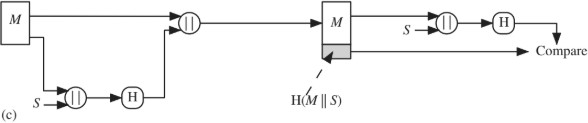
**Charotar University of Science and Technology [CHARUSAT]**

**Chandubhai S. Patel Institute of Technology [CSPIT]**

**U & P U. Patel Department of Computer Engineering**

**INS External Practical Exam**

***Aim:***



Refer the figure (c) given above. Bob is preparing to send message to Alice. Bob and Alice both secretly computes the code (s) without sharing on any communication channel using Deffi-Hallman key exchange algorithm. Bob applies SHA256 hash algorithm on original message (M) plus code (s) and send hash of (M||s) with original message to Alice. Alice will receive bundle of H(M||s) and original message (M). Alice will first append code (s) with received message (M) and produce hash of the message (H) that compare with H(M||s) to make sure message is not altered by any attackers.

**Task to perform:**

1. Use deffi-Hall-Man key exchange algorithm to calculate value of s which must be unique at sender and receiver side.

Implementation can be done using any programming language such as c, c++, java, python, c#, javascript, php etc.

***Program Code:***

import hashlib

if \_\_name\_\_ == "\_\_main\_\_":

    # public keys G and P

    P = int(input('P : '))

    G = int(input('G : '))

    # get the generated key for sender

    a = int(input('a : '))

    x = int(pow(G, a, P))

    # get the generated key for reciever

    b = int(input('b : '))

    y = int(pow(G, b, P))

    # Secret key for sender

    ka = int(pow(y, a, P))

    # Secret key for reciever

    kb = int(pow(x, b, P))

    # Source Side

    message = 'INS Practical Exam- Rajiv Kumar Gupta:- 18CE137'

    print('Message:', message, sep='\n')

    # Calculate Hash of message + private key

    hash = hashlib.sha256((message+str(ka)).encode()).hexdigest()

    print('Message + Private Key Hash:', hash, sep='\n')

    # Append Hash to the message

    cipherText = message + str(hash)

    print('Message + Hash:', cipherText, sep='\n')

    # Destination Side

    # Extract message and hash of message from decryptedText

    extMessage, extHash = cipherText[:-64], cipherText[-64:]

    print('Extracted Message:', extMessage, sep='\n')

    print('Extracted Hash:', extHash, sep='\n')

    # Calcluate hash of extracted message

    destHash = hashlib.sha256((extMessage+str(ka)).encode()).hexdigest()

    # if extracted hash and calculated hash matches i.e. are same then the message is not tempered

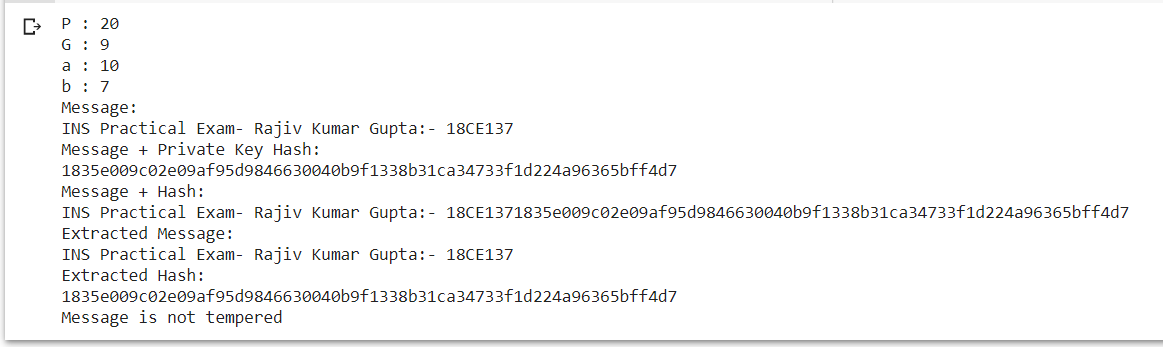
    if(destHash == extHash):

        print('Message is not tempered')

    else:

        print('Message is tempered.')

***Screenshot/Output:***



***Conclusion:***

By performing this practical we learned about how secret keys can be exchanged between two communicating parties using Diffie-Hellman key exchange algorithm. We also implemented and tested working of Diffie-Hellman key exchange algorithm.

|  |  |  |  |
| --- | --- | --- | --- |
| **Prepared By:** | Rajiv Kumar Gupta (18CE137) | **Date:** | 05-13-2021 |