CSC B58: Bombe Machine Breakdown

James Camano

April 2, 2019

(**Last Edited - April 2, 2019**)

- added components section
- added algorithm section

This document sets out to describe the components of the *Bombe machine* - the encryption setting-cracker counterpart to Enigma.

1 Introduction

Project Enigma sets out to imitate both the German *Enigma* text cipher machine, and its mechanical counterpart - the Bombe machine.

This version of the Bombe machine will take advantage of the incremental patterns used by the Enigma machine, and will iteratively filter through potential solutions by method of contradiction.
In this paper, we will assume the same definitions from the Enigma document.

2 Components

The Bombe machine consists of:

- A state counter, similar to that of Enigma's Rotor Component,
- A letter counter / incrementer circuit
- A checker circuit that uses simple lexicographic arithmetic to deduce contradictions.

3 Algorithm

Suppose that we are given the following sequence of letters S_E, outputted by Enigma: $S_E = \langle s_0, s_1, s_2, ..., s_n \rangle$, where the $s_i \in \Sigma$.
To decrypt any message from Enigma, we need only look at the subsequence $\langle s_0, s_1, s_2 \rangle$, and determine a starting rotor configuration $R = (1, x)$ for which the following are true:

$$\hat{f}(s_0, R) = A \tag{1}$$
$$\hat{f}(s_1, g(R)) = B \tag{2}$$
$$\hat{f}(s_2, g(g(R))) = C \tag{3}$$
$$\tag{4}$$

This comes from the fact that we have constrained any message of the enigma machine to be encrypted with the prefix "ABC".
With this in mind, the algorithm is:

```
Algorithm: Bombe(S)
Input: S: A message that is assumed to be encrypted by the Enigma machine.
Result: Finds and returns the correct starting rotor position of Enigma, if one exists.
rotor_position:= 0;
found_correct_position:= $false$;
$S := \langle s_0, s_1, s_2 \rangle$;

while (rotor_position $\neq 26$ and not(found_correct_position)) do
    if (decode(S[0:2], x) == $\langle A, B, C \rangle$) then
        found_correct_position:= $true$;
    else
        rotor_position++;
    end
end

if (found_correct_position) then
    return rotor_position ;
else
    return 11111 ;
end
```

Algorithm 1: High-Level Algorithm for Bombe machine.

```
Algorithm: decode(S, x)
Input: S: The 3-letter required prefix for Encrypted messages from Enigma.
Result: Checks the given sequence $S = \langle s_0, s_1, s_2 \rangle$ and back-translates it. Returns
        whether or not the shifting value $x$ is the correct rotor starting position.
$s_0 = S[0]$;
$s_1 = S[1]$;
$s_2 = S[2]$;

$S' := \langle \mathbf{lexshift}(s_0, -x), \mathbf{lexshift}(s_1, -(x+1)), \mathbf{lexshift}(s_2, -(x+2)) \rangle$;

return $S' == \langle A, B, C \rangle$;
```

Algorithm 2: The **decode**(S, x) algorithm. Note that the function **lexshift**(s, x) represents a lexicographical shift in the direction of the sign of x.

4 Justification for the Algorithm

Obviously, this algorithm is heavily inspired by the realization that known repeated words like *Wetterbericht* (Weather Report) were used to deduce the Enigma code (at least, for that day.)

This justification relies on the fact that for a specific rotor configuration $R = (m, n)$, the sequence $\langle f(A, R), f(B, g(R)), f(C, g(g(R))) \rangle$ is unique for each configuration R (this is immediately obvious if one is to consider $f(A, R)$).

Thus, if one assumes a correct Enigma message, then one may deduce the rotor value inital $n \in \{0, 1, \ldots, 25\}$ by finding such n that satisfies that in the decryption requirements.

The message constraints that requre the message to be appended with an extra encrypted B and C could be interpreted as “insurance letters”: knowing that the first three letters are what you expect is better than having only one.

5 Additional Notes