GRONSFELD CIPHER

A technical report

Prepared By:
Sapthami Upadhya,
MIST WC

Gronsfeld Cipher: An Overview

1. Introduction

The Gronsfeld cipher is a polyalphabetic substitution cipher similar to the Vigenère cipher. The key distinction between the Gronsfeld and Vigenère ciphers lies in the nature of the key used for encryption and decryption.

While the Vigenère cipher employs an alphabetic key, the Gronsfeld cipher simplifies this by utilizing a numeric key, making it more straightforward to handle and implement.

This numeric approach makes the Gronsfeld cipher easier to understand and operate manually, which was particularly advantageous in historical contexts before the advent of modern computational tools.

2. Historical Context

Named after Johann Franz Gronsfeld, the cipher emerged during the early modern period, around the 17th century, and was used for secure communication. It aimed to provide a more accessible method for encryption while maintaining a reasonable level of security against the cryptanalytic techniques of its time. Although it has been surpassed by more advanced cryptographic methods, the Gronsfeld cipher remains an important educational tool for understanding the basics of polyalphabetic substitution and the evolution of encryption techniques.

3. Mechanism of the Gronsfeld Cipher

The Gronsfeld cipher involves shifting letters of the plaintext according to a repeating sequence of digits from the numeric key. Here is a detailed examination of the encryption and decryption processes.

3.1 Encryption Process

1. Choose a numeric key K, consisting of digits (k1, k2, ..., kn). For example, let K = 314159.
2. The plaintext P is prepared for encryption. For example, P = HELLO.
3. Repeat the key to match the length of the plaintext. If the plaintext length is m, the key sequence becomes K' = k1, k2,..., kn, k1, k2,... until its length is m. For "HELLO", K' = 31415.
4. For each character pi in the plaintext, calculate the corresponding ciphertext character ci using the formula:

   ci = (pi + ki') mod 26

   where pi and ci are the positions of the plaintext and ciphertext characters in the alphabet (0-indexed).

Example:
- H (7) shifted by 3 -> K (10)
- E (4) shifted by 1 -> F (5)
- L (11) shifted by 4 -> P (15)
- L (11) shifted by 1 -> M (12)
- O (14) shifted by 5 -> T (19)

Thus, the ciphertext for HELLO with key 314159 is KFPMT.

3.2 Decryption Process

1. Use the same numeric key K for decryption.
2. The ciphertext C is prepared for decryption. For example, C = KFPMT.
3. Repeat the key K to match the length of the ciphertext, resulting in the key sequence K'
4. For each character ci in the ciphertext, calculate the corresponding plaintext character pi using the formula:
   pi = (ci - ki' + 26) mod 26
   where pi and ci are the positions of the plaintext and ciphertext characters in the alphabet (0-indexed).

Example:
- K (10) shifted back by 3 -> H (7)
- F (5) shifted back by 1 -> E (4)
- P (15) shifted back by 4 -> L (11)
- M (12) shifted back by 1 -> L (11)
- T (19) shifted back by 5 -> O (14)

Thus, the decrypted plaintext for KFPMT with key 314159 is HELLO.

4. Mathematical Foundations

The Gronsfeld cipher operates within the framework of modular arithmetic, where each character in the plaintext is treated as a position within the alphabet. The alphabet is typically represented by the numbers 0 through 25, corresponding to the letters A through Z. This ensures that any shifts wrapping around the end of the alphabet correctly cycle back to the beginning.

4.1 Modular Arithmetic

Modular arithmetic is a system of arithmetic for integers, where numbers wrap around after reaching a certain value—the modulus. In the context of the Gronsfeld cipher, the modulus is 26, corresponding to the 26 letters of the English alphabet. Operations are performed as follows:
(a+b) mod 26
(a−b+26) mod 26
These operations ensure that results stay within the range of 0 to 25, thus mapping to valid alphabetic characters.

5. Cryptanalysis

The security of the Gronsfeld cipher relies on the unpredictability and length of the key. However, it is susceptible to various attacks if the key is short or reused frequently.

5.1 Frequency Analysis

If the key is short relative to the message length, the ciphertext can exhibit repeating patterns. This vulnerability can be exploited using frequency analysis, similar to attacks on the Vigenère cipher. By

analyzing the frequency of letters in the ciphertext and comparing them to expected frequencies in the language, an attacker can deduce the key length and subsequently the key itself. Or if a portion of the plaintext is known, it can be used to deduce the key.

5.2 Kasiski Examination

The Kasiski examination is a method to determine the key length by identifying repeating sequences in the ciphertext. By measuring the distances between these sequences, the greatest common divisor (GCD) of these distances often reveals the key length. Originally developed for the Vigenère cipher, it can be adapted to attack the Gronsfeld cipher as:

1. Identify repeated sequences in the ciphertext.
2. Calculate the distances between these repetitions.
3. Find the greatest common divisor of these distances to estimate the key length.

5.3 Index of Coincidence

The index of coincidence (IC) is a statistical measure that helps in determining the key length by comparing the IC of the ciphertext with the expected IC for random text and the plaintext language. A significant deviation indicates a polyalphabetic cipher and can aid in key length determination.

Example Implementation in Python

Here is a detailed Python implementation for both encryption and decryption processes of the Gronsfeld cipher:

```
def gronsfeld_encrypt(plaintext, key):

    alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

    ciphertext = ''

    p_length=len(plaintext)

    k_length=len(key)

    key_repeat = (key * ((p_length//k_length) + 1))

    key_final=key_repeat[:len(plaintext)]


    for i in range(p_length):

        p=plaintext[i].upper()

        k=key_final[i]

        if p in alphabet:

            shifted_index = (alphabet.index(p) + int(k)) % 26

            ciphertext += alphabet[shifted_index]
```

```
        else:
            ciphertext += p

    return ciphertext

def gronsfeld_decrypt(ciphertext, key):
    alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    plaintext = ''
    c_length=len(ciphertext)
    k_length=len(key)
    key_repeat=(key*((c_length//k_length)+1))
    key_final = key_repeat[:len(ciphertext)]  #set length of key same as text

    for i in range(c_length):
        c=ciphertext[i].upper()
        k=key_final[i]
        if c in alphabet:
            shifted_index = (alphabet.index(c) - int(k)) % 26
            plaintext += alphabet[shifted_index]
        else:
            plaintext += c

    return plaintext


plaintext= input("Enter the plain text: ")
key=input("Enter the key: ")
```

```
print(gronsfeld_encrypt(plaintext,key))

ciphertext= input("Enter the cipher text: ")

key=input("Enter the key: ")

print(gronsfeld_decrypt(ciphertext,key))
```

Conclusion

The Gronsfeld cipher is a fascinating example of historical encryption techniques. While its security is limited by modern standards, understanding its mechanics provides valuable insights into the evolution of cryptography. The use of numeric keys simplifies the process but introduces vulnerabilities if not used with sufficient complexity and care.

References

"The Gronsfeld cipher," [Online]. Available: http://www.cs.trincoll.edu/cryptography/gronsfeld.html#:~:text=The%20Gronsfeld%20cipher%20is%20a,be%20used%20in%20the%20key

"Gronsfeld Cipher," Rumkin.com, [Online]. Available: https://rumkin.com/tools/cipher/gronsfeld/

"Gronsfeld," CryptoCrack, [Online]. Available: https://sites.google.com/site/cryptocrackprogram/user-guide/cipher-types/substitution/gronsfeld

"Gronsfeld Cipher Manual," GCWizard, [Online]. Available: https://blog.gcwizard.net/manual/en/gronsfeld-cipher/01-what-is-the-gronsfeld-cipher/