases
isehtdnseaiecehaioneededp
vkaewaplxosdrrernnisdavua
```

and the new cluster "C" of (monoalphabetic) ciphertext, comprising columns 2, 3, 4, 8, 12:

```
...
JRYDQYCBYGGIYEIYGVYWNPHYH
RHYEIXBHYPRYVYEPRCBYYXYXD
KZPYLPDUMCHXGGYGBBRHXPKJP
...
BHNRLWGRYPYRKCYJYYYWUE
...
IWYIPYRJGPYQYBINNYYYGO
...
```

Note that for joining further columns we must not work with the (proposed) plaintext columns because the transformation between plaintext and ciphertext is not a simple shift.

Comparing the adjacent columns with cluster C we obtain

```
C vs  1: Optimal shift is  1 with chi = 0.0642 (next is 0.0632)
C vs  5: Optimal shift is 15 with chi = 0.0844 (next is 0.0686)
C vs  7: Optimal shift is 20 with chi = 0.0676 (next is 0.0621)
C vs  9: Optimal shift is  6 with chi = 0.0695 (next is 0.0653)
C vs 11: Optimal shift is  5 with chi = 0.0695 (next is 0.0638)
C vs 13: Optimal shift is 23 with chi = 0.0684 (next is 0.0588)
```

The best value seems that for column 13, so let's try this one first (skipping the dead end via column 5). The new cluster D is

```
...
JRYDQYCBYGGIYEIYGVYWNPHYH     uiepfeonerrtehtercegyases
RHYEIXBHYPRYVYEPRCBYYXYXD     isehtdnseaiecehaioneededp
KZPYLPDUMCHXGGYGBBRHXPKJP     vkaewaplxosdrrernnisdavua
...
BHNRLWGRYPYRKCYJYYYWUE        nsyiwgrieaeivoeueeeglh
...
IWYIPYRJGPYQYBINNYYYGO        tgetaeiuraefentyyeeerz
EPJVIYDYHBBWXLEHDHAICY        hauctepesnngdwhspsmtoe
...
```

This looks good, and detecting the two `th`'s between the cleartext columns 12 and 13 we try column 14 next.

```
D vs 14: Optimal shift is 16 with chi = 0.0945 (next is 0.0793)
```

If we rely on this result, we get the next cluster E:

```
...
JRYDQYCBYGGIYEIYGVYWNPHYH     uiepfeonerrtehtercegyases
RHYEIXBHYPRYVYEPRCBYYXYXD     isehtdnseaiecehaioneededp
KZPYLPDUMCHXGGYGBBRHXPKJP     vkaewaplxosdrrernnisdavua
...
BHNRLWGRYPYRKCYJYYYWUE        nsyiwgrieaeivoeueeeglh
...
IWYIPYRJGPYQYBINNYYYGO        tgetaeiuraefentyyeeerz
EPJVIYDYHBBWXLEHDHAICY        hauctepesnngdwhspsmtoe
PBDCAPHBYCIYIPYCIPPEPC        anpomasneotetaeotaahao
...
```

Good! Let's continue with column 15:

```
E vs 15: Optimal shift is  0 with chi = 0.0719 (next is 0.0574)
```

Joining the resulting "cleartext" to columns 12, 13, 14 gives the disturbing result

```
tgetaeiuraefentyyeeerz
hauctepesnngdwhspsmtoe
anpomasneotetaeotaahao
evkpceqeqhktjtdngdegeh
```

Therefore we dismiss this proposal. Unfortunately also the next best χ-value gives no sensible result. On the other hand the shifts giving a possible complement to the `th` have a quite small χ-value. Therefore we leave column 15 and retry column 1:

```
E vs  1: Optimal shift is  1 with chi = 0.0631 (next is 0.0577)
```

This would give us cluster F:

```
...
FXGRCKGYEIPPXDQNJEYPPEXGN     qdriovrehtaadpfyuheaahdry
JRYDQYCBYGGIYEIYGVYWNPHYH     uiepfeonerrtehtercegyases
RHYEIXBHYPRYVYEPRCBYYXYXD     isehtdnseaiecehaioneededp
KZPYLPDUMCHXGGYGBBRHXPKJP     vkaewaplxosdrrernnisdavua
...
BHNRLWGRYPYRKCYJYYYWUE        nsyiwgrieaeivoeueeeglh
...
IWYIPYRJGPYQYBINNYYYGO        tgetaeiuraefentyyeeerz
EPJVIYDYHBBWXLEHDHAICY        hauctepesnngdwhspsmtoe
PBDCAPHBYCIYIPYCIPPEPC        anpomasneotetaeotaahao
...
```

The plaintext now begins with `.quiv...`. A dictionary search finds hits such as "equivalent", "equivocal", and "a quiver". We compare cluster F with column 1 and look for shifts that make the first letter `a` (`P` in our secondary alphabet) or `e` (`Y`). We have luck! The optimal shift gives `e`, so we take this as our favourite solution:

```
F vs  0: Optimal shift is  7 with chi = 0.0717 (next is 0.0696)
```

and form the next cluster G:

```
YGCVHCYXIULYIRCCXHEHUHBCY     erocsoedtlwetioodshslsnoe
FXGRCKGYEIPPXDQNJEYPPEXGN     qdriovrehtaadpfyuheaahdry
JRYDQYCBYGGIYEIYGVYWNPHYH     uiepfeonerrtehtercegyases
RHYEIXBHYPRYVYEPRCBYYXYXD     isehtdnseaiecehaioneededp
KZPYLPDUMCHXGGYGBBRHXPKJP     vkaewaplxosdrrernnisdavua
...
BHNRLWGRYPYRKCYJYYYWUE        nsyiwgrieaeivoeueeeglh
...
IWYIPYRJGPYQYBINNYYYGO        tgetaeiuraefentyyeeerz
EPJVIYDYHBBWXLEHDHAICY        hauctepesnngdwhspsmtoe
PBDCAPHBYCIYIPYCIPPEPC        anpomasneotetaeotaahao
...
```

Noting the fragments `ciphe` in "line" 4 (fourth column in the schema above) and `ipher` in "line" 14, we cannot resist completing them as `cipher`.

```
G vs  5: Optimal shift is 11 with chi = 0.0708 (next is 0.0697)
G vs 19: Optimal shift is 21 with chi = 0.0775 (next is 0.0585)
```

Note that we now see how misleading our former results for column 5 were. This is caused by the six `a`'s in this column that the χ-method tried to associate with the `e`'s of other columns.

Adding both of these results in one step gives cluster H:

```
YGCVHCYXIULYIRCCXHEHUHBCY     erocsoedtlwetioodshslsnoe
FXGRCKGYEIPPXDQNJEYPPEXGN     qdriovrehtaadpfyuheaahdry
JRYDQYCBYGGIYEIYGVYWNPHYH     uiepfeonerrtehtercegyases
RHYEIXBHYPRYVYEPRCBYYXYXD     isehtdnseaiecehaioneededp
KZPYLPDUMCHXGGYGBBRHXPKJP     vkaewaplxosdrrernnisdavua
YDLKHDYYYGEYVLRLZMZLYGRWW     alsrolaaandaysesgtgsanecc
...
BHNRLWGRYPYRKCYJYYYWUE        nsyiwgrieaeivoeueeeglh
...
IWYIPYRJGPYQYBINNYYYGO        tgetaeiuraefentyyeeerz
EPJVIYDYHBBWXLEHDHAICY        hauctepesnngdwhspsmtoe
PBDCAPHBYCIYIPYCIPPEPC        anpomasneotetaeotaahao
...
YAYIAECJYDPVXLRIHYYJIZ        emetmhouepacdwitseeutk
```

We see that column 6 should start with `l` (`U`). And this is also the "χ-optimal" solution:

```
H vs  6: Optimal shift is 10 with chi = 0.0734 (next is 0.0554)
```

And column 7 should start with `e` (`Y`):

```
H vs  7: Optimal shift is 20 with chi = 0.0647 (next is 0.0639)
```

We are not amused, also the next best χ is unwanted. However the shift that gives `e` has a χ-value of 0.0639 that is acceptable. We fill in columns 6 and 7:

```
YGCVHCYXIULYIRCCXHEHUHBCY     erocsoedtlwetioodshslsnoe
FXGRCKGYEIPPXDQNJEYPPEXGN     qdriovrehtaadpfyuheaahdry
JRYDQYCBYGGIYEIYGVYWNPHYH     uiepfeonerrtehtercegyases
RHYEIXBHYPRYVYEPRCBYYXYXD     isehtdnseaiecehaioneededp
KZPYLPDUMCHXGGYGBBRHXPKJP     vkaewaplxosdrrernnisdavua
YDLKHDYYYGEYVLRLZMZLYGRWW     alsrolaaandaysesgtgsanecc
UYRHGCDIVIYHDPJIRACQJGYY      leisroptctespautimofuree
YHUUCBYHIESHIXGEBPAIDPI       esllonesthbstdrhnamtpat
BHNRLWGRYPYRKCYJYYYWUE        nsyiwgrieaeivoeueeeglh
...
IWYIPYRJGPYQYBINNYYYGO        tgetaeiuraefentyyeeerz
EPJVIYDYHBBWXLEHDHAICY        hauctepesnngdwhspsmtoe
PBDCAPHBYCIYIPYCIPPEPC        anpomasneotetaeotaahao
...
YAYIAECJYDPVXLRIHYYJIZ        emetmhouepacdwitseeutk
```

It's time, for easier reading, to arrange our findings in the right order where "columns" are columns:

```
equivalen...tha....e   rdiskless...gan....m
oreeasily...eup....e   ciphersli...tco....t
softworow...atm....m   ovedalong...eea....h
eronpaper...ips....o   denslats
theexacti...uen....u   ltraonthe...rse....e
warisdeba...ano....p   eatedasse...ent....a
tdecrypti...fge....c   iphersadv...edt....d
oftheeuro...nwa....w   oyears
duringthe...the....i   shcontinu...yso....t
heenigmae...ypt....s   sagessome...esa....e
layedafte...ema....e   shadanupg...eth....u
ndseveral...roa....t   oreduceth...zeo....k
eyspace
```

Now its easy to complete the text: In the first row read `equivalent` and complete column 9. In the fourth row read `cipher slide` and complete column 10. Then read `with` in the first row and complete column 11. Then in the last two rows we recognize `the size of ... keyspace`, this allows us to complete column 15. Now in the first two rows we read `cipher disk` and complete the remaining columns 16, 17, 18.

This is the final solution:

```
equivalentwithaciphe   rdisklesselegantbutm
oreeasilymadeupisthe   cipherslideitconsist
softworowsthatmaybem   ovedalongsideeachoth
eronpaperstripsorwoo   denslats
theexactinfluenceofu   ltraonthecourseofthe
warisdebatedanoftrep   eatedassessmentistha
tdecryptionofgermanc   iphersadvancedtheend
oftheeuropeanwarbytw   oyears
duringthewarthebriti   shcontinuallysolvedt
heenigmaencryptedmes   sagessometimesabitde
layedafterthemachine   shadanupgradetheyfou
ndseveralapproachest   oreducethesizeofthek
eyspace
```

3.16 Modeling a Language by a Markov Process

For deriving theoretical results a common model of language is the interpretation of texts as results of Markov processes. This model was introduced by Shannon in his fundamental papers published after World War II.

If we look at letter frequencies only, we define a Markov process of order 0. If we also incorporate bigram frequencies into our model, it becomes a Markov process of order 1, if we include trigram frequencies, of order 2, and so on.

In this section we want to derive theoretical expectation values for κ, φ, and χ. For this the order of the Markov model is irrelevant.

Message Sources

Let the alphabet Σ be equipped with a probability distribution, assigning the probability p_s to the letter $s \in \Sigma$. In particular $\sum_{s\in\Sigma} p_s = 1$. We call (Σ, p) a **message source** and consider random variables X in Σ, that is mappings $X\colon \Omega \longrightarrow \Sigma$ where Ω is a finite probability space with probability measure P, such that $P(X^{-1}s) = p_s$ for all $s \in \Sigma$.

Picking a letter of Σ at random from the message source is modeled as evaluating $X(\omega)$ for some $\omega \in \Omega$. We calculate the expectation values of the Kronecker symbols for random variables $X, Y\colon \Omega \longrightarrow \Sigma$ and letters $s \in \Sigma$

where Y may belong to a message source (Σ, q) with a possibly different probability distribution $q = (q_s)_{s\in\Sigma}$:

$$\delta_{sX}(\omega) = \begin{cases} 1 & \text{if } X(\omega) = s \\ 0 & \text{otherwise} \end{cases} \qquad \delta_{XY}(\omega) = \begin{cases} 1 & \text{if } X(\omega) = Y(\omega) \\ 0 & \text{otherwise} \end{cases}$$

Lemma 5 (i) $\mathrm{E}(\delta_{sX}) = p_s$ *for all* $s \in \Sigma$.

(ii) *If X and Y are independent, then* $\mathrm{E}(\delta_{XY}) = \sum_{s\in\Sigma} p_s q_s$.

(ii) *If X and Y are independent, then δ_{sX} and δ_{sY} are independent.*

Proof. (i) Since δ takes only the values 0 and 1, we have

$$\mathrm{E}(\delta_{sX}) = 1 \cdot P(X^{-1}s) + 0 \cdot P(\Omega - X^{-1}s) = P(X^{-1}s) = p_s.$$

(ii) In the same way, using the independence of X and Y,

$$\begin{aligned} \mathrm{E}(\delta_{X,Y}) &= 1 \cdot P(\omega \mid X(\omega) = Y(\omega)) + 0 \cdot P(\omega \mid X(\omega) \neq Y(\omega)) \\ &= P(X = Y) = \sum_{s\in\Sigma} P(X^{-1}s \cap Y^{-1}s) \\ &= \sum_{s\in\Sigma} P(X^{-1}s) \cdot P(Y^{-1}s) = \sum_{s\in\Sigma} p_s q_s \end{aligned}$$

(iii) $\delta_{sX}^{-1}(1) = \{\omega | X(\omega) = s\} = X^{-1}s$, and $\delta_{sX}^{-1}(0) = \Omega - X^{-1}s$. The same for Y. The assertion follows because $P(X^{-1}s \cap Y^{-1}s) = P(X^{-1}s) \cdot P(Y^{-1}s)$. $\diamond$

Picking a random text of length r is modeled by evaluating an r-tuple of random variables at some ω. This leads to the following definition:

Definition. A **message** of length r from the message source (Σ, p) is a sequence $X = (X_1, \ldots, X_r)$ of random variables $X_1, \ldots, X_r\colon \Omega \longrightarrow \Sigma$ such that $P(X_i^{-1}s) = p_s$ for all $i = 1, \ldots, r$ and all $s \in \Sigma$.

Note. In particular the X_i are identically distributed. They are not necessarily independent.

The Coincidence Index of Message Sources

Definition. Let $Y = (Y_1, \ldots, Y_r)$ be another message of length r from a possibly different message source (Σ, q). Then the **coincidence index** of X and Y is the random variable

$$\mathsf{K}_{XY}\colon \Omega \longrightarrow \mathbb{R}$$

defined by

$$\mathsf{K}_{XY}(\omega) := \frac{1}{r} \cdot \#\{i = 1, \ldots, r \mid X_i(\omega) = Y_i(\omega)\} = \frac{1}{r} \cdot \sum_{i=1}^{r} \delta_{X_i(\omega), Y_i(\omega)}$$

We calculate its expectation under the assumption that each pair of X_i and Y_i is independent. From Lemma 5, using the additivity of E, we get

$$\mathrm{E}(\mathsf{K}_{XY}) = \frac{1}{r} \cdot \sum_{i=1}^{r} \mathrm{E}(\delta_{X_i,Y_i}) = \frac{1}{r} \cdot r \cdot \sum_{s\in\Sigma} p_s q_s = \sum_{s\in\Sigma} p_s q_s$$

independently of the length r. Therefore it is adequate to call this expectation the **coincidence index** κ_{LM} **of the two message sources** L, M. We have proven:

Theorem 4 *The coincidence index of two message sources* $L = (\Sigma, p)$ *and* $M = (\Sigma, q)$ *is*

$$\kappa_{LM} = \sum_{s\in\Sigma} p_s q_s$$

Now we are ready to calculate theoretical values for the "typical" coincidence indices of languages under the assumption that the model "message source" fits their real behaviour:

Example 1, random texts versus any language M**:** Here all $p_s = 1/n$, therefore $\kappa_{\Sigma*} = n \cdot \sum_{s\in\Sigma} 1/n \cdot q_s = 1/n$.

Example 2, English texts versus English: From Table 3.18 we get the value 0.0653.

Example 3, German texts versus German: The table gives 0.0758.

Example 4, English versus German: The table gives 0.0664.

Note that these theoretical values for the real languages differ slightly from the former empirical values. This is due to two facts:

- The model—as every mathematical model—is an approximation to the truth.
- The empirical values underly statistical variations and depend on the kind of texts that were evaluated.

The Cross-Product Sum of Message Sources

For a message $X = (X_1, \ldots, X_r)$ from a message source (Σ, p) we define the (relative) letter frequencies as random variables

$$\mathrm{M}_{sX}\colon \Omega \longrightarrow \mathbb{R}, \quad \mathrm{M}_{sX} = \frac{1}{r} \cdot \sum_{i=1}^{r} \delta_{sX_i},$$

Table 3.18: Calculating theoretical values for coincidence indices

Letter s	English p_s	German q_s	Square p_s^2	Square q_s^2	Product $p_s q_s$
A	0.082	0.064	0.006724	0.004096	0.005248
B	0.015	0.019	0.000225	0.000361	0.000285
C	0.028	0.027	0.000784	0.000729	0.000756
D	0.043	0.048	0.001849	0.002304	0.002064
E	0.126	0.175	0.015876	0.030625	0.022050
F	0.022	0.017	0.000484	0.000289	0.000374
G	0.020	0.031	0.000400	0.000961	0.000620
H	0.061	0.042	0.003721	0.001764	0.002562
I	0.070	0.077	0.004900	0.005929	0.005390
J	0.002	0.003	0.000004	0.000009	0.000006
K	0.008	0.015	0.000064	0.000225	0.000120
L	0.040	0.035	0.001600	0.001225	0.001400
M	0.024	0.026	0.000576	0.000676	0.000624
N	0.067	0.098	0.004489	0.009604	0.006566
O	0.075	0.030	0.005625	0.000900	0.002250
P	0.019	0.010	0.000361	0.000100	0.000190
Q	0.001	0.001	0.000001	0.000001	0.000001
R	0.060	0.075	0.003600	0.005625	0.004500
S	0.063	0.068	0.003969	0.004624	0.004284
T	0.091	0.060	0.008281	0.003600	0.005460
U	0.028	0.042	0.000784	0.001764	0.001176
V	0.010	0.009	0.000100	0.000081	0.000090
W	0.023	0.015	0.000529	0.000225	0.000345
X	0.001	0.001	0.000001	0.000001	0.000001
Y	0.020	0.001	0.000400	0.000001	0.000020
Z	0.001	0.011	0.000001	0.000121	0.000011
Sum	1.000	1.000	0.0653	0.0758	0.0664

or more explicitly,

$$\mathrm{M}_{sX}(\omega) = \frac{1}{r} \cdot \#\{i \mid X_i(\omega) = s\} \quad \text{for all } \omega \in \Omega.$$

We immediately get the expectation

$$\mathrm{E}(\mathrm{M}_{sX}) = \frac{1}{r} \cdot \sum_{i=1}^{r} \mathrm{E}(\delta_{sX_i}) = p_s.$$

Definition. Let $X = (X_1, \ldots, X_r)$ be a message from the source (Σ, p), and $Y = (Y_1, \ldots, Y_t)$, a message from the source (Σ, q). Then the **cross-product sum** of X and Y is the random variable

$$\mathsf{X}_{XY} \colon \Omega \longrightarrow \mathbb{R}, \quad \mathsf{X}_{XY} := \frac{1}{rt} \cdot \sum_{s\in\Sigma} \mathrm{M}_{sX}\mathrm{M}_{sY}.$$

To calculate its expectation we assume that each X_i is independent of all Y_j, and each Y_j is independent of all X_i. Under this assumption let us call the messages X and Y **independent**. Then from Lemma 5 and the formula

$$\mathsf{X}_{XY} := \frac{1}{rt} \cdot \sum_{s\in\Sigma} \sum_{i=1}^{r} \sum_{j=1}^{t} \delta_{sX_i}\delta_{sY_j}$$

we get

$$\mathrm{E}(\mathsf{X}_{XY}) = \frac{1}{rt} \cdot \sum_{s\in\Sigma} \sum_{i=1}^{r} \sum_{j=1}^{t} \mathrm{E}(\delta_{sX_i})\mathrm{E}(\delta_{sY_j}) = \sum_{s\in\Sigma} p_s q_s$$

again independently of the length r. Therefore we call this expectation the **cross-product sum** χ_{LM} **of the two message sources** L, M. We have proven:

Theorem 5 *The cross-product sum of two message sources $L = (\Sigma, p)$ and $M = (\Sigma, q)$ is*

$$\chi_{LM} = \sum_{s\in\Sigma} p_s q_s.$$

The Inner Coincidence Index of a Message Source

Let $X = (X_1, \ldots, X_r)$ be a message from a source (Σ, p). In analogy with Sections 3.10 and D.5 we define the random variables

$$\Psi_X, \Phi_X \colon \Omega \longrightarrow \mathbb{R}$$

by the formulas

$$\Psi_X := \sum_{s\in\Sigma} M_{sX}^2, \qquad \Phi_X := \frac{r}{r-1} \cdot \Psi_x - \frac{1}{r-1}.$$

We try to calculate the expectation of Ψ_X first:

$$\begin{aligned}\Psi_X &= \frac{1}{r^2}\cdot\sum_{s\in\Sigma}\left(\sum_{i=1}^r \delta_{sX_i}\right)^2\\ &= \frac{1}{r^2}\cdot\sum_{s\in\Sigma}\left(\sum_{i=1}^r \delta_{sX_i} + \sum_{i=1}^r\sum_{j\neq i}\delta_{sX_i}\delta_{sX_j}\right)\end{aligned}$$

since $\delta_{sX_i}^2 = \delta_{sX_i}$. Taking the expectation value we observe that for a sensible result we need the assumption that X_i and X_j are *independent* for $i \neq j$.

In the language of Markov chains this means that we assume a Markov chain of order 0: The single letters of the messages from the source are independent from each other.

Under this assumption we get

$$\begin{aligned}\mathrm{E}(\Psi_X) &= \frac{1}{r^2}\cdot\sum_{s\in\Sigma}\left(\sum_{i=1}^r p_s + \sum_{i=1}^r\sum_{j\neq i}\mathrm{E}(\delta_{sX_i})\mathrm{E}(\delta_{sX_j})\right)\\ &= \frac{1}{r^2}\cdot\left(\sum_{i=1}^r\underbrace{\sum_{s\in\Sigma}p_s}_{1} + \sum_{s\in\Sigma}p_s^2\cdot\underbrace{\sum_{i=1}^r\sum_{j\neq i}1}_{r\cdot(r-1)}\right)\\ &= \frac{1}{r} + \frac{r-1}{r}\cdot\sum_{s\in\Sigma}p_s^2.\end{aligned}$$

For Φ_X the formula becomes a bit more elegant:

$$\mathrm{E}(\Phi_X) = \frac{r}{r-1}\cdot\left(\frac{r-1}{r}\cdot\sum_{s\in\Sigma}p_s^2 + \frac{1}{r}\right) - \frac{1}{r-1} = \sum_{s\in\Sigma}p_s^2.$$

Let us call this expectation $\mathrm{E}(\Phi_X)$ the **(inner) coincidence index** of the message source (Σ, p), and let us call (by abuse of language) the message source **of order** 0 if its output messages are Markov chains of order 0 only. (Note that for a mathematically correct definition we should have included the “transition probabilities” into our definition of message source.) Then we have proved

Theorem 6 *The coincidence index of a message source $L = (\Sigma, p)$ of order 0 is*

$$\varphi_L = \sum_{s\in\Sigma}p_s^2.$$

The assumption of order 0 is relevant for small text lengths and negligeable for large texts, because for "natural" languages dependencies between letters affect small distances only. Reconsidering the tables in Section D.4 we note in fact that the values for texts of lengths 100 correspond to the theoretical values, whereas for texts of lengths 26 the values are suspiciously smaller. An explanation could be that repeated letters, such as `ee`, `oo`, `rr`, are relatively rare and contribute poorly to the number of coincidences. This affects the power of the φ-test in an unfriendly way.

On the other hand considering SINKOV's test for the period in Section 3.13 we note that the columns of a polyalphabetic ciphertext are decimated excerpts from natural texts where the dependencies between letters are irrelevant: The assumption of order 0 is justified for SINKOV's test.

3.17 Stochastic Languages

The stochastic model of language as a stationary MARKOV process easily led to useful theoretic results that fit well with empirical observations. On the other hand it is far from the computer scientific model that regards a language as a fixed set of strings with certain properties and that is intuitively much closer to reality. In fact the MARKOV model may produce *every* string in Σ^* with a non-zero probability! (We assume that each letter $s \in \Sigma$ has a non-zero probability—otherwise we would throw it away.) Experience tells us that only a very small portion of all character strings represent meaningful texts in any natural language. Here we consider an alternative model that respects this facet of reality, but otherwise is somewhat cumbersome.

Recall from Chapter 1 that a language is a subset $M \subseteq \Sigma^*$.

A Computer Theoretic Model

The statistical cryptanalysis of the monoalphabetic substitution relied on the hypothesis—supported by empirical evidence—that the average relative frequencies of the letters $s \in \Sigma$ in texts of sufficient length from this language approximate typical values p_s. This is even true when we consider only fixed positions j in the texts, at least for almost all j—the first letters of texts for example usually have different frequencies.

Now we try to build a mathematical model of language that reflects this behaviour. Let $M \subseteq \Sigma^*$ a language, and $M_r := M \cap \Sigma^r$ for $r \in \mathbb{N}$ the set of texts of length r. The average frequency of the letter $s \in \Sigma$ at the position $j \in [0 \ldots r-1]$ of texts in M_r is

$$\mu_{sj}^{(r)} := \frac{1}{\#M_r} \cdot \sum_{a \in M_r} \delta_{sa_j}$$

(This sum counts the texts $a \in M_r$ with the letter s at position j.)

Example Let $M = \Sigma^*$ Then

$$\mu_{sj}^{(r)} := \frac{1}{n^r} \cdot \sum_{a \in \Sigma^r} \delta_{sa_j} = \frac{1}{n} \quad \text{for all } s \in \Sigma,\ j = 1, \ldots, r-1,$$

because there are exactly n^{r-1} possible texts with fixed $a_j = s$.

Definition

The language $M \subseteq \Sigma^*$ is called **stochastic** if there is at most a finite exceptional set $J \subseteq \mathbb{N}$ of positions such that

$$p_s := \lim_{r \to \infty} \mu_{sj}^{(r)}$$

exists uniformly in j and is independent from j for all $j \in \mathbb{N} - J$ and all $s \in \Sigma$.

The p_s are called the **letter frequencies** of M and obviously coincide with the limit values for the frequencies of the letters over the complete texts.

Examples and Remarks

1. The exceptional set J for natural languages usually consists only of the start position 0 and the end position. That is, the first and last letters of texts may have different frequencies. For example in English the letter “t” is the most frequent first letter instead of “e”, followed by “a” and “o”. In German this is “d”, followed by “w”, whereas “t” almost never occurs as first letter.
2. The language $M = \Sigma^*$ is stochastic.
3. Because always $\sum_{s \in \Sigma} \mu_{sj}^{(r)} = 1$, also $\sum_{s \in \Sigma} p_s = 1$.

Note that this notation is not standard in the literature.

Also note that we consider a *theoretical model.* For a natural language it may not be well-defined whether a given text is meaningful or not, not even if it is taken from a newspaper.

The Mean Coincidence Between Two Languages

Let $L, M \subseteq \Sigma^*$ two stochastic languages with letter frequencies q_s and p_s for $s \in \Sigma$. We consider the mean value of the coincidences of texts of length r:

$$\kappa_{LM}^{(r)} := \frac{1}{\#L_r} \cdot \frac{1}{\#M_r} \cdot \sum_{a \in L_r} \sum_{b \in M_r} \kappa(a, b)$$

Theorem 7 *The mean coincidence of the stochastic languages L and M is asymptotically*

$$\lim_{r\to\infty} \kappa_{LM}^{(r)} = \sum_{s\in\Sigma} p_s q_s$$

The proof follows.

Interpretation: The coincidence of sufficiently long texts of the same length is approximately

$$\kappa(a,b) \approx \sum_{s\in\Sigma} p_s q_s$$

An Auxiliary Result

Lemma 6 *Let M be a stochastic language. Then the average deviation for all letters $s \in \Sigma$*

$$\frac{1}{r}\cdot\sum_{j=0}^{r-1}\left(\mu_{sj}^{(r)} - p_s\right) \to 0 \quad \textit{for } r\to\infty$$

Proof. Fix $\varepsilon > 0$, and let r large enough that

1. $r \geq 4\cdot\frac{\#J}{\varepsilon}$,
2. $|\mu_{sj}^{(r)} - p_s| < \frac{\varepsilon}{2}$ for all $j \in [0\ldots r] - J$.

For $j \in J$ we have $|\mu_{sj}^{(r)} - p_s| \leq |\mu_{sj}^{(r)}| + |p_s| \leq 2$. Therefore

$$\frac{1}{r}\cdot\sum_{j=0}^{r-1}|\mu_{sj}^{(r)} - p_s| < \frac{1}{r}\cdot 2\cdot\#J + \frac{r-\#J}{r}\cdot\frac{\varepsilon}{2} \leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon.$$

$\diamond$

Remark The mean frequency of s in texts of length r is

$$\mu_s^{(r)} = \frac{1}{r}\cdot\sum_{j=0}^{r-1}\mu_{sj}^{(r)} = \frac{1}{r}\cdot\frac{1}{\#M_r}\cdot\sum_{a\in M_r}\delta_{sa_j}$$

For this we get the limit

Corollary 5 $\lim_{r\to\infty}\mu_s^{(r)} = p_s$

Proof of the Theorem

$$\begin{aligned}
\kappa_{LM}^{(r)} &= \frac{1}{\#L_r \cdot \#M_r} \cdot \sum_{a\in L_r} \sum_{b\in M_r} \left(\frac{1}{r} \cdot \sum_{j=0}^{r-1} \sum_{s\in\Sigma} \delta_{sa_j}\delta_{sb_j} \right) \\
&= \sum_{s\in\Sigma} \frac{1}{r} \cdot \sum_{j=0}^{r-1} \left[\frac{1}{\#L_r} \sum_{a\in L_r} \delta_{sa_j} \right] \cdot \left[\frac{1}{\#M_r} \sum_{b\in M_r} \delta_{sb_j} \right] \\
&= \sum_{s\in\Sigma} \frac{1}{r} \cdot \sum_{j=0}^{r-1} [q_s + \varepsilon_{sj}] \cdot [p_s + \eta_{sj}] \\
&= \sum_{s\in\Sigma} \left[p_s q_s + \frac{p_s}{r} \cdot \sum_{j=0}^{r-1} \varepsilon_{sj} + \frac{q_s}{r} \cdot \sum_{j=0}^{r-1} \eta_{sj} + \frac{1}{r} \cdot \sum_{j=0}^{r-1} \varepsilon_{sj}\eta_{sj} \right]
\end{aligned}$$

The second and third summands converge to 0 by the lemma. The fourth converges to 0 because $|\varepsilon_{sj}\eta_{sj}| \leq 1$. Therefore the sum converges to $\sum_{s\in\Sigma} p_s q_s$. $\diamond$

Chapter 4

Cylinder Ciphers

4.1 Introduction

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/4_Cylinder/Cylinder.html`

4.2 Idea and History of Cylinder Ciphers

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/4_Cylinder/HistCyl.html`

4.3 Mathematical Description of Cylinder Ciphers

This section assumes knowledge of the mathematical excursion to permutations in the Appendix to the Chapter on monoalphabetic ciphers.

Parameters

A cylinder cipher depends on the following parameters:

- The number $n = \#\Sigma$ of letters in the alphabet Σ
- The number q of disks, where $q \geq 1$. If all disks are different, then $q \leq (n-1)!$. [See below for an explanation why we don't need to take $n!$ for the upper bound.]
  - Each disk is characterized by a permutation $\tau \in \mathcal{S}(\Sigma)$.
  - Therefore the collection of disks can be described as a q-tuple $(T_1, \ldots, T_q) \in \mathcal{S}(\Sigma)^q$.

  Assume the disks are numbered from 1 to q.
- The number l of selected disks, where $1 \leq l \leq q$

- The key is a sequence $(\tau_0, \ldots, \tau_{l-1})$ consisting of different members of the q-tuple $(T_1, \ldots, T_q)$, and described by the corresponding sequence of numbers in $[1 \ldots q]$.
- The number of choices for the key is

  $$\#K = q \cdot (q-1) \cdots (q-l+1) = \frac{q!}{(q-l)!}$$

  some of which could coincide if some of the disks have identical alphabets.

Examples

Jefferson **cylinder:** $l = q = 36$, $\#K = 36!$, effective key length ≈ 138.

Bazeries **cylinder:** $l = q = 20$, $\#K = 20!$, effective key length ≈ 61.

M-94: $l = q = 25$, $\#K = 25!$, effective key length ≈ 84.

M-138-A: $l = 30$, $q = 100$, $\#K = 100!/70!$, effective key length ≈ 190.

Encryption and Decryption

The cylinder cipher is polyalphabetic with period l, the number of disks on the cylinder.

Attention: Don't confuse the permutation $\tau \in \mathcal{S}(\Sigma)$ written on the circumference of the disk with the permutation $\sigma \in \mathcal{S}(\Sigma)$ that defines the substitution alphabet realized by the disk. We subsequently examine the relationship between these two permutations.

As usual identify the alphabet Σ (in a fixed order) with $\mathbb{Z}/n\mathbb{Z}$, the integers mod n. Then, using the first generatrix, encrypting a plaintext block $(a_0, \ldots, a_{l-1})$ looks like this:

$$\begin{array}{lclcl}
a_0 & \ldots & a_i & \ldots & a_{l-1} \\
\hline
 & & \tau_i(0) & & \\
 & & \vdots & & \\
\text{Search entry } x \text{ such that} & & \tau_i(x) & = a_i & \\
 & & \tau_i(x+1) & = c_i & \text{corresponding cipher letter} \\
 & & \vdots & & \\
 & & \tau_i(n-1) & &
\end{array}$$

where the center column $\tau_i(0), \ldots, \tau_i(n-1)$ represents the marking of the i-th disk. Therefore

$$c_i = \tau_i(x+1) = \tau_i(\tau_i^{-1} a_i + 1)$$

The corresponding decryption function is

$$a_i = \tau_i(\tau_i^{-1} c_i - 1)$$

This derivation proves:

Theorem 8 (Cylinder Cipher Theorem) *The relation between the permutation $\tau \in \mathcal{S}(\Sigma)$ written on the circumference of the disk and the permutation $\sigma \in \mathcal{S}(\Sigma)$ that defines the substitution alphabet realized by the disk using the first generatrix is given by the formulas*

$$\begin{array}{rcl} \sigma(a) & = & \tau(\tau^{-1}a + 1) \\ \sigma^{-1}(c) & = & \tau(\tau^{-1}c - 1) \end{array}$$

Or in other words: σ is a cyclic permutation and τ is the cycle representation of σ.

There are $(n-1)!$ different cycles of length n. As n different disk definitions τ result in the same cyclic permutation σ we could make the restriction $q \leq (n-1)!$ for the number of possible different disks.

Corollary 6 *Using the j-th generatrix the formulas become*

$$\begin{array}{rcl} \sigma_j(a) & = & \tau(\tau^{-1}a + j) \\ \sigma_j^{-1}(c) & = & \tau(\tau^{-1}c - j) \end{array}$$

if we denote by σ_j the substitution by the j-th generatrix.

Example: Let $\Sigma = \{\text{A}, \ldots, \text{Z}\}$, and let the disk inscription be

$$\tau = \text{QWERTZUIOPASDFGHJKLYXCVBNM}$$

Then σ is the permutation

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

4.4 The Bazeries Cylinder

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/4_Cylinder/Bazeries.html`

4.5 Cryptanalysis of Cylinder Ciphers

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/4_Cylinder/AnalysisCyl.html`

4.6 Breaking the Bazeries Cylinder

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/4_Cylinder/deViaris.html`

4.7 Consequences from Cryptanalysis

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/4_Cylinder/ConsCyl.html`

4.8 Key Generators with Long Periods

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/4_Cylinder/LongPeriods.html`

Chapter 5

Rotor Machines

5.1 One-Rotor Ciphers

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/5_Rotor/OneRotor.html`

5.2 Mathematical Description of Rotors

Identify the alphabet Σ with $\mathbb{Z}/n\mathbb{Z}$, the integers $\bmod n$. Let ρ be the monoalphabetic substitution that the rotor performs in its initial position. Moving the rotor by one position forward the new substitution is

$$\rho^{(1)}(a) = \rho(a-1) + 1$$

Denote by τ the shift by 1 of the alphabet $\Sigma = \mathbb{Z}/n\mathbb{Z}$, that is $\tau(a) = a + 1$. Then the formula looks like this:

$$\rho^{(1)}(a) = \tau\rho\tau^{-1}(a)$$

By induction we immediately get part (i) of the following theorem:

Theorem 9 (The secondary alphabets of a rotor)

(i) *If a rotor in its initial position performs the substitution with the primary alphabet ρ, then after rotation by t positions forward it performs the substitution with the conjugate alphabet $\rho^{(t)} = \tau^t\rho\tau^{-t}$. In particular all secondary alphabets have the same cycle type.*

(ii) *The diagonals of the corresponding alphabet table each contain the standard alphabet (cyclically wrapped around).*

Proof. Assertion (i) is proved above. Assertion (ii) follows immediately by interpreting it as a formula:

$$\rho^{(i)}(j) = \tau^i \rho \tau^{-i}(j) = \rho(j-i) + i = \rho^{(i-1)}(j-1) + 1$$

◇

The definition of “cycle type” is given in Appendix A.

The formula makes it obvious why—in contrast with the cipher disk—for a rotor the (unpermuted) standard alphabet is completely useless: It corresponds to the identity permutation, therefore all its conjugates are identical.

In general the conjugate alphabet $\rho^{(t)}$ is identical with the primary alphabet ρ if and only if ρ is in the centralizer of the shift τ^t. The designer of a rotor might wish to avoid such wirings.

Examples.

1. If n is a prime number, then all the shifts τ^t for $t = 1, \ldots, n-1$ are cycles of length n. Therefore all their centralizers are identical to the cyclic group $< \tau >$ spanned by τ. If the designer avoids these n trivial wirings, then all the n conjugated alphabets are distinct.
2. If $\gcd(t, n) = d > 1$, then τ^t splits into d cycles of length $\frac{n}{d}$, $\tau^t = \pi_1 \cdots \pi_d$, and centralizes all permutations of the type $\pi_1^{s_1} \cdots \pi_d^{s_d}$. These are not in the cyclic group $< \tau >$ unless all exponents s_i are congruent mod $\frac{n}{d}$.
3. In the case $n = 26$ the shifts τ^t are cycles, if t is coprime with 26. However τ^t splits into two cycles of length 13, if t is even. All the powers τ^t, t even, $2 \leq t \leq 24$, span the same cyclic group because 13 is prime. The permutation τ^{13} splits into 13 transpositions. For example τ^2 centralizes the permutation $(ACE \ldots Y)$, and τ^{13} centralizes the transposition (AB), where we denoted the alphabet elements by the usual letters A, ..., Z. Therefore in wiring the rotors the designer should avoid the centralizers of τ^2 and of τ^{13}.

5.3 Cryptanalysis of One-Rotor Ciphers (with Unknown Alphabet)

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/5_Rotor/Anal1Rot.html`

5.4 Rotor Machines

General Description

Rotor machines are electromechanical devices that consist of several rotors in series connection. Figure 5.1 gives an impression of the electric flow through such a machine.

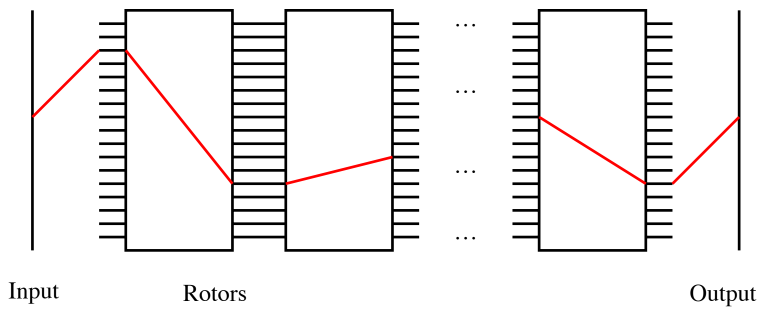


Figure 5.1: *Rotor machine circuit*

With each input letter the rotors move in individual ways, some by one position, some by several positions, some only after several steps. *The cryptographic security of a rotor machine depends* on the number of rotors, the multitude of different settings, and, in a crucial way, *on the complexity of the rotor movements.*

Operating a Rotor Machine

The operator hits a key on the keyboard that corresponds to the next plaintext letter. This action closes an electric circuit powering a light-bulb that corresponds to the ciphertext letter. Or it powers a type bar that prints the ciphertext letter. The rotors move according to their control logic, in general before the circuit is closed. See the FAQ at `http://www.staff.uni-mainz.de/pommeren/Cryptology/FAQ.html`.

Rotor machines are the state of the art in encryption during the period from 1920 until 1970. The mystic and irregularly rotating wheelwork that makes the desk tremble with each key hit looks very attractive and impresses the general or diplomat who wants to buy security.

Mathematical Description

The following abstract model describes an idealized rotor machine. Concrete historic machines each have their own peculiar details.

As before we identify the alphabet Σ with $\mathbb{Z}/n\mathbb{Z}$, the integers mod n. A rotor machine has the following characteristic parameters:

- A set $R \subseteq \mathcal{S}(\Sigma)$ of $p = \#R$ rotors. Each of these defines a primary alphabet, that is a permutation $\rho_i \in \mathcal{S}(\Sigma)$ that corresponds to the wiring of the rotor.

- A choice $\rho = (\rho_1, \ldots, \rho_q) \in \mathcal{S}(\Sigma)^q$ of q different rotors $\rho_i \in R$. There are $p \cdot (p-1) \cdots (p-q+1)$ choices if we assume that all rotors are differently wired ($q \leq p$). This choice serves as "primary key" and is usually fixed for several messages, say for an entire day.

- A state vector $z = (z_1, \ldots, z_q) \in (\mathbb{Z}/n\mathbb{Z})^q$ that describes the current rotor positions. The initial state $z^{(0)}$ serves as "secondary key" that usually changes with each message. The number of different initial states is n^q. Sometimes it is convenient to map the states to $\mathbb{Z}/n^q\mathbb{Z}$, the integers $\bmod n^q$, using the representation of integers in base n. The state vector $z = (z_1, \ldots, z_q) \in (\mathbb{Z}/n\mathbb{Z})^q$ then corresponds to the integer $\zeta = z_1 \cdot n^{q-1} + \cdots + z_q$.

- A state-transition function

$$g \colon \mathbb{N} \times \Sigma^q \longrightarrow \Sigma^q$$

  that transforms the state at time i, $z^{(i)}$, to the state at time $i+1$, $z^{(i+1)} = g(i, z^{(i)})$, where "time" is discrete and simply counts the plaintext letters. This function g represents the control logic and is realized for example by more or less complex gear drives. In most rotor machines the state-transition function is independent of the time i.

- The substitution in state z:

$$\sigma_z := \rho_q^{(z_q)} \circ \cdots \circ \rho_1^{(z_1)} \quad \text{where } \rho_j^{(z_j)} := \tau^{z_j} \circ \rho_j \circ \tau^{-z_j}$$

Ideally the map $\Sigma^q \longrightarrow \mathcal{S}(\Sigma)$, $z \mapsto \sigma_z$ would be injective, that is each state defines a different substitution. Unfortunately no useful general results seem to exist beyond the case $q = 1$ treated in Subsection 5.2.

Perl programs for encryption and decryption by rotor machines are in the web directory `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Perl/` as `rotmach.pl` and `rotdecr.pl`.

The Key Space

By the description above a key of our idealized rotor machine consists of

- a choice of rotors

- an initial state

Therefore the key space K has

$$\#K = n^q \cdot \frac{p!}{(p-q)!}$$

elements. In a typical example (HEBERN machine) we have $p = q = 5$, $n = 26$, $\#K = 120 \cdot 26^5 = 712882560$, and the effective key length is $d(F) \approx 29.4$. That was good enough in 1920. Today, against an enemy with a computer, this is much too little.

> In fact the HEBERN machine was not good enough even in 1920 because it allows attacks far more efficient than exhaustion.

Encryption and Decryption

The plaintext $a = (a_1, \ldots, a_r) \in \Sigma^r$ is encrypted by the formula

$$c_i = \sigma_{z^{(i)}}(a_i)$$

At full length this formula reads

$$c_i = \tau^{z_q^{(i)}} \circ \rho_q \circ \tau^{z_{q-1}^{(i)} - z_q^{(i)}} \circ \cdots \circ \tau^{z_1^{(i)} - z_2^{(i)}} \circ \rho_1 \circ \tau^{-z_1^{(i)}}(a_i)$$

Decryption follows the formula

$$a_i = \tau^{z_1^{(i)}} \circ \rho_1^{(-1)} \circ \tau^{z_2^{(i)} - z_1^{(i)}} \circ \cdots \circ \tau^{z_q^{(i)} - z_{q-1}^{(i)}} \circ \rho_q^{(-1)} \circ \tau^{-z_q^{(i)}}(c_i)$$

Technically for decryption we simply have to route the current through the machine in the reverse direction, of course interchanging the keyboard and lightbulbs. The sequence of states is identical for encryption and decryption.

The Rotor Machine as a Finite-State Automaton

Figure 5.2 shows an abstract model of a rotor machine.

Usually the state-transition function is independent of the step i. Then it has the simpler form

$$g \colon \Sigma^q \longrightarrow \Sigma^q$$

This makes the states periodic as shown in the next subsection.

Periods of State Changes

Let M be a finite set with $m = \#M$. We may think of the elements of M as "states". Consider a map ("state transition")

$$g : M \longrightarrow M.$$

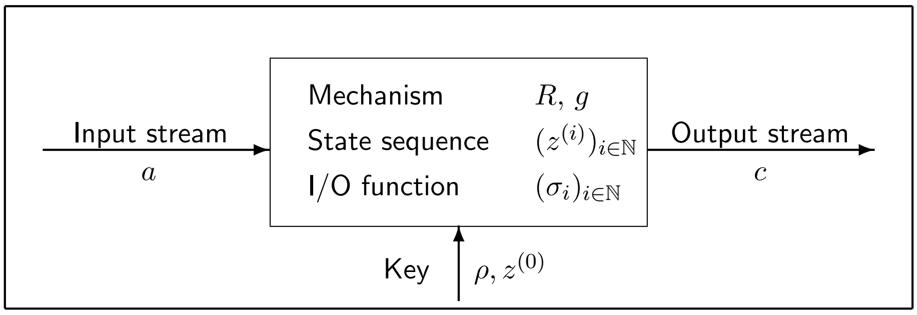

Mechanism R, g
State sequence $(z^{(i)})_{i\in\mathbb{N}}$
I/O function $(\sigma_i)_{i\in\mathbb{N}}$
Input stream
a
Output stream
c
Key
$\rho, z^{(0)}$

Figure 5.2: *Rotor machine as finite-state automaton*

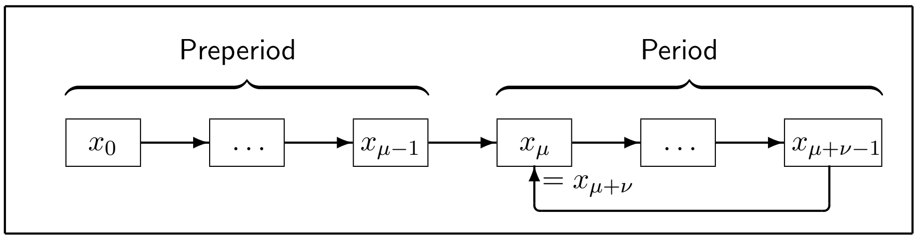


Figure 5.3: *Period and preperiod*

For each element ("initial state") $x_0 \in M$ we define a sequence $(x_i)_{i\in\mathbb{N}}$ in M by the recursion formula $x_i = g(x_{i-1})$ for $i \geq 1$. After a preperiod of length μ this sequence becomes periodic with a period of ν, see Figure 5.3, an explanation follows below.

Since M is finite there are smallest integers $\mu \geq 0$ and $\nu \geq 1$ such that $x_{\mu+\nu} = x_\mu$: Take for μ the smallest index such that the element x_μ reappears somewhere in the sequence, and for $\mu+\nu$ the index where the first repetition occurs. Then also

$$x_{i+\nu} = x_i \quad \text{for } i \geq \mu.$$

Obviously $0 \leq \mu \leq m-1$, $1 \leq \nu \leq m$, $\mu+\nu \leq m$. The values $x_0, \ldots, x_{\mu+\nu-1}$ are all distinct, and the values $x_0, \ldots, x_{\mu-1}$ never reappear in the sequence.

Definition: μ is called (length of the) **preperiod**, ν is called (length of the) **period**.

5.5 The Control Logic of a Rotor Machine

We treat several approaches to rotor stepping. The first three are streamlined versions of real control mechanisms that in practice are implemented in a more complex way: the odometer, the gear drive with gaps, the gear drive with different number of cogs. We also treat the ultimate mechanism: the pseudorandom stepping, and a historical one: the HEBERN mechanism. For the stepping of the Enigma we refer to Chapter 6.

The insight that an irregular movement is the essential ingredient for a secure rotor machine is apparently due to FRIEDMAN after he broke he HEBERN machine. He himself, together with his collaborator ROWLETT, then in several steps developed the top-level rotor machine, the SIGABA.

Example 1: The Odometer Logic

The rotors step like in a mechanical counter or electricity meter. Assume the rotors are mounted as in Figure 5.4. The rightmost rotor moves by one position for each input letter. Each rotor, after completing one revolution, by some kind of protrusion makes its left neighbor move by one position.

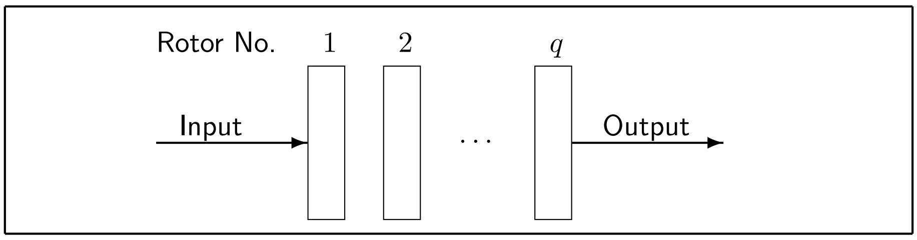


Figure 5.4: *Odometer logic*

Using the identification of the states with the integers mod n^q the sequence of states simply corresponds to the natural sequence of numbers beginning with the initial state.

Remarks

1. In this example the rightmost rotor, rotor number q, is a "fast" rotor, it moves with every step.
2. The leftmost rotor, number 1, is a "slow" rotor. It moves only after n^{q-1} steps, that is almost never, or only for very long messages. For this reason it makes little sense to use more then three rotors with odometer stepping. The effect of all additional rotors together only amounts to a fixed substitution. In the best case they could move once during encryption, effecting two different fixed substitutions.
3. Of course we could also implement the converse stepping where rotor 1 is fast and rotor q is slow.
4. The sequence of states has period n^q.

Example 2: Gaps

Figure 5.5 shows the principle of this control logic. For an implementation we have several mechanical options, for example a pin wheel.

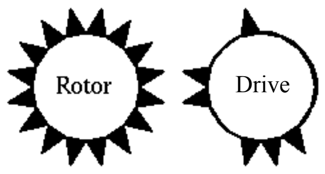


Figure 5.5: *Gear drive with tooth gaps*

A single wheel is characterized by a binary vector

$$u_{(j)} = (u_{j0}, \ldots, u_{j,t-1}) \in \mathbb{F}_2^t \quad \text{for } j = 0, \ldots, t-1$$

where t is the circumference of the wheel, not necessarily $t = n$. A 1 denotes a cog, a 0, a gap. We may describe all the wheels together by a binary matrix

$$u = \begin{pmatrix} u_{10} & \ldots & u_{1,t-1} \\ \vdots & \ddots & \vdots \\ u_{q0} & \ldots & u_{q,t-1} \end{pmatrix} \in M_{qt}(\mathbb{F}_2)$$

The column vectors

$$u^{(i)} = (u_{1i}, \ldots, u_{qi}) \in \mathbb{F}_2^q \quad \text{for } i = 0, \ldots, q-1$$

apply one after the other from left to right, cyclically repeated. This gives a sequence of period t for the states of the gear drive. The states of the rotors generally have a much larger period.

In the simplest case this logic steps the rotor j

- by one position, if $u_{ji} = 1$,
- not at all, if $u_{ji} = 0$,

for the i-th plaintext letter. This gives the formula

$$z^{(i+1)} = z^{(i)} + u^{(i)}$$

where addition is vector addition in $(\mathbb{Z}/n\mathbb{Z})^q$.

Another way to use gap wheels is turning them around a full turn in each step. Then the each of the rotors moves a number of steps given by the corresponding row sum in the matrix. This logic is equivalent with Example 3 below.

Example 3: Different Gear Wheels

Each rotor is driven by its own gear wheel. These share a common axis and make a full turn in each step. If wheel i has n_i cogs, then rotor i moves by n_i positions. The states occur with a period of $\mathrm{lcm}(n_1, \ldots, n_q)$.

The first models of Enigma (A and B) had a control like this.

Example 4: Pseudorandom Stepping

The rotor stepping is controlled by a (pseudo-) random generator, that is a mechanism or an algorithm that generates numbers indinguishable from pure random such as generated with the help of dice. This is easy for a computer simulation. For an (electro-) mechanical rotor machine one can use a key generating mechanism such as in one of the (later) HAGELIN machines.

FRIEDMAN was the first to detect the weaknesses of a regular rotor stepping when he analyzed the then current rotor machines in the 1920's. He came up with the idea of an irregular stepping by a pseudorandom mechanism. First he tried a punched paper tape, but this proved not robust enough. Then ROWLETT had the idea of realizing the stepping control by another set of rotors. Thus the American super rotor machine SIGABA was invented.

For details see the book

> Stephen J. Kelly: *Big Machines.* Aegean Park Press, Walnut Creek 2001, ISBN 0-89412-290-8.

Example 5: The HEBERN Machine

The HEBERN machine has $q = 5$ rotors and uses the standard alphabet with $n = 26$. The stepping follows an odometer logic, but with a complex mechanism that doesn't affect the neighboring rotor but another one, in more detail:

- Rotors 2 and 4 don't rotate at all. They are "stators".
- Rotor 5 moves by 1 position with every step, it is a fast rotor.
- Rotor 1 moves by 1 position with each complete turn of rotor 5. It is a "semi-fast" rotor.
- Rotor 3 moves by 1 position with each complete turn of rotor 1. It is a slow rotor.

Moreover the rotors move in the other direction compared with the description in Section 5.2.

The equation for the state change—not yet the correct one!—is

$$g(z_1, z_2, z_3, z_4, z_5) = (z_1 + \lambda(z_5), z_2, z_3 + \lambda(z_1)\lambda(z_5), z_4, z_5 + 1)$$

where $\lambda(x) = \delta_{x,25}$ is the KRONECKER symbol. The states occur with period $26^3 = 17576$.

Characteristic features:

- That the rotors 2 and 4 are static doesn't harm the security of the machine. By the odometer logic they would move only after 26^3 or 26^4 steps, that is only for extremely long messages.
- The stepping of rotor 1 (resp. 3) is induced by rotor 5 (resp. 1) moving from position "N" to position "O". The correct equation for the state change is left as an **exercise** to the reader.
- The wiring between the keyboard and rotor 1 as well as from rotor 5 to the light bulbs is irregular but static. It therefore is assumed as known to the enemy. We may interpret this wiring as two additional stators, one at each end of the rotor pack.
- For decryption there is a switch "direct/reverse" that interchanges input contacts and output contacts.
- The HEBERN rotors are symmetric: they may be mounted with their sides interchanged. This makes the number of possible primary keys larger by a factor of 2^5.

5.6 Historical Rotor Machines

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology /Classic/5_Rotor/HistRot.html`

5.7 Historical Data on Cryptanalysis

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology /Classic/5_Rotor/AnalHist.html`

5.8 Cryptanalysis of Rotor Machines

The cryptanalysis of rotor machines is complex and depends on the details of the machine under examination. The book by DEAVOURS and KRUH [4] is a standard work and contains many elaborate examples. Here we only depict some general ideas:

- Superimposition
- Meet-in-the-middle
- Isomorphs

Superimposition

Assume that the cryptanalyst got hold of several ciphertexts that are encrypted with the same key, then he may align them in such a way that he gets monoalphabetically encrypted columns. Note that this is a ciphertext-only attack. However it needs lots of messages.

Note that operators impede this attack by changing the key (or initial position) for each message. Nevertheless in some scenarios they have to send many messages, think of war. Then with high probability the cryptanalyst will observe many ciphertexts that are produced by the same rotor positions, not necessarily at the same position in the text. She identifies these concordances by extensive calculation of coincidence indices.

Identification of a Fast Rotor

Assume that the set of rotors is known but not their actual choice. Assume that the last rotor at the output side steps by one position with each letter, and that the other rotors move infrequently. The attacker has no known plaintext.

Enumerate the rotors from 1 (= input rotor, slow) to q (= output rotor, fast), and assume the current flows from left to right as in Figure 5.6.

Now assume we have a ciphertext section of length m where only rotor q moved, and for simplicity use the indices 1 to m for this sequence of ciphertext letters. The rotors 1 to $q-1$ together effect a constant substitution μ.

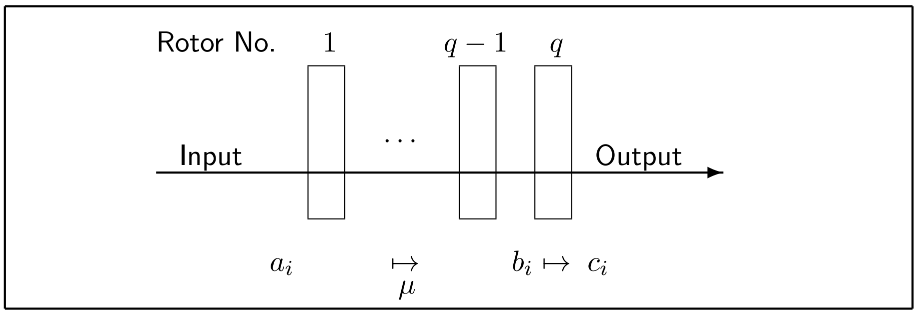


Figure 5.6: *Identifying the fast rotor*

Therefore this part of the encryption follows the schema

$$\begin{array}{ccccccc}
a_1 & \mapsto & b_1 := \mu(a_1) & \mapsto & \rho_q^{(z_1)}\mu(a_1) & = c_1 \\
a_2 & \mapsto & b_2 := \mu(a_2) & \mapsto & \rho_q^{(z_1+1)}\mu(a_2) & = c_2 \\
\vdots & & \vdots & & & \vdots \\
a_m & \mapsto & b_m := \mu(a_m) & \mapsto & \rho_q^{(z_1+m-1)}\mu(a_m) & = c_m
\end{array}$$

Here $b = (b_1, \ldots, b_m) \in \Sigma^m$ is a monoalphabetic image of $a = (a_1, \ldots, a_m)$. We can also look at b "from the other side":

$$\begin{aligned} b_1 &= \left[\rho_q^{(z_1)}\right]^{-1}(c_1) \\ b_2 &= \left[\rho_q^{(z_1+1)}\right]^{-1}(c_2) \\ \vdots & \quad \vdots \\ b_m &= \left[\rho_q^{(z_1+m-1)}\right]^{-1}(c_m) \end{aligned}$$

These formulas enable an exhaustion of the p choices for rotor q and of the n choices for its initial position z_1.

- A wrong choice of the rotor or its initial position makes b look as a random text having coincidence index $\varphi(b) \approx \frac{1}{n}$.
- For the correct choice b is a monoalphabetically encrypted meaningful text having coincidence index $\varphi(b) \approx \kappa_M$, the coincidence index of the plaintext language.

This observation may lead to the identification of the fast rotor and its state for this section of the text at the price of $n \cdot p$ calculations of coincidence indices of texts of length m. But note that the coincidence test for $m = 26$ has little power, it will miss most positive events.

Remarks

1. In principle the method works at each position of the text. Therefore the very beginning of the text is worth a try.
2. In the unfavourable case one of the other rotors moved during the encryption of the m letters. Then the intermediate ciphertext b consists of two different monoalphabetic pieces. With a bit of luck this also leads to a somewhat conspicuous coincidence index.

Continuation of the Attack

As soon as the fast rotor is identified we can strip its effect off like a superencryption. In this way the intermediate ciphertext $(b_1, \ldots, b_m)$ extends to a ciphertext $c' \in \Sigma^r$ that is the result of encrypting the plaintext a by a much simpler machine.

If for example the rotors move like an odometer, and if the ciphertext is long enough ($\approx n^2$), then in a similar way we can identify the next rotor and strip its effect off.

Or we try to cryptanalyze the monoalphabetic parts of c' that we expect $\lfloor \frac{r}{n} \rfloor$ in number of length n plus one or two fragments of total length $r \bmod n$.

We also might first try to find the locations were the second rotor moves.

Known Plaintext Attack

Assume we know or guess a piece of plaintext $a = (a_1, \ldots, a_m)$, say a probable word. An essential step is finding text chunks with identical numerical patterns, also called **isomorphs**. Therefore this attack is known as **Method of Isomorphs**. More generally looking at an intermediate step of an encryption algorithm from both sides, is called **Meet-in-the-Middle**.

Identification of a Fast Output Rotor

If we have a piece of known plaintext we may identify a fast rotor by simple pattern comparisons without calculating coincidence indices: Check if the intermediate text $(b_1, \ldots, b_m)$ shows the same numerical pattern as $(a_1, \ldots, a_m)$.

Identification of a Fast Input Rotor

Known plaintext $a = (a_1, \ldots, a_m)$ also allows the identification of the fast rotor for a *reverse* odometer control where the left rotor is the fast one. In this case we consider the situation of Figure 5.7.

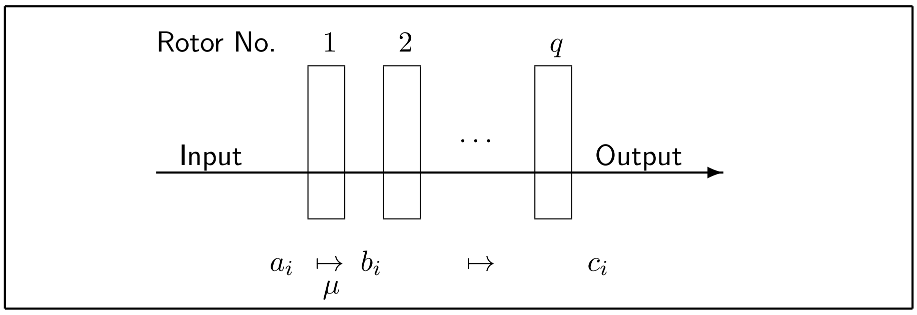


Figure 5.7: *Identifying a fast input rotor*

This part of the encryption follows the schema

$$\begin{array}{ccccc}
a_1 & \mapsto & b_1 := \rho_1^{(z_1)}(a_1) & \mapsto & \mu(b_1) & = c_1 \\
a_2 & \mapsto & b_2 := \rho_1^{(z_1+1)}(a_2) & \mapsto & \mu(b_2) & = c_2 \\
\vdots & & \vdots & & & \vdots \\
a_m & \mapsto & b_m := \rho_1^{(z_1+m-1)}(a_m) & \mapsto & \mu(b_m) & = c_m
\end{array}$$

Here $b = (b_1, \ldots, b_m)$ is a monoalphabetic image of $c = (c_1, \ldots, c_m)$. We try all p rotors in all their n initial positions until the numerical patterns of b and c coincide.

Chapter 6

The Enigma

6.1 General Description

For a general description of this German World War II cipher machine see the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/6_Enigma/EnigmaDescr.html`.

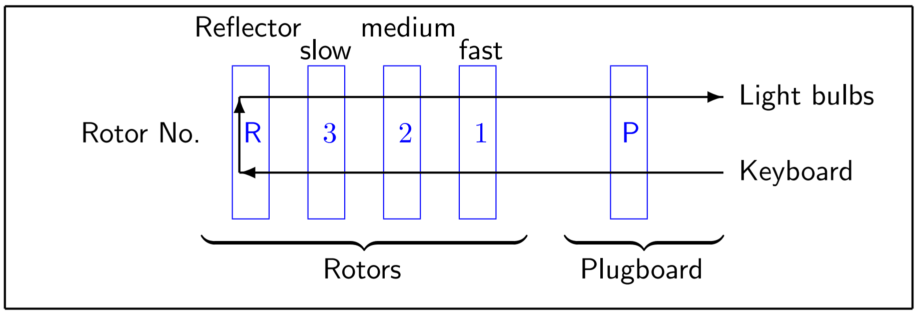


Figure 6.1: *Current flow through Enigma*

6.2 Mathematical Description

Here we give a mathematical description of the Enigma I ("Wehrmachts-Enigma") with 5 selectable rotors denoted by the roman numerals I to V (whereas the arabic numerals 1 to 3 denote the order in which three rotors are mounted). For a bit of mathematical background on permutations see Appendix A.

The Key Space

The key of an Enigma message has several components:

- The operator choses 3 rotors from a set of 5 and mounts them in a certain order. This gives $\frac{5!}{2!} = 60$ different options ("Walzenlage").
- He adjusts each of the 3 alphabet rings to one of 26 possible positions. This gives another $26^3 = 17576$ options. Since the alphabet ring of the slow rotor has no effect on the encryption, only $26^2 = 676$ of these options contribute to the key space.
- He inserts 10 plugs into the plugboard. Each plug connects 2 letters. He has $\frac{26!}{(2^{10} \cdot 10! \cdot 6!)} = 150,738,274,937,250 \approx 1.5 \cdot 10^{14} \approx 2^{47}$ different choices. This formula is derived in Appendix A. If the operator is allowed to use also less than the maximum 10 plugs this number grows to about $2.1 \cdot 10^{14}$.
- Finally he sets the rotors to their initial positions, another $26^3 = 17576$ possibilities.

Multiplied together these numbers make up a key space of

$$60 \cdot 676 \cdot 150,738,274,937,250 \cdot 17576 = 107,458,687,327,250,619,360,000$$
$$\approx 10^{23} \approx 1.4 \times 2^{76}$$

or a key length of 76 bits (in modern language). However it is not clear at all (and even hardly likely) that all keys define different substitutions. Therefore we can conclude only that the effective key length is *at most* 76 bits. And 47 of these bits are due to the plug-board.

The Control Logic

The current flows through the three movable rotors first from right to left. Accordingly we denote the fast rotor by 1, the middle one by 2, and the slow one by 3. Taking the irregularity in the stepping of rotor 2 into account, and denoting the position of the notch that moves the next rotor by m_i, the formula for the state transition function is

$$g(z_1, z_2, z_3) = (z_1, z_2 + \lambda_1(z_1) + \lambda_1(z_1)\lambda_2(z_2), z_3 + \lambda_1(z_1)\lambda_2(z_2))$$

where $\lambda_i(x) = \delta_{x,m_i}$ is the Kronecker symbol.

Due to the direction of the labeling of the rotors and the corresponding wiring between input keys or output bulbs and rotors, the substitution by a single rotor in step i is $\rho^{(i)} = \tau^{-i} \circ \rho \circ \tau^i$ where ρ is the rotor substitution and τ the alphabet shift, as explained in Chapter 5.

The Enigma Substitution

The rotors being in the state $z = (z_1, z_2, z_3)$ the rotor substitution describes the effect of transversing them from right to left:

$$\sigma_z := \rho_3^{(z_3)} \circ \rho_2^{(z_2)} \circ \rho_1^{(z_1)}$$

The effect of the reflecting rotor is a proper involution π, no element is mapped to itself. The plug-board also provides an involution, η. Together this gives the **Enigma substitution** in state z:

$$\rho_z = \eta^{-1} \circ \sigma_z^{-1} \circ \pi \circ \sigma_z \circ \eta$$

or, with more details, the **Enigma equation** for encryption

$$c_i = \eta^{-1}\tau^{-z_1}\rho_1^{-1}\tau^{z_1-z_2}\rho_2^{-1}\tau^{z_2-z_3}\rho_3^{-1}\tau^{z_3}\pi\tau^{-z_3}\rho_3\tau^{z_3-z_2}\rho_2\tau^{z_2-z_1}\rho_1\tau^{z_1}\eta\,(a_i)$$

Theorem 10 *The Enigma substitution ρ_z in state z is a proper involution.*

Proof. a) Involution:

$$\rho_z^{-1} = \eta^{-1} \circ \sigma_z^{-1} \circ \pi^{-1} \circ \sigma_z \circ \eta = \rho_z$$

since $\pi^{-1} = \pi$.

b) Proper: Assume $\rho_z(s) = s$ for a letter $s \in \Sigma$. Then

$$\sigma_z\eta(s) = \sigma_z\eta\rho_z(s) = \pi\sigma_z\eta(s)$$

hence $\pi(t) = t$ for $t = \sigma_z\eta(s) \in \Sigma$. This contradicts the fact that π is a proper involution. $\diamond$

Note. The proof didn't use the fact that η is an involution. This limitation of the plug-board had purely practical reasons: It reduced errors in operation. Variable plugs between the keyboard or light-bulbs and the first rotor would give more degrees of freedom. But this would require 26 cables instead of the 10 double-plug cables.

6.3 Cryptanalysis of Enigma: General Remarks

The number of variants of Enigma and of the corresponding appropriate approaches to cryptanalysis is hardly manageable in an introductory text. For this reason we only treat three selected topics:

1. The Enigma without plugboard

2. Message key analysis after Rejewski

3. Wehrmacht-Enigma and known plaintext

Special Features of Enigma

- Control logic: Because the middle rotor moves only after 26 steps, and the slow rotor moves almost never, the ciphertext essentially consists of sections of length 26 where only the fast rotor moves by one position with each step.
- The decomposition of a rotor permutation into cycles is not affected by the plugboard. The substitution by the set of rotors is simply conjugated by the plugboard substitution.
  - If the attacker has enough known plaintext she finds cycles, see Section 6.7.
  - The diverse rotor orders differ by their cycle types [Rejewski's catalogue, Turing's "classes"].
  - In this way the attacker gets information on the rotor order.
- Negative pattern search allows to narrow down the position of known plaintext.

In World War II this last effect allowed for the detection of test messages by the Italians that consisted only of LLL...LLL. This was a stroke of genius by the british cryptanalyst Mavis Lever who noticed that several cipher messages didn't contain any L. This observation turned out to be an essential step in uncovering the wiring of newly introduced rotors.

6.4 Cryptanalysis of the Enigma Without Plugboard

The Commercial Enigma

The types C and D of Enigma had a reflecting rotor but no plugboard. They were sold on the free market and could be comprehensively analyzed by everyone.

In the Spanish civil war all parties used the Enigma D. All big powers broke it.

The substitution of the commercial Enigma simplifies to

$$c_i = \sigma_z^{-1}\pi\sigma_z(a_i)$$

where σ_z is the substitution by the three rotors in state $z = (z_1, z_2, z_3)$. The reflecting rotor was fixed during encryption but could be inserted in any of 26 positions.

Searching for Isomorphs

In a section of the text where only rotor 1 moves, the two inner rotors together with the reflecting rotor yield a constant involution $\tilde{\pi}$. If the plaintext for this section (say of length m) is known, then we have equations

$$\begin{aligned}
c_1 &= \left[\rho_1^{(z_1)}\right]^{-1} \tilde{\pi}\rho_1^{(z_1)}(a_1)\\
c_2 &= \left[\rho_1^{(z_1+1)}\right]^{-1} \tilde{\pi}\rho_1^{(z_1+1)}(a_2)\\
&\dots\\
c_m &= \left[\rho_1^{(z_1+m-1)}\right]^{-1} \tilde{\pi}\rho_1^{(z_1+m-1)}(a_m)
\end{aligned}$$

Hence for $i = 1, \dots, m$ the intermediate text

$$c'_i = \rho_1^{(z_1+i-1)}(c_i) = \tilde{\pi}\rho_1^{(z_1+i-1)}(a_i)$$

is the monoalphabetic image $c'_i = \tilde{\pi}(a'_i)$ of the intermediate text

$$a'_i = \rho_1^{(z_1+i-1)}(a_i)$$

under the involution $\tilde{\pi}$.

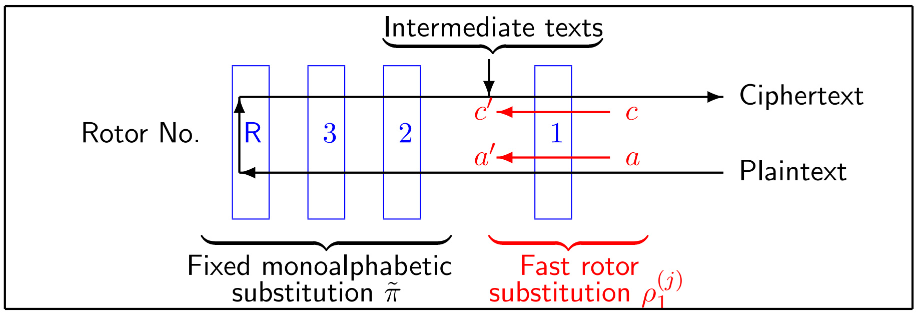

Intermediate texts
Rotor No.
R
3
2
1
c'
c
a'
a
Ciphertext
Plaintext
Fixed monoalphabetic substitution $\tilde{\pi}$
Fast rotor substitution $\rho_1^{(j)}$

Figure 6.2: *Searching for isomorphs*

Therefore pattern search identifies the fast rotor and its state by testing all rotors and all initial states. For determining a'_i from a_i we have to test all three rotors with all 26 start positions, and determine c'_i from c_i with the same rotor in the same position. This exhaustion comprises $3 \times 26 = 78$ different constellations, each of which has to be tested for a matching pattern. Probably there are several false solutions in addition to the correct one.

The next sieving step uses the fact that $\tilde{\pi}$ is a fixed involution. If for a possible solution we find a coincidence $c'_i = a'_j$ with $j \neq i$, then we test for

$$a'_i \mapsto c'_i = a'_j \mapsto c'_j \overset{?}{=} a'_i$$

If no, we discard the solution. If yes, we even identified a 2-cycle of $\tilde{\pi}$, reducing the number of $26^2 = 676$ possible states of the two inner rotors. A useful tool for this is a precomputed table of length 676 for each of the 6 different combinations of these two rotors that contains the cycle decomposition of $\tilde{\pi}$ for all states, making a total of $6 \times 676 = 4056$ involutions.

Precomputing the lookup table is easy: Let the cycles of π be $(a_1, b_1), \ldots, (a_{13}, b_{13})$. Let $\xi = \rho_3^{(z_3)} \circ \rho_2^{(z_2)}$ be the combined substitution by rotors 2 and 3. Then the cycle decomposition of $\tilde{\pi} = \xi^{-1} \circ \pi \circ \xi$ is

$$\tilde{\pi} = (\xi^{-1}a_1, \xi^{-1}b_1), \ldots, (\xi^{-1}a_{13}, \xi^{-1}b_{13})$$

We only need to apply the fixed substitution ξ^{-1} to the string $a_1b_1 \ldots a_{13}b_{13}$.

The location of known plaintext, if not known a priori, may be narrowed down by negative pattern search.

Conclusion

The introduction of the reflecting rotor aimed at a significant gain for the security of Enigma by doubling the number of rotor passages. This turned out to be an illusory complication. The attack by isomorphs reduces the cryptanalysis to the exhaustion of position and state of three rotors only, and even this is reduced in a substantial manner.

To prevent this attack the Wehrmacht (= army) introduced the plugboard when adopting the Enigma.

6.5 Example

Lacking a working simulation for the commercial Enigma we use a military Enigma I omitting the plugboard. Further differences with the commercial Enigma D are

- The reflector is mounted in a fixed position. This will facilitate our task slightly compared with a true Enigma D.
- The rotors (including the reflectors) are differently wired. We consider the wiring as known.
- The input wiring is from keyboard-A to input-A etc., whereas the commercial Enigma had the contacts wired in the order of the keys, i. e. keyboard-Q to input-A, keyboard-W to input-B and so on. This makes no cryptanalytic difference because it amounts to a known renaming of the standard alphabet.
- The notches that move the rotors are fixed at the alphabet rings instead of the rotor bodies, allowing a displacement with respect to the rotor contacts, and thus effecting a slight variablity in the stepping of

the rotors. In our example we ignore this complication that is irrelevant for the commercial Enigma.

The primary rotor alphabets are

```
Clear:    A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Rot I:    E K M F L G D Q V Z N T O W Y H X U S P A I B R C J
Rot II:   A J D K S I R U X B L H W T M C Q G Z N P Y F V O E
Rot III:  B D F H J L C P R T X V Z N Y E I W G A K M U S Q O
Refl B:   Y R U H Q S L D P X N G O K M I E B F Z C W V J A T
```

The cycle decomposition of the reflector is

```
(AY)(BR)(CU)(DH)(EQ)(FS)(GL)(IP)(JX)(KN)(MO)(TZ)(VW)
```

Now assume we got the ciphertext:

```
NMSHH EZJOU OEAJA IDCWS VVMFY IVZQO QWSYO KCEVE QSTLC YMJKT
PFVK
```

We suspect it to be in Spanish but we don't use this conjecture. However it seems likely that it begins with the probable word GENERAL. Negative pattern search yields no contradiction to this assumed known plaintext, however also excludes only very few of other possible positions.

Now we test all three rotors in each possible position in the search for an isomorph. For Rotor I we get 26 pairs of intermediate texts:

```
Pos A: PBURWXL  Pos B: TNQULJH  Pos C: WRNHVNR  Pos D: JUJMBQY
       XWFPJHW   ===>  FEXJQMI         UTMQRGM         QPWRZNP

Pos E: OHTGVDQ  Pos F: IMANAIF  Pos G: PGSOBCP  Pos H: QNHWTJV
       NMCZOOC         JIWOKWH         TSBKHLB         AZCHDHI

Pos I: YORLYKP  Pos J: NWXHSSU  Pos K: JLREOHV  Pos L: GHWAADN
       SRUDNEJ         HGZNUAR   ===>  RQTUMKG         XWPMBRC

Pos M: CEXKEAS  Pos N: MAPRHWM  Pos O: TKUJUGI  Pos P: LROYZNU
       RQBBLJZ         WVFLRYV         XWIRLIF         POVLQOM

Pos Q: AJKITFY  Pos R: KYWOAUB  Pos S: QIAIBEO  Pos T: KODNJKT
 ===>  UTAQRIE         ONURJNT         KJBJOOD         WVCOIGJ

Pos U: PIQOYEN  Pos V: QNVGUJU  Pos W: IOPLRKV  Pos X: NGWFNCD
 ===>  AZKIELD         DCZEQFI         QPVQUBJ         VUSUXNB

Pos Y: HLXBXHS  Pos Z: DFFNEBO
       POOXKRG         WVYKPUA
```

We find 4 isomorphs, all with the pattern `1234567`. All four yield a contradiction with the involutory property (a “crash”): For position `B` the letter `Q` crashes, for position `K`, `R`, for position `Q`, `T`, for position `U`, `I`.

The analoguous result for Rotor II is:

```
Pos A: TPNTALS  Pos B: VRCWPNF  Pos C: YTUFHPG  Pos D: HWHAWSO
       LKVDRFK         AZBRNAM         NMQNFOO         CBIFUKR

Pos E: CFIORBU  Pos F: QAQKZWJ  Pos G: MOWCSKB  Pos H: EKLRYGQ
       UTXUHCA         HGSHWRV   ===>  IHAWOEJ         QPTOBTF

Pos I: TCDEFYL  Pos J: GRSTUNT  Pos K: VENLCAM  Pos L: NTVYEPS
 ===>  WVZBCLX         LKGCKYM         DCVKQZZ   ===>  SRDQFHO

Pos M: ALOZGHZ  Pos N: BYUHJUO  Pos O: JZBNSVW  Pos P: PHQCNDY
       NMFFXNG         VUHXMCT         ONKMHUU         UTTHPJC

Pos Q: ENYUBJA  Pos R: WCAJXYD  Pos S: LUCEPQM  Pos T: GJFMEFH
       BAOPIEI   ===>  QPCIOMX         YXYOVFP         AZQVKLE

Pos U: OEOFRAV  Pos V: HMJLGIR  Pos W: NFXSYBJ  Pos X: ULTHLHY
       CBFKSSZ         FESSUHH   ===>  ONHUWPA         JIZWZRG

Pos Y: JSLPMOL  Pos Z: RHARUDA
       XWMZITN         TSNIDWC
```

We find 5 isomorphs, again all with the pattern `1234567`. All five contradict an involution.

Finally for Rotor III:

```
Pos A: OAFNPWZ  Pos B: PMSOMIS  Pos C: QNBRJJB  Pos D: TOUOGKC
       XWJRURV         CBQUHOH         FENHRRI   ===>  SRKRWEJ

Pos E: QRDRSNJ  Pos F: TOEETKG  Pos G: GRLOUND  Pos H: QEITVAA
       BAHWZOM         UTTZMTJ         DCUMVWM         EDVVOJZ

Pos I: VOFWWKM  Pos J: YTCJXPN  Pos K: LWOSNSO  Pos L: UJPLZFP
       LKWOXSJ         IHXXYLO         FEYYFUR         CBOFCVE

Pos M: NSQUAOQ  Pos N: WLRVBHR  Pos O: XUSCEQH  Pos P: EVTZBRT
       ONACZCN         POBZWZG         QPCWIWP         RQFIJTQ

Pos Q: BCJWEYU  Pos R: YZVTRVV  Pos S: VWWFBSY  Pos T: HTXGGPV
 ===>  SRCJKFX         TSFKLGU         JISLMHR         VUCMNIO
```

```
Pos U: IFAHJBY  Pos V: JGXIWCL  Pos W: KHAJFDV  Pos X: LINKYEA
 ===>  WVHNDJA         XWKDPKB         AZXPQAC   ===>  XWGQRMD

Pos Y: MJXAHFD  Pos Z: CKCMIGQ
       AZZRUNE         NMIUROF
```

This time we find 4 isomorphs. Only the last one is compatible with an involution. It gives us 7 cycles of the involution $\tilde{\pi}$: `(AD)(EM)(GN)(IW)(KQ)(LX)(RY)`, the letters `BCFHJOPSTUVZ` remaining.

If our assumption on the probable word `GENERAL` was correct, then the fast rotor is Rotor III with initial position `X`. Now we use the lookup table for the involution $\tilde{\pi}$ containing all $2 \times 26^2 = 1318$ possibilities for Rotors I and II in each order and all initial positions. This is the file `vReflB_tr.xls` in the directory `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Files/`. There is exactly one involution that contains the obligatory cycles: The slow rotor 3 is Rotor I in initial position `H`, and the medium rotor is Rotor II in initial position `D`. Trying these settings on the online simulation at `http://enigmaco.de/` we obtain the plaintext

```
General Franco llegara a Sevilla en la noche. Notifica al
alcalde.
```

For successfully cryptanalyzing the Enigma without plugboard we only needed a short cryptogram (54 letters) and a few letters (only 7) of known plaintext. The attack by isomorphs is quite strong.

Compared with the attack on a linearly ordered ("straight-through") rotor machine the reflecting rotor *reduces* the workload because the involutory property excludes most isomorphs. On the other hand stripping off the last rotor is easier with a straight-through machine. But in summary the reflecting rotor turns out to be an illusory complication.

6.6 Message Key Analysis by Rejewski

The German Army adopted the Enigma in 1930 as Enigma I. In the first years this variant of the Enigma also had three rotors only—as had the commercial Enigma—but had the rotors wired in another way. Furthermore the additional plugboard, sitting between in/output and the rotors, substantially increased the key space, see Section 6.2.

The crucial point for the first break-in by the Polish cryptanalysts was a weakness in key handling:

- The key consisted of a daily basic setting and an individual message key.

- The daily basic setting consisted of the rotor order, the ring positions, and the plug connections—first at most 6 plugs—as well as an initial position of the rotors. This setting was valid for all messages of the day—in the first years even for several days. It was known to all participants of the communication network.
- The message key consisted of the initial positions of the three rotors. These could be changed quickly and were to be set by the operator in a random way. This key changed with every message and thereby precluded the alignment in depth of all the messages encrypted with the same daily basic setting.
- The receiver of the message knew the basic setting but not the message key. Therefore the operator encrypted the message key, consisting of three letters, with the basic setting and prefixed this three-letter-cryptogram to the message. This is no diminution of security as long as the keys are selected in a purely random way. In practice they were not.
- Because the radiocommunication was interference-prone, and a distorted key would garble the entire message, the message key was encrypted twice. Thus the proper message had a six-letter prefix. Adding redundancy to a message is not good idea in classical cryptography.

The operator hence had to encrypt six letters, a repeated trigram, using the basic setting, then to set the message key—the rotor positions—and then to encrypt the proper message.

The Polish intercepted the encrypted radio messages of the German Army but couldn't read them—until in 1932 they hired the mathematician Rejewski and his colleagues Różicky und Zygalski.

We describe their approach following Bauer's book [1] whose presentation relies on Rejewski's own description. At first we disregard the obstruction of the analysis that is caused by the (unknown) ring setting, that is, by the unknown stepping of the middle and maybe also the slow rotor.

Some Intercepted Messages

Suppose the first six letters of each of 65 intercepted messages from a single day were (in alphabetic order)

```
AUQ AMN | IND JHU | PVJ FEG | SJM SPO | WTM RAO
BNH CHL | JWF MIC | QGA LYB | SJM SPO | WTM RAO
BCT CGJ | JWF MIC | QGA LYB | SLM SPO | WTM RAO
CIK BZT | KHB XJV | RJL WPX | SUG SMF | WKI RKK
DDB VDV | KHB XJV | RJL WPX | SUG SMF | XRS GNM
EJP IPS | LDR HDE | RJL WPX | TMN EBY | XRS GNM
FBR KLE | LDR HDE | RJL WPX | TMN EBY | XOI GUK
GPB ZSV | MAW UXP | RFC WQQ | TAA EXB | XYW GCP
HNO THD | MAW UXP | SYX SCW | USE NWH | YPC OSQ
HNO THD | NXD QTU | SYX SCW | VII PZK | YPC OSQ
HXV TTI | NXD QTU | SYX SCW | VII PZK | ZZY YRA
IKG JKF | NLU QFZ | SYX SCW | VQZ PVR | ZEF YOC
IKG JKF | OBU DLZ | SYX SCW | VQZ PVR | ZSJ YWG
```

Two observations catch the eye:

1. Frequently even different operators use the same message keys. This could hint at certain stereotypes. Looking for different messages with the same six-letter prefix a coincidence calculation shows that they in fact are encrypted with the same key.

2. The repetition of the three letters of the message key is obvious: If two messages coincide in the first letters, then also their fourth letters coincide. For example a `Z` at position 1 implies a `Y` at position 4. The same holds for positions 2 and 5 (`U` implies `M`) and 3 and 6 (`W` implies `P`).

Therefore the handling of the message keys could be detected from the pure ciphertext, if it was not known already. In any case the cryptanalyst has a lot of ciphertext in depth: The first six letters of each message. If according to the operating instructions the message keys were randomly selected, this observation wouldn't be of much use. However, as it turned out, the message keys were non-random!

REJEWSKI's **Approach**

REJEWSKI started his analysis by looking at the repeated message keys. Suppose

- $a_1a_2a_3$ is the message key, hence the plaintext starts with the six letters $a_1a_2a_3a_1a_2a_3$.

- The ciphertext starts with the six letters $c_1c_2c_3c_4c_5c_6$.

- The first six Enigma substitutions, starting with the basic setting (+ the first rotor stepping before the first letter is encrypted), are $\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_6$.

Then we have

$$\begin{array}{llll} c_1 = \rho_1 a_1, & c_4 = \rho_4 a_1, & a_1 = \rho_1 c_1, & c_4 = \rho_4 \rho_1 c_1 \\ c_2 = \rho_2 a_2, & c_5 = \rho_5 a_2, & a_2 = \rho_2 c_2, & c_5 = \rho_5 \rho_2 c_2 \\ c_3 = \rho_3 a_3, & c_6 = \rho_6 a_3, & a_3 = \rho_3 c_3, & c_6 = \rho_6 \rho_3 c_3 \end{array}$$

Figure 6.3 illustrates this situation.

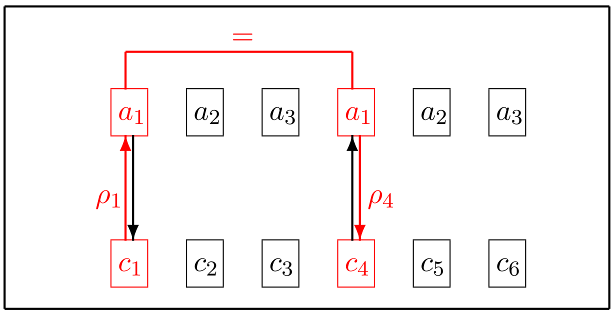


Figure 6.3: *Repeated message key*

The combined permutations $\tau_1 = \rho_4\rho_1$, $\tau_2 = \rho_5\rho_2$, $\tau_3 = \rho_6\rho_3$ are known if we have enough different message keys. In the example the 40 different six-letter groups completely determine τ_1:

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

and τ_2:

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

and τ_3:

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

In REJEWSKI's terminology the triple (τ_1, τ_2, τ_3) was called the **characteristic of the day**.

However we are far from knowing $\rho_1, \ldots, \rho_6$, and far from knowing the basic setting, or even a single message key!

At first sight the plugboard makes trouble. But REJEWSKI as a mathematician knew that the Enigma substitutions with or without plugboard differ only by conjugation with the plugboard substitution η. Therefore there is an *invariant* immune to the effect of the plugboard: the cycle type of the permutations τ_1, τ_2, τ_3, see Appendix A. The cycle decompositions are

τ_1 :	`(A)(BC)(DVPFKXGZYO)(EIJMUNQLHT)(RW)(S)`	of type $[10, 10, 2, 2, 1, 1]$
τ_2 :	`(AXT)(BLFQVEOUM)(CGY)(D)(HJPSWIZRN)(K)`	of type $[9, 9, 3, 3, 1, 1]$
τ_3 :	`(ABVIKTJGFCQNY)(DUZREHLXWPSMO)`	of type $[13, 13]$

From this point the analysis has two possible continuations:

- Assume the rotor wirings are unknown. The cryptanalyst assumes that the message keys are chosen in a stereotypic way—an assumption that in the case of the Wehrmacht-Enigma turned out to be true, see below. This assumption and the material delivered be a German spy and containing the basic settings for a few days including the plug connections enabled RÓŚICKY to derive the wiring of the fast rotor. Since the basic settings changed, each rotor sometimes occupied position 1, so eventually the wirings of all three rotors became known.

- Assume the wirings are known. Then the basic setting can be completely determined and all the messages of the day can be decrypted.

These approaches lead to successes, but not always. REJEWSKI and his colleagues also found some other attack methods, in particular using known plaintext. Here we omit this.

Determining the Enigma Substitution from the Characteristics of the Day

We return to our example and try to determine the first six Enigma substitutions in basic setting, $\rho_1, \ldots, \rho_6$, from the known products $\tau_1 = \rho_4\rho_1$, $\tau_2 = \rho_5\rho_2$, $\tau_3 = \rho_6\rho_3$ whose cycle decomposition is given above. We start with the schema

```
(A) (BC) (DVPFKXGZYO)
(S) (WR) (THLQNUMJIE)

(D) (AXT) (BLFQVEOUM)
(K) (YGC) (NRZIWSPJH)

(ABVIKTJGFCQNY)
(OMSPWXLHERZUD)
```

see Appendix A. We immediately coinclude that ρ_1 and ρ_4 both have the 2-cycle `(AS)`, and ρ_2 and ρ_5 both have the 2-cycle `(DK)`. But even for the 2-cycles of τ_1 we don't get a unique solution: ρ_1 could have the cycles `(BW)(CR)` and ρ_4 the cycles `(BR)(CW)`, or conversely.

To get on we assume—following Rejewski—that `aaa` is the most popular message key with the German operators. (If this would turn out as erroneous we would try some other stereotype.) If we are right, then this corresponds to the encrypted message key `SYX SCW` that occurs five times, and implies the cycles

`(AS)` in ρ_1, `(AS)` in ρ_4,
`(AY)` in ρ_2, `(AC)` in ρ_5,
`(AX)` in ρ_3, `(AW)` in ρ_6.

This is nothing new for ρ_1 and ρ_4. But for τ_2 it means that the alignment of the 3-cycles is correct, and we read off the 2-cycles

`(AY)(XG)(TC)` in ρ_2, `(AC)(GT)(XY)` in ρ_5.

For τ_3 the correct alignment is

```
(ABVIKTJGFCQNY)
(XLHERZUDOMSPW)
```

and we find the unique solution

ρ_3 = `(AX)(BL)(CM)(DG)(EI)(FO)(HV)(JU)(KR)(NP)(QS)(TZ)(WY)`
ρ_6 = `(AW)(BX)(CO)(DF)(EK)(GU)(HI)(JZ)(LV)(MQ)(NS)(PY)(RT)`

Now let's look at other encrypted message keys. The first one in our table is `AUQ AMN`, partially decrypting to the plaintext

```
s?s s?s
```

We suspect the stereotypical message key `sss`. If we are right, then ρ_2 has the 2-cycle `(SU)`, and ρ_5 has the 2-cycle `(MS)`. This gives the correct alignment of the 9-cycles ot τ_2:

```
(D) (AXT) (BLFQVEOUM)
(K) (YGC) (JHNRZIWSP)
```

and completely determines ρ_2 and ρ_5:

ρ_2 = `(AY)(BJ)(CT)(DK)(EI)(FN)(GX)(HL)(MP)(OW)(QR)(SU)(VZ)`
ρ_5 = `(AC)(BP)(DK)(EZ)(FH)(GT)(IO)(JL)(MS)(NQ)(RV)(UW)(XY)`

The encrypted message key `RJL WPX` occurs four times, and partially decrypts as

```
?bb ?bb
```

Again we are quite sure that this reveals a stereotypical message key: `bbb`. We conclude that ρ_1 has the cycle `(BR)`—hence also the cycle `(CW)`—and ρ_4 has the cycle `(BW)`, hence also the cycle `(CR)`.

For the complete solution the only open problem left is the alignment of the two 10-cycles of τ_1. We look at the group `LDR HDE` and partially decrypt it as

```
?kk ?kk
```

We are quite sure of the message key `kkk`. Then ρ_1 has the 2-cycle `(KL)`, the correct alignment is

```
(A) (BC) (DVPFKXGZYO)
(S) (RW) (IETHLQNUMJ)
```

and the complete solution is

$$\begin{aligned}\rho_1 &= \texttt{(AS)(BR)(CW)(DI)(EV)(FH)(GN)(JO)(KL)(MY)(PT)(QX)(UZ)}\\ \rho_4 &= \texttt{(AS)(BW)(CR)(DJ)(EP)(FT)(GQ)(HK)(IV)(LX)(MO)(NZ)(UY)}\end{aligned}$$

Now we can decrypt all message keys for the actual basic setting. However we do not yet know the basic setting itself, and we cannot decrypt a single message. In particular we do not know the ring setting and the positions of the rotors corresponding to the message keys.

Rejewski's Catalogue

In our example the permutations $\tau_1 = \rho_4\rho_1$, $\tau_2 = \rho_5\rho_2$, and $\tau_3 = \rho_6\rho_3$ are completely determined and their cycle types are the partitions

```
[10 10 2 2 1 1], [9 9 3 3 1 1], [13 13]
```

of the number 26. Now we ask how characteristic is this triple of partitions for the basic setting of the Enigma. The plug connections are irrelevant for this problem. We consider the rotor order as an element of the permutation group $\mathcal{S}_3$, and the initial positions of the three rotors as elements of the cyclic group $\mathbb{Z}/26\mathbb{Z}$. If we disregard the plugboard and the ring settings, the possible basic settings form the set $\mathcal{S}_3 \times (\mathbb{Z}/26\mathbb{Z})^3$. On the other hand we have the set $\mathcal{P}_{13}$ consisting of all the 101 partitions of the number 13 (in bijective correspondence with the partitions of the number 26 in pairwise equal parts), and we have a map

$$\mathcal{S}_3 \times (\mathbb{Z}/26\mathbb{Z})^3 \longrightarrow (\mathcal{P}_{13})^3$$

We would like this map to be injective. This seems not unrealistic in view of the cardinalities: 105,456 different basic settings, $101^3 = 1{,}030{,}301$ different partitions.

To get the complete value table of this map REJEWSKI designed a simple Enigma simulator called Cyclometer that run through all basic settings in about one year. The result, called REJEWSKI's Catalogue, got lost. But there is a recent reconstruction in the paper

Alex KUHL: Rejewski's Catalog. Cryptologia 31 (2007), 326–331.

It turned out that the above map is *not* injective, but "almost" so: Many triples of partitions have a unique preimage, most have only a few ones. However a few triples occur quite frequently, the top ten being

Triple of partitions			**Frequency**
[13 13]	[13 13]	[13 13]	1771
[12 12 1 1]	[13 13]	[13 13]	898
[13 13]	[13 13]	[12 12 1 1]	866
[13 13]	[12 12 1 1]	[13 13]	854
[11 11 2 2]	[13 13]	[13 13]	509
[13 13]	[12 12 1 1]	[12 12 1 1]	494
[13 13]	[13 13]	[11 11 2 2]	480
[12 12 1 1]	[13 13]	[12 12 1 1]	479
[13 13]	[11 11 2 2]	[13 13]	469
[12 12 1 1]	[12 12 1 1]	[13 13]	466

All in all there are 21230 different triples in the image of the map. 19604 of these, that is 92%, occur at most ten times, the numbers of these rare triples are

Pre-Im	1	2	3	4	5	6	7	8	9	10
Freq	11466	3381	1658	958	660	456	343	265	234	183

Using the catalogue the Polish cryptanalysts usually found the correct basic setting in at most 20 minutes. It is unknown what they did in the exceptional situations where there are too many false positives. Certainly some other useful details could be used. In any case we may assume that the method was successful for at least 92% of all triples, corresponding to roughly 50% of all cases.

We neglected the effect of the ring setting. This causes a rotor movement because the stepping mechanism is connected with the alphabet ring. Now, what could happen? As long as only the fast rotor moves we are in a situation included in the catalogue. The analysis is hampered if the middle rotor moves between two of the first six letters. The chances are 5 of 26 ring settings, that is about 19%. This lowers the total probability of success from 50% to about 40%.

There is even more potential for drawing conclusions from the collected message keys. For example the moving of the middle rotor gives information about the ring setting of the first rotor. An approach to determining the plugboard connections uses the fact that in the first years at most six letter pairs were interchanged. If the cryptanalysts assume that there are no plugs at all, then some true plaintext shows through the tentatively decrypted text. This enables them to reconstruct the plugboard connections.

Epilogue

The plugboard turns out to be an illusory complication: It slows the cryptanalyst down a bit, but not as much as the increase in keylength from 29 to 76 bits—expressed in terms of today—suggested. The main cost of the cryptanalysis is exhausting the rotor order and positions, and this could be made efficient by compiling lookup tables.

By the way the decrypted 40 different message keys from the list of 65 above are:

AUQ AMN	sss	IKG JKF	ddd	QGA LYB	xxx	VQZ PVR	ert
BNH CHL	rfv	IND JHU	dfg	RJL WPX	bbb	WTM RAO	ccc
BCT CGJ	rtz	JWF MIC	ooo	RFC WQQ	bnm	WKI RKK	cde
CIK BZT	wer	KHB XJV	lll	SYX SCW	aaa	XRS GNM	qqq
DDB VDV	ikl	LDR HDE	kkk	SJM SPO	abc	XOI GUK	qwe
EJP IPS	vbn	MAW UXP	yyy	SUG SMF	asd	XYW GCP	qay
FBR KLE	hjk	NXD QTU	ggg	TMN EBY	ppp	YPC OSQ	mmm
GPB ZSV	nml	NLU QFZ	ghj	TAA EXB	pyx	ZZY YRA	uvw
HNO THD	fff	OBU DLZ	jjj	USE NWH	zui	ZEF YOC	uio
HXV TTI	fgh	PVJ FEG	tzu	VII PZK	eee	ZSJ YWG	uuu

The astonishingly naive habits of the German cipher operators become obvious by looking at the keyboard layout of Enigma:

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

All message keys belong to one of three groups of stereotypes

- iterated letters: `sss`, `fff`, `ddd`, `ooo`, ...
- three consecutive keys: `rfv`, `rtz`, `wer`, `ikl`, ...
- three letters in alphabetic order: `abc`, `uvw`

Before World War II the British cryptanalysts failed with the cryptanalysis of Enigma because they tried to determine the wiring between in-/output and first rotor. The commercial Enigma D connected `Q` with `A`, `W` with `B`, `E` with `C` and so on in the order of the keyboard. Assuming this for Enigma I didn't work. REJEWSKI who knew the Germans since he was a student at Göttingen simply assumed that the wiring in any case should follow a simple scheme, and succeeded with the assumption "`A` is connected to `A`, `B` to `B` etc."

The point: Enigma C also had had this simple wiring, and this information could be found in the patent file in the British Patent Office ...

For later attacks (from 1938 on) of the Polish cryptanalysts against the Enigma, including a complete example, see the paper

> David LINK, Resurrecting *Bomba Kryptologiczna*: Archeology of Algorithmic Artefacts, I. Cryptologia 33 (2009), 166–182.

6.7 Wehrmacht Enigma and Known Plaintext

The Polish break into the Enigma relies on the way in which the German operators handled the message keys. With the beginning of the war the method of message keying changed and the pre-war cryptanalytic approaches broke down.

Equations for Known Plaintext

Already the Polish cryptanalysts had exlored the idea of using known plaintext—starting from the observation that the German military in their messages used a lot of stereotypical phrases such as "Heil Hitler" or "Oberkommando der Wehrmacht" (= Army's High Command). Chunks of known plaintext (called "cribs" by the british cryptanalysts) allow narrowing down the exhaustive search to an amount that eventually may be mastered with the help of some cleverly constructed electro-mechanical machines. Alan TURING largely and systematically expanded this approach.

Here is an example (Example 1, taken from [4] as virtually all authors of cryptographic texts do). Let the ciphertext

```
ULOEB ZMGER FEWML KMTAW XTSWV UINZP R ...
```

be given. We suppose the message contains the phrase "Oberkommando der Wehrmacht" near the beginning. A negative pattern search over the first 12 possible positions yields exactly one hit:

```
U L O E B Z M G E R F E W M L K M T A W X T S W V U I N Z P R
o b e r k o = m a n d o d e r w e h r m a c h t
  o b = r k o m m a n d o d e r w e h r m a c h t
    = b e r k o m m a n d o d e r w e h r m a c h t
      o = e r k o m m a n d o d e r w e h r m a c h t
        o b e r k o m m a n d o d e r = e h r m a c h t
===>      o b e r k o m m a n d o d e r w e h r m a c h t
            o b = = k o m = a n d o d e r w e h r m a c h t
              o b e r k o = m a n d o d e r w e h r m a c h t
                o b e r k o m m a n d o d e r = e h r m a c h
                  o b = r k o m = a n d o d e r w e h r m a c
                    o b e r k o = m = n d o d e r w e h r m a
                      o b e r = o m m a n d o d e r w e h r m
```

We assume the rotor wirings of all five rotors as known. The naive approach—exhaustion by brute force and assuming that the ring settings don't interfere with the crib—would go through all 60 possible rotor orders, all $26^3 = 17576$ start positions, and all $> 10^{14}$ plug configurations, each time decrypt the ciphertext, and look if the known plaintext results. The huge number of plug configurations makes this approach hopeless, the "virtual" keylength for this approach being about 67 bits ($10^{23}/26^2 \approx 1.6 \cdot 10^{20} \approx 2^{67}$). (We first neglect the ring settings that have little impact on the cryptanalysis.)

Fortunately, using known plaintext, we may find conditions that involve *only a single plug*. Recall the general situation as shown in Figure 6.4.

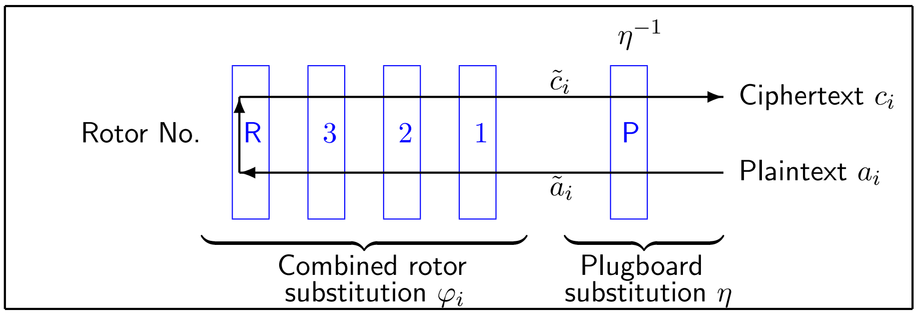


Figure 6.4: *Enigma with plugboard*

Assume a sequence $a_1 \ldots a_m$ of known plaintext is given with corresponding ciphertext $c_1 \ldots c_m$, the respective combined rotor substitutions being $\varphi_1, \ldots, \varphi_m$ and the "full" Enigma substitutions, $\rho_i = \eta^{-1}\varphi_i\eta$. This gives the equations

$$\begin{aligned} c_1 = \rho_1 a_1 &= \eta^{-1}\,\varphi_1\,\eta\,a_1 \\ &\vdots \\ c_m = \rho_m a_m &= \eta^{-1}\,\varphi_m\,\eta\,a_m \end{aligned}$$

or $\eta c_i = \varphi_i \eta a_i$. Denoting the image of a letter under the (fixed but unknown) plugboard substitution by a tilde we get:

Lemma 7 *For a sequence $a_1 \ldots a_m$ of known plaintext we have*

$$\tilde{c}_i = \varphi_i\,\tilde{a}_i \quad \textit{and} \quad \tilde{a}_i = \varphi_i\,\tilde{c}_i \quad \textit{for } i = 1, \ldots, m.$$

For the second equation we used the fact that the combined rotor substitutions φ_i are involutions.

Looking for Cycles

Returning to Example 1 we consider the special situation

```
                      1 1 1 1 1 1 1 1 1 1 2 2 2 2 2
  i   = 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4
  c_i = Z M G E R F E W M L K M T A W X T S W V U I N Z
  a_i = O B E R K O M M A N D O D E R W E H R M A C H T
```

From such a plaintext-ciphertext pair we extract the Turing **graph**: The nodes correspond to the letters A ... Z of the standard alphabet. For each pair (a_i, c_i) of plaintext letter and corresponding ciphertext letter an edge is drawn between these two letters, and this edge is labeled by the index i. Due to the reciprocity between plaintext and ciphertext, the situation is modeled by an undirected graph. An edge with label j between nodes s and t means that $t = \rho_j s$ and $s = \rho_j t$—or $\tilde{t} = \varphi_j \tilde{s}$ and $\tilde{s} = \varphi_j \tilde{t}$. Figure 6.5 shows the Turing graph for Example 1.

Turings approach uses the cycles in this graph ("closures" in Turing's way of speaking). In the notation of Lemma 7 we find:

$$\mathtt{E} = \rho_7\,\mathtt{M}, \quad \mathtt{M} = \rho_9\,\mathtt{A}, \quad \mathtt{A} = \rho_{14}\,\mathtt{E}, \quad \text{and} \quad \tilde{\mathtt{E}} = \varphi_7\,\tilde{\mathtt{M}}, \quad \tilde{\mathtt{M}} = \varphi_9\,\tilde{\mathtt{A}}, \quad \tilde{\mathtt{A}} = \varphi_{14}\,\tilde{\mathtt{E}},$$

and combine these three equations into one **cycle equation**

$$\mathtt{E} = \rho_7\,\rho_9\,\rho_{14}\,\mathtt{E}. \quad \text{and} \quad \tilde{\mathtt{E}} = \varphi_7\,\varphi_9\,\varphi_{14}\,\tilde{\mathtt{E}}.$$

In general we have:

Theorem 11 (Fixed Point Theorem of Rejewski/Turing) *Let ρ_i be the Enigma substitution in position i, and $\varphi_i = \eta\rho_i\eta^{-1}$ be the substitution without plugs. Then a letter a is a fixed point of a composition $\rho_{i_1} \cdots \rho_{i_k}$ if and only if the plugged letter $\tilde{a}$ is a fixed point of $\varphi_{i_1} \cdots \varphi_{i_k}$.*

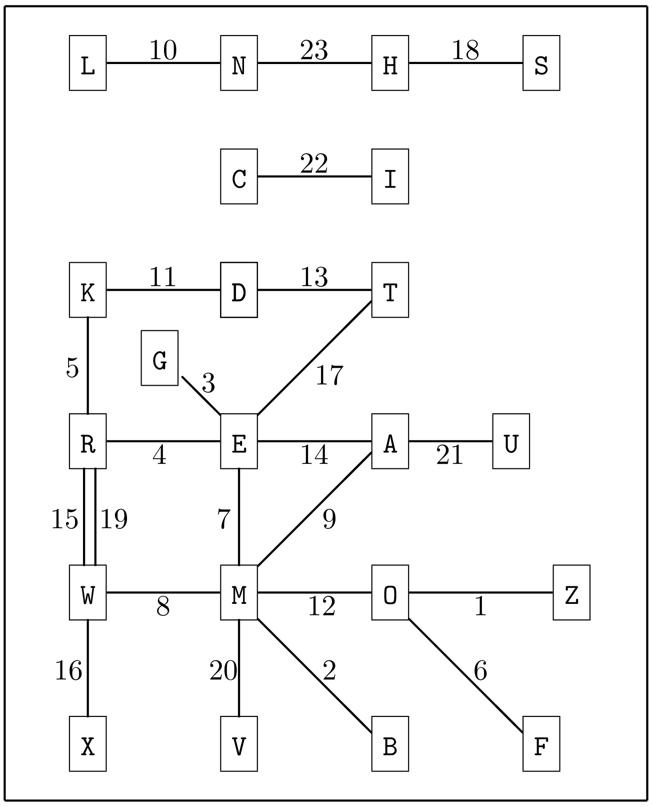


Figure 6.5: TURING *graph for Example 1*

Thus the fixed point property of a cycle is in a certain sense independent of the plug connections.

Corollary 7 (TURING's cycle condition) *Each loop in the* TURING *graph gives a necessary condition for the correct key of the Enigma encryption in the form*

$$\tilde{a} = \varphi_{i_1} \dots \varphi_{i_k} \tilde{a}$$

for a letter a. In particular $\tilde{a}$ is a fixed point of the corresponding composition of unplugged Enigma substitutions.

Although mathematically trivial this theorem and its corollary are the keys to eliminating the complexity of the plugboard by a meet-in-the-middle attack.

What is the benefit of TURING's cycle condition? Suppose in Example 1 we try all 26 possible values for $\tilde{\texttt{E}} = \eta\,\texttt{E}$ and all 26^3 possible rotor positions for all 60 possible rotor orders, searching for fixed points of $\varphi_7\,\varphi_9\,\varphi_{14}$—an exhaustion of $60 \times 26^4 = 27,418,560$ cases. Then the probability that the cycle condition is fulfilled is about $1/26$. This rules out $\approx 25/26 \approx 96\%$ of all cases and leaves us with $\approx 60 \times 26^3$ cases—not really impressive, but it could be a good start: Suppose we find two cycles involving E, then we are left with $\approx 60 \times 26^2$ cases, for three cycles with $\approx 60 \times 26$ cases, for four cycles with ≈ 60 cases, i. e. with the exhaustion of the possible rotor orders. And the outcome of this search is:

- The correct initial rotor positions for our known plaintext
- The correct plugboard images for all letters that occur in one of the cycles—a significant part of the complete plug configuration

Now in our Example 1 (that is in fact DEAVOUR's and KRUH's) we see two other cycles involving E:

$$\tilde{\texttt{E}} = \varphi_4\,\tilde{\texttt{R}}, \quad \tilde{\texttt{R}} = \varphi_{15}\,\tilde{\texttt{W}}, \quad \tilde{\texttt{W}} = \varphi_8\,\tilde{\texttt{M}}, \quad \tilde{\texttt{M}} = \varphi_7\,\tilde{\texttt{E}},$$

and

$$\tilde{\texttt{E}} = \varphi_4\,\tilde{\texttt{R}}, \quad \tilde{\texttt{R}} = \varphi_5\,\tilde{\texttt{K}}, \quad \tilde{\texttt{K}} = \varphi_{11}\,\tilde{\texttt{D}}, \quad \tilde{\texttt{D}} = \varphi_{13}\,\tilde{\texttt{T}}, \quad \tilde{\texttt{T}} = \varphi_{17}\,\tilde{\texttt{E}},$$

giving the two additional cycle conditions

$$\tilde{\texttt{E}} = \varphi_4\,\varphi_{15}\,\varphi_8\,\varphi_7\,\tilde{\texttt{E}}, \quad \tilde{\texttt{E}} = \varphi_4\,\varphi_5\,\varphi_{11}\,\varphi_{13}\,\varphi_{17}\,\tilde{\texttt{E}}.$$

The complete cycle constellation may be visualized by Figure 6.6.

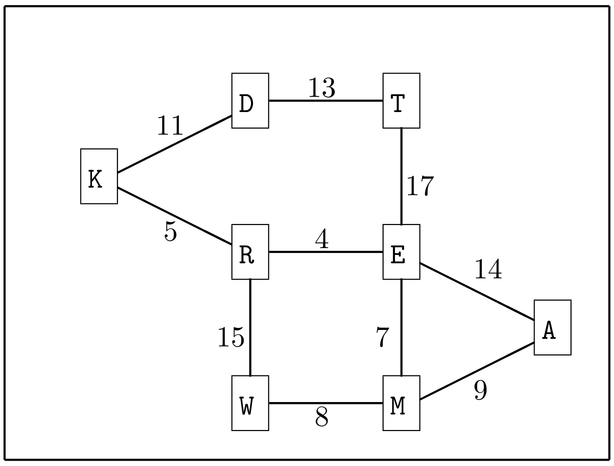


Figure 6.6: TURING *cycles in Example 1*

Evaluating the Cycle Conditions

In evaluating the cycle conditions one sets the rotors to start positions and then steps Rotor 1 only. In lucky cases also in the real situation only Rotor 1 moves. In bad cases Rotor 2 moves, maybe even Rotor 3. Since the ring setting is unknown, these stepping positions are unknown. Because in the example all the cycles are between plaintext positions 4 and 17, the length of the effectively used plaintext segment is 14, and the probability for a stepping of Rotor 2 in between is 13/26 = 50%, a stepping that would invalidate the approach, and a good argument for using rather short cribs.

Now assume that we have identified the correct rotor order and the correct initial positions of all the rotors, and no interfering movement of Rotors 2 and 3 occurs for the involved plaintext section $a_1 \dots a_m$. Then the combined rotor substitutions $\varphi_1, \dots, \varphi_m$ are known, and the plug image $\tilde{s} = \eta s$ is known for all letters s that occur in the cycles. In the example we know $\tilde{\mathtt{E}} = \eta\mathtt{E}$ and consequently

$$\tilde{\mathtt{R}} = \varphi_4\tilde{\mathtt{E}}, \quad \tilde{\mathtt{K}} = \varphi_5\tilde{\mathtt{R}}, \quad \tilde{\mathtt{M}} = \varphi_7\tilde{\mathtt{E}}, \quad \tilde{\mathtt{W}} = \varphi_8\tilde{\mathtt{M}}, \quad \tilde{\mathtt{A}} = \varphi_9\tilde{\mathtt{M}},$$

$$\tilde{\mathtt{D}} = \varphi_{11}\tilde{\mathtt{K}}, \quad \tilde{\mathtt{O}} = \varphi_{12}\tilde{\mathtt{M}}, \quad \tilde{\mathtt{T}} = \varphi_{13}\tilde{\mathtt{D}}, \quad \tilde{\mathtt{X}} = \varphi_{16}\tilde{\mathtt{W}}.$$

Furthermore we find $\tilde{\mathtt{F}} = \varphi_6\tilde{\mathtt{O}}$. Since η is an involution the inverse relations might involve further letters. That is we know the plugboard substitutes of at least 11 letters.

What is yet missing is

- The plugboard substitutes of the remaining letters
- The stepping position of Rotor 2

To continue assume first that the remaining letters are unchanged by the plugboard and decrypt $c_{m+1}, \ldots$ As soon as the resulting plaintext is unreadable either a new plugboard connection or the stepping position is detected. If the crib occurred in the middle of the ciphertext, we run the same procedure backwards to the beginning of the message.

Conclusion

The huge number of possible plug settings turns out to be an illusory complication: The exhaustion used the plug connection of a single letter only. In good cases where the procedure yields a unique solution of the cycle conditions the effort was testing 26 plug connections with 26^3 start positions for each of the 60 rotor orders, that is $27,418,560 \approx 1.6 \cdot 2^{24}$ cases. In each case we have to do some trial encryptions for the letters in the cycles plus some house-keeping plus some finishing. So we may guess that the search space is dropped to about 30 bits.

As soon as the daily key—rotor order, ring settings, plug connections, initial positions of the rotors—is known, reading all further messages of the day comes for almost no additional costs because all message keys are encrypted with the same initial rotor positions.

A Note on the Technical Realization: Turing's Bombe

Turing's Bombe consisted of a battery of several Enigmas (without plugboards), called "scramblers" and in one-to-one correspondence with the nodes of the Turing graph, synchronously stepping through all 26^3 rotor positions. For each edge two scramblers were connected by a cable, and set to start positions differing by the number that corresponded to the label of the edge. Therefore the physical arrangement of the components was an exact model of the graph. The cable had 26 wires, so all choices for the plug connection of a selected letter ($\tilde{\texttt{E}}$ in Example 1) could be tested in parallel. The cycle conditions corresponded to closed electrical circuits that made the bombe stop. Then the operator noted the actual rotor positions and restarted the bombe with the next set of positions.

Using enough scramblers even all the sixty rotor orders could be tested in parallel, dropping the effective search costs to 26^3, equivalent with a complexity of 14 bits only. A complete run of the bombe took 11 minutes. (Today a simulation on a PC without parallel execution takes about 5 minutes.)

Unfortunately in general the solution was far from unique, so the bombe produced a huge number of "false positive" stops. An idea of Welchman largely reduced the number of false positives by a clever add-on to the bombe, see Section 6.8 below, and this was crucial for the success of the British cryptanalysts against the Enigma.

6.8 Example 2

Now we go through an example step by step and produce a complete solution for the ciphertext

```
ZIDPV USABH HEABG RZMOP UWVJD MLPCS PFTSH ISJMR RFSKU KHUAT
SFDNB GWTAN CSZZW HPHNP DDSAX GTRGY OZPKO EAGRG YSGQD KKNIT
DWFZZ INSYH UTSZR KJDVJ JLJIJ MQHCB RINYI
```

Aligning Known Plaintext

We believe the plaintext contains "Oberleutnant zur See" as the rank of the sender, that is we assume a crib near the end of the message, and assume that at most 20 letters follow, containing the name. The scheme

```
      RGYSGQDKKNITDWFZZINSYHUTSZRKJDVJJLJIJMQHCBRINYI
[ 89] xstopxoberleutnantxzurxseex
[ 90]  xstopxoberleutnantxzurx=eex
[ 91]   x=topxoberleutna=txz=rxseex
[ 92]    xstopxoberleutnantxzurxseex
[ 93]     xstopxoberleut=antxzurxseex
[ 94]      xstopxoberleutnantxzu=xseex
[ 95]       xstopxoberleutnan=x=urxseex
[ 96]        xstopxoberleutnantxzurxseex
[ 97]         xstopxoberleutnantxzurxseex
[ 98]          xs=opxoberleutnantxzurxseex
[ 99]           xstopxoberle==nantxzurxseex
[100]            xstopxoberleutnantxzurxseex
[101]             xstopxoberleutnantxzurxseex
[102]              xstopxoberleutnantxzurxseex
[103]               xstopxoberleutnantxzurxseex
[104]                xstopxoberleutnantxzurxseex
[105]                 xstopxoberleutnantxzurxseex
[106]                  xstopxobe=leutnantxzurxseex
[107]                   x=topxoberleutnantxzurxseex
[108]                    xstopxoberleutnantxzurxseex
[109]                     xstopxoberleutnantxzurxseex
      RGYSGQDKKNITDWFZZINSYHUTSZRKJDVJJLJIJMQHCBRINYI
```

gives 12 hits for the negative pattern search among the 21 considered positions: 89, 92, 96, 97, 100, 101, 102, 103, 104, 105, 108, 109—at least a slight reduction for manual cryptanalysis.

Constructing a Turing Graph

Somewhere along the way we test position 103 and consider the crib

```
FZZINSYHUTSZRKJDVJJLJIJMQHC
xstopxoberleutnantxzurxseex
```

We derive the cycle diagram in Figure 6.7.

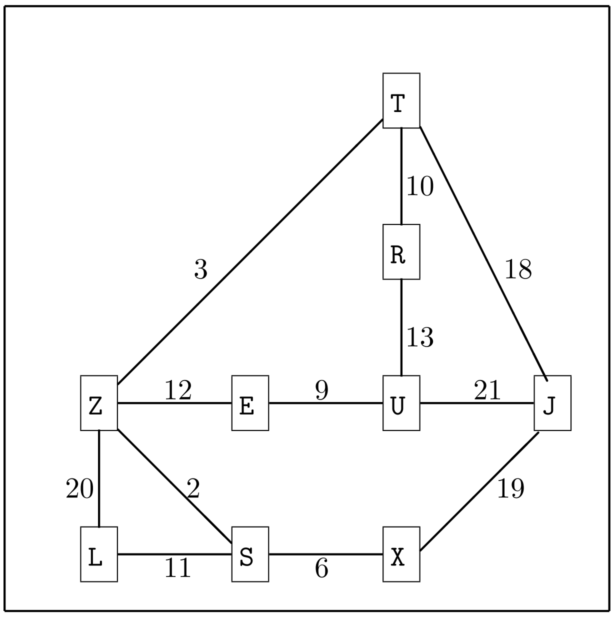


Figure 6.7: TURING *cycles for Example 2*

Therefore as "menu"—the chunk of known plaintext to be examined—we use the sequence of length 20 starting from position 104 (that corresponds to the edge with label 2):

```
ZZINSYHUTSZRKJDVJJLJ
STOPXOBERLEUTNANTXZU
```

To exhaust all the rotor orders, starting positions, and plug connections for this chunk of known plaintext we use Jean-François Bouchaudy's TURING Bombe Simulator, to be found at `http://cryptocellar.web.cern.ch/cryptocellar/simula/jfb/BP12.zip`.

> In a virtual machine on a 2.93 GHz Intel Core-i7 processor it needed 5 minutes for all 60 rotor orders and produced exactly one solution in "WELCHMAN mode" (the diagonal board, see later).

Using only the rotors I, II, and III and disabling the diagonal board—that we haven't introduced yet—we get 6 "solutions" in a few seconds

(1)	I II III KFX
(2)	I II III WHV
(3)	II I III ZYN
(4)	III I II JXS
(5)	III II I IES
(6)	III II I QSV

Exploring Solution (1)

Let us try the first proposed solution. We begin by decrypting the ciphertext with a ring setting that causes no stepping of the middle rotor for the next 20 positions, and no plugs in the plugboard. Missing plugs will be detected by the following considerations.

> The assumption on the ring setting is somewhat optimistic. It it fails for all of the solutions, we have to try harder, experimenting with shorter cribs or guessing the ring setting of the fast rotor.

We use the rotor order I (slow), II (middle), III (fast), and the start positions `KFX`. This gives the trial decryption

```
ZZINSYHUTSZRKJDVJJLJIJMQHCBRINYI
XPMEJJXPGQBMIVVUKRSISPTNFVAZEQTG
```

This doesn't look like plaintext, but we have not yet explored the plugs. We start with the plug connection $\tilde{\mathtt{Z}}$ of `Z`, the letter with the maximum number of edges in the graph. We try all 26 possible connections, see Table 6.1

Only line `X` is compatible with the cycle, giving $\tilde{\mathtt{Z}} = \mathtt{X}$. For a manual check of the other cycles we use the complete description of the combined rotor substitutions $\varphi_2, \ldots, \varphi_{21}$ given in Table 6.2. The “plugged” cycles fit “unplugged” ones:

$$\tilde{\mathtt{Z}} \xrightarrow{3} \tilde{\mathtt{T}} \xrightarrow{10} \tilde{\mathtt{R}} \xrightarrow{13} \tilde{\mathtt{U}} \xrightarrow{9} \tilde{\mathtt{E}} \xrightarrow{12} \tilde{\mathtt{Z}} \quad \text{fits} \quad \mathtt{X} \xrightarrow{3} \mathtt{I} \xrightarrow{10} \mathtt{Y} \xrightarrow{13} \mathtt{F} \xrightarrow{9} \mathtt{L} \xrightarrow{12} \mathtt{X}$$

$$\tilde{\mathtt{Z}} \xrightarrow{2} \tilde{\mathtt{S}} \xrightarrow{6} \tilde{\mathtt{X}} \xrightarrow{19} \tilde{\mathtt{J}} \xrightarrow{21} \tilde{\mathtt{U}} \xrightarrow{9} \tilde{\mathtt{E}} \xrightarrow{12} \tilde{\mathtt{Z}} \quad \text{fits}$$
$$\mathtt{X} \xrightarrow{2} \mathtt{Z} \xrightarrow{6} \mathtt{F} \xrightarrow{19} \mathtt{N} \xrightarrow{21} \mathtt{F} \xrightarrow{9} \mathtt{L} \xrightarrow{12} \mathtt{X}$$

$$\tilde{\mathtt{T}} \xrightarrow{10} \tilde{\mathtt{R}} \xrightarrow{13} \tilde{\mathtt{U}} \xrightarrow{21} \tilde{\mathtt{J}} \xrightarrow{18} \tilde{\mathtt{T}} \quad \text{fits} \quad \mathtt{I} \xrightarrow{10} \mathtt{Y} \xrightarrow{13} \mathtt{F} \xrightarrow{21} \mathtt{N} \xrightarrow{18} \mathtt{I}$$

Therefore the cycle conditions hold indeed.

However we didn't need to check them because reading off the plug connections from the first loop, row “X” in Table 6.1, we get $\tilde{\mathtt{Z}} = \mathtt{X}$, $\tilde{\mathtt{S}} = \mathtt{Z}$, and this already is a contradiction.

Therefore solution (1) was a false alarm. This observation leads to Welchman's **plug condition** expressing the fact that the plug substitution is an involution:

> *If $\tilde{a} = b$, then also $\tilde{b} = a$ for each pair of letters $a, b \in \Sigma$.*

$\tilde{Z}$	$\xrightarrow{2}$	$\tilde{S}$	$\xrightarrow{11}$	$\tilde{L}$	$\xrightarrow{20}$	$\tilde{Z}$
A		C		V		W
B		L		H		G
C		A		M		B
D		F		N		R
E		G		K		U
F		D		Z		E
G		E		T		A
H		O		R		N
I		V		C		P
J		M		A		T
K		U		W		V
L		B		I		F
M		J		P		C
N		S		Q		J
O		H		L		S
P		R		O		Y
Q		Y		X		D
R		P		J		Q
S		N		F		I
T		W		U		K
U		K		G		H
V		I		B		M
W		T		E		Z
X		Z		D		X
Y		Q		S		L
Z		X		Y		O

Table 6.1: Example 2—Possible plug connections for the first cycle

Substition in	Substitution table																									
rotor position	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
φ_2: KFX	C	L	A	F	G	D	E	O	V	M	U	B	J	S	H	R	Y	P	N	W	K	I	T	Z	Q	X
φ_3: KFY	D	C	B	A	Y	S	L	J	X	H	O	G	N	M	K	Z	R	Q	F	V	W	T	U	I	E	P
φ_4: KFZ	N	X	E	F	C	D	P	S	M	Q	U	Y	I	A	V	G	J	T	H	R	K	O	Z	B	L	W
φ_5: KFA	B	A	X	V	N	Y	K	Q	O	Z	G	M	L	E	I	U	H	T	W	R	P	D	S	C	F	J
φ_5: KFB	U	D	L	B	M	Z	O	Y	V	S	T	C	E	Q	G	W	N	X	J	K	A	I	P	R	H	F
φ_5: KFC	Z	U	O	T	X	H	L	F	P	Y	Q	G	V	S	C	I	K	W	N	D	B	M	R	E	J	A
φ_5: KFD	J	D	U	B	Y	Q	R	X	S	A	T	P	O	Z	M	L	F	G	I	K	C	W	V	H	E	N
φ_5: KFE	R	C	B	W	H	L	O	E	J	I	M	F	K	S	G	U	T	A	N	Q	P	X	D	V	Z	Y
φ_{10}: KFF	M	Z	H	X	W	P	T	C	Y	R	O	U	A	Q	K	F	N	J	V	G	L	S	E	D	I	B
φ_{11}: KFG	M	I	V	Z	T	N	K	L	B	P	G	H	A	F	R	J	S	O	Q	E	W	C	U	Y	X	D
φ_{12}: KFH	F	Z	R	W	V	A	T	I	H	Y	O	X	N	M	K	U	S	C	Q	G	P	E	D	L	J	B
φ_{13}: KFI	J	S	U	G	W	Y	D	K	L	A	H	I	R	P	Q	N	O	M	B	V	C	T	E	Z	F	X
φ_{14}: KFJ	V	Y	O	W	F	E	H	G	K	S	I	P	T	R	C	L	U	N	J	M	Q	A	D	Z	B	X
φ_{15}: KFK	F	R	W	K	Y	A	M	P	X	V	D	N	G	L	Q	H	O	B	U	Z	S	J	C	I	E	T
φ_{16}: KFL	B	A	I	V	J	S	H	G	C	E	Q	O	N	M	L	T	K	U	F	P	R	D	Z	Y	X	W
φ_{17}: KFM	R	J	I	O	K	Y	M	X	C	B	E	P	G	Q	D	L	N	A	Z	W	V	U	T	H	F	S
φ_{18}: KFN	R	Q	S	P	U	H	L	F	N	K	J	G	T	I	Z	D	B	A	C	M	E	W	V	Y	Z	O
φ_{19}: KFO	W	V	E	K	C	N	X	Z	O	R	D	Y	P	F	I	M	S	J	Q	U	T	B	A	G	L	H
φ_{20}: KFP	T	M	P	X	Z	I	H	G	F	Q	U	S	B	R	Y	C	J	N	L	A	K	W	V	D	O	E
φ_{21}: KFQ	C	T	A	V	M	N	Y	Z	J	I	Q	O	E	F	L	X	K	W	U	B	S	D	R	P	G	H

Table 6.2: Example 2—Combined rotor substitutions for rotor order I, II, III without turnover of Rotor II. Calculated using the online Enigma simulation at `http://enigmaco.de/`.

Exploring Solution (2)

We try the second proposed solution. As before we begin by decrypting the ciphertext, starting from position 103, rotor order I, II, III. Because `V` is the turnover point of Rotor III we have to turn Rotor II back by one position in order to get the correct start positions `WGV`. The trial decryption gives

```
ZZINSYHUTSZRKJDVJJLJIJMQHCBRINYI
STOPXOBERLEUTNANTXZURXSEEXJAEGER
```

—a perfect result. We see that indeed `V` is the true turnover point of Rotor III, that means that the ring setting of this rotor is `A`. Moreover all letters except `F` and `W` occur, proving that they are unplugged, and the only possible plug connection could be between `F` and `W`.

From position 103 we go back for 26 positions and start with the rotor setting `WFV`. We get

```
RGYOZPKOEAGRGYSGQDKKNITDWF
ISTXLEUCHTTONNEXKNULLNEUNX
```

This proves that also `F` and `W` are unplugged. The only key element yet unknown is the ring setting of rotor II.

We go back for another 26 letters and start with the rotor positions `WEV`. This gives the trial decryption

```
FDNBGWTANCSZZWHPHNPDDSAXGT
SHKTDFEEFXMAMPPGAGRJIXKMXN
```

and the end rotor positions `XFV` instead of `WFV`. Something must have happened in between, and this could only be the stepping of Rotor I. The position of Rotor II then must have been `E`. Because of the double stepping of Rotor II the rotor start positions for this section of text must be `VDV`. Let's try this:

```
FDNBGWTANCSZZWHPHNPDDSAXGT
XHDREIZEHNXSTOPXERLOSCHENX
```

This is correct plaintext and proves that Rotor II has turnover point `E`, corresponding to ring setting `A`.

We conclude that the rotor start positions for the complete text are `VCW`, and get the decryption

```
ZIDPVUSABHHEABGRZMOPUWVJDMLPCSPFTSHISJMRRFSKUKHUATS
MELDUNGXVONXFREGATTEXGERMANIAXSTOPXPLANQUADRATXQELF

FDNBGWTANCSZZWHPHNPDDSAXGTRGYOZPKOEAGRGYSGQDKKNITDWF
XHDREIZEHNXSTOPXERLOSCHENXISTXLEUCHTTONNEXKNULLNEUNX

ZZINSYHUTSZRKJDVJJLJIJMQHCBRINYI
STOPXOBERLEUTNANTXZURXSEEXJAEGER
```

or, written in a more readable form,

> Meldung X von X Fregatte X Germania X Stop X Planquadrat X Qelf X Hdreizehn X Stop X Erloschen X ist X Leuchttonne X Knullneun X Stop X Oberleutnant X zur X See X Jaeger

A Note on the Technical Realization: Welchman's Diagonal Board

To systematically explore Welchman's plug conditions we consider the connected component of the Turing graph that we used. Assume it consists of the set $M = \{s_1, \ldots, s_r\}$ of letters. When the bombe stops it also provides the plug connection of the selected letter, say s_1 with $\tilde{s}_1$, and allows to derive the set of plug connections $\tilde{M} = \{\tilde{s}_1, \ldots, \tilde{s}_r\}$.

For the false "solution" (1) we had $M = \{\mathtt{E,J,L,R,S,T,U,X,Z}\}$, and the provided or derived plug connections

$$\tilde{\mathtt{E}} = \mathtt{L},\ \tilde{\mathtt{J}} = \mathtt{N},\ \tilde{\mathtt{L}} = \mathtt{D},\ \tilde{\mathtt{R}} = \mathtt{Y},\ \tilde{\mathtt{S}} = \mathtt{Z},\ \tilde{\mathtt{T}} = \mathtt{I},\ \tilde{\mathtt{U}} = \mathtt{F},\ \tilde{\mathtt{X}} = \mathtt{F},\ \tilde{\mathtt{Z}} = \mathtt{X}.$$

We observe two kinds of contradictions:

1. $\tilde{\mathtt{U}} = \mathtt{F}$, $\tilde{\mathtt{X}} = \mathtt{F}$: Two letters in M cannot be connected to the same letter in $\tilde{M}$.
2. $\tilde{\mathtt{E}} = \mathtt{L}$, $\tilde{\mathtt{L}} = \mathtt{D}$, hence $\eta\mathtt{E} = \tilde{\mathtt{E}} \in M \cap \tilde{M}$ and $\eta^2\mathtt{E} \neq \mathtt{E}$. In the same way $\tilde{\mathtt{S}} = \mathtt{Z}$, $\tilde{\mathtt{Z}} = \mathtt{X}$, $\eta^2\mathtt{S} \neq \mathtt{S}$, and $\tilde{\mathtt{Z}} = \mathtt{X}$, $\tilde{\mathtt{X}} = \mathtt{F}$, $\eta^2\mathtt{Z} \neq \mathtt{Z}$.

Checking for these contradictions in software is easy. Welchman's ingenious idea was to imagine and construct a simple device, the diagonal board, that was attached to the bombe and prevented stops in situations that contained contradictions to the plug conditions.

The improved bombe, called Turing-Welchman Bombe, provided only very few false positives. Moreover it not only used the letters in the cycles but also "non-cycle" letters connected to a cycle, in other words, a complete connected component of the Turing graph. In fact it even worked when the graph didn't have any cycles.

6.9 Example 3

Since Example 2 turned out to be quite simple, we analyze one more example. The ciphertext is

```
CZSTQ GJYNF ZYOLR TLXBR YXJCE MONAS XIPHU CXSAD BGEEQ ROBPI
QMUDP LWYDD GRCMC MJLGW TWBDK BHCPM UMEIB TMCUR DOVPU XNGBZ
QRBKD RPCKL XQKYM CSLGP NHIGD LOHBM PYPNV MTZVU EBDCZ AZLSX
OSZHL GSSZN MBBWS FDTUW IAXEH HLQGR LXMVA MXLWF QGOOA RZXUH
VUAWM KQDXH ZOIJI AMXCI TQNUM ZTZIW CKSBH HRZBH HRNZE WZCGV
BQ
```

and we are quite sure that the plaintext begins with “Befehl X des X Fuehrers X Stop X”. We align this with the ciphertext:

```
CZSTQ GJYNF ZYOLR TLXBR YXJCE
BEFEH LXDES XFUEH RERSX STOPX
```

Negative pattern search yields no contradiction. From positions 1 to 20 we derive the Turing graph whose largest connected component is shown in Figure 6.8. It has three cycles that overlap, two of them of length 2. Running the Bombe Simulator in “Turing mode” for these three cycles yields about $1500 \approx 60 \cdot 26$ solutions, as expected. The (lexicographically) first of them is

Rotor order	I II III
Start position	`ZPB`

Table 6.3 describes the transformations $\varphi_2, \ldots, \varphi_{20}$.

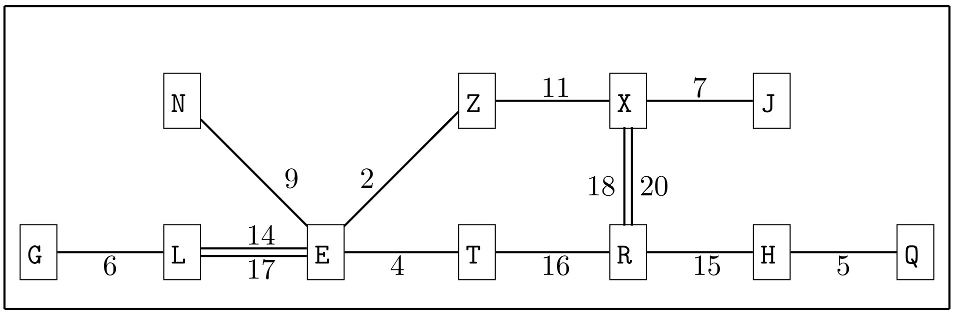


Figure 6.8: Turing *graph for Example 3, largest connected component*

Now we consider the `E-L-E` cycle and the `E-Z-X-R-T-E` cycle, see Table 6.4. The `L-E` cycle has 6 compatible plug connections for `E` and `L`. The `E-Z-X-R-T-E` cycle boils this number down to 1. The third cycle, `X-R-X`, fits into the picture, because $\varphi_{20}\tilde{\texttt{X}} = \varphi_{20}\texttt{I} = \texttt{B} = \tilde{\texttt{R}}$.

Again the Welchman conditions rule out this solution because of the contradiction in the first row: $\tilde{\texttt{L}} = \texttt{B}$ in column 2, $\tilde{\texttt{R}} = \texttt{B}$ in column 6. And indeed, running the Bombe Simulator in “Welchman mode” yields a unique solution:

Rotor order	III II I
Start position	`BMX`

with the plugs `A-Z`, `C-X`, `E-V`. A trial decryption with these plugs and ring settings `AAA` shows parts, but not all of the known plaintext:

```
   EUEHLXHECXGFEHRERLXZTOPX
    *    * * **      * *
(B)EFEHLXDESXFUEHRERSXSTOPX
```

Substition in	Substitution table																									
rotor position	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
φ_2: ZPB	N	G	E	S	C	I	B	R	F	W	X	U	O	A	M	Y	Z	H	D	V	L	T	J	K	P	Q
φ_3: ZPC	M	J	S	H	Q	O	K	D	W	B	G	V	A	U	F	Z	E	Y	C	X	N	L	I	T	R	P
φ_4: ZPD	F	L	H	N	I	A	T	C	E	R	X	B	Y	D	Z	Q	P	J	V	G	W	S	U	K	M	O
φ_5: ZPE	V	D	G	B	J	T	C	K	U	E	H	Y	W	Z	S	R	X	P	O	F	I	A	M	Q	L	N
φ_6: ZPF	P	T	I	U	J	Z	Q	M	C	E	Y	S	H	W	X	A	G	V	L	B	D	R	N	O	K	F
φ_7: ZPG	R	D	I	B	M	Q	U	V	C	Y	O	T	E	X	K	Z	F	A	W	L	G	H	S	N	J	P
φ_8: ZPH	Q	L	F	T	K	C	P	R	Z	S	E	B	X	W	U	G	A	H	J	D	O	Y	N	M	V	I
φ_9: ZPI	D	X	J	A	L	Q	I	S	G	C	U	E	W	R	Z	V	F	N	H	Y	K	P	M	B	T	O
φ_{10}: ZPJ	S	W	X	L	R	U	Q	T	O	M	Y	D	J	Z	I	V	G	E	A	H	F	P	B	C	K	N
φ_{11}: ZPK	P	E	O	H	B	Z	Q	D	N	R	W	Y	U	I	C	A	G	J	X	V	M	T	K	S	L	F
φ_{12}: ZPL	R	M	S	Y	L	U	T	Q	P	X	Z	E	B	V	W	I	H	A	C	G	F	N	O	J	D	K
φ_{13}: ZPM	J	P	S	G	Y	N	D	Z	Q	A	T	U	V	F	X	B	I	W	C	K	L	M	R	O	E	H
φ_{14}: ZPN	B	A	Z	W	Y	R	I	O	G	T	U	X	Q	V	H	S	M	F	P	J	K	N	D	L	E	C
φ_{15}: ZPO	H	M	S	Y	O	R	L	A	T	U	P	G	B	X	E	K	W	F	C	I	J	Z	Q	N	D	V
φ_{16}: ZPP	K	F	D	C	R	B	S	T	U	N	A	P	V	J	Z	L	X	E	G	H	I	M	Y	Q	W	O
φ_{17}: ZPQ	B	A	V	L	Y	S	U	O	K	M	I	D	J	P	H	N	Z	X	F	W	G	C	T	R	E	Q
φ_{18}: ZPR	N	I	J	Q	T	U	M	W	B	C	V	S	G	A	Y	X	D	Z	L	E	F	K	H	P	O	R
φ_{19}: ZPS	Q	P	K	R	U	J	Z	N	L	F	C	I	W	H	T	B	A	D	Y	O	E	X	M	V	S	G
φ_{20}: ZPT	V	I	G	L	Z	P	C	M	B	N	S	D	H	J	Y	F	X	U	K	W	R	A	T	Q	O	E

Table 6.3: Example 3—Combined rotor substitutions for rotor order I, II, III without turnover of Rotor II. Calculated using the online Enigma simulation at `http://enigmaco.de/`.

$\tilde{\mathtt{E}}$	$\xrightarrow{14}$	$\tilde{\mathtt{L}}$	$\xrightarrow{17}$	$\tilde{\mathtt{E}}$	$\xrightarrow{2}$	$\tilde{\mathtt{Z}}$	$\xrightarrow{11}$	$\tilde{\mathtt{X}}$	$\xrightarrow{18}$	$\tilde{\mathtt{R}}$	$\xrightarrow{16}$	$\tilde{\mathtt{T}}$	$\xrightarrow{4}$	$\tilde{\mathtt{E}}$
A		B		A		N		I		B		F		A
B		A		B		G		Q		D		C		H
C		Z		Q	†									
D		W		T	†									
E		Y		E		C		O		Y		W		U
F		R		X	†									
G		I		K	†									
H		O		H		R		J		C		D		N
I		G		U	†									
J		T		W	†									
K		U		G	†									
L		X		R	†									
M		Q		Z	†									
N		V		C	†									
O		H		O		M		U		F		B		L
P		S		F	†									
Q		M		J	†									
R		F		S	†									
S		P		N	†									
T		J		M	†									
U		K		I	†									
V		N		P	†									
W		D		L	†									
X		L		D	†									
Y		E		Y		P		A		N		J		R
Z		C		V	†									

Table 6.4: Example 3—Possible plug connections for the first two loops

To get on we use a second connected component of the Turing graph, see Figure 6.9.

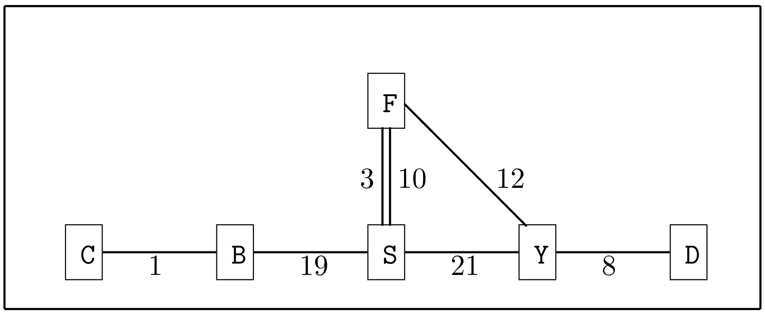


Figure 6.9: Turing *graph for Example 3, second connected component*

Trying the cycle `S-F-S` with φ_3 and φ_{10} using all the plugs for `S` that are yet free gives two possible solutions: `S-U-S` and `U-S-U`. The second one violates the Welchman condition for `S`. The first one yields the plugs `S-S` and `F-U`. Furthermore we get $\tilde{\texttt{Y}} = \varphi_{12}\tilde{\texttt{F}} = \varphi_{12}\texttt{U} = \texttt{B}$, and $\tilde{\texttt{D}} = \varphi_8\tilde{\texttt{Y}} = \varphi_8\texttt{B} = \texttt{W}$.

Up to now we identified the plugs `A-Z`, `B-Y`, `C-X`, `D-W`, `E-V`, `F-U`. Trial decryption yields the perfect plaintext

```
EFEHLXDESXFUEHRERSXSTOPX
```

So we try to decrypt the complete ciphertext with the rotor order III II I, the ring settings `AAA`, the plugs `A-Z`, `B-Y`, `C-X`, `D-W`, `E-V`, `F-U`, and the start positions `BMW`, and get

```
BEFEH LXDES XFUEH RERSX STOPX IMXFA LLEXZ XZTXU NWAHR SQEIN
LIQEN XFRAN ZOESI SQENX ANGRI FFSXS INDXD IEXWE STBEF ESTIG
UNGEN XJEDE RXZAH LENMA ESSIG ENXUE BERLE GENHE ITXZU MXTRO
TZXZU XHALT ENXST OPXFU EHRUN GXUND XTRUP PEXMU ESSEN XVONX
DIESE RXEHR ENPFL IQTXD URQDR UNGEN XSEIN XSTOP XHEIL XHITL
ER
```

> Befehl des Fuehrers STOP Im Falle z. Zt. unwahrscheinlichen franzoesischen Angriffs sind die Westbefestigungen jeder zahlenmaessigen Ueberlegenheit zum Trotz zu halten STOP Fuehrung und Truppe muessen von dieser Ehrenpflicht durchdrungen sein STOP Heil Hitler

We observe that the slow rotor didn't step during this decryption. In general the a priori probability for its stepping was 257 letters of text divided by 676 possible positions of the other two rotors ≈ 0.38.

6.10 Discussion

- TURING's attack against the cycles of the graph also works for non-involutory rotor machines. Then the graph model is a *directed* graph and the attacker has to find directed cycles. These are quite rare, therefore the attack loses most of its power.
- **Exercise.** Find the directed cycles in Figures 6.5, 6.7, 6.8, 6.9.
- The TURING-WELCHMAN Bombe used the involutary characters of the complete Enigma substitution as well as of the plugboard. The inventors of both of these "features" apparently didn't see the weaknesses.
- Nevertheless the addition of the plugboard made the machine much stronger. The isomorph attack worked by paper and pencil. Attacking the Wehrmacht Enigma only worked with the help of heavy machinery.

Chapter 7

Aperiodic Polyalphabetic Ciphers

Overview Over Polyalphabetic Ciphers

	Monoalph. Substitution	**Periodic Polyalph. Substitution**	**Aperiodic Polyalph. Substitution**
Standard Alphabet	Shift Cipher (CAESAR)	BELLASO cipher ("Vigenère")	Running-Text Cipher
Non-Standard Alphabet	General Monoalph. Substitution	PORTA's General Polyalph. Cipher	Stream Cipher

The table is not completely exact. The running-text cipher is only a (but the most important) special case of an aperiodic polyalphabetic substitution using the standard alphabet. An analogous statement holds for PORTA's disk cipher and a general periodic polyalphabetic substitution. In contrast by stream cipher we denote an even more general construct.

7.1 Running-Text Ciphers

Method

Assume we have a plaintext of length r. We could encrypt it with the BELLASO cipher (and the TRITHEMIUS table). But instead of choosing a keyword and periodically repeating this keyword we use a keytext of the same length r as the plaintext. Then we add plaintext and keytext letter for letter (using the table).

The abstract mathematical description uses a group structure on the alphabet Σ with group operation $*$. For a plaintext $a \in M_r = M \cap \Sigma^r$ we choose a key $k \in \Sigma^r$ and calculate

$$c_i = a_i * k_i \quad \text{for } 0 \le i \le r-1.$$

We may interpret this as shift cipher on Σ^r. The formula for decryption is

$$a_i = c_i * k_i^{-1} \quad \text{for } 0 \le i \le r-1.$$

If the key itself is a meaningful text $k \in M_r$ in the plaintext language, say a section from a book, then we call this a **running-text cipher**.

Example

Equip $\Sigma = \{\mathtt{A}, \ldots, \mathtt{Z}\}$ with the group structure as additive group of integers mod 26.

```
Plaintext:  i a r r i v e t o m o r r o w a t t e n o c l o c k
Keytext:    I F Y O U C A N K E E P Y O U R H E A D W H E N A L
            ---------------------------------------------------
Ciphertext: Q F P F C X E G Y Q S G P C Q R A X E Q K J P B C V
```

A Perl program is `runkey.pl` in the web directory `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Perl/`.

Practical Background

To avoid a period in a polyalphabetic substitution we choose a key that is (at least) as long as the plaintext. On the other hand we need a key that is easily remembered or transferred to a communication partner.

A common method of defining such a key is taking a book and beginning at a certain position. The effective key is the number triple (page, line, letter). This kind of encryption is sometimes called a **book cipher**. Historically the first known reference for this method seems to be

> Arthur Hermann: *Nouveau système de correspondence secrète. Méthode pour chiffrer et déchiffrer les dépêches secrètes.* Paris 1892.

But note that there are also other ways to use a book for encryption, see `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/1_Monoalph/Variants.html`.

A modern version could use the contents of a CD beginning with a certain position.

Exercise: How large is the keyspace of this cipher, when the attacker knows which CD was used?

7.2 Cryptanalytic Approaches to Running-Text Ciphers

Cryptanalysis of running-text ciphers is laborious. There are several approaches that should be combined in practice. Automated procedures are proposed in

> E. Dawson and L. Nielsen: Automated cryptanalysis of XOR plaintext strings. Cryptologia XX (1996), 165–181.
>
> A. Griffing: Solving the running key cipher with the Viterbi algorithm. Cryptologia XXX (2006), 361–367.

The first of these considers running-text ciphers where plaintext and key are combined via binary addition (XOR) instead of addition mod 26. This distinction not essential for the method (but of course for the use of the program).

Approach 0: Exhaustion

Exhaustion of all possible keytexts is practically infeasible when there is no a priori idea what the keytext could be. Exhaustion is feasible when the attacker knows the source of the keytext, say a certain book. If the source text has length q and the ciphertext has length r, then there are only $q - r$ choices for the start of the key text. This is troublesome for the pencil and paper analyst, but easy with machine support.

Approach 1: Probable Word and Zigzag Exhaustion

When in the example above the attacker guesses the probable word "arrive" in the plaintext and shifts it along the ciphertext, already for the second position she gets the keytext `FYOUCA`. With a little imagination she guesses the phrase `IFYOUCAN`, yielding the plaintext fragment `IARRIVET`, and expands this fragment to `IARRIVETOMORROW`. This in turn expands the keytext to `IFYOUCANKEEPYOU`. Proceeding in this way alternating between plaintext and keytext is called **zigzag exhaustion** (or cross-ruff method). For some time during this process it may be unclear whether a partial text belongs to plaintext or key.

A dictionary is a useful tool for this task. Or a pattern search in a collection of literary texts may lead to success.

Approach 2: Frequent Word Fragments

If the attacker cannot guess a probable word she might try common word fragments, bearing in mind that plaintext as well as keytext are meaningful texts. Shifting words or word fragments such as

```
THE AND FOR WAS HIS NOT BUT ARE ING ION ENT
THAT THIS FROM WITH HAVE TION
```

along the ciphertext will result in many meaningful trigrams or tetragrams that provide seed crystals for a zigzag exhaustion. Recognizing typical combinations such as

```
THE + THE = MOI
ING + ING = QAM
THAT + THAT = MOAM
```

may be useful.

Approach 3: Frequency Analysis

Let $p_0, \ldots, p_{n-1}$ be the letter frequencies of the (stochastic) language M over the alphabet $\Sigma = \{s_0, \ldots, s_{n-1}\}$. Then running-key ciphertexts will exhibit the typical letter frequencies

$$q_h = \sum_{i+j=h} p_i \cdot p_j \quad \text{for } 0 \leq h \leq n-1.$$

Even though the distribution is much more flat compared with plain language, it is not completely uniform, and therefore leaks some information on the plaintext. For example it gives a hint at the method of encryption.

Example: Letter frequencies of running-text cryptograms in **English** (values in percent). Coincidence index = 0.0400.

A	B	C	D	E	F	G	H	I	J	K	L	M
4.3	3.5	3.2	2.5	4.7	3.8	4.4	4.4	4.8	2.9	3.5	4.5	4.3
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
3.1	3.2	3.6	3.0	4.4	4.5	4.0	3.2	4.9	4.7	3.8	3.3	3.5

Example: Letter frequencies of running-text cryptograms in **German** (values in percent). Coincidence index = 0.0411.

A	B	C	D	E	F	G	H	I	J	K	L	M
4.2	2.6	2.3	2.4	5.0	3.7	3.7	4.3	5.8	2.9	3.7	4.4	4.9
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
3.2	3.0	3.1	3.3	5.7	3.4	3.2	3.4	5.9	4.5	3.9	3.9	3.6

Even more helpful is the distribution of bigrams and trigrams. Each bigram in the ciphertext has $26^2 = 676$ different possible sources whose probabilities however show large differences. For trigrams most sources even have probabilities 0.

A systematic description of this approach is in

Craig Bauer and Christian N. S. Tate: A statistical attack on the running key cipher. Cryptologia XXVI (2002), 274–282.

Approach 4: Frequent Letter Combinations

Frequency analysis (approach 3) is cumbersome, at least for manual evaluation. FRIEDMAN refined this approach in a systematic way that doesn't need known plaintext. See the next section.

7.3 Cryptanalysis According to FRIEDMAN

FRIEDMAN proposed a systematic approach to solving running-key ciphers in the article

> W. F. Friedman: *Methods for the Solution of Running-Key Ciphers.* Riverbank Publication No. 16 (1918). In: The Riverbank Publications Vol 1, Aegean Park Press 1979.

Consider a running-text cryptogram. FRIEDMAN's method starts from the observation that a significant fraction of the ciphertext letters arise from a combination of two frequent plaintext letters.

The frequency distribution (in percent) of the nine most frequent German letters is:

E	N	I	R	S	A	T	D	U
18.0	10.6	8.1	7.2	6.9	6.0	5.7	5.4	4.6

Therefore these letters account for 72.5% of a German text.

Assuming that the key is sufficiently independent of the plaintext we expect that about 53% ciphertext letters arose from a combination of two of these letters in plaintext or key. This fact is not overly impressive. In the example

```
                                  |   | | |     |   |
  Plaintext:  i c h k o m m e m o r g e n u m z e h n
  Key:        V O M E I S E B E F R E I T S I N D S T
              ---------------------------------------
  Ciphertext: D Q T O W E Q F Q T I K M G M U M H Z G
```

this applies only to 6 of 20 letters. The method won't work well for this example.

Let us take another example (from the football world championships 2002):

```
            | | | | |       | |         | |   |       |   | |
Plaintext:  d e u t s c h l a n d b e s i e g t p a r a g u a y
Key:        E I N E N A T U E R L I C H E S P R A C H E H A T T
            ---------------------------------------------------
Ciphertext: H M H X F C A F E E O J G Z M W V L P C Y E N U T R
```

Here we see 13 of 26 letters as interesting. We use this example to explain the method.

Let's begin with the first four letters, and consider all combinations that lead to them

```
Plaintext:  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Key:        H G F E D C B A Z Y X W V U T S R Q P O N M L K J I
                  | |               |             |
Ciphertext: H H H H H H H H H H H H H H H H H H H H H H H H H H

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

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

Plaintext:  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Key:        X W V U T S R Q P O N M L K J I H G F E D C B A Z Y
                  | |                             | |
Ciphertext: X X X X X X X X X X X X X X X X X X X X X X X X X X
```

The most probable pairs are flagged. We condense this observation:

```
DENU EISTU DENU DETU
EDUN IEUTS EDUN UTED
H    M     H    X
```

There is a total of $4 \cdot 5 \cdot 4 \cdot 4 = 320$ possible combinations of these pairs. Some of them may be eliminated immediately, for example we may exclude that plaintext or key begin with the letters `DS`.

If we start with the pair `D-E` we might continue with `E-I` or `U-S`. The first case has only one meaningful continuation:

```
DEUT
EINE
```

The second case could proceed with `D-E`, but no fourth pair fits. A possible pair number 3 is `N-U` also, and then `E-T` or `T-E` fit as pair number 4. Therefore we note two more options, both of them not really convincing:

```
DEUT  DUNE  DUNT
EINE  ESUT  ESUE
```

Starting with `E-D` we find an exactly symmetric situation and get the same three options but with plaintext and key interchanged.

Starting with `N-U` we might continue with `I-E` or `U-S`. The first case has `E-D` as only plausible continuation, and then `T-E`:

```
DEUT  DUNE  DUNT  NIET
EINE  ESUT  ESUE  UEDE
```

The second case could proceed with `D-E` (and then `E-T`) or `N-U` (and then there is no good continuation). So we found one more option:

```
DEUT  DUNE  DUNT  NIET  NUDE
EINE  ESUT  ESUE  UEDE  USET
```

Taking all the symmetric ones into account we face a total of 10 somewhat plausible options—*under the assumption that the first four letters of plaintext and key belong to the nine most frequent German letters.*

Of our five (+ five symmetric) options the first looks best. But also the fourth is reasonably good, bearing in mind that the keytext might begin in the middle of a word (for example “müde” = `(M)UEDE`). In any case let’s begin with the first option that looks somewhat better. It suggests the continuation `SCH`. This seems promising:

```
DEUTSCH
EINENAT
```

Of course if this fails we would also try for example `DEUTLICH` or `DEUTEN`.

As next letter in the first row we would try `E` or `L` and note that `L` gives a better continuation in the second row (`U` better than `B`). Therefore the begin `DEUTSCHLAND` is decrypted—but we don’t yet know whether it is plaintext or key. From this point we struggle ahead in zigzag as noted before.

7.4 Other Applications of Running-Text Analysis

Key Re-Use

Consider an alphabet Σ with a group structure, and consider an (aperiodic or periodic) polyalphabetic cipher that uses the CAESAR operation: For a plaintext $a = (a_0, a_1, a_2, \ldots)$ and a keystream $k = (k_0, k_1, k_2, \ldots)$ the ciphertext $c = (c_0, c_1, c_2, \ldots)$ is given by

$$c_i = a_i * k_i \quad \text{for } i = 0, 1, 2, \ldots.$$

Because the key is not necessarily meaningful text the cryptanalytic methods for running-text ciphers don’t apply.

But suppose another plaintext $b = (b_0, b_1, b_2, \ldots)$ is encrypted with the *same* key k, resulting in the ciphertext $d = (d_0, d_1, d_2, \ldots)$,

$$d_i = b_i * k_i \quad \text{for } i = 0, 1, 2, \ldots.$$

The attacker recognizes this situation by coincidence analysis.

Then the difference (or quotient, depending on the notation of the group law) is given by

$$d_i * c_i^{-1} = b_i * k_i * k_i^{-1} * a_i^{-1} = b_i * a_i^{-1} \quad \text{for } i = 0, 1, 2, \ldots.$$

In this way the attacker who knows the ciphertexts c and d finds the difference $b_i * a_i^{-1}$ that is the composition of two meaningful texts she doesn't know but wants to. She therefore applies the methods for running-text encryption and eventually finds a and b and then even k.

Historical Notes

This kind of analysis was a main occupation of the cryptanalysts in World War II and in the following Cold War. In particular teleprinter communication used additive stream ciphers (mostly XOR) with keystreams from key generators and very long periods. In case of heavy message traffic often passages of different messages were encrypted with the key generator in the same state. Searching such passages was called "in-depth-analysis" and relied on coincidence calculations. Then the second step was to subtract the identified passages and to apply running-text analysis.

Some known examples for this are:

- Breaking the Lorenz cipher teleprinter SZ42 ("Schlüsselzusatz") by the British cryptanalysts at Bletchley Park in World War II (project "Tunny").
- Breaking HAGELIN's B21 in 1931 and the Siemens-Geheimschreiber T52 in 1940 by the Swedish mathematician Arne BEURLING. The T52 was also partially broken at Bletchley Park (project "Sturgeon").
- The latest politically relevant application of this cryptanalytic technique occurred in the 1950es. US cryptanalysts broke Sovjet ciphertexts and by the way debunked the spy couple Ethel und Julius ROSENBERG (project "Venona"). The Sovjet spys used a one-time pad—in principle. But because key material was rare keys were partly reused.

Large Periods

Another application is the TRITHEMIUS-BELLASO cipher with a large period l, large enough that the standard procedure of arranging the ciphertext in columns and shifting the alphabets fails.

Then the attacker may consider the ciphertext shifted by l positions and subtract it from the original ciphertext:

$$c_{i+l} - c_i = a_{i+l} - a_i.$$

Or, if the key consists of meaningful text, directly treat the cipher as a running-text cipher.

Exercise.

```
BOEKV HWXRW VMSIB UXBRK HYQLR OYFWR KODHR JQUMM SJIQA THWSK
CRUBJ IELLM QSGEQ GSJFT USEWT VTBPI JMPNH IGUSQ HDXBR ANVIS
VEHJL VJGDS LVFAM YIPJY JM
```

Hints.

- Find evidence for a period of 38 or 76.
- Try the probable word `AMERICA` as part of the key.

7.5 Random Keys

All cryptanalytic methods collapse when the key is a random letter sequence, chosen in an independent way for each plaintext, and never repeated. In particular all the letters in the ciphertexts occur with the same probability. Or in other words, the distribution of the ciphertext letters is completely flat.

This encryption method is called **One-Time Pad** (OTP). Usually Gilbert VERNAM (1890–1960) is considered as the inventor in the World War II year 1917. But the idea of a *random* key is due to MAUBORGNE who improved VERNAM's periodic XOR cipher in this way. The German cryptologists KUNZE, SCHAUFFLER, and LANGLOTZ in 1921—presumably independently from MAUBORGNE—proposed the "individuellen Schlüssel" ("individual key") for running-text encryption of texts over the alphabet $\{\mathtt{A}, \ldots, \mathtt{Z}\}$.

In other words: The idea "was in the air". In 2011 Steve Bellovin discovered a much earlier proposal of the method by one Frank MILLER in 1882 who however was completely unknown as a crypologist and didn't have any influence on the history of cryptography.

Steven M. Bellovin. *Frank Miller: Inventor of the One-Time Pad.* Cryptologia 35 (2011), 203–222.

Uniformly Distributed Random Variables in Groups

This subsection contains evidence for the security of using random keys. The general idea is:

> "Something + Random = Random" or "Chaos Beats Order"
> (the Cildren's Room Theorem)

We use the language of Measure Theory.

Theorem 12 *Let G be a group with a finite, translation invariant measure μ and Ω, a probability space. Let $X, Y : \Omega \longrightarrow G$ be random variables, X uniformly distributed, and X, Y independent. Let $Z = X * Y$ (where $*$ is the group law of composition). Then:*

(i) *Z is uniformly distributed.*

(ii) *Y and Z are independent.*

Comment The independency of X and Y means that

$$P(X^{-1}A \cap Y^{-1}B) = P(X^{-1}A) \cdot P(Y^{-1}B) \quad \text{for all measurable } A, B \subseteq G.$$

The uniform distribution of X means that

$$P(X^{-1}A) = \frac{\mu(A)}{\mu(G)} \quad \text{for all measurable } A \subseteq G.$$

In particular the measure P_X on G defined by $P_X(A) = P(X^{-1}A)$ is translation invariant, if μ is so.

Remark Z is a random variable because $Z = m^{-1} \circ (X, Y)$ with $m = *$, the group law of composition. This is measurable because its g-sections,

$$(m^{-1}A)_g = \{h \in G \mid gh \in A\}$$

are all measurable, and the function

$$g \mapsto \mu(m^{-1}A)_g = \mu(g^{-1}A) = \mu(A)$$

is also measurable. A weak form of Fubini's theorem gives that $m^{-1}A \subseteq G \times G$ is measurable, and

$$(\mu \otimes \mu)(m^{-1}A) = \int_G (m^{-1}A)_g \, dg = \mu(A) \int_G dg = \mu(A)\mu(G).$$

Counterexamples We analyze whether the conditions of the theorem can be weakened.

1. What if we don't assume X is uniformly distributed? As an example take $X = \mathbf{1}$ (unity element of group) constant and Y arbitrary; then X and Y are independent, but $Z = Y$ in general is not uniformly distributed nor independent from Y.
2. What if we don't assume X and Y are independent? As an example take $Y = X^{-1}$ (the group inverse); the product $Z = \mathbf{1}$ in general is not uniformly distributed. Choosing $Y = X$ we get $Z = X^2$ that in general is not uniformly distributed nor independent from Y. (More concrete example: $\Omega = G = \mathbb{Z}/4\mathbb{Z}, X =$ identity map, $Z =$ squaring map.)

General proof of the Theorem

(For an elementary proof of a practically relevant special case see below.)

Consider the product map

$$(X, Y)\colon \Omega \longrightarrow G \times G$$

and the extended composition

$$\sigma\colon G \times G \longrightarrow G \times G, \quad (g, h) \mapsto (g * h, h).$$

For $A, B \subseteq G$ we have (by definition of the product probability)

$$(P_X \otimes P_Y)(A \times B) = P_X(A) \cdot P_Y(B) = P(X^{-1}A) \cdot P(Y^{-1}B);$$

because X and Y are independent we may continue this equation:

$$\begin{aligned} &= P(X^{-1}A \cap Y^{-1}B) = P\{\omega \mid X\omega \in A, Y\omega \in B\} \\ &= P((X,Y)^{-1}(A \times B)) = P_{(X,Y)}(A \times B). \end{aligned}$$

Therefore $P_{(X,Y)} = P_X \otimes P_Y$, and for $S \subseteq G \times G$ we apply FUBINI's theorem:

$$P_{(X,Y)}(S) = \int_{h \in G} P_X(S_h) \cdot P_Y(dh).$$

Especially for $S = \sigma^{-1}(A \times B)$ we get

$$\begin{aligned} S_h &= \{g \in G \mid (g * h, h) \in A \times B\} = \begin{cases} A * h^{-1}, & \text{if } h \in B, \\ \emptyset & \text{else,} \end{cases} \\ P_X(S_h) &= \begin{cases} P_X(A * h^{-1}) = \frac{\mu(A)}{\mu(G)}, & \text{if } h \in B, \\ 0 & \text{else.} \end{cases} \end{aligned}$$

Therefore

$$\begin{aligned} P(Z^{-1}A \cap Y^{-1}B) &= P\{\omega \in \Omega \mid X(\omega) * Y(\omega) \in A, Y(\omega) \in B\} \\ &= P((X,Y)^{-1}S) = P_{(X,Y)}(S) \\ &= \int_{h \in B} \frac{\mu(A)}{\mu(G)} \cdot P_Y(dh) = \frac{\mu(A)}{\mu(G)} \cdot P(Y^{-1}B). \end{aligned}$$

Setting $B = G$ we conclude $P(Z^{-1}A) = \frac{\mu(A)}{\mu(G)}$, which gives (i), and from this we immediately conclude

$$P(Z^{-1}A \cap Y^{-1}B) = P(Z^{-1}A) \cdot P(Y^{-1}B)$$

which proves also (ii). $\diamond$

Proof for countable groups

In the above proof we used general measure theory, but the idea was fairly simple. Therefore we repeat the proof for the countable case, where integrals become sums and the argumentation is elementary. For the cryptographic application the measure spaces are even finite, so this elementary proof is completely adequate.

Lemma 8 *Let G, Ω, X, Y, and Z be as in the theorem. Then*

$$Z^{-1}(A) \cap Y^{-1}(B) = \bigcup_{h \in B} [X^{-1}(A * h^{-1}) \cap Y^{-1}h]$$

for all measurable $A, B \subseteq G$.

The proof follows from the equations

$$\begin{aligned}
Z^{-1}A &= (X,Y)^{-1}\{(g,h) \in G \times G \mid g * h \in A\} \\
&= (X,Y)^{-1}\left[\bigcup_{h \in G} A * h^{-1} \times \{h\}\right] \\
&= \bigcup_{h \in G} (X,Y)^{-1}(A * h^{-1} \times \{h\}) \\
&= \bigcup_{h \in G} [X^{-1}(A * h^{-1}) \cap Y^{-1}h], \\
Z^{-1}A \cap Y^{-1}B &= \bigcup_{h \in G} [X^{-1}(A * h^{-1}) \cap Y^{-1}h \cap Y^{-1}B] \\
&= \bigcup_{h \in B} [X^{-1}(A * h^{-1}) \cap Y^{-1}h].
\end{aligned}$$

Now let G be countable. Then

$$\begin{aligned}
P(Z^{-1}A \cap Y^{-1}B) &= \sum_{h\in B} P[X^{-1}(A*h^{-1}) \cap Y^{-1}h] \\
&= \sum_{h\in B} P[X^{-1}(A*h^{-1})] \cdot P[Y^{-1}h] \quad \text{(because } X, Y \text{ are independent)} \\
&= \sum_{h\in B} \frac{\mu(A*h^{-1})}{\mu(G)} \cdot P[Y^{-1}h] \quad \text{(because } X \text{ is uniformly distributed)} \\
&= \frac{\mu(A)}{\mu(G)} \cdot \sum_{h\in B} P[Y^{-1}h] \\
&= \frac{\mu(A)}{\mu(G)} \cdot P\left[\bigcup_{h\in B} Y^{-1}h\right] \\
&= \frac{\mu(A)}{\mu(G)} \cdot P(Y^{-1}B).
\end{aligned}$$

Setting $B = G$ we get $P(Z^{-1}A) = \frac{\mu(A)}{\mu(G)}$, which gives (i), and immediately conclude

$$P(Z^{-1}A \cap Y^{-1}B) = P(Z^{-1}A) \cdot P(Y^{-1}B),$$

which proves (ii). $\diamond$

Discussion

The theorem says that a One-Time Pad encryption results in a ciphertext that "has nothing to do" with the plaintext, in particular doesn't offer any lever for the cryptanalyst.

Why then isn't the One-Time Pad the universally accepted standard method of encryption?

- Agreeing upon a key is a major problem—if we can securely transmit a key of this length, why not immediately transmit the message over the same secure message channel? Or if the key is agreed upon some time in advance—how to remember it?
- The method is suited at best for a two-party communication. For a multiparty communication the complexity of key distribution becomes prohibitive.
- When the attacker has known plaintext she is not able to draw any conclusions about other parts of the text. But she can exchange the known plaintext with another text she likes more: *The integrity of the message is at risk.*

7.6 Autokey Ciphers

The first one to propose autokey ciphers was Bellaso in 1564. Also this cipher is often attributed to Vigenère.

Encryption and Decryption

The alphabet Σ is equipped with a group operation $*$. As key chose a string $k \in \Sigma^l$ of length l. For encrypting a plaintext $a \in \Sigma^r$ one concatenates k and a and truncates this string to r letters. This truncated string then serves as keytext for a running-key encryption:

Plaintext:	a_0	a_1	$\ldots$	a_{l-1}	a_l	$\ldots$	a_{r-1}
Keytext:	k_0	k_1	$\ldots$	k_{l-1}	a_0	$\ldots$	a_{r-l-1}
Ciphertext:	c_0	c_1	$\ldots$	c_{l-1}	c_l	$\ldots$	c_{r-1}

The formula for encryption is

$$c_i = \begin{cases} a_i * k_i & \text{for } i = 0, \ldots l-1, \\ a_i * a_{i-l} & \text{for } i = l, \ldots r-1. \end{cases}$$

Example, $\Sigma = \{\mathtt{A}, \ldots, \mathtt{Z}\}$, $l = 2$, $k = \mathtt{XY}$:

```
P L A I N T E X T
X Y P L A I N T E
-----------------
M J P T N B R Q X
```

Remark: Instead of the standard alphabet (or the Trithemius table) one could also use a permuted primary alphabet.

Here is the formula for decryption

$$a_i = \begin{cases} c_i * k_i^{-1} & \text{for } i = 0, \ldots l-1, \\ c_i * a_{i-l}^{-1} & \text{for } i = l, \ldots r-1. \end{cases}$$

A Perl program is `autokey.pl` in the web directory `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Perl/`.

Approaches to Cryptanalysis

The four most promising approaches are:

- Exhaustion for small l.
- Interpretation as running-key cipher from position l, in case of a key word or phrase from the plaintext language even from the beginning of the ciphertext:

  - Probable word and zigzag exhaustion
  - Frequent word fragments
  - Frequency analysis
  - Frequent letter combinations

  The repetition of the plaintext in the key makes the task considerably easier.

- Similarity with the TRITHEMIUS-BELLASO cipher, see Section 7.8 below
- Algebraic cryptanalysis (for known plaintext): Solving equations. We describe this for a commutative group, the group operation written as addition, that is, we consider Σ, Σ^r, and Σ^{r+l} as $\mathbb{Z}$-modules.

We interpret the encryption formula as a system of linear equations with an $r \times (r+l)$ coefficient matrix:

$$\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{l-1} \\ c_l \\ \vdots \\ c_{r-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \ldots & 1 & & & & \\ & 1 & 0 & \ldots & 1 & & & \\ & & \ddots & \ddots & & & \ddots & \\ & & & 1 & 0 & \ldots & & 1 \end{pmatrix} \begin{pmatrix} k_0 \\ k_1 \\ \vdots \\ k_{l-1} \\ a_0 \\ \vdots \\ a_{r-1} \end{pmatrix}$$

This is a system of r linear equations with the $r+l$ unknowns (the components of) $k \in \Sigma^l$ and $a \in \Sigma^r$. "In general" such a system is solvable as soon as l of the unknowns are guessed, that means known plaintext of length l (not necessarily connected). Since the involved $\mathbb{Z}$-modules are (in most interesting cases) not vector spaces, solving linear equations is a bit intricate but feasible. This is comprehensively treated in the next chapter.

Ciphertext Autokey

Using ciphertext instead of plaintext as extension of the l-letter key is a useless variant, but also proposed by VIGENÈRE. We only describe it by an example:

Example, $\Sigma = \{\mathtt{A}, \ldots, \mathtt{Z}\}$, $l = 2$, $k = \mathtt{XY}$:

```
P L A I N T E X T
X Y M J M R Z K D
-----------------
M J M R Z K D H W
```

Exercise. Give a formal description of this cipher. Why is cryptanalysis almost trivial? Work out an algorithm for cryptanalysis.

Exercise. Apply your algorithm to the cryptogram

```
IHTYE VNQEW KOGIV MZVPM WRIXD OSDIX FKJRM HZBVR TLKMS FEUKE
VSIVK GZNUX KMWEP OQEDV RARBX NUJJX BTMQB ZT
```

Remark: Using a nonstandard alphabet makes this cipher a bit stronger.

7.7 Example: Cryptanalysis of an Autokey Cipher

The Cryptogram

Suppose we got the ciphertext

```
LUSIT FSATM TZJIZ SYDZM PMFIZ REWLR ZEKLS RQXCA TFENE YBVOI
WAHIE LLXFK VXOKZ OVQIP TAUNX ARZCX IZYHQ LNSYM FWUEQ TELFH
QTELQ IAXXV ZPYTL LGAVP ARTKL IPTXX CIHYE UQR
```

The context suggests that the plaintext language is French.

Here are some statistics. The letter count

A	B	C	D	E	F	G	H	I	J	K	L	M
8	1	3	1	9	6	1	4	10	1	4	11	4
N	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
3	3	5	7	6	5	10	4	5	3	9	6	9

as well as the coincidence index 0.0437 suggest a polyalphabetic cipher, the autocincidence spectrum shows no meaningful period. The frequency distribution of the single letters hints at a running-key or autokey cipher that uses the standard alphabet (= TRITHEMIUS table).

A Probable Word

Since the message probably originated from the french embassy at Berlin in 1870 we may assume that the plaintext contains the word "allemand". Moving this word along the ciphertext and subtracting the probable word—with the help of the Perl script `probwd.pl`—we get 4 good matches (plus some weak ones):

```
000: LJHEHFFX     015: SNSVAPZC     030: ZTZHGRDU
001: UHXPTSNQ     016: YSOIDMSF     031: EZAOFQKZ
002: SXIBGAGJ     017: DOBLAFVW     032: KAHNEXPX
003: IIUOOTZQ     018: ZBEITIMO <-- 033: LHGMLCNQ
004: TUHWHMGW     019: MEBBWZEB     034: SGFTQAGC
```

```
005: FHPPATMG     020: PBUENRRT     035: RFMYOTSB
006: SPIIHZWF     021: MUXVFEJI     036: QMRWHFRK
007: AIBPNJVW     022: FXONSWYO     037: XRPPTEAB
008: TBIVXIMP     023: IOGAKLEW     038: CPIBSNRV
009: MIOFWZFV     024: ZGTSZRMB     039: AIUABELY <--
010: TOYENSLA <== 025: RTLHFZRH     040: TUTJSYOS
011: ZYXVGYQW     026: ELANNEXI <== 041: FTCAMBIL <--
012: JXOOMDMJ     027: WAGVSKYP     042: ECTUPVBF
013: IOHURZZM     028: LGOAYLFO     043: NTNXJOVT
014: ZHNZNMCJ     029: ROTGZSEN     044: ENQRCIJX

045: YQKKWWNE     060: VMDGNOIN     075: AGOYLIMV <--
046: BKDEKAUF     061: XDZVCVDF     076: RORTWZLE
047: VDXSOHVB     062: OZOKJQVM     077: ZRMENYUN
048: OXLWVIRI     063: KODREICQ     078: CMXVMHDI
049: ILPDWEYI     064: ZDKMWPGX     079: XXOUVQYK
050: WPWESLYU     065: OKFEDTNR     080: IONDELAP <==
051: AWXAZLKC     066: VFXLHAHK     081: ZNWMZNFV
052: HXTHZXSH     067: QXEPOUAU     082: YWFHBSLJ
053: ITAHLFXS     068: IEIWINKX     083: HFAJGYZC
054: EAATTKIU     069: PIPQBXNO     084: QACOMMST
055: LAMBYVKL     070: TPJJLAEW     085: LCHUAFJR
056: LMUGJXBH     071: AJCTORMZ     086: NHNITWHB
057: XUZRLOXW     072: UCMWFZPU     087: SNBBKURN
058: FZKTCKML     073: NMPNNCKF     088: YBUSIEDQ
059: KKMKYZBS     074: XPGVQXVW     089: MULQSQGB

090: FLJAETRI     105: IPMTJZCV     120: AGIGZICQ
091: WJTMHEYC     106: AMMRNPLQ     121: RIZHWPGU
092: UTFPSLSE     107: XMKVDYGI     122: TZAEDTKU
093: EFIAZFUN     108: XKOLMTYI     123: KAXLHXKZ
094: QITHTHDQ     109: VOEUHLYD     124: LXEPLXPF
095: TTABVQGB     110: ZENPZLTX     125: IEITLCVE
096: EAUDETRI <== 111: PNIHZGNS     126: PIMTQIUV
097: LUWMHEYN     112: YIAHUAIM     127: TMMYWHLB
098: FWFPSLDF     113: TAACOVCX     128: XMREVYRR
099: HFIAZQVX     114: LAVWJPNO     129: XRXDMEHN
100: QITHEINU     115: LVPRDAEQ
101: TTAMWAKU     116: GPKLORGH
102: EAFEOXKS     117: AKEWFTXI
103: LFXWLXIW     118: VEPNHKYF
104: QXPTLVMM     119: PPGPYLVM
```

Four good matches

The first good match occurs at position 10:

```
            1
01234 56789 01234 56789
LUSIT FSATM TZJIZ SYDZM PMFIZ
            ALLEM AND
            TOYEN SLA
```

A plausible completion to the left could be `CITOYENS`, giving

```
            1
01234 56789 01234 56789
LUSIT FSATM TZJIZ SYDZM PMFIZ
         RE ALLEM AND
         CI TOYEN SLA
```

The second good match occurs at position 26:

```
            1           2           3
01234 56789 01234 56789 01234 56789 01234 56789
LUSIT FSATM TZJIZ SYDZM PMFIZ REWLR ZEKLS RQXCA
                               ALLE MAND
                               ELAN NEXI
```

A plausible completion to the right could be `LANNEXIONDE` (“l’annexion de”), so we get

```
            1           2           3
01234 56789 01234 56789 01234 56789 01234 56789
LUSIT FSATM TZJIZ SYDZM PMFIZ REWLR ZEKLS RQXCA
                               ALLE MANDE ENT
                               ELAN NEXIO NDE
```

The third good match occurs at position 80:

```
7           8           9
01234 56789 01234 56789 01234 56789
TAUNX ARZCX IZYHQ LNSYM FWUEQ TELFH
            ALLEM AND
            IONDE LAP
```

The previous letter could be `T` (“. . . tion de la p. . . ”), providing not much help:

```
5           6           7           8           9
01234 56789 01234 56789 01234 56789 01234 56789 01234 56789
WAHIE LLXFK VXOKZ OVQIP TAUNX ARZCX IZYHQ LNSYM FWUEQ TELFH
                                  E ALLEM AND
                                  T IONDE LAP
```

And the fourth good match at position 96 also is not helpful:

```
8           9           10          11
01234 56789 01234 56789 01234 56789 01234 56789
IZYHQ LNSYM FWUEQ TELFH QTELQ IAXXV ZPYTL LGAVP
                   ALLE MAND
                   EAUD ETRI
```

Zig-Zag Exhaustion

The four good matches occur as two pairs whose positions differ by 16. This is a bit of evidence for an autokey cipher with a 16 letter key.

This is easily tested: If we really have an autokey cipher, then the fragments should match at another position too, preferably 16 positions apart. Let's try the longest one, `ELANNEXIONDE`, at position 26. We expect exactly one match beside the one we already know, at position $26 - 16 = 10$, or $26 + 16 = 42$. And we get

```
000: HJSVGBVSFZQV   026: ALLEMANDEENT <===
001: QHIGSODLYGWF   027: SARMRGOKDDUY
002: OXTSFWWEFMGE   028: HGZRXHVJCKZW
003: EIFFNPPLLWFV   029: NOEXYOUIJPXP
004: PUSNGIWRVVWO   030: VTKYFNTPONQB
005: BHAGZPCBUMPU   031: AZLFEMAUMGCA
006: OPTZGVMALFVZ   032: GASEDTFSFSBJ
007: WIMGMFLRELAV   033: HHRDKYDLRRKA
008: PBTMWECKKQWI   034: OGQKPWWXQABU
009: IIZWVVVQPMJL   035: NFXPNPIWZRVX
010: POJVMOBVLZMI   036: MMCNGBHFQLYR
011: VYIMFUGRYCJB   037: TRAGSAQWKOSK
012: FXZFLZCEBZCE   038: YPTSRJHQNILE
013: EOSLQVPHYSFV   039: WIFRAABTHBFS
014: VHYQMISERVWN   040: PUEARUENAVTW
015: ONDMZLPXUMOA   041: BTNRLXYGUJXD
016: USZZCIIALEBS   042: ACELORRAINEE <===
017: ZOMCZBLRDRTH   043: JTYOIKLOMUFA
018: VBPZSECJQJIN   044: ANBIBEZSTVBH
019: IEMSVVUWIYOV   045: UQVBVSDZURIH
020: LBFVMNHOXEWA   046: XKOVJWKAQYIT
```

```
021: IUIMEAZDDMBG   047: RDIJNDLWXYUB
022: BXZERSOJLRHH   048: KXWNUEHDXKCG
023: EORRJHURQXIO   049: ELAUVAODJSHR
024: VGEJYNCWWYPN   050: SPHVRHOPRXST
025: NTWYEVHCXFOM   ...  ...
```

a perfect accord with our expectations. This gives

```
3           4           5           6           7
01234 56789 01234 56789 01234 56789 01234 56789 01234 56789
ZEKLS RQXCA TFENE YBVOI WAHIE LLXFK VXOKZ OVQIP TAUNX ARZCX
              ELA NNEXI ONDE
              ACE LORRA INEE
```

and suggests “Alsace-Lorraine”. We complete the middle row that seems to be the keytext:

```
3           4           5           6           7
01234 56789 01234 56789 01234 56789 01234 56789 01234 56789
ZEKLS RQXCA TFENE YBVOI WAHIE LLXFK VXOKZ OVQIP TAUNX ARZCX
          A INELA NNEXI ONDE
          A LSACE LORRA INEE
```

If we repeat the fragment from row 3 in row 2 at position $55 = 39 + 16$ we see the very plausible text “l’annexion de l’Alsace-Lorraine”, and fill up the rows:

```
3           4           5           6           7
01234 56789 01234 56789 01234 56789 01234 56789 01234 56789
ZEKLS RQXCA TFENE YBVOI WAHIE LLXFK VXOKZ OVQIP TAUNX ARZCX
          A INELA NNEXI ONDEL ALSAC ELORR AINEE
          A LSACE LORRA INEET LAFFI RMATI ONDEL
```

To find the key we go backwards in zig-zag:

```
            1           2           3           4
01234 56789 01234 56789 01234 56789 01234 56789 01234 56789
LUSIT FSATM TZJIZ SYDZM PMFIZ REWLR ZEKLS RQXCA TFENE YBVOI
                           IR EALLE MANDE ENTRA INELA NNEXI
                           AI NELAN NEXIO NDELA LSACE LORRA
            1           2           3           4
01234 56789 01234 56789 01234 56789 01234 56789 01234 56789
LUSIT FSATM TZJIZ SYDZM PMFIZ REWLR ZEKLS RQXCA TFENE YBVOI
        SCI TOYEN SLAVI CTOIR EALLE MANDE ENTRA INELA NNEXI
        IRE ALLEM ANDEE NTRAI NELAN NEXIO NDELA LSACE LORRA
```

```
                1           2           3           4
01234 56789 01234 56789 01234 56789 01234 56789 01234 56789
LUSIT FSATM TZJIZ SYDZM PMFIZ REWLR ZEKLS RQXCA TFENE YBVOI
AUXAR MESCI TOYEN SLAVI CTOIR EALLE MANDE ENTRA INELA NNEXI
LAVIC TOIRE ALLEM ANDEE NTRAI NELAN NEXIO NDELA LSACE LORRA
```

Now it's certain that we have an autokey cipher and the key is "Aux armes, citoyens"—a line from the "Marseillaise". Using the key we easily decipher the complete plaintext:

> La victoire allemande entraîne l'annexion de l'Alsace-Lorraine et l'affirmation de la puissance allemande en Europe au détriment de l'Autriche-Hongrie et de la France.
>
> [Consequences of the German victory are the annexation of Alsace-Lorraine and the affirmation of the German power at the expense of Austria-Hungary and France.]

7.8 Similarity of Ciphers

Let Σ be an alphabet, $M \subseteq \Sigma^*$ a language, and K a finite set (to be used as keyspace).

Definition [SHANNON 1949]. Let $F = (f_k)_{k\in K}$ and $F' = (f'_k)_{k\in K}$ be ciphers on M with encryption functions

$$f_k, f'_k \colon M \longrightarrow \Sigma^* \quad \text{for all } k \in K.$$

Let $\tilde{F}$ and $\tilde{F'}$ be the corresponding sets of encryption functions. Then F is called **reducible** to F' if there is a bijection $A\colon \Sigma^* \longrightarrow \Sigma^*$ such that

$$A \circ f \in \tilde{F'} \quad \text{for all } f \in \tilde{F}.$$

That is, for each $k \in K$ there is a $k' \in K$ with $A \circ f_k = f'_{k'}$, see the diagram below.

F and F' are called **similar** if F is reducible to F', and F' is reducible to F.

$$\begin{tikzcd} & M \arrow[dl, "f_k"'] \arrow[dr, "f'_{k'}"] & \\ \Sigma^* \arrow[rr, "A"'] & & \Sigma^* \end{tikzcd}$$

Application. Similar ciphers F and F' are cryptanalytically equivalent—provided that the transformation $f \mapsto f'$ is efficiently computable. That means an attacker can break F if and only if she can break F'.

Examples

1. **Reverse** CAESAR. This is a monoalphabetic substitution with a cyclically shifted exemplar of the reverse alphabet `Z Y ... B A`, for example

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

   We have $K = \Sigma = \mathbb{Z}/n\mathbb{Z}$. Let $\rho(s) := n - s$ the reversion of the alphabet. Then encryption is defined by

$$f_k(s) := k - s \quad \text{for all } k \in K.$$

   This encryption function is involutory: $f_k \circ f_k(s) = k - (k - s) = s$. The ordinary CAESAR encryption is

$$f'_k(s) := k + s \quad \text{for all } k \in K.$$

   Then

$$\rho \circ f_k(s) = \rho(k - s) = n + s - k = (n - k) + s = f'_{n-k}(s),$$

   whence $\rho \circ f_k = f'_{\rho(k)}$. Because also the corresponding converse equation holds CAESAR *and Reverse* CAESAR *are similar.*

2. **The** BEAUFORT **cipher** [SESTRI 1710]. This is a periodic polyalphabetic substitution with a key $k = (k_0, \ldots, k_{l-1}) \in \Sigma^l$ (periodically continued):

$$f_k(a_0, \ldots, a_{r-1}) := (k_0 - a_0, k_1 - a_1, \ldots, k_{r-1} - a_{r-1}).$$

   Like Reverse CAESAR it is involutory. The alphabet table over the alphabet $\Sigma = \{\texttt{A}, \ldots, \texttt{Z}\}$ is in Figure 7.1. Compare this with TRITHEMIUS-BELLASO encryption:

$$f'_k(a_0, \ldots, a_{r-1}) := (k_0 + a_0, k_1 + a_1, \ldots, k_{r-1} + a_{r-1}).$$

   Then as with Reverse CAESAR we have $\rho \circ f_k = f'_{\rho(k)}$, and in the same way we conclude: *The* BEAUFORT *sipher is similar with the* TRITHEMIUS-BELLASO *cipher.*

3. **The Autokey cipher.** As alphabet we take $\Sigma = \mathbb{Z}/n\mathbb{Z}$. We write the encryption scheme as:

$$\begin{array}{rcl|rcl|rcl}
c_0 & = & a_0 + k_0 & & & & & & \\
c_1 & = & a_1 + k_1 & & & & & & \\
\vdots & & & & & & & & \\
c_l & = & a_l + a_0 & c_l - c_0 & = & a_l - k_0 & & & \\
\vdots & & & & & & & & \\
c_{2l} & = & a_{2l} + a_l & c_{2l} - c_l & = & a_{2l} - a_0 & c_{2l} - c_l + c_0 & = & a_{2l} + k_0 \\
\vdots & & & & & & & &
\end{array}$$

Let

$$A(c_0, \ldots, c_i, \ldots, c_{r-1}) = (\ldots, c_i - c_{i-l} + c_{i-2l} - \ldots, \ldots).$$

In explicit form the i-th component of the image vector looks like:

$$\sum_{j=0}^{\lfloor i \rfloor} (-1)^j \cdot c_{i-jl}.$$

and as a matrix A looks like

$$\begin{pmatrix} 1 & & -1 & & 1 & \\ & \ddots & & \ddots & & \ddots \\ & & 1 & & -1 & \\ & & & \ddots & & \ddots \\ & & & & 1 & \\ & & & & & \ddots \end{pmatrix}$$

Then

$$A \circ f_k(a) = f'_{(k,-k)}(a),$$

where $f'_{(k,-k)}$ is the TRITHEMIUS-BELLASO cipher with key $(k_0, \ldots, k_{l-1}, -k_0, \ldots, -k_{l-1}) \in \Sigma^{2l}$. Hence *the Autokey cipher is reducible to the* TRITHEMIUS-BELLASO *cipher with period* twice *the key length.* [FRIEDMAN und SHANNON] The converse is not true, the ciphers are not similar: This follows from the special form of the BELLASO key of an autokey cipher.

Note that A depends only on l. The reduction of the autokey cipher to the TRITHEMIUS-BELLASO cipher is noteworthy but practically useless: The encryption algorithm and the cryptanalysis are both more complicated when using this reduction. And the reduction is possible only after the keylength l is known.

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

Figure 7.1: The alphabet table of the Sestri-Beaufort cipher

Chapter 8

Transpositions

All the cryptographic procedures that we considered up to now worked by replacing each plaintext letter by another one, letter per letter. In this chapter we follow a complementary approach: Don't change the letters but instead change their order. This approach also goes back to anitiquity.

8.1 Transpositions and Their Properties

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/8_Transpos/Definition.html`

8.2 Examples

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/8_Transpos/Examples.html`

Constructing a Turning Grille

Let $l \in \mathbb{N}$ be a natural number ≥ 2. Draw a $2l \times 2l$ square and divide it into four $l \times l$ squares.

1	$\ldots$	l		$\ldots$	1
$\vdots$		$\vdots$	$\vdots$		$\vdots$
	$\ldots$	l^2	l^2	$\ldots$	l
l	$\ldots$	l^2	l^2	$\ldots$	
$\vdots$		$\vdots$	$\vdots$		$\vdots$
1	$\ldots$		l	$\ldots$	1

In the first square (upper left) enumerate the positions consecutively from 1 to l^2, and transfer these numbers to the other three squares, rotating the scheme by 90° to the right in each step, as shown in the table above.

A key consists of a choice of one of the four $l \times l$ squares for each of the numbers $1, \ldots, l^2$. Then make a hole at the corresponding position in the corresponding square, for a total of l^2 holes.

Thus the size of the keyspace is 4^{l^2}. For small l this amounts to:

Parameter l:	3	4	5	6
# Keys:	2^{18}	2^{32}	2^{50}	2^{72}

For $l = 6$ or more the keyspace is sufficiently large. However this doesn't make the cipher secure.

8.3 Cryptanalysis of a Columnar Transposition (Example)

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/8_Transpos/ColTrAnal.html`

8.4 Cryptanalytic Approaches

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/8_Transpos/Approach.html`

Conditional Bigram Log-Weights

Let L be a language over the alphabet $\Sigma = (s_0, \ldots, s_{n-1})$ with letter probabilities p_i and bigram probabilities p_{ij} for the bigrams $s_i s_j$. Then the conditional bigram probabilities are given by

$$p_{j|i} = p_{ij}/p_i \quad \text{for } i, j = 0, \ldots, n-1.$$

The number $p_{j|i}$ is the probability that given the letter s_i as beginning of a bigram (an event that occurs with probability p_i) the second letter of the bigram is s_j. For convenience we set $p_{j|i} = 0$ if $p_i = 0$.

Then for a set of independent bigrams the probabilities multiply, and it's usual to consider the logarithms of the probabilties to get sums instead of products. Adding a constant to the sum amounts to multiplying the probabilities by a constant factor. With an eye to the conditional bigram frequencies of natural languages, see the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/8_Transpos/Bigrams.html`, we choose a factor of 1000 and define the **conditional Bigram Log-Weight** (cBLW) of the bigram $s_i s_j$ by the formula

$$w_{ij} = \begin{cases} {}^{10}\log(1000 \cdot p_{j|i}) & \text{if } 1000 \cdot p_{j|i} > 1, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } i, j = 0, \ldots, n-1.$$

Given a family $\mathcal{B}$ of bigrams we define its **cBLW score** as

$$S_3(\mathcal{B}) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} k_{ij}(\mathcal{B}) \cdot w_{ij}$$

where $k_{ij}(\mathcal{B})$ is the number of occurrences of the bigram $s_i s_j$ in $\mathcal{B}$.

8.5 Bigram Frequencies

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology /Classic/8_Transpos/Bigrams.html`

8.6 The Values of Bigram Scores

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology /Classic/8_Transpos/cBLWsc.html`

Theoretical Values for Random Bigrams

Let $\Sigma = (s_0, \ldots, s_{n-1})$ be an alphabet and consider a probability distribution that assigns the probabilities p_i to the letters s_i. Choosing two letters independently from this distribution assigns the probability $p_i p_j$ to the bigram $s_i s_j$. Giving the bigrams whatever weights w_{ij} and scoring a set of bigrams by summing their weights the expected value of the weight of a bigram is

$$\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} w_{ij} p_i p_j.$$

Using this formula with the letter and bigram frequencies of natural languages and the corresponding conditional bigram log-weights we get the table

English:	1.47	German:	1.54	French:	1.48

Theoretical Values for True Bigrams

For a “true” bigram we first choose the first letter s_i with probability p_i, then we choose the second letter s_j with conditional probability $p_{j|i}$. This assigns the probability $p_i p_{j|i} = p_{ij}$ to the bigram $s_i s_j$, and the expected conditional bigram log-weight is

$$\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} w_{ij} p_{ij}.$$

Using this formula with the letter and bigram frequencies of natural languages and the corresponding conditional bigram log-weights we get the table

English:	1.94	German:	1.96	French:	1.99

Empirical Values for Natural Languages

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/8_Transpos/cBLWsc.html`

8.7 A more systematic approach

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/8_Transpos/Analysis2.html`

8.8 The Similarity of Columnar and Block Transpositions

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/8_Transpos/Similar.html`

Permutation Matrices

Let $\sigma \in \mathcal{S}_p$ be a permutation of the numbers $1, \ldots, p$.

Let R be a ring (commutative with 1). Then σ acts on R^p, the free R-module with basis

$$e_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \ldots, \quad e_p = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix},$$

as the linear automorphism

$$\rho(\sigma) \quad \text{defined by} \quad \rho(\sigma)e_i = e_{\sigma i}.$$

This gives an injective group homomorphism

$$\rho\colon \mathcal{S}_p \longrightarrow GL(R^p).$$

How to express $\rho(\sigma)$ as a matrix? The vector

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_p \end{pmatrix} = x_1 e_1 + \cdots + x_p e_p$$

maps to

$$\rho(\sigma)x = x_1 e_{\sigma 1} + \cdots + x_p e_{\sigma p} = \begin{pmatrix} x_{\sigma^{-1}1} \\ \vdots \\ x_{\sigma^{-1}p} \end{pmatrix}.$$

Thus the matrix P_σ corresponding to $\rho(\sigma)$ is given by

$$P_\sigma \begin{pmatrix} x_1 \\ \vdots \\ x_p \end{pmatrix} = \begin{pmatrix} x_{\sigma^{-1}1} \\ \vdots \\ x_{\sigma^{-1}p} \end{pmatrix} \quad \text{for all } x \in R^p.$$

Therefore

$$P_\sigma = (a_{ij})_{1 \le i,j \le p} \quad \text{where} \quad a_{ij} = \begin{cases} 1, & \text{if } i = \sigma j, \\ 0 & \text{otherwise.} \end{cases}$$

Hence the matrix P_σ has exactly one 1 in each row and in each column, all other entries being 0. We call P_σ the **permutation matrix** belonging to σ.

Matrix Description of a Block Transposition

The permutation σ defines a block transposition f_σ over the alphabet $\Sigma = \mathbb{Z}/n\mathbb{Z}$: For $(a_1, \ldots, a_p) \in \Sigma^p$ let

$$f_\sigma(a_1, \ldots, a_p) = \left[P_\sigma \begin{pmatrix} a_1 \\ \vdots \\ a_p \end{pmatrix} \right]^T = (a_{\sigma^{-1}1}, \ldots, a_{\sigma^{-1}p}).$$

This moves the i-th letter a_i of the block to position σi.

More generally let $r = pq$ and $a = (a_1, \ldots, a_r) \in \Sigma^r$. Then

$$c = f_\sigma(a) = (a_{\sigma^{-1}1}, \ldots, a_{\sigma^{-1}p}, a_{p+\sigma^{-1}1}, \ldots, a_{p+\sigma^{-1}p}, \ldots, a_{(q-1)p+\sigma^{-1}p}).$$

From this we derive the general encryption formula:

$$c_{i+(j-1)p} = a_{\sigma^{-1}i+(j-1)p} \quad \text{for } 1 \le i \le p, 1 \le j \le q.$$

We may express this in matrix notation writing the plaintext as a matrix with $a_{i+(j-1)p}$ in row i and column j:

$$A = \begin{pmatrix} a_1 & a_{p+1} & \cdots & a_{(q-1)p+1} \\ \vdots & \vdots & a_{i+(j-1)p} & \vdots \\ a_p & a_{2p} & \cdots & a_{qp} \end{pmatrix} \in M_{p,q}(\mathbb{Z}/n\mathbb{Z}).$$

Analogously we write the ciphertext as $C \in M_{p,q}(\mathbb{Z}/n\mathbb{Z})$ where $C_{ij} = c_{i+(j-1)p}$ for $1 \le i \le p$, $1 \le j \le q$.

Then the encryption formula simply is the matrix product:

$$C = P_\sigma A$$

with the permutation matrix P_σ.

Matrix Description of a Columnar Transposition

The permutation σ also defines a columnar transposition g_σ over the alphabet $\Sigma = \mathbb{Z}/n\mathbb{Z}$: Writing the plaintext row by row in a $q \times p$-matrix gives just the transposed matrix A^T (again assume $r = pq$):

$$\begin{array}{rccccccc}
 & & & & & \downarrow & & \downarrow \\
\rightarrow & a_1 & \dots & a_p & & a_{\sigma^{-1}1} & \dots & a_{\sigma^{-1}p} \\
\rightarrow & a_{p+1} & \dots & a_{2p} & \mapsto & a_{p+\sigma^{-1}1} & \dots & a_{p+\sigma^{-1}p} \\
 & \vdots & a_{(\mu-1)p+\nu} & \vdots & & \vdots & a_{(\mu-1)p+\sigma^{-1}\nu} & \vdots \\
\rightarrow & a_{(q-1)p+1} & \dots & a_{qp} & & a_{(q-1)p+\sigma^{-1}1} & \dots & a_{(q-1)p+\sigma^{-1}p}
\end{array}$$

and the ciphertext is read off, as the little arrows suggest, column by column in the order given by σ. Thus the encryption function is given by:

$$\tilde{c} = g_\sigma(a_1, \dots a_r) = (a_{\sigma^{-1}1}, a_{p+\sigma^{-1}1}, \dots, a_{\sigma^{-1}p}, \dots, a_{(q-1)p+\sigma^{-1}p}).$$

The encryption formula is:

$$\begin{aligned}
\tilde{c}_{\mu+(\nu-1)q} &= a_{(\mu-1)p+\sigma^{-1}\nu} \quad \text{for } 1 \le \mu \le q, 1 \le \nu \le p \\
&= c_{\nu+(\mu-1)p}.
\end{aligned}$$

If we arrange $\tilde{c}$ column by column as a matrix

$$\tilde{C} = \begin{pmatrix} \tilde{c}_1 & \tilde{c}_{q+1} & \dots & \tilde{c}_{(p-1)q+1} \\ \vdots & \vdots & \tilde{c}_{\mu+(\nu-1)q} & \vdots \\ \tilde{c}_q & \tilde{c}_{2q} & \dots & \tilde{c}_{pq} \end{pmatrix} \in M_{q,p}(\mathbb{Z}/n\mathbb{Z}),$$

we see that

$$\tilde{C}^T = C = P_\sigma A.$$

This shows:

Proposition 6 *The result of the columnar transposition corresponding to $\sigma \in \mathcal{S}_p$ on Σ^{pq} arises from the result of the block transposition corresponding to σ by writing the latter ciphertext in p rows of width q and transposing the resulting matrix. This produces the former ciphertext in q rows of width p.*

In particular columnar transposition and block transposition are similar.

(The proposition describes the required bijection of Σ^* for strings of length pq.)

For texts of a length not a multiple of p this observation applies after padding up to the next multiple of p. For a columnar transposition with an uncompletely filled last row this does not apply. In spite of this we assess columnar and block transpositions as similar, and conclude: Although a columnar transposition permutes the text over its complete length without period, and therefore seems to be more secure at first sight, it turns out to be an *illusory complication.*

Chapter 9

Linear Ciphers

In 1929 the mathematician Lester HILL proposed the use of matrices for encryption. He published his idea in the American Mathematical Monthly. This cryptographic application of linear algebra piqued the curiosity of mathematicians. But its obvious weaknesses soon became evident, so it never found a serious application. The true importance of the method relied on the fact that it was the first systematic use of algebraic methods in cryptology. And by the way its cryptanalysis made clear how dangerous linearity in encryption functions is.

> Jack LEVINE later mentioned that he used this kind of cipher already in 1924 for a contribution to a youth magazine when he was a high-school student.

In this section we use the Appendix E on the Euclidean algorithm.

9.1 Matrices over Rings

Let R be a ring (commutative with 1). The "multiplicative group" of R is the group of invertible elements

$$R^\times = \{a \in R \mid ab = 1 \text{ for some } b \in R\} = \{a \in R \mid a \text{ divides } 1\}.$$

In the same way the (non-commutative) R-algebra $M_{qq}(R)$ of $q \times q$-matrices over R has a group of invertible elements ("general linear group")

$$GL_q(R) = \{A \in M_{qq}(R) \mid AB = \mathbf{1}_q \text{ for some } B \in M_{qq}(R)\}.$$

The determinant defines a multiplicative map

$$\mathrm{Det}\colon M_{qq}(R) \longrightarrow R,$$

and

$$A \in GL_q(R) \Longrightarrow AB = \mathbf{1}_q \text{ for some } B \Longrightarrow \operatorname{Det} A \cdot \operatorname{Det} B = \operatorname{Det} \mathbf{1}_q = 1$$
$$\Longrightarrow \operatorname{Det} A \in R^\times.$$

The converse implication is also true. For a proof we consider the adjoint matrix $\tilde{A} = (\tilde{a}_{ij})$ where

$$\tilde{a}_{ij} = A_{ji} = \operatorname{Det} \begin{pmatrix} a_{11} & \dots & a_{1,i-1} & a_{1,i+1} & \dots & a_{1q} \\ \vdots & & \vdots & \vdots & & \vdots \\ a_{j-1,1} & \dots & a_{j-1,i-1} & a_{j-1,i+1} & \dots & a_{j-1,q} \\ a_{j+1,1} & \dots & a_{j+1,i-1} & a_{j+1,i+1} & \dots & a_{j+1,q} \\ \vdots & & \vdots & \vdots & & \vdots \\ a_{q1} & \dots & a_{q,i-1} & a_{q,i+1} & \dots & a_{qq} \end{pmatrix}$$

Using this we can prove:

Proposition 7 *For $A \in M_{qq}(R)$ the following holds:*
(i) $A\tilde{A} = \operatorname{Det} A \cdot \mathbf{1}_q$.
(ii) $A \in GL_q(R) \Longleftrightarrow \operatorname{Det} A \in R^\times$*; if this is true, then*

$$A^{-1} = \frac{1}{\operatorname{Det} A}\tilde{A}.$$

Proof. (i) is the expansion rule for determinants.
(ii) immediately follows from (i). $\diamond$

In particular Det induces a group homomorphism $GL_q(R) \longrightarrow R^\times$.

Example For $R = \mathbb{Z}/n\mathbb{Z}$ the statement (ii) of Proposition 7 can be rewritten as:

$$A \in M_{qq}(\mathbb{Z}) \text{ is invertible } \bmod n \Longleftrightarrow \operatorname{Det} A \text{ is coprime with } n.$$

Remarks

1. The expenses for calculating the inverse matrix A^{-1} are, if statement (ii) is naively evaluated:
   - one $q \times q$-determinant with $q!$ summands, each with q factors,
   - q^2 determinants of size $(q-1) \times (q-1)$.

   This is extremely inefficient—it is exponential in q.
2. Using Gaussian elimination the expenses drop to $\mathrm{O}(q^3)$. But this is not quite true: Exact calculation produces rational numbers with *huge* numerators and denominators that require additional resources.

There is a modification of the elimination algorithm that uses only integers and is much more efficient, see the next section. However also this procedure produces large intermediate results.

An alternative algorithm uses the Chinese Remainder Theorem: Each ring homomorphism $\varphi\colon R \longrightarrow R'$ induces a homomorphism of R-algebras

$$\varphi_q\colon M_{qq}(R) \longrightarrow M_{qq}(R')$$

by componentwise evaluation. If $A \in M_{qq}$ is invertible, then

$$\varphi_q(A)\varphi_q(A^{-1}) = \varphi_q(AA^{-1}) = \varphi_q(\mathbf{1}_q) = \mathbf{1}_q.$$

Hence also $\varphi_q(A)$ is invertible. Furthermore $\operatorname{Det}\varphi_q(A) = \varphi(\operatorname{Det} A)$, so we have a commutative diagram

$$\begin{tikzcd} M_{qq}(R) \arrow[r, "\varphi_q"] \arrow[d, "\mathrm{Det}"'] & M_{qq}(R') \arrow[d, "\mathrm{Det}"] \\ R \arrow[r, "\varphi"'] & R' \end{tikzcd}$$

Applying this to $R = \mathbb{Z}$ we use the residue class homomorphisms $\mathbb{Z} \longrightarrow \mathbb{F}_p$ (p prime) for sufficiently many primes p such that the product of these primes is $> \operatorname{Det} A$. Then we calculate

- $\operatorname{Det} A \bmod p$ in all the fields $\mathbb{F}_p$ (avoiding huge numbers, since all intermediate results may be represented as numbers between 0 and $p-1$),
- $\operatorname{Det} A \in \mathbb{Z}$ using the Chinese Remainder Theorem.

9.2 Elimination over the Integers

How to solve systems of linear equations over the ring $\mathbb{Z}$ of integers? How to calculate determinants efficiently? How to find an inverse matrix? Like in linear algebra over fields also in the more general situation over rings the *triangularization* of matrices is crucial for finding efficient algorithms.

For a sufficiently general framework we consider three classes of rings (commutative, with 1, without zero divisors):

- **Factorial rings** (or UFD domains): All elements have a decomposition into primes, in particular any two elements have a greatest common divisor gcd (in general not unique).
- **Principal ideal domains:** Each ideal is a principal ideal. Principal ideal domains are factorial, and the gcd of any two elements is a linear combination of these two.

- **Euclidean rings:** They have a division with remainder. Euclidean rings are principal ideal domains. The gcd of two elements as well as its linear represenation can be efficiently calculated by the extended Euclidean algorithm.

The set of invertible matrices with determinant 1 over a ring is called the "special linear group" $SL_n(R) \subseteq GL_n(R)$. It is the kernel of the determinant homomorphism on $GL_n(R)$.

Lemma 9 *Let R be a principal ideal domain, $a_1, \ldots, a_n \in R$, and d a* $\operatorname{gcd}(a_1, \ldots, a_n)$*. Then there is an invertible matrix $U \in SL_n(R)$ such that*

$$U \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} d \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Proof. Since the case $n = 1$ is trivial we may assume $n \geq 2$.

If all $a_i = 0$, then the assertion is trivial. Otherwise we may assume without restriction that $a_1 \neq 0$ (after a permutation that is merged into U as permutation matrix—if necessary replace a 1 by -1 to make the determinant $= 1$).

Let $d_2 := \operatorname{gcd}(a_1, a_2)$ (*any* gcd because in general this is not unique). Then $d_2 \neq 0$ and $d_2 = c_1 a_1 + c_2 a_2$ is a linear combination. From this we get the equation

$$\begin{pmatrix} c_1 & c_2 \\ -\frac{a_2}{d_2} & \frac{a_1}{d_2} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} c_1 a_1 + c_2 a_2 \\ -\frac{a_2 a_1}{d_2} + \frac{a_1 a_2}{d_2} \end{pmatrix} = \begin{pmatrix} d_2 \\ 0 \end{pmatrix}$$

where the matrix of coefficients

$$C = \begin{pmatrix} c_1 & c_2 \\ -\frac{a_2}{d_2} & \frac{a_1}{d_2} \end{pmatrix} \quad \text{has } \operatorname{Det} C = \frac{c_1 a_1}{d_2} + \frac{c_2 a_2}{d_2} = 1$$

and therefore is invertible.

We proceed be induction: Assume for the general step that for some $i \geq 2$

$$U' \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} d' \\ 0 \\ \vdots \\ 0 \\ a_i \\ \vdots \\ a_n \end{pmatrix} \qquad \text{where } a_i \neq 0$$

Then as before we change two coordinates:

$$\begin{pmatrix} d' \\ a_i \end{pmatrix} \rightsquigarrow \begin{pmatrix} d'' \\ 0 \end{pmatrix}.$$

In this way we successively build the matrix U. $\diamond$

Remark The inverse of the matrix C in the proof is

$$C^{-1} = \begin{pmatrix} \frac{a_1}{d_2} & -c_2 \\ \frac{a_2}{d_2} & c_1 \end{pmatrix}$$

From this formula we see that U and U^{-1} together can be calculated by at most $n-1$ executions of the Euclidean algorithm, plus $n-1$ multiplications of $n \times n$-matrices plus at most $n-1$ multiplications of permutation matrices.

With the help of this lemma we can triangularise matrices. (A more refined analysis would lead to the HERMITEan normal form.)

Theorem 13 (i) *Let R be a principal ideal domain, and $A \in M_{pq}(R)$. Then there exists an invertible matrix $U \in SL_p(R)$ such that $H = UA$ has the form*

$$\begin{pmatrix} * & \dots & * \\ & \ddots & \vdots \\ & & * \\ 0 & & \end{pmatrix} \quad \textit{für } p \geq q, \qquad \begin{pmatrix} * & \dots & \dots & * \\ & \ddots & \dots & \\ 0 & & * & \end{pmatrix} \quad \textit{für } p < q.$$

(ii) *If R is Euclidean, then U and U^{-1} together can be calculated by at most $\frac{p(p-1)}{2}$ executions of the extended Euclidean algorithm.*

Special case Let $A \in M_{pp}(R)$ be a square matrix, and determine $H = UA$ as in the Theorem. Then

$$\operatorname{Det} A = \operatorname{Det} H = h_{11} \cdots h_{pp}.$$

If A is invertible, then $A^{-1} = (U^{-1}H)^{-1} = H^{-1}U$. The calculation of the inverse H^{-1} of the triangular matrix H is easy. Thus calculation of determinant and inverse are reduced to triangularisation.

Proof. We prove this by describing an algorithm. Let $r := \min\{p, q\}$. Initialize the algorithm by

$$H := A, \quad U := \mathbf{1}_p, \quad V := \mathbf{1}_p.$$

Then loop over $j = 1, \dots r$. The relations $UA = H$, $UV = \mathbf{1}_p$ are loop invariants.

- Assume that at the beginning of the j-th step H has the form:

$$\begin{pmatrix} * & & & & \\ & \ddots & & & * \\ & & * & & \\ & & & h_{jj} & \\ & 0 & & \vdots & \\ & & & h_{pj} & \end{pmatrix}$$

  If $h_{jj} = \ldots = h_{pj} = 0$ we finish step j. Otherwise we use the lemma and find a matrix $U' \in SL_{p-j+1}(R)$ together with its inverse $(U')^{-1}$ such that

$$U' \begin{pmatrix} h_{jj} \\ \ldots \\ h_{pj} \end{pmatrix} = \begin{pmatrix} d_j \\ 0 \\ \ldots \\ 0 \end{pmatrix}$$

  We have $\left(\begin{smallmatrix} \mathbf{1} & 0 \\ 0 & U' \end{smallmatrix}\right) \in SL_p(R)$. At the end of the loop we replace

$$U := \begin{pmatrix} \mathbf{1} & 0 \\ 0 & U' \end{pmatrix} U, \quad H := \begin{pmatrix} \mathbf{1} & 0 \\ 0 & U' \end{pmatrix} H, \quad V := V \begin{pmatrix} \mathbf{1} & 0 \\ 0 & (U')^{-1} \end{pmatrix}.$$

After finishing the last loop U and H have the desired form. $\diamond$

Summarizing the expenses we have to add $\frac{p(p-1)}{2}$ matrix multiplications and the same number of multiplications by permutation matrices. However the total expenses are not yet covered because bounds for the intermediate results are yet missing. More exact considerations give expenses of the order $\mathrm{O}(m^2n^5)$ where m is an upper bound for the number of digits of the entries of A and $n = \max(p, q)$. For further optimizations of this bound search the literature on algebraic algorithms.

Elimination in Residue Class Rings

Now how to invert a matrix $A \in GL_q(\mathbb{Z}/n\mathbb{Z})$? First interpret A as an integer matrix and determine $U \in SL_q(\mathbb{Z})$ such that $H = UA$ is an integer upper triangular matrix as in Theorem 13. Reduction $\bmod n$ conserves the equation $H = UA$ as well as $A^{-1} = H^{-1}U$. Since $A \bmod n$ is invertible all diagonal elements of H are invertible $\bmod n$.

9.3 The Linear Cipher

Description

The **alphabet** is $\Sigma = \mathbb{Z}/n\mathbb{Z}$ with the structure as a finite ring.

The **keyspace** is $K = GL_l(\mathbb{Z}/n\mathbb{Z})$, the multiplicative group of invertible matrices. Section 9.4 estimates the size of the keyspace.

We **encrypt** blockwise taking blocks of length l: For $k \in GL_l(\mathbb{Z}/n\mathbb{Z})$ and $(a_1, \ldots, a_l) \in (\mathbb{Z}/n\mathbb{Z})^l$ set

$$\begin{pmatrix} c_1 \\ \vdots \\ c_l \end{pmatrix} = f_k(a_1, \ldots, a_l) = k \cdot \begin{pmatrix} a_1 \\ \vdots \\ a_l \end{pmatrix}$$

or elementwise

$$c_i = \sum_{j=1}^{l} k_{ij} a_j \quad \text{für } i = 1, \ldots, l.$$

We **decrypt** with the inverse matrix:

$$\begin{pmatrix} a_1 \\ \vdots \\ a_l \end{pmatrix} = k^{-1} \cdot \begin{pmatrix} c_1 \\ \vdots \\ c_l \end{pmatrix}.$$

Related Ciphers

Special case: Taking k as permutation matrix P_σ for a permutation $\sigma \in \mathcal{S}_l$ the encryption function f_k is the block transposition defined by σ.

Generalization: The affine cipher. Choose as key a pair

$$(k, b) \in GL_l(\mathbb{Z}/n\mathbb{Z}) \times (\mathbb{Z}/n\mathbb{Z})^l.$$

Encrypt by the formula

$$c = ka + b.$$

Choosing the unit matrix for k (as special case) gives the Bellaso cipher with key b.

Remark The original cipher proposed by Hill first permuted the alphabet before applying the linear map. The correspondence between the letters and the numbers $0, \ldots, 25$ is treated as part of the key.

Example

As an illustration we take a "toy example" of unreasonable small dimension $l = 2$ and

$$k = \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}.$$

Then $\operatorname{Det} k = 77 - 24 = 53 \equiv 1 \bmod 26$ and

$$k^{-1} = \begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix}.$$

The table

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

gives the correspondence between letters and numbers.

Now the plaintext `Herr` $= (7, 4, 17, 17)$ is encrypted as

$$\begin{aligned} \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix} \begin{pmatrix} 7 \\ 4 \end{pmatrix} &= \begin{pmatrix} 77+32 \\ 21+28 \end{pmatrix} = \begin{pmatrix} 109 \\ 49 \end{pmatrix} = \begin{pmatrix} 5 \\ 23 \end{pmatrix}, \\ \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix} \begin{pmatrix} 17 \\ 17 \end{pmatrix} &= \begin{pmatrix} 187+136 \\ 51+119 \end{pmatrix} = \begin{pmatrix} 323 \\ 170 \end{pmatrix} = \begin{pmatrix} 11 \\ 14 \end{pmatrix}. \end{aligned}$$

Thus $f_k($`Herr`$) = (5, 23, 11, 14) =$ `FXLO`.

We verify this by decrypting:

$$\begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix} \begin{pmatrix} 5 & 11 \\ 23 & 14 \end{pmatrix} = \begin{pmatrix} 35+414 & 77+252 \\ 115+253 & 253+154 \end{pmatrix} = \begin{pmatrix} 7 & 17 \\ 4 & 17 \end{pmatrix}.$$

Assessment

+ The linear cipher is stronger than block transposition and BELLASO cipher.

+ The frequency distribution of the ciphertext letters is nearly uniform. An attack with ciphertext only doesn't find useful clues.

− The linear cipher is extremely vulnerable for an attack with known plaintext, see Section 9.5.

9.4 The Number of Invertible Matrices over a Residue Class Ring

We want as clearly as possible to get an idea how large the number

$$\nu_{ln} := \#GL_l(\mathbb{Z}/n\mathbb{Z})$$

of invertible $l \times l$ matrices over the residue class ring $\mathbb{Z}/n\mathbb{Z}$ is.

In the special case $l = 1$ the number ν_{1n} simply counts the invertible elements of $\mathbb{Z}/n\mathbb{Z}$ and is given as the value $\varphi(n)$ of the EULER φ-function.

In the general case we easily find a trivial *upper bound* for ν_{ln}:

$$\nu_{ln} \leq \#M_{ll}(\mathbb{Z}/n\mathbb{Z}) = n^{l^2}.$$

To find a *lower bound* we note that (over any ring R) matrices of the form

$$\begin{pmatrix} 1 & & \\ & \ddots & \\ * & & 1 \end{pmatrix} \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_l \end{pmatrix} \begin{pmatrix} 1 & & * \\ & \ddots & \\ & & 1 \end{pmatrix}$$

are always invertible if $d_1, \ldots, d_l \in R^\times$. This gives an injective map

$$R^{\frac{l(l-1)}{2}} \times (R^\times)^l \times R^{\frac{l(l-1)}{2}} \longrightarrow GL_l(R).$$

(Proof of injectivity: **Exercise.**) This gives the bound

$$\nu_{ln} \geq n^{\frac{l(l-1)}{2}} \cdot \varphi(n)^l \cdot n^{\frac{l(l-1)}{2}} = n^{l^2-l} \cdot \varphi(n)^l.$$

Taken together this yields:

Proposition 8

$$n^{l^2-l} \cdot \varphi(n)^l \leq \nu_{ln} \leq n^{l^2}.$$

Remarks

1. The idea of writing matrices as $A = VDW$ as above—where D is a diagonal matrix, V, a lower triangular matrix with only 1's in the diagonal, and W, an upper triangular matrix likewise with only 1's in the diagonal—gives an easy way of constructing invertible matrices without resorting to trial and error and calculating determinants. This method gives "almost all" invertible matrices—in the theory of algebraic groups this is the "big BRUHAT cell". Matrices of this type can be easily inverted by the formula $A^{-1} = W^{-1}D^{-1}V^{-1}$.

2. Two lower bounds for the φ-function that we cite without proofs yield handy bounds for ν_{ln}. The first of these bounds is

$$\varphi(n) > \frac{6}{\pi^2} \cdot \frac{n}{\ln n} \quad \text{for } n \geq 7.$$

   This yields

$$\nu_{ln} > n^{l^2-l} \cdot \left(\frac{6}{\pi^2} \cdot \frac{n}{\ln n}\right)^l = \frac{6^l}{\pi^{2l}} \cdot \frac{n^{l^2}}{(\ln n)^l} \quad \text{for } n \geq 7.$$

3. The other bound is

$$\varphi(n) > \frac{n}{2 \cdot \ln \ln n} \quad \text{for almost all } n.$$

   This yields

$$\nu_{ln} > \frac{1}{(2 \cdot \ln \ln n)^l} \cdot n^{l^2}$$

or

$$\frac{1}{(2\cdot\ln\ln n)^l} \quad < \quad \frac{\nu_{ln}}{n^{l^2}} \quad < \quad 1$$

for almost all n.

Conclusion "Very many" to "almost all" matrices in $M_{ll}(\mathbb{Z}/n\mathbb{Z})$ are invertible. But also note that asymptotically the quotient ν_{ln}/n^{l^2} is not bounded away from 0.

Example For $n = 26$ we give a coarser but very simple version of the lower bound from Proposition 8: From $\varphi(26) = 12$ we get

$$\nu_{l,26} \geq 26^{l^2-l}12^l > 16^{l^2-l}8^l = 2^{4l^2-l}.$$

This gives the bounds $\nu_{2,26} > 2^{14}$, $\nu_{3,26} > 2^{33}$, $\nu_{4,26} > 2^{60}$, $\nu_{5,26} > 2^{95}$. We conclude that the linear cipher is secure from exhaustion at least for block size 5.

Finally we derive an exact formula for ν_{ln}.

Lemma 10 *Let $n = p$ prime. Then*

$$\nu_{lp} = p^{l^2}\cdot\rho_{lp} \quad \textit{where} \quad \rho_{lp} = \prod_{i=1}^{l}\left(1-\frac{1}{p^i}\right).$$

In particular for fixed l the relative frequency of invertible matrices, ρ_{lp}, converges to 1 with increasing p.

Proof. We successively build an invertible matrix column by column and count the possibilities for each column. Since $\mathbb{Z}/p\mathbb{Z} = \mathbb{F}_p$ is a field the first column is an arbitrary vector $\neq 0$. This makes $p^l - 1$ choices.

Assume we have already chosen i columns. These must be linearly independent hence span a linear subspace of $\mathbb{F}_p^l$. This subspace consists of p^i elements. The $(i+1)$-th column then is an arbitrary vector outside of this subspace for which we have $p^l - p^i$ choices. Summing up this yields

$$\prod_{i=0}^{l-1}(p^l-p^i) = \prod_{i=0}^{l-1}p^l(1-p^{i-l}) = p^{l^2}\prod_{j=1}^{l}\left(1-\frac{1}{p^j}\right)$$

choices. $\diamond$

Lemma 11 *Let $n = p^e$ with p prime and $e \geq 1$.*

(i) *Let $A \in M_{ll}(\mathbb{Z})$. Then $A \bmod n$ is invertible in $M_{ll}(\mathbb{Z}/n\mathbb{Z})$ if and only if $A \bmod p$ is invertible in $M_{ll}(\mathbb{F}_p)$.*

(ii) *The number of invertible matrices in $M_{ll}(\mathbb{Z}/n\mathbb{Z})$ is*

$$\nu_{ln} = n^{l^2} \cdot \rho_{lp}.$$

(iii) *The relative frequency of invertible matrices in $M_{ll}(\mathbb{Z}/p^e\mathbb{Z})$ is ρ_{lp}, independent of the exponent e.*

Proof. (i) Since $\gcd(p, \operatorname{Det} A) = 1 \iff \gcd(n, \operatorname{Det} A) = 1$, both statements are equivalent with $p \nmid \operatorname{Det} A$.

(ii) Without restriction we may assume that A has all its entries in $[0 \dots n-1]$. Then we write $A = pQ + R$ where all entries of R are in $[0 \dots p-1]$ and all entries of Q are in $[0 \dots p^{e-1}-1]$. The matrix $A \bmod n$ is invertible if and only if $R \bmod p$ is invertible. For R we have ν_{lp} choices by Lemma 10, and for Q we have $p^{(e-1)l^2}$ choices. Taken together this proves the claim.

(iii) is a direct consequence of (ii). $\diamond$

Lemma 12 *For m and n coprime $\nu_{l,mn} = \nu_{lm}\nu_{ln}$.*

Proof. The Chinese Remainder Theorem gives a ring isomorphism

$$\mathbb{Z}/mn\mathbb{Z} \longrightarrow \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$$

and extends to an isomorphism of the (non-commutative) rings

$$M_{ll}(\mathbb{Z}/mn\mathbb{Z}) \longrightarrow M_{ll}(\mathbb{Z}/m\mathbb{Z}) \times M_{ll}(\mathbb{Z}/n\mathbb{Z}).$$

The assertion follows from the equality of the numbers of invertible elements. $\diamond$

Induction immediately yields:

Theorem 14 *For $n \in \mathbb{N}$*

$$\nu_{ln} = n^{l^2} \cdot \prod_{\substack{p \text{ prime} \\ p|n}} \rho_{lp}.$$

In particular the relative frequency of invertible matrices $\rho_{ln} = \nu_{ln}/n^{l^2}$ is independent from the exponents of the prime factors of n. The explicit formula is

$$\rho_{ln} = \prod_{\substack{p \text{ prime} \\ p|n}} \rho_{lp} = \prod_{\substack{p \text{ prime} \\ p|n}} \prod_{i=1}^{l} \left(1 - \frac{1}{p^i}\right).$$

Example For $n = 26$ the explicit formula is

$$\nu_{l,26} = 26^{l^2} \cdot \prod_{i=1}^{l} \left(1 - \frac{1}{2^i}\right)\left(1 - \frac{1}{13^i}\right)$$

This evaluates as $\nu_{1,26} = 12$, $\nu_{2,26} = 157,248$, $\nu_{3,26} = 1,634,038,189,056 \approx 1.5 \cdot 2^{40}$. Comparing this value of $\nu_{3,26}$ with the lower bound 2^{33} from above shows how coarse this bound is. For $l = 4$ we even get $\nu_{4,26} \approx 1.3 \cdot 2^{73}$, almost secure from exhaustion.

Exercise Let $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, ... the increasing sequence of the primes. Let $n_r = p_1 \cdots p_r$ for $r \geq 1$. Show that for fixed l

$$\lim_{r\to\infty} \rho_{ln_r} = 0.$$

This means that the relative frequency of invertible matrices is decreasing for this sequence of moduli. *Hint*: Let ζ be the RIEMANN ζ-function. Which values has ζ at the natural numbers $i \geq 1$?

9.5 Cryptanalysis of the Linear Cipher

Block Length

The block length l leaves its trace as a divisor of the ciphertext length. If however the sender conceals the procedure by padding with meaningless text the cryptanalyst has no choice than to try all possible lengths by brute force.

Known Plaintext

Cryptanalyzing the linear cipher needs known plaintext—or some probable plaintext and a bit of trial and error to find the correct position. If the cryptanalyst knows the block length l and has l blocks of known plaintext she only has to solve a system of linear equations. This amounts to known plaintext of l^2 letters, corresponding to the length of the key. In a few degenerate cases she needs some additional known plaintext.

Let $(a_{11}, \ldots, a_{l1}), \ldots, (a_{1l}, \ldots, a_{ll})$ be the blocks of known plaintext, not necessarily contiguous, and $(c_{11}, \ldots, c_{l1}), \ldots, (c_{1l}, \ldots, c_{ll})$, the corresponding ciphertext blocks.

This yields the matrix equation

$$\begin{pmatrix} k_{11} & \ldots & k_{1l} \\ \vdots & \ddots & \vdots \\ k_{l1} & \ldots & k_{ll} \end{pmatrix} \begin{pmatrix} a_{11} & \ldots & a_{1l} \\ \vdots & \ddots & \vdots \\ a_{l1} & \ldots & a_{ll} \end{pmatrix} = \begin{pmatrix} c_{11} & \ldots & c_{1l} \\ \vdots & \ddots & \vdots \\ c_{l1} & \ldots & c_{ll} \end{pmatrix},$$

in short: $kA = C$ in $M_{ll}(\mathbb{Z}/n\mathbb{Z})$. Note that the lowercase letter k also denotes an $l \times l$-matrix. In the lucky (but common) case where A is invertible we immediately solve for k and get the key

$$k = CA^{-1}.$$

Inverting a matrix is efficient by Section 9.2. Furthermore with high probability A is invertible, see Section 9.4. Otherwise the cryptanalyst needs some more plaintext. Instead of explicating the solution in detail we consider an example.

Example

Imagine the example of Section 9.3 is part of a longer text, and the plaintext `Herr` is known as well as its location. It consists of two blocks and defines the matrix

$$A = \begin{pmatrix} 7 & 17 \\ 4 & 17 \end{pmatrix}.$$

The determinant is $\operatorname{Det} A = 17 \cdot (7 \cdot 1 - 4 \cdot 1) = 17 \cdot 3 = 51 \equiv -1 \bmod 26$. The cryptanalyst has luck. She immediately calculates the inverse:

$$A^{-1} = \begin{pmatrix} 9 & 17 \\ 4 & 19 \end{pmatrix}.$$

From this she gets the key matrix:

$$k = \begin{pmatrix} 5 & 11 \\ 23 & 14 \end{pmatrix} \begin{pmatrix} 9 & 17 \\ 4 & 19 \end{pmatrix} = \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}.$$

Solving the Affine Cipher

For solving the affine cipher $c = ka + b$ the cryptanalyst in general needs $l + 1$ blocks of known plaintext $a_0, \ldots, a_l$. By forming differences she gets

$$\begin{aligned} c_l - c_0 &= k \cdot (a_l - a_0), \\ &\ldots \\ c_l - c_{l-1} &= k \cdot (a_l - a_{l-1}). \end{aligned}$$

This reduces the cryptanalysis to that of the linear cipher with l known plaintext blocks.

Summary

Linearity makes a cipher extremely vulnerable for a known plaintext attack. The reason is that systems of linear equations are easily solved, at least over rings that allow practical calculations. (This however is a basic prerequisite for a ring to be useful for cryptography.)

In constructing secure ciphers on wants to prevent known plaintext attacks. Therefore one has to bring in nonlinearity: Solving algebraic equation of higher degree is much more complex. Hence the memento:

Known plaintext is adversary to linearity.

Exercise. HILL's proposal comprised a permutation of the alphabet before applying the linear map. That means executing a monoalphabetic substitution first. Explore the effect on cryptanalysis.

Chapter 10

Theoretical Security

The theory of this section goes back to Claude SHANNON[22] (with later simplifications by HELLMAN[11]). In his paper SHANNON developed the first general mathematical model of cryptology as well as the analysis of cryptosystems by information theoretical methods. The basic question this theory asks is:

> *How much information about the plaintext is preserved in the ciphertext?*

(no matter how difficult or expensive the extraction of this information is.) If this information doesn't suffice to determine the plaintext, then the cipher is secure.

SHANNON's ideas are based on the information theory that he had developed before [21].

The practical value of SHANNON's theory is limited. But besides it there are almost no sufficient criteria for the security of cryptographic methods that are mathematically proved. In contrast there are lots of necessary criteria derived from cryptanalytic procedures. Lacking better ideas one tries to optimize the cryptographic procedures for these necessary conditions. We saw and shall see many instances of this in these lecture notes.

10.1 A Priori and A Posteriori Probabilities

Model Scenario

Consider

- a finite set $M_0 \subseteq M$ of possible plaintexts—for example all plaintexts of length r or of length $\leq r$,
- a finite set K of keys,
- a cipher $F = (f_k)_{k \in K}$ with $f_k : M \longrightarrow \Sigma^*$.

The restriction to a finite set M_0 allows us to handle probabilities in the naive way. It is no real restriction since plaintexts of lengths $> 10^{100}$ are extremely unlikely in this universe that has at most 10^{80} elementary particles.

Motivating Example

For English plaintexts of length 5 we potentially know exact a priori probabilities, say from a lot of countings. A small excerpt from the list is

Plaintext	Probability
hello	$p > 0$
fruit	$q > 0$
xykph	0
...	...

Now assume we see a monoalphabetically encrypted English text `XTJJA`. Without knowing the key—that is in a situation where all keys have the same probability—and without further context information we nevertheless assign to the single plaintexts different "a posteriori probabilities":

Plaintext	Probability
hello	$p_1 >> p$
fruit	0
xykph	0
...	...

Thus knowledge of the ciphertext alone (and knowledge of the encryption method) changed our information on the plaintext.

A "Bayesian" approach gives a general model of this observation.

Model

The probability of plaintexts is given as a function

$$P\colon M_0 \longrightarrow [0,1] \quad \text{where} \quad P(a) > 0 \quad \text{for all } a \in M_0$$
$$\text{and} \quad \sum_{a \in M_0} P(a) = 1.$$

(This is the a priori probability of plaintexts.)

The probability of keys is likewise given as a function

$$P\colon K \longrightarrow [0,1] \quad \text{such that} \quad \sum_{k \in K} P(k) = 1.$$

(By abuse of notation denoted by the same letter P.) In general we assume a uniform distribution $P(k) = 1/\#K$ for all $k \in K$.

The probability of ciphertexts derives from the probabilities of plaintexts and keys, implicitly assumed as independently chosen:

$$P\colon \Sigma^* \longrightarrow [0,1], \quad P(c) := \sum_{a\in M_0} \sum_{k\in K_{ac}} P(a)\cdot P(k),$$

where $K_{ac} := \{k \in K \mid f_k(a) = c\}$ is the set of all keys that transform a to c.

Remark 1 Only finitely many $c \in \Sigma^*$ have $P(c) \neq 0$. These form the set

$$C_0 := \{c \in \Sigma^* \mid P(c) > 0\}$$

of "possible ciphertexts".

Remark 2 We have

$$\begin{aligned}\sum_{c\in\Sigma^*} P(c) &= \sum_{c\in\Sigma^*} \sum_{a\in M_0} \sum_{k\in K_{ac}} P(a)\cdot P(k)\\ &= \sum_{a\in M_0} \sum_{k\in K} P(a)\cdot P(k)\\ &= \sum_{a\in M_0} P(a)\cdot \sum_{k\in K} P(k)\\ &= 1.\end{aligned}$$

The conditional probability for a ciphertext to stem from a given plaintext $a \in M_0$ is modeled by the function

$$P(\bullet|a)\colon \Sigma^* \longrightarrow [0,1], \quad P(c|a) := \sum_{k\in K_{ac}} P(k).$$

Remark 3 $\sum_{c\in\Sigma^*} P(c|a) = \sum_{k\in K} P(k) = 1.$

Remark 4 $P(c) = \sum_{a\in M_0} P(a)\cdot P(c|a).$

A Posteriori Probabilities of Plaintexts

The cryptanalyst is interested in the converse, the conditional probability $P(a|c)$ of a plaintext $a \in M_0$ if the ciphertext $c \in \Sigma^*$ is given.

First we describe the probability of the simultaneous occurrence of a and c as

$$P\colon M_0 \times \Sigma^* \longrightarrow [0,1], \quad P(a,c) := P(a)\cdot P(c|a).$$

Remark 5 Then

$$\sum_{a\in M_0} P(a,c) = \sum_{a\in M_0} P(a)\cdot P(c|a) = P(c).$$

The conditional probability of a plaintext is given by a function $P(\bullet|c)$ with $P(a,c) = P(c) \cdot P(a|c)$ by the BAYESian formula

$$P(a|c) := \begin{cases} \frac{P(a)\cdot P(c|a)}{P(c)} & \text{if } P(c) \neq 0, \\ 0 & \text{if } P(c) = 0. \end{cases}$$

Remark 6 $\sum_{c\in\Sigma^*} P(c)\cdot P(a|c) = \sum_{c\in\Sigma^*} P(a)\cdot P(c|a) = P(a)$ by Remark 3.

10.2 Perfect Security

Definition 1 The cipher F is called **perfectly secure** on M_0 (the finite set of all possible plaintexts) if $P(\bullet, c) = P$ on M_0 for all ciphertexts $c \in \Sigma^*$ of positive probability $P(c) > 0$.

Interpretation: This condition assures that the a posteriori probability $P(a|c)$ of each plaintext $a \in M_0$ is the same as the a priori probability $P(a)$. Or in other words, the cryptanalyst doesn't get any additional information on the plaintext by knowing the ciphertext.

Lemma 13 $\#M_0 \leq \#C_0$.

Proof. Let $l \in K$ be a fixed key with $P(l) > 0$. For every ciphertext $c \in f_l(M_0)$, say $c = f_l(b)$, we then have

$$P(c) = \sum_{a\in M_0} P(a) \cdot \sum_{k\in K_{ac}} P(k) \geq P(b) \cdot P(l) > 0.$$

Hence $c \in C_0$. From this follows that $f_l(M_0) \subseteq C_0$. Since f_l is injective also $\#M_0 \leq \#C_0$. $\diamond$

Lemma 14 *If F is perfectly secure, then $K_{ac} \neq \emptyset$ for all $a \in M_0$ and all $c \in C_0$.*

Proof. Assume $K_{ac} = \emptyset$. Then

$$P(c|a) = \sum_{k\in K_{ac}} P(k) = 0.$$

Hence $P(a|c) = 0 \neq P(a)$, contradiction. $\diamond$

Therefore each possible plaintext can be transformed into each possible ciphertext. The next lemma says that the number of keys must be *very* large.

Lemma 15 *If F is perfectly secure, then $\#K \geq \#C_0$.*

Proof. Since $\sum P(a) = 1$, we must have $M_0 \neq \emptyset$. Let $a \in M_0$. Assume $\#K < \#C_0$. Then there exists a $c \in C_0$ with $f_k(a) \neq c$ for every key $k \in K$, whence $K_{ac} = \emptyset$, contradiction. $\diamond$

Theorem 15 [Shannon] *Let F be perfectly secure. Then*

$$\#K \geq \#M_0.$$

That is the number of keys is at least as large as the number of possible plaintexts.

Proof. This follows immediately from Lemmas 13 and 15. $\diamond$

Theorem 16 [Shannon] *Let F be a cipher with*

$$P(k) = \frac{1}{\#K} \quad \textit{for all } k \in K$$

(that is all keys have the same probability) and

$$\#K_{ac} = s \quad \textit{for all } a \in M_0 \textit{ and all } c \in C_0.$$

with a fixed $s \geq 1$. Then F is perfectly secure. Furthermore $\#K = s \cdot \#C_0$.

Proof. Let $c \in C_0$ be a possible cipherext. Then for any possible plaintext $a \in M_0$:

$$\begin{aligned} P(c|a) &= \sum_{k \in K_{ac}} \frac{1}{\#K} = \frac{\#K_{ac}}{\#K} = \frac{s}{\#K}, \\ P(c) &= \sum_{a \in M_0} P(a) \cdot P(c|a) = \frac{s}{\#K} \cdot \sum_{a \in M_0} P(a) = \frac{s}{\#K} = P(c|a), \\ P(a|c) &= \frac{P(c|a)}{P(c)} \cdot P(a) = P(a). \end{aligned}$$

Therefore F is perfectly secure. The second statement follows from

$$K = \dot{\bigcup_{c \in C_0}} K_{ac}$$

for all $a \in M_0$. $\diamond$

10.3 Examples of Perfect Security

Trivial Examples

Example 0: $\#M_0 = 1$. This example is cryptological nonsense since the cryptanalyst knows the only possible plaintext a priori. Hence she cannot gain any additional information on the plaintext by knowing the ciphertext.

Let $M_0 = \{a\}$. For all $c \in C_0$ trivially $P(a|c) = 1 = P(a)$. Hence F is perfectly secure, no matter how it is defined.

Example 1: $\#M_0 = 2$. The smallest nontrivial example involves two possible plaintexts. Without restriction we may assume that $M_0 = \{0, 1\} = C_0 = K$. Let f_0 be the identity map on $\{0, 1\}$, and f_1, the transposition of 0 and 1. Furthermore let the two keys 0 and 1 have the same probability: $P(0) = P(1) = \frac{1}{2}$.

Then $K_{00} = K_{11} = \{0\}$, $K_{01} = K_{10} = \{1\}$. Theorem 16 tells us that F is perfectly secure.

The Shift Cipher

We provide $M_0 = K = C_0$ with a group structure, and let $F\colon M_0 \times K \longrightarrow C_0$ be the group composition, hence $f_k(a) = a * k$. The sets

$$K_{ac} = \{k \in K \mid a * k = c\} = \{a^{-1} * c\}$$

each consist of one element only. We let $P(k) = \frac{1}{\#K}$ for all keys $k \in K$. Then F is perfectly secure.

The Examples 0 and 1 are the special cases of the one- or two-element group. Also Examples 2 and 3 will be special cases.

Example 2: The Caesar Cipher. This is the shift cipher on the cyclic group $\Sigma = \mathbb{Z}/n\mathbb{Z}$ of order n.

Hence the Caesar cipher is perfecly secure, *if we encrypt messages of length 1 only and randomly choose an independent new key for each message.*

Example 3: The One-Time Pad. This is the collection of the shift ciphers on the groups $\Sigma^r = M_0$ where $\Sigma = \mathbb{Z}/n\mathbb{Z}$. Messages are texts of length r, and keys are *independently and randomly chosen* letter sequences of the same length r.

Because one has to choose a new key for each message this cipher has its name **One-Time Pad**. Imagine a tear-off calendar where each sheet contains a random letter. After use it is torn off and destroyed.

The One-Time Pad is the prototype of a perfect cipher.

The special case $\Sigma = \{0, 1\}$ gives the binary VERNAM/MAUBORGNE cipher, that is the bitstram encryption with a completely random sequence of key bits.

Counterexample: The Monoalphabetic Substitution. Set $M_0 = \Sigma^r$ and $K = \mathcal{S}(\Sigma)$. For $r = 5$ we saw already that

$$P(\texttt{fruit}|\texttt{XTJJA}) = 0 < q = P(\texttt{fruit}).$$

Therefore the monoalphabetic substitution is not perfect (for $r \geq 2$ and $n \geq 2$). For $r = 1$ it is perfect by Theorem 16 (with $s = (n-1)!$).

10.4 Density and Redundancy of a Language

SHANNON's theory provides an idea of an unbreakable cipher via the concept of perfection. Moreover it develops the concept of "unity distance" as a measure of the difference to perfection. This concept takes up the observation that the longer a ciphertext, the easier is its unique decryption.

We don't want to develop this theory in a mathematically precise way, but only give a rough impression. For a mathematiclly more ambitious approach see [14].

Unique Solution of the Shift Cipher

Let the ciphertext `FDHVDU` be the beginning of a message that was encrypted using a CAESAR cipher. We solved it by exhaustion applying all possible 26 keys in order:

Key	Plaintext	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$
0	`fdhvdu`	+					
1	`ecguct`	+	+				
2	`dbftbs`	+					
3	`caesar`	+	+	+	+	+	+
4	`bzdrzq`	+					
5	`aycqyp`	+	+				
6	`zxbpxo`	+					
7	`ywaown`	?					
8	`xvznvm`	?					
9	`wuymul`	+	+				
10	`vtxltk`	+					
11	`uswksj`	+	+	?			
12	`trvjri`	+	+				
13	`squiqh`	+	+	+	+		
14	`rpthpg`	+					
15	`qosgof`	+					
16	`pnrfne`	+	+				
17	`omqemd`	+	+				
18	`nlpdlc`	+					
19	`mkockb`	+					
20	`ljnbja`	+					
21	`kimaiz`	+	+	+	?	?	
22	`jhlzhy`	+					
23	`igkygx`	+	+				
24	`hfjxfw`	+					
25	`geiwev`	+	+	+	?		

The flags in this table stand for:

- +: The assumed plaintext makes sense including the t-th letter.
- ?: The assumed plaintext could make sense including the t-th letter but with low probability.

Given the first five letters only one of the texts seems to make sense. We would call this value 5 the “unicity distance” of the cipher.

Mathematical Model

Let us start again with an n-letter alphabet Σ. The “information content” of a letter is $\log_2 n$, for we need $\lceil \log_2 n \rceil$ bits for a binary encoding of all of Σ.

Example For $n = 26$ we have $\log_2 n \approx 4.7$. Thus we need 5 bits for encoding all letters differently. One such encoding is the teleprinter code.

Now let $M \subseteq \Sigma^*$ be a language. Then $M_r = M \cap \Sigma^r$ is the set of "meaningful" texts of length r, and $\Sigma^r - M_r$ is the set of "meaningless" texts. Denote the number of the former by

$$t_r := \#M_r.$$

Then $\log_2 t_r$ is the "information content" of a text of length r or the **entropy** of M_r. This is the number of bits we need for distinguishing the elements of M_r in a binary encoding.

Remark More generally the entropy is defined for a model that assigns the elements of M_r different probabilities. Here we implicitly content ourselves with using a uniform probability distribution.

We could consider the relative frequency of meaningful texts, t_r/n^r, but instead we focus on the **relative information content**,

$$\frac{\log_2 t_r}{r \cdot \log_2 n}:$$

For an encoding of Σ^r we need $r \cdot \log_2 n$ bits, for an encoding of M_r only $\log_2 t_r$ bits. The relative information content is the factor by which we can "compress" the encoding of M_r compared with that of Σ^r. The complimentary portion

$$1 - \frac{\log_2 t_r}{r \cdot \log_2 n}$$

is "redundant".

Usually one relates these quantities to $\log_2 n$, the information content of a single letter, and defines:

Definition 2 (i) The quotient

$$\rho_r(M) := \frac{\log_2 t_r}{r}$$

is called the r**-th density**, the difference $\delta_r(M) := \log_2 n - \rho_r(M)$ is called the r**-th redundancy** of the language M.

(ii) If $\rho(M) := \lim_{r\to\infty} \rho_r(M)$ exists, it is called the **density** of M, and $\delta(M) := \log_2 n - \rho(M)$ is called the **redundancy** of M.

Remarks

1. Since $0 \leq t_r \leq n^r$, we have $\overline{\lim}\, \rho_r(M) \leq \log_2 n$.
2. If $M_r \neq \emptyset$, then $t_r \geq 1$, hence $\rho_r(M) \geq 0$. If $M_r \neq \emptyset$ for almost all r, then $\underline{\lim}\, \rho_r(M) \geq 0$.
3. If $\rho(M)$ exists, then $t_r \approx 2^{r\rho(M)}$ for large r.

For natural languages one knows from empirical observations that $\rho_r(M)$ is (more or less) monotonically decreasing. Therefore density and redundancy exist. Furthermore $t_r \geq 2^{r\rho(M)}$. Here are some empirical values (for $n = 26$):

M	$\rho(M) \approx$	$\delta(M) \approx$
English	1.5	3.2
German	1.4	3.3

The redundancy of English is $\frac{3.2}{4.7} \approx 68\%$ (but [3] says 78%; also see [13]). One expects that an English text (written in the 26 letter alphabet) can be compressed by this factor. The redundancy of German is about $\frac{3.3}{4.7} \approx 70\%$ [13].

10.5 Unicity Distance

We now apply our findings on the redundancy to the exhaustion of the key space. We don't deal with the expenses but only consider the feasibility. We follow the simplified approach of Hellman.

Assumptions

1. All meaningful texts of length r have the same probability. [Otherwise we get more complicated formulas. For natural languages this assumption is clearly false when r is small. However for large r we might hope that it follows from the usual stochastic conditions.]

2. The densitiy $\rho(M)$ of the language M exists. [Otherwise we could derive only a bound.]

3. All keys $k \in K$ have the same probability and they are $h = \#K$ in number.

4. All encryption functions f_k for $k \in K$ respect the lengths of the texts, or in other words $f(M_r) \subseteq \Sigma^r$.

Now let $c \in \Sigma^r$ be a ciphertext. In general—if all encryption functions f_k are different—it fits h possible plaintexts of length r in Σ^r. By far not all of them are meaningful but only

$$h \cdot \frac{t_r}{n^r} \approx \frac{h \cdot 2^{r\rho(M)}}{2^{r \cdot \log_2 n}} = h \cdot 2^{-r\delta(M)}.$$

We expect a unique solution in M_r if

$$h \cdot 2^{-r\delta(M)} \leq 1, \quad \log_2 h - r\delta(M) \leq 0, \quad r \geq \frac{\log_2 h}{\delta(M)},$$

at least if all encryption functions f_k are different; otherwise we should replace $\log_2 h$ with $d = d(F)$, the effective key length of the cipher F.

This motivates the following definition:

Definition 3. For a cipher F with effective key length $d(F)$ defined on a language M of redundancy $\delta(M)$ we call

$$\mathrm{UD}(F) := \frac{d(F)}{\delta(M)}$$

the **unicity distance**.

Examples

We always assume the alphabet $\Sigma = \{\mathtt{A}, \ldots, \mathtt{Z}\}$ with $n = 26$, and the language $M =$ "English".

1. For the shift cipher we have $d = \log_2 26$, $\mathrm{UD} \approx 4.7/3.2 \approx 1.5$, not about 5 as suspected in the introductory example. This deviation might be due to the many inexact steps in the derivation. In particular for small r the approximation $t_r \approx 2^{r\rho(M)}$ is very inexact.

2. For the monoalphabetic substitution we have $d \approx 88.4$, $\mathrm{UD} \approx 88.4/3.2 \approx 27.6$. This result is in good concordance with empirical observations on the solvability of monoalphabetic cryptograms.

3. For the TRITHEMIUS-BELLASO cipher with period l we have $d \approx 4.7 \cdot l$, $\mathrm{UD} \approx 1.5 \cdot l$.

4. For PORTA's disk cipher we have $d \approx 88.4 + 4.7 \cdot l$, $\mathrm{UD} \approx 27.6 + 1.5 \cdot l$.

5. For the general polyalphabetic substitution with period l and independent alphabets $d \approx 122 \cdot l$, $\mathrm{UD} \approx 38 \cdot l$.

6. For the One-Time Pad over the group $G = \Sigma$ we have $M = K = C = \Sigma^*$, hence $\#K = \infty$. However it makes sense to interpret $d_r/\delta_r = r \cdot \log_2 n/0 = \infty$ as unicity distance.

10.6 Cryptological Applications

The unicity distance is a very coarse measure of the quality of a cipher. In modern cryptology it is almost never used. For an attack with known plaintext it is meaningless (except for perfect ciphers where it is ∞).

A large unicity distance is achieved by:

- a large key space,
- lowering the redundancy of the plaintext language, for example by compression.

Application 1: PORTA's disk cipher is not so much stronger than the TRITHEMIUS-BELLASO cipher because the unicity distance is greater only by the constant summand 27.6. For a longer period the complication by permuting the primary alphabet effects not much additional security.

Application 2: Another application of SHANNON's theory is to running text encryption. The cryptanalysis must derive two meaningful plaintexts of total length $2r$ from a ciphertext of length r. This can work only for a language of redundancy at least 50%.

More generally consider a q-fold running text encryption with q independent keytexts. If cryptanalysis is feasible, then meaningful plaintext of total length $(q+1)\cdot r$ is excavated from a ciphertext of length r. We conclude that the redundancy of the language is at least $\geq \frac{q}{q+1}$. Because the redundancy of German, 70%, is smaller than $\frac{3}{4}$ we conclude that a triple running text encryption is secure. For English that has somewhat less redundancy even a double running text encryption seems to be secure.

Application 3: The unicity distance may serve as an indication for how much ciphertext corresponding to a single key may be known to the enemy without being of use. Or in other words: How often the key must change.

A general short summary of SHANNON's theory consists of the rule: *A necessary condition for the solvability of a cipher is that "information content of the ciphertext + redundancy of the plaintext language" $\geq$ "information content of the plaintext + information content of the key".*

Appendix A

Permutations and Rejewski's Theorem

A.1 The Symmetric Group

A **permutation** is a bijective map of a set M onto itself. The permutations of M form a group $\mathcal{S}(M)$.

This group is (at least in discrete mathematics, including cryptologic applications) of particular interest when the set M is finite. In most applications the nature of the elements doesn't matter. (A more formal statement is: "A bijection between two sets M und N induces an isomorphism of the groups $\mathcal{S}(M)$ und $\mathcal{S}(N)$".) Therefore we often simply take the set $\{1,\ldots,n\}$ of natural numbers as our set M and denote the group $\mathcal{S}(M)$ by $\mathcal{S}_n$. This group is called the **symmetric group** of order n.

Proposition 1 *The symmetric group of order n has $n!$ elements:*

$$\#\mathcal{S}_n = n!.$$

Proof. A permutation π is uniquely determined by its values at the arguments $1,\ldots,n$. For $\pi(1)$ we have n possibilities, for $\pi(2)$ then $n-1$, ..., for $\pi(n-1)$ two and for $\pi(n)$ only one. This makes a total of $n!$. $\diamond$

(Note that the dots "..." are a sloppy version of a proof by complete induction. In the remainder of this text we write πx instead of $\pi(x)$.)

A.2 Description of Permutations

Often a permutation π of the set $\{1,\ldots,n\}$ is represented by its value table, written in two rows:

$$\begin{pmatrix} 1 & 2 & \ldots & n \\ \pi 1 & \pi 2 & \ldots & \pi n \end{pmatrix}.$$

Of course this representation my also be used with other sets M; for $M = \{\mathrm{A}, \ldots, \mathrm{Z}\}$, the alphabet of classic cryptology, a permutation is the same as a monoalphabetic substitution σ and denoted in the form

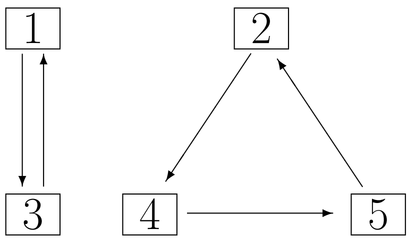
$$\begin{pmatrix} A & \ldots & Z \\ \sigma A & \ldots & \sigma Z \end{pmatrix}$$

(often without parantheses); below each letter we write its image under encryption.

Another description of a permutation π is the **cycle representation**. Let's illustrate this first with an example where $n = 5$: The permutation

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 1 & 5 & 2 \end{pmatrix}$$

has a natural graphical representation:



and this graph is completely characterized by the arrangement

$$(1\,3)(2\,4\,5)$$

of numbers. This means that each parenthesis defines a "cycle"—start with any element, write its image right of it, then the image thereof, and so on until you get back to the start. Then take any element that's not yet written down (if there is one) and do as before until all elements are met. Fixed points of the permutation yield cycles of length one. The general formula is

$$(a_1, \pi a_1, \ldots, \pi^{k_1-1} a_1) \cdots (a_i, \pi a_i, \ldots, \pi^{k_i-1} a_i) \cdots ,$$

where k_i is the smallest natural number ≥ 1 with $\pi^{k_i} a_i = a_i$.

This consideration shows:

Proposition 2 *Each permutation of a finite set has a decomposition into disjoint cycles. This representation is unique except for the order of the cycles and cyclic permutations of the elements inside the cycles.*

A.3 Group Theoretic Interpretation

A cycle by itself represents a permutation: permute its elements in the written order in a cyclic way, and let all other elements of M fixed.

Example: The cycle $(2\,4\,5)$ in $\mathcal{S}_5$ corresponds to the permutation

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 4 & 3 & 5 & 2 \end{pmatrix} \quad \text{or in cycle representation} \quad (1)(2\,4\,5)(3).$$

The cycle (i) in $\mathcal{S}_n$ defines the identity map, no matter which $i = 1, \ldots, n$ we choose. If we identify cycles with the permutations they describe, we immediately get:

Lemma 1 *Disjoint cycles commute as elements of the group $\mathcal{S}_n$.*

If we write the cycles of the cycle decomposition of a permutation next to each other, we just get the product of the corresponding permutations in $\mathcal{S}_n$. Therefore we may express Proposition 2 in the following way:

Corollary 8 *Each permutation is a product of disjoint cycles. This representation is unique except for the order of the factors.*

A.4 Partitions

If r_k is the number of cycles of length k of a permutation $\pi \in \mathcal{S}_n$, then we have

$$n \cdot r_n + \cdots + 1 \cdot r_1 = n.$$

Call a finite sequence $[s_1 s_2 \ldots s_m]$ of natural numbers with $s_1 \geq \ldots \geq s_m \geq 1$ a **partition** of n, if $n = s_1 + \cdots + s_m$. If we write down the cycle lengths of a permutation $\pi \in \mathcal{S}_n$ ordered by magnitude – each length with the multiplicity with which it occurs – then we get a partition of n. Call this the **(cycle) type** of π.

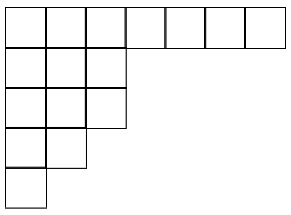
Example: The cycle type of

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 1 & 5 & 2 \end{pmatrix} = (1\,3)(2\,4\,5)$$

is

$$[3\,2].$$

We often visualise partitions by YOUNG **diagrams**. Given a partition $[s_1 s_2 \ldots s_m]$ of n we build the corresponding YOUNG diagram in the following way: Take m rows and put s_i squares in row i, left aligned. The partition $[7\,3\,3\,2\,1]$ of 16 has the diagram



(The defining condition of a YOUNG diagram is that none of the rows is longer than the row above it.)

A.5 Conjugate Permutations

Given $\pi, \rho \in \mathcal{S}_n$, how are the cycle representations of π and of the conjugate permutation $\rho\pi\rho^{-1}$ connected? First we consider the case of a single cycle π,

$$\pi = (a_1 \ \ldots \ a_k),$$

hence $\pi a_i = a_{1+(i \bmod k)}$ for $i = 1, \ldots, k$, all other elements being fixed by π. Then, for $b_i = \rho a_i$, we have

$$\rho\pi\rho^{-1}b_i = \rho\pi a_i = \rho a_{1+(i \bmod k)} = b_{1+(i \bmod k)},$$

hence

$$\rho\pi\rho^{-1} = (b_1 \ \ldots \ b_k).$$

Therefore also $\rho\pi\rho^{-1}$ is a cycle of length k.

Conjugating with ρ is an inner automorphism of the group $\mathcal{S}_n$, that means $\rho(\pi_1\pi_2)\rho^{-1} = (\rho\pi_1\rho^{-1})(\rho\pi_2\rho^{-1})$. Therefore in the general case we can conjugate the single cycles of π with ρ and get as a result the first part the following theorem:

Theorem 1 (i) *Let $\pi, \rho \in \mathcal{S}_n$ be two permutations. Then we get the cycle decomposition of the conjugate permutation $\rho\pi\rho^{-1}$ from that of π by replacing each cycle $(a_1 \ \ldots \ a_k)$ of π with the cycle $(\rho a_1 \ \ldots \ \rho a_k)$.*

(ii) *Two permutations of a finite set are conjugated if and only if they have the same cycle type.*

In other words: The conjugacy classes of the symmetric group $\mathcal{S}_n$ are in a natural correspondence with the partitions of n resp. with the YOUNG diagrams with exactly n squares.

Proof. We only have to show the inverse direction of statement (ii). To this end let $\sigma, \tau \in \mathcal{S}_n$ be of the same cycle type. Write the cycle decompositions of σ and τ below each other in such a way that cycles of the same length align; from this read off a permutation ρ with $\rho\sigma\rho^{-1} = \tau$: Simply map each element to the one below it. $\diamond$

This theorem, as simple as it is, is an essential ingredient to the cryptanalysis of the cipher machine Enigma, and therefore sometimes was called "the theorem that won world war II"; this is an obvious exaggeration, but with a certain confidence we may state that it *helped* in shortening the war in a significant way.

Exercise. Given $\sigma, \tau \in \mathcal{S}_n$, describe all solutions ρ of $\rho\sigma\rho^{-1} = \tau$. (For the case $\tau = \sigma$ see the next section.)

A.6 Centralizers of Permutations

Theorem 1 provides an easy approach to determining the centralizer of a permutation. First let us consider a single cycle $\pi = (a_1\, a_2\, \ldots\, a_k)$ of length $2 \leq k \leq n$. Then π acts transitively on the subset $A := \{a_1, a_2, \ldots, a_k\}$ and fixes all elements of the complement $\bar{A} = \{1, \ldots, n\} - A$. For $\rho \in \mathcal{S}_n$ the conjugate $\rho\pi\rho^{-1}$ is the cycle $(\rho a_1\, \ldots\, \rho a_k)$ by Theorem 1. By definition ρ centralizes π if and only if $\rho\pi\rho^{-1} = \pi$. Therefore for $\rho \in C_{\mathcal{S}_n}(\pi)$, the centralizer of π, we must have $\rho a_1 = a_i$ for some i, and then $\rho a_2 = a_{i+1}$ and so on, reducing the indices $\bmod n$ if necessary. That is, ρ acts on A as π^i, and on $\bar{A}$ as an arbitrary permutation. In the reverse direction each permutation with these properties centralizes π. Let $\mathcal{S}_n^A \leq \mathcal{S}_n$ be the subgroup of permutations that fix A elementwise. It is canonically isomorphic with $\mathcal{S}_{n-k}$. Using this notation we may formulate the result of our considerations as:

Proposition 3 *Let $\pi = (a_1\, a_2\, \ldots\, a_k) \in \mathcal{S}_n$ be a single cycle of length $2 \leq k \leq n$, and $A = \{a_1, a_2, \ldots, a_k\}$. Then the centralizer $C_{\mathcal{S}_n}(\pi)$ of π in $\mathcal{S}_n$ is the direct product of the subgroups $< \pi >$ and $\mathcal{S}_n^A$, and is isomorphic with the direct product $\mathcal{Z}_k \times \mathcal{S}_{n-k}$.*

Here $\mathcal{Z}_k$ is the cyclic group of order k.

We want to apply this result to arbitrary permutations. First we observe:

Proposition 4 *Let $\pi = \pi_1 \cdots \pi_s$ be a product of disjoint cycles π_i. For $k = 1, \ldots, n$ let*

$$A_k := \{a \mid 1 \leq k \leq n, a \text{ is in a cycle of } \pi \text{ of length } k\}.$$

Let $\rho \in \mathcal{S}_n$ centralize π. Then $\rho(A_k) = A_k$ for all k, and $\rho|A_k$ centralizes $\pi|A_k$.

Proof. Let $\pi_i = (a_{i1} \cdots a_{ik})$ be a cycle of length k. Then $\rho\pi_i\rho^{-1} = (\rho a_{i1} \cdots \rho a_{ik})$ is a cycle of length k, and $\rho\pi\rho^{-1} = \rho\pi_1\rho^{-1} \cdots \rho\pi_l\rho^{-1}$ is the unique decomposition into disjoint cycles. If ρ centralizes π, then $(\rho a_{i1} \cdots \rho a_{ik})$ is one of cycles of π of length k. Therefore $\rho(A_k) = A_k$. The second assertion follows because the actions of π and ρ on $\{1,\ldots,n\}$ directly decompose into the actions on the subsets A_k. $\diamond$

Proposition 4 reduces the task of determining the centralizer to the case where all the cycles π_i have the same length k. Let $\pi_i = (b_{i1}\ \ldots\ b_{ik})$, and $B_i := \{b_{i1},\ldots,b_{ik}\}$. Then $\{1,\ldots,n\} = B_1\dot\cup\cdots\dot\cup B_s$ (and $n = ks$).

Now consider the centralizer $C := C_{\mathcal{S}_n}(\pi)$, and take a $\rho \in C$. Then ρ doesn't necessarily respect the subsets B_i, but it permutes them: There is a unique $j = \bar\sigma i$—depending on ρ—such that $\rho(B_i) = B_j$. This defines a permutation $\bar\sigma \in \mathcal{S}_s$ of the indices $1,\ldots,s$. This way we get a group homomorphism

$$\Phi\colon C \longrightarrow \mathcal{S}_s, \quad \rho \mapsto \bar\sigma.$$

Lift $\bar\sigma i$ to a permutation $\sigma \in \Phi^{-1}(\bar\sigma) \subseteq \mathcal{S}_n$ by setting $\sigma b_{ih} := b_{\bar\sigma i,h}$. Then also $\sigma \in C$, and $\sigma^{-1}\rho$ is in the subgroup

$$C^\circ := \ker\Phi = \{\tau \in C \mid \tau(B_i) = B_i \text{ for } i = 1,\ldots,s\}$$

of permutations that centralize π and respect the B_i. The following characterization of this subgroup is immediate, because for $\tau \in C^\circ$ the restriction $\tau|B_i$ centralizes $\pi_i|B_i$ and therefore is a power of $\pi_i|B_i$.

Lemma 2 *The subgroup C° is the set of permutations with cycle decomposition of the type $\pi_1^{a_1}\cdots\pi_s^{a_s}$, and is isomorphic with the direct product $\mathcal{Z}_k^s$ of s cyclic groups $\mathcal{Z}_k$. This isomorphism defines an embedding $e\colon \mathcal{Z}_k^s \longrightarrow C$. The sequence*

$$\mathbf{1} \longrightarrow \mathcal{Z}_k^s \xrightarrow{e} C_{\mathcal{S}_n}(\pi) \xrightarrow{\Phi} \mathcal{S}_s \longrightarrow \mathbf{1}$$

is exact. The centralizer $C_{\mathcal{S}_n}(\pi)$ has $k^s\cdot s!$ elements.

This result easily generalizes to the general case. Let $\pi = \pi_1\cdots\pi_s$ be a product of disjoint cycles π_i, let k_i be the length of π_i, and let r_k be the number of cycles of length $k_i = k$, for $k = 1,\ldots,n$. Note that $r_1+\cdots+nr_n = n$, and many of the r_k are 0. Then we have a natural epimorphism

$$\Phi\colon C \longrightarrow \prod_{k=1}^{n}\mathcal{S}_{r_k},$$

with kernel

$$C^\circ := \ker\Phi = <\pi_1>\cdots<\pi_s> \cong \prod_{i=1}^{s}\mathcal{Z}_{k_i}$$

We sum this up to a Theorem.

Theorem 2 *For each permutation $\pi \in \mathcal{S}_n$ we have a natural exact sequence*

$$\mathbf{1} \longrightarrow \prod_{i=1}^{s}\mathcal{Z}_{k_i} \xrightarrow{e} C_{\mathcal{S}_n}(\pi) \xrightarrow{\Phi} \prod_{k=1}^{n}\mathcal{S}_{r_k} \longrightarrow \mathbf{1}$$

where the k_i are the lengths of the cycles of π and the r_k are the numbers of cycles of π of length k.

The centralizer $C_{\mathcal{S}_n}(\pi)$ of π has

$$\#C_{\mathcal{S}_n}(\pi) = \prod_{i=1}^{s} k_i \cdot \prod_{k=1}^{n} r_k!$$

elements.

Example. In $\mathcal{S}_n$ both permutations $(13)(245)$ and $(245) = (245)(1)(3)$ have a 6 element centralizer isomorphic with $\mathcal{Z}_3 \times \mathcal{Z}_2$. Its elements (in both cases) are the three different powers of (245) times the two different powers of (13).

A.7 Transpositions

A **transposition** is a cycle of length 2, that is a permutation that interchanges two elements and fixes all the other ones. The formula

$$(a_1\, a_2\, \ldots\, a_k) = (a_1\, a_k) \cdots (a_1\, a_3)(a_1\, a_2)$$

shows:

Lemma 3 *Each cycle of length k can be written as a product of $k-1$ transpositions.*

From this and Proposition 2 we conclude:

Corollary 9 *Each permutation π can be written as a product of $n-r$ transpositions where r is the number of cycles with more than one element in the cycle decomposition of π.*

Note that these transpositions need not be disjoint, therefore generally they don't commute, and the decomposition into transpositions is not unique. Even the number of transpositions is not unique; but at least we have:

Proposition 5 *If we write a permutation $\pi \in \mathcal{S}_n$ as a product of transpositions in different ways, then the number of transpositions either is always even or always odd.*

Proof. Let $\pi = \tau_1 \cdots \tau_s$ where the τ_i are transpositions. On the other hand let $\pi = \zeta_1 \cdots \zeta_r$ be the decomposition into disjoint cycles (complete, that means including all cycles of length 1). If we multiply π from the left with a transposition $\tau = (a\, b)$, we can distinguish two cases:

Case 1. a und b are in the same cycle. Because the cycles commute we may assume that this is the first one $\zeta_1 = (a_1 \ldots a_k)$, and $a = a_1$, $b = a_i$. Then $\tau\pi$ has the effect that

$$\begin{array}{rcl} a_1 \overset{\pi}{\mapsto} & a_2 & \overset{\tau}{\mapsto} a_2 \\ & \vdots & \\ a_{i-1} \mapsto & a_i & \mapsto a_1 \\ a_i \mapsto & a_{i+1} & \mapsto a_{i+1} \\ & \vdots & \\ a_k \mapsto & a_1 & \mapsto a_i \end{array}$$

Therefore $\tau\pi = (a_1 \ldots a_{i-1})(a_i \ldots a_k)\zeta_2 \cdots$ (all other cycles unchanged).

Case 2. a and b are in different cycles. Assume that these are the first two $\zeta_1 = (a_1 \ldots a_k)$ and $\zeta_2 = (b_1 \ldots b_l)$, and $a = a_1$, $b = b_1$. Then $\tau\pi = (a_1 \ldots a_k\, b_1 \ldots b_l)\zeta_3 \cdots$.

In any case the number of cycles grows by 1 or decreases by 1, hence is $r \pm 1$. If we multiply with another transposition from the left, the total number of cycles becomes $r + 2$, r or $r - 2$. After multiplication with q transpositions we have $r + t_q$ cycles, where $t_q \equiv q \pmod 2$. Therefore the product $\tau_s \cdots \tau_1\pi$ has $r + t_s$ cycles where $t_s \equiv s \pmod 2$. But this is the identy map $\pi^{-1}\pi$ and therefore $r + t_s = n$. Hence $s \equiv n - r \pmod 2$, no matter what was the starting decomposition into transpositions. $\diamond$

A.8 The Alternating Group

If we assign to each permutation in $\mathcal{S}_n$ the parity of the number of transpositions in an arbitrary decomposition, then, by the last section, we get a well-defined function

$$\mathrm{sgn} : \mathcal{S}_n \longrightarrow \mathbb{F}_2,$$

that obviously is a group homomorphism into the additive group. We call the kernel the **alternating group** of order n and denote it by $\mathcal{A}_n$. The elements of $\mathcal{A}_n$, that is the permutations that decompose into an even number of transpositions, are called **even** permutations, the other ones **odd**. $\mathcal{A}_n$ is a normal subgroup of index 2 in $\mathcal{S}_n$ and therefore has $n!/2$ elements.

A.9 Involutions

Call a permutation an **involution**, if it has order 2 as a group element in $\mathcal{S}_n$, or alternativly, if its cycle decomposition consists of transpositions (and fixed points) only. An involution ist **proper**, if it has no fixed points.

Of course this is possible only, if n is even. Then a proper involution is a product of $n/2$ disjoint 2-cycles (i. e. cycles of length 2).

A task that occurs in computing the total number of keys of Enigma, is determining the number of involutions in the symmetric group $\mathcal{S}_n$ that have exactly k 2-cycles where $0 \leq 2k \leq n$. It equals the number $d(n,k)$ of possibilities of choosing k pairs from n elements (where the order of the pairs does not matter).

Choose	**possibilities**	**choose**	**possibilities**
1st element: 1st partner:	n $n-1$	1st pair:	$n(n-1)/2$
2nd element: 2nd partner:	$n-2$ $n-3$	2nd pair:	$(n-2)(n-3)/2$
...	...	...	...
k-th element: k-th partner:	$n-2(k-1)$ $n-2(k-1)-1$	k-th pair:	$(n-2k+2)(n-2k+1)/2$

Adding all together and respecting the order we get

$$\frac{n(n-1)\cdots(n-2k+2)(n-2k+1)}{2^k} = \frac{n!}{(n-2k)!\cdot 2^k}$$

possibilities. If we now disregard the order we have always $k!$ identical choices. Hence we have shown:

Proposition 6 *The number of involutions in the symmetric group $\mathcal{S}_n$ that have exactly k 2-cycles is*

$$d(n,k) = \frac{n!}{2^k k!(n-2k)!} \quad \textit{for } 0 \leq 2k \leq n.$$

Example: In the case of the Wehrmacht Enigma we have $n = 26$ and $k = 10$, and the number of possible involutions is

$$\frac{26!}{2^{10}\cdot 10!\cdot 6!} = 150738274937250.$$

A.10 Products of Proper Involutions

The cryptanalysis of the Enigma by Rejewski involves products of two proper involutions σ and τ. Let $(a\,b)$ be a cycle of τ. If $(a\,b)$ is also a cycle of σ, then $\sigma\tau$ fixes the two elements a and b, hence has the two cycles (a) and (b) of length 1.

In the general case starting with an arbitrary element a_1 one finds a chain $a_1, a_2, a_3, \ldots, a_{2k}$ such that

$$\begin{aligned} \tau &= (a_1\,a_2)(a_3\,a_4)\cdots(a_{2k-1}\,a_{2k}) \quad \times \text{ other 2-cycles},\\ \sigma &= (a_2\,a_3)(a_4\,a_5)\cdots(a_{2k}\,a_1) \quad \times \text{ other 2-cycles}. \end{aligned}$$

In the product $\sigma\tau$ these become the two cycles

$$(a_1\, a_3\, \ldots\, a_{2k-1})(a_{2k}\, \ldots\, a_4\, a_2)$$

of length k. In particular all cycle lengths occur in an even number, the cycle type is **matched**.

Theorem 3 [Rejewski] *A permutation is the product of two proper involutions, if and only if its cycle type is matched.*

Proof. In order to prove the inverse direction we take a permutation π of matched type and give solutions σ, τ of the equation $\sigma\tau = \pi$.

In the simplest case, where π only consists of two cycles of the same length:

$$\pi = (p_1\, p_2\, \ldots\, p_k)(q_1\, q_2\, \ldots\, q_k),$$

an obvious solution is

$$\begin{aligned} \tau &= (p_1\, q_k)(p_2\, q_{k-1})\cdots(p_k\, q_1), \\ \sigma &= (p_2\, q_k)(p_3\, q_{k-1})\cdots(p_1\, q_1). \end{aligned}$$

In the general case we analogously construct the solution for each matching pair of cycles of the same length. ◇

Therefore the following procedure gives a decomposition of a partition of matched type into two proper involutions: Write cycles of the same length below each other, the lower one in reverse direction. Then read off the 2-cycles of τ by pairing the elements in the same column, and the 2-cycles of σ by pairing each element with the one diagonally to the left below it.

Example: Let $\pi =$ `(D)(K)(AXT)(CGY)(BLFQVEOUM)(HJPSWIZRN)`. Then we write down the scheme

```
(D)(AXT)(BLFQVEOUM)
(K)(YGC)(NRZIWSPJH)
```

and read off a solution of $\sigma\tau = \pi$:

τ = `(DK)(AY)(XG)(TC)(BN)(LR)(FZ)(QI)(VW)(ES)(OP)(UJ)(MH)`,
σ = `(DK)(XY)(TG)(AC)(LN)(FR)(QZ)(VI)(EW)(OS)(UP)(MJ)(BH)`.

It's also easy to find all solutions: Cyclically shift the lower cycles. If there are more then two cycles of the same length also consider all possible pairings. The solution is uniquely determined as soon as a 2-cycle of σ or τ is fixed for each cycle pair.

Exercise. Work out the formula for the number of solutions.

Appendix B

STIRLING's **Formula**

Following preliminary work by DE MOIVRE (1718) STIRLING in 1730 [27] stated his famous formula that expresses the factorial in a way that leads to a very useful assessment of its asymptotic behaviour. Here we reproduce the notably narrow bounds given by ROBBINS [19] following a method attributed to CESÀRO [2] and FISHER [6].

Theorem 1 *For all natural numbers* $n \geq 1$ *we have*

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \cdot e^{r_n}$$

where the error term r_n *is bounded by*

$$\frac{1}{12n+1} \leq r_n \leq \frac{1}{12n}$$

The approximation is illustrated by the following table, where s_n is the upper bound and t_n, the lower bound from the theorem.

n	1	2	3	4	5	6	7	8	9
s_n	1.002	2.001	6.001	24.001	120.003	720.01	5040.04	40320.2	362881.4
$n!$	1	2	6	24	120	720	5040	40320	362880
t_n	0.996	1.997	5.996	23.991	119.970	719.87	5039.33	40315.9	362850.1

This suggests that the upper bound is closer to the true value then the lower bound; and the absolute errors increase. The relative errors however decrease quite fast, see Corollary 10 below.

Proof. We consider the sequence

$$a_n = \frac{n!}{(\frac{n}{e})^n \cdot \sqrt{n}}$$

and show that it decreases monotonically; because all of its members are positive, we then know that it converges.

Dividing two consecutive terms we get

$$\frac{a_n}{a_{n+1}} = \frac{n!(\frac{n+1}{e})^{n+1}\cdot\sqrt{n+1}}{(\frac{n}{e})^n\cdot\sqrt{n}\cdot(n+1)!} = \frac{1}{e}\cdot(\frac{n+1}{n})^{n+1/2},$$

$$\log\frac{a_n}{a_{n+1}} = -1 + (n+\frac{1}{2})\cdot\log\frac{n+1}{n}.$$

Lemma 2 below immediately gives

$$0 < \frac{1}{12}\cdot(\frac{1}{n+\frac{1}{12}} - \frac{1}{n+\frac{1}{12}+1}) < \log\frac{a_n}{a_{n+1}} < \frac{1}{12}\cdot(\frac{1}{n} - \frac{1}{n+1}).$$

From the left inequality we conclude $a_n > a_{n+1}$ as claimed.

Now let $a = \lim_{n\to\infty} a_n$. Then $a \geq 0$ and by telescoping

$$\frac{1}{12}\cdot(\frac{1}{n+\frac{1}{12}} - \frac{1}{n+\frac{1}{12}+k}) < \log\frac{a_n}{a_{n+k}} < \frac{1}{12}\cdot(\frac{1}{n} - \frac{1}{n+k}).$$

For $k\to\infty$ we get

$$\frac{1}{12n+1} \leq \log\frac{a_n}{a} \leq \frac{1}{12n},$$

$$e^{\frac{1}{12n+1}} \leq \frac{a_n}{a} \leq e^{\frac{1}{12n}}.$$

To complete the proof of the theorem we have to show that $a = \sqrt{2\pi}$.

From WALLIS' product formula, see Lemma 3 below, and using $k! = a_k k^{k+1/2}/e^k$, we get

$$\sqrt{\pi} = \lim_{n\to\infty}\frac{a_n^2\cdot n^{2n+1}\cdot 2^{2n}\cdot e^{2n}}{e^{2n}\cdot a_{2n}\cdot(2n)^{2n+1/2}\cdot\sqrt{n+1/2}}$$

$$= a\cdot\lim_{n\to\infty}\frac{\sqrt{n}}{\sqrt{2}\cdot\sqrt{n+1/2}} = \frac{a}{\sqrt{2}}.$$

Therefore $a = \sqrt{2\pi}$. $\diamond$

Lemma 1 *For* $0 < x < 1$

$$\frac{3x}{3-x^2} < \frac{1}{2}\log\frac{1+x}{1-x} < x\cdot\left(1+\frac{1}{3}\cdot\frac{x^2}{1-x^2}\right).$$

Proof. For $|x| < 1$ we have the well-known power series expansion

$$\frac{1}{2}\log\frac{1+x}{1-x} = x + \frac{x^3}{3} + \frac{x^5}{5} + \ldots = \sum_{\nu=1}^{\infty}\frac{x^{2\nu-1}}{2\nu-1}.$$

For $0 < x < 1$ we get the upper bound

$$\frac{1}{2}\log\frac{1+x}{1-x} < x + \frac{x^3}{3} + \frac{x^5}{3}\cdots = x + \sum_{\nu=2}^{\infty}\frac{x^{2\nu-1}}{3} = x + \frac{x^3}{3}\left(1 + x^2 + x^4 + \cdots\right)$$

$$= x + \frac{x^3}{3}\cdot\frac{1}{1-x^2} = x\cdot\left(1 + \frac{1}{3}\cdot\frac{x^2}{1-x^2}\right).$$

For the lower bound we use

$$\frac{1}{2}\log\frac{1+x}{1-x} > x + \frac{x^3}{3} + \frac{x^5}{9}\cdots = \sum_{\nu=1}^{\infty}\frac{x^{2\nu-1}}{3^{\nu-1}} = x\cdot\sum_{\nu=0}^{\infty}\frac{x^{2\nu}}{3^{\nu}} = x\cdot\frac{1}{1-\frac{x^2}{3}}.$$

◇

Lemma 2 *For* $n \in \mathbb{N}_1$

$$2 + \frac{1}{6}\cdot\left(\frac{1}{n+\frac{1}{12}} - \frac{1}{n+\frac{1}{12}+1}\right) < (2n+1)\cdot\log\frac{n+1}{n} < 2 + \frac{1}{6}\cdot\left(\frac{1}{n} - \frac{1}{n+1}\right)$$

Proof. In Lemma 1 we substitute $x = \frac{1}{2n+1}$. Then

$$\frac{1+x}{1-x} = \frac{1+\frac{1}{2n+1}}{1-\frac{1}{2n+1}} = \frac{2n+2}{2n} = \frac{n+1}{n}.$$

This gives the upper bound

$$\frac{1}{2}\cdot\log\frac{n+1}{n} < \frac{1}{2n+1}\cdot\left(1 + \frac{1}{3}\cdot\frac{1}{4n^2+4n}\right) = \frac{1}{2n+1}\cdot\left(1 + \frac{1}{12}\cdot\frac{1}{n(n+1)}\right),$$

as claimed. At the lower bound we get

$$\frac{1}{2}\cdot\log\frac{n+1}{n} > \frac{3(2n+1)}{3(2n+1)^2-1},$$

whence

$$(2n+1)\cdot\log\frac{n+1}{n} > \frac{6(2n+1)^2}{3(2n+1)^2-1} = 2 + \frac{2}{3(2n+1)^2-1} = 2 + \frac{2}{12n^2+12n+2}.$$

The lower bound we aim at evaluates to

$$2 + \frac{1}{6}\cdot\left(\frac{1}{n+\frac{1}{12}} - \frac{1}{n+\frac{1}{12}+1}\right) = 2 + 2\cdot\left(\frac{1}{12n+1} - \frac{1}{12n+13}\right)$$

$$= 2+2\cdot\frac{12}{(12n+1)(12n+13)} = 2+2\cdot\frac{12}{12\cdot 12n^2 + 14\cdot 12n + 13} = 2+2\cdot\frac{2}{12n^2+14n+\frac{13}{12}}$$

which is clearly smaller for $n \geq 1$. ◇

Lemma 3 (Product formula of WALLIS)

$$\sqrt{\pi} = \lim_{n\to\infty} \frac{2^{2n}\cdot (n!)^2}{(2n)!\cdot\sqrt{n+1/2}}.$$

Proof. Starting with the product expansion of the sine function,

$$\sin(\pi x) = \pi x \cdot \prod_{k=1}^{\infty}(1-\frac{x^2}{k^2}),$$

and substituting $x = 1/2$, we get

$$1 = \frac{\pi}{2}\cdot\prod_{k=1}^{\infty}\frac{4k^2-1}{4k^2},$$

$$\frac{\pi}{2} = \prod_{k=1}^{\infty}\frac{(2k)^4}{(2k-1)2k\cdot 2k(2k+1)} = \lim_{n\to\infty}\frac{2^{4n}\cdot (n!)^4}{((2n)!)^2(2n+1)},$$

and this immediately gives the assertion. $\diamond$

Corollary 10 *If we replace $n!$ by $s_n = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \cdot e^{\frac{1}{12n}}$, the relative error is bounded by*

$$1 \leq \frac{s_n}{n!} < e^{\frac{1}{(12n)^2}}.$$

Proof. Let $t_n = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \cdot e^{\frac{1}{12n+1}}$. Then

$$1 \leq \frac{s_n}{n!} \leq \frac{s_n}{t_n} = e^{\frac{1}{12n}-\frac{1}{12n+1}} = e^{\frac{1}{12n(12n+1)}} < e^{\frac{1}{(12n)^2}}.$$

$\diamond$

Note that the “usual” textbook estimate gives the lower bound $1 \leq r_n$. From this we get the bound $e^{\frac{1}{12n}}$ for the relative error that has only a linear term in the denominator of the exponential instad of the quadratic one.

Corollary 11 *For all natural numbers $n \geq 1$*

$$\sqrt{2\pi n}\cdot\left(1+\frac{1}{13n}\right) < \frac{n!\,e^n}{n^n} < \sqrt{2\pi n}\cdot\left(1+\frac{1}{11n}\right).$$

For $n\to\infty$

$$\frac{n!\,e^n}{n^n} = \sqrt{2\pi n} + \mathrm{O}(\frac{1}{\sqrt{n}}).$$

Proof. We use the inequality $e^x > 1 + x$ for all real $x \neq 0$. For $0 < x < 1$ we therefore have $1 - x \leq e^{-x}$, whence $e^x \leq \frac{1}{1-x} = 1 + \frac{1}{\frac{1}{x}-1}$. Therefore

$$\frac{n!\,e^n}{n^n} < \sqrt{2\pi n} \cdot \left(1 + \frac{1}{12n-1}\right) \leq \sqrt{2\pi n} \cdot \left(1 + \frac{1}{11n}\right).$$

For the lower bound we have

$$\frac{n!\,e^n}{n^n} > \sqrt{2\pi n} \cdot \left(1 + \frac{1}{12n+1}\right) \geq \sqrt{2\pi n} \cdot \left(1 + \frac{1}{13n}\right).$$

$\diamond$

Corollary 12 *For all natural numbers $n \geq 1$*

$$\frac{1}{\sqrt{2\pi n}} \cdot \left(1 - \frac{1}{12n}\right) < \frac{n^n}{n!\,e^n} < \frac{1}{\sqrt{2\pi n}} \cdot \left(1 - \frac{1}{14n}\right).$$

For $n \to \infty$

$$\frac{n^n}{n!\,e^n} = \frac{1}{\sqrt{2\pi n}} + \mathrm{O}(\frac{1}{\sqrt{n^3}}).$$

Proof. The lower bound is immediate from $1-x \leq e^{-x}$. For the upper bound we use $e^{-x} < \frac{1}{1+x} = 1 - \frac{1}{\frac{1}{x}+1}$, and get

$$\frac{n^n}{n!\,e^n} < \frac{1}{\sqrt{2\pi n}} \cdot \left(1 - \frac{1}{12n+2}\right) \leq \frac{1}{\sqrt{2\pi n}} \cdot \left(1 - \frac{1}{14n}\right).$$

$\diamond$

Appendix C

KASISKI's Test: Couldn't the Repetitions be by Accident?

C.1 Repetitions in a Polyalphabetic Ciphertext

Kasiski's method finds the period of a polyalphabetic cipher in the following way: If a string of characters repeatedly appears in the ciphertext, assume that the distance between the occurrences is a multiple of the period. Find as many repetitions as possible and calculate the greatest common divisor of the distances. This gives the period or a small multiple of it.

For the historic context of this method see [15]; Babbage had invented the method ten years earlier than Kasiski but never published his results, see [24].

Kasiski's method is based on the following observations [16, Section 14]:

1. If a plaintext is encrypted by distinct alphabets that cyclically repeat with a period of l, and if a certain sequence of letters occurs k times in the text, then it will be encrypted with the same sequence of alphabets k/l times in the mean.

2. If a repeating sequence of letters is encrypted by the same sequence of alphabets, then the ciphertext contains a repeated pattern; the distance of the two occurrences is a multiple of the period l.

3. Not every repeated pattern in the ciphertext arises in this way; but the probability of an accidental repetition is noticeably smaller.

Because of observation 3 the cryptanalyst has to omit some of the distances—by intuition, but essentially by trial and error. Therefore an obvious and natural question is: Is the probability of an accidental repetition really much smaller, as stated in 3?

The answer is a simple exercise in probability theory, a corollary of the Birthday Paradox. In spite of its simplicity, there seems to be no explicit reference to this result in the cryptologic literature in the context of Kasiski's method.

The goal of this paper is to show that elementary calculus may give a satisfying answer to the question in the title. The intermediate results might be improved by refined theoretic considerations. There is room for experimental mathematics as well. The final section discusses some open problems that make up suitable undergraduate projects.

Note. The Birthday Paradox also has other applications in cryptology, the most renowned is to hash functions: the Birthday Paradox tells how long the hashes should be in order to avoid collisions (= repetitions), see [17, Sections 9.5 and 9.7] [20, Section 7.4] [26, Section 7.3]. For statistical applications see [5, Chapter II, Section 3].

C.2 Counting Repetitions

In several situations we want to know the probability that certain data agree or that certain events repeat. Here are three sample questions:

- What is the probability that at least two of a group of people meeting accidentally in the same room share their birthdays?
- What is the probability that at least two of r randomly and independently chosen character strings of length t are the same?
- Draw r balls from an urn containing N distinct balls (with replacement). What is the probability that you get at least one of the balls twice?

Let us calculate the probability in the urn experiment. There are N possible events of which we observe r (with possible repetitions).

- The probability that the first event is a repetition is 0.
- Therefore the probability that the first event *is not* a repetition is $1 = \frac{N}{N}$.
- The probability that the second event is not a repetition is $\frac{N-1}{N}$.
- The probability that **then also** the third event is not a repetition is $\frac{N-2}{N}$. (There are $N-2$ choices left that don't give a repetition.)
- The general case: If there was no repetition among the first $r-1$ events, then the probability is $\frac{N-r+1}{N}$ that also the r-th event is not a repetition.

From this we get the following well-known result [5, chapter II, Section 3]:

Theorem 1 *The probability of a repetition in a sequence of r independent events from a set of N is*

$$K(N,r) = 1 - Q(N,r)$$

where

$$Q(N,r) = \frac{N \cdot (N-1) \cdots (N-r+1)}{N^r} = [1 - \frac{1}{N}] \cdots [1 - \frac{r-1}{N}].$$

C.3 Applications

Birthdays: For $N \approx 365.22$, $r = 23$, we have $Q(N,r) \approx 0.493$, therefore the probability of a coincidence is ≈ 0.507. *If there are 23 people in the same room, the probability that two of them share their birthdays, is greater then $\frac{1}{2}$.* From this observation the Birthday Paradox got its name.

Character strings: Consider strings over the alphabet $\{A,...,Z\}$. Choose r strings of length t randomly and independently: This makes $N = 26^t$ possible events. The probability that at least two strings are identical is $K(26^t, r)$. For $r = 100, 300, 1000, 5000$ let these probabilities be p_t, q_t, r_t, s_t, respectively. Direct calculation from Theorem 1—with the help of a small computer program—gives Table C.1. The table shows for example, that for $r = 1000$ there is more than a 60% chance that we find two identical four letter strings; but two identical five letter strings are rather unlikely (probability < 5%).

$t \rightarrow$	1	2	3	4	5	6	7	$r \downarrow$
p_t	1	**1.000**	**0.246**	0.011	0.00042			100
q_t	1	1.000	**0.923**	**0.094**	0.0038	0.00015		300
r_t	1	1	1.000	**0.665**	**0.041**	0.0016		1000
s_t	1	1	1.000	1.000	**0.651**	**0.040**	0.0016	5000

Table C.1: Probabilities for repetitions of strings. Entries $< 10^{-4}$ are omitted. Values given as 1 are exact, values given as 1.000 are rounded off. In each row the cut point "50% probability" lies between the two entries in boldface.

C.4 Bounds for the Number of Repetitions

The formula in Theorem 1 is awkward for manual calculation; it also gives no direct idea of the order of magnitude of the probability. Fortunately, using some elementary calculus, we find convenient bounds that also show the behaviour for large values of the parameters. First we derive an upper bound for the number $K(N,r)$ of repetitions:

- The probability that the i-th event is a repetition is $\leq \frac{i-1}{N}$, because there were only $i-1$ events before.
- Therefore the probability that up to the r-th event there is a repetition is
$$K(N,r) \leq \frac{0}{N} + \cdots + \frac{i-1}{N} + \cdots + \frac{r-1}{N} = \frac{r(r-1)}{2N}.$$

From this we get the right inequalities of Theorem 2.

Theorem 2 (i) *The probability $K(N,r)$ of a repetition is bounded by*
$$1 - e^{-\frac{r(r-1)}{2N}} \leq K(N,r) \leq \frac{r(r-1)}{2N}.$$

(ii) *If $r \leq \sqrt{2N}$, then we have*
$$(1-\frac{1}{e}) \cdot \frac{r(r-1)}{2N} \leq K(N,r) \leq \frac{r(r-1)}{2N}.$$
or, somewhat weaker,
$$0.3 \cdot \frac{r(r-1)}{N} \leq K(N,r) \leq 0.5 \cdot \frac{r(r-1)}{N}.$$

(iii) *If $r \leq \sqrt{N}$, then $K(N,r) < \frac{1}{2}$.*

(iv) *If $r \geq 1 + \sqrt{2N \ln 2}$, then $K(N,r) > \frac{1}{2}$.*

Proof. The left inequality in (i) follows from the inequality $1 - x \leq e^{-x}$ for $x \in \mathbb{R}$, hence
$$Q(N,r) \leq e^{-\frac{1}{N}} \cdots e^{-\frac{r-1}{N}} \leq e^{-\frac{r(r-1)}{2N}},$$
and $K(N,r) = 1 - Q(N,r)$.

The lower bound in (ii) follows from the inequality $1 - e^{-x} \geq (1-\frac{1}{e})x$ in the real interval $0 \leq x \leq 1$; and this is true because the function $f(x) = 1 - e^{-x}$ is concave ($\cap$-shaped), $g(x) = (1-\frac{1}{e}) \cdot x$ is linear, and $f(0) = g(0)$, $f(1) = g(1)$.

For (iii) the upper bound simplifies to $K(N,r) < \frac{r^2}{2N} \leq \frac{N}{2N} = \frac{1}{2}$.

In (iv) we have $r(r-1) > 2N \ln 2$. Therefore the left hand side of (i) is $> \frac{1}{2}$. $\diamond$

Theorem 2 (iii) and (iv) together give the rule of thumb that appears in many cryptography textbooks, see [20, Section 7.4] [26, Section 7.3]:

The cut point "50% probability" for repetitions is close to $r = \sqrt{N}$.

More exactly it is between $\sqrt{N}$ and $1 + 1.18\sqrt{N}$. As a special case of Theorem 2 (iii) with $N = n^t$ we immediately get

Theorem 3 *For r random character strings of length t over an alphabet of n characters with $r \leq n^{t/2}$ the probability of a repetition is less than $\frac{1}{2}$.*

C.5 The Probability of Accidental Repetitions

Now we apply this to the substrings of a random character string of length r (over an n letter alphabet), where "random" means that each character is chosen independently and with probability $\frac{1}{n}$. We abandon the exact mathematical reasoning and make the **simplifying assumption** that the substrings are stochastically independent; this is clearly not perfectly correct, because the substrings overlap—but see the discussion in the final section. We also neglect the fact that a string of length r has only $r - t + 1$ substrings of length t. Then the probability that a repetition of length t occurs is (approximately) $K(n^t, r)$, and Table C.1 above illustrates the order of magnitude of these numbers (when $n = 26$).

Theorem 3 immediately gives: For a random character string of length $r \leq n^{t/2}$ (over an n letter alphabet) the probability of a repetition of length t is $< \frac{1}{2}$. That means: *For random strings up to a length of $n^{t/2}$ a repetition of any substring of length t is fairly unlikely.* Or to express it conversely:

For random strings of length r a repetition of any substring of length t is rather unlikely ($< 50\%$) as long as

$$\text{(A)} \qquad t \geq 2 \cdot \frac{\log r}{\log n}\,.$$

For $n = 26$ the bound (A) is approximately $t \geq 1.413 \cdot \log r$ (logarithm in base 10).

This is the main answer to the title question. For a non-mathematician maybe we would express it as follows:

- For texts of length 100, accidential repetitions of length 3 or more are rather unlikely; Table C.1 gives the more exact result that the probability is $< 25\%$.
- For texts of length 300, accidential repetitions of length 4 or more are rather unlikely (Table C.1: probability $< 10\%$), but at least one acci-

dental repetition of length 3 occurs with high probability (Table C.1: $> 90\%$).

And so on—use formula (A), Table C.1, or Theorem 2.

One might wish to derive more statistical results on the probabilities of repetitions. However the simple statements given here are sufficient as a justification for Kasiski's method; in particular considering the cut point "50%" seems adequate for the cryptanalyst, even if this simplistic view is somewhat unsatisfactory for the mathematician.

C.6 Kasiski's Test

When the cryptanalyst carries out Kasiski's test he doesn't examine a random text. In order to apply the results of the preceding section we have to make one **further simplifying assumption**: a polyalphabetic ciphertext behaves randomly except for the effect of the period. Now when we find a repetition of length t, and t is as least as large as in (A), then we are pretty sure that we have found a true (or causal) repetition and the period is a divisor of the distance. The smaller t is, the more we are prepared to reject some repetitions; again Table C.1 gives more precise hints for ciphertexts of lenghts 100, 300, 1000, or 5000. If we find a "long" repetition, we may assume with extremely high probability that it is a causal repetition.

C.7 Discussion

Are the theoretical results above exact enough for Kasiski's test in view of the simplyfing assumptions that we had to make? Here we give only some coarse empirical results, leaving room for more elaborate investigations.

1. May we really apply the theorems and the resulting Table C.1 to the substrings of a long character string? to get empirical evidence Wwe generated 100 random texts of lengths 100 and 300 each, 26 random texts of lengths 1000, 5000 each, over the 26 character alphabet, and found no remarkable deviations from the theoretical results derived for independent strings.

2. Is the number of accidental repetitions in a polyalphabetic ciphertext really as low as in a random text? We encrypted 100 English plaintexts of length 300 with keys of lengths 6, 10, and 17 each (with mixed alphabets by the way). Here we found small deviations: The ciphertexts seem to have *fewer* accidental repetitions than random texts, see figures C.1 and C.2 for key lenghts 6 and 17. A partial explanation is given below.

These simulation results confirm that the formulas in this paper apply to polyalphabetic ciphertexts with negligeable deviations.

Here are some observations and heuristic arguments that could merit some further investigations:

- Why is the number of accidental repetitions in item 2 smaller than in random strings? One major effect is: there can be no accidental repetitions whose distance is a multiple of l, the period of the cipher; each such repetition must be causal since ciphertext and key conform. Therefore we expect that the number of repetitions (for any length) is smaller by $1/l$. However this is not yet the complete truth: Non-accidental, but "false", repetitions may arise in some other ways, as shown in [1, Section 17.4]: when the key contains a repeated substring—as in "seventyseven"— or when key and plaintext contain the same word, for example if the key for a military text contains the word "division". It seems hard to adequately adjust a general model to fit these observations. But unfortunately in exceptional cases these effects can lead to annoying *long* "accidental" repetitions, not predicted by the estimates above.

- How many causal repetitions can we expect? This depends on the statistics of the plaintext language. Possible approaches are:

  - *Simulation.* In the experiment of item 2 we also counted the causal repetitions and found significantly more causal than accidental repetitions. See Figures C.1 and C.2 for repetitions of length 3.
  - Start with *trigram freqencies* and calculate the resulting probabilities for repetitions of length three in a ciphertext, depending on the key length, under suitable simplifying assumptions.
  - Model the language by a *Markov source* of low order and derive the relevant probabilities.

- Consider the distribution of the number of repetitions of a fixed length—in random texts, or accidental or causal repetitions in ciphertexts. They all seem to follow a Poisson distribution. Determine the parameters.

Figures C.1 and C.2 show a selection of typical simulation results. The x-axis represents the number of repetitions of length 3 in one text; note that one long repetition of length $t \geq 3$ counts as $t - 2$ repetitions of length 3. The y-value shows how often exactly x repetitions occurred in 100 texts (all of length 300). The fat gray line gives this frequency for random texts and serves as reference, it is the same in both diagrams. The thin gray line gives the frequency of x accidental repetitions, the black line the frequency of x causal repetitions in the polyalphabetic ciphertexts.

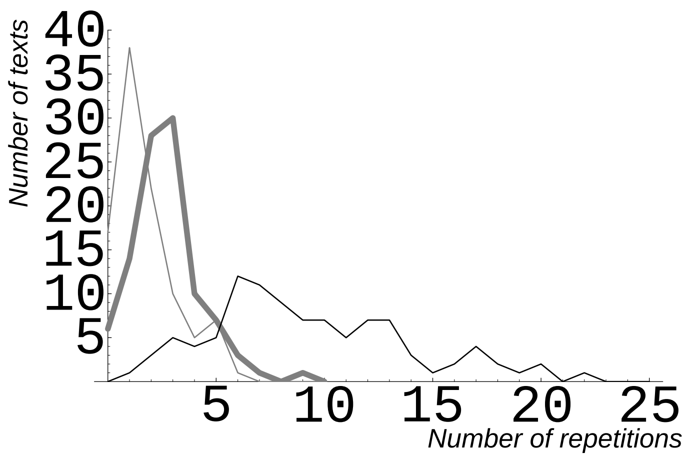


Figure C.1: Distribution of the number of repetitions in polyalphabetic ciphertexts, key length 6. x-axis: number of repetitions of length 3, y-axis: number of occurrences of x repetitions. Fat gray line: random texts, thin gray line: accidental repetitions, black line: causal repetitions; one count of 31 causal repetitions falls outside the picture.

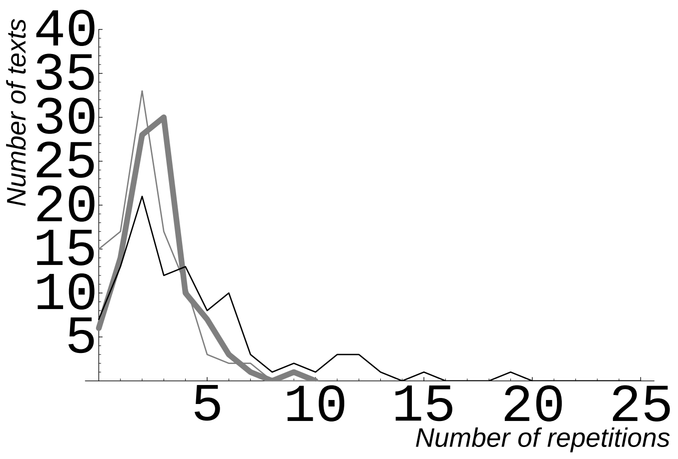


Figure C.2: Distribution of the number of repetitions, key length 17.

Appendix D

Empirical Experiments with Language Statistics

In this appendix we describe the experiments that led to empirical results on the language statistics

MFL = Most Frequent Letter score

BLW = Bigram Log-Weight score

κ = Coincidence Index of two texts

φ = Inner Coincidence Index of a text

χ = KULLBACK's Cross-Product Sum

for English and German (and some also for French).

D.1 Empirical Results on MFL Scores

For English we take a text of 20000 letters, an extract from the Project Gutenberg etext of *Kim*, by Rudyard Kipling, `http://www.gutenberg.org/ebooks/2226`. The partial 20000 letter text is at `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Files/Kim20K.txt`. We divide this text into 2000 substrings of 10 letters each. To this set of substrings we apply the Perl script `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Perl/fritestE.pl`. The results are collected and evaluated in a spreadsheet, found at `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Files/statFriE.xls`.

We do the same for random text, constructed by taking 20000 random numbers between 0 and 25 from `random.org`, see `.../Files/rnd10E.txt`. The Perl script `.../Perl/RandOrg.pl` transforms the random numbers to text.

Figure D.1 shows some characteristics of the distribution. Table D.1 compares the expected and observed distributions. For random texts they match well, taking into account variations caused by drawing a sample. Also for English the observations seem to match the predicted values. The empirical values amount to a power of 68% (instead of 67%) and a predictive value of 75% (75%).

We repeat this procedure for German and French. As texts we take *Schachnovelle* by Stefan Zweig, `http://gutenberg.spiegel.de/buch/7318/1`, and *De la Terre à la Lune* by Jules Verne, `http://www.gutenberg.org/ebooks/799`. The 20000 letter extracts are in `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Files/Schach20K.txt` and `.../Files/Lune20K.txt`. We generate independent random texts, see `.../Files/rnd10D.txt` and `.../Files/rnd10F.txt`. (The random texts being independent, the observed values for random texts differ.) The Perl scripts, adapted to the differing collections of most-frequent letters, are `.../Perl/fritestD.pl` and `.../Perl/fritestF.pl`.

The results are in Figures D.2 and D.3, and Tables D.2 and D.3. The comprehensive evaluation is in the spreadsheets `.../Files/statFriD.xls` and `.../Files/statFriF.xls`.

The empirical values amount to a power of 63% (theory: 67%) and a predictive value of 75% (75%) for German, and a power of 87% (86%) and a predictive value of 88% (87%).

Exercise. Verify the calculations of powers and predictive values.

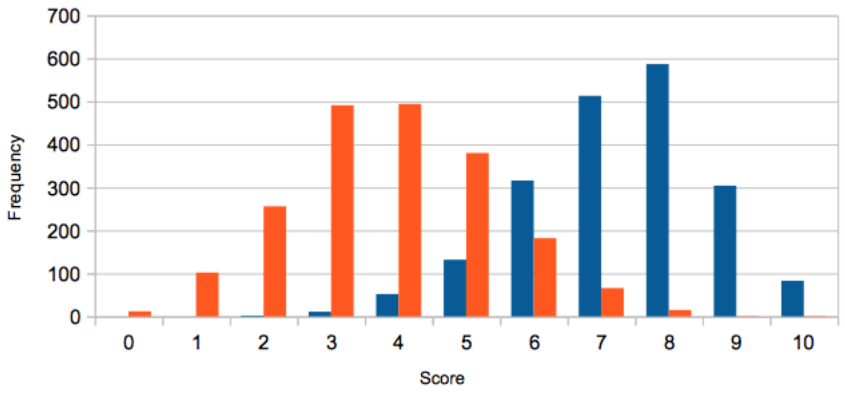


Figure D.1: *MFL scores for 2000 English (blue) and random (red) text chunks of 10 letters each*

Table D.1: *Expected and observed frequencies of MFL scores for 2000 English and 2000 random text chunks of 10 letters*

	Random		**English**	
score	expected	observed	expected	observed
0	16	12	0	0
1	98	102	0	0
2	274	256	2	2
3	456	491	8	11
4	500	494	40	52
5	374	380	134	132
6	194	182	318	316
7	70	66	514	513
8	16	15	546	587
9	2	1	344	304
10	0	1	98	83

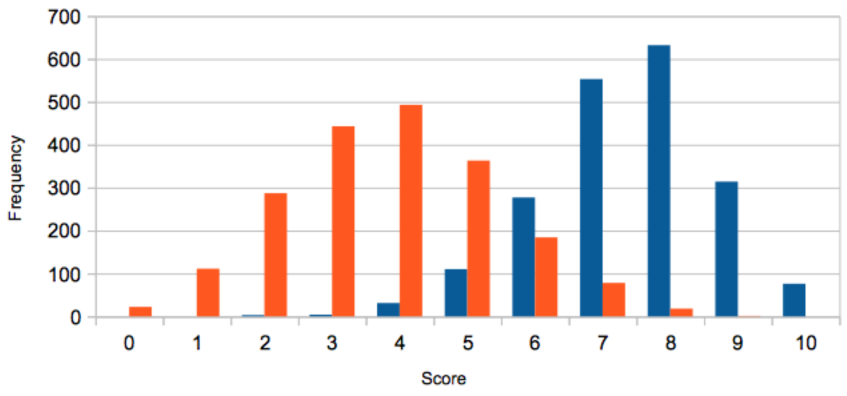


Figure D.2: *MFL scores for 2000 German (blue) and random (red) text chunks of 10 letters each*

Table D.2: *Expected and observed frequencies of MFL scores for 2000 German and 2000 random text chunks of 10 letters*

	Random		**German**	
score	expected	observed	expected	observed
0	16	22	0	0
1	98	111	0	0
2	274	287	0	3
3	456	443	6	4
4	500	493	32	31
5	374	363	116	110
6	194	184	290	277
7	70	78	500	553
8	16	18	564	632
9	2	1	378	314
10	0	0	114	76

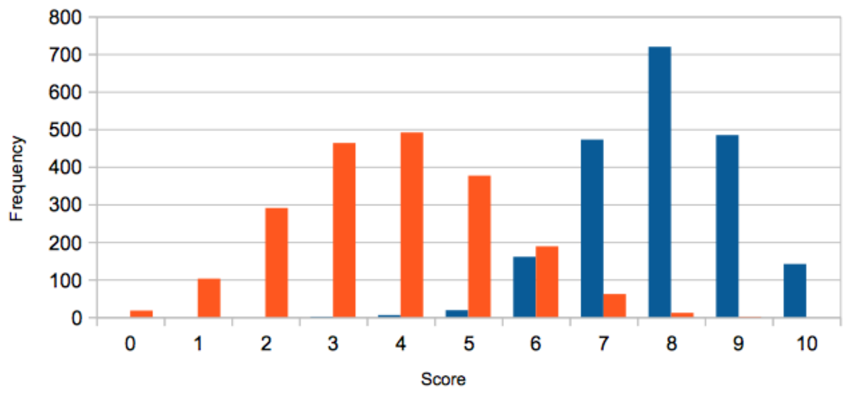


Figure D.3: *MFL scores for 2000 French (blue) and random (red) text chunks of 10 letters each*

Table D.3: *Expected and observed frequencies of MFL scores for 2000 French and 2000 random text chunks of 10 letters*

	Random		**French**	
score	expected	observed	expected	observed
0	16	17	0	0
1	98	102	0	0
2	274	290	0	0
3	456	463	2	1
4	500	491	14	5
5	374	376	62	18
6	194	188	196	160
7	70	61	424	472
8	16	11	602	719
9	2	1	506	484
10	0	0	192	141

D.2 Empirical Results on BLW Scores

We extract 20000 letters from each of the texts *Kim*, *Schachnovelle*, and *De la Terre à la Lune*, and decompose them into 2000 chunks à 10 letters, see the files `eng10a.txt`, `ger10a.txt`, and `fra10a.txt` in the directory `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Files/`. Likewise we generate random texts, see `rnd10Ea.txt`, `rnd10Da.txt`, and `rnd10Fa.txt`. We collect the results in the spreadsheets `ER10res.xls`, `DR10res.xls`, and `FR10res.xls`.

The results are summarized in Tables D.4, D.5, D.6, and Figures D.4, D.5, D.6

The empirical results for the 5%-level of the error of the first kind are as follows.

English. We take the threshold value $T = 11$ for English texts. Then 86 of 2000 English scores are $\leq T$, the error of the first kind is $\alpha = 86/2000 = 4.2\%$. For random texts 1964 of 2000 scores are $\leq T$, the power is $1964/2000 = 99.5\%$. There are 36 random scores and 1914 English scores $> T$, the predictive value for English is $1914/1950 = 98.2\%$.

German. We take the threshold value $T = 12$ for German texts. Then 84 of 2000 German scores are $\leq T$, the error of the first kind is $\alpha = 84/2000 = 4.2\%$. For random texts 1991 of 2000 scores are $\leq T$, the power is $1991/2000 = 99.6\%$. There are 9 random scores and 1916 German scores $> T$, the predictive value for German is $1916/1925 = 99.5\%$.

French. We take the threshold value $T = 11$ for French texts. Then 58 of 2000 French scores are $\leq T$, the error of the first kind is $\alpha = 58/2000 = 2.9\%$. For random texts 1967 of 2000 scores are $\leq T$, the power is $1967/2000 = 98.3\%$. There are 33 random scores and 1942 French scores $> T$, the predictive value for French is $1942/1975 = 98.3\%$.

The BLW score is significantly stronger than the MFL score.

Table D.4: *Frequencies of BLW scores for English vs. random 10 letter texts*

Score	Random	English
$0 \leq x \leq 1$	32	0
$1 < x \leq 2$	97	0
$2 < x \leq 3$	187	0
$3 < x \leq 4$	254	0
$4 < x \leq 5$	324	3
$5 < x \leq 6$	301	1
$6 < x \leq 7$	271	4
$7 < x \leq 8$	216	1
$8 < x \leq 9$	156	8
$9 < x \leq 10$	77	18
$10 < x \leq 11$	49	51
$11 < x \leq 12$	25	120
$12 < x \leq 13$	6	196
$13 < x \leq 14$	3	322
$14 < x \leq 15$	2	413
$15 < x \leq 16$	0	406
$16 < x \leq 17$	0	255
$17 < x \leq 18$	0	157
$18 < x \leq 19$	0	40
$19 < x < \infty$	0	5

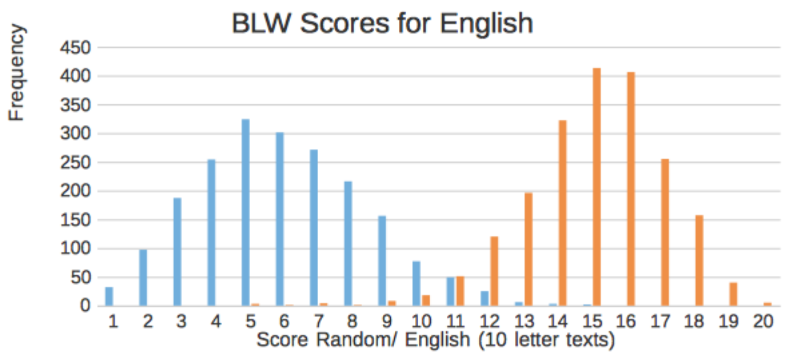


Figure D.4: *BLW scores for 2000 English (red) and random (blue) text chunks of 10 letters each*

Table D.5: *Frequencies of BLW scores for German vs. random texts*

Score	Random	German
$0 \leq x \leq 1$	38	0
$1 < x \leq 2$	105	0
$2 < x \leq 3$	207	0
$3 < x \leq 4$	269	0
$4 < x \leq 5$	296	0
$5 < x \leq 6$	319	0
$6 < x \leq 7$	256	0
$7 < x \leq 8$	185	1
$8 < x \leq 9$	143	2
$9 < x \leq 10$	96	15
$10 < x \leq 11$	47	21
$11 < x \leq 12$	30	45
$12 < x \leq 13$	4	95
$13 < x \leq 14$	4	202
$14 < x \leq 15$	1	332
$15 < x \leq 16$	0	411
$16 < x \leq 17$	0	396
$17 < x \leq 18$	0	298
$18 < x \leq 19$	0	134
$19 < x \leq 20$	0	41
$20 < x < \infty$	0	7

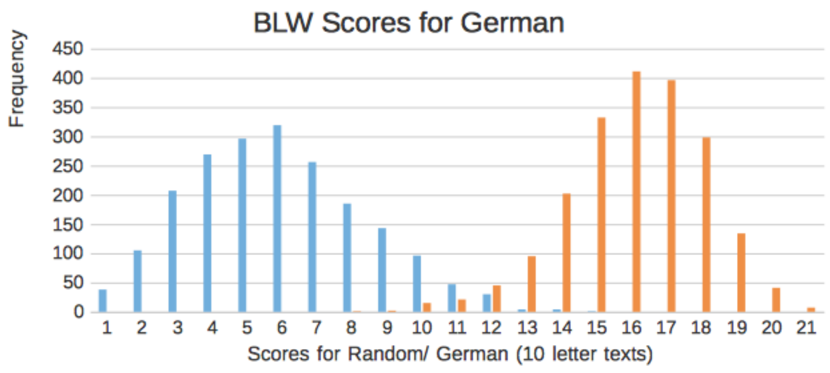


Figure D.5: *BLW scores for 2000 German (red) and random (blue) text chunks of 10 letters each*

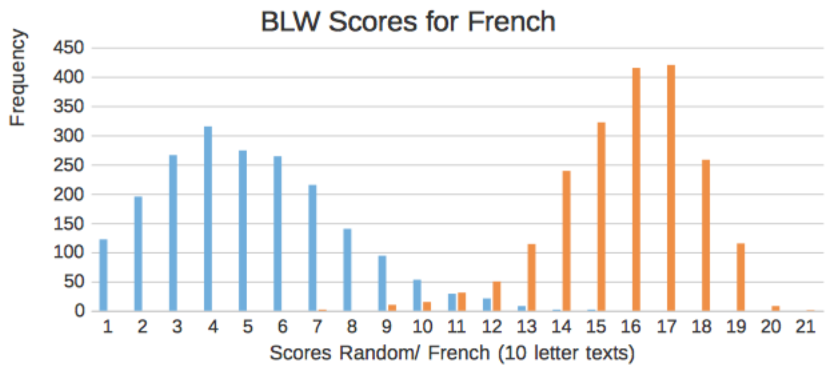


Figure D.6: *BLW scores for 2000 French (red) and random (blue) text chunks of 10 letters each*

Table D.6: *Frequencies of BLW scores for French vs. random texts*

Score	Random	French
$0 \le x \le 1$	122	0
$1 < x \le 2$	195	0
$2 < x \le 3$	266	0
$3 < x \le 4$	315	0
$4 < x \le 5$	274	0
$5 < x \le 6$	264	0
$6 < x \le 7$	215	2
$7 < x \le 8$	140	0
$8 < x \le 9$	94	10
$9 < x \le 10$	53	15
$10 < x \le 11$	29	31
$11 < x \le 12$	21	50
$12 < x \le 13$	8	114
$13 < x \le 14$	2	239
$14 < x \le 15$	2	322
$15 < x \le 16$	0	415
$16 < x \le 17$	0	420
$17 < x \le 18$	0	258
$18 < x \le 19$	0	115
$19 < x \le 20$	0	8
$20 < x < \infty$	0	1

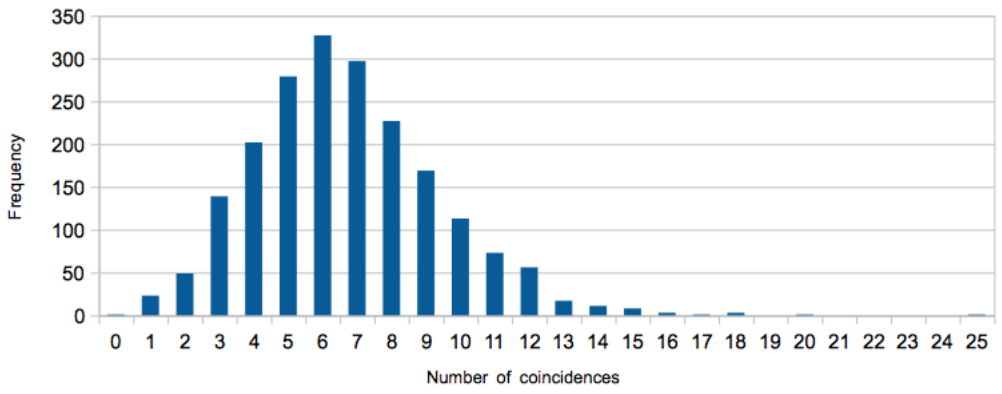


Figure D.7: *Frequency of coincidence counts for 2000 English text pairs of 100 letters—to get coincidence indices divide x-values by* 100

D.3 Empirical Values of the Coincidence Index

The Kappa Distribution for English Texts

We want to learn more about the distribution of coincidence indices $\kappa(a,b)$ for English texts (or text chunks) a and b. To this end we take a large English text—in this case the book *The Poisoned Pen* by Arthur B. Reeve (that by the way contains a cryptogram) from Project Gutenberg—and chop it into chunks $a, b, c, d, \ldots$ of r letters each. Then we count $\kappa(a,b)$, $\kappa(c,d)$, $\ldots$ and list the values in the first column of a spreadsheet for easy evaluation. See the Perl program `kapstat.pl` in `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Perl/` and the spreadsheet `EnglKap.xls` in `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Files/`

> In fact we also record the pure incidence counts as integers. This makes it easier drawing a histogram without generating discretization artefacts.

The text has 449163 letters. Taking $r = 100$ we get 2245 text pairs. We take the first 2000 of them. Table D.7 and Figure D.7 show some characteristics of the distribution.

The Kappa Distribution for German Texts

We repeat this procedure for German texts, using *Scepter und Hammer* by Karl May from the web page of the Karl-May-Gesellschaft. We take the first 2000 text pairs. The results are in Table D.8 and Figure D.8.

Table D.7: *Distribution of κ for 2000 English text pairs of 100 letters*

Minimum:	0.00		
Median:	0.06	Mean value:	0.0669
Maximum:	0.25	Standard dev:	0.0272
1st quartile:	0.05	5% quantile:	0.0300
3rd quartile:	0.08	95% quantile:	0.1200

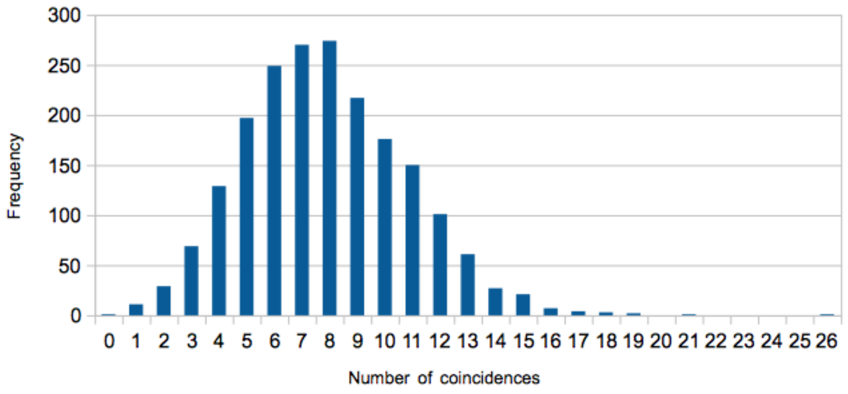


Figure D.8: *Frequency of coincidence counts for 2000 German text pairs of 100 letters—to get coincidence indices divide x-values by* 100

Table D.8: *Distribution of κ for 2000 German text pairs of 100 letters*

Minimum:	0.00		
Median:	0.08	Mean value:	0.0787
Maximum:	0.26	Standard dev:	0.0297
1st quartile:	0.06	5% quantile:	0.0300
3rd quartile:	0.10	95% quantile:	0.1300

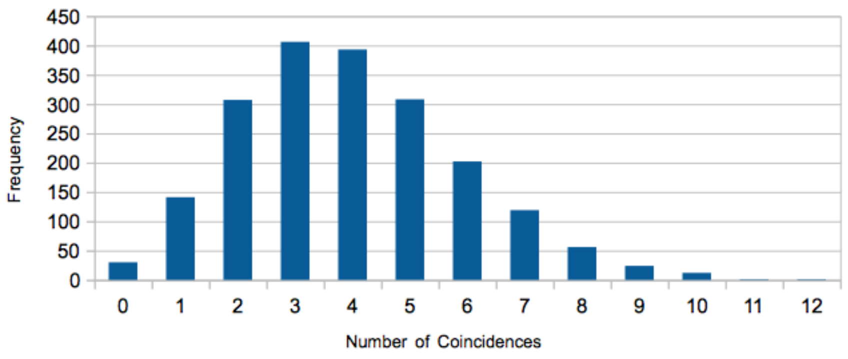


Figure D.9: *Frequency of coincidence counts for 2000 random text pairs of 100 letters—to get coincidence indices divide x-values by* 100

Table D.9: *Distribution of κ for 2000 random text pairs of 100 letters*

Minimum:	0.00		
Median:	0.04	Mean value:	0.040
Maximum:	0.12	Standard dev:	0.020
1st quartile:	0.03	5% quantile:	0.010
3rd quartile:	0.05	95% quantile:	0.070

The Kappa Distribution for Random Texts

Finally the same procedure for random texts. To this end we generate a 400000 character text by the built-in (pseudo-) random generator of Perl. Since the simulation might depend on the quality of the random generator we enhance the random text in the following way: We generate 8132 random letters by the cryptographically strong BBS-generator and use them as key for a BELASO encryption of our random text, repeating the key several times. In spite of this periodicity we may assume that the result gives a 400000 character random text of good quality. This provides us with 2000 text pairs of length 100. The results are in Table D.9 and Figure D.9. Note that the values fit the theoretical values almost perfectly.

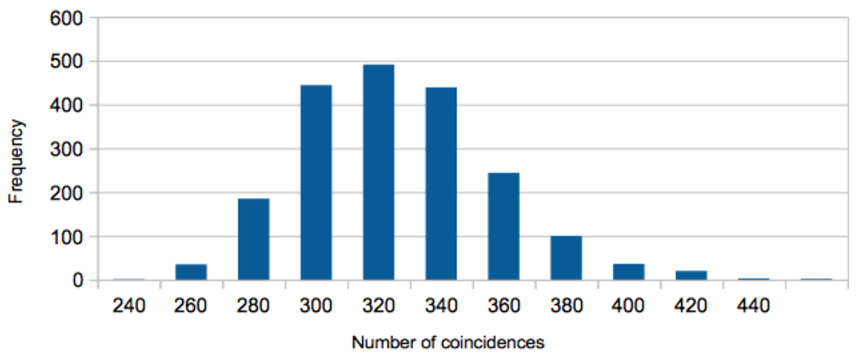


Figure D.10: *Frequency of inner coincidence counts for 2000 English texts of 100 letters—to get φ values divide x-values by* 4950

Table D.10: *Distribution of φ for 2000 English texts of 100 letters*

Minimum:	0.0481		
Median:	0.0634	Mean value:	0.0639
Maximum:	0.0913	Standard dev:	0.0063
1st quartile:	0.0594	5% quantile:	0.0549
3rd quartile:	0.0677	95% quantile:	0.0750

D.4 The Distribution of the Inner Coincidence Index

The Phi Distribution for English Texts

For empirically determining the distribution of the inner coincidence index $\varphi(a)$ we use the Perl program `phistat.pl` from `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Perl/`. For English texts (or text chunks) a, we again take a large English text—in this case the book *The Fighting Chance* by Robert W. Chambers from Project Gutenberg—and chop it into chunks $a, b, c, d, \ldots$ of r letters each. Then we count $\varphi(a)$, $\varphi(b)$, $\ldots$ and list the values in the first column of a spreadsheet. See the file `EnglPhi.xls` in `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Files/`. The text has 602536 letters. We take the first 262006 of them and consider the first 2000 pieces of 100 letters each. Table D.10 and Figure D.10 show some characteristics of the distribution.

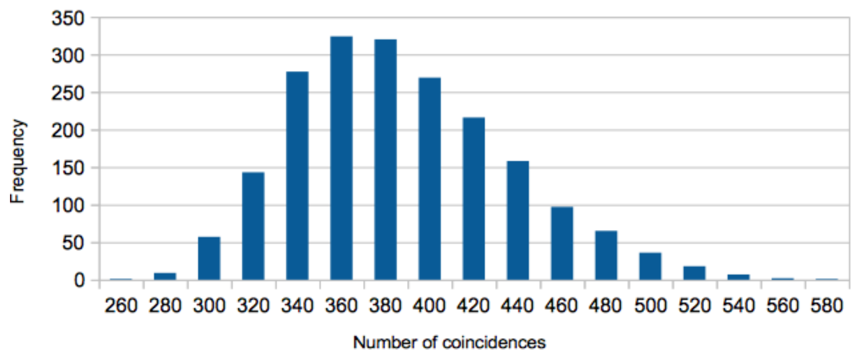


Figure D.11: *Frequency of inner coincidence counts for 2000 German texts of 100 letters—to get φ values divide x-values by* 4950

Table D.11: *Distribution of φ for 2000 German texts of 100 letters*

Minimum:	0.0517		
Median:	0.0752	Mean value:	0.0763
Maximum:	0.1152	Standard dev:	0.0099
1st quartile:	0.0689	5% quantile:	0.0618
3rd quartile:	0.0828	95% quantile:	0.0945

The Phi Distribution for German Texts

We repeat this procedure for German texts, using *Scepter und Hammer* by Karl May. We already consumed its first 400000 letters for κ. Now we take the next 200000 letters—in fact we skip 801 letters in between—and form 2000 text chunks with 100 letters each. The results are in Table D.11 and Figure D.11.

The Phi Distribution for Random Texts

And now the same procedure for random text. The results are in Table D.12 and Figure D.12.

The Phi Distribution for 26 Letter Texts

Since the φ test performs so excellently for 100 letter texts we dare to look at 26 letter texts—a text length that occurs in the Meet-in-the-Middle attack against rotor machines.

Here we give the results as tables only.

The decision threshold on the 5%-level is 0.0585. For English texts the test has a power of only 50%, for German, near 75%. So we have a method

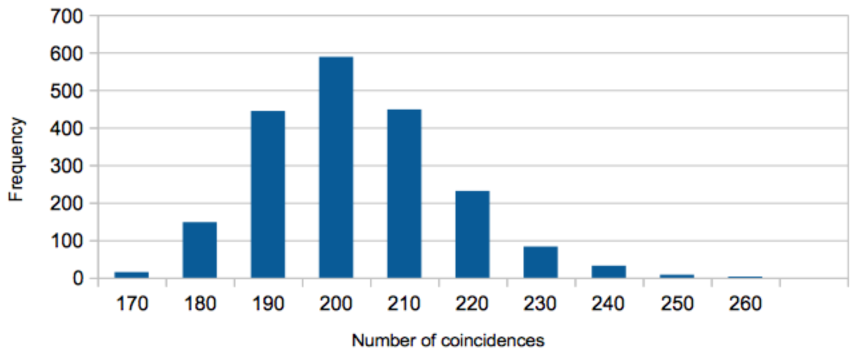


Figure D.12: *Frequency of inner coincidence counts for 2000 random texts of 100 letters—to get φ values divide x-values by* 4950

Table D.12: *Distribution of φ for 2000 random texts of 100 letters*

Minimum:	0.0331		
Median:	0.0398	Mean value:	0.0401
Maximum:	0.0525	Standard dev:	0.0028
1st quartile:	0.0382	5% quantile:	0.0360
3rd quartile:	0.0418	95% quantile:	0.0451

Table D.13: *Distribution of φ for 2000 English texts of 26 letters*

Minimum:	0.0227		
Median:	0.0585	Mean value:	0.0606
Maximum:	0.1385	Standard dev:	0.0154
1st quartile:	0.0492	5% quantile:	0.0400
3rd quartile:	0.0677	95% quantile:	0.0892

Table D.14: *Distribution of φ for 2000 German texts of 26 letters*

Minimum:	0.0308		
Median:	0.0708	Mean value:	0.0725
Maximum:	0.1785	Standard dev:	0.0204
1st quartile:	0.0585	5% quantile:	0.0431
3rd quartile:	0.0831	95% quantile:	0.1108

to recognize monoalphabetic ciphertext that works fairly well for texts as short as 26 letters.

Table D.15: *Distribution of φ for 2000 random texts of 26 letters*

Minimum:	0.0154		
Median:	0.0400	Mean value:	0.0401
Maximum:	0.0954	Standard dev:	0.0112
1st quartile:	0.0338	5% quantile:	0.0246
3rd quartile:	0.0462	95% quantile:	0.0585

D.5 Kullback's Cross-Product Sum Statistic

We collect empirical results for 2000 pairs of 100 letter texts using `chistat.pl`, from `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Perl/`. For English we use the book *Dr Thorndyke Short Story Omnibus* by R. Austin Freeman from Project Gutenberg. We extract a first part of 402347 letters (`Thorn1.txt`) and take the first 400000 of them for our statistic. In the same way for German we use *Die Juweleninsel* by Karl May from Karl-May-Gesellschaft (`Juwelen1.txt`, 434101 letters). For random texts we generate 400000 letters by Perl's random generator (`RndT400K.txt`). (All files in `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Files/`.)

The results are in Tables D.16, D.17, and D.18. We see that χ—in contrast with the coincidence index κ—performs extremely well, in fact in our experiments it even completely separates English and German texts from random texts of length 100. It is a test with power near 100% and error probability near 0%. The χ test even distinguishes between English and German texts at the 5% error level with a power of almost 75%. For this assertion compare the 95% quantile for English with the first quartile for German.

Table D.16: *Distribution of χ for 2000 English text pairs of 100 letters*

Minimum:	0.0500		
Median:	0.0660	Mean value:	0.0663
Maximum:	0.0877	Standard dev:	0.0049
1st quartile:	0.0630	5% quantile:	0.0587
3rd quartile:	0.0693	95% quantile:	0.0745

The results for 100 letter texts encourage us to try 26 letter texts. To this end we need 104000 letters for each language. We extract the next 104009 letters from *Dr Thorndyke Short Story Omnibus* (`Thorn2.txt`), and the next

Table D.17: *Distribution of χ for 2000 German text pairs of 100 letters*

Minimum:	0.0578		
Median:	0.0792	Mean value:	0.0794
Maximum:	0.1149	Standard dev:	0.0074
1st quartile:	0.0742	5% quantile:	0.0677
3rd quartile:	0.0840	95% quantile:	0.0923

Table D.18: *Distribution of χ for 2000 random text pairs of 100 letters*

Minimum:	0.0337		
Median:	0.0400	Mean value:	0.0400
Maximum:	0.0475	Standard dev:	0.0020
1st quartile:	0.0387	5% quantile:	0.0367
3rd quartile:	0.0413	95% quantile:	0.0433

Table D.19: *Distribution of χ for 2000 English text pairs of 26 letters*

Minimum:	0.0266		
Median:	0.0666	Mean value:	0.0666
Maximum:	0.1169	Standard dev:	0.0120
1st quartile:	0.0577	5% quantile:	0.0488
3rd quartile:	0.0740	95% quantile:	0.0873

104293 letters from *Die Juweleninsel* (`Juwelen2.txt`). We construct random text by taking 104000 random numbers between 0 and 25 from `random.org` (`RndT104K.txt`). The results are in Tables D.19, D.20, and D.21. The χ-test is quite strong even for 26 letters: At the 5% error level its power is around 91% for English, 98% for German.

Table D.20: *Distribution of χ for 2000 German text pairs of 26 letters*

Minimum:	0.0325		
Median:	0.0784	Mean value:	0.0793
Maximum:	0.1538	Standard dev:	0.0154
1st quartile:	0.0680	5% quantile:	0.0562
3rd quartile:	0.0888	95% quantile:	0.1065

Table D.21: *Distribution of χ for 2000 random text pairs of 26 letters*

Minimum:	0.0178		
Median:	0.0385	Mean value:	0.0386
Maximum:	0.0680	Standard dev:	0.0075
1st quartile:	0.0340	5% quantile:	0.0266
3rd quartile:	0.0429	95% quantile:	0.0518

Appendix E

The Euclidean Algorithm

E.1 The Algorithm

Euclid's algorithm gives the greatest common divisor (gcd) of two integers,

$$\gcd(a,b) = \max\{d \in \mathbb{Z} \mid d|a, d|b\}$$

If for simplicity we define $\gcd(0,0) = 0$, we have a function

$$\gcd : \mathbb{Z} \times \mathbb{Z} \longrightarrow \mathbb{N}$$

with the following properties:

Lemma 1 *For any $a, b, c, q \in \mathbb{Z}$ we have:*

(i) $\gcd(a,b) = \gcd(b,a)$.

(ii) $\gcd(a,-b) = \gcd(a,b)$.

(iii) $\gcd(a,0) = |a|$.

(iv) $\gcd(a-qb,b) = \gcd(a,b)$.

Proof. Trivial; for (iv) use the equivalence $d|a, b \Longleftrightarrow d|a-qb, b$. $\diamond$

One usually writes Euclid's algorithm as a sequence of divisions with remainder:

$$r_0 = |a|, r_1 = |b|, \ldots, r_{i-1} = q_i r_i + r_{i+1},$$

where q_i is the integer quotient and r_{i+1} is the unique division remainder with $0 \le r_{i+1} < r_i$. As soon as $r_n \neq 0$ and $r_{n+1} = 0$, we have $r_n = \gcd(a,b)$. For from Lemma 1 we get

$$\gcd(a,b) = \gcd(r_0,r_1) = \gcd(r_1,r_2) = \ldots = \gcd(r_n,0) = r_n.$$

Since moreover

$$r_1 > r_2 > \ldots > r_i \geq 0 \quad \text{for all } i,$$

we reach the terminating condition $r_{n+1} = 0$ after at most $n \leq |b|$ iteration steps (i. e. divisions).

A small additional consideration even gives more. Note that each r_i is an integer linear combination of the two preceeding division remainders, hence of $|a|$ and $|b|$:

$$r_{i+1} \in \mathbb{Z}r_i + \mathbb{Z}r_{i-1} \subseteq \ldots \subseteq \mathbb{Z}r_1 + \mathbb{Z}r_0 = \mathbb{Z}a + \mathbb{Z}b;$$

for r_0 and r_1 this is immediate, and in the general case it follows by induction: Let $r_j = |a|x_j + |b|y_j$ for $0 \leq j \leq i$. Then

$$\begin{aligned} r_{i+1} = r_{i-1} - q_i r_i &= |a|x_{i-1} + |b|y_{i-1} - q_i|a|x_i - q_i|b|y_i \\ &= |a|(x_{i-1} - q_i x_i) + |b|(y_{i-1} - q_i y_i). \end{aligned}$$

This consideration even gives an explicit construction for the coefficients; for they sastisfy the recursive formulas

$$x_{i+1} = x_{i-1} - q_i x_i \quad \text{with} \quad x_0 = 1,\ x_1 = 0,$$

$$y_{i+1} = y_{i-1} - q_i y_i \quad \text{with} \quad y_0 = 0,\ y_1 = 1,$$

that agree with the formula for the r_i except for the start values:

$$r_{i+1} = r_{i-1} - q_i r_i \quad \text{with} \quad r_0 = |a|,\ r_1 = |b|.$$

The **extended Euclidean algorithm** (sometimes called algorithm of LAGRANGE) is the synopsis of these three recursive formulas. In summary we have shown (if we properly adjust the signs of x_n and y_n):

Proposition 1 *The extended Euclidean algorithm gives the greatest common divisor d of two integers a and b and integer coefficients x and y with $ax + by = d$ in finitely many steps.*

Bemerkungen

1. The least common multiple is efficiently calculated by the formula

$$\operatorname{lcm}(a, b) = \frac{ab}{\gcd(a, b)}.$$

2. One calculates the greatest common divisor of several inegers by the formula

$$\gcd(\ldots(\gcd(\gcd(a_1, a_2), a_3) \ldots, a_r);$$

this allows for a bit of optimisation. An analogous statement holds for the least common multiple.

E.2 Analysis of Euclid's Algorithm

The algorithm of the last section has a hidden problem: Though the quotients and division remainders are safely bounded by the input parameters, the coefficients x_i and y_i are uncontrolled at first sight. How can we guarantee that we don't get an overflow, if we use the usual integer arithmetic with bounded precision? Now, the following reasoning controls the growth:

Lemma 2 *For the coefficients x_i and y_i in the extended Euclidean algorithm we have:*

(i) *$x_i > 0$, if i is even, $x_i \leq 0$, if i is odd, and $|x_{i+1}| \geq |x_i|$ for $i = 1, \ldots, n$.*

(ii) *$y_i \leq 0$, if i is even, $y_i > 0$, if i is odd, and $|y_{i+1}| \geq |y_i|$ for $i = 2, \ldots, n$.*

(iii) *$x_{i+1}y_i - x_iy_{i+1} = (-1)^{i+1}$ for $i = 0, \ldots, n$; in particular the x_i and y_i are always coprime for $i = 0, \ldots, n+1$.*

(iv) *$|x_i| \leq |b|, |y_i| \leq |a|$ for $i = 0, \ldots, n+1$, if $b \neq 0$ resp. $a \neq 0$.*

Proof. (Sketch.) Show (i), (ii), and (iii) by induction. From $0 = r_{n+1} = |a|x_{n+1} + |b|y_{n+1}$ then follows $x_{n+1}|b$ and $y_{n+1}|a$. $\diamond$

The Euclidean algorithm is very efficient—the number of iteration steps grows only linearly with the *number of digits* of the input parameters, the entire execution time only quadratically. In the following we perform a quite exact analysis. Without loss of generality we may assume $b \neq 0$.

Given the length n of the division chain—how large must b be? We have $r_n \geq 1, r_{n-1} \geq 2$, and $r_{i-1} \geq r_i + r_{i+1}$. The Fibonacci numbers F_n are recursively defined by

$$F_0 = 0,\ F_1 = 1,\ F_n = F_{n-1} + F_{n-2} \quad \text{for } n \geq 2.$$

Hence by induction we get $r_i \geq F_{n+2-i}$, where the induction starts with $r_n \geq 1 = F_2$, $r_{n-1} \geq 2 = F_3$; in particular we get $|b| \geq F_{n+1}$. In other words:

Proposition 2 (Binet 1841) *For $a, b \in \mathbb{Z}$ with $0 < b < F_{n+1}$ the Eucliden algorithm finds the greatest common divisor in at most $n-1$ iteration steps.*

Addendum. *This is true also for $b = F_{n+1}$, except if $a \equiv F_{n+2} \equiv F_n \pmod{b}$.*

This gives a quite elegant mathematical formulation, but not yet an explicit bound. However the growth of the Fibonacci numbers is well-known. One can express it by the golden section $\varphi = \frac{1+\sqrt{5}}{2}$, that is defined by $\varphi^2 - \varphi - 1 = 0$.

Lemma 3 *For a real number $c \in \mathbb{R}$ and an index $k \in \mathbb{N}$ let $F_k > c \cdot \varphi^k$ and $F_{k+1} > c \cdot \varphi^{k+1}$. Then $F_n > c \cdot \varphi^n$ for all $n \geq k$.*

Proof. (By induction.)

$$F_n = F_{n-1} + F_{n-2} > c\varphi^{n-1} + c\varphi^{n-2} = c\varphi^{n-2}(\varphi + 1) = c\varphi^n$$

for $n \geq k + 2$. $\diamond$

Corollary 1 *$F_{n+1} > 0.43769 \cdot \varphi^{n+1}$ for $n \geq 2$.*

Proof.

$$\begin{aligned} \varphi^2 &= \varphi + 1 = \frac{3 + \sqrt{5}}{2}, \\ \varphi^3 &= \varphi^2 + \varphi = 2 + \sqrt{5}, \\ \varphi^4 &= \varphi^3 + \varphi^2 = \frac{7 + 3\sqrt{5}}{2}. \end{aligned}$$

Therefore

$$\begin{aligned} \frac{F_3}{\varphi^3} &= \frac{2}{2 + \sqrt{5}} = \frac{2(\sqrt{5} - 2)}{1} = 2\sqrt{5} - 4 > 0.47, \\ \frac{F_4}{\varphi^4} &= \frac{3 \cdot 2}{7 + 3\sqrt{5}} = \frac{6(7 - 3\sqrt{5})}{49 - 45} = \frac{21 - 9\sqrt{5}}{2} > 0.43769 \end{aligned}$$

which proves the assertion. $\diamond$

Corollary 2 *Let $a, b \in \mathbb{Z}$ with $b \geq 2$. Then the number of iteration steps in the Euclidean algorithm for $\gcd(a, b)$ is less then $0.718 + 4.785 \cdot \log_{10}(b)$.*

Proof. If the division chain has length n, then $b \geq F_{n+1}$,

$$b \geq F_{n+1} > 0.43769 \cdot \varphi^{n+1},$$

$\log_{10}(b) > \log_{10}(0.43769) + (n+1) \cdot \log_{10}(\varphi) > -0.35884 + 0.20898 \cdot (n+1)$,

hence $n < 0.718 + 4.785 \cdot \log_{10}(b)$. $\diamond$

Somewhat coarser, but simply to remember, is the following version:

Corollary 3 *Let $a, b \in \mathbb{Z}$ with $b \geq 2$. Then the number of iteration steps in the Euclidean algorithm for $\gcd(a, b)$ is less then five times the number of digits of b except for $b = 8, a \equiv 5 \pmod 8$, where 5 iteration steps are needed.*

If we additionally consider the costs for the multiplication and division of large numbers depending on their number of digits, we get a working time that grows quadratically with the number of digits as shown in the following.

If a has m digits (with respect to a base B of the integers), and b has p digits, then the expense for the first division alone is already $\leq c \cdot (m-p) \cdot p$; here c is a constant that is at most twice as large as the constant that bounds the expense for "multiplying quotient $\times$ divisor back". Considering actual computer architectures we would take $B = 2^{32}$ or 2^{64}, and count the basic operations addition, subtraction, multiplication, division with remainder, and comparison of 1-digit numbers (in base B) as primitive steps. Fortunately the involved numbers shrink in an exponential way along the Euclidean division chain. The division step

$$r_{i-1} = q_i r_i + r_{i+1}$$

yet requires $\leq c \cdot \log_B(q_i) \log_B(r_i)$ primitive operations, hence the entire division chain needs

$$\begin{aligned} A(a,b) &\leq c \cdot \sum_{i=1}^{n} \log_B(q_i) \log_B(r_i) \leq c \cdot \log_B |b| \cdot \sum_{i=1}^{n} \log_B(q_i) \\ &= c \cdot \log_B |b| \cdot \log_B(q_1 \cdots q_n). \end{aligned}$$

We further estimate the product of the q_i:

$$|a| = r_0 = q_1 r_1 + r_2 = q_1(q_2 r_2 + r_3) + r_2 = \ldots = q_1 \cdots q_n r_n + \cdots \geq q_1 \cdots q_n$$

and get the coarse bound

$$A(a,b) \leq c \cdot \log_B |b| \cdot \log_B |a| \,.$$

Proposition 3 *The number of primitive operations in the Euclidean algorithm for two integers a and b with $\leq m$ digits is $\leq c \cdot m^2$.*

Note that c is a known small constant.

So the expense for the Euclidean algorithm with input a and b is not significantly larger then the expense for multiplying a and b. We won't discuss sharper estimates or potential enhancements of this bound. But note that an algorithm by LEHMER allows replacing a great amount of divisions of large numbers in the division chain by primitive operations.

E.3 Congruence Division

The extended Euclidean algorithm also provides a solution of the—not entirely trivial—problem of efficient division in the ring $\mathbb{Z}/n\mathbb{Z}$ of integers mod n.

Proposition 4 *Let $n \in \mathbb{N}$, $n \geq 2$, and $a, b \in \mathbb{Z}$ with $\gcd(b, n) = d$. Then a is divisible by b in $\mathbb{Z}/n\mathbb{Z}$, if and only if $d|a$. In this case there are exactly d solutions z of $zb \equiv a \pmod{n}$ with $0 \leq z < n$, and any two of them differ by a multiple of n/d. If $d = xn + yb$ and $a = td$, then $z = yt$ is a solution.*

Proof. If b divides a, then $a \equiv bz \pmod{n}$, so $a = bz + kn$, hence $d|a$. For the converse let $a = td$. By Proposition 1 we find x, y with $nx + by = d$; hence $nxt + byt = a$ and $byt \equiv a \pmod{n}$. If also $a \equiv bw \pmod{n}$, then $b(z - w) \equiv 0 \pmod{n}$, hence $z - w$ a multiple of n/d. $\diamond$

Proposition 4 contains an explicit algorithm for the division. An important special case is $d = 1$ with a notably simple formulation:

Corollary 1 *If b and n are coprime, then each a in $\mathbb{Z}/n\mathbb{Z}$ is divisible by b in a unique way.*

Since $d = 1$ the calculation of the inverse y of b follows immediately from the formula $1 = nx + by$; for $by \equiv 1 \pmod{n}$.

Corollary 2 $(\mathbb{Z}/n\mathbb{Z})^\times = \{b \bmod n \mid \gcd(b, n) = 1\}$.

Therefore the invertible elements of the ring $\mathbb{Z}/n\mathbb{Z}$ are exactly the equivalence classes of the integers coprime with n. The most important case is: $n = p$ prime:

Corollary 3 $\mathbb{F}_p := \mathbb{Z}/p\mathbb{Z}$ *is a field.*

Proof. For $b \in \mathbb{F}_p, b \neq 0$ there is exactly one $c \in \mathbb{F}_p$ with $bc = 1$. $\diamond$

Corollary 4 (Fermat's Little Theorem) $a^p \equiv a \pmod{p}$ *for all $a \in \mathbb{Z}$.*

Proof. The elements $\neq 0$ of $\mathbb{F}_p$ form the multiplicative group $\mathbb{F}_p^\times$. Because the order of an element always divides the group order, we have $a^{p-1} \equiv 1 \pmod{p}$ for a coprime with p. Otherwise we have $p|a$, hence $a \equiv 0 \equiv a^p \pmod{p}$. $\diamond$

E.4 The Chinese Remainder Algorithm

The Chinese remainder problem asks for the solution of simultaneous congruences. The simplest case worth of mention is:

Proposition 5 (Chinese Remainder Theorem) *Let m and n coprime natural numbers ≥ 1, and a, b arbitrary integers. Then there is exactly one integer x, $0 \leq x < mn$, such that*

$$x \equiv a \pmod{m}, \; x \equiv b \pmod{n}.$$

Proof. Let us first show the uniqueness: If y is another solution, then $y = x + km = x + ln$ with integers k und l, and $km = ln$. Since m and n are coprime we conclude $n|k$, $k = cn$,

$$y = x + cmn \equiv x \pmod{mn}.$$

For the existence proof we try $x = a + tm$; then necessarily $x \equiv a \pmod{m}$ and

$$x \equiv b \pmod{n} \Longleftrightarrow b - a \equiv x - a \equiv tm \pmod{n}.$$

Such a t exists by Proposition 4. Reduce this solution $x \bmod (mn)$. $\diamond$

The proof was constructive and easily leads to an algorithm. In the general case, for multiple congruences, the Chinese remainder problem looks like follows:

- Given q pairwise coprime integers $n_1, \ldots, n_q \geq 1$ and q integers $a_1, \ldots, a_q$,
- find an integer x such that $x \equiv a_i \pmod{n_i}$ for $i = 1, \ldots q$.

One approach is suitably adapting Proposition 5. More interesting is an abstract formulation that also comprises interpolation of polynomials; also in this more general formulation we recognise Proposition 5 together with its proof, if we bear in mind that for integers m and n with greatest common divisor d we have the equivalences:

$$m, n \text{ coprime} \Longleftrightarrow d = 1 \Longleftrightarrow \mathbb{Z}m + \mathbb{Z}n = \mathbb{Z}.$$

Proposition 6 (General Chinese Remainder Theorem) *Let R be a commutative ring with 1, $q \geq 1$, $\mathfrak{a}_1, \ldots, \mathfrak{a}_q \trianglelefteq R$ ideals with $\mathfrak{a}_i + \mathfrak{a}_j = R$ for $i \neq j$. Let $a_1, \ldots, a_q \in R$ be given. Then there exists an $x \in R$ with $x - a_i \in \mathfrak{a}_i$ for $i = 1, \ldots, q$, and the equivalence class $x \bmod \mathfrak{a}_1 \cap \cdots \cap \mathfrak{a}_q$ is uniquely determined.*

Proof. As before the uniqueness is quite simple: If $x - a_i, y - a_i \in \mathfrak{a}_i$, then $x - y \in \mathfrak{a}_i$; if this is true for all i, then $x - y \in \mathfrak{a}_1 \cap \cdots \cap \mathfrak{a}_q$.

We prove the existence by induction on q. In the case $q = 1$ we simply take $x = a_1$. Now let $q \geq 2$, and assume y with $y - a_i \in \mathfrak{a}_i$ for $i = 1, \ldots, q-1$ is already found. Idea: We can add to y an $s \in \mathfrak{a}_1 \cap \cdots \cap \mathfrak{a}_{q-1}$ without giving up what we already have, the solution of the first $q-1$ congruences. We need the statement: For each $r \in R$ there is an $s \in \mathfrak{a}_1 \cap \cdots \cap \mathfrak{a}_{q-1}$ with $r - s \in \mathfrak{a}_q$, or in other words,

$$(\mathfrak{a}_1 \cap \cdots \cap \mathfrak{a}_{q-1}) + \mathfrak{a}_q = R.$$

To prove this intermediate assertion we choose $c_i \in \mathfrak{a}_i$ for $i = 1, \ldots, q-1$ and $b_1, \ldots, b_{q-1} \in \mathfrak{a}_q$ with $b_i + c_i = 1$. Then

$$1 = (b_1 + c_1) \cdots (b_{q-1} + c_{q-1}) = c_1 \cdots c_{q-1} + b$$

with $c_1 \cdots c_{q-1} \in \mathfrak{a}_1 \cap \cdots \cap \mathfrak{a}_{q-1}$ and $b \in \mathfrak{a}_q$.

Now for $a_q - y \in R$ choose an $s \in \mathfrak{a}_1 \cap \cdots \cap \mathfrak{a}_{q-1}$ with $a_q - y - s \in \mathfrak{a}_q$, and set $x = y + s$. Then $x \equiv y \equiv a_i \pmod{\mathfrak{a}_i}$ for $i = 1, \ldots, q-1$, and $x \equiv y + s \equiv a_q \pmod{\mathfrak{a}_q}$. $\diamond$

Remarks and Examples

1. For $R = \mathbb{Z}$ or any principal ideal domain, and $\mathfrak{a}_i = Rn_i$ we have $\mathfrak{a}_1 \cap \cdots \cap \mathfrak{a}_q = R(n_1 \cdots n_q)$. From this we get the usual formulation of the Chinese Remainder Theorem.

2. If R is a principal ideal domain, then the construction of the solution proceeds as follows: If $\mathfrak{a}_i = Rn_i$, then choose s in the intermediate assertion such that $s = tn_1 \cdots n_{q-1}$ with

$$r - tn_1 \cdots n_{q-1} \in Rn_q$$

   (congruence division mod n_q). Therefore an explicit algorithm for the Chinese remainder problem exists in R, if one exists for the congruence division, in any case for $R = \mathbb{Z}$.

3. In the case $R = \mathbb{Z}$ we iteratively calculate

$$\begin{aligned} x_1 &= a_1 \bmod n_1, & s_1 &= n_1, \\ t_i \text{ with } 0 \le t_i &\le n_i - 1 \quad \text{and} & a_i - x_{i-1} - t_i s_{i-1} &\in Rn_i, \\ x_i &= x_{i-1} + t_i s_{i-1}, & s_i &= s_{i-1} n_i. \end{aligned}$$

   In particular $s_k = n_1 \cdots n_k$. By induction one immediately proves $0 \le x_i \le s_i - 1$ for all i. Finally one gets the solution $x = x_q$. This consideration guarantees that none of the intermediate results causes an overflow. The expense essentially consists of $q - 1$ congruence divisions and $2 \cdot (q-1)$ ordinary integer multiplications. Therefore the total expense is of order $cq\times$ (the expense for a multiplication of long integers) with a small constant c.

4. The general look of the solution formula is

$$x = x_1 + t_1 n_1 + \cdots + t_{q-1} n_1 \cdots n_{q-1}.$$

5. As an example we treat SUN-TSU's problem from the 1st Century. In our notation its formulation is: Find x such that
$$x \equiv 2 \pmod 3, \quad x \equiv 3 \pmod 5, \quad x \equiv 2 \pmod 7.$$
Our algorithm gives step by step:
$$\begin{aligned} x_1 = 2, &\quad s_1 = 3, \\ 1 - 3t_2 \in 5\mathbb{Z}, &\quad t_2 = 2, \\ x_2 = 2 + 2 \cdot 3 = 8, &\quad s_2 = 15, \\ -6 - 15t_3 \in 7\mathbb{Z}, &\quad t_3 = 1, \\ x = x_3 = 8 + 1 \cdot 15 = 23. & \end{aligned}$$

6. For the polynomial ring $K[T]$ over a field K the interpolation problem is a special case of the Chinese remainder problem. Our algorithm in this case is just NEWTON's interpolation procedure.

E.5 EULER's Phi Function

An important application of the Chinese Remainder Theorem follows; we assume $n \geq 2$. The integers mod n form the ring $\mathbb{Z}/n\mathbb{Z}$. The *multiplicative group* mod n consists of the invertible elements of this ring, and is compactly denoted by
$$\mathbb{M}_n := (\mathbb{Z}/n\mathbb{Z})^\times.$$
Its order is given by the EULER φ function:
$$\varphi(n) = \#\mathbb{M}_n = \#\{a \in [0 \cdots n-1] \mid a \text{ coprime with } n\}.$$

Corollary 1 *For m and n coprime, $\varphi(mn) = \varphi(m)\varphi(n)$.*

Proof. The Chinese Remainder Theorem just says that the natural ring homomorphism
$$F\colon \mathbb{Z}/mn\mathbb{Z} \longrightarrow \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}, \quad x \mapsto (x \bmod m, x \bmod n),$$
is bijective, hence even a ring isomorphism. Moreover $F(\mathbb{M}_{mn}) = \mathbb{M}_m \times \mathbb{M}_n$. Therefore
$$\varphi(mn) = \#\mathbb{M}_{mn} = \#\mathbb{M}_m \cdot \#\mathbb{M}_n = \varphi(m)\varphi(n),$$
as was to be shown. $\diamond$

If p is prime, then $\varphi(p) = p - 1$. More generally $\varphi(p^e) = p^e - p^{e-1} = p^e(1 - \frac{1}{p})$, if $e \geq 1$, because p^e exactly has the divisors px with $1 \leq x \leq p^{e-1}$. From Corollary 1 we conclude:

Corollary 2 *Let $n = p_1^{e_1} \cdots p_r^{e_r}$ be the prime decomposition (all $e_i \geq 1$). Then*

$$\varphi(n) = n \cdot \prod_{i=1}^{r} (1 - \frac{1}{p_i}).$$

Bibliography

[1] F. L. Bauer, *Decrypted Secrets; Methods and Maxims of Cryptology.* Springer, Berlin 1997.

[2] E. Cesàro, *Corso di analisi algebrica con introduzione al calcolo infinitesimale.* Bocca, Torino 1894.

[3] C. A. Deavours, Unicity points in cryptanalysis. Cryptologia 1 (1977), 469–684.

[4] C. A. Deavours, L. Kruh, *Machine Cryptography and Modern Cryptanalysis.* Artech House, Norwood 1985.

[5] W. Feller, *An Introduction to Probability Theory and Its Applications.* Volume I. Wiley, New York 1957.

[6] A. Fisher, *Mathematical Theory of Probabilities.* Macmillan, New York 1915.

[7] W. F. Friedman, *The Riverbank Publications Volume 1* (contains Publications No. 15, 16, 17, and 18). Aegean Park Press, Laguna Hills 1979.

[8] R. Ganesan, A. T. Sherman, *Statistical Techniques for Language Recognition: An Introduction and Guide for Cryptanalysts.* Cryptologia 17 (1993), 321–366.

[9] R. Ganesan, A. T. Sherman, *Statistical Techniques for Language Recognition: An Empirical Study Using Real and Simulated English.* Cryptologia 18 (1994), 289–331.

[10] A. M. Gleason, *Elementary Course in Probability for the Cryptanalyst.* Aegean Park Press, Laguna Hills 1985.

[11] M. E. Hellman, An extension of the Shannon theory approach to cryptography. IEEE Trans Information Theory 23 (1977), 289–294.

[12] A. M. Jaglom, I. M, Jaglom, *Wahrscheinlichkeit und Information.* VEB Deutscher Verlag der Wissenschaften, Berlin 1967.

[13] H. Jürgensen, Language redundancy and the unicity point. Cryptologia 7 (1983), 37–48.

[14] H. Jürgensen, D. E. Matthews, Some results on the information theoretic analysis of cryptosystems. CRYPTO 83, 303–356.

[15] D. Kahn, *The Codebreakers.* Macmillan, New York 1967.

[16] S. Kullback, *Statistical Methods in Cryptanalysis.* Aegean Park Press, Laguna Hills 1976.

[17] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone, *Handbook of Applied Cryptography.* CRC Press, Boca Raton 1997.

[18] J. Reeds, Entropy calculations and particular methods of cryptanalysis. Cryptologia 1 (1977), 235–254.

[19] H. Robbins, A remark on Stirling's formula. Amer. Math. Monthly 62 (1955), 26–29.

[20] B. Schneier, *Applied Cryptography.* John Wiley, New York 1996.

[21] C. E. Shannon, A mathematical theory of communication. Bell System Technical Journal 27 (1948), 379–423, 623–656.

[22] C. E. Shannon, Communication theory of secrecy systems. Bell System Technical Journal 28 (1949), 656–715.

[23] C. E. Shannon, The entropy of printed english. Bell System Technical Journal 30 (1941), 50–64.

[24] S. Singh, *The Code Book.* Fourth Estate, London 1999.

[25] A. Sinkov, *Elementary Cryptanalysis.* The Mathematical Association of America, Washington, 1966.

[26] D. R. Stinson, *Cryptography – Theory and Practice.* CRC Press, Boca Raton 1995.

[27] J. Stirling, *Methodus Differentialis: sive Tractatus de Summatione et Interpolatione Serierum Infinitarum.* G. Strahan, Londini (London) 1730.