Computer Science • 15 (4) 2014 http://dx.doi.org/10.7494/csci.2014.15.4.365

Anna Borowska
Elżbieta Rzeszutko

THE CRYPTANALYSIS OF THE ENIGMA CIPHER. THE PLUGBOARD AND THE CRYPTOLOGIC BOMB

Abstract *We study the problem of decoding secret messages encrypted by the German Army with the M3 Enigma machine after September 15, 1938. We focused our attention on the algorithmization and programming of this problem. A completion and optimization of Zygalski's sheets method were presented previously. We describe below the missing algorithm solving the problem of the plugboard settings with an algebraic justification. This method is the original idea of the authors, and we can use it for cryptanalysis together with both Zygalski's sheets method and Rejewski's bomb method. Next, we present a reconstruction of the cryptologic bomb. We enclose an implementation of both algorithms in Cpp language.*

Keywords cryptologic bomb, M3 Enigma machine, plugboard

1. Introduction

The military Enigma machine was a portable electro-mechanical rotor encrypting machine used during the Second World War, mainly by the German military and government services. Beginning in 1932, Polish cryptologists (M. Rejewski, J. Różycki, and H. Zygalski) systematically worked on decoding ciphers, constantly modified manners of generating secret messages, and modernized the construction of Enigma machines.

The algorithms presented below can be used to decode messages transmitted after September 15, 1938. That day, the German service withdrew *initial drum settings* from *tables of daily key settings* and changed the manner of announcing *message settings*.

The proposed plugboard algorithm is the authors' idea. The cryptologists could not guess the connections of the plugboard with this method within 20 minutes with the use of the technology of that time. They used various tricks that relied on knowledge of the German language. The presented algorithm does not depend on any language.

The second algorithm is a reconstruction of Rejewski's bomb. This algorithm was assembled on the basis of information which had been found in the literature (mainly historical sources). Historians often make factual mistakes; therefore, ambiguously-described facts were completed with the authors' observations. To get the total algorithm, we tested different possibilities and chose the ones which provided the proper result.

The German service used different kinds of Enigma machines (also commercial), modified their construction, and changed the manner of generating secret messages. However, we are only interested in the M3 Enigma machine. For the reader's convenience, we described the construction, the parameters of this machine, and the manner of generating messages transmitted after September 15, 1938 in Appendix A. In Appendix B, we described the work of Polish cryptologists. These sections make up a brief survey of well-known information taken from publications [8, 23, 9, 7, 10, 2, 5, 16]. We suggest reading Appendix A in the beginning to better understand the terms and facts that we use. These terms are denoted in this paper by *. In sections 3 and 5, we provide some mathematical facts concerning permutations and, in particular, 1-cycles (which are essential to understand the presented methods). In section 4, the reader can find a mathematical analysis of the M3 Enigma machine. Section 6 contains a description and justification of the plugboard algorithm (the authors' idea). In section 7, we provide a reconstruction of the bomb method. By means of these two algorithms, we can generate a complete *daily key settings** ; i.e., the ring settings, the choice of drums, the order of drums, and the plugboard settings on the basis of a given set of messages intercepted after September 15, 1938. This allows us to read the encrypted messages. We enclose an implementation of both of these algorithms in Cpp language.

2. Ciphers

We can find the first ciphers in antiquity. Then, among other things, steganography was used. That is, secret information was concealed by means of different techniques; for example, secret text was hidden in an unimportant text. In Egypt and China, invisible ink was used. We also know the Caesar cipher and scytale method. Later, the following groups of classical ciphers were used: transposition ciphers, substitution ciphers (monoalphabetic (e.g., the affine cipher), homophonic and polyalphabetic (e.g., the Hill cipher, the Vigenere cipher)).

In the early 19th century, electro-mechanic devices were designed for encrypting messages, and cryptologists made use of the telegraph and radio. The World Wars brought about an abundance of ciphering machines, the most noted being: Enigma (German rotor machine), Purple (Japanese machine), the German Lorenz machine, American SIGABA, and British Typex. These machines were designed to protect military and diplomatic information, but similar devices were used in commerce.

The development of digital computers provided cipher designers with great computational power. The effect of this transition into the digital domain was the rise of binary ciphering algorithms. On the other hand, common access to cheap electronic devices forced cipher designers to create safer products. Ciphers used in contemporary cryptographic systems should remain secure even if the adversary possesses full knowledge of the ciphering algorithm. That is, security of the key used should be sufficient to maintain confidentiality of communication when under attack (Kerckhoffs's principle). A cryptologic algorithm becomes threatened when an exhaustive attack on a key becomes possible as a result of progress in computing technology. Conditionally-secure ciphers are not very sensitive to the increment of computing power of the attacker. To prevent an effective attack, designers add larger security parameters.

Regarding the type of key, we distinguish two classes of ciphering algorithms: symmetric-key algorithms and public-key (asymmetric-key) algorithms. In symmetric-key algorithms, both the sender and receiver share the same key (sometimes their keys are different but related in an easily-computable way). Asymmetric-key algorithms use a pair of different keys: a public one is used to encrypt and a private one to decrypt. The two keys are mathematically related in such a way that there is no effective method of finding the private key on the basis of the public one. Recently, the following symmetric-key algorithms played a major role: DES (Data Encryption Standard), 3DES, CDMF (Commercial Data Masking Facility), IDEA (International Data Encryption Algorithm), the Rijndael algorithm, Standard AES (Advanced Encryption Standard), RC2, RC4, RC5, RC6, and Blowfish. Some of the important asymmetric-key algorithms include: RSA (Rivest, Shamir, Adelman) and the ElGamal algorithm.

In order to protect a cryptosystem from an exhaustive attack (by means of current computing power), cipher designers combine different methods and technologies. They apply substitution boxes (S-boxes), permutation boxes (P-boxes), and substitution-permutation networks, which carry out encryption in multiple rounds

(DES, 3DES, and FEAL). Other means involve a Feistel network, different functions (e.g., one-way functions), bit operations, dividing data into blocks, and combining them with pseudorandom character streams. Other important factors include key-strength analysis, or the use of intractable mathematical problems: the integer factorization problem (RSA), the discrete logarithm problem (ElGamal, RSA), and the knapsack problem (Merkle-Hellman cryptosystem). Cryptologists also make use of cyclic groups and elliptic curves which are currently widely studied (RSA, ElGamal). (cf. [21, 25, 13, 4, 15, 28])

3. Elements of permutation theory

For the reader's convenience, we provide some definitions and facts from permutation theory (cf. [18, 20, 24]). A *permutation* of an n-element set $X = \{1, 2, \ldots, n\}$ is a map $\sigma : X \mapsto X$ of X into itself such that if $i, j \in X$ and $i \neq j$ then $\sigma(i) \neq \sigma(j)$. The set of permutations of X we denote by Sn. We shall represent any permutation $\sigma \in Sn$ by $2 \times n$ matrix $\sigma = \begin{pmatrix} 1 & 2 & \ldots & n \\ \sigma(1) & \sigma(2) & \ldots & \sigma(n) \end{pmatrix}$.

The *inverse* of a permutation σ (denoted by σ^{-1}) is defined as the map $\sigma^{-1} : X \mapsto X$ such that $\sigma^{-1}(k) =$ unique integer j such that $\sigma(j) = k$. We shall form a *product* (a *composition*) of two permutations $f, g \in Sn$ in the same way as M. Rejewski in [23] i.e.,

$$f \circ g = \begin{pmatrix} 1 & 2 & \ldots & n \\ f(1) & f(2) & \ldots & f(n) \end{pmatrix} \circ \begin{pmatrix} 1 & 2 & \ldots & n \\ g(1) & g(2) & \ldots & g(n) \end{pmatrix} = \\ = \begin{pmatrix} 1 & 2 & \ldots & n \\ g(f(1)) & g(f(2)) & \ldots & g(f(n)) \end{pmatrix}; \quad \text{e.g.,}$$

$$f \circ g = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 2 & 4 \end{pmatrix} \circ \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 1 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 3 & 2 \end{pmatrix}.$$

This notation is different from standard notation, but it is used by authors interested in the Enigma machine. We usually omit the circle $\circ$, and write fg for the composite map. The composition of permutations is associative.

For any permutation σ, we have $\sigma^{-1}\sigma = id_X$ and $\sigma\sigma^{-1} = id_X$, where id_X is the *identity* permutation, that is, the permutation such that $id_X(i) = i$ for all $i = 1, 2, \ldots, n$. A permutation $\sigma \in Sn$ is called a *cycle of length k* or *k-cycle* if there are k elements a_1, a_2, ..., a_k of the domain $X = \{1, 2, \ldots, n\}$ such that $\sigma(a_1) = a_2$, $\sigma(a_2) = a_3$, ..., $\sigma(a_{k-1}) = a_k$, $\sigma(a_k) = a_1$ and for the remaining elements of X we have $\sigma(a_i) = a_i$. A more compact way of writing a k-cycle is $\sigma = (a_1, a_2, \ldots, a_k)$. It is understood that σ maps each a_i, $i = 1, 2, \ldots, k-1$ into the element listed on its immediate right, a_k into a_1, and each unlisted integer of X into itself. A cycle of length two is called a *transposition*.

4. Mathematical analysis

Let $\mathbf{P} = \{\mathtt{A}, \mathtt{B}, \ldots, \mathtt{Z}\}$ be a set of possible plaintexts, and let $\mathbf{C} = \mathbf{P}$ be a set of possible ciphertexts. The M3 Enigma machine ciphers text T by using a poly-alphabetic substitution cipher. Each letter $a \in \mathbf{P}$ of the message is transformed according to the following permutation (cf. [23])

$$\Lambda = SH(Q^z R Q^{-z} Q^y M Q^{-y} Q^x L Q^{-x}) B (Q^x L^{-1} Q^{-x} Q^y M^{-1} Q^{-y} Q^z R^{-1} Q^{-z}) H^{-1} S^{-1} \tag{1}$$

S – is a permutation describing the *plugboard** transformation (S consists of transpositions and 1-cycles only),
B – is a permutation describing the *reflector** transformation (B consists of 13 transpositions),
L, M, R – are permutations describing transformations of the three *cipher drums**,
H – is a transformation of the *entry wheel** (H is the identity permutation),
$Q = (\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}, \mathtt{G}, \mathtt{H}, \mathtt{I}, \mathtt{J}, \mathtt{K}, \mathtt{L}, \mathtt{M}, \mathtt{N}, \mathtt{O}, \mathtt{P}, \mathtt{Q}, \mathtt{R}, \mathtt{S}, \mathtt{T}, \mathtt{U}, \mathtt{V}, \mathtt{W}, \mathtt{X}, \mathtt{Y}, \mathtt{Z})$ – a cycle of length 26,
$n(\alpha)$ – the number of a letter α (value from set $\mathbf{IP} = \{0, 1, \ldots, 25\}$),
x, y, z – are the positions of *rotors** before pressing any key (values from set $\mathbf{IP}$),
$x = (26 + n(\alpha) - n(\delta))\%26$ for the left rotor,
$y = (26 + n(\beta) - n(\epsilon))\%26$ for the middle rotor,
$z = (26 + n(\gamma) - n(\zeta))\%26$ for the right rotor (cf. [8]),
α, β, γ – positions of *drums** (left, middle and right) before pressing any key,
δ, ϵ, ζ – positions of *rings** (left, middle and right) ($\alpha, \beta, \gamma, \delta, \epsilon, \zeta \in \mathbf{P}$)

5. 1-cycles

Let us assume that we eavesdropped four messages on the same day with the following *headlines**. This means that these messages were generated for the same *daily key settings**.

(1) XFI ADR AXF (2) TWP HNP LNR (3) ADM DOD YKD (4) ZHO IEF YEM

The first 3 letters of each message make up the *initial drum settings**. The Enigma codes the next 6 letters of a message (meant as double coded *message settings**) using the following permutations (cf. [23]).

$$A = SHQ^{z+1} R Q^{-(z+1)} Q^y M Q^{-y} Q^x L Q^{-x} B Q^x L^{-1} Q^{-x} Q^y M^{-1} Q^{-y} Q^{z+1} R^{-1} Q^{-(z+1)} H^{-1} S^{-1}$$
$$B = SHQ^{z+2} R Q^{-(z+2)} Q^y M Q^{-y} Q^x L Q^{-x} B Q^x L^{-1} Q^{-x} Q^y M^{-1} Q^{-y} Q^{z+2} R^{-1} Q^{-(z+2)} H^{-1} S^{-1}$$
$$C = SHQ^{z+3} R Q^{-(z+3)} Q^y M Q^{-y} Q^x L Q^{-x} B Q^x L^{-1} Q^{-x} Q^y M^{-1} Q^{-y} Q^{z+3} R^{-1} Q^{-(z+3)} H^{-1} S^{-1}$$
$$D = SHQ^{z+4} R Q^{-(z+4)} Q^y M Q^{-y} Q^x L Q^{-x} B Q^x L^{-1} Q^{-x} Q^y M^{-1} Q^{-y} Q^{z+4} R^{-1} Q^{-(z+4)} H^{-1} S^{-1}$$
$$E = SHQ^{z+5} R Q^{-(z+5)} Q^y M Q^{-y} Q^x L Q^{-x} B Q^x L^{-1} Q^{-x} Q^y M^{-1} Q^{-y} Q^{z+5} R^{-1} Q^{-(z+5)} H^{-1} S^{-1}$$
$$F = SHQ^{z+6} R Q^{-(z+6)} Q^y M Q^{-y} Q^x L Q^{-x} B Q^x L^{-1} Q^{-x} Q^y M^{-1} Q^{-y} Q^{z+6} R^{-1} Q^{-(z+6)} H^{-1} S^{-1}$$

We can obtain permutations A, B,..., F in the following way. We set up our Enigma machine in the same way as the coder had set his machine up during ciphering.

Next, we type all of the letters of the alphabet in order without using the *turning mechanism** (cf. [10]). In the case of the message (1), we receive

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

Permutations A, D consist of 13 transpositions. On the basis of double-ciphered message settings ADR AXF, we can notice that A as well as D permutations contain a transposition $(x, \mathtt{A})$, where x is a ciphered letter (different from A). In the case of the message (1), that is the transposition (A, Q). Therefore, the product AD assigns the letter A to the letter A and the letter Q to the letter Q. Thus, there are 1-cycles in the permutation AD. Below, we write down the permutation AD as a product of disjoint cycles

$$AD: \quad (\mathtt{A})\,(\mathtt{BIFJ})\,(\mathtt{CMTRVEZS})\,(\mathtt{DKNWUOPL})\,(\mathtt{GXHYG})\,(\mathtt{Q})$$

We can see that the first letter of the message settings was the letter Q. We proceed in the same way when identical letters are located in positions two and five (or three and six) in double-ciphered message settings, because it means that a permutation BE (or CF) contains 1-cycles (cf. [8]).

6. Plugboard algorithm (the authors' idea)

The Zygalski's sheets method and the cryptologic bomb were used to generate the daily key settings after September 15, 1938. Neither of these methods give plugboard settings. It is a well-known fact that the plugboard connections do not have any influence on the work of both of these methods. This means that, with these methods, we can obtain the first 3 elements of the daily key settings regardless of how the coder set his plugboard up during ciphering.

The presented plugboard algorithm can be used for cryptanalysis together with both Zygalski's method and the bomb method. We used the fact that identical letters in positions one and four (or two and five or three and six) in double-ciphered message settings mean that a permutation AD (or BE or CF) contains 1-cycles.

We received the following results for real connections of both drums and plugboard that were used by Wehrmacht. We assume (according to the knowledge of that time) that we know connections of all kinds of drums. Given examples were executed for drums: L = I, M = II, R = III, for the reflector B = UKW B and for the plugboard connections $S = (\mathtt{ET})(\mathtt{IX})(\mathtt{MQ})(\mathtt{NV})(\mathtt{PU})(\mathtt{YZ})$. Let us fix, that we shall treat the letters A, B,..., Z of the Latin alphabet as the numbers from the set $\mathbf{IP} = \{0, 1, \ldots, 25\}$.

The output contacts of cipher drums I, II, and III (cf. [8]):

	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
I:	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
II:	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
III:																										

The turnover positions for selected drums: I – Q, II – E, III – V (cf. [8]).
The reflector connections (cf. [8]):

UKW B : (AY) (BR) (CU) (DH) (EQ) (FS) (GL) (IP) (JX) (KN) (MO) (TZ) (VW)

Below, we present two protocols of a cryptologist's work. (A) when the algorithm returns all daily key settings (with connections of plugboard), and (B) when the algorithm gives daily key settings without plug connections (the cryptologist receives a permutation S, which represents a plugboard in another way and he sets up a plugboard himself). For greater clarity, we shall notice that $S = S^{-1}$ (S consists of transpositions and 1-cycles only) and write the permutation Λ (cf. (1), §4) in another way.

$$\Lambda = S\Lambda_H S^{-1}, \quad \text{where}$$
$$\Lambda_H = H(Q^z R Q^{-z} Q^y M Q^{-y} Q^x L Q^{-x}) B (Q^x L^{-1} Q^{-x} Q^y M^{-1} Q^{-y} Q^z R^{-1} Q^{-z}) H^{-1} \tag{2}$$

Example 6.1 Two protocols of a cryptologist's work
Text T: PLUGBOARD ALGORITHM
The daily key settings: drums: I, II, III, the ring settings: Rs = EHM, plug connections S = (ET) (IX) (MQ) (NV) (PU) (YZ)
EFE – initial positions of drums
IOP – message settings
HDB RYU – double-ciphered message settings
DCVEOYBEPVSHFLXGZJ – a ciphered text
EFE HDB RYU DCVEOYBEPVSHFLXGZJ – a message
A protocol (A) – the algorithm returns all daily key settings

- Cryptologist sets up his machine (all parameters without message settings).
- He sets the drums to EFE.
- He types the text HDBRYU and the program returns message settings IOP(IOP).
- He sets the drums to IOP.
- He types the text DCVEOYBEPVSHFLXGZJ and the program returns a result PLUGBOARDALGORITHM.

A protocol (B) – the algorithm gives daily key settings without plug connections (cryptologist receives plug connections S = (ET) (IX) (MQ) (NV) (PU) (YZ) in another way).

- Cryptologist sets up his machine (all parameters without message settings and plugboard settings).
- He reconstructs message settings.
  - He sets the drums to EFE.
  - He changes manually the text HDBRYU to HDBRZP (according to a permutation S).
  - He types the text HDBRZP and the program returns message settings XOU(XOU).

- ◦ He changes manually the text XOUXOU to IOPIOP (according to S). IOP makes up message settings.
- • He decodes the ciphered text.
- ◦ He sets the drums to IOP.
- ◦ He changes manually the text DCVEOYBEPVSHFLXGZJ to DCNTOZBTUNSHFLIGYJ.
- ◦ He types DCNTOZBTUNSHFLIGYJ and the program returns ULPGBOARDALGORXEHQ.
- ◦ He changes manually the text ULPGBOARDALGORXEHQ to PLUGBOARDALGORITHM.

6.1. Schema of the plugboard algorithm

We denote by (X^{14}) any established message of the form $\alpha\beta\gamma\ kx_2x_3\ kx_5x_6$, where α, β, γ, x_2, x_3, x_5, x_6, k belong to the set $\{\mathtt{A}, \mathtt{B}, \ldots, \mathtt{Z}\}$. Similarly, by (X^{25}), (X^{36}) we denote messages with identical letters in positions two and five (or three and six) in double-ciphered message settings. In order to guess the plug connections, we shall use messages of these forms. For these messages, we shall calculate permutations Λ_H (where $\Lambda = S\Lambda_H S^{-1}$) instead of permutations Λ. We obtain a permutation Λ_H by substituting the identity permutation H for a permutation S.

Messages of the form (X^{14}) (or (X^{25}) or (X^{36})) that are received for the same daily key settings only make up an input to the plugboard algorithm.

Example 6.2 We shall use the set of messages from the table 1 to determine plugboard settings.

Schema of the plugboard algorithm

1. Set up your machine according to the daily key settings without plugboard connections (i.e., plug connections are represented by the identity permutation H)
2. Add messages to the vector `M` (compare the example 6.2).
3. Split messages into two vectors `V1` and `V2`.
   - Add to the vector `V2` couples of messages of the form (X^{14}) with the same recurrent letter k; e.g., messages (6) `XFP EAZ EEQ` and (7) `DMA EFP EMR`. Each two messages are represented in `V2` by means of one object of the `_1cycle` class which contains three fields (`c` – a recurrent letter, `s1` – a string with 1-cycles of a permutation $A_{1H}D_{1H}$ for the first message and `s2` – 1-cycles of a permutation $A_{2H}D_{2H}$ for the second message). We draw the reader's attention to the fact that a permutation can contain several 1-cycles. For messages (6) and (7), the program creates the object [`c=E, s1=GT, s2=DEFMNSTY`].
   - Add to the vector `V1` the remaining messages; i.e., these messages for which we cannot find another message with the same letter in positions one and four in double-ciphered message settings (e.g., the message (1) `ICE AJA ADV`). Each message in `V1` is represented by one object of the `_1cycle` class which contains two fields (`c` – a recurrent letter and `s1` – a string with 1-cycles of a permutation $A_{1H}D_{1H}$ for the analyzed message). For message (1), the program generates the object [`c=A, s1=AY`].

Table 1

The set of messages which were used in experiments.

	messages	1-cycles of $A_H D_H$
(1)	ICE AJA ADV NEHUMKYXSJUPKGNS	AY
(2)	BID BOZ BFG NVCKOXAHZRUOBAKG	BC
(3)	CFP CUR CNU ZMCMCWSKNKQLCYQK	AC
(4)	APQ CEK CTR DDLCKIEGBXUAGNCE	CH
(5)	HGN DQG DVV LHKSWWVOBZOUFBWH	DT
(6)	XFP EAZ EEQ ZOXFCZMUFBQMPMGS	GT
(7)	DMA EFP EMR JTLWAFNHKGWCZMMM	DEFMNSTY
(8)	CBN FHM FAZ OWDJMRCPBYTPUGCH	EFLN
(9)	YIN FHO FKQ OFEQHJNVQUWLJDFS	FW
(10)	DHS GPA GVF WCLLXWYYMWULIOLW	GN
(11)	XFC HLE HTW CHDGUHRJUOQJXQJV	GH
(12)	EIB IWQ IZA XPKRJWAMXKJOGFOK	DIMX
(13)	ABR IEP IHD PQFRHEYNQABMKIDY	AUXY
(14)	ABE JSZ JXQ AYOINRILBOPPYUAR	FJ
(15)	ZCB KEB KYT RKPJGDEKWBYFGVSP	KOQU
(16)	PRS LRL LIX ZKWSPKQJIOEFMAEB	BLVZ
(17)	BCD MFA MFL OKMXDPKPNYKEAHCE	QU
(18)	QDM MFQ MUS UREPNBWUDNOCPTBR	QY
(19)	WHE NAH NXU HLZXEJILZHAOSTPT	CV
(20)	DIT OTL OFQ AVRWNAIGEGWJWUEK	MO
(21)	QIV RAM RRQ BBLQYVRUMOOCCPQE	RW
(22)	AAR SLP SBA MRRXMOJOGUAFHPMQ	OSUZ
(23)	CGC UYP UDL WMGCSRAGFTIRYYWR	NP
(24)	LDL WQI WMA LEHNNHXVWFBVPYWI	NW
(25)	XFF YDK YPX XHPJKTVPHXWINRLO	FZ

4. An analysis of the vector `V2`

   - The program analyzes objects of the vector `V2` and reconstructs a permutation S. The permutation S transforms a recurrent letter `c` into a letter of the alphabet which occurs in both strings `s1` and `s2`. For messages (6) and (7) $S(\mathtt{E}) = \mathtt{T}$ and $S(\mathtt{T}) = \mathtt{E}$. Each object of `V2` gives one or two positions of the permutation S.

5. An analysis of the vector `V1`

   - The program analyzes objects of the vector `V1` and reconstructs the permutation S. The permutation S certainly transforms a recurrent letter `c` into one of the letters of a string `s1`. An analyzed object allows us to reconstruct the permutation S only if positions of this permutation were reconstructed in advance for all the letters of a string `s1` except for one letter (e.g., $S(\mathtt{V}) = \mathtt{Z}$, $S(\mathtt{X}) = -$, $S(\mathtt{Y}) = -$, $S(\mathtt{Z}) = \mathtt{Y}$ and `s1` = `VXZ`). Then $S(\mathtt{c})$ = the only

letter of s1 for which S is not defined. By "–" we denote no reconstructed positions of a permutation.

- The vector V1 is analyzed repeatedly. We end the analysis of V1 when the previous iteration did not give any new position of the permutation S.

Example 6.2 Continuation. Executing the plugboard algorithm.

1. We set up the Enigma in the following way: drums I, II, III, ring settings Rs = EHM, plug connections – according to the identity permutation.
2. We shall analyze the given messages.
3. Vectors V2 and V1

c	C	E	F	I	M
s1	AC	GT	EFLN	DIMX	QU
s2	CH	DEFMNSTY	FW	AUXY	QY

c	A	B	D	G	H	J	K	L	N	O	R	S	U	W	Y
s1	AY	BC	DT	GN	GH	FJ	KOQU	BLVZ	CV	MO	RW	OSUZ	NP	NW	FZ

4. Below we present the permutation S after analyzing the vector V2.

```
   ABCDEFGHIJKLMNOPQRSTUVWXYZ
S  --C-TF--X---Q---M--E---I--
```

5. The analysis of objects of the vector V1. The first iteration.
   Object [A, AY]: Either $S(\mathtt{A}) = \mathtt{A}$ or $S(\mathtt{A}) = \mathtt{Y}$. We cannot reconstruct any position of S.
   Object [B, BC]: Either $S(\mathtt{B}) = \mathtt{B}$ or $S(\mathtt{B}) = \mathtt{C}$. Since $S(\mathtt{C}) = \mathtt{C}$, therefore $S(\mathtt{B}) = \mathtt{B}$.

```
   ABCDEFGHIJKLMNOPQRSTUVWXYZ
S  -BC-TF--X---Q---M--E---I--
```

   Object [D, DT]: Either $S(\mathtt{D}) = \mathtt{D}$ or $S(\mathtt{D}) = \mathtt{T}$. Since $S(\mathtt{E}) = \mathtt{T}$, therefore $S(\mathtt{D}) = \mathtt{D}$.

```
   ABCDEFGHIJKLMNOPQRSTUVWXYZ
S  -BCDTF--X---Q---M--E---I--
```

   Objects [G, GN] and [H, GH] do not give any new position of S.
   Object [J, FJ]: Either $S(\mathtt{J}) = \mathtt{F}$ or $S(\mathtt{J}) = \mathtt{J}$. Since $S(\mathtt{F}) = \mathtt{F}$, therefore $S(\mathtt{J}) = \mathtt{J}$.

```
   ABCDEFGHIJKLMNOPQRSTUVWXYZ
S  -BCDTF--XJ--Q---M--E---I--
```

   Objects [K, KOQU] and [L, BLVZ] do not give any new positions of S.
   Object [N, CV]: Either $S(\mathtt{N}) = \mathtt{C}$ or $S(\mathtt{N}) = \mathtt{V}$. $S(\mathtt{C}) = \mathtt{C}$, thus $S(\mathtt{N}) = \mathtt{V}$ and $S(\mathtt{V}) = \mathtt{N}$.

```
   ABCDEFGHIJKLMNOPQRSTUVWXYZ
S  -BCDTF--XJ--QV--M--E-N-I--
```

   After the first iteration we get the following permutation S.

```
   ABCDEFGHIJKLMNOPQRSTUVWXYZ
S  -BCDTF--XJ--QVOUM--EPN-IZY
```

After the second iteration we get the complete permutation S.

```
    ABCDEFGHIJKLMNOPQRSTUVWXYZ
S   ABCDTFGHXJKLQVOUMRSEPNWIZY
```

6.2. Correctness of the plugboard algorithm

We are interested in any established messages of the form (X^{14}) that were generated for the same daily key settings (the choice and order of drums, ring settings, plug connections). For example

(6) XFP EAZ EEQ ZOXFCZMUFBQMPMGS

// 1-cycles of AD : (E)(G), 1-cycles of A_HD_H : (G)(T)

In section 5, we established that permutations A, D of each message of this form contain the same transposition (k, x), where a letter k occurs in positions one and four in double ciphered message settings and x is the first sign of message settings (x is different from k). That is, permutations A, D transform k into x and x into k, where $x = A(k) = D(k)$. Hence, the permutation AD contains at least two 1-cycles.

```
A :   (AP) (BJ) (CL) (DO) (EG) (FY) (HV) (IZ) (KM) (NT) (QS) (RW) (UX)
D :   (AX) (BU) (CO) (DZ) (EG) (FH) (IM) (JL) (KV) (NW) (PY) (QT) (RS)
AD :  (AYHKIDCJU) (BLOZMVFPX) (E) (G) (NQR) (STW)
```

Let us assume that we broke all elements of the daily key settings except the plugboard settings that are represented by a permutation S. In order to guess the plug connections, we shall replace the permutation S with the identity permutation H. Then, during ciphering, permutations A_H, D_H will be generated instead of permutations A and D (cf. (2), §6).

```
A_H :   (AU) (BJ) (CL) (DO) (EV) (FZ) (GT) (HN) (IP) (KQ) (MS) (RW) (XY)
D_H :   (AI) (BP) (CO) (DY) (EM) (FH) (GT) (JL) (KN) (QX) (RS) (UZ) (VW)
A_HD_H : (AZHKXDCJP) (BLOYQNFUI) (EWS) (G) (MRV) (T)
```

Let us consider any established message X_1 of the form (X^{14}).

Lemma 6.1 Let us assume that permutations A, D (generated for this message) contain the same transposition (k, x). Then, corresponding to them, permutations A_H, D_H also contain the identical transposition (k_H, x_H), where $k_H = S(k)$, $x_H = A_H(k_H) = D_H(k_H) = SA_H(k) = SD_H(k)$.[1]

It follows from the fact that the identity permutation H meets the assumptions of the permutation S (S consists of transpositions and 1-cycles only).

Let us consider two established messages X_1 and X_2 of the form (X^{14}) with the same letter k. Let permutations A_{1H}, D_{1H} for message X_1 contain the identical transposition (k_{1H}, x_{1H}) and permutations A_{2H}, D_{2H} for message X_2 contain transposition (k_{2H}, x_{2H}). We know that $k_{1H} = k_{2H} = S(k)$ (cf. lemma 6.1).

[1] We remind the reader that we use another notation for a product of permutations.

Hence, in order to reconstruct the permutation S, we need several couples of messages of the form (X^{14}) with a different letter k for each couple. Then, we determine permutations A_{1H}, D_{1H} and A_{2H}, D_{2H} and transpositions (k_{1H}, x_{1H}), (k_{2H}, x_{2H}) for each couple of messages. Since $k_{1H} = k_{2H} = S(k)$, we obtain $S(k) = k_{1H} = k_{2H}$ and $S(k_{1H}) = S(k_{2H}) = k$.

Example 6.3 Let us consider the two following messages
(6) XFP EAZ EEQ //1-cycles of AD: (E)(G), 1-cycles of $A_H D_H$: (G)(T)
(7) DMA EFP EMR //1-cycles of AD: (D)(E)(F)(Q)(S)(T)(V)(Z),
//1-cycles of $A_H D_H$: (D)(E)(F)(M)(N)(S)(T)(Y)
Permutations A_{1H}, D_{1H} for message (6) contain the same transposition (G, T) and permutations A_{2H}, D_{2H} for message (7) contain the same transposition (N, T). Since the letter T occurs in both transpositions, we guess that $k_{1H} = k_{2H} = \mathtt{T} = S(k) = S(\mathtt{E})$. Therefore, $S(\mathtt{E}) = \mathtt{T}$ and $S(\mathtt{T}) = \mathtt{E}$.

6.3. Implementation

The `PlugBoard()` method of the Enigma class generates a permutation S which represents plug connections according to the schema given in section 6.1. This method adds messages to the vector `M`, splits them into two vectors `V1` and `V2`, and then reconstructs positions of the permutation S (by means of two methods `GenerateS2()` and `GenerateS1()`). The `GenerateS2()` method analyzes each element (an object of the `_1cycle` class) of `V2` in order to reconstruct the permutation S. First, it checks whether $S(\mathtt{c})$ is known or not. If $S(\mathtt{c}) = -$, the method looks for a common letter in strings `s1` and `s2`. The common letter is element k_H, so $S(\mathtt{c}) = k_H$ and $S(k_H) = \mathtt{c}$. The `GenerateS1()` method checks whether a position $S(\mathtt{c})$ is defined or not. If $S(\mathtt{c}) = -$, the method checks if in a string `s1` exists exactly one letter (e.g., A), for which $S(\mathtt{A})$ is not defined. If such a letter exists, the permutation S is reconstructed as follows $S(\mathtt{c}) = \mathtt{A}$ and $S(\mathtt{A}) = \mathtt{c}$. Otherwise, S is not reconstructed. The `createAorD()` method generates a permutation Λ (cf. the formula (1), §4) for the ring settings `rs` and for the drum settings `dr`. `HE` is an object of the `Enigma` class. The `codeLetter()` method ciphers a letter described by the first parameter for the rotor settings determined by the next three parameters. The `moveD()` method shifts drum settings `dr` for `k` presses of keys. By `tpR` we denote the turnover position for the right drum. The `ITC()` method changes a number from the set $\mathbf{IP} = \{0, 1, \ldots, 25\}$ into a suitable letter. The `CTI()` method does the opposite operation. The `Fem()` method returns a string with 1-cycles of the permutation for which it was called.

```
( 1) void Plugs::PlugBoard(){
( 2) Perm *A1, *D1, *AD, *S; _1cycle *_1c;
( 3) String c="", s1="", s2="", s=""; unsigned int i=0;
( 4) HE->readEnigma();
//add messages to the vector M
( 5) while(i<M.size()){
( 6)   A1 = createAorD(HE->Rs,HE->moveD(M[i]->s1,1));
( 7)   D1 = createAorD(HE->Rs,HE->moveD(M[i]->s1,4));
( 8)   AD = A1->Prod(D1);
( 9)   c=M[i]->s2[1]; s1=AD->Fem(); s2="";
```

```
(10)   _1c = new _1cycle(c,s1,s2);
(11)   delete A1; delete D1; delete AD;
(12)   if(i+1<M.size() && M[i]->s2[1]==M[i+1]->s2[1]){i++;
(13)      A1 = createAorD(HE->Rs,HE->moveD(M[i]->s1,1));
(14)      D1 = createAorD(HE->Rs,HE->moveD(M[i]->s1,4));
(15)      AD = A1->Prod(D1);
(16)      _1c->s2=AD->Fem();
(17)      delete A1; delete D1; delete AD;
(18)      V2.push_back(_1c);}
(19)   else V1.push_back(_1c);
(20)   i++;}
(21) S = GenerateS2();
(22) GenerateS1(S);}
//------
(23) Perm* Plugs::GenerateS2(){
(24) Perm *S = new Perm(26);
(25) for(unsigned int i=0; i<V2.size(); i++)
(26)   if(S->P[CTI(V2[i]->c[1])]=='_')
(27)      for(int j=1; j<=V2[i]->s1.Length(); j++)
(28)         if(V2[i]->s2.Pos(V2[i]->s1[j])!=0){
(29)            S->P[CTI(V2[i]->c[1])]=V2[i]->s1[j];
(30)            S->P[CTI(V2[i]->s1[j])]=V2[i]->c[1]; break;}
(31) return S;}
//------
(32) void Plugs::GenerateS1(Perm* S){
(33) bool b=true; String s1=""; int p1, p2;
(34) while(b){b=false;
(35)   for(unsigned int i=0; i<V1.size(); i++)
(36)      if(S->P[CTI(V1[i]->c[1])]=='_'){s1="";
(37)         for(int j=1; j<=V1[i]->s1.Length(); j++)
(38)            s1+=S->P[CTI(V1[i]->s1[j])];
(39)         p1=s1.Pos('_'); p2=s1.SubString(p1+1,s1.Length()).Pos('_');
(40)         if(p1!=0 && p2==0){b=true;
(41)            S->P[CTI(V1[i]->c[1])]=V1[i]->s1[p1];
(42)            S->P[CTI(V1[i]->s1[p1])]=V1[i]->c[1];}}}}
//------
(43) Perm* Plugs::createAorD(String rs, String dr){
(44) HE->setEnigma(rs,dr);
(45) char* A = new char[27]; A[0]=26;
(46) for(int i=1; i<=26; i++) A[i]=HE->codeLetter(pH[i],HE->Rt[1],HE->Rt[2],HE->Rt[3]);
(47) return new Perm(A);}
//------
(48) String Enigma::moveD(String dr, int k){
(49) String BE = dr, BEH="";
(50) for(int i=1; i<=k; i++){BEH=BE;
(51)   BE[3]=ITC((CTI(BE[3])+1)%26);
(52)   if(BEH[3]==tpR) BE[2]=ITC((CTI(BE[2])+1)%26);
(53)   if(BEH[2]==tpM){
(54)      BE[1]=ITC((CTI(BE[1])+1)%26); BE[2]=ITC((CTI(BE[2])+1)%26);}}
(55) return BE;}
```

6.4. How to transform messages

In order to reconstruct a permutation S, we can use messages (X^{25}) and (X^{36}) in the same way. We can mix them, but if we analyze messages with the same letter k, we must take a couple of messages of the same form. If we have two messages with the same letter k and these messages are of different kinds (e.g., (X^{14}) and (X^{25})), we can transform one of them into the other form.

Example 6.4 Let us consider the following message

PBT VAD CAG

WPO makes up message settings for this message. After we shift PBT drum settings by one key press we get PBU drum settings. Let us set up drums to PBU (instead of PBT) and type two times message settings WPO (but shifted to the right by one letter i.e., POWPO? instead of WPOWPO). We obtain ADCAG? instead of VADCAG. We can see that double-ciphered message settings are also shifted to the right by one letter. Thus if we have two messages

ICE AJA ADV

PBT VAD CAG

one of the form (X^{14}) and the other of the form (X^{25}) but both with the same letter k, we can transform one of them into the other form. We shall obtain

ICE AJA ADV

PBU ADC AG?

Next, program will add both these messages to the vector V2. Certainly, we can transform messages of the form (X^{36}) in the same way.

Justification. Let us consider a message X_1 of the form (X^{25}) i.e., $\alpha\beta\gamma\ x_1kx_3\ x_4kx_6$, where α, β, γ, x_1, x_3, x_4, x_6, k belong to the set $\{\mathtt{A}, \mathtt{B}, \ldots, \mathtt{Z}\}$. Let us assume that $K_1K_2K_3$ makes up message settings of this message; i.e.,

$$A(K_1) = x_1,\ B(K_2) = k,\ C(K_3) = x_3,\ D(K_1) = x_4,\ E(K_2) = k,\ F(K_3) = x_6.$$

Let us press any key before coding and begin ciphering with the second letter.

$$B(K_2) = k,\ C(K_3) = x_3,\ D(K_1) = x_4,\ E(K_2) = k,\ F(K_3) = x_6.$$

We obtained $kx_3x_4kx_6$? for drum settings shifted by one key press.

7. Reconstruction of the cryptologic bomb method

7.1. Historical specification of the bomb method

Rejewski's bomb method was used to break the daily key settings of messages transmitted after September 15, 1938. This semiautomatic device consisted of three couples of the M3 Enigma machines in which rotors were propelled by an electric motor. Each couple of machines analyzed one message of the form (X^{14}) (or (X^{25}) or (X^{36})). Additionally, all three messages had to contain the same recurrent letter k (in double-ciphered message settings). For each couple of machines, drums of the first Enigma were shifted on the basis of initial drum settings (divergent information cf. [8, 9, 10]). Drums of the second Enigma were shifted by pressing three keys in relation to drums of the first Enigma (cf. [9]). Next, the bomb was activated. The machine stopped when each couple of Enigmas reached a common code of recurrent letter k (different for each couple). Then, cryptologists read ring settings and activated the bomb again. The bomb usually stopped a few times only (0, 1, 2, or 3). Next, all of the

obtained ring settings were checked manually. The bomb method did not solve plug connections, but the plugboard settings did not have any influence on the work of this method. The bomb did not guess the daily key settings when a recurrent letter k had been changed by the plugboard (cf. [8, 9, 10]).

Example 7.1 Let us consider the two following messages that were generated for the same ring settings and for the same order of drums I, II, III with turnover positions Q, E, V respectively.

(1) CPV QTM KZE
(2) CQX MHW EVE

Double-ciphered message settings of these messages was generated for the following drum settings.

L: CCCCCCCC
M: PQQQQQQQ
R: VWXYZABC
QTMKZE
MHWEVE

Let us notice that the right drum reached a turnover position V. It caused the middle drum to turn from position P to Q. Since the third letter of the message (1) and the first letter of the second message are the same and both of them are coded for the same CQY drum settings, they are the code of the same letter x. Therefore, the sixth letter of the message (1) and the fourth letter of the message (2) are also the code of the same letter x (since six first letters make up double ciphered message settings). These letters are identical (equal E) because both of them are the code of the same sign x and both are ciphered for the same CQB drum settings. If these two letters were not the same, it would mean the drums had been set in an order different than I, II, III (cf. [8]). The existence of these messages does not guarantee that the drums were set in order I, II, III. It may be a coincidence (because the Enigma ciphers signs on the basis of rotor settings (i.e., differences between drum settings and ring settings)), but we usually obtain only a few such coincidences (0, 1, 2 or 3).

7.2. Schema of the bomb algorithm

1. Select 3 messages of the form (X^{14}) or (X^{25}) or (X^{36}) with the same letter k.
2. Create 6 Enigma machines (two for each message). Let us name them E_{11}, E_{12}, E_{21}, E_{22}, E_{31}, E_{32}, where the first index means the number of the message and the second – the number of the Enigma in the couple.
3. Set up 6 Enigma machines in the following way.
   - For each Enigma choose identical drums and set them up in the same order (check all orders of drums – $5 * 4 * 3 = 60$ possibilities for 5 drums)
   - Set rings in any established way (the same for each Enigma)
   - Set a plugboard according to the identity permutation (identically for each Enigma)

- For each couple of machines, shift the drums of the first Enigma (on the basis of initial drum settings) to obtain a code of recurrent letter k by means of a permutation A.
- For each couple of machines, shift the drums of the second Enigma by pressing three keys in relation to the drums of the first Enigma (i.e., the second Enigma codes a letter k by means of a permutation D).

4. By means of the instruction `while` test all possible ring settings and for each of them determine codes of recurrent letter k (e.g., for $k =$ `A`). Write down the ring settings (0, 1, 2 or 3 possibilities) for which codes of recurrent letter are identical (different for each couple). Check these possibilities and simultaneously guess the plugboard settings. You can use the plugboard algorithm from section 6.

Example 7.2 Let us consider three messages

(1) `CPV ADC AYL ABBYYYZKDOFAYOYAJKIXRDMVABMSCKCSWYZDHQZMRUZKBZROAULQOFJ`
(2) `EQC QZA UJA JTDCVNMYFJYLWIKWGWSZEZHRTPWRHKNSYZYDTMFUPVYWQWDDCWTGGKS`
(3) `FQM WAZ OAV ULFROOQRMTYVCRSBMPUUACABVQVZVZPHDCLCGVJPOKVDSASPAGZNAQF`

We reveal that, these messages were generated for ring settings $Rs =$ `EHM`, for drums I, II, III and for the plugboard connections $S =$ `(ET)(IX)(MQ)(NV)(PU)(YZ)`.

Executing the bomb algorithm:

3. We show how the program will analyze messages for the order of drums I, II, III.

- Set ring settings to $Rs =$ `ZZZ`.
- Set up drums of 6 machines in the following way.
  E_{11}: `CPV`, E_{12}: `CQY` (drums of E_{12} are shifted by 3 key presses in relation to drums of E_{11})
  E_{21}: `EQE`, E_{22}: `EQH` (drums of E_{21} are shifted by 2 key presses in relation to initial drum settings `EQC`)
  E_{31}: `FQN`, E_{32}: `FQQ` (drums of E_{31} are shifted by 1 key press in relation to initial drum settings `FQM`)

4. The program returns two results $Rs =$ `EHM` and $Rs =$ `UDW`.

- Check whether `EHM` makes up ring settings or not, i.e., set up drums in the order L = I, M = II, R = III, set rings to `EHM` and execute the plugboard algorithm. The program will return the following permutation S.

  `ABCDEFGHIJKLMNOPQRSTUVWXYZ`
  S `ABCDTFGHXJKLQVOUMRSEPNWIZY`

- Set up plugboard according to the permutation S, set drums to `CPV` and type double-ciphered message settings `ADCAYL`. Enigma will return a text `RTY(RTY)`. Thus, `EHM` ring settings are proper.
- If you execute the plugboard algorithm for `UDW` ring settings, you will not get permutation S, because inside the `plugBoard()` method (cf. §6.3) there is the `readEnigma()` method, which reads ring settings.

7.3. Implementation of the algorithm

The `Rings()` method of the `Bomb` class generates potential ring settings for the current order of drums and three messages of the form (X^{14}) (or (X^{25}) or (X^{36})). In the beginning, 6 Enigmas are adjusted; i.e., for all machines rings are set to `ZZZ` and drums are set up according to three-letter strings remembered in a table `BE`. By means of a loop `while` all possibilities of ring settings are tested (17,576 possibilities). First, each Enigma determines a code of a recurrent letter k by means of the `codeNext()` method. If codes of this letter for each couple of Enigmas are the same, the `Rings()` method writes down potential ring settings. Next, by means of the `moveR()` method, the next ring settings are established for all Enigma machines. Let us notice that, according to the specification of the M3 Enigma machine, the `codeNext()` method first shifts drums, next calculates rotor settings and finally codes a sign. The `moveD()` method is defined in section 6.3.

```
( 1) void Bomb::Rings(String *BE, char c){
( 2) char c1, c2, c3, c4, c5, c6; int i=0; String s;
( 3) E11->setEnigma("ZZZ",BE[1]); E12->setEnigma("ZZZ",BE[2]); E21->setEnigma("ZZZ",BE[3]);
( 4) E22->setEnigma("ZZZ",BE[4]); E31->setEnigma("ZZZ",BE[5]); E32->setEnigma("ZZZ",BE[6]);
( 5) while(i<17576){i++;
( 6)   c1 = E11->codeNext(c); c2 = E12->codeNext(c); c3 = E21->codeNext(c);
( 7)   c4 = E22->codeNext(c); c5 = E31->codeNext(c); c6 = E32->codeNext(c);
( 8)   if(c1==c2 && c3==c4 && c5==c6) cout << "Rs = " + E11->Rs;
( 9)   s = E11->moveR(E11->Rs,1);
(10)   E11->setRs(s); E12->setRs(s); E21->setRs(s); E22->setRs(s); E31->setRs(s); E32->setRs(s);}}
//------
(11) char Enigma::codeNext(char c){
(12) String DRH=moveD(Ds, 1);
(13) int* RT = new int[4];
(14) for(int i=1; i<=3; i++) RT[i]=(26+CTI(DRH[i])-CTI(Rs[i]))%26;
(15) return codeLetter(c,RT[1],RT[2],RT[3]);}
//------
(16) String Enigma::moveR(String ri, int k){
(17) String RI = ri, RIH="";
(18) for(int i=1; i<=k; i++){RIH=RI;
(19)   RI[3]=ITC((CTI(RI[3])+1)%26);
(20)   if(RIH[3]=='Z'){
(21)      RI[2]=ITC((CTI(RI[2])+1)%26);
(22)      if(RIH[2]=='Z')RI[1]=ITC((CTI(RI[1])+1)%26);}}
(23) return RI;}
```

7.4. Computational complexity

The total running time of the `Rings()` method for the established order of drums, by using a computer with an AMD Turion 64 X2 processor clocked at 1.9GHz, is about 5 seconds. The coding of letters (which is called 6 times for each of the 17576 ring settings) is the most time-consuming operation within this method. We have to repeat the method 6 times (i.e., for each possible order of 3 drums). The `Rings()` method returns several (0, 1, 2 or 3) ring settings. To guess which result is correct, we can call the `PlugBoard()` method for each of the listed ring settings. This method produces a result immediately (in a fraction of a second). Thus, we get a proper daily key (for 3 drums) after about 30 seconds. The cryptologists used 6 crypto bombs (one bomb for each order of drums), which were activated simultaneously. They obtained

the result after about 2 hours. When the German service added two additional drums, cryptologists needed 60 bombs. The running time of the `Rings()` method for 5 drums is about 300 seconds. We can see that a computer program gives a quicker result. H. Zygalski designed a better and cheaper method – perforated sheets. This method is completely independent from plug connections. An implementation of Zygalski's method is presented in full in [2].

8. The implications of the work and conclusions

We can solve the Enigma cipher (by analyzing and completing historic information) because trained Polish and (later) British cryptologists did it earlier. The Enigma cipher is not trivial and its breaking on the basis of eavesdropped messages (without the help of spies, mistakes of operators and numerous favorable coincidences) is practically impossible even nowadays. Zygalski's sheets method is more complicated but effective in each case. The reader can find the full algorithm in [2]. Both of these methods are interesting exercises and encourage the study of current problems of cryptology. If the reader has the required skills, one can try to build his own electronic variant of the Enigma-E machine.

Let us assume that we know the decryption method and we have a computer and then a cryptologist introduces a change. We can guess that this change can thwart all work of crypto analyzers. We can notice that breaking an Enigma key takes time, even today. Decryption algorithms always have huge computational complexity. The interesting history of the Enigma machine confirms the fact that cryptology will always be of great (commercial, diplomatic, and military) importance.

References

[1] Bauer F.L.: *Decrypted Secrets. Methods and Maxims of Cryptology.* Springer-Verlag Berlin Heidelberg, 2007.

[2] Borowska A.: The Cryptanalysis of the Enigma Cipher. *Advances in Computer Science Research*, vol. 10, pp. 19–38, 2013.

[3] Brynski M.: *Elements of the Galois Theory.* Alfa Publishing House, Warsaw, 1985.

[4] Buchmann J.A.: *Introduction to Cryptography.* PWN, Warsaw, 2006.

[5] Christensen C.: Polish Mathematicians Finding Patterns in Enigma Messages. *Mathematics Magazine*, pp. 247–273, 2007.

[6] Deavours C.A., Kruh L.: *Machine Cryptography and Modern Cryptanalysis.* Artech House Publishers, 1985.

[7] Garlinski J.: *Enigma. Mystery of the Second World War.* University of Maria Curie-Sklodowska Publishing House, Lublin, 1989.

[8] Gay K.: *The Enigma Cypher. The Method of Breaking.* Communication and Connection Publishing House, Warsaw, 1989.

[9] Grajek M.: *Enigma. Closer to the Truth.* REBIS Publishing House, Poznan, 2007.

[10] Gralewski L.: *Breaking of Enigma. History of Marian Rejewski.* Adam Marszalek Publishing House, Torun, 2005.
[11] Kahn D.: *Enigma Unwrapped.* In: New York Times Book Review, 29.XII.1974.
[12] Kasperek C., Woytak R.: Polish and British Methods of Solving Enigma. In: *Enigma: How the German Machine Cipher Was Broken and How It Was Read by the Allies in World War Two.* University Publications of America, 1984. Appendix F of Enigma by W. Kozaczuk.
[13] Katz J., Lindell Y.: *Introduction to Modern Cryptography.* Chapman and Hall/CRC, 2007.
[14] Koblitz N.: *Algebraic Aspects of Cryptography.* WNT, Warsaw, 2000.
[15] Koblitz N.: *A Course in Number Theory and Cryptography.* WNT, Warsaw, 2006.
[16] Kozaczuk W.: *How the German Machine Cipher Was Broken and How It Was Read by the Allies in World War Two.* University Publications of America, 1984. (Edited and translated by Christopher Kasparek).
[17] Kozaczuk W., Straszak J.: *Enigma. How the Poles Broke the Nazi Code.* Hippocrene Books, 2004.
[18] Lang S.: *Linear Algebra.* Springer-Verlag, New York, 1987.
[19] Menezes A.J., van Oorschot P.C., Vanstone S.A.: *Handbook of Applied Cryptography (Discrete Mathematics and Its Applications).* CRC Press, 1996.
[20] Mostowski A., Stark M.: *Elements of Higher Algebra.* PWN, Warsaw, 1970.
[21] Pieprzyk J., Hardjono T., Seberry J.: *Fundamentals of Computer Security.* Springer-Verlag Berlin Heidelberg, Germany, 2003.
[22] Rejewski M.: An Application of the Theory of Permutations in Breaking the Enigma Cipher. *Applications Mathematicae*, vol. 16(4), 1980.
[23] Rejewski M.: How did Polish Mathematicians Decipher the Enigma. *Polish Mathematics Association Yearbooks. Series 2nd: Mathematical News*, (23), 1980.
[24] Scott W.R.: *Group Theory.* Courier Dover Publications, 1964.
[25] Seberry J., Pieprzyk J.: *Cryptography: An Introduction to Computer Security.* Prentice-Hall, Sydney, 1988.
[26] Stoklosa J., Bilski T., Pankowski T.: *Data Security in Computer Systems.* PWN, Warsaw, 2001.
[27] Wythoff G.: The Invention of Wireless Cryptography. *The Appendix – Futures of the Past*, vol. 2(3), 2014.
[28] Zielinska E., Mazurczyk W., Szczypiorski K.: Trends in Steganography. *Communications of the ACM*, vol. 57(3), pp. 86–95, 2014.

Appendix A.
The construction of the M3 Enigma machine

The M3 Enigma machine is an electro-mechanical device. This machine consists of an alphabetical 26-letter keyboard, a lampboard (set of 26 lights), a *plugboard*, a set

of three rotating, encrypting discs (called *drums*) placed on a shared shaft, two fixed wheels: an *entry wheel* and a *reflector*, a battery and a *turning mechanism* (used to turn one, two or three drums after pressing any key).

Figure 1 presents a simplified diagram of how the Enigma works (a hypothetical electric circuit). Pressing any key (e.g., the key `W`) makes the closure of an electric circuit. After that, the current flows through the different components in the present configuration of the circuit. It starts from the battery (3) and flows through a connection under the pressed key (2) to the plugboard (1). Next, it flows through the entry wheel (E), via three drums (R, M and L) to the reflector (B). The reflector inverts the signal (but using a completely different route). From the reflector, the current passes through drums L, M and R to the entry wheel (E), to the plugboard (1) and finally to an appropriate lamp (4) (which represents a letter different from `W`), causing it to light up (cf. [9]).

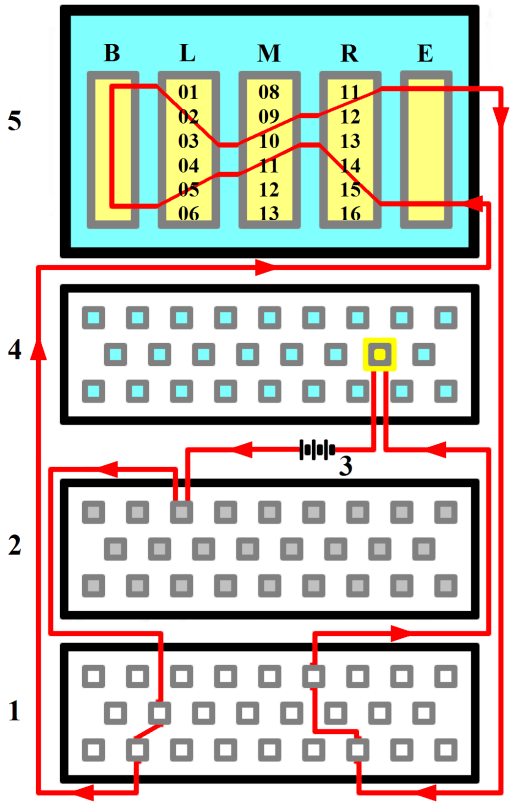

Figure 1. The diagram presents how the military Enigma machine works (1) the plugboard, (2) the keyboard, (3) the battery, (4) the lampboard, (5) disks: three drums (L, M, R), the entry wheel (E) and the reflector (B).

The German Land Forces and Air Force used five kinds of drums denoted by I, II, III, IV, V. Each *drum* is a disc with a diameter of approximately 10cm. Inside the drum, there is a second disc (called a *rotor*). On one side of each rotor, there are 26 brass spring pins, and on the other side, there are 26 flat electrical contacts. Both pins and contacts are arranged in a circle near the edge of the rotor and represent 26 letters of the alphabet. Each rotor hides 26 insulated wires. The wires connect the pins on one side to the contacts on the other in an established way (different for each type of drum) (cf. [10, 8]). As the drums are mounted side-by-side on the shaft, the pins of one drum touch the contacts of the neighboring one, forming 26 fragments of an electrical circuit (cf. [10]). Each drum has a metal rotating *ring* applied to the rotor. Engraved numbers (on this ring) correspond to the 26 letters of the alphabet. On the edge of the rotor, there is one distinguished place. The letter on the ring which is engraved opposite this position is treated as the *ring setting*. Individual kinds of cipher drums also differ by the so-called *turnover positions*. The turnover positions of the five kinds of drums were as follows I – `Q`, II – `E`, III – `V`, IV – `J`, V – `Z` (cf. [8]).

Pressing any key causes additionally one, two, or three cipher drums to turn one twenty-sixth of a full rotation (before the electrical circuit closure). More precisely, after pressing the key, the right drum turns 1/26 of a full rotation. When this drum reaches the *turnover position*, the middle drum turns 1/26 of a full rotation as well. When the second drum reaches the turnover position, the left and middle drums turn 1/26 of a full rotation (so-called "double step") (cf. [7, 10]). In this way, each letter is coded with the drums in different positions.

The position of each movable drum is given as a number (engraved on a ring) which can be seen through a window in the lid of the machine. The *rotor position* is defined as the difference between the position of the drum and the ring setting.

Connections of the *entry wheel* in the M3 Enigma machine are represented by the identity permutation (cf. [8]). The *reflector* (reversal drum) pairs the outputs of the last rotor, redirecting the current back through the drums using a different path.

In order to decode a text encrypted with the M3 Enigma machine, the receiver had to set up his Enigma in the same way as the sender had set up his during ciphering. Each military unit that used the Enigma was provided with the Enigma machine's initial settings in the form of tables of the *daily key settings*. Daily key settings (since September 15, 1938) consisted of the *wheel order* (i.e., the choice of drums and the order in which they were fitted), the *ring settings* and connections of the plugboard (cf. [7]). *Since September 15, 1938, the Germans* [...] *changed the manner of announcing message settings. Starting with this date, the operator was forced to choose his own arbitrary three letters, which he placed in the headline of the message without ciphering* [these letters played a role of the initial drum settings]. *Next, he set the drums to these letters and chose three other letters as the message settings. These letters* [...] *after two-time coding, were placed at the beginning of the message and then the drums were set to the message settings and the actual ciphering of the message began* (cf. [23]).

The security of the Enigma cipher depends on its large key space. The full size of the key space equals the product of the following values: the number of possible drum orders (N_{Do}), the number of possible ring settings (N_{Rs}), the number of possible initial drum settings (N_{Ds}) and the number of possible plugboard settings (N_{Ps}), where:

$N_{Do} = \dfrac{k!}{(k-3)!}$ for $k = 3$, $N_{Do} = 6$ and for $k = 5$, $N_{Do} = 60$, where k is the number of drums,

$N_{Ds} = 26^3 = 17576$ Since we have to place one of the 26 letters on each of the three positions[2] (until September 15, 1938).

$N_{Rs} = 26^3 = 17576$ We establish a ring setting for each of 3 drums.

$N_{Ps} = \dfrac{\binom{26}{2n}\binom{2n}{n}}{2^n} n!$ where n is the number of plug connections (cf. [8])

For instance:

- From 1.10.1936 to 15.09.1938 $k = 3$ and $n = 5 - 8$,
  e.g., for $n = 8$ we have $6 \times 17576 \times 17576 \times 10,767,019,638,375$ possible keys.
- From 15.09.1938 to 15.12.1938 $k = 3$ and $n = 5 - 8$,
  e.g., for $n = 8$ we have $6 \times 17576 \times 10,767,019,638,375$ possible keys.
- From 15.12.1938 to 1.1.1939 $k = 5$ and $n = 5 - 8$,
  e.g., for $n = 8$ we have $60 \times 17576 \times 10,767,019,638,375$ possible keys.
- From 1.1.1939 to 9.1939 $k = 5$ and $n = 7 - 10$,
  e.g., for $n = 10$ we have $60 \times 17576 \times 150,738,274,937,250$ possible keys.

The fact that the Allies read German messages was ultra-secret information (*the biggest secret of the Second World War after the atomic bomb* (cf. [11])). Nearly until the end of the war, the German command-in-chief treated the Enigma cipher as absolute. On the basis of opinions of experts from special committees, they precluded any effective decryption. They were aware that, in theory, the Enigma cipher could be broken. Therefore, from time to time, they introduced changes in the way of generating messages and the construction of the Enigma machine. For example, on December 15, 1938, two additional drums were added; on September 15, 1939, German cryptologists changed the manner of announcing message settings. They introduced these changes in order to foil any attempts of decryption (cf. [8])

Appendix B. The work of Polish mathematicians

Three Polish mathematicians (M. Rejewski, H. Zygalski and J. Różycki) broke the German Enigma at the beginning of 1933. In the years 1933–1939, their daily duties in the Cipher Bureau (the B.S.-4) consisted in breaking daily keys for individual Enigma networks, detecting changes in the manners of generating messages and changes in the

[2]The set of all initial drum settings consists of 26×26×26 possible settings. However, the full rotation of three drums gives 26×25×26 possible settings (due to the *double step* on the middle drum).

Enigma machine construction as well as designing new ways of breaking daily keys. They gave the results of their work to the chiefs of Cipher Bureau, who decided on the importance and the order of reading messages. Since the Enigma ciphers a text by means of involutions, in order to decode a message, decoders had to set up their Enigma in the same way as the sender had set up his while ciphering. Messages were read by initiated operators. To that end, the AVA factory made 17 Enigma doubles in the years 1933–1939. In 1938, an experiment was carried out. Its result was that a 10-person team (consisting of cryptologists and operators) of the B.S.-4 read 75% of intercepted Enigma messages. On September 15, 1938, German cryptologists changed the manner of announcing message settings. The decryption method (based on characteristics catalogues) became useless. In October 1938, M. Rejewski designed the cryptologic bomb method. The bombs returned a result within 2 hours. Coders activated 6 machines simultaneously (one bomb for each order of 3 drums). On December 15, 1938, the German service increased the number of drums from 3 to 5. That is, in the machine there were still only three drums on the shaft, but from then on the three were chosen from a set of five. M. Rejewski quickly discovered the connections of two new drums. Breaking daily keys by means of the bomb method was no longer effective, as operators needed 60 bombs instead of 6. On January 1, 1939, the German service increased the number of pairs of letters changed by the plugboard. From then on, they applied from 7 to 10 connections. As Rejewski constructed the crypto bomb, at the same time, another decryption method (Zygalski's sheets method) was designed in the B.S.-4. The cryptologists needed to make 6 sets of perforated sheets (one set for each order of 3 drums). Each set consisted of 26 sheets. They had to cut out (with a razor blade) about a thousand perforations in each sheet of paper. They only made 2 of 6 sets because of the lack of initiated staff. In July 1939, Polish cryptologists met British and French cryptologists. Poles revealed all of their knowledge and abilities to read Enigma. After the outbreak of WWII (September 1, 1939), some of the staff of the Cipher Bureau were evacuated to Romania and then to France. The French-Polish radio-intelligence unit "Bruno" (about 70 people) was stationed about 40 kilometers from Paris. All materials, equipment, and machines (with the exception of two Enigmas) had been destroyed before leaving Poland. The British provided the unit with the whole collection of 60 sets of 26 perforated Zygalski sheets (for 5 drums). Since the British Cipher Bureau had more resources at that time, out of every 100 keys that were recovered, 83 came from the British and 17 came from the Poles. In May 1940, German cryptologists again radically changed the mechanism of using Enigma. French general G. Bertrand wrote: *Superhuman efforts and incessant work day and night were needed in order to overcome this new obstacle: on 20 May* [thanks to Polish cryptologists] *decryption was restarted* (cf. [8]). In June 1940 France signed a truce with Germany. On 24 June, the staff of "Bruno" (15 Poles and 7 Spaniards) were transported to Algeria. From Africa, the cryptologists returned to France where, in October 1940, they began work in the unit "Cadix" (32 employees), which stayed in contact with London. There, Polish cryptologists solved ciphers different from the Enigma cipher. The British cryptologists dealt with Enigma.

Affiliations

Anna Borowska
Faculty of Computer Sciences, Bialystok University of Technology, Bialystok, Poland, a.borowska@pb.edu.pl

Elżbieta Rzeszutko
Faculty of Electronics and Information Technology, Warsaw University of Technology, Warsaw, Poland, E.Rzeszutko@tele.pw.edu.pl

Received: 12.08.2014
Revised: 8.11.2014
Accepted: 11.11.2014