

Advanced Vigenère Encryption System Using Taylor Swift Songs with Integrated Hash Security Mechanism

Yasmin Farisah Salma - 13522140
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
¹13522140@std.stei.itb.ac.id

Abstract— The Advanced Vigenère Encryption System using Taylor Swift Swongs with Integrated Hash Security is an innovative cryptographic solution that revitalizes the traditional Vigenère cipher with a unique twist. At its core, this cipher utilizes the Taylor Swift's songs as a dynamic source for key generation, offering an unprecedented blend of cultural richness and encryption complexity. This approach transforms the familiar landscape of cryptographic key generation by tapping into the vast, ever-evolving library of pop music. The system is designed to generate keys that are not only diverse and unpredictable but also resonate with contemporary cultural elements, adding an extra layer of security through obscurity. The integration of a specialized hash security mechanism further enhances the system's robustness. This mechanism ensures the integrity and authenticity of encrypted messages, providing an additional defense against tampering and unauthorized decryption attempts. Overall, this cipher stands out as a novel encryption method, bridging the gap between traditional cryptography and modern cultural influences, and offers a fresh perspective on secure digital communication.

Keywords—Cryptography, Vigenère Cipher, Hash Security, Taylor Swift.

I. INTRODUCTION

Information security not only emerged as a paramount component of technological advancement but also as a bulwark against the rising tide of cyber threats in this swiftly evolving digital era. The relentless escalation of sophisticated cyber-attacks necessitates a paradigm shift in encryption methodologies—a shift towards systems that are robust, resilient, and adaptive to the landscape of digital compromise. In this context, there is an urgent call for the reexamination and revitalization of historical cryptographic methods, such as the Vigenère cipher. This classic cipher, with its polyalphabetic approach, was once the vanguard of secrecy, a testament to the ingenuity of Renaissance cryptographers. Despite being superseded due to its vulnerabilities to modern analytical attacks, the Vigenère cipher's methodology continues to inspire and inform the development of contemporary encryption strategies.

The Vigenère cipher's concept of key-dependent shifting provides a foundational understanding of polyalphabetic encryption, which is pivotal to developing advanced encryption

techniques. However, to meet the contemporary standards of security, there is an imperative to enhance and adapt these classical methods. This is where the integration of discrete mathematics and unconventional key sources comes into play.

Delving into the cultural domain, the utilization of Taylor Swift's song lyrics as a substrate for key generation transcends novelty and ventures into the realm of strategic ingenuity. By tapping into the unpredictability and cultural richness of musical content, this method introduces an unprecedented layer of complexity to the encryption algorithm. The selection of Taylor Swift's songs from a globally recognized artist adds a dimension of cultural dynamism, making the encryption keys as fluid and evolving as the music itself. This approach mirrors the broader trend of harnessing non-traditional, culturally embedded elements to reinforce cryptographic security, thereby broadening the horizons of what can be deemed as sources of cryptographic keys.

This system does not merely rely on the intricacies of Taylor Swift's songs, but also integrates a robust hash function to fortify the encryption and decryption process. The integration of this hash security mechanism significantly amplifies the system's resilience against the most tenacious brute-force attacks and sophisticated cryptographic analyses. It is a testament to the system's ingenuity that it not only encrypts but also validates, ensuring the sanctity of the data is maintained throughout its digital journey. Furthermore, the inclusion of a hash security mechanism addresses the need for message integrity and authentication—essential aspects of secure communication. The hash function's role is to ensure that any alteration to the encrypted message is detectable, thus safeguarding against tampering and providing a checksum for data integrity.

II. THEORY

A. Cryptography

Cryptography, derived from the Greek word for "secret writing," is both a science and an art dedicated to securing messages by transforming them into an encoded form that appears meaningless. The primary goal of cryptography is to ensure that confidential messages remain inaccessible to unauthorized individuals. In this context, the message, referred

to as plaintext, is any readable and understandable data or information. Once encrypted, this message becomes ciphertext, which is the encoded version that loses its original meaning and interpretation.

Example:

Plaintext : culik anak itu jam 11 siang
Ciphertext : t^\$gfUi9rewoFpfdWqL:[uTcxZy

Cryptosystem[3] is an implementation of cryptographic techniques and their accompanying infrastructure to provide information security services. A cryptosystem is also referred to as a cipher system.

Encryption is the process of converting plain, readable data (known as plaintext) into a coded form (called ciphertext) to prevent unauthorized access. This transformation is achieved using an algorithm and a key. The algorithm dictates the method of transformation, while the key specifies the exact changes to be made to the plaintext. The strength of encryption lies in the complexity of the algorithm and the secrecy of the key. Encryption is vital in protecting sensitive information from being accessed by unintended recipients, ensuring that the data remains confidential during transmission or while stored.

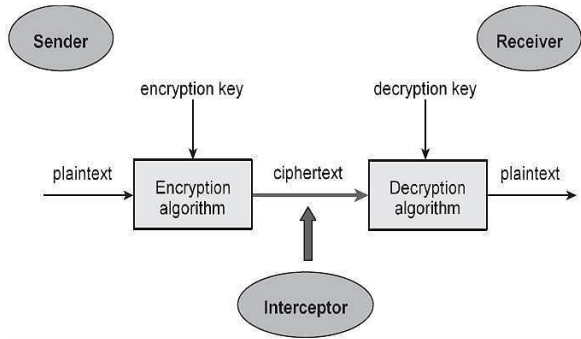


Fig 2.8 Cryptosystem
 Source: [3]

Decryption is the reverse process of encryption. It involves converting the ciphertext back into plaintext, making the information accessible and understandable again. Decryption requires the use of the same or a corresponding key used during the encryption process, depending on whether symmetric or asymmetric encryption is employed. In symmetric encryption, the same key is used for both encrypting and decrypting the data. In contrast, asymmetric encryption uses a pair of related keys – a public key for encryption and a private key for decryption.

B. Vigenère Cipher

The Vigenère Cipher stands as a classic example of polyalphabetic substitution in the field of cryptography, embodying a significant leap in cryptographic techniques since its inception in the 16th century. Named after Blaise de Vigenère, this cipher enhances the basic Caesar cipher by employing a series of different Caesar ciphers based on the letters of a keyword. The true elegance of the Vigenère Cipher lies in its simplicity and effectiveness: it uses a straightforward

algorithm to create a significantly more complex encryption than its monoalphabetic counterparts.

In the Vigenère cipher, the shift applied to each letter in the plaintext varies, altering the position of that letter in the alphabet for each instance. Consequently, the frequency of letters in the ciphertext does not directly mirror their frequency in the plaintext. Furthermore, repeated letters in the ciphertext do not necessarily indicate the repetition of the same letters at those specific points in the plaintext. The Vigenère cipher encryption system/scheme is symmetric and therefore, both the sender and receiver of the message should know how the message has been encrypted and ensure that the information is kept secret from any unwanted third person.

Vigenère Cipher Encryption Example

(a)

(b)

(c)

(d)

Fig 2.9 Vigenère cipher encryption process
 Source: [4]

C. Taylor Swift

Taylor Alison Swift (born December 13, 1989) is an American singer-songwriter. She has been recognized for her songwriting, musical versatility, artistic reinventions, and influence on popular culture and the music industry.

Taylor Swift, a globally celebrated artist, has achieved remarkable success in the music industry, distinguishing herself through a diverse and expansive discography. Known for her evocative storytelling and chart-topping hits, Swift has garnered a legion of fans and numerous accolades, including several Grammy Awards.



Fig 2.10 Taylor Swift on her world tour

Source: <https://pin.it/5nH39ps>

Her song titles, in particular, offer a unique blend of creativity, diversity, and cultural resonance, making them an intriguing choice for cryptographic key generation. The utilization of Taylor Swift's song titles as a basis for key generation in our encryption system is inspired by the sheer variety and distinctiveness of her titles. Each title, being a unique combination of words, presents a vast array of potential key material, rich in variability and less predictable than standard random sequences. This innovative approach to key generation leverages the ubiquitous and dynamic nature of popular culture, infusing a novel element into the cryptographic domain. The popularity and recognition of Swift's titles add an extra dimension of security through obscurity, ensuring that the keys derived from her titles are not only diverse but also resonate with a contemporary cultural context. By integrating these elements of popular music into cryptography, we are redefining traditional key generation methods, offering enhanced security through the cultural and linguistic complexity inherent in Taylor Swift's song titles.

D. Hashing

Hashing is a technique which accepts a variable length message as Input and produces a fixed length and unique string. In cryptography whatever the new techniques are adopted to secure the data, by one or other way that techniques are venerable to various types of attacks. The security mechanisms in cryptography are encipherment that is encryption and decipherment that is decryption. The other Mechanism of security in cryptography is hashing mechanism which uses a hash function and generates the hash values. The security of cryptographic algorithms is dependent on various key ingredients like Plaintext, cipher text, encryption and decryption process, the length of the key. If the length of the key is more than that algorithm is considered to be more secure. There are two ways of using the key in cryptography. One is using the

same key for encryption and decryption process which is called as symmetric key cryptography. Other is using different key one is for encryption and other is for decryption which is called as asymmetric key cryptography[6].

In cryptographic applications, hashing is crucial for several reasons. First, it provides a method to verify the integrity of data. Since any modification, even slight, to the original data results in a significantly different hash, it becomes an efficient tool to detect alterations or tampering. This property is particularly important in scenarios like secure data transmission and digital signatures. Second, hashing is instrumental in storing sensitive information, such as passwords, where storing the actual data can pose a security risk. By storing hash values instead, the security of the data is preserved even if the hash values are compromised, as the original data can't be easily reconstructed.

Moreover, the effectiveness of a hash function is measured by its ability to minimize collisions (different inputs producing the same hash) and its resistance to cryptographic attacks. A good hash function produces hash values that appear random and distributes them uniformly across the possible hash space. This randomness and unpredictability are what make hashing an indispensable tool in cryptography.

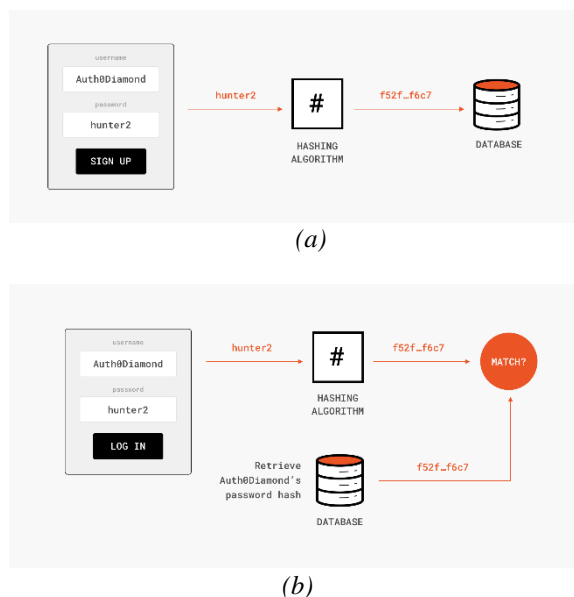


Fig 2.11 Hashing illustration

Source: [5]

III. METHODOLOGY

A. Data Collection

For the Advanced Vigenère Encryption System using Taylor Swift Songs with Integrated Hash Security, the data collection process primarily involved compiling a comprehensive list of Taylor Swift's song titles. This critical step was crucial as these titles served as the foundation for generating encryption keys, infusing both cultural relevance and complexity into the cryptographic process. The primary source for collecting these song titles was Spotify, a widely recognized and accessible digital music streaming service. Spotify's extensive database provided a reliable and up-to-date repository of Taylor Swift's

discography, encompassing all her released songs across various albums and singles. This platform was chosen for its comprehensive coverage, ease of access, and its ability to provide the latest updates on the artist's releases, which is vital for maintaining the dynamism and relevance of the encryption system.

B. Encryption Algorithm

The core of this encryption system is a modified Vigenère cipher, which traditionally uses a keyword to perform polyalphabetic substitution. In this innovative approach, the 'keyword' is replaced with a key generated from Taylor Swift's song titles. The algorithm involves several steps that include key generation, message encryption, and the integration of a hash function for added security.

Key Generation

```
# TODO: Randomly select a song title
from database.txt as the key
```

```
def select_random_song(songs):
    return random.choice(songs)
```

1) Song Title Selection

From the collected database of Taylor Swift's songs, a specific title is selected. This selection can be random or based on certain criteria (e.g., most recent song, a song from a specific album).

2) Key Formation

The selected song title is then processed to form the encryption key. This involves removing spaces and special characters, and converting all letters to a standard case (usually uppercase).

3) Key Expansion

If the key is shorter than the plaintext, it is repeated until it matches the length of the plaintext. This ensures that every character of the plaintext has a corresponding character in the key.

Encryption Process

```
# TODO: Vigenère Cipher Encryption
```

```
def vigenere_encrypt(plain_text, key):
    encrypted_text = ''
    for i in range(len(plain_text)):
        char = plain_text[i]
        if char.isalpha():
            shift = ord(key[i % len(key)].upper()) - 65
            encrypted_char = chr((ord(char.upper()) + shift - 65) % 26 + 65)
            encrypted_text += encrypted_char
        else:
```

```
        encrypted_text += char
    return encrypted_text
```

1) Preprocessing of Plaintext

The plaintext message is prepared for encryption. This involves removing non-alphabetic characters and converting all letters to the same case as the key.

2) Character-by-Character Encryption

For each character in the plaintext, a shifted character is generated based on the corresponding character in the key. The shift is determined by the position of the key character in the alphabet. For example, if the key character is 'B' (2nd position in the alphabet), the plaintext character is shifted by one position.

3) Polyalphabetic Substitution

Each character in the plaintext is substituted with the shifted character generated in the previous step. This creates the ciphertext, which appears as a random string of letters.

Integration of Hash Function

```
# TODO: Encrypt text and merge with hash
code
```

```
def encrypt(plain_text, songs):
    song_title = select_random_song(songs)
    encrypted_text = vigenere_encrypt(plain_text, song_title)
    hash_code = generate_hash(encrypted_text)
    with open('decrypt.txt', 'a') as file:
```

```
    file.write(f"{encrypted_text}#{hash_code}
): {plain_text}\n")
```

```
    return f"{encrypted_text}#{hash_code}"
```

1) Hash Generation

Once the ciphertext is generated, a hash of the plaintext is also created using a robust hash function. This hash is appended to or stored with the ciphertext.

2) Purpose of Hash

The hash serves to verify the integrity of the message upon decryption. It ensures that the decrypted message matches the original plaintext, providing a layer of security against tampering.

C. Hashing Mechanism & Decryption Algorithm

The hashing mechanism serves as a critical component, ensuring the integrity of the data by generating a unique hash value of the plaintext. This value acts as a seal, verifying that the decrypted message remains untampered and authentic. On the other hand, the decryption algorithm is ingeniously designed to reverse the encryption process using the same culturally-derived keys, ensuring that only authorized parties can access and

reconstruct the original message. Together, these mechanisms form a formidable defense against a range of cyber threats, embodying a cutting-edge solution in the realm of digital security.

Hashing Mechanism

```
# TODO: Generate hash

def generate_hash(text):
    return
hashlib.sha256(text.encode()).hexdigest(
)
```

```
# TODO: Generate a pseudo-random key
from a list of song titles

def generate_key(songs, text_length):
    key = ''.join(songs).replace(' ',
        '').upper()
    return (key * (text_length //
        len(key) + 1))[:text_length]
```

1) Hash Function Utilization

After encrypting the plaintext, the algorithm generates a hash value of the original plaintext using a cryptographic hash function. Common hash functions include SHA-256 or SHA-3.

2) Hash Value Generation

The hash function takes the plaintext as input and produces a fixed-size string (hash). This hash acts as a digital fingerprint of the plaintext.

3) Storage of Hash Value

The generated hash value is stored along with the ciphertext or sent separately to the receiver. This hash is used later during decryption to verify the integrity of the decrypted message.

Decryption Algorithm

```
# TODO: Vigenère Cipher Decryption

def vigenere_decrypt(encrypted_text,
key):
    decrypted_text = ''
    for i in range(len(encrypted_text)):
        char = encrypted_text[i]
        if char.isalpha():
            shift = ord(key[i %
                len(key)].upper()) - 65
            decrypted_char =
                chr((ord(char.upper()) -
                shift - 65) % 26 + 65)
            decrypted_text +=
                decrypted_char
        else:
            decrypted_text += char
    return decrypted_text
```

1) Key Preparation

The same song title used in the encryption process is processed to form the decryption key, following the same method as key generation in encryption.

2) Decrypting the Ciphertext

The decryption process is essentially the reverse of the encryption algorithm. For each character in the ciphertext, the algorithm reverses the shift applied during encryption, based on the corresponding character in the key.

3) Reconstruction of Plaintext

The reverse shifting reconstructs the original plaintext from the ciphertext.

Verification Using Hash

```
# TODO: Find original text from
decrypt.txt

def find_original_text(encrypted_text,
hash_code):
    with open('decrypt.txt', 'r') as
        file:
        for line in file:
            if line.startswith
                (f"{encrypted_text}#{hash_code}"):
                return line.split(':', 1)[1].strip()
    return "No matching record found."
```

```
# TODO: Decrypt text and write to file

def decrypt(merged_text, song_title,
original_text):
    encrypted_text, hash_code =
merged_text.split('#')
    if generate_hash(encrypted_text) ==
hash_code:
        decrypted_text =
            vigenere_decrypt(encrypted_text,
                song_title)
        with open('decrypt.txt','a') as file:
            file.write(f"Original:
                {original_text}, Decrypted:
                {decrypted_text}, Hash:
                {hash_code}\n")
        return decrypted_text
    else:
        return "Invalid hash code. Decryption
            failed."
```

1) Hash Comparison

After decrypting the ciphertext, the algorithm generates a hash value of the decrypted message using the same hash function used during encryption. This new hash is compared with the original hash received with the ciphertext.

2) Integrity Check

If the hashes match, it confirms that the decrypted

message is identical to the original plaintext, ensuring the message's integrity and authenticity. If the hashes do not match, it indicates possible tampering or an error in the decryption process.

Security Aspects

The hash mechanism provides a strong layer of security by ensuring the integrity of the decrypted message. By using the same song title for both encryption and decryption, the system maintains consistency and ensures that only authorized parties can access the original message. The hash comparison acts as a safeguard against tampering, making the system robust against various cryptographic attacks.

IV. RESULT AND ANALYSIS

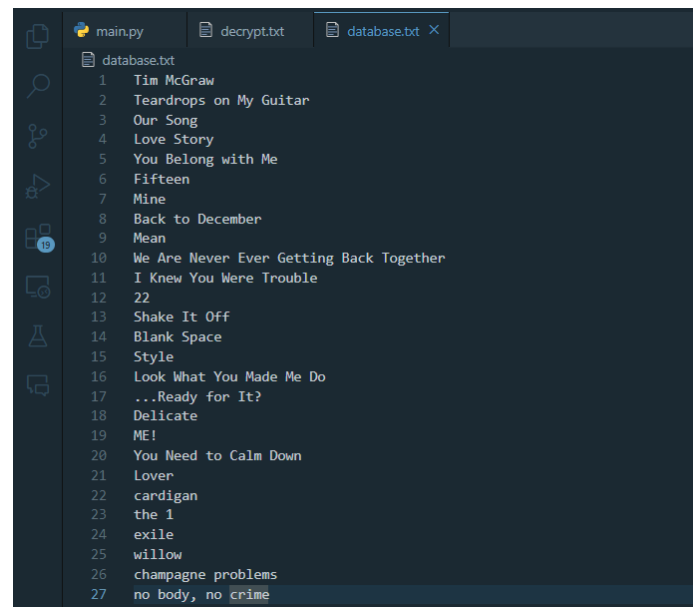
Program Overview

This program is an innovative application of the Vigenère Cipher, a well-established method in classical cryptography. What sets this implementation apart is its unique use of Taylor Swift's song titles as keys, integrating a creative element into the cryptographic process. The program offers users three distinct functionalities: encrypting a message, decrypting a message, and exiting the application. This user-friendly design caters to both encryption needs and the verification of encrypted messages.

Program Execution

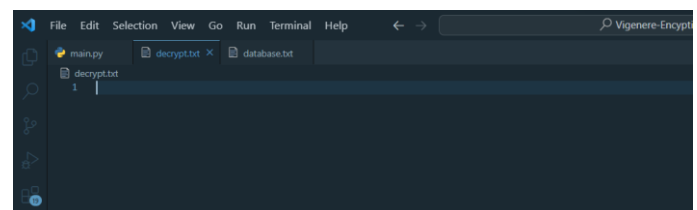
1) Encryption Process

- Upon choosing to encrypt, the user is prompted to enter their plaintext message.
- The program then selects a Taylor Swift song title at random from the database.txt file. This title serves as the key for the encryption algorithm.
- Utilizing the Vigenère Cipher, the plaintext is transformed into a ciphertext (encrypted text), ensuring that each letter in the message is shifted according to the key.
- The encrypted message is then concatenated with a SHA-256 hash code, creating a secure and unique identifier for the encrypted data.
- This combination of encrypted text and hash code is both displayed to the user and stored in decrypt.txt, enabling a record for future decryption.



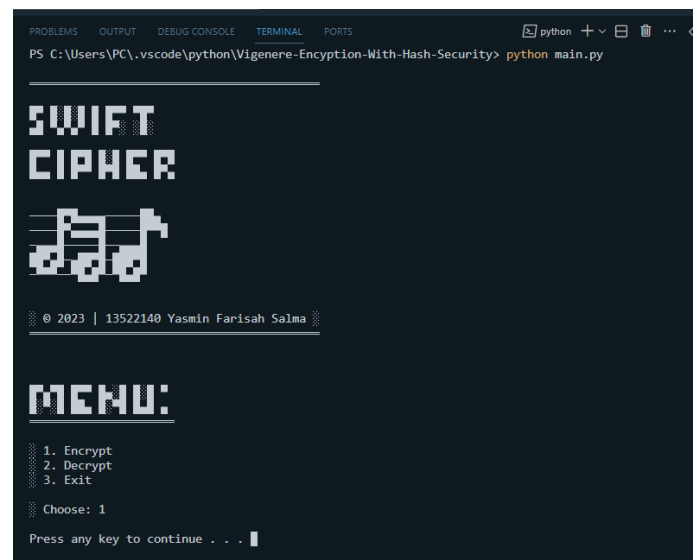
```
main.py | decrypt.txt | database.txt X
database.txt
1 Tim McGraw
2 Teardrops on My Guitar
3 Our Song
4 Love Story
5 You Belong with Me
6 Fifteen
7 Mine
8 Back to December
9 Mean
10 We Are Never Ever Getting Back Together
11 I Knew You Were Trouble
12 22
13 Shake It Off
14 Blank Space
15 Style
16 Look What You Made Me Do
17 ...Ready for It?
18 Delicate
19 ME!
20 You Need to Calm Down
21 Lover
22 cardigan
23 the 1
24 exile
25 willow
26 champagne problems
27 no body, no crime
```

Fig 4.1.1 Initiating Database for Storing Key Value




```
File Edit Selection View Go Run Terminal Help
main.py | decrypt.txt | database.txt
decrypt.txt
1 |
```

Fig 4.1.2 Initiating Database for Storing Text & Hash Value



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\PC\.vscode\python\Vigenere-Encryption-With-Hash-Security> python main.py

SWIFT
CIPHER



© 2023 | 13522140 Yasmin Farisah Salma

MENU:

1. Encrypt
2. Decrypt
3. Exit

Choose: 1

Press any key to continue . . . |
```

Fig 4.1.3 Main program and menu execution 1

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS python + v [ ] [ ]
ENCRIPTION

Plain text      : matematika diskrit
Encrypted text  : ETRPQSMGVE WGD0JBR#f4f891166084ce8f4e2df132a751ac7f11670fb43803d04
a61d4269b72270

CONTINUE

>> Do you want to encrypt/decrypt more? (y/n): y
Press any key to continue . . . █

```

Fig 4.1.4 Text encryption and continue

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS python + v [ ] [ ]
ENCRIPTION

Plain text      : teknik informatika
Encrypted text  : WIVVKK MQJZZOAMWNE#bf7ce20ba91ba29aa309dd944049dfba7f0f29333
1f2ed887fa24636fd2e5a0

CONTINUE

>> Do you want to encrypt/decrypt more? (y/n): n
Press any key to continue . . . █

```

Fig 4.1.5 Text encryption and stop

```

main.py | decrypt.txt M X | database.txt
decrypt.txt
1 ETRPQSMGVE WGD0JBR#f4f891166084ce8f4e2df132a751ac7f11670fb43803d043ffa61d4269b72270: matematika diskrit
2 WIVVKK MQJZZOAMWNE#bf7ce20ba91ba29aa309dd944049dfba7f0f2933361f2ed887fa24636fd2e5a0: teknik informatika
3


```

Fig 4.1.6 The Program Has Written the Database

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS python + v [ ] [ ]
PS C:\Users\PC\.vscode\python\Vigenere-Encryption-With-Hash-Security> python main.py

SWIFT
CIPHER



© 2023 | 13522140 Yasmin Farisah Salma

MENU:

1. Encrypt
2. Decrypt
3. Exit

Choose: 2

Press any key to continue . . . █

```

Fig 4.2.1 Main program and menu execution 2

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS python + v [ ] [ ]
DECRYPTION

Decrypted text  : ETRPQSMGVE WGD0JBR
Hash code       : f4f891166084ce8f4e2df132a751ac7f11670fb43803d043ffa61d4269b72270
Original Text   : matematika diskrit

CONTINUE

>> Do you want to encrypt/decrypt more? (y/n): y
Press any key to continue . . . █

```

Fig 4.2.2 Text decryption 1 and continue

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS python + v [ ] [ ]
DECRYPTION

Decrypted text  : WIVVKK MQJZZOAMWNE
Hash code       : bf7ce20ba91ba29aa309dd944049dfba7f0f2933361f2ed887fa24636fd2e5a0
Original Text   : teknik informatika

CONTINUE

>> Do you want to encrypt/decrypt more? (y/n): y
Press any key to continue . . . █

```

Fig 4.2.2 Text decryption 2 and continue

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS python + v [ ] [ ]
DECRYPTION

Decrypted text  : dummy
Hash code       : 10239132apaya

Decryption FAILED.
Invalid encrypted text or hash code.

CONTINUE

>> Do you want to encrypt/decrypt more? (y/n): n
Press any key to continue . . . █

```

Fig 4.2.2 Decryption failed

2) Decryption Process

- In decryption mode, the user inputs the encrypted text along with its corresponding hash code.
- The program searches `decrypt.txt` for a matching record. If found, it verifies the integrity of the message by comparing hash codes.
- On successful verification, the program decrypts the message, revealing the original plaintext. This step confirms the accuracy and reliability of the encryption process.
- If the input does not match any record or if the hash code is invalid, the program alerts the user, indicating a possible error or tampering with the encrypted data.

3) Exiting The Program

The exit option offers a straightforward way to terminate the program. It ensures that the user can easily conclude their session in a secure manner without leaving any operation incomplete.

Analysis

1) Encryption Effectiveness

Complexity and Uniqueness

The program's use of the Vigenère Cipher for encryption, combined with the novel approach of using Taylor Swift's song titles as keys, introduces a unique layer of complexity. This choice not only diversifies the encryption keys but also adds a cultural element to the cryptographic process. Each song title brings a different sequence of characters as a key, making the encryption less predictable and more secure against pattern recognition attacks.

Adaptability

The encryption method is highly adaptable to various lengths and types of text. It maintains the essence of the original message while transforming it into a secure format, ensuring that the encrypted text appears entirely different from the plaintext.

2) Decryption Accuracy

Integrity and Authenticity

The decryption process in the program is designed to precisely reverse the encryption, provided that the correct encrypted text and hash code are supplied. This ability to accurately retrieve the original text speaks to the program's effectiveness in maintaining the integrity and authenticity of data.

Hash Code Verification

The use of SHA-256 for hash code generation plays a crucial role in verifying the encrypted text. It ensures that any alterations to the encrypted message post-encryption are detected, thus safeguarding the fidelity of the information being decrypted.

3) User Interface

Ease of Use

The program's interface is crafted to be intuitive and straightforward. Users are smoothly guided through the process via a simple, menu-driven interface, making the program accessible to a wide range of users, regardless of their technical background.

Interactive Experience

The incorporation of system pause and screen clear commands contributes to a clean and interactive user experience. This not only prevents information overload but also ensures that users can focus on one step at a time, thereby reducing errors and enhancing user engagement.

4) Error Handling

Robustness

The program demonstrates robustness in error handling, especially during the decryption process. By

accurately identifying and informing the user of invalid or mismatched inputs, it prevents unauthorized access to the original text.

Security Implications

This feature is crucial for maintaining the security of the cryptographic process. It ensures that only individuals with the correct encrypted text and hash code can access the original message, thereby protecting the data from potential security breaches.

5) File Management

Efficient Data Storage

`decrypt.txt` serves as an efficient data storage mechanism. By recording each encryption and decryption operation, the file provides a reliable history of cryptographic activities.

Data Retrieval

The program's ability to quickly search and retrieve data from `decrypt.txt` demonstrates its efficiency in managing stored information. This is particularly important for verifying past encryptions or for audit purposes.

Organizational Aspect

The organized manner in which data is stored — linking encrypted messages, hash codes, and original texts — aids in maintaining a clear and easily navigable record. This organization is beneficial for tracking and reviewing cryptographic operations over time.

V. CONCLUSION

The program stands as a robust tool for encryption and decryption, balancing innovation with functionality. Its effective encryption mechanism, reliable decryption process, user-friendly interface, meticulous error handling, and efficient file management collectively contribute to its utility as a secure and accessible cryptographic solution. The unique integration of cultural elements into the cryptographic process further enhances its appeal, making it not just a tool for security but also a reflection of creative thinking in technology.

VII. ACKNOWLEDGMENT

The author express profound gratitude to the Almighty for His guidance and blessings, which enabled the successful completion of this paper. Special thanks are due to Dr. Ir. Rinaldi Munir, M.T., and Mr. Monterico Adrian, S.T., M.T., for their invaluable instruction during the IF2120 Discrete Mathematics course, Class K-03, which greatly contributed to the development of this paper. The author is also sincerely grateful for the supports and motivations received from family and friends during the learning process.

VIII. ATTACHMENT

The source code for this program can be accessed on GitHub:
<https://github.com/caernations/Vigenere-Encyption-With-Hash-Security/>

REFERENCES

- [1] Cryptography: Theory and Practice, Third Edition (Discrete Mathematics and Its Applications), 2005, by Douglas R. Stinson, Chapman and Hall/CRC
- [2] Munir, Rinaldi. (2020). "Aljabar Boolean (Bagian 1)". [https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/11-Aljabar-Boolean-\(2023\)-bagian1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/11-Aljabar-Boolean-(2023)-bagian1.pdf) (accessed on December 1st 2023).
- [3] Tutorialspoint (2023). "Cryptosystems". <https://www.tutorialspoint.com/cryptography/cryptosystems.htm> (accessed on December 4th 2023).
- [4] Sarkar, Subhasish (2020). "How Much Do You Know About The Vigenère Cipher?". <https://community.ibm.com/> (accessed on December 4th 2023).
- [5] Sarafianou, Eva (2019). "How Secure Are Encryption, Hashing, Encoding, and Obfuscation?". <https://auth0.com/blog/how-secure-are-encryption-hashing-encoding-and-obfuscation/> (accessed on December 4th 2023).
- [6] Kishore, N., & Kapoor, B. (2016). Attacks on and advances in secure hash algorithms. IAENG International Journal of Computer Science

STATEMENT

I hereby declare that the paper I wrote is my own writing, not an adaptation, or translation of someone else's paper, and not plagiarized.

Bandung, December 7th 2023



Yasmin Farisah Salma | 13522140