ByteCrypt

Introduction to Computers and Programming in C (ES202)

Project Report

Submitted by: Aayan Khan

Section: 1CSE-1Y

Enrolment No.: A2305225054

Batch: 2025-29

Teacher: Dr. Ashish Mani

AMITY UNIVERSITY

Objective

To develop a C-based project capable of encrypting and decrypting files by performing direct bit-level manipulation of their data. The project aims to demonstrate practical understanding of low-level file handling, buffer-based processing, and the use of bitwise operators—specifically XOR—to secure digital content. The system will read binary data in chunks, apply a dynamically generated key derived from a user-provided password, and overwrite the file in place, ensuring both efficiency and reversibility of the process.

Apparatus/Software Used

- A computer with a C Compiler (VSCode)
- Operating System: Windows

Procedure

1) Create a file “main.c”
    a. Include the main function. Take two inputs from the user through the command line:
        - File Path
        - Password (used for generating key for encryption)
    b. Handle any case of invalid usage. Display a message showing the correct usage method.
2) Create a file “crypt.c”
    a. Create a function to generate the key value. Take the password as a parameter.
        - Utilize the Daniel Bernstein (DJB2) Rolling Algorithm.
        - Assign a base value to the key (5381 for best results).
        - Use a rolling multiplier (33 for best results)
        - For each character in the password:

$$key = key * multiplier + (int)character;$$

        - Finally, return key % 256 since XOR works on bytes (0-255)
    b. Now create the main encrypting/decrypting function . Take the password and file path as parameters.
        - Open the file in binary read-write mode (rb+) using a file pointer. If the file path is invalid, display an appropriate error message and stop execution.
        - Generate the numeric key value from the password using the key-generation function.
        - Start an infinite loop
            - Create a chunk buffer (char array) of size 4096 bytes.
            - Read from the file using fread() and store the number of bytes read in a variable, say, n.
            - If n is 0 that means we have reached the end of the file so break the loop.
            - For each byte read in the chunk till n:

$$chunk[byte] = chunk[byte] \text{ ^ } key;$$

            - Seek the file pointer backwards by n bytes from the current position using fseek().

        - Write the updated chunk back into the file using fwrite().
        - Flush the changes made to the file using fflush().
    - Close the file using fclose(). Display a message to declare that the file has been successfully secured.

c. Compile both the files together and test the program with different file types and different passwords to verify correct operation.
d. Record and present the sample outputs, and tabulate all relevant data.

(P.T.O. ->)

Programs

main.c

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

void cryptify(char file[], char password[]);
int seed(char password[]);
      You, 2 weeks ago • Initial …
int main(int argc, char* argv[]) {
    system("cls");
    printf("    ____        __       ______                __ \n   / __ )__  __/ /___
    if (argc != 3) {
        printf("\x1B[31mUsage: <file_path> <password>\x1B[0m");
        return 0;
    }
    char file[256], password[32];
    strcpy(file, argv[1]);
    strcpy(password, argv[2]);
    cryptify(file, password);
    return 0;
}
```

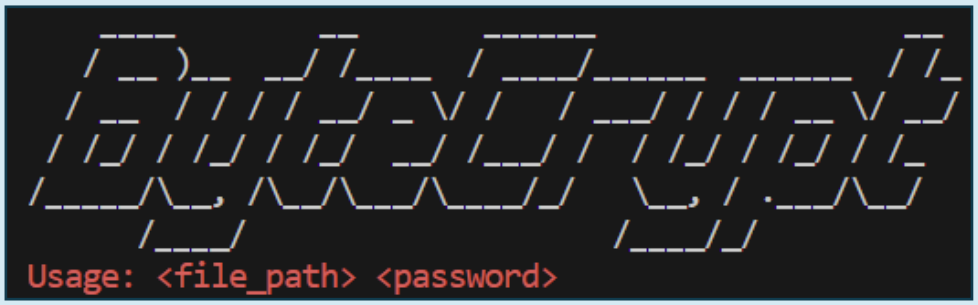
#crypt.c

```c
#include <stdio.h>
#include <Windows.h>

int seed(char password[]) {
    unsigned long key = 5381, multiplier = 33;
    for (int i = 0; password[i] != '\0';i++) {
        key = key * multiplier + (int)password[i];  //DJB (Daniel Bernstein) Algo
    }
    return key % 256;
}
void cryptify(char file[], char password[]) {
    SetConsoleOutputCP(CP_UTF8);
    FILE* file_pointer = fopen(file, "rb+");
    if (file_pointer == NULL) {
        printf("\x1B[31mInvalid File Path. ❌\x1B[0m");
        return;
    }
    int key = seed(password);
    while (1) {
        char chunk[4096];
        size_t bytes_read = fread(&chunk, 1, sizeof(chunk), file_pointer);
        if (bytes_read == 0) {
            break;
        }
        for (size_t i = 0; i < bytes_read; i++) {
            chunk[i] ^= key;
        }
        fseek(file_pointer, -bytes_read, SEEK_CUR);
        fwrite(&chunk, 1, bytes_read, file_pointer);
        fflush(file_pointer);
    }
    fclose(file_pointer);
    printf("\x1B[32mFile has been secured.\x1B[0m ✅");
    return;
}
```

Sample Output

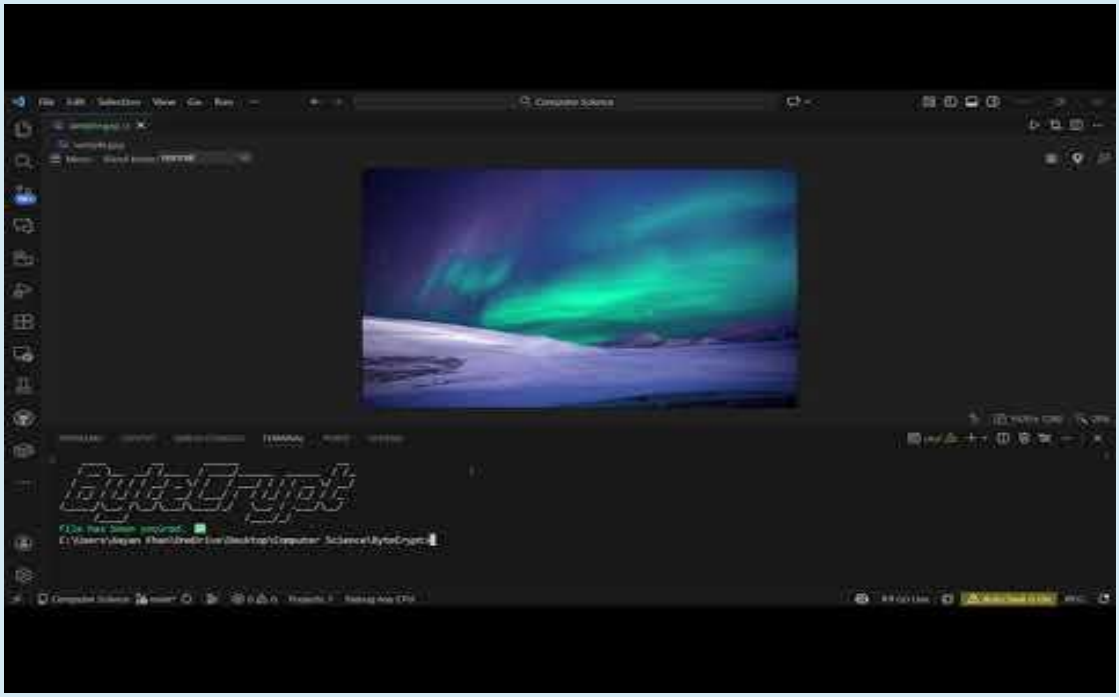
 # Invalid Usage



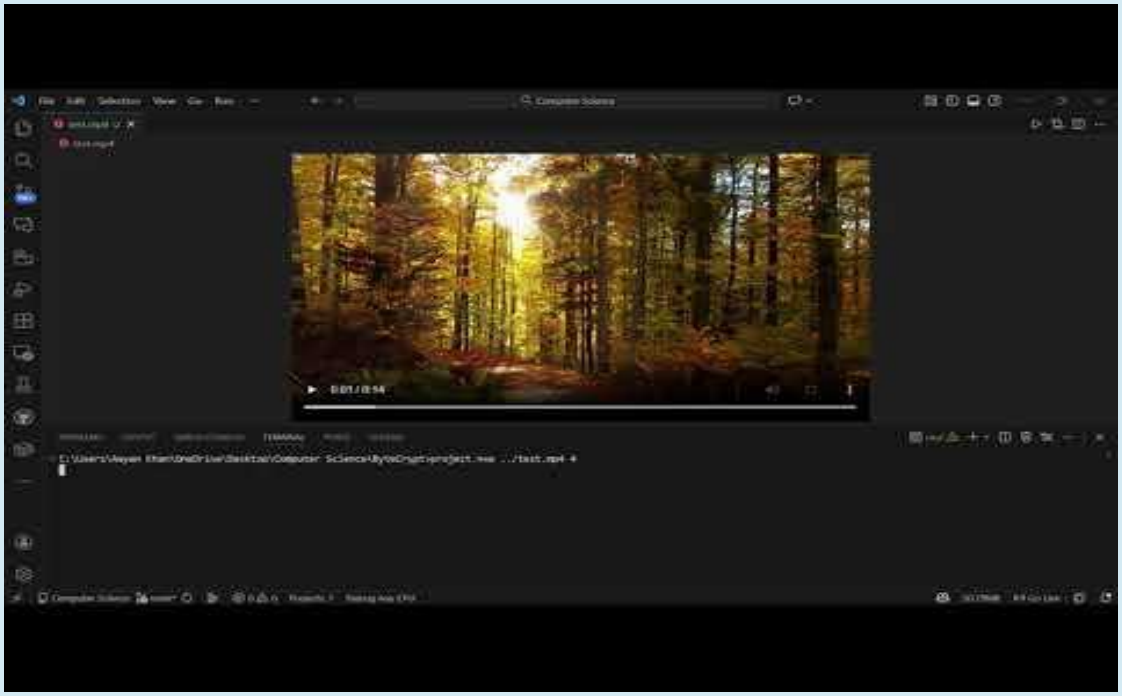
 # Invalid File Path Provided

 # Text File Encryption/Decryption (Link: https://youtu.be/a-M6XE3AdyQ)

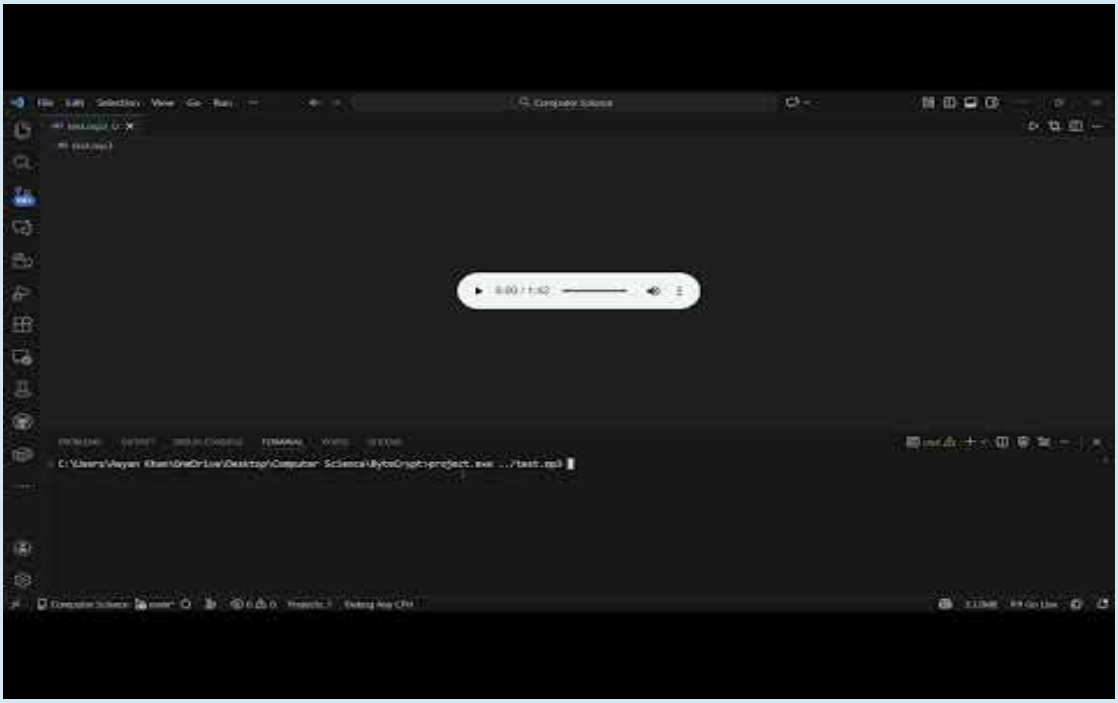
 # Image File Encryption/Decryption (Link: https://youtu.be/3yTuUDFJfWI)



 # Video File Encryption/Decryption (Link: https://youtu.be/bulEWPGu72M)



 # Audio file Encryption/Decryption (Link: https://youtu.be/6s-Cgjei9RI)



 # PDF File Encryption/Decryption (Link: https://youtu.be/u3RCV40ChoA)

Data Table

Concept Tested	Input	Predicted Output	Actual Output
Daniel Bernstein Rolling Algorithm $key = key * multiplier + (int)character;$	key = 5381, multiplier = 33 password = “Hello”	121	121
Invalid usage of program	project.exe	Usage: <file_path> <password>	Usage: <file_path> <password>
Invalid file path provided	project.exe invalid.txt word	Invalid File Path. ❌	Invalid File Path. ❌
Valid Program Usage	project.exe words.txt Hello	File has been secured. ✅	File has been secured. ✅

Analysis and Discussion

This project helps in understanding how XOR encryption can be used to transform the bits of any manipulable data into a scrambled format using a password-based key, thereby encrypting the file. Applying the same XOR operation again with the same key reverses the transformation, allowing the file to be decrypted.

Conclusion

Successfully compiled and executed the C project demonstrating XOR-based encryption and decryption on various file types using user-defined passwords with appropriate status messages.

Credit to Pixabay (https://pixabay.com/) for sample files for demonstration.