Student Project for CompE270 and CompE160

My student project for CompE270/160 is a basic encryption device running off a Raspberry Pi. It will ask the user to authenticate themselves through the FPGA. The FPGA has two outputs on pins 0P0 and 0P1 to signal to the Raspberry Pi that the user has either succeeded or failed in inputting the password. This will cover the CompeE160 section of the project.

Design and Overview

For my project design, I used the Raspberry Pi's wiringPi library to accept input from an outside source (in this case, the FPGA). As for encryption and decryption, I am using bitwise XOR functions to encrypt or decrypt the file, with a key. In this case, I set the key to be `FiShEyEpLaCeBo\0`. The encryption program will iterate each `char` and XOR the `char` with the corresponding key `char`. For cases in which the message to be encrypted is longer than my key, the key will wrap around with a modulo back to the beginning of the key. The message will then be written into a file named `encrypted.txt`. This file can be decrypted by running the decryption program (which is the same as the encryption program, with the exception that it opens a file and reads input from there instead of accepting a user input).

Code

Encryption (User Input + Authentication)

```
#include <stdio.h>
#include <string.h>
#include <wiringPi.h>

//Constants
#define KEYSIZE 15 //key WITH '\0'
#define SUCCESS 0
#define FAILURE 1

//Function prototypes
char encrypt(char plain, char key); //Ln. 45
void authenticate();                //Ln. 50

main() {
  char *message;
  char key[KEYSIZE] = "FiShEyEpLaCeBo";
  int i, length;
  FILE *enc = fopen("encrypted.txt", "w");

  //WiringPi setup
  wiringPiSetup();
  pinMode(SUCCESS, INPUT);
  pinMode(FAILURE, INPUT);

  //Starts authentication process
  printf("Please enter credentials on authentication device.\n");
  delay(10);
  authenticate();

  //Setting up message buffer.
  message = malloc(1000);
  printf("Type message to encrypt: ");
  fgets(message, 1000, stdin);
  length = strlen(message);

  //Converting the string to ciphertext
  for(i = 0; i < length; i++) message[i] = encrypt(message[i], key[i%KEYSIZE]);

  //Writing message into file
  for(i = 0; i < length; i++) fputc(message[i], enc);

  printf("Encryption finished.\n");
} //End main()

```

```
//returns encrypted char
char encrypt(char plain, char key) {
  return plain ^ key;
} //End encrypt()

//Authenticates the user, exits program if authentication fails
void authenticate() {
  //Wait for a success/failure signal from the FPGA
  while(1) {
    if (digitalRead(SUCCESS)) {
      printf("==========AUTHORIZED==========\n"); delay(500);
      printf("Encrypting/Decrypting message...\n"); delay(500);
      return 1;
    }
    if (digitalRead(FAILURE)) {
      printf("==========ACCESS DENIED=========\n"); delay(500);
      printf("Intruder disposal drones have been dispatched...\n"); delay(500);
      printf("Have a nice day.");
      exit(1);
    }
  }
} //End authenticate()
```

Decryption (Stream from File)

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

//Constants
#define KEYSIZE 15 //key WITH '\0'

//Function prototypes
void getMessage(char* message, FILE *input); //Ln. 33
char decrypt(char enc, char key);            //Ln. 42
int getSize(FILE *file);                     //Ln. 47

main() {
  char *message;
  char key[KEYSIZE] = "FiShEyEpLaCeBo";
  int i, length;
  FILE *enc = fopen("encrypted.txt", "r");
  FILE *plain = fopen("plain.txt", "w");

  //Setting up message buffer.
  message = malloc(1000);
  getMessage(message, enc);
  length = getSize(enc);

  //Converting the ciphertext to plaintext
  for(i = 0; i < length; i++) message[i] = decrypt(message[i], key[i%KEYSIZE]);

  //Writing message into file
  for(i = 0; i < length; i++) fputc(message[i], plain);

  printf("Decryption finished.\n");
} //End main()

//Copies the file into the character buffer
void getMessage(char* message, FILE *input) {
  int n, character;

  while((character = fgetc(input)) != EOF) {
    message[n++] = (char) character;
  }
} //End getMessage()
```

```
41
42  //returns decrypted char
43  char decrypt(char enc, char key) {
44    return enc ^ key;
45  } //End encrypt()
46
47  //returns the size of the file in bytes
48  int getSize(FILE *file) {
49    int n;
50
51    fseek(file, 0, SEEK_END);
52    n = ftell(file);
53    fseek(file, 0, SEEK_SET);
54
55    return n;
56  } //End getSize()
```