PROJECT-1

Encryption & Decryption with Poly-Alphabetic Substitution

Presented by :

Paras Dhiman(2021482)

Aekansh Kathunia (2021127)

Introduction

Objective: The primary goal of Project 1 is to implement encryption, decryption, and a brute-force attack using poly-alphabetic substitution.

Overview:

- **Encryption:** The process of converting plaintext into ciphertext using a cryptographic algorithm and a key.
- **Decryption:** Reversing the encryption process to obtain the original plaintext from ciphertext.

Significance:

- **Secure Communication:** Secure communication is crucial in network security to protect sensitive data from unauthorized access or interception.
- **Role of Encryption:** Encryption ensures that data transmitted over networks remains confidential and tamper-proof, enhancing the overall security posture.

Methodology

1. **Use of MD5 Hash Function for Plaintext Recognition:**
   - Utilization of the MD5 hash function to generate a unique hash for the plaintext.
   - The hash is appended to the plaintext to ensure recognizability during encryption and decryption processes.
2. **Poly-Alphabetic Substitution for Encryption and Decryption:**
   - Implementation of poly-alphabetic substitution technique for both encryption and decryption.
   - Each character of the plaintext is substituted with a character from the key, providing enhanced security through multiple alphabetic substitutions.
3. **Explanation of Brute-Force Attack Algorithm:**
   - Description of the brute-force attack algorithm employed to discover the encryption key.
   - Iterative process of trying all possible combinations of the key to decrypt the ciphertext and validate the resulting plaintext.
   - Objective is to exhaustively search for the correct key that satisfies the specified property of the plaintext.

Implementation

Programming Language Used:

- Python programming language chosen for the implementation.
- Python's versatility and ease of use make it suitable for cryptographic tasks.

Code Snippets:

```python
def md5_hash(input_string):
    md5 = hashlib.md5()
    md5.update(input_string.encode('utf-8'))
    hshd=md5.hexdigest()
    hsh=""
    for i in hshd:
        if i.isnumeric():
            hsh+=chr(ord('a')+int(i))
        else:
            hsh+=chr(10+ord(i))
    return hsh

def check_pi(input_string):
    if(len(input_string)<=32):
        return False
    ot=input_string[:len(input_string)-32]
    hsh=input_string[-32:]
    return md5_hash(ot)==hsh
```

```python
def encrypt(plain_text, key):
    cipher_text = ""
    key_length = len(key)

    for i in range(len(plain_text)):
        c = plain_text[i]
        key_char = key[i % key_length]
        cipher_text+=chr(ord('a')+((ord(c)-ord('a'))+(ord(key_char)-ord('a')))%26)

    return cipher_text

def decrypt(cipher_text, key):
    plain_text = ""
    key_length = len(key)

    for i in range(len(cipher_text)):
        c = cipher_text[i]
        key_char = key[i % key_length]
        plain_text+=chr(ord('a')+((ord(c)-ord('a'))-(ord(key_char)-ord('a')))%26)

    return plain_text
```

Brute Force Attack Analysis

- The brute-force attack algorithm implemented in the project exhaustively searches through all possible key combinations to decrypt the ciphertext. The algorithm iterates through four nested loops, each ranging from 0 to 25, representing the 26 possible characters in the key. For every combination of the four characters, the key is generated and applied to decrypt the ciphertext.
- Despite its simplicity, this brute-force attack approach guarantees the discovery of the correct key, albeit with a significant computational cost. The time complexity of this algorithm is O(264)O(264), making it feasible for small key spaces but impractical for larger ones. Further optimization or alternative strategies may be necessary for more extensive key spaces.

```
def brute_force(lst):
    for key1 in range(26):
        for key2 in range(26):
            for key3 in range(26):
                for key4 in range(26):
                    key = ''.join([chr(ord('a') + k) for k in [key1, key2, key3, key4]])
                    if(all([check_pi(decrypt(ciphertext,key)) for ciphertext in lst])):
                        return key
    return "WKEY"
```

Results

```
(base) parasdhiman@Parass-MacBook-Air-2 Downloads % python -u "/Users/parasdhiman/Downloads/polyalpha (1).py"
orignal text:
['helloyouaredull', 'whatsup', 'takeachillpill', 'itsreallycold', 'slowandsteady']
_____________________________________________________
plain text:
['helloyouaredullbcpmmaieeecoadlplkebgmdpjjjdfedk', 'whatsupfhlkcdlhimbpnhmikmelpdcfpgpeanjk', 'takeachillpillcgmeemnhekfbacioepckmippldgkikma', 'itsreallycoldjfgmkfioicccolckblhjf
oghjhceobcl', 'slowandsteadypbggjgpkehionflilnmacmjoohglgkkh']
_____________________________________________________
do they satisfy the property pi:
[True, True, True, True, True]
_____________________________________________________
encrypted ciphertext :
['lionscrwevhfypodgtpoemhgigrchpsnoieiqhslnnghihn', 'aldvwyshlpnehpkkqfsplqlmqiorhgirkthcrnn', 'xengegkkppskppfiqihorlhmjfdemshrgopkttofkolmqe', 'mxvtieoncgrnhniiqoiksmfegsoeofojnj
rilnkeiseep', 'wpryerguxidfcteiknjroikksrinmpqoegplsskipknml']
_____________________________________________________
decrypted ciphertext :
['helloyouaredullbcpmmaieeecoadlplkebgmdpjjjdfedk', 'whatsupfhlkcdlhimbpnhmikmelpdcfpgpeanjk', 'takeachillpillcgmeemnhekfbacioepckmippldgkikma', 'itsreallycoldjfgmkfioicccolckblhjf
oghjhceobcl', 'slowandsteadypbggjgpkehionflilnmacmjoohglgkkh']
_____________________________________________________
Brute forcing solution:
eedc
eedc
```

Our project successfully encrypted a set of recognizable plaintexts using a chosen key, producing ciphertexts that were then decrypted back to the original texts, confirming the accuracy of our encryption and decryption algorithms. Additionally, our implementation satisfied the defined property π for all plaintexts, ensuring their recognizability. Furthermore, a brute-force attack was performed to rediscover the original key, yielding the correct result. These results underscore the reliability and effectiveness of our cryptographic techniques in securing communication channels.

Conclusion

- **Key Findings:**
  - Successful implementation of encryption, decryption, and brute-force attack.
  - Effective use of poly-alphabetic substitution for secure communication.
- **Successful Implementation:**
  - Achieved project objectives.
  - Encryption and decryption processes executed flawlessly.
  - Encryption key successfully discovered through brute-force attack.
- **Importance of Secure Communication:**
  - Crucial for safeguarding sensitive data.
  - Demonstrates the significance of robust encryption in network security.

THANK YOU