THE CRYPTANALYSIS OF THE ENIGMA CIPHER

Anna Borowska

Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: In this paper we study cryptanalysis of the military Enigma machine which was used by the German Army during the Second World War. We are interested in the problem of decoding secret messages transmitted after 15 September 1938. We give a complete algorithm which can be used to: generate the ring settings, guess what kinds of drums are chosen and determine the order of the drums on a shared shaft. The proposed algorithm is an optimization of the Zygalski's sheets method. Before we present it, we will describe the mystery which is hidden in the sheets (author's observations). In order to read the encrypted messages we need the plugboard settings. Connections of the plugboard influence neither Zygalski's method (which is a well-known fact) nor the presented algorithm. The missing (original) algorithm solving the problem of the plugboard along with an algebraic description will appear very soon.

Keywords: Zygalski's sheets method, Enigma machine, ring settings, message settings

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. The first intercepted German text, encrypted by using the Enigma, was broken by M. Rejewski in 1932. Since then the Polish cryptologists (M. Rejewski, J. Rozycki and H. Zygalski) systematically worked on: decoding ciphers, constantly modified manners of generating secret messages and modernized constructions of Enigma machines.

The algorithm presented below can be used to decode messages transmitted after 15 September 1938. That day the Germans withdrew the *initial drum settings* from tables of *daily key settings*. We assume (according to a knowledge of that time) that we know connections of all kinds of drums.

The algorithm relies on Zygalski's sheets. We propose an implementation of this method in cpp language. But we stress the point that no source (mainly historical

Advances in Computer Science Research, vol. 10, pp. 19-38, 2013.

sources) describes this method precisely. Historians make a lot of mistakes and they do not describe all the elements of this method. In this case accuracy is essential. Therefore, the given algorithm is a result of many tests in order to get the *initial ring settings*. The cryptologists might have done it in a different way during World War II.

The proposed algorithm was assembled on the basis of different facts which had been found in literature. These facts were completed with author's observations and ideas. The facts that are described clearly in literature are denoted in this paper by $(|_{[1!]})$. The other facts are presented in various ways in different books. To get the total algorithm we tested different possibilities and we chose the ones, which give a proper result. We denoted this ambiguously described information by $(|_{[2!]})$. We also used the facts which are described very vaguely in literature (non-concrete facts) and we had to complete them with our ideas (these ideas are denoted by $(|_{[3!]})$. Historians do not describe all the elements of the Zygalski's method. Thus, some facts are presented in this paper as author's observations. These facts and the author's own ideas are denoted by $(|_{[4!]})$.

We optimize the Zygalski's method. We browse fields which represent permutations on a first sheet only, more precisely, on its specified fragment. The fields which represent permutations with 1-cycles are remembered in a list. Next they are processed until the program returns a proper result. In section 7.6 we propose an additional improvement concerning fields which lie in an interfered area. We suppose, these fields gave the cryptologists a hard time when they were guessing the initial ring settings.

The Germans used different kinds of Enigma machines (also commercial), modified their constructions and changed the manner of generating secret messages. Therefore we describe the construction and the parameters of the machine for which we do cryptanalysis. We are interested in the M3 Enigma machine. Sections 2, 3 and 4 make up a brief survey of information taken from publications [4], [8], [5], [3], [6]. In section 5 we give some facts about 1-cycles which are essential to understand a presented method. Section 6 contains a description of sheets (author's observations). We received these observations with the help of a lot of tests. In section 7 we give an actual algorithm. By means of this algorithm we can generate the initial ring settings and guess the order of drums on the basis of a given set of messages intercepted after 15 September 1938.

A lot of facts which we present here must have been known by cryptologists during World War II. Since they did not disclose them, we had to study the secret of the Enigma machine from the beginning.

2. The construction of the military Enigma machine

The M3 Enigma machine is a combination of electrical and mechanical subsystems. This machine consists of an alphabetical 26-letter keyboard, a lampboard (set of 26 lights), a set of three rotating, encrypting disks (called *drums*) placed on a shared shaft, two fixed wheels: an *entry wheel* and a *reflector*, a *plugboard*, a battery, and a turning mechanism (used to turn one, two or three drums after pressing any key).

Figure 1 shows a simplified diagram of how the Enigma works. Pressing any key (e.g. the key `E`) causes the closure of an electric circuit. Then current flows through the various 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 an entirely 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 `E`), causing it to light up (cf. [5], p.236).

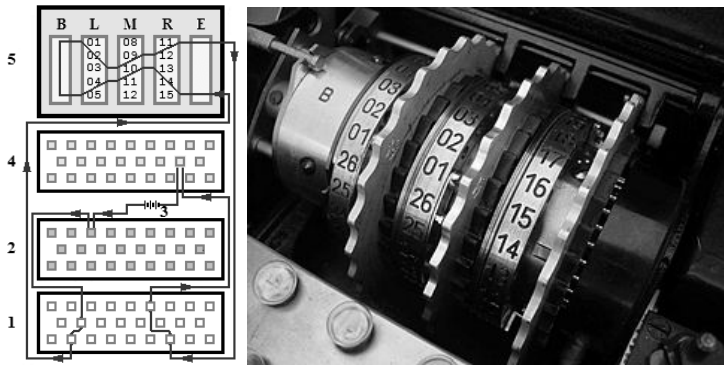

Fig. 1. I. The diagram presents how the military Enigma machine works (a hypothetical electric circuit). (1) the plugboard, (2) the keyboard, (3) the battery, (4) the lampboard, (5) disks: three drums (L, M, R), the entry wheel (E) and the reflector (B) (cf. [9]).
II. Assembled set of disks (three movable drums between the entry wheel and the reflector) (cf. [9]).

Each *drum* is in the shape of a wheel. Inside the drum there is a disk (called a *rotor*). On one side of each rotor there are 26 brass spring pins (arranged in a circle near the edge of the rotor). On the other side there is a corresponding number of flat electrical contacts. 26 contacts (and 26 pins), on either side, 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 manner (different for various kinds of drums) (cf. [6], [4]). Since the drums are mounted side-by-side on the shaft, the

pins of one drum touch the contacts of the neighbouring one, forming 26 fragments of an electrical circuit (cf. [6]). 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* (cf. [4]). Individual kinds of cipher drums differ not only by connections of pins and contacts but also by the so-called *turnover positions*. The turnover positions of five kinds of drums (denoted by I, II, III, IV, V) used by the German Land Forces and Air Force were as follows I - Q, II - E, III - V, IV - J, V - Z (cf. [4]).

Pressing any key of the Enigma causes one, two or three cipher drums to turn one twenty-sixth of a full rotation (before the electrical circuit closes). Thus the signal path changes constantly. 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, too. When the second drum reaches the turnover position, the left drum turns $1/26$ of a full rotation (cf. [3], [6]). In this way each letter is coded with the drums in different positions.

The position of each movable drum is characterized by a number (engraved on a ring) which we can see through a window in the lid of the machine. The *rotor position* is described as the difference between the ring setting and the position of the drum.

Connections of the *entry wheel* in the military Enigma are represented by the identity permutation (cf. [4]). The *reflector* (reversal drum) connects the outputs of the last rotor into pairs, redirecting the current back through the drums using a different path. The reflector does not move (cf. [6]).

The *plugboard* contains 26 plugs and sockets (one pair for each letter of the alphabet). If the operator connects, by means of a cable, a plug of one pair with a socket of the second one (and inversely), then two letters are swapped both before and after the signal flows through the rotors. If a plug and a socket of a given letter are not connected with a cable to a plug and a secket of another letter, they are internally connected with each other (cf. [3], [4]).

In order to decode a text encrypted with the Enigma, the receiver had to set up his Enigma in the same way as the sender set up his during ciphering. The Enigma machine's initial settings were delivered to each military unit which used the Enigma, in the form of tables of the *daily key settings*. Daily key settings (until 15 September 1938) consisted of the *wheel order* (i.e. the choice of drums and the order in which they were fitted), the *ring settings*, plug connections (i.e. connections of the plugs in the plugboard) and initial positions of the drums (cf. [3]). [*Since 15 September 1938 the Germans, neither changed nor added anything to the machine, but 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 positions of the drums). *Next, he set the drums to these letters and chose three other letters as the message settings. These letters, as before, 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. [8]).

3. Mathematical analysis

Let **P** = {A, B, C,..., Z} be a set of possible plaintexts, and let **C** = **P** be a set of possible ciphertexts. The military Enigma machine codes text T by using a poly-alphabetic substitution cipher. Each letter $p \in \mathbf{P}$ of the message is transformed according to the following permutation (cf. [8])

$$\Lambda = SH(Q^zRQ^{-z})(Q^yMQ^{-y})(Q^xLQ^{-x})B(Q^xL^{-1}Q^{-x})(Q^yM^{-1}Q^{-y})(Q^zR^{-1}Q^{-z})H^{-1}S^{-1} \tag{1}$$

S – is a permutation describing the plugboard transformation,
B – is a permutation describing the reflector transformation, ($B = B^{-1}$),
L, M, R – are permutations describing transformations of the three cipher drums,
H – is a transformation of the entry wheel (H is an identity permutation),
Q – is a cyclic permutation mapping A to B, B to C to D, ..., Z to A,
x, y, z – are the positions of rotors before pressing any key (values from set **IP** = $\{0,1,\ldots,25\}$),
$x = (n(\alpha) - n(\delta))\%26$ for the left rotor,
$y = (n(\beta) - n(\varepsilon))\%26$ for the middle rotor,
$z = (n(\gamma) - n(\zeta))\%26$ for the right rotor (cf. [4]),
α, β, γ – positions of drums (left, middle, right) before pressing any key ($\alpha, \beta, \gamma \in \mathbf{P}$)
δ, ε, ζ – positions of rings (left, middle and right) (values from set **P**)
$n(\alpha)$ – the number of a letter α (value from set **IP**)

The Enigma codes the first 6 letters (meaning the double coded message settings) using the following permutations (cf. [8])

$$A = SH(Q^{z+1}RQ^{-(z+1)}Q^yMQ^{-y}Q^xLQ^{-x})B(Q^xL^{-1}Q^{-x}Q^yM^{-1}Q^{-y}Q^{z+1}R^{-1}Q^{-(z+1)})H^{-1}S^{-1}$$
$$B = SH(Q^{z+2}RQ^{-(z+2)}Q^yMQ^{-y}Q^xLQ^{-x})B(Q^xL^{-1}Q^{-x}Q^yM^{-1}Q^{-y}Q^{z+2}R^{-1}Q^{-(z+2)})H^{-1}S^{-1}$$
$$C = SH(Q^{z+3}RQ^{-(z+3)}Q^yMQ^{-y}Q^xLQ^{-x})B(Q^xL^{-1}Q^{-x}Q^yM^{-1}Q^{-y}Q^{z+3}R^{-1}Q^{-(z+3)})H^{-1}S^{-1}$$
$$D = SH(Q^{z+4}RQ^{-(z+4)}Q^yMQ^{-y}Q^xLQ^{-x})B(Q^xL^{-1}Q^{-x}Q^yM^{-1}Q^{-y}Q^{z+4}R^{-1}Q^{-(z+4)})H^{-1}S^{-1}$$
$$E = SH(Q^{z+5}RQ^{-(z+5)}Q^yMQ^{-y}Q^xLQ^{-x})B(Q^xL^{-1}Q^{-x}Q^yM^{-1}Q^{-y}Q^{z+5}R^{-1}Q^{-(z+5)})H^{-1}S^{-1}$$
$$F = SH(Q^{z+6}RQ^{-(z+6)}Q^yMQ^{-y}Q^xLQ^{-x})B(Q^xL^{-1}Q^{-x}Q^yM^{-1}Q^{-y}Q^{z+6}R^{-1}Q^{-(z+6)})H^{-1}S^{-1}$$

We calculate a product AD in the same way as M. Rejewski in [8].

4. Cryptanalysis (Zygalski's sheets)

Zygalski's sheets were a cryptologic technique used to decrypt messages ciphered on the German Enigma machines after 15 September 1938. This method was applied in order to guess which (three out of five) drums were put on the shaft, to calculate the order of the drums and to determine the ring settings (cf. [5]). Three three-letter fragments of the message were used for analysis, i.e. initial positions of the drums and double ciphered message settings (cf. [6]).

(1) ZXV ZBB GIL (2) AFV AFB LFF (3) FUA RUF CHJ (4) QDV LXU WQH

Cryptologists took advantage of the fact that identical letters on positions one and four (or two and five or three and six) in the double ciphered message settings mean that a permutation *AD*, (or *BE* or *CF*) contains 1-cycles (named *females*). They created a catalogue of all products *AD* (*BE*, *CF*) in which 1-cycles appear. Next they compared these products with products (containing 1-cycles) which were generated for initial drum positions of messages eavesdropped the same day (cf. [6]).

For each possible order[1] of drums they made a set of 3×26 sheets (one sheet for each letter of the alphabet, for each product *AD*, *BE*, *CF*) (cf. [5]). Each sheet contained a square divided into 51×51 small fields. On the top (and on the bottom) of the square the letters A−Z, A−Y were written down. These letters meant possible positions of the middle drum (cf. [4] $\mathrm{I}_{[2!]}$). Similarly the sides of the square were described. These letters meant possible positions of the right drum (cf. [4] $\mathrm{I}_{[2!]}$). Each field $[c][r]$ of the square represented a product *AD* (*BE* or *CF*). If the product contained 1-cycles, the field was perforated. [*When the sheets were superposed and moved in the proper sequence and the proper manner with respect to each other, in accordance with a strictly defined program, the number of visible apertures gradually decreased. And, if a sufficient quantity of data was available, then finally remained a single aperture, probably corresponding to the right case, that is, to the solution. . . . From the position of the aperture one could calculate the order of the drums and the setting of their rings*] (cf. [8]).

5. 1-cycles

Let us assume that we have at our disposal the following set of messages eavesdropped during the same day (i.e. received for the same daily key settings).

(1) IHE BHX BZR (2) ZHH OAR OGI (3) ADU UXM JXS (4) EDL PCW ZMW

[1] At the beginning Germans applied 3 kinds of drums. We have $3*2*1=6$ ordered sequences of 3 distinct elements of the set **D** = {I, II, III}. Then they added two extra kinds of drums and a number of orders increased to $5*4*3=60$ (cf. [5]).

We can obtain each of the permutations A, B, C, D, E, F (with which the three-letter message settings are double ciphered) in the following way. We set up our Enigma machine in an identical way as the coder set his machine up during ciphering. Next we press in order all the letters of the alphabet without using the turning mechanism (cf. [6]). In 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:	B	A	H	U	L	W	R	C	Y	N	Z	E	O	J	M	T	V	G	X	P	D	Q	F	S	I	K
D:	B	A	W	X	U	Q	V	O	K	P	I	T	Z	Y	H	J	F	S	R	L	E	G	C	D	N	M
AD:	A	B	O	E	T	C	S	W	N	Y	M	U	H	P	Z	L	G	V	D	J	X	F	Q	R	K	I

Permutations A, D consist of 13 transpositions. On the base of the double ciphered message settings BHX BZR we can conclude that both A and D permutations contain a transposition (x,B), where x is a ciphered letter (different from B). In the case of the message (1) that is the transposition (A,B). Therefore the product AD assigns the letter B to the letter B and the letter A to the letter A. Thus there are females (1-cycles) in the permutation AD. Let us write down the permutation AD as a product of disjoint cycles

AD: (A)(B)(COZINPLUXRVF)(DETJYKMHWQGS)

We can see that the first letter of the message settings was the letter A. For the message (3) we calculate the product BE, because the repeated letter X occurs on the positions two and five. Below we present the permutation BE as a product of disjoint cycles

BE: (AGPQWUT)(BNK)(CJI)(DFESROL)(HY)(M)(VZ)(X)

We can guess that the second letter of the message settings was the letter M.

6. The sheets (author's observations)

Observations presented below we received for real connections of both drums and plugboard, which were used by Wehrmacht. Given examples were executed for drums: L = I, M = II, R = III, for the reflector B = UKW B and for the plugboard connections: $S = (\mathrm{A},\mathrm{G})(\mathrm{C},\mathrm{F})(\mathrm{K},\mathrm{O})(\mathrm{L},\mathrm{Y})(\mathrm{R},\mathrm{W})(\mathrm{S},\mathrm{Z})$. But one can generalize obtained facts. The sheets were generated for the products AD, but we shall show that one can observe similar results for permutations BE and CF (author's observations). 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. [4])

	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
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
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
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

The turnover positions for selected drums: I - Q, II - E , III - V (cf. [4])

The reflector connections (cf. [4]):
UKW B: (AY)(BR)(CU)(DH)(EQ)(FS)(GL)(IP)(JX)(KN)(MO)(TZ)(VW)

In order to present the mystery hidden in the sheets, we represent them by means of two-dimensional matrices. We number them horizontally and vertically from 0 to 25 (for simplification). In each of the 60 sets (we can place 5 drums on 3 positions in $60 = 5*4*3$ manners) there are 26 sheets (one sheet for each letter). Let us denote by $[s][c][r]$ a square field $[c][r]$ on a sheet number s, where c is a number of a column and r is a number of a row ($|_{[2!]}$). The value s represents a number (or a letter) on the left drum, c is a number (or a letter) on the middle drum and r - a number (or a letter) on the right drum. Each field on any sheet corresponds to a product of permutations A and D and is coloured in black (for permutation AD with 1-cycles) or white (for permutation AD without females).

Example 6.1 Let us assume that we have the ring settings BBB. The field [B][A][A] represents a product AD for the drum settings BAA. This field is coloured in white, because a permutation AD does not contain any females. However, the field [B][B][A] represents a product AD for the drum settings BBA and it is coloured in black because AD contains 1-cycles.
AD: (ABZLQNIUHXY)(CEKFDPTGWJV)(MR)(OS) (for the drum settings BAA)
AD: (AZBFDNMXR)(CHG)(EYPILWOKJ)(Q)(STU)(V) (for the drum settings BBA)

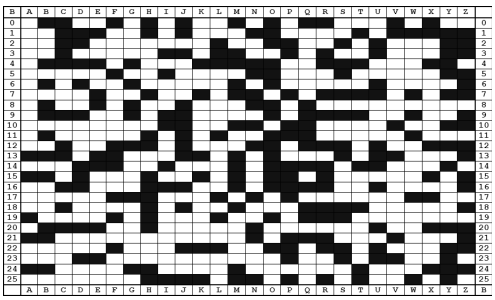

Fig. 2. The sheet B for the ring settings BBB.

Let us fix the meanings of the two phrases. By the *initial ring settings* (in short: IRS) we understand these ring settings for which the set of messages was ciphered. By the *starting ring settings* (in short: SRS) we define these ring settings ($|_{[3!]}$) for which we do calculations in order to obtain the initial ring settings.

The tests confirmed that it does not matter what set of sheets (i.e. for what SRS) we use to determine the initial ring settings ($|_{[3!]}$). In each set there are sheets which contain identical (with some exceptions defined below) appropriately moved repre-

sentations of permutations AD with females ($|_{[4!]}$). For instance the sheet `B` from the set for SRS = `BBB` corresponds to the sheet `A` from the set for SRS = `AAA`.

Figure 3 shows that if we move the sheet `A` down one row and one column to the right then we obtain the sheet `B`. The discrepancies on each sheet (marked with grey colour) are caused by the turnover position (in short tp) of a right drum ($|_{[4!]}$). Since the order of drums was the same the whole day, the discrepancies on each sheet appeared in the same rows (i.e. in $(r-i)$-th rows, where $i = 0,1,2,3$, $tp_R = r$ denotes a turnover position of the right drum) ($|_{[4!]}$). Because for the drum III tp = `V`, so in the picture 3 we can see discrepancies in the rows `V`$-\{0,1,2,3\}$.

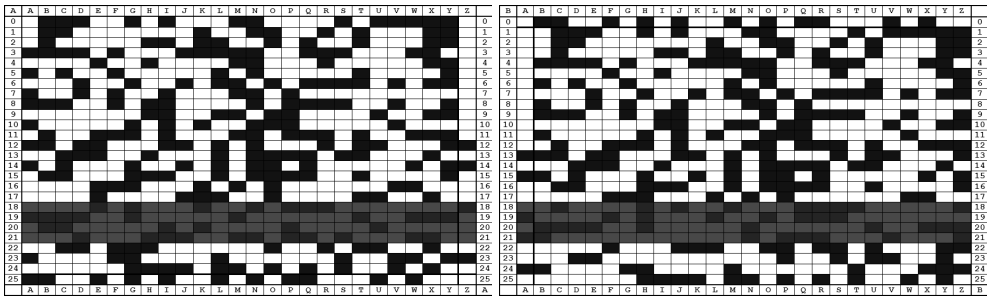

Fig. 3. The sheets `A` and `B` for SRS = `AAA` and SRS = `BBB` (appropriately).

Below we give three methods of the `Sheets` class. Let x be any sheet (from the set for SRS = $X_1X_2X_3$). The method `compareSh()` determines a sheet y (from the set for SRS = $Y_1Y_2Y_3$) which corresponds to x (cf. lines $13-15$). We used the following formulas

$y = (x+Y_1-X_1)\%26$ – the sheet (from the set for SRS = $Y_1Y_2Y_3$) which corresponds to the sheet x (from the set for SRS = $X_1X_2X_3$) ($|_{[4!]}$),

$c = (X_2-Y_2)\%26$ – it is necessary to move the sheet y to the right by c columns relative to the sheet x ($|_{[4!]}$),

$r = (X_3-Y_3)\%26$ – it is necessary to move the sheet y down by r rows relative to the sheet x ($|_{[4!]}$).

The method `compareSh()` additionally checks (for confirmation) that fields with females agree (after suitable shift) on both sheets. This method omits the fields from the interfered areas. Checking that the field $[j][i]$ on a sheet x or y (a sheet y is moved right c columns and down r rows relative to a sheet x) is in the interfered area takes place in lines $(20-22)$. The method `pushSh()` generates a string `"(j,i)"` for a product AD with 1-cycles and an empty string `""` for a product without females. The method `ITC()` exchanges a number from the set $\mathbf{IP} = \{0,1,\ldots,25\}$ for a suitable letter. The method `CTI()` does an opposite operation. `HE` is an object of the `Enigma` class. By tpR we signify the turnover position for the right drum. `moveD()` shifts drum settings `dr` for k presses of keys. `createAorD()` generates a permutation Λ (cf. the formula (1), section 3) for the

ring settings SRS = `rs` and for the drum settings `dr`. If a permutation *AD* contains any females the method `Fem()` returns `true`. `codeLetter()` ciphers a letter described by a first parameter for the rotor settings determined by three next parameters.

```
( 1) String Sheets::pushSh(String rs, char x, int i, int j){
( 2) Perm *A1, *D1, *AD;
( 3) String dr="", s="", crr="";
( 4) crr=IntToStr(j)+","+IntToStr(i);
( 5) dr=x; dr+=ITC(j%26); dr+=ITC(i%26);
( 6) A1 = createAorD(rs,HE->moveD(dr,1));
( 7) D1 = createAorD(rs,HE->moveD(dr,4));
( 8) AD = A1->Prod(D1);
( 9) if(AD->Fem()){PF.push_back(crr); s="("+crr+")";}
(10) delete A1; delete D1; delete AD;
(11) return s;}
//----
(12) bool Sheets::compareSh(String rs1, String rs2, char x){
(13) char y = ITC((26+CTI(x)+CTI(rs2[1])-CTI(rs1[1]))%26);
(14) int c = (26+CTI(rs1[2])-CTI(rs2[2]))%26;
(15) int r = (26+CTI(rs1[3])-CTI(rs2[3]))%26;
(16) cout « rs1 « "," « rs2 « "," « x « "," «y « "," « IntToStr(c) « "," «IntToStr(r);
(17) String s1="", s2=""; bool b=true, b1, b2;
(18) for(int i=0; i<=25; i++)
(19)   for(int j=0; j<=25; j++){
(20)       b1=cM(i,r,0)||cM(i,r,1)||cM(i,r,2)||cM(i,r,3);
(21)       b2=cM(i,0,0)||cM(i,0,1)||cM(i,0,2)||cM(i,0,3);
(22)       if(!(b1||b2)){
(23)          s1=pushSh(rs1,x,(26+i+r)%26,(26+j+c)%26);
(24)          s2=pushSh(rs2,y,i,j);
(25)          if((s1==""&&s2!="")||(s1!=""&&s2==""))b=false;}}
(26) return b;}
//----
(27) Perm* Sheets::createAorD(String rs, String dr){
(28) HE->setEnigma(rs,dr);
(29) char* A = new char[27]; A[0]=26;
(30) for(int i=1; i<=26; i++){
(31)   A[i]=HE->codeLetter(pH[i],HE->Rt[1],HE->Rt[2],HE->Rt[3]);}
(32) return new Perm(A);}
//----
(33) String Enigma::moveD(String dr, int k){
(34) String DR = dr, DRH="";
(35) for(int i=1; i<=k; i++){DRH=DR;
(36)   DR[3]=ITC((CTI(DR[3])+1)%26);
(37)   if(DRH[3]==tpR){
(38)       DR[2]=ITC((CTI(DR[2])+1)%26);
(39)       if(DRH[2]==tpM)DR[1]=ITC((CTI(DR[1])+1)%26);}}
(39)return DR;}
```

Example 6.2 Below we present a more complicated example. Let $X_1X_2X_3$ = `ZAE`, $Y_1Y_2Y_3$ = `CIW`, x = `A`. For the products *AD* the algorithm determined the following values y = `D`, $c = 18$, $r = 8$. It means that the sheet `D` (for SRS = `CIW`) is moved right 18 columns and down 8 rows relative to the sheet `A` (for SRS = `ZAE`) (cf. Fig. 4).

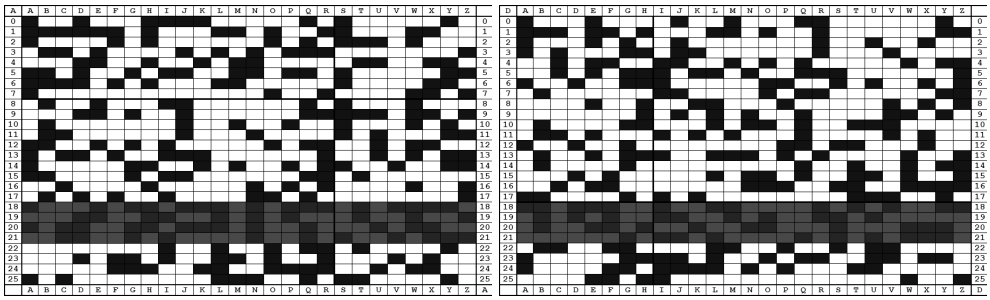

Fig. 4. The sheet A (for SRS = ZAE) and the sheet D (for SRS = CIW).

The author observed that in the case of the products *BE* and *CF* the discrepancies on sheets (caused by the turnover position of the right drum) take place in $(r-i)$-th rows, where $i = 0,1,2,3,4$ (for the products *BE*) and $i = 0,1,2,3,4,5$ (for the products *CF*) ($|_{[4!]}$). We can observe this fact in Figures 5, 6.

Example 6.3 Let $X_1X_2X_3 =$ ZAE , $Y_1Y_2Y_3 =$ CIW , $x =$ A.

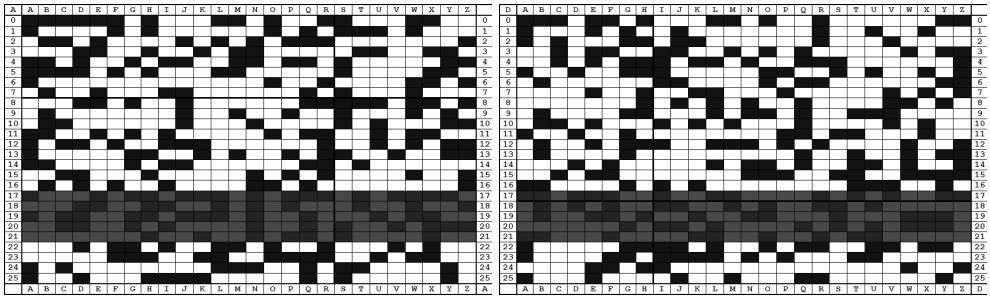

Fig. 5. For the products *BE* the algorithm determined the following values $y =$ D, $c = 18$, $r = 8$.

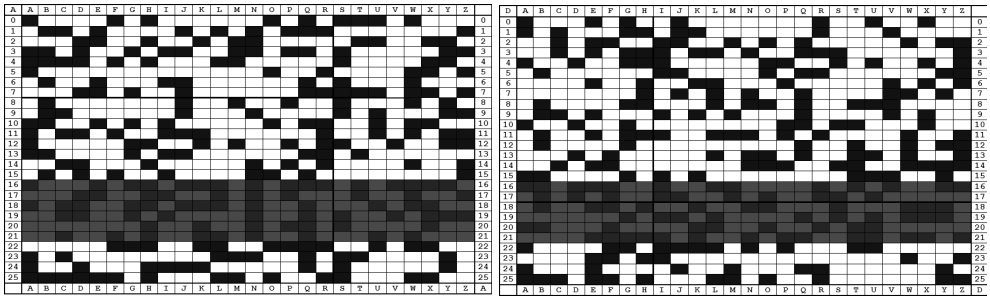

Fig. 6. For the products *CF* the algorithm determined the following values $y =$ D, $c = 18$, $r = 8$.

7. The cryptanalysis (the optimized algorithm)

The algorithm presented below relies on the Zygalski's sheets method.

7.1 Initial activities

A cryptologist chooses 3 out of 5 drums and places them on the shaft in a selected order. We used the drums L = I, M = II, R = III for obtaining results presented in this paper. Next he establishes his own ring settings ($|_{[2!]}$), for example SRS = AAA. He eliminates messages in which initial positions of the drums cause a shift of the middle drum ($|_{[2!]}$) and he loads to the list M maximum 26^2 messages (i.e. 3 three-letter groups: the initial drum settings and double ciphered message settings). It is not necessary to load them in alphabetical order. It is possible to use several messages with the same first letter in the initial drum settings[3] ($|_{[4!]}$). We do not need to analyse all 26 messages. We usually get a result after calculations for $10-20$ messages.

7.2 The manner of moving the sheets (optimization - author's ideas)

The manner of moving the sheets given in literature (mainly by historians) is not precise enough. Moreover, we can find several different versions for this operation. Thus, we are not sure if the algorithm presented below does calculations in the exact same way as the cryptologists did it during World War II. We obtained the manner as a result of many checks of different hints which we had found.

Table 1. The set I (messages which we used in experiments)

(0) ABH YHJ YEU	(6) GBH QAC QIS	(12) MDH XLI XQI	(18) SDH UDK UTO	(24) YHB UPH UPN
(1) BJW ESX EQV	(7) HBE VNS VSZ	(13) NCF HKE HUL	(19) TDD QHQ QYN	(25) ZHH DMH DIM
(2) CDA LNZ LBJ	(8) IHE BHX BZR	(14) OFE RFI RVG	(20) UII RUN RHA	
(3) DBD YSM YKG	(9) JIA LFK LQI	(15) PEE UTI UZN	(21) VAH AWL AIR	
(4) EEE DNQ DDK	(10) KFF HCM HZS	(16) QHC DRH DOL	(22) WCF OFG OFV	
(5) FEC WMU WLZ	(11) LIC UBL UUW	(17) RGA VVP VFL	(23) XID DON DCP	

First we choose a sheet for a message (0). Let Z be this sheet. Next we put the sheet A (this sheet corresponds to the message (1)) on the sheet Z. We move the sheet A right c columns and down r rows relative to the sheet Z, where (cf. [6] $|_{[2!]}$)

$$c = (\mathrm{B}-\mathrm{J})\%26 = 25, \qquad r = (\mathrm{H}-\mathrm{W})\%26 = 24.$$

Next we move remaining sheets relative to the sheet (0) in an analogous manner, i.e., according to the following formulas ($|_{[2!]}$)

$c = (D_M^{(0)} - D_M^{(i)})\%26$ // $D_M^{(i)}$ - a setting of a drum M for a message (i),
$r = (D_R^{(0)} - D_R^{(i)})\%26$ // $D_R^{(i)}$ - a setting of a drum R for a message (i).

[2] The value 26 was established for the sake of historical facts. In every set, during World War II, there was only one sheet for each letter. The program will accept any number of messages.
[3] Cryptologists were not able to do it because they had only one sheet for each letter in each set.

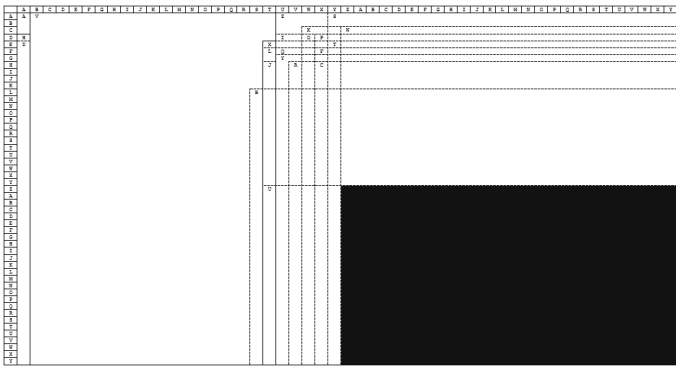

Fig. 7. The manner of moving the sheets.

In the above picture each sheet is signified by means of the first letter of the initial drum settings. We cannot see any of them because they are under other sheets. The rectangle coloured in black shows the area which is common for all used sheets. Within this rectangle we look for the representations of permutations *AD* with females. We are not interested in the remaining fragments of the sheets, that is we do not do any calculations for fields outside the rectangle ($|_{[4!]}$). We avoid the time-consuming calculations. The optimization of Zygalski's method relies on this idea. The method `rectangle()` of the `Sheets` class determines an area of a rectangle and writes down a result in the table `Rec[]` as follows. `Rec[1]` - the first column, `Rec[2]` - the last column, `Rec[3]` - the first row, `Rec[4]` - the last row. String `s1` is a field of an object of the `Message` class and represents the initial drum settings of a message (for the message (0) `s1 = ABH`).

```
( 1) void Sheets::rectangle(){
( 2) Rec = new int[5];
( 3) int maxc=0, maxr=0, mc=0, mr=0;
( 4) for(unsigned int i=1; i<M.size(); i++){
( 5)   mc=(26+CTI(M[0]->s1[2])-CTI(M[i]->s1[2]))%26;
( 6)   mr=(26+CTI(M[0]->s1[3])-CTI(M[i]->s1[3]))%26;
( 7)   maxc=(mc>maxc)?mc:maxc;
( 8)   maxr=(mr>maxr)?mr:maxr;}
( 9) Rec[1]=maxc; Rec[2]=50; Rec[3]=maxr; Rec[4]=50;}
```

For messages from the set I the above algorithm determined the following rectangle `Rec[1]=25` , `Rec[2]=50` , `Rec[3]=25` , `Rec[4]=50` .

7.3 Determining products *AD* with 1-cycles (optimization - author's ideas)

Another optimization of Zygalski's method realizes the following idea. We check whether products *AD* contain 1-cycles or not only for permutations *AD* whose representations are within the determined rectangle and only for a message (0) ($|_{[4!]}$).

In the method `createSh()` we assign a sheet x to a message (0). Next we determine a product AD for each field $[x][j][i]$ which is situated in the rectangle. If AD contains 1-cycles, we place a pair `"(j,i)"` at the end of the list `PF` (look: the method `pushSh()`, section 6). Next (in the method `updateSh()`) we analyse remaining messages as follows. Let `M[i]` be a current message. We take from the list `PF` in turn each pair `"(c,r)"` (pairs `"(c,r)"` represent products AD with 1-cycles on a sheet (0)). And we check whether a field $[(c+D_M^{(i)}-D_M^{(0)})\%26][(r+D_R^{(i)}-D_R^{(0)})\%26]$ (on a sheet (i)), which covers a field $[x][c][r]$ (on a sheet (0)), also represents a product AD with 1-cycles. If this AD contains females, a pair `"(c,r)"` remains in the list `PF`, otherwise we remove it from `PF`.

If at the beginning we assigned a correct sheet to a message (0), then after analysing all the sheets, the list `PF` is empty or contains several fields. If the list is empty, it means we used improper drums or an order of drums is wrong. If the list `PF` contains several pairs, then these pairs usually represent the same product AD, that is we can obtain pairs (a,b), $(a+26,b)$, $(a,b+26)$, $(a+26,b+26)$.

```
( 1) void Sheets::updateSh(char* sh){
( 2) String crr="", dr="", s=""; char x;
( 3) int c, r, c1, c2, hlp, pf; Perm *A1, *D1, *AD;
( 4) for(unsigned int i=1; i<M.size(); i++){
( 5)   x = sh[i]; s="SHEET "; s+=x;
( 6)   pf=PF.size();
( 7)   for(int j=0; j<pf; j++){
( 8)       crr=PF[0]; PF.erase(&PF[0]);
( 9)       hlp = crr.Pos(',');
(10)       c = StrToInt(crr.SubString(1,hlp-1));
(11)       r = StrToInt(crr.SubString(hlp+1,crr.Length()-1));
(12)       c1 = (26+c+CTI(M[i]->s1[2])-CTI(M[0]->s1[2]))%26;
(13)       r1 = (26+r+CTI(M[i]->s1[3])-CTI(M[0]->s1[3]))%26;
(14)       dr=x; dr+=ITC(c1); dr+=ITC(r1);
(15)       A1 = createAorD(HE->Rs,HE->moveD(dr,1));
(16)       D1 = createAorD(HE->Rs,HE->moveD(dr,4));
(17)       AD = A1->Prod(D1);
(18)       if(AD->Fem()){PF.push_back(crr); s+="("+crr+")";}
(19)       delete A1; delete D1; delete AD;}
(20)   cout « s;}
(21) PF.clear();}
//----
(22) void Sheets::createSh(char x){
(23) String s="SHEET "; s+=x;
(24) for(int i=Rec[3]; i<=Rec[4]; i++)
(25)   for(int j=Rec[1]; j<=Rec[2]; j++)
(26)       s+=pushSh(HE->Rs,x,i,j);
(27) cout « s;}
```

Below we present a result of the calculations for the set I of messages.

```
SHEET Z(27,25) (31,25) (34,25) (35,25) (37,25) (40,25) (41,25) (42,25) (49,25) (50,25) (27,26) (28,26) (32,26) (39,26) (40,26) (43,26)
```

```
(45,26) (46,26) (47,26) (48,26) (50,26) (28,27) (35,27) (36,27) (37,27) (38,27) (43,27) (44,27) (28,28) (30,28) (33,28) (34,28) (36,28)
(37,28) (40,28) (43,28) (44,28) (45,28) (47,28) (48,28) (25,29) (27,29) (28,29) (32,29) (33,29) (34,29) (35,29) (36,29) (42,29) (48,29)
(49,29) (26,30) (35,30) (38,30) (43,30) (48,30) (49,30) (50,30) (29,31) (30,31) (31,31) (37,31) (39,31) (40,31) (44,31) (45,31) (46,31)
(48,31) (50,31) (25,32) (26,32) (29,32) (33,32) (34,32) (35,32) (37,32) (39,32) (40,32) (42,32) (45,32) (48,32) (30,33) (31,33) (34,33)
(35,33) (38,33) (39,33) (44,33) (45,33) (30,34) (31,34) (39,34) (40,34) (41,34) (46,34) (48,34) (49,34) (26,35) (29,35) (31,35) (32,35)
(33,35) (35,35) (37,35) (39,35) (45,35) (28,36) (29,36) (32,36) (36,36) (38,36) (42,36) (44,36) (50,36) (28,37) (30,37) (35,37) (36,37)
(40,37) (41,37) (42,37) (43,37) (44,37) (47,37) (25,38) (27,38) (41,38) (43,38) (49,38) (26,39) (27,39) (29,39) (33,39) (34,39) (38,39)
(39,39) (40,39) (41,39) (43,39) (44,39) (47,39) (49,39) (25,40) (26,40) (27,40) (31,40) (33,40) (34,40) (47,40) (49,40) (50,40) (29,41)
(30,41) (31,41) (32,41) (33,41) (34,41) (40,41) (42,41) (49,41) (29,42) (33,42) (34,42) (38,42) (40,42) (41,42) (43,42) (44,42) (49,42)
(50,42) (25,43) (28,43) (30,43) (31,43) (32,43) (36,43) (40,43) (42,43) (50,43) (28,44) (33,44) (34,44) (35,44) (36,44) (37,44) (40,44)
(41,44) (42,44) (43,44) (25,45) (29,45) (31,45) (34,45) (36,45) (40,45) (46,45) (47,45) (48,45) (50,45) (25,46) (26,46) (27,46) (29,46)
(34,46) (36,46) (37,46) (38,46) (43,46) (44,46) (45,46) (48,46) (50,46) (26,47) (31,47) (34,47) (36,47) (38,47) (42,47) (43,47) (45,47)
(46,47) (47,47) (48,47) (49,47) (26,48) (31,48) (34,48) (35,48) (38,48) (39,48) (43,48) (49,48) (25,49) (29,49) (39,49) (40,49) (42,49)
(44,49) (46,49) (47,49) (26,50) (27,50) (28,50) (39,50) (40,50) (41,50) (47,50) (50,50)
SHEET A(27,25) (31,25) (34,25) (40,25) (42,25) (50,25) (32,26) (45,26) (46,26) (50,26) (28,27) (28,28) (37,28) (48,28) (28,29) (33,29)
(34,29) (35,29) (48,29) (50,30) (45,31) (46,31) (25,32) (26,32) (33,32) (34,32) (40,32) (35,33) (38,33) (39,34) (40,34) (49,34) (26,35)
(29,35) (32,35) (39,35) (29,36) (38,36) (44,36) (50,36) (30,37) (36,37) (40,37) (41,37) (42,37) (47,37) (41,38) (26,39) (29,39) (34,39)
(41,39) (27,40) (31,40) (34,40) (47,40) (49,40) (30,41) (31,41) (42,41) (29,42) (44,42) (49,42) (30,43) (32,43) (36,43) (42,43) (50,43)
(33,44) (25,45) (31,45) (34,45) (36,45) (46,45) (25,46) (26,46) (29,46) (38,46) (26,47) (31,47) (26,48) (31,48) (34,48) (35,48) (49,48)
(29,49) (42,49) (44,49) (47,49) (41,50) (47,50)
SHEET B(34,25) (42,25) (50,25) (50,26) (28,27) (33,29) (34,29) (48,29) (50,30) (46,31) (25,32) (26,32) (33,32) (40,32) (39,34) (40,34)
(49,34) (26,35) (29,35) (38,36) (50,36) (40,37) (42,37) (41,38) (49,40) (30,41) (29,42) (44,42) (49,42) (30,43) (36,43) (42,43) (50,43)
(33,44) (31,45) (46,45) (25,46) (29,46) (38,46) (26,48) (42,49) (41,50)
SHEET C(50,26) (33,29) (48,29) (50,30) (26,32) (49,34) (50,36) (42,37) (36,43) (46,45) (25,46) (38,46) (26,48)
SHEET D(33,29) (26,32) (50,36) (42,37) (26,48)
SHEET E(26,32) (50,36)
SHEET F(26,32)
SHEET G(26,32)
SHEET H(26,32) SHEET I(26,32) SHEET J(26,32) SHEET K(26,32) SHEET L(26,32) SHEET M(26,32) SHEET N(26,32) SHEET O(26,32)
SHEET P(26,32) SHEET Q(26,32) SHEET R(26,32) SHEET S(26,32) SHEET T(26,32) SHEET U(26,32) SHEET V(26,32) SHEET W(26,32)
SHEET X(26,32) SHEET Y(26,32)
```

7.4 Determining the initial ring settings

Let us assume that we assigned the sheet Z to a message (0) and as a result we obtained the pair $(c,r) = (26,32)$. This means that each field (on each sheet) which covers the field $[\mathtt{Z}][26][32] = [\mathtt{Z}][0][6] = [\mathtt{Z}][\mathtt{A}][\mathtt{G}]$ (on the sheet Z) represents a product AD with females. Then we determine the initial ring settings using formulas given below (cf. [4] $|_{[3!]}$)

$$\begin{aligned}
\mathrm{IRS}_L &= (\mathrm{SRS}_L + (D_L^{(0)} - s))\%26,\\
\mathrm{IRS}_M &= (\mathrm{SRS}_M + (D_M^{(0)} - c))\%26,\\
\mathrm{IRS}_R &= (\mathrm{SRS}_R + (D_R^{(0)} - r))\%26.
\end{aligned}$$

We can treat $[D_L^{(0)} - s, D_M^{(0)} - c, D_R^{(0)} - r]$ as a vector where SRS is an initial point and IRS is a terminal point ($|_{[3!]}$). For the given set I of messages and for SRS = AAA we calculate the initial ring settings as follows.

$$\begin{aligned}
\mathrm{IRS}_L &= (\mathtt{A}+(\mathtt{A}-\mathtt{Z}))\%26 = 1 = \mathtt{B},\\
\mathrm{IRS}_M &= (\mathtt{A}+(\mathtt{B}-0))\%26 = 1 = \mathtt{B},\\
\mathrm{IRS}_R &= (\mathtt{A}+(\mathtt{H}-6))\%26 = 1 = \mathtt{B}.
\end{aligned}$$

We obtained IRS = BBB and in fact we generated the set I of messages for the initial ring settings BBB.

7.5 The sequence of hypotheses concerning a sheet (0)

Next problem lies in the fact that we do not know which sheet we ought to assign to a message (0). Historians write very generally about this problem. For instance [*When the sheets were superposed and moved in the proper sequence and the proper manner* ...] (cf. [8]). In the program we used a very general suggestion that cryptologists made hypotheses concerning sheets (cf. [5]). Next we observed, on the basis of many tests, that we can determine i-th sheet (for a message (i)) from the formula ($|_{[4!]}$).

$$\texttt{sheet[i]} = (\mathrm{SRS}_L + D_L^{(i)} + j)\%26 \quad \text{for } i = 0,1,\ldots,\texttt{numberOfMessages} - 1$$

j ($j = 0,1,\ldots,25$) denotes the number of the realized hypothesis. That is, in the first step we superpose sheets on a `sheet[0]` for $j = 0$, next we superpose sheets on a `sheet[0]` for $j = 1$, etc. The method `IRS()` solves the problem of the initial ring settings. We make all possible hypotheses (26 hypotheses) concerning assigning a first sheet to a message (0).

```
( 1) void Sheets::IRS(){
       // loading messages to the list M
( 2) char* sheet = new char[27];
( 3) rectangle();
( 4) for(int j=0; j<=25; j++){
( 5)   for(unsigned int i=0;i<M.size(); i++)
( 6)       sheet[i]=ITC((CTI(HE->Rs[1])+CTI(M[i]->s1[1])+j)%26);
( 7)   createSh(sheet[0]);
( 8)   updateSh(sheet);}}
```

For the given set I of messages we obtained a proper result for a hypothesis number $j = 25$. Then the program assigned the sheet `Z` to a message (0).

7.6 The discrepancies (optimization - author's ideas)

In the case of the set I of messages we received a proper result (i.e. the pair $(26,32)$) when we assigned the sheet `Z` to the message (0). Let us notice, that each field (on each sheet with the exception of the sheet `A`) which covers the field $[\texttt{Z}][26][32]$ is outside an interfered area. In the case of the sheet `A` (for a message (1)) the field $[\texttt{Z}][26][32]$ is covered by the field $[\texttt{A}][8][21]$. The field $[\texttt{A}][8][21]$ corresponds to the drum settings `AIV`. Let us notice that the turnover position of the right drum is $tpr = V = 21$. Thus $[\texttt{A}][8][21]$ is located inside the interfered area. But this field represents a product with females (it is coloured in black). The algorithm given above returned a proper solution, because the discrepancies did not influence the result.

Case one. Let us assume that a field $[s][c][r]$ (on a sheet s) covers the field $[\texttt{Z}][26][32]$, $[c][r]$ is inside an interfered area and does not represent a product AD with 1-cycles.

Then a correct result will be removed from the list `PF` during analysing a sheet s and finally the list `PF` will be empty. We shall interpret this fact as an improper order of drums.

How did cryptologists solve this problem? Historians write that cryptologists before beginning of the analysis of messages, eliminated some of them in order to avoid the situation described above. For each order of drums they had to eliminate another set of messages. We can find different formulas describing which messages ought to be removed.

The improvement of Zygalski's method is a realization of the following idea. Let us assume that we have case one. Before determining a product *AD* for a field $[s][c][r]$ we check whether this field is inside an interfered area or not. If it lies outside, we check a product *AD* like previously, otherwise we leave a pair `"(c,r)"` in the list `PF` (we treat this field as a representation of a product *AD* with 1-cycles) ($|_{[4!]}$). After analysing all messages we can get additional, wrong results, but we shall avoid a situation we omit a proper order of drums. If we obtain some results, it is not difficult to verify that a given one is wrong. We can analyse an additional message, for instance.

```
( 1) void Sheets::updateSh(char* sh){
( 2) String crr="", dr="", s=""; char x;
( 3) int c, r, c1, r1, pf, hlp; Perm *A1, *D1, *AD;
( 4) for(unsigned int i=1; i<M.size(); i++){
( 5)   x =sh[i]; s="SHEET "; s+=x;
( 6)   pf=PF.size();
( 7)   for(int j=0; j<pf; j++){
( 8)      crr=PF[0]; PF.erase(&PF[0]);
( 9)      hlp = crr.Pos(',');
(10)      c = StrToInt(crr.SubString(1,hlp-1));
(11)      r = StrToInt(crr.SubString(hlp+1,crr.Length()-1));
(12)      c1 = (26+c+CTI(M[i]->s1[2])-CTI(M[0]->s1[2]))%26;
(13)      r1 = (26+r+CTI(M[i]->s1[3])-CTI(M[0]->s1[3]))%26;
(14)      dr=x; dr+=ITC(c1); dr+=ITC(r1);
(15)      if(cM(r1,0,0)||cM(r1,0,1)||cM(r1,0,2)||cM(r1,0,3)){
(16)         PF.push_back(crr); s+="("+crr+")";}
(17)      else{
(18)         A1 = createAorD(HE->Rs,HE->moveD(dr,1));
(19)         D1 = createAorD(HE->Rs,HE->moveD(dr,4));
(20)         AD = A1->Prod(D1);
(21)         if(AD->Fem()){PF.push_back(crr); s+="("+crr+")";}
(22)         delete A1; delete D1; delete AD;}}
(23)   cout « s;}
(24) PF.clear();}
//----
(25) String Sheets::pushSh(String rs, char x, int i, int j){
(26) Perm *A1, *D1, *AD;
(27) String dr="", s="", crr="";
(28) crr=IntToStr(j)+","+IntToStr(i);
```

```
(29) dr=x; dr+=ITC(j%26); dr+=ITC(i%26);
(30) if(cM(i,0,0)||cM(i,0,1)||cM(i,0,2)||cM(i,0,3)){
(31)   PF.push_back(crr); s="("+crr+")";}
(32) else{
(33)   A1 = createAorD(rs,HE->moveD(dr,1));
(34)   D1 = createAorD(rs,HE->moveD(dr,4));
(35)   AD = A1->Prod(D1);
(36)   if(AD->Fem()){PF.push_back(crr); s="("+crr+")";}
(37)   delete A1; delete D1; delete AD;}
(38) return s;}
```

We do not change a body of the method `createSh()` . In the method `pushSh()` we added lines $(30-31)$ in which we check whether a field $[x][j][i]$ (on a sheet (0)) is located in an interfered area or not. If it is, we add a pair `"(j,i)"` to the list `PF`. In the other case we proceed like in section 6, that is we add a pair `"(j,i)"` if only *AD* contains any females. With the other messages we proceed in the method `updateSh()` in a similar way. We added lines $(15-16)$ in which we check whether a field $[x][c1][r1]$ (on a sheet (i)) is located in an interfered area or not. If it is, we leave a pair `"(c,r)"` in the list `PF`. Otherwise we proceed like in section 6, that is we leave a pair `"(c,r)"` in the list `PF` if only *AD* contains any females. Otherwise, we remove it.

Table 2. The set II (messages obtained for IRS = `EHM`)

(0) EFE QML QTU	(6) DRE WKD WWY	(12) UBA BGS BOF	(18) OBE FOE FDG	(24) VCM KWY KXG
(1) HOP PXP PBW	(7) NMD WLU WMR	(13) GOZ IRX IOP	(19) QRZ YTJ YVW	(25) XAB HIH HWQ
(2) LRM ZAX ZVV	(8) CEW MTY MGZ	(14) IRB UUT USU	(20) SCH YQU YGW	
(3) MAX ERG EKR	(9) PBW WYY WHM	(15) JAD FOB FFT	(21) WBC ZQW ZVD	
(4) BTX FIJ FOI	(10) RCI XCZ XSE	(16) QDZ QQH QBG	(22) YDF XZZ XXA	
(5) AAY VDN VQW	(11) TCE XZB XCA	(17) KBW VHT VEB	(23) ZDL IQO IOM	

Example 7.1 Let us analyse the set II of messages by means of the improved algorithm (for SRS = `AAA`).

The program assigned the following sheets to the given above messages:
`A, D, H, I, X, W, Z, J, Y, L, N, P, Q, C, E, F, M, G, K, M, O, S, U, V, R, T.`
Finally, as a result we obtained two pairs $(24,44)$, $(50,44)$. This means that each field (on each sheet) which covers the field `[A][24][44]` (or `[A][50][44]`) (on the sheet `A`) either represents a product *AD* with females or it is located inside the interfered area. Let us calculate IRS.

$\text{IRS}_L = (\texttt{A}+(\texttt{E}-\texttt{A}))\%26 = 4 = \texttt{E}$,
$\text{IRS}_M = (\texttt{A}+(\texttt{F}-24))\%26 = 7 = \texttt{H}$,
$\text{IRS}_R = (\texttt{A}+(\texttt{E}-18))\%26 = 12 = \texttt{M}$.

Thus IRS = `EHM`. Let us notice, we analysed messages (16) and (19). The first letter of the initial drum settings in both cases is the same (the letter `Q`). We assign the

same sheet M to these messages . The cryptologists did not have this possibility. They had only one sheet for each letter.

8. The plugboard settings

We know the initial ring settings. But it does not mean that we can read messages. We still have to find the plugboard settings. It is a well-known fact that the connections of the plugboard do not influence the shapes of products AD, BE, CF (cf. [6]). That is, if a product AD contains 1-cycles for the plugboard settings $S1$, then it contains 1-cycles for the plugboard settings $S2$. Therefore, the plugboard settings do not have any influence on the work of the above algorithm.

Example 8.1 We generated sets I and II of messages and executed all the above calculations for connections of the plugboard $S = (\mathtt{A,G})(\mathtt{C,F})(\mathtt{K,O})(\mathtt{L,Y})(\mathtt{R,W})(\mathtt{S,Z})$. Next we set up the plugboard as follows $S1 = (\mathtt{C,O})(\mathtt{D,I})(\mathtt{F,R})(\mathtt{H,U})(\mathtt{J,W})(\mathtt{L,S})(\mathtt{T,X})$ and we analysed the set II of messages once again. As the reader can guess, we obtained the same result.

References

[1] Baginski, C.: *Introduction to Group Theory*, SCRIPT, Warsaw, 2012.

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

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

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

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

[6] Gralewski L.: *Breaking of Enigma. History of Marian Rejewski*, Adam Marszalek Publishing House, Torun, 2005.

[7] Mostowski A., Stark M.: *Elements of Higher Algebra*, PWN, Warsaw, 1970.

[8] Rejewski M.: How did Polish Mathematicians Decipher the Enigma, *Polish Mathematics Association Yearbooks*, Series 2nd: Mathematical News XXIII (1980).

[9] http://pl.wikipedia.org/wiki/Enigma.

KRYPTOANALIZA SZYFRU ENIGMY

Streszczenie Tematem pracy jest kryptoanaliza Enigmy wojskowej używanej przez siły zbrojne oraz inne służby państwowe Niemiec podczas II wojny światowej. Jesteśmy zainteresowani problemem dekodowania zaszyfrowanych depesz transmitowanych po 15 września 1983 roku. Prezentujemy pełny algorytm służący do generowania ustawień pierścieni, odgadywania które rodzaje bębenków zostały wybrane i wyznaczania ich porządku na wspólnej osi. Proponowany algorytm jest uzupełnieniem i optymalizacją metody płacht Zygalskiego. W pracy opisane są istotne własności płacht wynikające z konstrukcji maszyny i teorii permutacji (spostrzeźenia autora). Do czytania depesz zaszyfrowanych za pomocą Enigmy potrzebne są jeszcze ustawienia łącznicy wtyczkowej. Połączenia łącznicy nie mają wpływu ani na metodę Zygalskiego (znany fakt) ani na przedstawiony algorytm. Brakujący (oryginalny) algorytm rozwiązujący problem łącznicy (wraz z algebraicznym opisem) pojawi się niebawem.

Słowa kluczowe: metoda płacht Zygalskiego, ustawienia pierścieni, ustawienia depeszy