University of Glasgow | School of Computing Science

Assignment: Decrypting Ciphertexts Using Cryptographic Attacks
A Study of Known Plaintext Attack (KPA) and Ciphertext-Only Attack (COA)

By
Ayush Jaipuriyar
3043047J

Under the guidance of
Dr. Nguyen Truong

COMPSCI5079
School of Computing Science
University of Glasgow
2024-2025

Contents

1 Introduction

This report presents a comprehensive study of two cryptographic attacks — the Known Plaintext Attack (KPA) and the Ciphertext-Only Attack (COA) — applied to decrypt two ciphertexts produced by an 8-rotor encryption machine. The rotor machine, implemented in Java, uses modular arithmetic and rotor transformations to encode plaintext into ciphertext using a secret key drawn from a predefined list of common passwords.

The goal of the Known Plaintext Attack (KPA) is to exploit partial knowledge of the plaintext — specifically, the first two characters — to systematically test potential keys through a dictionary attack. By iterating through all possible passwords, the attack identifies the key that produces a plaintext matching the known fragment, ultimately decrypting the entire ciphertext.

In contrast, the Ciphertext-Only Attack (COA) operates without any information about the plaintext. Instead, it relies on statistical properties of the English language, such as letter and word frequency analysis, bigram and quadgram detection, and dynamic threshold calculations, to assess whether a decrypted output is a valid English sentence. This method ranks candidate plaintexts by score, allowing the most likely key to be identified.

Both attacks leverage the `Rotor96Crypto.java` class for encryption and decryption, while the COA further integrates probabilistic models and multithreading techniques to handle large search spaces efficiently.

The report not only outlines the implementation of both attacks but also discusses the probability of key collisions, the methodology for distinguishing valid plaintexts from random strings, and the unicity distance — the theoretical minimum ciphertext length required for unambiguous decryption. Experimental results are compared to theoretical predictions to assess the effectiveness of both attacks.

Ultimately, this study aims to provide insights into the vulnerabilities of rotor-based encryption systems and the effectiveness of classic cryptanalytic techniques in compromising such ciphers.

2 Known Plaintext Attack (KPA)

2.1 Methodology

The KPA attack assumes knowledge of the first two plaintext characters, given as `"We"`. The key identification process is outlined as follows:

1. **Dictionary Attack:** The ciphertext is decrypted iteratively using each key from a provided password file. This file contains a list of common passwords, and each password serves as a potential key for the rotor encryption machine. The attack methodically tests each password, attempting to uncover the correct decryption key by brute force.

2. **Decryption Attempt:** The `Rotor96Crypto.java` class is used to decrypt the ciphertext with each key. This class simulates the functioning of an 8-rotor encryption machine, where the key determines the initial rotor positions and the transformations applied to the plaintext during encryption. For every key tested, the program uses the machine's decryption function to produce a candidate plaintext.

3. **Plaintext Matching:** For each decryption attempt, the resulting text is checked to see if it starts with the known plaintext fragment `"We"`. This step leverages the information provided for the Known Plaintext Attack (KPA), where the first two characters of the plaintext are given. If the decrypted output begins with these characters, the corresponding key is marked as a potential match.

4. **Key Verification:** Keys that produce matching plaintexts are logged for further analysis. Given the possibility of key collisions — where multiple keys may produce plaintexts starting with `"We"` due to the mathematical structure of the rotor machine — all candidate keys are recorded. The correct key is later identified by examining the decrypted outputs for grammatical correctness, sentence structure, and coherence in English, ensuring the final selection is the most plausible plaintext.

The encryption mechanism of the rotor machine is modeled by the following equations:

$$c = e[p + f] \mod 96$$
$$p + f = d[c] \mod 96$$

where:

- p is the plaintext character, representing the numeric value of the character in the 96-character set,
- c is the ciphertext character, similarly represented as a numeric value,
- f is the rotor offset, which shifts characters according to the rotor's internal configuration and current position,
- $e[]$ is the encryption mapping function that translates plaintext to ciphertext based on rotor positions,
- $d[]$ is the decryption mapping function, the inverse of $e[]$, used to reverse the transformation and recover plaintext from ciphertext.

The $\mod 96$ operation accounts for the 96-character output space of the machine, ensuring that all transformations wrap around the rotor set, preventing character values from exceeding the defined range.

In practice, the rotor machine works by advancing the rotors with each character encrypted, creating a dynamic offset that changes with every step. This results in a polyalphabetic cipher, where identical plaintext characters can map to different ciphertext characters depending on rotor positions — significantly complicating frequency analysis.

The decryption process, therefore, must reverse not only the static mappings defined by the key but also account for the rotational shifts introduced during encryption. This complex interplay of modular arithmetic and rotor progression is what allows for the possibility of multiple keys yielding plaintexts with matching prefixes, a phenomenon explored further in the probability analysis section.

2.2 Results

The results of the Known Plaintext Attack (KPA) are summarized below, showcasing the identified key and the successfully decrypted message:

- **Identified Key:** `giveme`

  The key `giveme` was identified through a dictionary attack, where each password from the provided password file was tested systematically using the rotor machine's decryption method. The correct key was determined by checking if the resulting plaintext started with the known fragment `"We"`. Once the key `giveme` produced a matching output, the full plaintext was decrypted and verified for coherence and grammatical correctness.

- **Decrypted Message:** `Welcome to the Cryptography and Secure Development course 2024-2025, Ayush. This is the secret message (i.e., plaintext) for you to be decrypted. Please make sure that this message is only designed for you by checking whether your ID number 3043047 is correct :) Good luck!`

The successful decryption of the message highlights the effectiveness of the KPA approach. By leveraging the given plaintext fragment and systematically testing potential keys, the correct key was efficiently identified. Additionally, the verification step using the recipient's ID number provides a clear indicator that the decrypted output is accurate and not a false positive.

These results not only demonstrate the practical application of the KPA but also underscore the importance of validating decrypted messages, especially in scenarios where multiple keys may produce plausible-looking outputs. The next section will delve into the theoretical and experimental analysis of multiple key possibilities and their associated probabilities.

2.3 Multiple Key Possibilities

The modular arithmetic and rotor transformations may yield multiple keys that result in the same initial plaintext sequence. For instance, two distinct keys, `password123` and `secure456`, could both lead to a decrypted message starting with `"We"`.

The decryption formula for the rotor machine is given by:

$$p + f = d[c] \mod 96$$

where:

- p is the plaintext character
- c is the ciphertext character
- f is the rotor offset (derived from the key)
- $d[]$ represents the decryption mapping
- $\mod 96$ reflects the rotor's 96-character space

Rearranging to solve for p:

$$p = (d[c] - f) \mod 96$$

Due to the modulo operation, different keys can produce offsets f_1 and f_2 such that:

$$f_1 \equiv f_2 \mod 96$$

leading to the same plaintext result.

2.3.1 Example:

Consider the following:

- Ciphertext character $c = 75$ (ASCII character 'K')
- Decryption mapping $d[c] = 50$
- Two possible rotor offsets:
  - $f_1 = 10$ (derived from the key `password123`)
  - $f_2 = 106$ (derived from the key `secure456`)

Calculating the plaintext character:

$$p = (50 - 10) \mod 96 = 40$$

$$p = (50 - 106) \mod 96 = (-56) \mod 96 = 40$$

Although the rotor offsets differ, the modulo operation causes the results to "wrap around", producing the same plaintext character.

2.3.2 Collision Condition:

The keys produce the same plaintext whenever:

$$f_1 \equiv f_2 \mod 96$$

This means adding any multiple of 96 to a rotor offset will not affect the plaintext output. Therefore, it is possible for two or more distinct keys to generate decrypted messages with the same initial characters, such as `"We"`.

Example:

- Key: `password123` → `"We shall overcome..."`
- Key: `secure456` → `"We saw the sunrise..."`

Although both outputs start with `"We"`, further analysis based on English language characteristics can help distinguish the correct key.

2.4 Probability of Multiple Keys

The probability of multiple keys producing the same initial plaintext sequence can be estimated using the following formula:

$$P = \frac{N}{96^L} \tag{1}$$

where:

- N is the number of possible keys tested (9473 keys from the provided password file),
- 96 represents the size of the character set used by the rotor machine, corresponding to the 96 printable ASCII characters,
- L is the length of the known plaintext segment (2 characters, "We").

Substituting the values:

$$P = \frac{9473}{96^2} = \frac{9473}{9216} \approx 1.028 \tag{2}$$

The calculated probability, $P \approx 1.028$, suggests that there is a significant chance of encountering key collisions — where multiple keys produce decrypted outputs starting with the same two-character sequence.

Interpretation: A probability greater than 1 implies that, on average, more than one key is likely to produce the same initial plaintext fragment. This occurs because the number of possible key combinations (9473) exceeds the number of possible two-character plaintext combinations ($96^2 = 9216$). As a result, it becomes statistically probable that different keys will generate identical plaintext prefixes, leading to key collisions.

Implications: This phenomenon highlights a fundamental limitation of using short plaintext segments in Known Plaintext Attacks. The overlap in possible plaintext outputs means that simply finding a match with the known fragment ("We") does not guarantee the correct key has been identified — multiple candidates may need further validation.

To resolve this, additional checks — such as assessing the grammatical correctness of the full decrypted text or using personalized markers like the recipient's ID number — become crucial for distinguishing the correct key from false positives.

This probability estimate also emphasizes the importance of testing longer plaintext sequences, as increasing L exponentially reduces the likelihood of key collisions, given the 96^L denominator grows rapidly with each additional character.

3 Ciphertext-Only Attack (COA)

3.1 Methodology

The Ciphertext-Only Attack (COA) is a cryptanalysis technique where the attacker attempts to deduce the plaintext and the key using only the given ciphertext, without any prior knowledge of the plaintext. For this assignment, the COA was implemented by systematically testing a list of potential keys to identify the correct plaintext. The approach consisted of the following structured steps:

1. **Dictionary Attack:** Each key from the provided password file was tested iteratively. The file contains 9473 possible keys, all of which were used as input to attempt decryption of the ciphertext.
2. **Decryption:** The decryption process was handled by the `Rotor96Crypto.java` class, which simulates an 8-rotor encryption machine. Each key was passed into the decryption method:

```
public String encdec(int mode, String key, String text)

mode = DEC (2), indicating decryption
key = current password from dictionary
text = ciphertext provided
```

The output for each key was stored for further evaluation.

3. **Plaintext Detection:** Given the absence of known plaintext, plaintext detection relied on a scoring mechanism designed to identify outputs most likely to be valid English sentences. The scoring system incorporated the following techniques:
   - **Letter and word frequency analysis:** The frequency of individual letters and common English words (such as "the," "and," "is") was compared against known statistical distributions of the English language.
   - **Detection of bigrams and quadgrams:** Recognizing common letter pairs (bigrams) like "th" and "he" and four-letter combinations (quadgrams) like "tion" and "ther" helped identify linguistically plausible text.
   - **Dynamic threshold calculations:** To detect unambiguous decoding, a dynamic threshold was calculated based on the mean and standard deviation of scores. This threshold was used to filter out low-probability plaintexts, ensuring only high-confidence candidates were considered.
4. **Key Identification:** After scoring all decrypted outputs, the key corresponding to the highest-scoring plaintext was selected as the most likely key. The plaintext associated with this key was then recorded as the decrypted message. The program also grouped scores into ranges (e.g., every 100 points) to check for score-based clustering, further validating the results.
5. **Multithreading Optimization:** To enhance performance, multithreading was implemented using Java's `ExecutorService` and `CountDownLatch`. This allowed multiple keys to be processed concurrently, significantly reducing the overall runtime of the attack.

The combined use of frequency analysis, n-gram detection, and dynamic scoring thresholds provided a robust method for identifying correct plaintext without any initial knowledge, allowing for effective execution of the COA.

3.2 Results

- **Identified Key:** `octopus 2025-02-19T15:58:14.8681378366 MailTo: 3043047J@student.gla.ac.uk. 8366 is a random number generated only for you, Jaipuriyar, in the second task of the Project Assignment. In this Ciphertext Only Attack (COA) of the Project Assignment, you are expected to decode it with the assumption that this plaintext is in English sentences; however, it can contain some special characters such as ! @ £ $ \& * ( ). Finger-crossed!`

The identified key, `octopus`, successfully decrypted the given ciphertext into a coherent and meaningful plaintext message. The decrypted message confirmed the student's unique identifier (3043047) and included a timestamp (2025-02-19T15:58:14.8681378366), adding further authenticity to the plaintext.

The presence of special characters like `textbackslash ! \@ \£ \$ \& \* \( \` highlights the rotor machine's capability to handle the full ASCII character set, reinforcing the necessity for the scoring system to distinguish valid plaintext from random noise. The dynamic threshold calculation ensured that the selected plaintext scored significantly higher than others, providing confidence in the correctness of the decryption.

Additionally, score-based clustering was employed to verify that the `octopus` key stood out among others, with no close competitors in its score range. This further corroborated the unambiguous decoding of the ciphertext.

3.3 Method for Identifying Valid Plaintext

A plaintext is considered valid if its score exceeds a dynamic threshold, calculated using the mean and standard deviation of letter and word frequencies. The scoring mechanism is implemented through a step-by-step analysis of decrypted text, incorporating the following:

- **Word Frequency Analysis:** The program checks for the presence of common one-letter words ("a", "i"), two-letter words ("of", "to"), three-letter words ("the", "and"), and four-letter words ("that", "with"). Each matching word contributes to the score based on its length.

- **N-gram Detection:** Bigrams (e.g., "th", "er") and trigrams (e.g., "the", "and") are detected within the plaintext, with each occurrence adding a weighted score. Quadgrams (e.g., "tion", "ther") further boost the score for realistic English text.
- **Penalty for Special Characters:** Special characters such as !, @, and $ incur slight score penalties, preventing random or nonsensical outputs from falsely achieving high scores.
- **Length-Based Scaling:** To prioritize longer, more coherent plaintexts, the score is scaled using a logarithmic function:

$$\text{score} = \text{raw score} \times \log_{10}(\text{length of text} + 10) \tag{3}$$

  This ensures that longer texts with proper word patterns are favored over short, potentially misleading outputs.
- **Dynamic Threshold:** Once all plaintext scores are computed, the mean () and standard deviation () are calculated. The dynamic threshold is then defined as:

$$T = \mu + k\sigma \tag{4}$$

  where is an adjustable constant used to set the strictness of the threshold. Only plaintexts with scores exceeding are considered valid.

Additionally, score-based clustering is used to identify patterns within the score distribution, further isolating high-confidence plaintext candidates. By grouping scores and analyzing the frequency of English letter combinations, the method ensures that only the most plausible plaintexts are selected for final verification.

3.4 Experiment: Minimum Ciphertext for Unambiguous Decoding

Procedure:

1. Decrypted progressively larger portions of the ciphertext, starting from small segments and incrementally adding characters.
2. For each segment, all candidate plaintexts were scored using the dynamic threshold method.
3. Measured the point at which only one plaintext output passed the dynamic threshold, indicating unambiguous decoding.
4. Recorded the minimum number of characters required for this unambiguous result.

Results:

- **Theoretical Minimum Characters Needed (Unicity Distance):** The unicity distance is the minimum number of ciphertext characters required for unambiguous decoding, calculated using the formula:

$$U = \frac{H(K)}{H(P)}$$

  where:
  - bits (entropy of the key space),
  - is the entropy of the plaintext, typically around 1.5 bits per character for English text.

  Thus:

$$U = \frac{13.22}{1.5} \approx 8.81$$

- **Actual Characters Needed:** After testing various portions of the ciphertext, it was found that a minimum of `110` characters was required to produce a single plaintext output exceeding the dynamic threshold. This value, higher than the theoretical unicity distance, reflects the additional complexity introduced by score-based clustering, dynamic thresholds, and statistical noise.

4 Conclusion

This report successfully demonstrated the decryption of ciphertexts using both Known Plaintext Attack (KPA) and Ciphertext Only Attack (COA) methods. The combination of dictionary attacks, cryptanalysis techniques, and probabilistic reasoning proved effective in identifying key candidates and recovering the corresponding plaintext messages.

In the KPA, the dictionary attack methodology, which utilized common passwords and the first two known plaintext characters, was able to efficiently narrow down the possible keyspace, allowing us to determine the key used for encryption. This technique relies heavily on the assumption that the plaintext shares common patterns or known phrases, making it a powerful approach when such information is available.

The COA, on the other hand, used a more sophisticated approach, involving cryptanalysis and frequency analysis techniques such as letter/word frequency detection, bigram/quadgram detection, and dynamic threshold calculation. By leveraging multithreading and parallel processing, we were able to efficiently process a large number of potential keys and evaluate their likelihood of generating meaningful plaintext. This approach demonstrated the importance of scoring mechanisms that prioritize high-probability candidates based on language models and statistical patterns, further improving the detection process.

One of the key concepts explored in this report was the unicity distance, which played a significant role in determining the minimum ciphertext length required for unambiguous decryption. The unicity distance is defined as the length of ciphertext at which there is only one possible key that could produce a given plaintext. By analyzing different ciphertext lengths, we were able to determine the minimum required length for each method and understand the limitations of the decryption techniques.

The probabilistic reasoning and scoring mechanisms used in the COA were particularly instrumental in refining our search for the correct key. The ability to cluster results based on score ranges and calculate standard deviation-based thresholds provided a more focused and efficient search, reducing the complexity of the problem. Additionally, the dynamic adjustment of thresholds based on statistical analysis allowed for a more flexible and adaptive decryption process, especially when faced with ciphertexts of varying characteristics.

Ultimately, the combination of these methods demonstrated the feasibility of decrypting ciphertexts even in situations with limited prior knowledge, highlighting the strength of cryptanalysis and the importance of statistical techniques in modern cryptography. Through the use of well-established cryptanalytic methods and computational tools, we were able to recover plaintexts and make informed decisions regarding the likelihood of different key candidates.

In conclusion, this report not only reinforced the practical application of KPA and COA in cryptanalysis but also illustrated the crucial role of mathematical and statistical methods in the field of cryptography. The use of dynamic scoring, clustering, and probabilistic analysis enhanced the decryption process and paved the way for further exploration into more advanced cryptanalysis techniques. As encryption methods evolve, the need for more robust cryptanalysis strategies remains crucial, and the methods presented in this report can serve as foundational tools in the ongoing effort to break more complex encryption systems.