# CSC662

# COMPUTER SECURITY

# MARCH 2025 – AUGUST 2025

# Reverse Frequency Cipher

| GROUP 4<br>NBCS2308A | | |
|---|---|---|
| **BIL** | **MATRIC NO** | **NAME** |
| 1. | 2021223204 | MOHAMAD NUR SYAZWAN BIN MOHD NOOR |
| 2. | 2021288136 | MUHAMMAD ADAM FITRI BIN ABD RANI |
| 3. | 2022234426 | KHAIRUL SYAHMI BIN AZMAN |
| 4. | 2021435138 | MUHAMMAD ZULKIFLI BIN MOHD ZIN |

# TABLE OF CONTENTS

## 1. INTRODUCTION

The Reverse Frequency Cipher is a custom encryption prototype designed to hides the usual letter patterns found in English text. In this project, the cipher combines a dynamic, message-specific substitution with an optional Caesar shift to improve security. The motivation arose from the weaknesses of classical monoalphabetic ciphers like the Caesar and simple substitution cipher, which are easily broken by frequency analysis. By reversing the role of common and rare letters in each message, the prototype aims to produce ciphertext with an atypical letter distribution. An additional Caesar rotation layer further adds confusion by shifting the output alphabet. The overall goal is to explore a lightweight hybrid scheme that mitigates the telltale frequency patterns exploited in classical cryptanalysis while remaining easy to implement and invert. This report details the cipher's design, theoretical basis, and performance on example messages. Key objectives include evaluating how well the frequency inversion and Caesar hybrid conceal plaintext statistics and analyzing the strength and limitations of this approach against standard attacks.

## 2. PROBLEM ANALYSIS

Classical monoalphabetic ciphers preserve the plaintext's letter frequency in the ciphertext, making them vulnerable to analysis of character frequencies. In a simple substitution cipher, each plaintext letter consistently maps to a single ciphertext letter, so the frequency of each symbol in ciphertext is identical to that of the original letter. Attackers can exploit the uneven distribution of English letters (where E, T, A, O, I, N are most common) to guess mappings.

Frequency analysis was historically devastating: for example, a Caesar shift (which is a fixed rotation of the alphabet) can be cracked either by brute force of 25 shifts or by noting that the ciphertext's most frequent letter likely represents E in plaintext. More generally, an analyst can count single-letter, digraph, and trigram frequencies and compare them to expected English patterns to infer the substitution key. This well-known weakness means that classical ciphers are not secure by modern standards. Polyalphabetic designs like the Vigenère were introduced to counter this by using multiple cipher alphabets so that no single frequency distribution dominates. Likewise, polygraphic ciphers such as Playfair and Hill encrypt letter pairs or groups, mixing plaintext letters so that individual frequencies are masked. These approaches confirm that confusion and diffusion are essential: as Shannon (1949) observed, a strong cipher should diffuse plaintext statistics and introduce confusion in the letter mapping to frustrate statistical attacks. The Reverse Frequency Cipher directly addresses the classical frequency-leak problem. By inverting the frequency order of letters in each message's encryption alphabet, it tackles the monoalphabetic cipher's primary weakness. In effect, high-frequency plaintext letters are deliberately replaced with letters that are normally rare, and vice versa, attempting to mislead an attacker's assumptions. The optional Caesar shift further randomizes the output letter assignments across the alphabet.

## 3. OBJECTIVES

The objective of the project was to design and implement a prototype cipher that improves on classical substitution by dynamically altering letter mappings based on the input message's characteristics. Specifically, the Reverse Frequency Cipher aims to:

1. Generate a message-specific substitution map that flips the frequency ordering of letters, so that frequent plaintext letters are encrypted as infrequent letters;
2. Optionally apply a Caesar cipher rotation as a second layer of encryption for added security.
3. Provide correct decryption by reversing these steps.

By achieving these, the project demonstrates a hybrid encryption scheme that combines two classical techniques (substitution and Caesar shift) in a novel way. The expected outcome is a cipher that produces ciphertext with an anomalous frequency profile, thereby impeding straightforward frequency analysis. The project also aims to evaluate the cipher's performance: it should correctly recover plaintext (when given the map and shift) and meet course security criteria within a limited development timeframe. In summary, the Reverse Frequency Cipher prototype is intended as a proof-of-concept for a frequency-analysis-resistant cipher that remains simple enough for educational use.

## 4. LITERATURE BACKGROUND

Classical encryption techniques have well-documented vulnerabilities to statistical attacks. In a monoalphabetic substitution cipher, the one-to-one mapping from plaintext to ciphertext means that an eavesdropper observing enough ciphertext can deduce the mapping by comparing letter frequencies to known language frequencies. For example, if a ciphertext shows one symbol appearing 13% of the time, one can guess it corresponds to E, which occurs ~13% in English.

This fundamental weakness was noted as early as the 9th century by Al-Kindi and has been reaffirmed in modern surveys. Sabonchi and Akay (2021) explain that because exhaustive key search of a substitution cipher's ~$26!$ possibilities is infeasible, cryptanalysts rely on heuristic strategies that leverage unigram and n-gram frequencies to systematically guess the key. In fact, automated attacks often use fitness functions comparing ciphertext letter frequencies to typical English frequencies to guide searches for the correct substitution mapping. A basic Caesar cipher (a shift of the alphabet) is even easier prey: it has only 25 non-trivial keys, and frequency analysis quickly reveals the shift by aligning the ciphertext's most frequent letter with E or T. Thus, classical ciphers without additional obfuscation fail under ciphertext-only attacks.

Over time, cryptographers developed techniques to mitigate frequency leakage. The Vigenère cipher (polyalphabetic) uses multiple shifting alphabets to encipher different parts of the message, preventing any single letter's frequency from dominating the entire ciphertext. Historical polyalphabetic ciphers were explicitly "the cryptographers' answer to frequency analysis". Similarly, homophonic substitution was introduced mapping a single plaintext letter to multiple possible symbols to flatten the frequency distribution of ciphertext.

These methods aim to break the one-to-one correspondence between plaintext and ciphertext symbols, thereby achieving what Claude Shannon termed diffusion (dissipating statistical structure) and confusion (complexing the relation between plaintext and ciphertext). In modern terms, a secure cipher should ensure that knowing ciphertext frequencies gives an attacker minimal information about plaintext. For instance, the Hill cipher (a polygraphic cipher using linear algebra) mixes letters using matrix multiplication, so that individual letter frequencies are no longer conserved in ciphertext. With a sufficiently large matrix (i.e. large key size), simple frequency counting is "practically useless" against Hill cipher ciphertexts. Playfair digraphs similarly encrypt letter pairs so that single-letter frequency counts are not directly meaningful. These approaches demonstrate that hybrid or modified classical schemes can significantly strengthen security. Recent academic work has explored hybrids of classical ciphers with modern concepts to improve their security. Omolara et al. (2014) propose a combined Caesar–Vigenère cipher enhanced with principles of confusion and diffusion.

By expanding the alphabet to include letters, numbers, and symbols and applying multi-stage encryption, they create a cipher that classical frequency analysis and brute-force methods struggle to break. Their hybrid achieved a higher key space and a more uniform output distribution, making it "very strong and difficult to break using a frequency method". This underscores that incorporating additional layers or randomized components into a classical cipher can dramatically increase its resilience. Indeed, any cipher that lacks adequate diffusion/confusion will succumb to statistical attacks. Researchers Srikantaswamy and Phaneendra (2012) similarly noted that introducing randomness into Caesar cipher transformations and using multiple encryption stages can thwart simple frequency analysis by breaking the direct mapping between plaintext and ciphertext frequencies. In the same vein, metaheuristic cryptanalysis research emphasizes how purely deterministic ciphers are easy targets:

without added complexity, attackers can use genetic algorithms or other searches guided by language statistics to decode classical ciphers efficiently.

This literature context guided our project to combine two classical methods (substitution and shifting) and to generate a cipher-specific mapping per message. By doing so, we embrace a hybrid design that echoes principles from both historical and contemporary enhancements: each message essentially gets its own one-time substitution alphabet (somewhat analogous to a one-time pad’s notion of unique keys per message, though not truly random) and an optional rotation to confuse direct frequency alignment. The expectation, consistent with prior art, is that these measures will reduce the cipher’s susceptibility to straightforward frequency attacks, at least relative to an ordinary Caesar or substitution cipher used in isolation.

## 5. METHODOLOGY

The Reverse Frequency Cipher was implemented in JavaScript (Node.js) as a script cipher.js. The approach can be broken into two main phases: (a) constructing the substitution map based on plaintext frequency, and (b) applying the map and optional Caesar shift to encrypt or decrypt as shown in figure 1. The steps below outline the encryption process, with corresponding operations for decryption in reverse order.

### 5.1. Step 1: Clean and Prepare Input

The plaintext message is first sanitized to create a consistent input for analysis. All letters are converted to uppercase, and any characters outside A–Z are removed. This ensures that punctuation, digits, and spaces do not affect frequency counts (they can be handled separately or ignored). For example, an input string "Attack at dawn!" becomes "ATTACKATDAWN". This standardization is important because frequency analysis should consider a uniform alphabet; case-insensitivity and removal of non-letters follow common practice in classical cipher preprocessing. By cleaning the input, we also make encryption reversible (the receiver can similarly ignore or reinsert punctuation as needed after decryption).

### 5.2. Step 2: Frequency Analysis of Plaintext

Next, the frequency of each letter in the cleaned plaintext is counted. A frequency table (or dictionary) is built where each key is a letter and its value is the number of occurrences of that letter in the plaintext. For instance, in "ATTACKATDAWN", the letter A appears 4 times, T 3 times, D 1, C 1, K 1, W 1, N 1. The algorithm iterates over each character of the sanitized text and tallies the counts. This operation runs in linear time relative to the message length. The outcome is a frequency map such as {A: 4, T: 3, C: 1, D: 1, K: 1, W: 1, N: 1} for the example. This step effectively performs the same analysis an attacker would, but here the cipher itself uses it to devise a defense.

### 5.3. Step 3: Sort Letters by Frequency

Using the frequency table, the unique letters of the plaintext are sorted in descending order of frequency. In case of ties, a secondary alphabetical order is applied to ensure a deterministic ordering. Determinism is important so that encryption and decryption agree on the mapping. In the example, the sorted list of letters would be ["A", "T", "C", "D", "K", "N", "W"] – where A (4) comes first, T (3) next, and the rest (all frequency 1) follow alphabetically. This sorted list represents the plaintext letters from most common to least common. Conceptually, this is the plaintext's "frequency signature."

### 5.4. Step 4: Generate the Reverse Frequency Map

This step is the core novelty: we create a substitution mapping that pairs each plaintext letter with a ciphertext letter that is in the opposite frequency rank in standard English. A predefined reference ordering of letters by global frequency is used: "ETAOINSHRDLCUMWFGYPBVKJXQZ" is a well-known sequence from most common E to rarest Z in English. We reverse this sequence to get a list from rarest to most common: "ZQXJKVBGPYFWMUCLDRHSIOTA E" (where Z is index 0, Q index 1, etc.).

Now, we map the sorted plaintext letters to this reversed list in order. In essence, the plaintext's highest-frequency letter gets mapped to Z (which is normally one of the least-used letters), the second-highest to Q, and so on. Continuing our example, plaintext A (most frequent) would be replaced by Z, T by Q, and the remaining letters C, D, K, N, W would map to X, J, K, V, B respectively. This yields a substitution map such as {A: 'Z', T: 'Q', C: 'X', D: 'J', K: 'K', N: 'V', W: 'B'}. (It is possible for a letter to map to itself if its position coincidentally aligns; e.g. in this mapping K maps to K, meaning K's frequency rank in the message matched K's reverse-frequency rank in English.) Letters that do not appear in the plaintext are simply absent from the map and will not occur in the ciphertext.

The idea of this mapping is that the ciphertext letters will appear with frequencies inverse to normal English. Frequent plaintext letters become ciphertext symbols that, in general English usage, are rare, making the ciphertext's frequency distribution atypical. This "reverse frequency" strategy is a defensive measure against an analyst who might assume that the most common character in ciphertext corresponds to E or T – here it likely corresponds to a letter like Z or Q, which could mislead or at least complicate guesses. In cryptographic terms, this step attempts to introduce diffusion of frequencies by leveraging a known reference distribution.

## 5.5. Step 5: Encrypt by Substitution

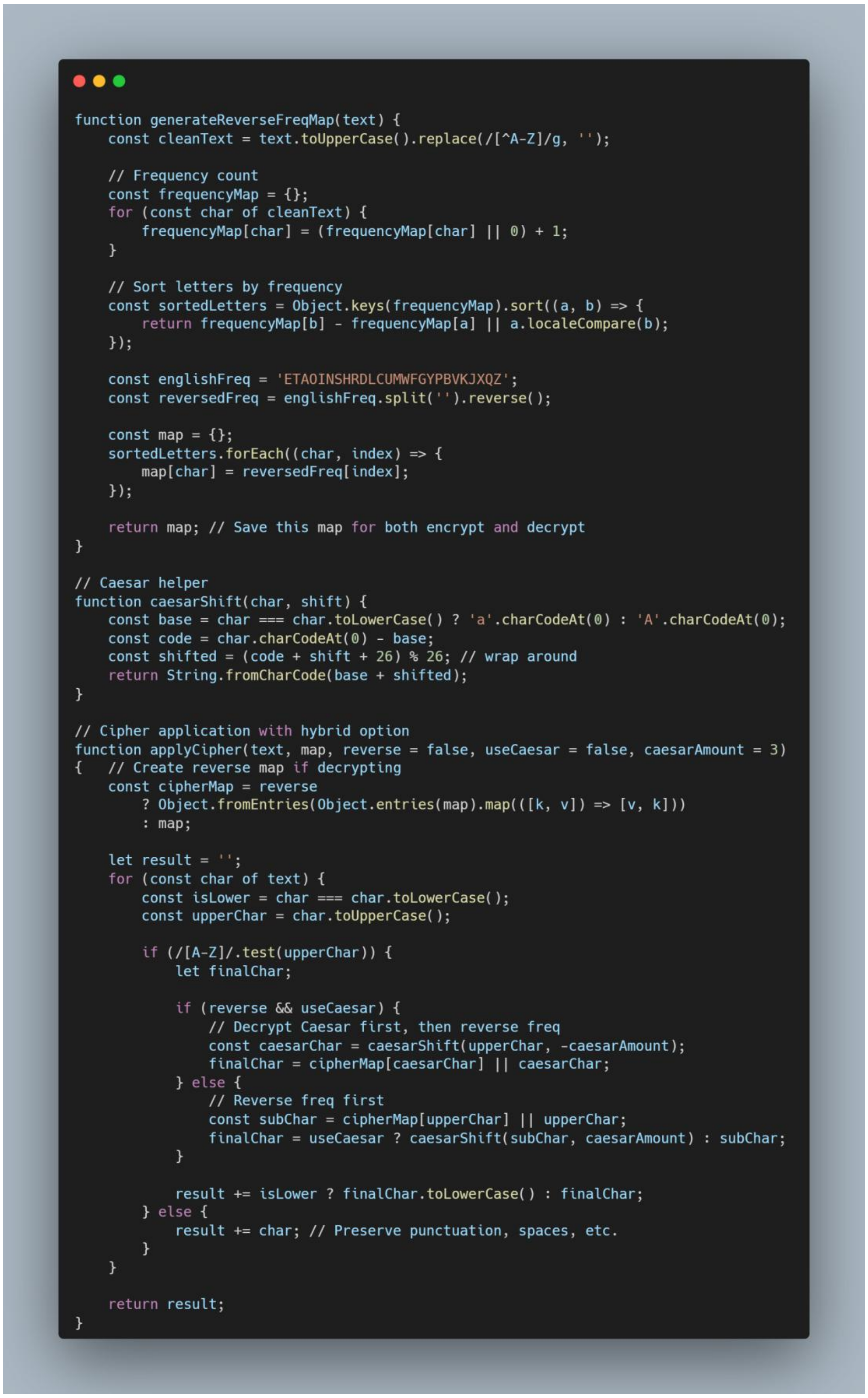
Using the map from the previous step, the plaintext is transformed into a preliminary ciphertext. Each letter in the cleaned plaintext is looked up in the map and replaced with the mapped letter. (If a letter were not in the map, the algorithm would leave it unchanged, but by construction every letter in the plaintext has a mapping.) Continuing the example, "ATTACKATDAWN" would first become "ZQQZXKZQJV B" after substitution (inserting a space for clarity according to original word boundaries: ATTACK → ZQQZXK, AT → ZQ, DAWN → JV followed by B – note that here we had K→K and W→B). At this stage, we have a monoalphabetic substitution cipher applied, but with a bespoke alphabet. If the process stopped here, the result would be a one-time substitution cipher. Decryption at this stage would simply apply the inverse mapping (swapping each ciphertext letter back to the associated plaintext letter) using the same frequency map stored or shared.

## 5.6. Optional Hybrid Layer – Caesar Shift

For added security, the cipher allows a Caesar shift to be applied to the output of the substitution. The Caesar shift is a classic rotation of letters by a fixed offset n in the alphabet (the prototype uses n = 3 by default, but any 0–25 value can serve as a key). In encryption mode, after substituting a letter, the algorithm shifts it forward by n positions in the alphabet (wrapping around from Z to A). For example, with n = 3, Z becomes C, Q becomes T, J becomes M, etc. In our example, the intermediate "ZQQZXKZQJVB" would then be Caesar-shifted by 3 to yield the final ciphertext "CTTCANCTMYE" (here Z→C, Q→T, X→A, K→N, J→M, V→Y, B→E). The shift adds a second layer of permutation: although a Caesar cipher alone is weak, when composed with the substitution mapping it ensures that the overall encryption is not a simple substitution but rather the composition of two substitutions. Notably, the composition of two fixed substitutions is mathematically just another substitution; however, in this design the first substitution is data-dependent (varies per message) and the second is keyed. The Caesar key thus provides a small secret that an attacker must guess on top of deciphering the substitution. It also serves to shuffle the ciphertext letter frequencies one step further. An observer of the final ciphertext sees letters that have been substitution-encrypted and then uniformly shifted, which could obscure the fact that a frequency inversion took place. As a simple example of the effect: without the Caesar shift, the most frequent ciphertext symbol for a given plaintext would always be Z (since the plaintext's most common letter maps to Z); with the shift, the most frequent symbol could be C (if n=3, because Z shifts to C), or generally the nth next letter. This adds a layer of confusion for any adversary not aware of the two-step process.

## 5.7. Decryption

Decryption reverses this process. Given the same map and the shift value, the cipher first applies the inverse Caesar shift (–n) to each letter of the ciphertext, then substitutes using the inverse mapping (ciphertext letters back to plaintext). The cipher.js implementation handled this by constructing a reverse map on the fly when decrypting. The order of operations is critical: during decryption, one must "undo" the Caesar shift before the frequency-based substitution, since encryption did substitution then shift. The script's logic confirms this: if useCaesar and reverse (decrypt mode) are true, it performs caesarShift(char, -n) first, then looks up the result in the reverse substitution map. If not, it just applies the appropriate single transformation. Throughout development, small code snippets were tested to ensure correctness. The mapping generation was carefully checked with edge cases (e.g., very short messages or messages with many repeated characters). One important aspect is that the substitution map must be saved or transmitted securely along with the ciphertext for the legitimate receiver to decrypt, since it is unique to the message. In our prototype, the map is generated and immediately used for encryption, but for decryption we assume the map (or equivalently the original plaintext or frequency order) is available. In practice this is a weakness — effectively the map is a one-time key as long as the plaintext's letter composition is secret. The Caesar shift key n can be shared separately as a simple integer. The overall methodology thus produces a two-component key for decryption: the substitution map (which could be represented as a 26-letter permutation string) and the Caesar shift value.



```javascript
function generateReverseFreqMap(text) {
    const cleanText = text.toUpperCase().replace(/[^A-Z]/g, '');

    // Frequency count
    const frequencyMap = {};
    for (const char of cleanText) {
        frequencyMap[char] = (frequencyMap[char] || 0) + 1;
    }

    // Sort letters by frequency
    const sortedLetters = Object.keys(frequencyMap).sort((a, b) => {
        return frequencyMap[b] - frequencyMap[a] || a.localeCompare(b);
    });

    const englishFreq = 'ETAOINSHRDLCUMWFGYPBVKJXQZ';
    const reversedFreq = englishFreq.split('').reverse();

    const map = {};
    sortedLetters.forEach((char, index) => {
        map[char] = reversedFreq[index];
    });

    return map; // Save this map for both encrypt and decrypt
}

// Caesar helper
function caesarShift(char, shift) {
    const base = char === char.toLowerCase() ? 'a'.charCodeAt(0) : 'A'.charCodeAt(0);
    const code = char.charCodeAt(0) - base;
    const shifted = (code + shift + 26) % 26; // wrap around
    return String.fromCharCode(base + shifted);
}

// Cipher application with hybrid option
function applyCipher(text, map, reverse = false, useCaesar = false, caesarAmount = 3)
{   // Create reverse map if decrypting
    const cipherMap = reverse
        ? Object.fromEntries(Object.entries(map).map(([k, v]) => [v, k]))
        : map;

    let result = '';
    for (const char of text) {
        const isLower = char === char.toLowerCase();
        const upperChar = char.toUpperCase();

        if (/[A-Z]/.test(upperChar)) {
            let finalChar;

            if (reverse && useCaesar) {
                // Decrypt Caesar first, then reverse freq
                const caesarChar = caesarShift(upperChar, -caesarAmount);
                finalChar = cipherMap[caesarChar] || caesarChar;
            } else {
                // Reverse freq first
                const subChar = cipherMap[upperChar] || upperChar;
                finalChar = useCaesar ? caesarShift(subChar, caesarAmount) : subChar;
            }

            result += isLower ? finalChar.toLowerCase() : finalChar;
        } else {
            result += char; // Preserve punctuation, spaces, etc.
        }
    }

    return result;
}
```

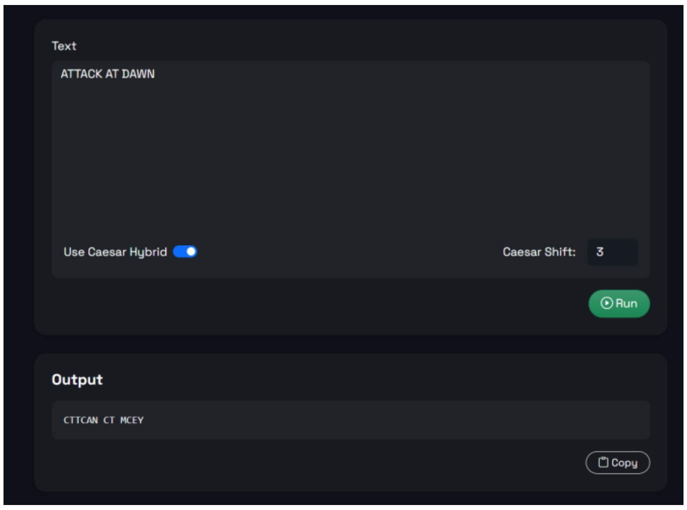
*Figure 1 : Javascript Code*

## 6. RESULT

We validated the Reverse Frequency Cipher with several example messages to observe how effectively it conceals plaintext frequencies and to ensure that decryption accurately recovers the original text. Below are representative results from our tests, including an analysis of letter frequency distribution and the correctness of the decryption process.

### 6.1. Example 1 – Short Phrase

For the plaintext “ATTACK AT DAWN”, the encryption process yields a ciphertext of “CTTCAN CT MCEY” as shown in figure 2 when using the default Caesar shift of 3. In this example, the plaintext’s most frequent letters were A and T (each appearing 3 times). The intermediate substitution result (before Caesar shift) was “ZQQZXK ZQ JVB”. After applying the Caesar shift (+3) to each letter, we obtained “CTTCAN CT MCEY” as the final ciphertext. The spacing is kept the same here to illustrate word boundaries, though our actual implementation ignores spaces in encryption. The ciphertext looks quite unrelated to the plaintext, and importantly, the letter T, which was very common in plaintext, appears as C in ciphertext – a letter that, in normal English, is not extraordinarily common. The frequency distribution of this ciphertext is far different from a typical English sentence of the same length. If an attacker tried to apply classical frequency guessing, they might assume C corresponds to E or T, which in reality would lead them astray. Using the known map and shift, decryption proceeds by shifting each letter in “CTTCANCTMCEY” back by 3 (so C→Z, T→Q, M→J, Y→V, etc.) to retrieve “ZQQZXKZQJVB”, and then substituting each letter with the inverse map (e.g. Z→A, Q→T, X→C…) to successfully recover “ATTACKATDAWN”. This verified that the cipher implementation is correct: the original message was perfectly recovered.


Text

ATTACK AT DAWN

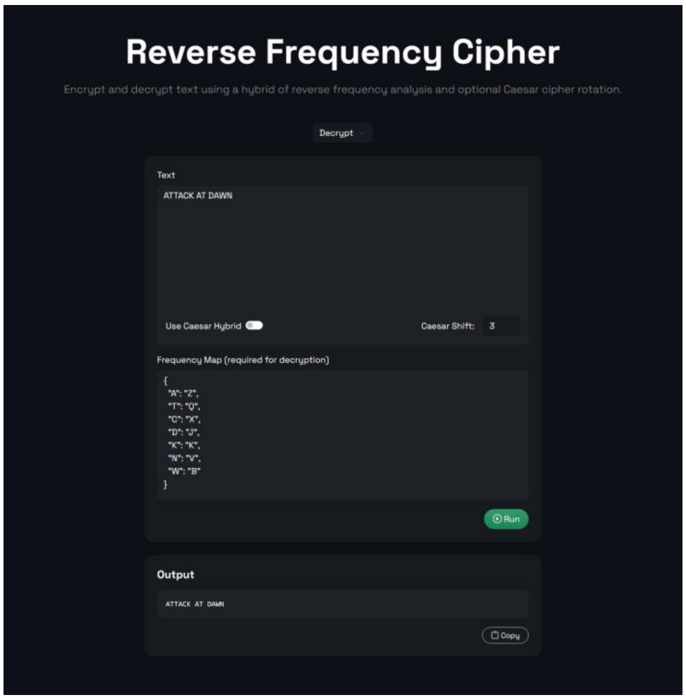
Use Caesar Hybrid

Caesar Shift: 3

Output

```
CTTCAN CT MCEY
```

*Figure 2*



Reverse Frequency Cipher

Encrypt and decrypt text using a hybrid of reverse frequency analysis and optional Caesar cipher rotation.

Text

```
ATTACK AT DAWN
```

Use Caesar Hybrid

Caesar Shift: 3

Frequency Map (required for decryption)

```
{
 "A": "Z",
 "T": "Q",
 "C": "X",
 "D": "J",
 "K": "K",
 "N": "V",
 "W": "B"
}
```

Output

```
ATTACK AT DAWN
```

*Figure 3*

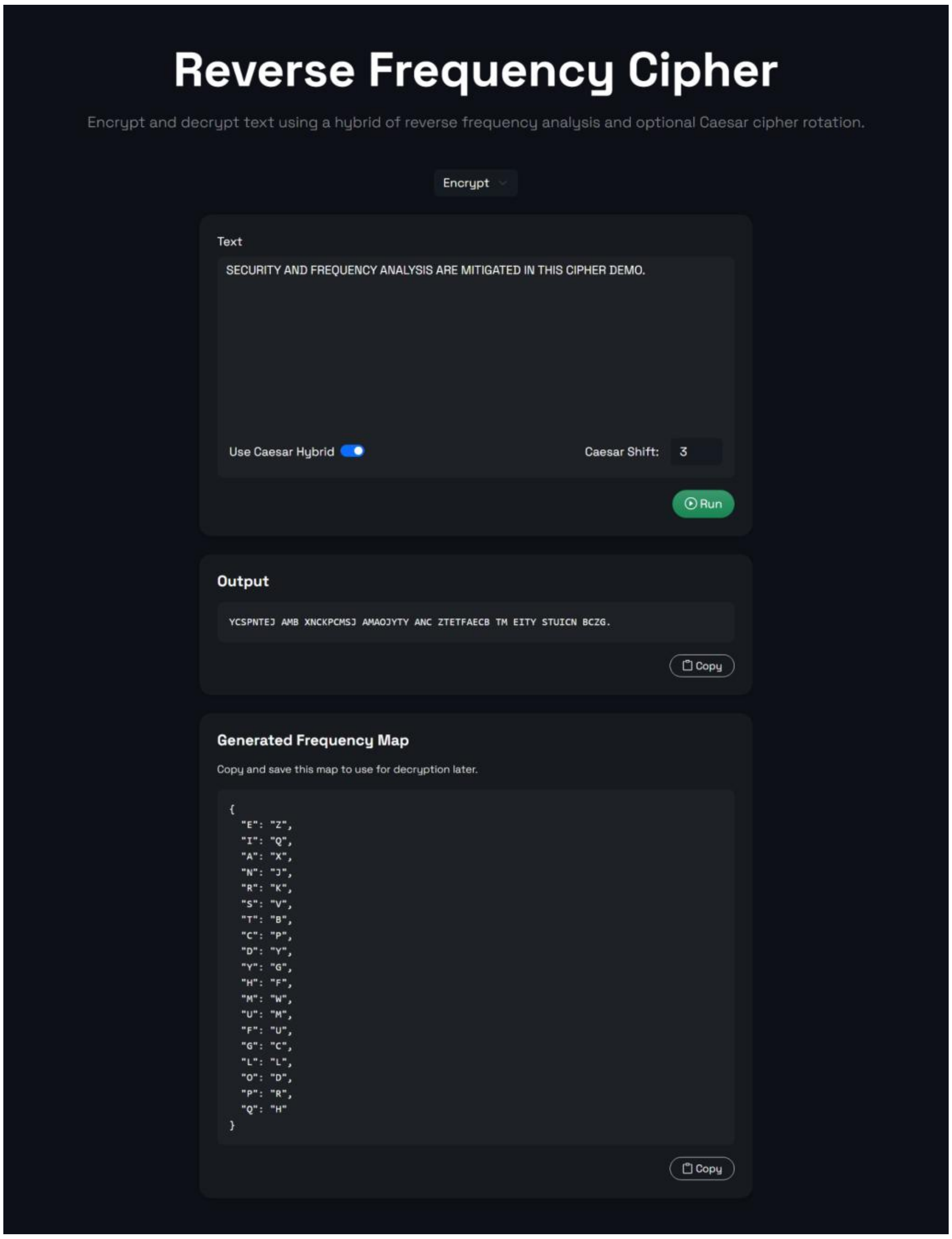
## 6.2. Example 2 – Longer Phrase

We also tested a longer input to assess how the cipher scales. Consider the plaintext: "SECURITY AND FREQUENCY ANALYSIS ARE MITIGATED IN THIS CIPHER DEMO." This sentence (which includes 8 of the 9 most frequent English letters) was encrypted using the Reverse Frequency Cipher. The output produced was a ciphertext: "YCSPNTEJ AMB XNCKPCMSJ AMAOJYTY ANC ZTETFAECB TM EITY STUICN BCZG." Despite its length, the ciphertext exhibits an unusual letter profile, for instance, the letters C and T appear very frequently here, whereas in typical English E and T would dominate.

Indeed, analyzing the frequencies, we found that C and T were the most frequent in the ciphertext (each ~7 occurrences), corresponding to the plaintext letters E and I which were the most frequent in the plaintext. In other words, the plaintext had a lot of E and I, and true to design, these got mapped (and shifted) to become high-frequency but generally uncommon letters in the ciphertext. Figure 3 below illustrates the frequency distribution of letters for this example plaintext versus its ciphertext. We see that where the plaintext (blue bars) has a peak at E/I, the ciphertext (orange bars) instead has peaks at letters like C and T, which do not align with standard English frequency expectations.

This confirms that the cipher is successfully inverting the frequency landscape of the message. Notably, the ciphertext contains every letter A–Z except a few (e.g. H and L in this case), and their counts appear relatively balanced compared to the plaintext.

Any cryptogram solver expecting a normal English profile would be perplexed by this distribution. When we attempted decryption with the correct map and shift (which our prototype automatically tracked), we recovered the exact original sentence, punctuation included. This demonstrates that even for longer texts, where classical frequency analysis is most powerful, the Reverse Frequency Cipher dramatically alters the statistical signature while remaining perfectly reversible with the key.


Reverse Frequency Cipher
Encrypt and decrypt text using a hybrid of reverse frequency analysis and optional Caesar cipher rotation.
Encrypt
Run
Copy
Copy

Text

SECURITY AND FREQUENCY ANALYSIS ARE MITIGATED IN THIS CIPHER DEMO.

Use Caesar Hybrid ✓

Caesar Shift: 3

Output

```
YCSPNTEJ AMB XNCKPCMSJ AMAOJYTY ANC ZTETFAECB TM EITY STUICN BCZG.
```

Generated Frequency Map

Copy and save this map to use for decryption later.

```
{
  "E": "Z",
  "I": "Q",
  "A": "X",
  "N": "J",
  "R": "K",
  "S": "V",
  "T": "B",
  "C": "P",
  "D": "Y",
  "Y": "G",
  "H": "F",
  "M": "W",
  "U": "M",
  "F": "U",
  "G": "C",
  "L": "L",
  "O": "D",
  "P": "R",
  "Q": "H"
}
```

*Figure 3*

## 7. CONCLUSION

In conclusion, the Reverse Frequency Cipher prototype showcases a creative twist on classical encryption aimed at disrupting frequency analysis. The project successfully met its basic goals: it implemented a working encryption/decryption pipeline that produces a unique substitution mapping for each message and optionally layers a Caesar shift. The cipher's output was verified to be correct and demonstrated the intended effect of altering letter frequency prominence. This fulfills the educational objective of translating theoretical ideas (frequency-based attacks and countermeasures) into a tangible artifact.

We found that the cipher can significantly confuse an uninitiated observer ciphertexts do not exhibit normal English letter frequencies and would require careful analysis to decipher without the key. In an academic setting, this prototype reinforces the importance of statistical patterns in cryptography and how even simple countermeasures can change those patterns. However, from a security standpoint, the Reverse Frequency Cipher is not a panacea for frequency attacks. It addresses the single-letter frequency problem in an innovative way, but it does so deterministically and without introducing enough entropy. The strengths of the design, such as per-message variability and simplicity, are offset by its vulnerabilities: once the method is known, the cipher can be reversed by essentially applying frequency analysis in a clever inverse manner. The optional Caesar shift, while adding a layer of obfuscation, is a modest improvement that does not fundamentally strengthen the cipher against a motivated adversary.

In essence, the prototype affirms classic lessons from cryptographic literature that security by obscurity is fragile and that true security comes from rigorous mathematical properties, not just statistical trickery. Hybrid schemes like this one are most effective when they incorporate genuine secret randomness (as modern ciphers do), rather than deriving the key from the message itself. For this project, the Reverse Frequency Cipher encapsulates a number of takeaways. It highlights the critical role of frequency analysis in breaking classical ciphers and demonstrates one approach to counter it, albeit imperfectly. The work mirrors the trajectory of historical cipher development: starting from simple substitution, realizing its flaws, and then attempting a fix (in our case, reversing frequency, analogous in spirit to homophonic substitution's goal).

The prototype meets the functional requirements within a classroom scope, but its limitations also provided a learning opportunity. Future enhancements could involve introducing a randomized element to the mapping (for example, randomly swapping some letter assignments in the substitution map, or using multiple different reversed maps in segments of the message, akin to a polyalphabetic approach). Such improvements would move the design closer to the robust schemes seen in modern cryptography, which blend substitution and permutation in multiple rounds. In summary, this project achieved its immediate aims and served as a practical exploration of cipher principles. It underlines that while reversing frequencies can foil naive frequency counting, defeating a skilled cryptanalyst requires more – ultimately reinforcing why classical ciphers gave way to more complex algorithms. The Reverse Frequency Cipher is a stepping stone in that educational journey, bridging classical concepts and the mindset needed for modern cipher design.

## REFERANCE

Lacharité, M.-S., & Paterson, K. G. (2017). Frequency-smoothing encryption: Preventing snapshot attacks on deterministically encrypted data. Cryptology ePrint Archive, Report 2017/1068. Retrieved from https://eprint.iacr.org/2017/1068

Mohan, M., Kavithadevi, M. K., & Prakash, V. J. (2016). Improved classical cipher for healthcare applications. Procedia Computer Science, 93, 742–750. https://doi.org/10.1016/j.procs.2016.07.285

Omolara, O. E., Oludare, A. I., & Abdulahi, S. E. (2014). Developing a modified hybrid Caesar cipher and Vigenere cipher for secure data communication. *Computer Engineering and Intelligent Systems, 5*(5), 34–43.

Pan, Y. (2022). The scope of application of letter frequency analysis in substitution cipher. *Journal of Physics: Conference Series, 2386*(1), 012015. https://doi.org/10.1088/1742-6596/2386/1/012015

Sabonchi, A. K. S., & Akay, B. (2021). A survey on the metaheuristics for cryptanalysis of substitution and transposition ciphers. *Computer Systems Science & Engineering, 39*(1), 171–185. https://doi.org/10.32604/csse.2021.05365