Network Security

Assignment - 1

Transposition Cipher

Suyash Kumar - 2021293
Akash Kushwaha - 2021514

AIM

We were required to develop executable programs to encrypt, decrypt with a key, and launch a brute-force attack to discover the key given a cipher text. The encryption and decryption of the plaintext and ciphertext must be done using a Transposition Algorithm.

MD5 Hash Function

MD5 Message-Digest Algorithm 5 is a widely-used hash function producing a 128-bit hash value. Although it is known to have vulnerabilities and is susceptible to hash collisions, it is still used in various non-security applications and as an introductory algorithm in educational settings.

Function: calculate_md5

Purpose: To generate a fixed-length, 32-character hexadecimal hash from any given input string. Input: input_string - the string to hash.

Output: A 32-character hexadecimal string representing the MD5 hash. We are using hashlib library to implement this hash function. Each hex digit represents 4 bits, and thus 128 bits create a 32-digit hexadecimal number.

Encryption

Function: encrypt_transposition

Purpose: Encrypts plaintext using a columnar transposition method dictated by a key.

Input: plaintext - the text to be encrypted. key - a string used to determine the order of transposition.

Output: The ciphertext, where characters of the plaintext are rearranged according to the key.

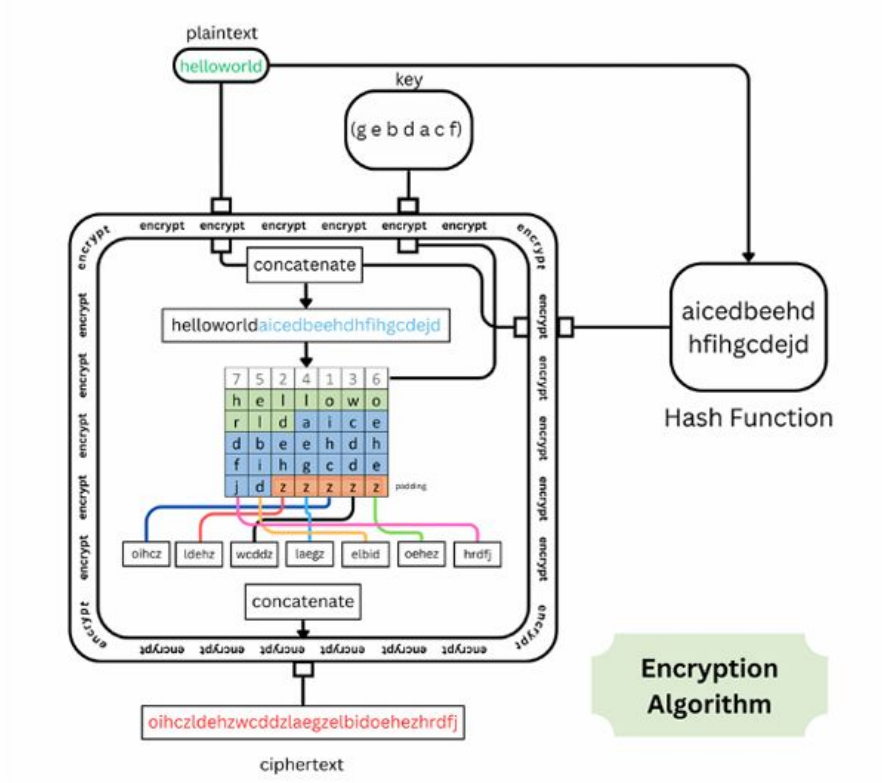
How It Works: Key Length and Mapping: Calculate the length of the key (Create a n). key_map which maps each character in the key to its position in a sorted version of the key. This determines the new order of columns in the encryption grid.

Padding: If the plaintext length isn't a multiple of perfectly into rows of length

Creating Rows:Divide the padded plaintext into rows, each of length

Columnar Rearrangement: n . Construct the ciphertext by collecting characters column by column based on the order defined in the key_map . For each column index from 0 to n-1 , find its actual position in the key and append the corresponding character from each row to the ciphertext. Concatenation: Combine the collected characters to form the final ciphertext string.

Encryption


plaintext
helloworld
key
(g e b d a c f)
encrypt
concatenate
helloworldaicedbeehdhfihgcdejd
7 5 2 4 1 3 6
h e l l o w o
r l d a i c e
d b e e h d h
f i h g c d e
j d z z z z z
padding
oihcz
ldehz
wcddz
laegz
elbid
oehez
hrdfj
concatenate
oihczldehzwcddzlaegzelbidoehezhrdfj
ciphertext
aicedbeehd
hfihgcdejd
Hash Function
Encryption
Algorithm

Decryption

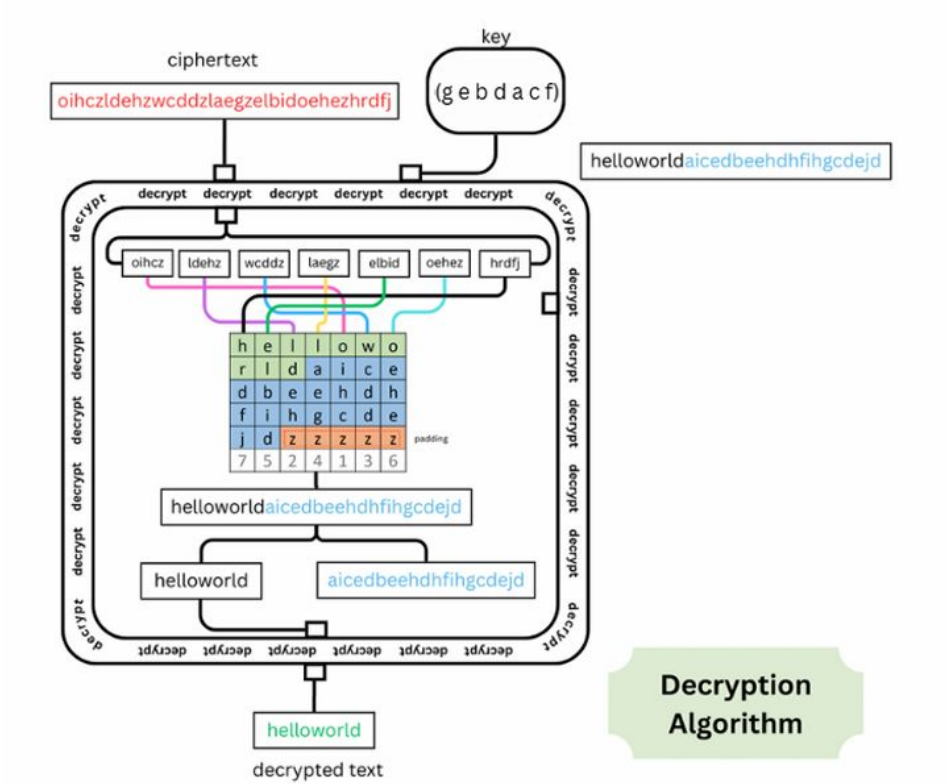
Decrypts ciphertext that was encrypted using a columnar transposition method.

Input: ciphertext - the text to be decrypted. key - the string used during the encryption process.

Output: The decrypted plaintext, reconstructed to its original form.

- Determine the length of the key (Create a n). key_map that identifies the position of each character in the key relative to its sorted order, similar to the encryption process.
- Calculate the number of rows in the transposition grid, which is the total length of the ciphertext divided by n . Create an array to store the columns, which will initially be empty strings.
- Populate the columns with characters from the ciphertext based on the original column positions determined by the key_map . Characters are extracted sequentially from the ciphertext and filled into their respective columns as per the positions outlined in the key_map .
- Build the plaintext by reading the characters row-wise across the columns, effectively reversing the columnar rearrangement process done during encryption.
- Strip any padding ($) that was added during the encryption process to align the plaintext with the key length.

Decryption


ciphertext
oihczldehzwcddzlaegzelbidoehezhrdfj
key
(g e b d a c f)
helloworldaicedbeehdhfihgcdejd
decrypt
oihcz
ldehz
wcddz
laegz
elbid
oehez
hrdfj
h e l l o w o
r l d a i c e
d b e e h d h
f i h g c d e
j d z z z z z
padding
7 5 2 4 1 3 6
helloworldaicedbeehdhfihgcdejd
helloworld
aicedbeehdhfihgcdejd
helloworld
decrypted text
Decryption Algorithm

Brute Force Attack

Purpose: To find the key that successfully decrypts given ciphertexts to their correct plaintext forms that satisfy the predefined property π.

Input: ciphertexts - a list of encrypted texts. alphabet - a sorted sequence of characters used in the key. max_key_length - the maximum permissible length of the key, based on computational feasibility.

Output: The key that correctly decrypts all the provided ciphertexts or None if no suitable key is found.

Brute Force Attack

- Use permutations of the given alphabet up to the max_key_length to generate all possible key combinations. Iterate through each generated key, testing it against all the ciphertexts.
- For each key, decrypt each ciphertext and verify the result using the property π (e.g., checking if the decrypted text concatenated with its hash matches the original format).
- The key validation function and checks the property π. check_key_with_ciphertext performs the decryption.
- If a key successfully decrypts a ciphertext and the resultant plaintext satisfies the property π for all given ciphertexts, this key is considered correct.If a key fails for any ciphertext, it is discarded, and the next key permutation is tested.
- If a valid key is discovered, return this key. Otherwise, return None indicating that no valid key could be found within the tested range.

Key

The key is a sequence of unique letters used to decide the transposition in the encryption and decryption processes.

- While keys often consist of numbers, they can also be made up of letters or symbols as we have used.
- If the maximum key length is 9, similar to numeric keys ranging from 1 to 9, our keys are permutations of any 9 unique letters in lowercase from the alphabets.
- And during the brute force attack we provide the sorted key, so that the unique characters are known.
- During decryption, the same key used in encryption is necessary to accurately reorder the transposed characters and retrieve the original plaintext.