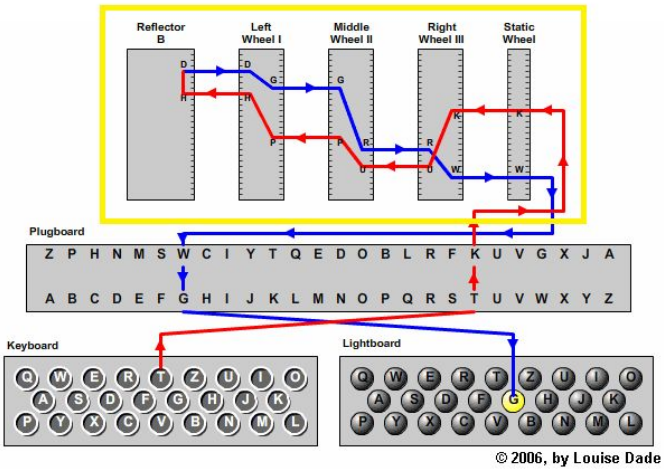
Parallel Enigma Cracking

Evan Mesterhazy (etm2131)
Samuel Meshoyrer (sm3604)

Introduction

The proposed project is a parallelized cypher-text only solver of the Enigma machine. To constrain the complexity of the project, we will initially focus only on the rotor settings of the machine. Time permitting, we can expand the project scope to incorporate the plugboard settings as well. The given cypher-text will be decrypted using all possible initial rotor configurations. Each candidate result will be compared against a reference document using a fitness function, such as single letter frequencies or trigraph scores, to identify potential valid rotor configurations. The highest ranked results will then be used to recover the rotor ring settings, and the top candidate plaintexts will be output for manual consideration.

Enigma Rotors



The fundamental operation of the enigma rotor is a *substitution cypher*, a cypher which substitutes each letter for another. Early substitution cyphers, like the Caesar cypher, employed simple rules, e.g. offset each letter by 3. The substitution cyphers implemented by the rotors use a more complicated permutation cypher. A key flaw of the enigma machine is that no enciphered letter is ever mapped back onto itself. This flaw enables sophisticated known-plaintext attacks like those famously employed by Alan Turing and others at Bletchley Park.

However, this flaw does not affect ciphertext-only attacks on the Enigma cipher, which remain computationally expensive. The full keyspace of the Enigma machine consists of the chosen rotors and their ordering, the starting position (indicator setting) and ring setting of each rotor, and the plugboard settings. The original German army “M3” Enigma machine used 3 rotors chosen from a set of 5, for a total of 10 combinations. Each set of 3 rotors could be permuted

into 6 different orderings for a total of 60 possible rotor arrangements. Later iterations of the Enigma machine, like the “M4” Naval Enigma chose 4 rotors from 8, resulting in 1,680 initial rotor orderings. In addition to the rotor selection, each rotor has 26 possible indicator settings, yielding 26^3 possible indicator settings for the three-rotor Enigma. The ring setting is also significant for two of the three rotors, and has 26 possible settings. Thus, before considering the plugboard settings, the 3-rotor Enigma has $60 * 26^5 = 712,882,560$ possible configurations. The 4-rotor Enigma has a significantly expanded keyspace of $1,680 * 26^7 = 13,493,441,095,680$ configurations excluding the plugboard.

Implementation

We will use a series of news articles from The Economist to form the baseline dataset used to compute letter frequencies. We will then encrypt an article not in the dataset with a random configuration. We will attempt to recover the configuration by decrypting the cyphertext in parallel using all possible starting rotor combinations. Candidate plaintexts will be compared to the reference documents using a fitness function, and the highest-ranked candidates will be used to recover the ring settings. The highest ranked rotor and ring setting pairs will be output for the user along with the associated candidate plaintexts.

Each decryption takes a constant amount of work - the full cyphertext needs to be decrypted. This special case of parallelization will allow us to compare both simple (parMap) and more complicated parallelization strategies. Our goal will be to find an optimal parallelization strategy for this “constant-work” case.

Possible Expansion

Although we plan to focus on cracking Enigma messages enciphered without employing the plugboard settings, the algorithmic complexity and runtime of the project can be expanded by attempting to recover paintexts from messages encrypted using the plugboard. Historically, the Enigma plugboard used 10 wires to swap 20 of the 26 letters. The total number of plugboard settings is $\frac{26!}{(10!\,6!\,2^{10})} = 150,738,274,937,250$. James Gillogy’s 1995 paper in Cryptologia[1] describes a stair-stepping algorithm to recover the plugboard settings without searching the entire keyspace, which would serve as a reference. The M4 Message Breaking Project[2] employed a similar technique in the mid 2000s to crack unbroken German Naval messages using a distributed network of hundreds of computers.

We believe that recovering a full set of 10 plugboard settings will be too computationally expensive to run on a single machine, even with modern multi-core processors. However, we believe we could meaningfully expand our project to recover messages enciphered with 2-3 plugboard settings since the underlying algorithm is not affected by the number of plugs.

---

[1] Gillogly, J., 1995. CIPHERTEXT-ONLY CRYPTANALYSIS OF ENIGMA. Cryptologia, 19(4), pp.405-413.
[2] M4 Message Breaking Project, http://www.bytereef.org/m4_project.html. Accessed 22 Nov. 2020