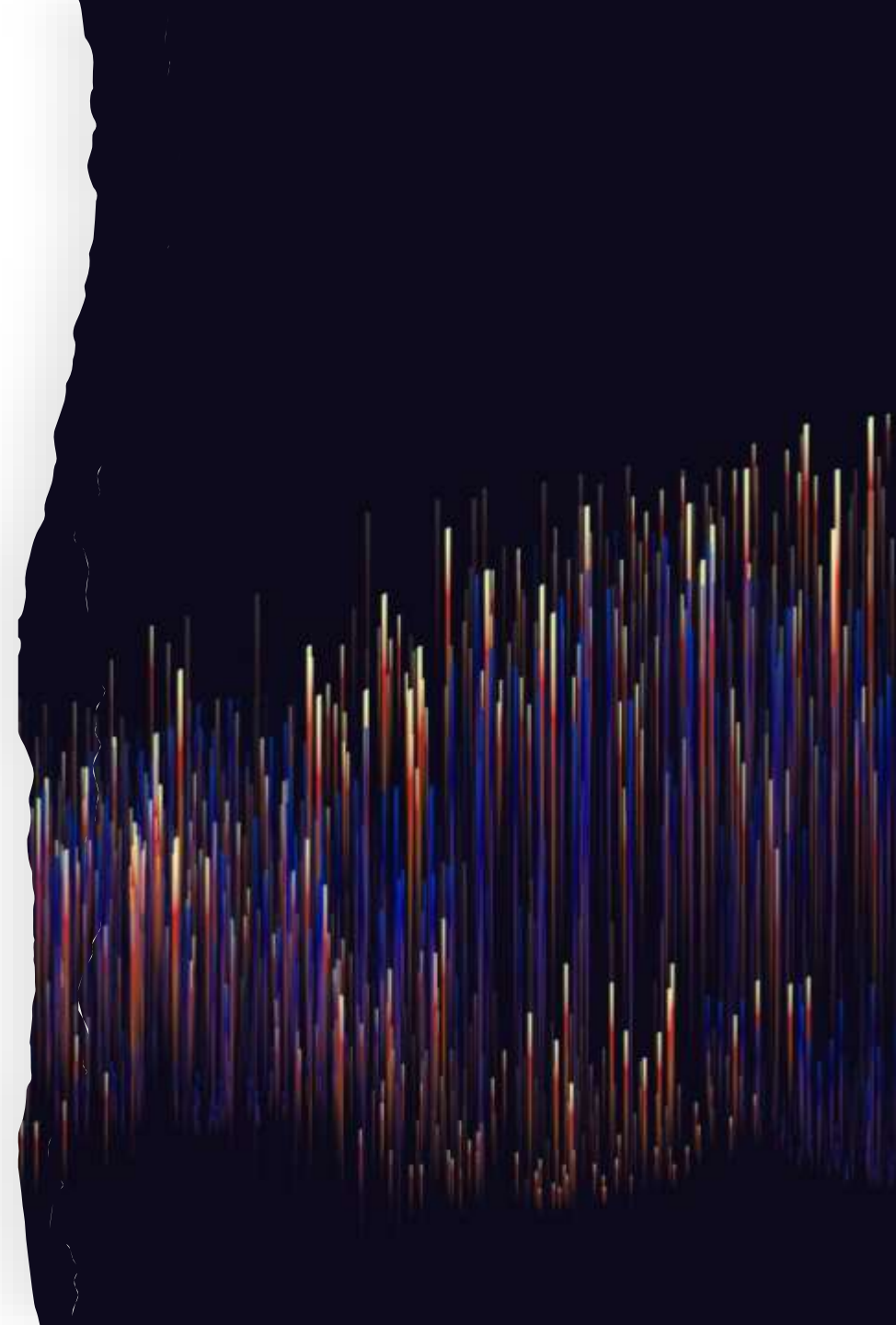


# TEAM:5

Message Encryption and  
Decryption with OTP



# REQUIREMENTS:

- → Pointers :-
- → DEV C++ :-
- → C Compiler :-

+

•

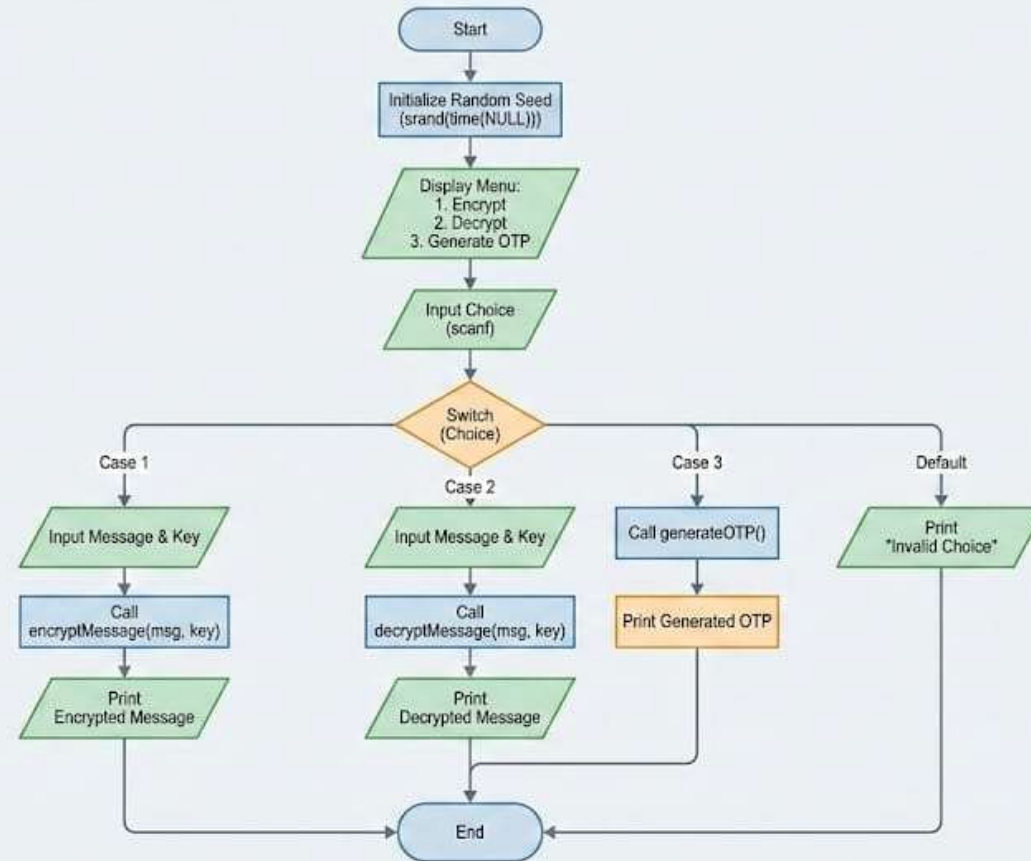
○

# INTRODUCTION

- 1. The program demonstrates core cryptography concepts.
- 2. It offers encryption, decryption, and OTP generation features.
- 3. Encryption works by shifting characters based on a numeric key.
- 4. Decryption reverses the shift to restore the original message.
- 5. OTPs are generated using seeded random numbers for security.
- 6. A menu-driven interface allows users to select an operation.
- 7. Users input both a message and a secret key for secure processing.
- 8. Results are displayed instantly for quick understanding.
- 9. The code modifies the string directly using loops.
- 10. ASCII value manipulation showcases C's efficient string handling.
- 11. The project is ideal for educational demonstrations and prototypes.
- 12. It introduces symmetric encryption principles that can be scaled to advanced systems.

# FLOWCHART

C Program Flowchart: Encryption, Decryption & OTP Generation



# ALGORITHM

- ALGORITHM: Message Encryption & Decryption System
  - START
  - MAIN MENU
- Display:
- Generate OTP
  - Encrypt Message (OTP)
  - Decrypt Message (OTP)
  - Caesar Cipher Encrypt
  - Caesar Cipher Decrypt
  - Exit
- Input choice  
Call respective function
- 
- GENERATE OTP
- Input: message length  
FOR i = 0 to length - 1  
otp[i] = random(0, 25)  
Display and save OTP
- 
- ENCRYPT WITH OTP
- Input: message, otp  
FOR each character in message  
IF alphabet THEN  
char\_value = character - 'A' (or 'a')  
encrypted = (char\_value + otp[i]) mod 26  
ciphertext[i] = encrypted + 'A' (or 'a')  
ELSE  
ciphertext[i] = character  
Display ciphertext

- DECRYPT WITH OTP

Input: ciphertext, otp

FOR each character in ciphertext

IF alphabet THEN

char\_value = character - 'A' (or 'a')

decrypted = (char\_value - otp[i] + 26) mod 26

plaintext[i] = decrypted + 'A' (or 'a')

ELSE

plaintext[i] = character

Display plaintext

- CAESAR ENCRYPT

Input: message, shift\_key

FOR each character in message

IF alphabet THEN

char\_value = character - 'A' (or 'a')

encrypted = (char\_value + shift\_key) mod 26

ciphertext[i] = encrypted + 'A' (or 'a')

ELSE

ciphertext[i] = character

Display ciphertext

- CAESAR DECRYPT

Input: ciphertext, shift\_key

FOR each character in ciphertext

IF alphabet THEN

char\_value = character - 'A' (or 'a')

ELSE

plaintext[i] = character

Display plaintext

END

•

# PROGRAM MODULE

## 1. Header Inclusion Module

```
#include <stdio.h> // Input/output functions
#include <string.h> // String manipulation
#include <stdlib.h> // Standard library functions
#include <time.h> // Time functions for random seed
```

## 2. Encryption Module

```
void encryptMessage(char *msg, int key)
{ for (int i = 0; msg[i] != '\0'; i++)
{ msg[i] = msg[i] + key; // Shift characters by key
}
}
```

Purpose: Shifts each character's ASCII value forward by the key. [trytoprogram](#)

## 3. Decryption Module

```
void decryptMessage(char *msg, int key)
{ for (int i = 0; msg[i] != '\0'; i++)
{ msg[i] = msg[i] - key; // Reverse shift by key
} }
```

Purpose: Reverses encryption by subtracting the key from each character's ASCII value.

## 4. OTP Generation Module

```
int generateOTP()
{
return rand() % 900000 + 100000; // Generates 6-digit OTP (100000-999999)
}
```

Purpose: Creates random 6-digit one-time password using seeded rand(). [scaler](#)

## 5. Main Control Module

```
int main() { char message[100]; // Message buffer
int key, choice; // Key and menu choice variables
srand(time(NULL)); // Initialize random seed
// Menu display and input handling
printf("=== Message Encourterment / Discourterment & OTP Generation ===\n"); // ... menu options and switch statement }
```

# LOGIC WISE MODULE

- Case 1: Encrypt Message
- Ask user for the message (string).
- Ask user for key (numeric shift).
- Call `encryptMessage(message, key)`.
- Display encrypted message.
- Purpose: Increase ASCII value of each character → unreadable text.
- 
- Case 2: Decrypt Message
- Ask user for encrypted message.
- Ask user for same key used earlier.
- Call `decryptMessage(message, key)`.
- Show decrypted original message.
- Purpose: Decrease ASCII value back → recover original text.
- 
- Case 3: OTP Generation
- Call `generateOTP()`.
- Display generated OTP to user.
- Purpose: Produce a unique random 6-digit security code.
- 
- Case 4: Exit Program

# MODULE INTEGRATION

```
•  #include <stdio.h>

•  #include <string.h>

•  #include <stdlib.h>

•  #include <time.h>

•  void encryptMessage(char *msg, int key) {

•      for (int i = 0; msg[i] != '\0'; i++) {

•          msg[i] = msg[i] + key; // Shift characters

•      }

•  }

•  void decryptMessage(char *msg, int key) {

•      for (int i = 0; msg[i] != '\0'; i++) {

•          msg[i] = msg[i] - key; // Reverse shift

•      }

•  }

•  int generateOTP() {

•      return rand() % 900000 + 100000; // Always a 6-digit number

•  }

•  int main() {

•      char message[100];

•      int key, choice;

•      srand(time(NULL)); // Initialize random seed

•      printf("=== Message Encounterment / Discounterment & OTP Generation ===\n");

•      printf("1. Encrypt Message\n");

•      printf("2. Decrypt Message\n");
```

```

•   printf("3. Generate OTP\n");
•   printf("Enter your choice: ");
•   scanf("%d", &choice);
•   getchar(); // Clear newline from buffer
•   switch (choice) {
•       case 1:
•           printf("Enter message to encrypt: ");
•           fgets(message, sizeof(message), stdin);
•           printf("Enter key (number): ");
•           scanf("%d", &key);
•           encryptMessage(message, key);
•           printf("Encrypted Message: %s\n", message);
•           break;
•       case 2:
•           printf("Enter message to decrypt: ");
•           fgets(message, sizeof(message), stdin);
•           printf("Enter key used during encryption: ");
•           scanf("%d", &key);
•           decryptMessage(message, key);
•           printf("Decrypted Message: %s\n", message);
•           break;
•       case 3:
•           printf("Generated OTP: %d\n", generateOTP());
•           break;
•       default:
•           printf("Invalid choice.\n");
•   }
•   return 0;

```

# OUTPUT FORMAT

- `=== Message Encounterment / Discounterment & OTP  
Generation ===`
- `1. Encrypt Message`
- `2. Decrypt Message`
- `3. Generate OTP`
- `Enter your choice: 3`
- `Generated OTP: 102599`
- `-----`
- `Process exited after 93.27 seconds with return  
value 0`
- `Press any key to continue . . .`



# THANK YOU

- **PROJECT DONE BY:-**

**25A31A0144 -CHARAN**

**25A31A0133 -MAHADEV**

**25A31A0107 -SWAPNA PRIYANKA**

**25A31A0106 -LARHANA**

