**Lab Practicals – Phase – 2 (Traditional Ciphers)**

**Execute the following programs using gmp library in C , or in Python.**

**Note:** Do not use predefined functions from any Library or Header file, as far as possible. Instead write your own user define function for it.

For **Traditional Ciphers**, consider the plain text space as alphanumeric characters, which has following numeric values –

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PT=> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | space | a | b | c | d | e | f | g | h |
| Val=> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 |

Secret key, **K\_trad** = <first 25 distinct characters of your full name, without spaces>, if full name contains less than 25 distinct characters then repeat the characters from starting, also increment the character by one till it become distinct from all the previous characters. Assume I and J as same character. E.g. Name = Syed Taqi Ali, then K\_trad = **syedtaqil**uzfgvbrkm*wchnxoy*

**K\_roll1** = last\_digit\_of\_Rollno , **K\_roll2** = last\_2nd digit\_of\_Rollno

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Program** | **COs mapping** |
| 1. | Implement Caesar Cipher. Hard-code secret key as **K\_roll1.**  Input to the program is plain text in alphanumeric characters (with 36 letters), as mentioned in above table.  Execution Protocol:  Terminal $>gcc prg1.c -o prg1 -lgmp (compile)  $>prg1 system and network security (enter, execution)  Here plain text in Lower case is “system and network security”  **Or**  $>python3 prg1.py system and network security (enter)  Sample output in one line in UPPER case,  NETWORKXYZABTESTHAOI8390DYD | CO3 |
| 2. | Implement Affine Cipher. Hard-code secret key, k1 as **K\_roll1** and k2 as **K\_roll2.**  Input and execution protocol is same as previous, with program name as prg2(.c or .py) | CO3 |
| 3. | Implement Autokey Cipher, with initial key value as **K\_roll1** (hard-coded). Name it as prg3(.c or .py) | CO3 |
| 4. | Implement Playfair Cipher. Hard code secret key as **K\_trad** in 5x5 matrix form. | CO1, CO3 |
| 5. | Implement Vigenere Cipher, with hard-coded secret key as **K\_trad.** | CO3 |
| 6. | Implement Hill Cipher, with hard-coded 5x5 square matrix secret key as **K\_trad.** | CO3 |
| 7. | Implement Rotar Cipher, with hard code initial secret key mapping is like above table, in place of “Val=>” row substitute **K\_trad** characters followed by space and numbers 0 to 9. Consider i and j as distinct character while computing **K\_trad**.  e.g. for updated **K\_trad\*** (with I and j distinct) = **syedtaqil**uzfgvbrjm*wchkxnyo*0123456789   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | PT=> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | space | a | b | c | d | e | f | g | h | | Val=> | s | y | e | d | t | a | q | i | l | u | z | f | g | v | b | r | j | m | w |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | | c | h | k | x | n | y | o |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | CO3 |
| 8. | Implement keyless Rail Fence Cipher. | CO3 |
| 9. | Implement Keyless Transposition Cipher, by writing PT in row x row table and then transmitting it in column x column order. | CO3 |
| 10. | Implement Keyed Transposition Cipher, by consider the table constructed in program 7 as a permutation table, before using convert all the letters, present in the table in both rows, to its equivalent numeric values (as mentioned earlier), name it as **K\_permute**  e.g. the equivalent permutation table (with numeric values) **K\_permute** is,   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | |  | 29 | 35 | 15 | 14 | 30 | 11 | 27 | 19 | 22 | 31 | 36 | 16 | 17 | 32 | 12 | 28 | 20 | 23 | 33 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | | 13 | 18 | 21 | 34 | 24 | 35 | 25 | 10 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | CO3 |

In all the program print only exact output in a single line.

We also do **plagiarism** test of each program, if percentage of similarity is higher (around 70%) then marks will not be given. Actual percentage fixation will be decided later.

Course Outcomes:

|  |  |
| --- | --- |
| CO1 | Develop an understanding of information assurance as practiced in computer operating systems, distributed systems, networks and representative applications. |
| CO2 | Gain familiarity with prevalent network and distributed system attacks, defenses against them, and forensics to investigate the aftermath. |
| CO3 | Develop a basic understanding of cryptography, how it has evolved, and some key encryption techniques used today. |
| CO4 | Develop an understanding of security policies (such as authentication, integrity and confidentiality), as well as protocols to implement such policies in the form of message exchanges. |