Martin Schrimpf 14829265

Takuma Sato 1259130

Highly Secure System – 2014 Semester 2

Lecture: Andrew Ensor

Assignment one: Symmetric Cipher

Chosen topic: Enigma machine

1. History and Usage

The Enigma machine is a rotor cipher machine which was widely used during the period between 1930 and 1950 for encryption. The core mechanism of Enigma machine was first developed in 1918 by Arthur Scherbious, a German engineer in the commercial companies in secure communication. Over the decades after this invention, German military has produced its own version with more complex plugs and electronic circuits to avoid interceptor to decrypt the message.

The first break though of cryptanalysis against commercial Enigma was first done by three Polish mathematicians in 1929. They have later created a mechanical device called the “Bomba” that simulates 26*26*26 positions of the 3 Enigma rotors. By the year 1939, Bomba could simulate up to 6 rotors. When these mathematicians fled abroad during WW2, their knowledge and materials were passed to polish allies and most notably the Britain's Government Code and Cipher School where Alan Turing was working. After 3 months, the famous “Turing Bomb” was created which had 36 rotors to simultaneously test possible cases of twelve 3-rotor Enigma machine. The machine used a known-plaintext attack on words such as “General” or “To the group”.

2. Description of the Algorithm

Our implementation of the algorithm contains mechanism to adjust the number of rotors and the initial rotation of the rotor (key), with an option to turn using a fixed reflector off and on through subclassing.

2.1 Encryption

Each rotor is given a scrambled sequence of the alphabet along with an integer denoting the current rotor position. All communication between the rotors is done through int values of the chars (indices) for efficiency and clarity. The encryption starts by initializing each rotors to a given position according to the key (e.g. key for 3 rotors can be “ABC”). In the following, the algorithm adds the index of the current rotor character to the index of the first letter of the plaintext (contrary to doing a lookup of the plaintext-character). This index is then used as the input for the next rotor to be substituted again and so on and so forth until it goes through all rotors or reaches the reflector. The reflector is a fixed rotor, meaning there are only 13 possible permutation that work both ways (e.g K↔M). After the permutation using the reflector, the output travel backwards till the first rotor. After a letter is encoded, the first rotor is shifted by 1, and if a full rotation of a rotor n was reached then the following rotor n+1 is also rotated and so forth. Repeat until every letter is encrypted.

Martin Schrimpf 14829265

Takuma Sato 1259130

2.2 Decryption

Without a reflector: Rotate rotors in the same way using the same key but do the conversion from most outer rotor to most inner rotor (contrary to encryption where the conversion goes from most inner to most outer rotor).

With a reflector: Use the same algorithm with the same key.

2.3 Strengths

When using the reflector, the same algorithm can be used for encryption as well as decryption.

Has a high number of possible keys for a polyalphabetic substitution cipher (in particular, the amount of possible keys is exponential to the number of rotors which can make a brute-force attack difficult).

2.4 Weaknesses

The reflector was once considered as a great improvement as it will double the number of rotors, and the same Enigma machine could be used to decipher without switching between modes. On the other hand, a vulnerability has been exploited as it turns out that using reflector causes any letter to never encrypt itself (e.g. G will never become G).

It is vulnerable against cribs: known-plaintext attack.

The keys (initial rotor setting) were often not really random (e.g. AAA, initials of girlfriend, abbreviations)

3. Cryptanalysis

Our chosen cryptanalysis attempts to find the key of the given cipher text with a “sophisticated” brute-force attack and a frequency analysis of the plaintext against a bi-gram chart. This algorithm assumes that the encoded plaintext is of the English language and that the Enigma machined used to encrypt the plain text is available for observation, i.e., our algorithm relies on the fact that we know the number of rotors used and the letter sequences on each rotors.

Start by preparing every possible key for number of rotors used to encrypt the plain-text (E.g. if number of rotor is 3 than generate key from “AAA” to “ZZZ”). For each key generated, decrypt the text then validate the decrypted text before passing on the frequency analysis. Most importantly, the validation process checks that no letter in the plaintext has been encoded to the same letter in the cipher text and thereby eliminates impossible keys early saving computation time on the frequency analysis (which is more resource-heavy). After the validation, the decrypted text goes through a frequency analysis against a bi-gram chart to calculate a score for the key that was used to decrypt the cipher text (Note: We also tested q-grams, hence the classes QGramCalculator and QGramIndexCalculator – bi-grams turned out to be the most reliable while also being the most efficient). The score is the greater the more likely the found bi-grams are in the English language. Finally, the key resulting in the plaintext with the highest score is used.

Martin Schrimpf 14829265

Takuma Sato 1259130

4. Evaluation

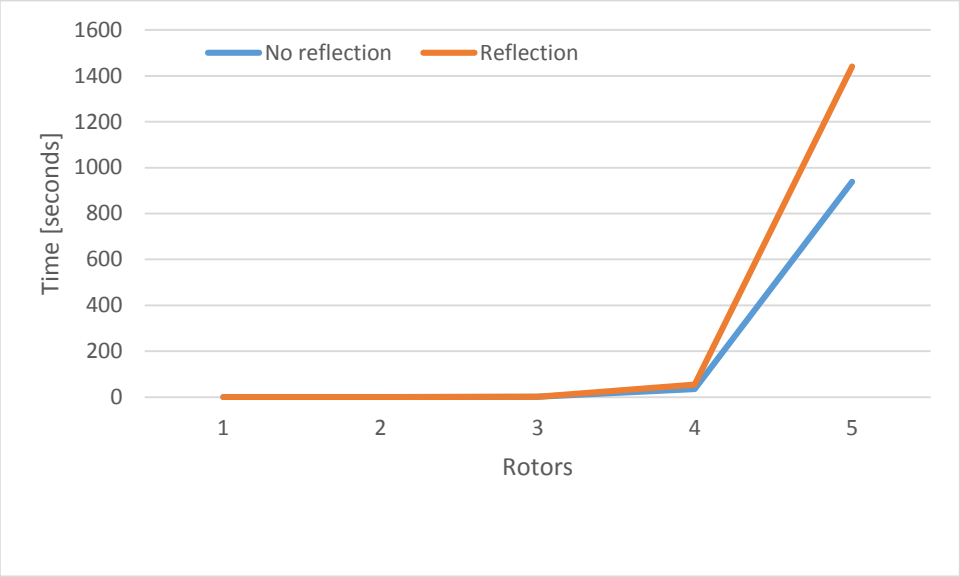

Figure 1: Duration of cryptanalysis attacks with a single 1.7 GHz processor on 56 characters of text encrypted with an Enigma machine

Graph above shows the result of cryptanalysis on cipher texts which were encrypted using a standard Enigma machine without the reflector. From the result of our cryptanalysis it is clear that every rotor adds exponential amount of time against brute-force and frequent analysis. This algorithm has time complexity of O(n^x).

5. Instructions

Run the provided JAR-file using *java –jar Enigma.jar*.

The program's help prints the following
Usage: java -jar Enigma.jar

-e|--encrypt|-d|--decrypt|-a|--attack

-i|--input <input text> [-k|--key <key>] [-r|--rotors <amount of rotors>]