|  |  |
| --- | --- |
|  | **Department of Information Technology** |

|  |  |
| --- | --- |
| **Semester** | T.E. Semester V – INFT |
| **Subject** | Computer Network Security |
| **Laboratory Teacher:** | Prof. Vinita Bhandiwad |
| **Theory**  **Teacher:** | Prof. Vinita Bhandiwad |
| **Laboratory** | Microsoft Teams |

|  |  |  |
| --- | --- | --- |
| **Student Name** | Pranali Sharad Darekar | |
| **Roll Number** | 19101B0032 | |
| **Grade and Subject Teacher’s Signature** |  |  |

|  |  |  |
| --- | --- | --- |
| **Experiment Number** | 3 | |
| **Experiment Title** | Implementation of Substitution Cipher – Affine Cipher | |
| **Objective** | To apply the knowledge of symmetric cryptography to implement classical ciphers. | |
| **Resources / Apparatus Required** | Hardware:  PC with the Configuration of Intel Dual core Processor or higher, Minimum 2 GB RAM, Minimum 40 GB Hard disk, Network interface card. | Software:  Python 3.9/Pycharm |
| **Theory** | An affine cipher, (like a shift cipher), is an example of a substitution cipher: In encryption using a substitution cipher, each time a given letter occurs in the plaintext, it always is replaced by the same ciphertext letter. For example, the plaintext letter ‘e’ might be replaced by the ciphertext letter ‘K’ each time it occurs. The method used for this replacement in affine encryption can be viewed as a generalization of the method used for encryption using a shift cipher. Shift ciphers are a particular type of affine cipher. The encryption key for an affine cipher is an ordered pair of integers, both of which come from the set {0, . . . , n − 1}, where n is the size of the character set being used (for us, the character set is the English alphabet, so we have n = 26). It is important to note that some of the possible pairs of integers from the set {0, . . . , n − 1} are not valid as affine encryption keys.  E = ( P \* K1 + K2 ) % 26  D = ( P \* K-1 – K2) % 26 | |
| **Code** | text = [**'a'**, **'b'**, **'c'**, **'d'**, **'e'**, **'f'**, **'g'**, **'h'**, **'i'**, **'j'**, **'k'**, **'l'**, **'m'**, **'n'**, **'o'**, **'p'**, **'q'**, **'r'**, **'s'**, **'t'**, **'u'**, **'v'**,  **'w'**, **'x'**, **'y'**, **'z'**]   def encrypt(plaintext, k1, k2):  result = **""** for i in plaintext:  if i != **" "**:  result += text[(text.index(i) \* k1 + k2) % 26]  else:  result += plaintext[i]  return result   def keyinverse(key):  for i in range(0, 26):  if ((i \* key) % 26 == 1):  return i   def decrypt(ciphertext, k1, k2):  result = **" "** for i in ciphertext:  if i != **" "**:  result += text[((text.index(i) - k2) \* k1) % 26]  else:  result += ciphertext[i]  return result  choice = 0 while (True):  print(**"-"**\*25,**"AFFINE CIPHERING METHOD"**,25\***"-"**)  print(**"1. Generate Cipher Text**\n**2. Generate Plain Text**\n**3. Quit"**)  choice = int(input())  if (choice == 3):  break  if (choice == 1):  plaintext = input(**"Enter Plain Text:**\n**"**)  k1 = int(input(**"Enter multiplicative key:"**))  k2 = int(input(**"Enter additive key:"**))  print(**"Cipher Text :"**, encrypt(plaintext, k1, k2))  elif (choice == 2):  ciphertext = input(**"Enter Cipher Text:**\n**"**)  k1 = int(input(**"Enter multiplicative key:"**))  k2 = int(input(**"Enter additive key:"**))  print(**"Plain Text :"**, decrypt(ciphertext, keyinverse(k1), k2))  else:  print(**"Invalid choice!!!"**) | |
| **Output** |  | |