8 Cryptanalysis of Rotor Machines

The cryptanalysis of rotor machines is complex and depends on the details of the machine under examination. The book by Deavours and Kruh [2] is a standard work and contains many elaborate examples. Here we only depict some general ideas:

- Superimposition
- Meet-in-the-middle
- Isomorphs

Superimposition

Assume that the cryptanalyst got hold of several ciphertexts that are encrypted with the same key, then he may align them in such a way that he gets monoalphabetically encrypted columns. Note that this is a ciphertext-only attack. However it needs lots of messages.

Note that operators impede this attack by changing the key (or initial position) for each message. Nevertheless in some scenarios they have to send many messages, think of war. Then with high probability the cryptanalyst will observe many ciphertexts that are produced by the same rotor positions, not necessarily at the same position in the text. He identifies these concordances by extensive calculation of coincidence indices.

Identification of a Fast Rotor

Assume that the set of rotors is known but not their actual choice. Assume that the last rotor at the output side steps by one position with each letter, and that the other rotors move infrequently. The attacker has no known plaintext.

Enumerate the rotors from 1 (= input rotor, slow) to q (= output rotor, fast), and assume the current flows from left to right as in Figure 6.

Now assume we have a ciphertext section of length m where only rotor q moved, and for simplicity use the indices 1 to m for this sequence of ciphertext letters. The rotors 1 to $q-1$ together effect a constant substitution μ.

Therefore this part of the encryption follows the schema

$$\begin{array}{ccccccc}
a_1 & \mapsto & b_1 := \mu(a_1) & \mapsto & \rho_q^{(z_1)}\mu(a_1) & = c_1 \\
a_2 & \mapsto & b_2 := \mu(a_2) & \mapsto & \rho_q^{(z_1+1)}\mu(a_2) & = c_2 \\
\vdots & & \vdots & & & \vdots \\
a_m & \mapsto & b_m := \mu(a_m) & \mapsto & \rho_q^{(z_1+m-1)}\mu(a_m) & = c_m
\end{array}$$

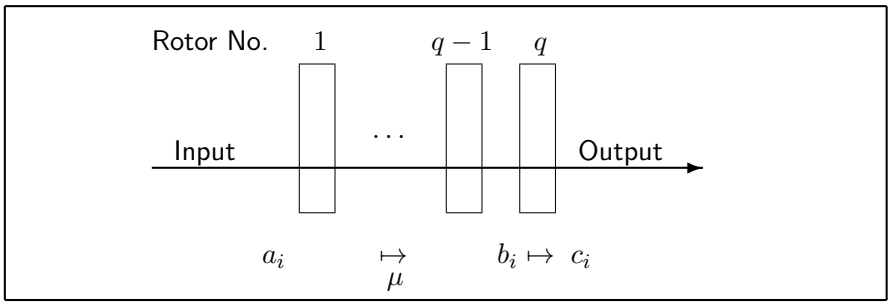

Figure 6: *Identifying the fast rotor*

Here $b = (b_1, \ldots, b_m) \in \Sigma^m$ is a monoalphabetic image of $a = (a_1, \ldots, a_m)$. We can also look at b "from the other side":

$$\begin{aligned} b_1 &= \left[\rho_q^{(z_1)}\right]^{-1}(c_1) \\ b_2 &= \left[\rho_q^{(z_1+1)}\right]^{-1}(c_2) \\ &\vdots \quad\quad \vdots \\ b_m &= \left[\rho_q^{(z_1+m-1)}\right]^{-1}(c_m) \end{aligned}$$

These formulas enable an exhaustion of the p choices for rotor q and of the n choices for its initial position z_1.

- A wrong choice of the rotor or its initial position makes b look as a random text having coincidence index $\varphi(b) \approx \frac{1}{n}$.
- For the correct choice b is a monoalphabetically encrypted meaningful text having coincidence index $\varphi(b) \approx \kappa_M$, the coincidence index of the plaintext language.

This observation may lead to the identification of the fast rotor and its state for this section of the text at the price of $n \cdot p$ calculations of coincidence indices of texts of length m. But note that the coincidence test for $m = 26$ has little power, it will miss most positive events.

Remarks

1. In principle the method works at each position of the text. Therefore the very beginning of the text is worth a try.
2. In the unfavourable case one of the other rotors moved during the encryption of the m letters. Then the intermediate ciphertext b consists of two different monoalphabetic pieces. With a bit of luck this also leads to a somewhat conspicuous coincidence index.

Continuation of the Attack

As soon as the fast rotor is identified we can strip its effect off like a superencryption. In this way the intermediate ciphertext $(b_1, \ldots, b_m)$ extends to a ciphertext $c' \in \Sigma^r$ that is the result of encrypting the plaintext a by a much simpler machine.

If for example the rotors move like an odometer, and if the ciphertext is long enough ($\approx n^2$), then in a similar way we can identify the next rotor and strip its effect off.

Or we try to cryptanalyze the monoalphabetic parts of c' that we expect $\lfloor \frac{r}{n} \rfloor$ in number of length n plus one or two fragments of total length r mod n.

We also might first try to find the locations were the second rotor moves.

Known Plaintext Attack

Assume we know or guess a piece of plaintext $a = (a_1, \ldots, a_m)$, say a probable word. An essential step is finding text chunks with identical numerical patterns, also called **isomorphs**. Therefore this attack is known as **Method of Isomorphs**. More generally looking at an intermediate step of an encryption algorithm from both sides, is called **Meet-in-the-Middle**.

Identification of a Fast Output Rotor

If we have a piece of known plaintext we may identify a fast rotor by simple pattern comparisons without calculating coincidence indices: Check if the intermediate text $(b_1, \ldots, b_m)$ shows the same numerical pattern as $(a_1, \ldots, a_m)$.

Identification of a Fast Input Rotor

Known plaintext $a = (a_1, \ldots, a_m)$ also allows the identification of the fast rotor for a *reverse* odometer control where the left rotor is the fast one. In this case we consider the situation of Figure 7.

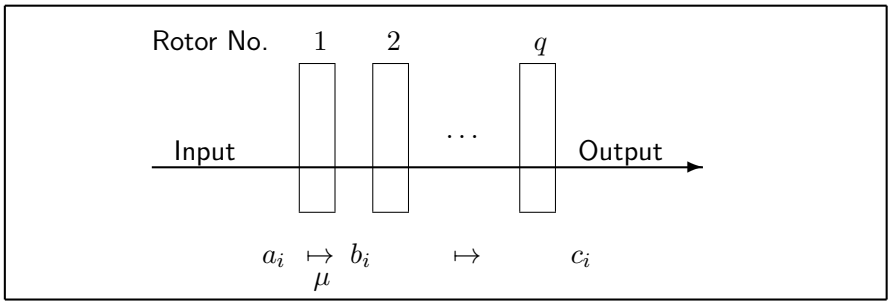

Figure 7: *Identifying a fast input rotor*

This part of the encryption follows the schema

$$\begin{array}{lllll} a_1 & \mapsto & b_1 := \rho_1^{(z_1)}(a_1) & \mapsto & \mu(b_1) = c_1 \\ a_2 & \mapsto & b_2 := \rho_1^{(z_1+1)}(a_2) & \mapsto & \mu(b_2) = c_2 \\ \vdots & & \vdots & & \vdots \\ a_m & \mapsto & b_m := \rho_1^{(z_1+m-1)}(a_m) & \mapsto & \mu(b_m) = c_m \end{array}$$

Here $b = (b_1, \ldots, b_m)$ is a monoalphabetic image of $c = (c_1, \ldots, c_m)$. We try all p rotors in all their n initial positions until the numerical patterns of b and c coincide.