VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

B.Tech. Winter Semester 2024-25

School Of Computer Science and Engineering (SCOPE)

Digital Assignment - I

Cryptography and Network Security Lab

Apurva Mishra: 22BCE2791

Date: 9 Feb, 2025

Contents

1 Ceaser Cipher

1.1 Code

Code 0: main.l

```c
#include <ctype.h>
#include <stdio.h>
#include <string.h>

#define MAX_LEN 1000

void encrypt(char *text, int key) {
  for (int i = 0; text[i] != '\0'; i++) {
    if (isalpha(text[i])) {
      char base = isupper(text[i]) ? 'A' : 'a';
      text[i] = (text[i] - base + key) % 26 + base;
    }
  }
}

void decrypt(char *text, int key) {
  for (int i = 0; text[i] != '\0'; i++) {
    if (isalpha(text[i])) {
      char base = isupper(text[i]) ? 'A' : 'a';
      text[i] = (text[i] - base - key + 26) % 26 + base;
    }
  }
}

int main() {
  char text[MAX_LEN];
  int key;
  int op;

  printf("Enter text: ");
  fgets(text, MAX_LEN, stdin);
  text[strcspn(text, "\n")] = '\0';

  printf("Enter key value: ");
  scanf("%d", &key);

  printf("Choose operation: 1 for Encryption, 2 for Decryption: ");
  scanf("%d", &op);

  switch (op) {
  case 1:
    encrypt(text, key);
    printf("Encrypted text: %s\n", text);
    break;
  case 2:
    decrypt(text, key);
    printf("Decrypted text: %s\n", text);
```

```
48      break;
49    default:
50      printf("Invalid choice!\n");
51      break;
52    }
53
54    return 0;
55  }
```

1.2 Output

```
da/ass1/q1 via C v16.0.0-clang
❯ just run
zig cc main.c -o main
./main
Enter text: there
Enter key value: 3
Choose operation: 1 for Encryption, 2 for Decryption: 1
Encrypted text: wkhuh

da/ass1/q1 via C v16.0.0-clang took 10s
❯ just run
zig cc main.c -o main
./main
Enter text: wkhuh
Enter key value: 3
Choose operation: 1 for Encryption, 2 for Decryption: 2
Decrypted text: there

da/ass1/q1 via C v16.0.0-clang took 7s
❯
```

2 Playfair Cipher

2.1 Code

Code 0: main.l

```
1  #include <ctype.h>
2  #include <stdio.h>
3  #include <string.h>
4
5  #define SIZE 5
6  #define MAX_TEXT 100
7
```

```
char keySquare[SIZE][SIZE];

void generateKeySquare(const char *key) {
  int map[26] = {0};
  int x = 0, y = 0;
  char processedKey[26] = "";
  int index = 0;

  for (int i = 0; key[i] != '\0'; i++) {
    char ch = toupper(key[i]);
    if (ch == 'J')
      ch = 'I';
    if (!map[ch - 'A'] && isalpha(ch)) {
      map[ch - 'A'] = 1;
      processedKey[index++] = ch;
    }
  }

  for (char ch = 'A'; ch <= 'Z'; ch++) {
    if (ch == 'J')
      continue;
    if (!map[ch - 'A']) {
      processedKey[index++] = ch;
    }
  }

  index = 0;
  for (int i = 0; i < SIZE; i++) {
    for (int j = 0; j < SIZE; j++) {
      keySquare[i][j] = processedKey[index++];
    }
  }
}

void findPosition(char ch, int *row, int *col) {
  if (ch == 'J')
    ch = 'I';
  for (int i = 0; i < SIZE; i++) {
    for (int j = 0; j < SIZE; j++) {
      if (keySquare[i][j] == ch) {
        *row = i;
        *col = j;
        return;
      }
    }
  }
}

void prepareText(char *text) {
  int len = strlen(text);
  for (int i = 0; i < len; i++) {
    text[i] = toupper(text[i]);
    if (text[i] == 'J')
```

```
      text[i] = 'I';
  }

  char newText[MAX_TEXT];
  int newIndex = 0;

  for (int i = 0; i < len; i++) {
    if (!isalpha(text[i]))
      continue;
    newText[newIndex++] = text[i];
    if (i + 1 < len && text[i] == text[i + 1]) {
      newText[newIndex++] = 'X';
    }
  }

  if (newIndex % 2 != 0) {
    newText[newIndex++] = 'X';
  }
  newText[newIndex] = '\0';
  strcpy(text, newText);
}

void playfairCipher(char *text, int encrypt) {
  for (int i = 0; i < strlen(text); i += 2) {
    int r1, c1, r2, c2;
    findPosition(text[i], &r1, &c1);
    findPosition(text[i + 1], &r2, &c2);

    if (r1 == r2) {
      text[i] = keySquare[r1][(c1 + encrypt + SIZE) % SIZE];
      text[i + 1] = keySquare[r2][(c2 + encrypt + SIZE) % SIZE];
    } else if (c1 == c2) {
      text[i] = keySquare[(r1 + encrypt + SIZE) % SIZE][c1];
      text[i + 1] = keySquare[(r2 + encrypt + SIZE) % SIZE][c2];
    } else {
      text[i] = keySquare[r1][c2];
      text[i + 1] = keySquare[r2][c1];
    }

    text[i] = tolower(text[i]);
    text[i + 1] = tolower(text[i + 1]);
  }
}

int main() {
  char key[MAX_TEXT], text[MAX_TEXT];
  int op;

  printf("Enter key: ");
  fgets(key, MAX_TEXT, stdin);
  key[strcspn(key, "\n")] = '\0';

  generateKeySquare(key);
```

```c

    printf("Enter text: ");
    fgets(text, MAX_TEXT, stdin);
    text[strcspn(text, "\n")] = '\0';

    prepareText(text);

    printf("Choose operation (1 for Encryption, 2 for Decryption): ");
    scanf("%d", &op);

    switch (op) {
    case 1:
      playfairCipher(text, 1);
      printf("Encrypted text: %s\n", text);
      break;
    case 2:
      playfairCipher(text, -1);
      printf("Decrypted text: %s\n", text);
      break;
    default:
      printf("Invalid choice!\n");
      break;
    }

    return 0;
}
```

2.2 Output

```
da/ass1/q2 via C v16.0.0-clang
❯ just run
zig cc main.c -o main
./main
Enter key: batter
Enter text: heater
Choose operation (1 for Encryption, 2 for Decryption): 1
Encrypted text: grterb

da/ass1/q2 via C v16.0.0-clang took 10s
❯ just run
zig cc main.c -o main
./main
Enter key: batter
Enter text: grterb
Choose operation (1 for Encryption, 2 for Decryption): 2
Decrypted text: heater

da/ass1/q2 via C v16.0.0-clang took 7s
❯
```

3 Rail Fence Cipher

3.1 Code

Code 0: main.l

```c
#include <ctype.h>
#include <stdio.h>
#include <string.h>

#define MAX_LEN 1000

void encrypt(char *text, int key) {
  int len = strlen(text);
  char rail[key][len];

  for (int i = 0; i < key; i++) {
    for (int j = 0; j < len; j++) {
      rail[i][j] = '\n';
    }
  }

  int row = 0, dir_down = 0;
```

```
  for (int i = 0; i < len; i++) {
    if (row == 0 || row == key - 1) {
      dir_down = !dir_down;
    }
    rail[row][i] = text[i];
    row += (dir_down) ? 1 : -1;
  }

  printf("Encrypted Text: ");
  for (int i = 0; i < key; i++) {
    for (int j = 0; j < len; j++) {
      if (rail[i][j] != '\n') {
        printf("%c", rail[i][j]);
      }
    }
  }
  printf("\n");
}

void decrypt(char *cipher, int key) {
  int len = strlen(cipher);
  char rail[key][len];

  for (int i = 0; i < key; i++) {
    for (int j = 0; j < len; j++) {
      rail[i][j] = '\n';
    }
  }

  int row = 0;
  int dir_down = 0;
  for (int i = 0; i < len; i++) {
    if (row == 0 || row == key - 1) {
      dir_down = !dir_down;
    }
    rail[row][i] = '*';
    row += (dir_down) ? 1 : -1;
  }

  int index = 0;
  for (int i = 0; i < key; i++) {
    for (int j = 0; j < len; j++) {
      if (rail[i][j] == '*' && index < len) {
        rail[i][j] = cipher[index++];
      }
    }
  }

  row = 0, dir_down = 0;
  printf("Decrypted Text: ");
  for (int i = 0; i < len; i++) {
    if (row == 0 || row == key - 1) {
      dir_down = !dir_down;
```

```c
    }
    printf("%c", rail[row][i]);
    row += (dir_down) ? 1 : -1;
  }
  printf("\n");
}

int main() {
  char text[MAX_LEN];
  int key, choice;

  printf("Enter text: ");
  fgets(text, MAX_LEN, stdin);
  text[strcspn(text, "\n")] = '\0';

  printf("Enter key (number of rails): ");
  scanf("%d", &key);

  printf("Choose operation: 1 for Encryption, 2 for Decryption: ");
  scanf("%d", &choice);
  getchar();

  if (choice == 1) {
    encrypt(text, key);
  } else if (choice == 2) {
    decrypt(text, key);
  } else {
    printf("Invalid choice!\n");
  }

  return 0;
}
```

3.2 Output

```
da/ass1/q3 via C v16.0.0-clang
❯ just run
zig cc main.c -o main
./main
Enter text: scatter
Enter key (number of rails): 3
Choose operation: 1 for Encryption, 2 for Decryption: 1
Encrypted Text: stctear

da/ass1/q3 via C v16.0.0-clang took 8s
❯ just run
zig cc main.c -o main
./main
Enter text: stctear
Enter key (number of rails): 3
Choose operation: 1 for Encryption, 2 for Decryption: 2
Decrypted Text: scatter

da/ass1/q3 via C v16.0.0-clang took 16s
❯
```

4 Vigenere Cipher

4.1 Code

Code 0: main.l

```c
#include <ctype.h>
#include <stdio.h>
#include <string.h>

#define MAX_LEN 1000

void encrypt(char *text, char *key) {
  int textLen = strlen(text);
  int keyLen = strlen(key);
  char encryptedText[MAX_LEN];

  for (int i = 0, j = 0; i < textLen; i++) {
    if (isalpha(text[i])) {
      char base = isupper(text[i]) ? 'A' : 'a';
      char keyBase = isupper(key[j % keyLen]) ? 'A' : 'a';
      encryptedText[i] =
          (text[i] - base + (key[j % keyLen] - keyBase)) % 26 + base;
```

```
      j++;
    } else {
      encryptedText[i] = text[i];
    }
  }
  encryptedText[textLen] = '\0';
  printf("Encrypted Text: %s\n", encryptedText);
}

void decrypt(char *text, char *key) {
  int textLen = strlen(text);
  int keyLen = strlen(key);
  char decryptedText[MAX_LEN];

  for (int i = 0, j = 0; i < textLen; i++) {
    if (isalpha(text[i])) {
      char base = isupper(text[i]) ? 'A' : 'a';
      char keyBase = isupper(key[j % keyLen]) ? 'A' : 'a';
      decryptedText[i] =
          (text[i] - base - (key[j % keyLen] - keyBase) + 26) % 26 + base;
      j++;
    } else {
      decryptedText[i] = text[i];
    }
  }
  decryptedText[textLen] = '\0';
  printf("Decrypted Text: %s\n", decryptedText);
}

int main() {
  char text[MAX_LEN], key[MAX_LEN];
  int choice;

  printf("Enter text: ");
  fgets(text, MAX_LEN, stdin);
  text[strcspn(text, "\n")] = '\0';

  printf("Enter key: ");
  fgets(key, MAX_LEN, stdin);
  key[strcspn(key, "\n")] = '\0';

  printf("Choose operation: 1 for Encryption, 2 for Decryption: ");
  scanf("%d", &choice);
  getchar(); // Consume newline

  if (choice == 1) {
    encrypt(text, key);
  } else if (choice == 2) {
    decrypt(text, key);
  } else {
    printf("Invalid choice!\n");
  }

```

```
71    return 0;
72  }
```

4.2 Output

```
da/ass1/q4 via C v16.0.0-clang took 6s
❯ just run
zig cc main.c -o main
./main
Enter text: there
Enter key: air
Choose operation: 1 for Encryption, 2 for Decryption: 1
Encrypted Text: tpvrm

da/ass1/q4 via C v16.0.0-clang took 8s
❯ just run
zig cc main.c -o main
./main
Enter text: tpvrm
Enter key: air
Choose operation: 1 for Encryption, 2 for Decryption: 2
Decrypted Text: there

da/ass1/q4 via C v16.0.0-clang took 7s
❯
```