NeMa - NEue MAschine

Analyzing the Swiss Cipher Machine

Christoph Ehret,
Jacek Jonczy,
Jürg Nietlispach,
and Nicolas Zwahlen

September 5, 2007

Abstract

This report describes the NeMa project realized in the context of the repetition course of the cryptology section in the year 2007. The main goals of the project are: (1) to analyze the former Swiss cipher machine called NeMa (NEue MAschine) with respect to cryptological features, and (2) to implement a software simulation program for this machine.

Contents

1 General Objectives

The following list enumerates the objectives which we think can be accomplished within 10 days of work.

1. To acquire a deep understanding of NeMa mode of operation by considering the literature and by playing with the machine itself;
2. A detailed analysis of NeMa cryptographic complexity
3. NeMa Cryptanalysis: possible attacks and cipher breaking example
4. To devise a software simulator for NeMa

2 Introduction

The Swiss cipher machine NeMa was, for its period, a very sophisticated successor of the German Enigma. In fact, NeMa has been designed in order to replace a Swiss version of Enigma. The Enigma is very well known and has been extensively studied, probably because of its meaningful role it played during World War II. However, the information available about the NeMa is rather poor. Wikipedia provides a very short article [9]. Scanned copies of the original user's manual and some pictures can be found on [3]. Probably the most important source is the article [6], together with other information found on [7]. A good overview of historical cipher machines is provided in [5]. Other sources found are [4, 1] on the web, as well as a student thesis [2].

3 NeMa Functionality

3.1 Overview

The NeMa rotor cipher machine is based on the german Enigma and works on the same principle. With the keyboard - 50ies typewriter-style - one can type the plaintext in single characters. Each character will be substituted by any another (polyalphabetic substitution). A push on a character button cause multiple encryption mechanisms and at the end of the encryption chain the cipher caracter will be enlighted on a second cipher keyboard. The numerous encryption methods are realized in mechanical and electrical constructions of the machine and are described below.

3.2 Hardware construction

The NeMa is equiped with 10 wheels. Each containing a set of 26 european standard alphabet characters ('A' to 'Z') named the alphabet ring. Every wheel is furthermore outfitted with a "tooth

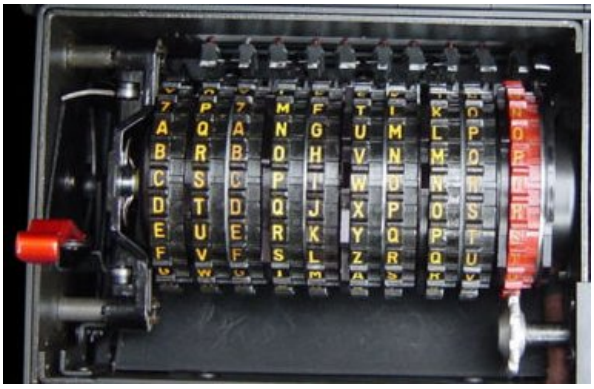

Figure 1: Front view on the NeMa wheels.

ring" beside the alphabet ring (right hand sinde). Every teeth is associated with a character. Those wheels are the encryption core. There are several types of wheels:

- Umkehrwalze (UKW): Position 10
- Contactwheels: Positions 2, 4, 6, 8
- Drivewheels: Positions (1), 3 ,5, 7, 9
- Eintrittswalze (ETW): Position 1 (red)

Contact wheels The contactwheels are labelled as follows: A, B, C, D, E, F. The electrical signals (incoming from the ETW) enter the contactwheel on the right hand side (flexible contacts) and leave it on the left hand side (V-contacts). The port1 for an incoming signal is on the same position as the pierced letter I, port2 with J, port3 with K, ..., port 26 with G. The arrangement of ports is in direction to the ascending alphabet direction and counter clockwise when watched the wheel from the right hand side. Only 4 of the contactwheels can be used for one machine setting, therefore the selection of a subset of those wheels are part of the encryption key.[1]

Figure 2: Contact wheel front view (turn to the left hand side)

Figure 3: Contact wheel rear view (turn to the right hand side)

Umkehrwalze UKW The UKW is a special Contactwheel. An incoming (entering through the ETW) electric signal is beeing reflected. There are different UKWs but one UKW used per Machine, since it can only be removed with the axis. The wiring is shown in the redirection table 3.3.1 and 2.

Drive wheels The drivewheels haven't got any electrical units, they work exclusively mechanical. On the left hand side of a drivewheel a notch ring is additionally attached. This notch ring with its slots and notches affects the position of the following contact wheel (see 3.3.2). The rotation positions can be changed via the mechanical machinery (the "arms" or "levers" inside the machine). The label for the drivewheel is determined by the unique label of the notch ring.

Eintrittswalze ETW (rot) The ETW is a hybrid: on one hand a contact wheel has a wiring and cannot be removed. On the other hand it has notch rings, therefore it is a drivewheel. On the right hand side of the ETW is a second notch ring. Its task is explained later (see 3.3.2).

The wiring of the ETW is simple: it works as a conductor in every wheelposition (and performs no electrical encryption - though). There is a hidden encryption inside or beside the ETW (we didn't found the reel): The wiring for the ETW[2] is realized in order to the keyboard: port1 - Q, port2 - W, port3 - E, port4 - R, ..., port26 - M, in the opponent direction to the typing rotation (and the pierced alphabet direction on the wheel). It does not depend on the ETW wheel position so this encryption is static. We found out that the port1 is on the highest position, so every time we stroke 'Q' the top position of the ETW sends a signal.

[1] The two redundant wheels are stored in the rear of the NeMa box.
[2] In future we equate the meaning of the wiring of the hidden reel and the wiring of the ETW.

3.3 Encryption and decryption

3.3.1 Electrical encryption with contactwheels

As mentioned - the contact wheels are wired and this wiring is "random" for each wheel. This means that the incoming and the outgoing ports are linked "randomly". The specific redirections are listed in the table below. This wiring performs the electrical encryption. For example: An incoming electrical signal on port 1 is redirected to port 5 and leaves the wheel on this specific position. Naturally, the signal on its way from the ETW to the reflecting UKW and back, is redirected in this way in each contact wheel. The UKW and the ETW have their own specific wiring (see table 3.3.1 and 2 for the UKW and the mapping table 4 for the ETW). Every wheel performs a substitution. Therefore the plain character P becomes the cipher character C with the following mapping:

$$P \rightarrow ETW \Rightarrow W_2 \Rightarrow W_4 \Rightarrow W_6 \Rightarrow W_8 \Rightarrow UKW \Rightarrow W_8 \Rightarrow$$

$$W_6 \Rightarrow W_4 \Rightarrow W_2 \Rightarrow ETW = C$$

	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Wheel	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
A	05	14	15	19	13	02	22	10	04	18	16	26	24	09	23	25
B	04	7	18	09	20	15	08	11	16	01	10	24	19	25	13	22
...																
F									...							
UKW									...							
ETW	09	15	23	10	12	18	13	01	07	08	19	11	22	24	25	26

Table 1: Table of redirections 'I' to 'X'

	Y	Z	A	B	C	D	E	F	G	H
Wheel	17	18	19	20	21	22	23	24	25	26
A	08	20	06	11	03	01	12	21	07	17
B	14	21	03	02	17	06	12	05	23	26
...										
F					...					
UKW					...					
ETW	02	03	04	17	16	21	14	20	06	05

Table 2: Table of redirections 'Y' to 'H'

3.3.2 Mechanical encryption with drivewheels and stepping levers

The mechanical encryption is supported with ten stepping levers, which effects the rotation of the wheels (see figure 4). There are two types of stepping levers:

- Toothring levers
- Toothring levers dependent on notchring pattern of the ancessor(= notchring lever)

These levers build a set of "claw-fingers" and their motion sequence is determined by one simple pattern: Every stepping lever causes a rotation step on the toothring when stroking a key. With

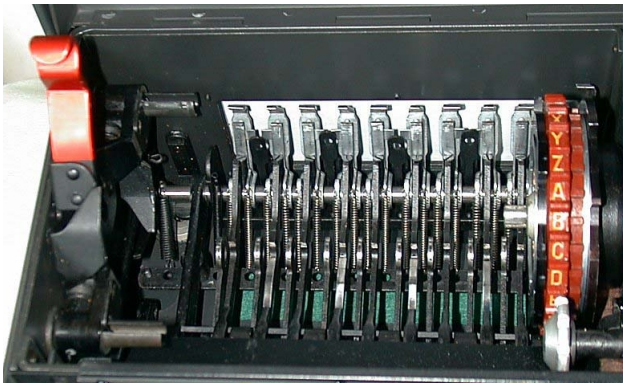

Figure 4: The figure shows the "claw finger" machinery which perform the rotation of the wheels. We can destinguish between the toothring levers and the notchring levers.

this configuration it is guaranteed that all wheels rotate together at once. Now, the lever jumps backwards if one releases the keyboard button.

To achieve more irregularity in encryption it would be useful to interrupt several wheels arbitrary. This goal could be reached with the second type of levers. Thats why they are wider than the other levers. The wider levers can be arised by the (inactive higher) position of the neighbouring (right) notch ring when executing a button stroke. The higher position of the notch ring prevents a rotation step of the neighbouring (left) toothring. This is the key feature of the mechanical encryption. A pair of toothring levers and toothring levers dependent on the notchring pattern are collected in (blocking) arms C1, C2, C3, C4, C5.[3]

There are 23 known notch ring patterns. A selection of the patterns is seen below.

N.R.	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
2	0	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	1	0
12	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1
⋮									⋮									
22									...									
23									...									

N.R.	S	T	U	V	W	X	Y	Z
1	1	0	0	0	0	0	1	1
2	0	0	0	0	0	0	0	0
12	1	0	1	1	1	1	1	1
⋮				⋮				
22				...				
23				...				

Table 3: The notch ring patterns. The 1-region prevent the neighbouring wheel to move (inactive), the 0-region enables to move the next wheel.

The mechanical encryption based on the principle described above can be upgraded and made more difficult:
The notchring on the right hand side of the red ETW is linked to a sensing lever which can

[3]Those arms can be (initially) blocked. In our configuration C2 and C4 are blocked

influence arms. Together with the control of the neighbouring toothwheel via a notchringlever, the notchringlever itself can be controlled by the notch ring pattern of the ETW notch ring.

Additional to the sensing lever some pairs of stepping levers can initially defined as active/inactive. The corresponding pair of levers has to be fixed with screws.

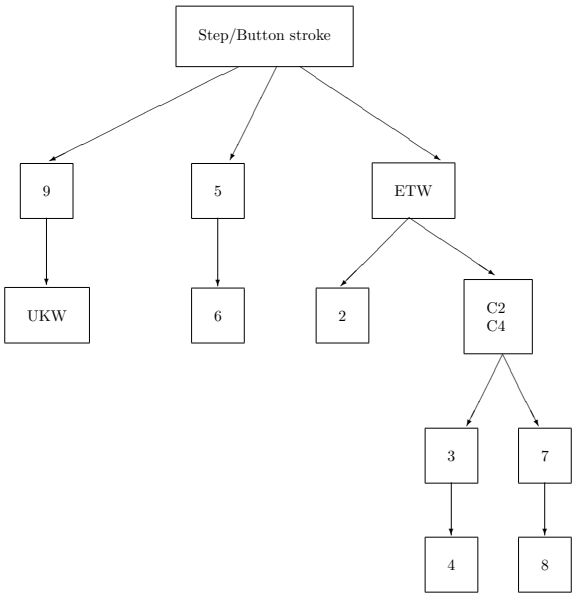

Figure 5: NeMa statediagram of wheelpositions. The toothlevers principally move every time (and therefore not displayed in the diagram) unless they are blocked by a notch ring or the (blocking) arms. Those dependencies are shown in the diagram.

The current configuration works with wheel Nr. 1, 5, 9 moving every time; 2, 6, 10 dependent on (1, 5, 9); 3, 7 dependent on (2, 6, right ETW notch ring); 4, 8 dependent on (3,7); (see figure **??**). As one can see the dependencies are inherited.

The whole encryption of a plaintext character P to the ciper character C can now be formulated in terms of:

$$P \rightarrow ETW \Rightarrow \circledast_1(W2) \Rightarrow \circledast_3(W4) \Rightarrow \circledast_5(W6) \Rightarrow$$
$$\circledast_7(W8) \Rightarrow \circledast_9(UKW) \Rightarrow ... \Rightarrow ETW = C$$

where $\circledast_i$ is the current position in the specific notch pattern (including dependencies).

3.3.3 An example

With the knowledge of the encryption mechanisms described above, it is now possible to track the signal on the way from the keystroke to the enlighted lamp depending on the wheelpositions, notch patterns and wheelorders.

Assume that the wheel order is 15D-14C-13B-12A-1/22, where the letter is the contactwheel ID and the number the drivewheel ID, the wheelpositions are

$$A_{UKW} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_{ETW}$$

and the (initially blocked) blocking arms are C2 and C4. 1/22 means: left/right notch ring on red ETW[4].

[4]It is very important not to align the key on the highest wheelposition. The key has to be aligned to the chassis next to the lampboard!

When we type the letter A the position of the wheels change to

$$Z_{UKW} Z_9 A_8 A_7 Z_6 Z_5 A_4 A_3 Z_2 Z_{ETW}.$$

The blocking arms C1, C3 and C5 did not block, therefore ETW,2; 5,6; 9,10, stepped forward. The notchrings of ETW, 5, 9 had together the inactive region (otherwise they would have blocked). C2 and C4 blocked 3,4 and 7,8, therefore those wheels didn't move. C2 and C4 block initially, they could only have been activated with the right notch ring of the ETW. Therefore the notchring on the right hand side of the ETW was on region 0.

The signal is now entering the ETW from the top position (port1 - Q). The track of the signal with the corresponding (portnumbers) and the substituted characters is shown in the two figures below. The first picture shows the detailed encryption for character 'Q' the second for character 'A'. The horizontal arrows describe the port position differences where the vertical arrows the substitutions inside a contactwheel show. We eased the calculation of the position differences between signal output through the ETW to the first contact wheel (A) in building a tabular with the specific entries and offsets (see table 4).

3.3.4 Decryption

The decriptionmethod is very similar to the method used by ENIGMA:

1. Select the key (contactwheels & order, notchwheels & order, initial positions of wheels, blocking arms) = K
2. Type the plaintext (and encrypt it with k $\rightarrow e_k(P) = C$) and build the ciphertext characterwise
3. Transmit the ciphertext
4. Decode the ciphertext by typing the ciphertext on the machine with the same configuration (K) and get the plaintext

The principle on which the encryption and especially the decryption is based is very simple and convenient (one can only enter the ciphertext with the same configuration K and get the plaintext): The electricity enters in the case of the encryption portX goes through the encryption mechanisms and leaves portY. The enlighted character on the lampboard is linked with portY see figure 6.

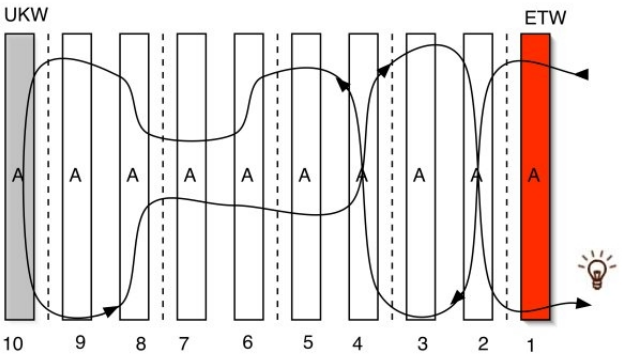

Figure 6: The "encryption signal" entering through the ETW enlights the ciphertext character C after contactwheel-substitutions (2, 4, 6, 8, UKW). For the decryption the same configuration K is needed: The "decryption signal" flows simply the reverse direction and enlights the specific plaintext character P.

In case of the decryption (see figure **??**)the current is flowing in the opposite direction of the circuit (the same configurations as in case of the encryption, though) from portY to portX. The lamp on portX enlights the original plaintextcharacter.

1	I	Q	1
2	J	W	26
3	K	E	25
4	L	R	24
5	M	T	23
6	N	Z	22
7	O	U	21
8	P	I	20
9	Q	O	19
10	R	P	18
11	S	A	17
12	T	S	16
13	U	D	15
14	V	F	14
15	W	G	13
16	X	H	12
17	Y	J	11
18	Z	K	10
19	A	L	9
20	B	Y	8
21	C	X	7
22	D	C	6
23	E	V	5
24	F	B	4
25	G	N	3
26	H	M	2

Table 4: The first two rows describe the mappings between channels and character of the contact wheel A. The second pair of row shows the character on the keyboard and the position for the electric signal leaving when a button was stroked.

3.3.5 Comparison with Enigma

The later Enigma types also used a set of 5 contact wheels. So - as mentionned above - one big difference between the NeMa and Enigma ciphermachines is the use of notch rings in the NeMa. The complexity of encryption with the notch rings is therefore a huge times bigger (see Section 4).

There is a second difference: the rotation of the wheels. Enigma has a "clock-system" where for the first 26 entered character only the first wheel moves. The next 26 character are encrypted additionally with the second wheel etc.

The NeMa hasn't got any cables to perform additionaly keyboard encryption. This means that the Enigma cables redirect some keyboard letters to different ports in the ETW which is not that mutch powerful in encryption than the use of notch rings.

3.3.6 Improvements

NeMa does not solve the problem of involutions and fixpoint free permutations. The machine called SIGABA uses a switch for the encryption/decryption mode and prohibits these problems coming along with the UKW.

There is probably a basic encryption model hurt: not the algorithm has to be secret, the key has to be!

The encryption could be made more powerful increasing the number of wheels used at one time (e. g. SIGABA uses 10 contact wheels).

4 NeMa Cryptographic and Combinatorial Properties

There used to be two models of NeMa, a *Training model* and an *Operational model*. Here we concentrate on the latter, since it offers a greater diversity.

4.1 Initial Configurations

A great number of initial machine settings results from the high number of cryptographic components. The effective number of elements is still unknown, but it may be assumed that more elements exist than ever was known officially. The following elements are available:

- A set of six contact wheels $A, \ldots, F$. Additionally, probably eight other contact wheels exist (spare parts).
- A total (potential) number of nine single digit notch rings (only used together with the red drive wheel).
- A total (supposed) number of 23 double digit notch rings)

For our analysis, we assume that six contact wheels are effectively in use. Four of those are plugged into the machine at the same time. Two notch rings are used with the red drive wheel (a single digit and a double digit, respectively), and one with each of the other four drive wheels, respectively. Furthermore, we neglect the possibility of adjusting the blocking arms; this would lead to even more initial settings. Taking these assumptions into account, we can determine the total number of possible initial configurations.

$$\#cw = \text{number of contact wheel orders } = \binom{6}{4} \cdot 4! = 360$$
$$\#nw = \text{number of notch wheel orders } = \binom{23}{4} \cdot 4! = 212520$$

The latter equation holds under the assumption that we ignore both notch rings of the red drive wheel (this is acceptable, since they were rarely changed). The effective number of possible notch wheel orders is much smaller because not all above combinations could be used. The reason for this lies in the cycle length issue which is described in the next subsection.

Finally, there are 26^5 possible starting positions for the drive wheels as well as for the contact wheels. This is because all 10 wheels are labeled with 26 letters.

4.2 Cycle Length

NeMa's five wired contact wheels do not all interact together, but can be considered as two distinct, independent groups. Wheels 10 and 6 are controlled by drive wheels 9 and 5, respectively. The latter step at each key stroke, like wheel 1. Wheels 2, 4, and 8 depend all on wheel 1; wheels 4 and 8 additionally depend on drive wheels 3 and 7 which in turn are controlled by drive wheel 1. Within these two groups, the wheels have active notch counts which must be relatively prime to each other and to the alphabet size, i.e. to 26. This yields the maximal cycle length per group:

- Group 1: wheels 10 and 6; cycle length $26^2 = 676$
- Group 2: wheels 8, 4, and 2; cycle length $26^3 = 17576$

This implies that the number of possible cycles is $26^{10}/26^3 = 26^7$. The cycle length of group 2 is also the overall machine cycle length since after 17576 steps also wheels 10 and 6 will have returned to their starting positions. In some special situations (certain machine configurations) however, the cycle length may be lower. But the rules determining the cycle length have not been studied in depth yet.

4.3 Cipher Properties

The NeMa cryptosystem allows the following alphabets:

$$\begin{aligned} \text{plaintext alphabet:} \quad & \mathcal{M} = \{A, \ldots, Z, 1, \ldots, 9\}, \\ \text{ciphertext alphabet:} \quad & \mathcal{C} = \{A, \ldots, Z\}, \\ \text{(outer) key alphabet:} \quad & \mathcal{K} = \{A, \ldots, Z\} \end{aligned}$$

NeMa produces a *polyalphabetic stream cipher* by sequencially substituting a plaintext symbol $m \in \mathcal{M}$ by another symbol $c \in \mathcal{C}$. The encryption of a plaintext symbol m by the encryption function E can be summarized as follows:

$$E_{k_{in},k_{out}}(m) = P_1 \circ P_2 \circ P_3 \circ P_4 \circ P_5 \circ P_4 \circ P_3 \circ P_2 \circ P_1(m) = c \tag{1}$$

where P_1 to P_5 are permutations over $\mathcal{M}$ (more specifically: involutions). P_5 is the permutation performed at the reflector. k_{in} and $k_{out} \in \mathcal{K}$ are the inner and the outer key, respectively. In fact, the P_i are classes of permutations in the following sense: for each plaintext symbol they change in a non linear manner by a semi-automatic stepping of the outer key, which depends on the initial setting of both the outer key and the inner key. The decryption proceeds in the reverse way to (1):

$$D_{k_{in},k_{out}}(c) = P_1 \circ P_2 \circ P_3 \circ P_4 \circ P_5 \circ P_4 \circ P_3 \circ P_2 \circ P_1(c) = m \tag{2}$$

The key length of k_{out} is 10, hence the outer key space has cardinality 26^{10}. The inner key k_{in} consists of a single machine configuration, hence the inner key space has size $\#cw \cdot \#nw$. The keys are subject of the next subsection.

4.4 Key Use and Generation

The NeMa secret key consists of two parts, namely an inner key and a code word. The exact procedure of setting up the inner key is described in a (secret) booklet issued by the Swiss government[5]. What is known is that the secret key was issued by special key instructions and that the validity period depended on the type of traffic and thus varied from case to case. The two key parts are as follows:

a). **The inner key (Wochenschlüssel):** sets up the order of the contact wheels and the drive wheels and is given in the following form:

$$x_1Y_1 - x_2Y_2 - x_3Y_3 - x_4Y_4$$

where $x_1, \ldots, x_4$ are double digit notch rings, and $Y_1 \ldots Y_4$ are contact wheels.

b). **A code word:** a string $s \in \mathcal{K}$ such that $|s| \geq 10$ which is used for generating the outer key.

The outer key, or message key, was different for every encrypted message in order to maintain cryptographic security. Its generation is described below.

Generating the outer key (Tagesschlüssel).

1. Choose a code word and set its first 10 letters from left to right on the letter rings of the 10 wheels.

2. Choose 10 random letters, divide them into two 5-letter groups, and place them at the beginning and at the end of the ciphertext.

[5] "Verschlüsselungsverfahren für die NeMa Maschine", geheim, Ausgabe Mai 1948.

3. Using the setting from 1, encrypt the 10 random letters from 2 and keep them noted.

4. The resulting encrypted letters constitute the secret message key which is now set on the wheels. The machine is now ready to encrypt messages.

Uppon receiving the ciphertext, the recipient takes the random letters from the beginning (or end) of the ciphertext, and using the code word she generates the outer key. Decryption then follows exactly the same procedure as encryption.

4.5 General Security Issues

The high number of possible machine settings together with the complex stepping offered for the period a reasonable security. Traditional cryptanalysis, i.e. by paper and pen, would be certainly quite hard. Also, the fact that the 10 random letters were transmitted in clear (together with the cipher) implicates that no information about the machine could be given away through the message key. Nevertheless, some weaknesses in both design and usage could give some indication for breaking a cipher, as explained in the sequel.

It follows from the technical properties of the machine that ciphertext and plaintext are never equal, i.e. $m \neq c$ always holds. In other words: a letter cannot be encrypted with itself. This was also the case in NeMa's predecessor, the German Enigma. This fundamental weakness can be exploited in a chosen-plaintext attack: if ever $m = c$ is found by comparing ciphertext and possible plaintext sequences, the corresponding key can be excluded.

Several further weaknesses of the system result rather from its improper usage. First, the inner keys ware changed rarely. Second, code words often consisted of certain (dictionnary) terms. And third, some cleartext sequences were often repeated. Such lack of caution makes cryptanalysis much easier, especially when an exemplary of a machine is at disposition.

There are no hints how to to choose the 10 random letters. Apparently, NeMa cryptographers assumed it is very unlikely that the same (or similar) letter sequences could be chosen for different messages (except by chance). And even if that would happen, it was assumed that the irregular motion of the wheels would in most cases prevent possible decipherment.

5 NeMa Cryptanalysis

5.1 Breaking NeMa: Theoretical Issues

If the machine is used more carefully with respect to security (i.e. random code word, non-repeating text sequences, etc.), the cryptanalyst is confronted with quite a large search space. Given the facts from the previous sections, we can make the following theoretical statements with respect to the complexity of possible attacks.

(1) A **brute force attack on the outer key** takes about 26^{10} or roughly 2^{47} steps.

(2) A **brute force attack on the inner key** takes about $\#cw \cdot \#nw = 76507200$ steps which roughly corresponds to 2^{26}, given that all possible machine configurations are considered.

Breaking solely the outer key (Attack 1) is already quite a challenge, depending on how many keys can be tested per second. Supposed that 10^6 keys per second can be checked, it may take about 50 years on a standard PC. If the initial machine setting is unknown (which is normally the case), the outer key space must be searched for every possible machine configuration, i.e. for each inner key. Thus the complexity grows even higher up to $2^{47} \cdot 2^{26} = 2^{73}$, which is too hard for today's standard computational means.

However, a pure brute force attack can be avoided by taking advantage of some additional properties. First, it has been already mentioned that only a subset of all machine configurations is used in order to maximize the cycle length. And second, the fact that $m \neq c$ always holds may be also exploited, see Subsection 4.5. Another feature which may be exploited in an attack is the

cycle length, thus repeating keys. However, this is not a great use because the cycle length (17576) is larger than most messages.

Yet another useful phenomenon for cryptanalysis may be the following observation: at certain machine configurations the contact wheels do not move at all, which is repeated furthermore every 26 letters. As a result, this produces an identical substitution at two positions in the message text at 26 letter intervals.

5.2 Breaking NeMa in Practice

In order to perform real cryptanalysis, we have implemented a simulation program for NeMa along with scripts used for cryptanalysis. In the following, we describe some attempts of breaking a NeMa cipher by using the simulator.

5.2.1 Brute-Force Attack

Using the software simulation of the NeMa cipher machine, a brute-force attack can be attempted. In this attack, it is assumed that the inner key (set of contact wheels and notch rings) is known, while the outer key (ten-letter password) is unknown. The attacker is in possession of an intercepted ciphertext, the plaintext being unknown.

The procedure of the attack is the following: each of the 26^{10} outer keys is used to decipher the ciphertext; if the obtained deciphered text is compatible with a spoken language (e.g. English), the key is considered possibly correct.

To identify a text as English, a χ^2 variable based on the letter frequencies of the text is used. The frequencies f_i of the ith letter of the alphabet for the English language were estimated by Beker and Piper [**?**] (see also figure **??**). For a text of length N containing N_i occurencies of the ith letter of the alphabet, we compute

$$\chi^2 = \sum_{i=0}^{25} \left(\frac{N_i - N f_i}{\sqrt{N f_i}} \right)^2$$

The reduced chi-square, χ^2/N is independent of the text length and is a powerful test to distinguish between English text and text consisting of random letters: typically for English we have $\chi^2/N < 0.5$, while for random letters $\chi^2/N > 1$ (see figure 7).

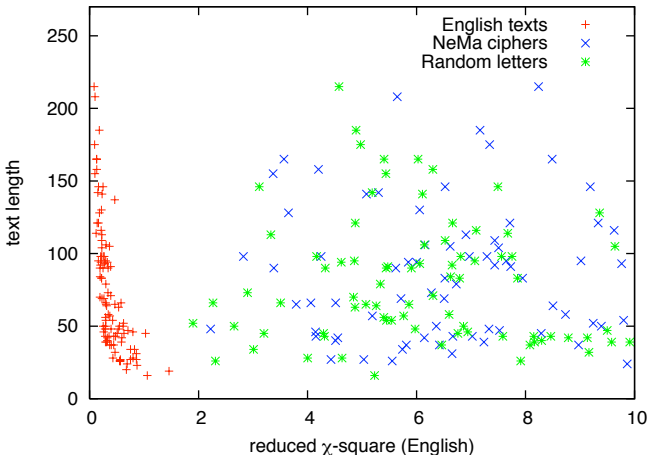

Figure 7: Reduced χ^2 distribution as a function of text length. Based on a sample of 100 English sentences, the corresponding NeMa ciphertexts using a random password, and random letter texts.

In the attack, a candidate key is considered possibly correct if the resulting $\chi^2/N < 0.45$. Wrong matches are possible but rare; running the simulation on the ciphertext with the possible key immediately identifies it as correct or wrong. This method requires a sufficiently long ciphertext for the frequencies to be meaningful. A length of about 50 letters seems to be enough; longer ciphertext can be truncated to speed up the attack.

The main issue of the brute-force attack is speed. There are a huge number of keys to be tried. Using our `C++` simulation on a 52-letter ciphertext with a 1.86 GHz processor, a rate of about 31'300 keys per second was reached. This implies that trying the whole 26^{10} keys would take about 145 years. The correct key was identified, and wrong matches occurred at a rate of about 1 for 1.9 million keys.

5.2.2 Known-Plaintext Attack

Here, both the ciphertext and the plaintext are known to the attacker. Even so, recovering the key is not immediate.

The procedure is again to try each of the 26^{10} outer keys to decipher the ciphertext one letter at a time, and compare this letter with the corresponding plaintext letter. As soon as both letters don't match, the candidate key can be discarded as wrong, and the next key can be tried. If the ciphertext is deciphered until the end without contradicting the plaintext, the candidate key can be identified as the correct key and the procedure stops.

Thus this attack is much faster than the brute-force attack, where all the ciphertext has to be deciphered. With our simulation, a rate of about 480'000 keys per second was reached, independently of the ciphertext length. Thus, trying all possible keys "only" takes about 13 years.

5.2.3 Lookup-Table Attack

The brute-force attack could be accelerated by using a look-up table. The table would contain the ciphertext letter corresponding to each plaintext letter for each outer key. Thus the encryption could be performed without running through the actual wiring of the machine. In the table, the outer keys should be ordered in the same way as they would appear during the encryption cycle, thus the software computation of the rotor wheel states could also be avoided.

However, storing such a table requires about 13'352 terabytes for each inner key (storing the entries as 32-bit integers; in principle a number $0 < i < 25$ can be stored on 5 bits). This is hardly achievable with the currently available storage devices.

6 NeMa Simulation

A first prototype simulation was written in `perl`. A second simulation was written in `C++`. The compiled `C++` code turns out to be about 50 times faster than the interpreted `perl` code, which is very useful when running attacks.

The NeMa simulation `perl` code can be found on the web at `http://lphe.epfl.ch/~nzwahlen/bonus/crypto/nema/`. The usage is the following, both for encryption and decryption:

```
./nema inputfile password
```

The `C++` simulation can be found at the same address and has the same arguments (see the online documentation for more details).

The plaintext input file can contain any characters; however only lowercase and uppercase letters will be enciphered. All other characters will be ignored by the programs. The `password` is the ten-letter outer key.

In both programs, the inner key is set to 12A–13B–14C–15D–22/1. It can be changed by editing the code. Contact wheels A, B, C, D, E, F and notch rings 1, 22, 12, 13, 14, 15, 17, 18 can be used.

During the encryption, the output shows the ciphertext corresponding to each plaintext letter, together with the rotor wheel state.

7 Conclusion

The investigation of NeMa turned out to be very fruitful, though not free of surprises. The complexity of this after all over 60 years old device is quite astonishing. It took a while to fully understand its functionality by studying existing (rather spare) literature as well as by trial and error. The implementation of a software simulator was a big help for that.

The great number of available elements (wheels and notch rings), the resulting combinatorial possibilities, and also the key length lead to a large key space which is infeasible to search by pure brute force. Some design weaknesses may be exploited in order to reduce cryptanalytic complexity. However, it is not that evident how to devise a sophisticated cryptanalytic tool in this case. Using the NeMa simulation, we have implemented two types of attacks: pure brute-force and known-plaintext.

A Figures

A.1 Signal Tracking Illustration

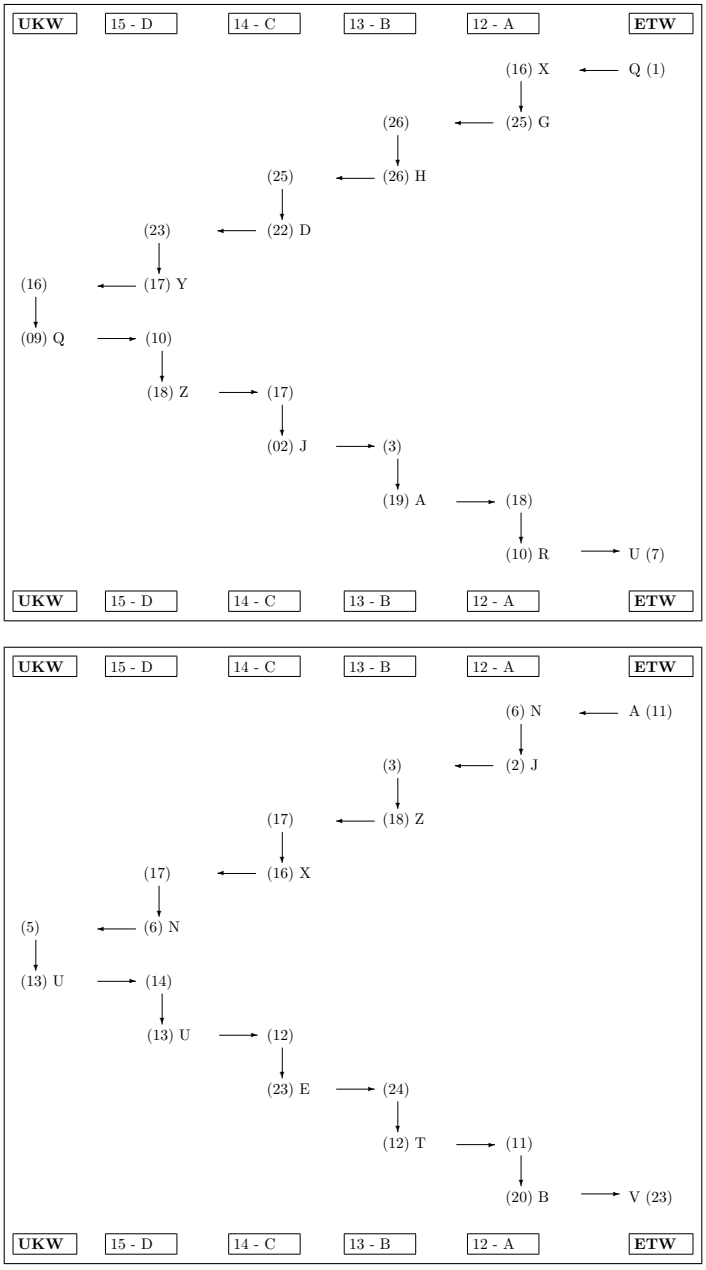

Figure 8: The whole signal track for the specific configuration K and the input character 'Q' (above) and the same for 'A' (below). The offset (horizontal keys) is in case A to Z -1, and in the case Z to A +1. The vertical arrows are the substitutions in the contact wheels. The substitutions can be seen in the redirection table (above)

References

[1] David Hamer. NEMA Neue Maschine. http://www.eclipse.net/∼dhamer/nema2.htm

[2] Christoph Lechleitner and Andreas Rumpler. Chiffriermaschine NEMA. Projektarbeit, Johannes-Kepler Universität Linz, 1995.

[3] Bob Lord. Swiss NEMA Encryption Machine. http://www.ilord.com/nema.html

[4] Jerry Proc. Crypto Machines Home Page. http://www.jproc.ca/crypto

[5] Michael Pröse. *Chiffriermaschinen und Entzifferungsgeräte im Zweiten Weltkrieg: Technikgeschichte und informatikhistorische Aspekte*. PhD thesis, Technische Universität Chemnitz, 2004.

[6] Geoff Sullivan and Frode Weierud. The Swiss NEMA Cipher Machine. *Cryptologia*, 23:310–328, October 1999.

[7] Frode Weierud. Frode Weierud's Crypto Cellar. http://frode.web.cern.ch/frode/crypto/

[8] François Weissbaum. Introduction to Cryptology. Script, Federal Cryptology Section, August 2006.

[9] Wikipedia. NEMA Neue Maschine. http://fr.wikipedia.org/wiki/NEue_MAschine