HARDWARE ENCRYPTED USB USING AES-256

By

Amir Abbas

Marvi Waheed

Kamran Anwar

Submitted to Department of Electrical Engineering, Military College of Signals National University of Sciences and Technology, Rawalpindi in partial fulfillment for the requirements of a B.E Degree in Electrical (Telecom) Engineering

June 2015

ABSTRACT

This paper provides a detailed documentation of the design and implementation of a hardware encrypted USB 2.0 device on an Arduino Mega 2560 board using 256 bit AES encryption/decryption algorithm. This USB is a plug and play device that is compatible with most new generation computers and operating systems. It provides the highest level of security to all the data stored and transferred through this device. The USB device includes a built in keypad which is used to enter the PIN code to access data which cannot be accessed otherwise. The data stored in the device is encrypted in the real time by the encryption algorithm stored within the device. The device is fully hardware encrypted and provides more efficient protection mechanism against malicious threats.

CERTIFICATE OF CORRECTNESS AND APPROVAL

It is hereby certified that the contents and form of the project report entitled ”Hardware Encrypted USB Using AES-256” submitted by 1) Amir Abbas 2) Marvi Waheed 3) Kamran Anwar have been found satisfactory as per the requirement of the B.E. Degree in Electrical (Telecom) Engineering.

Supervisor:

____

Col. Imran Rashid, PhD

Head of EE Department

Military College of Signals, NUST

Date: ___________________

DEDICATION

Allah, the Omnipotent

Faculty for its insightful help

And our parents for their support

ACKNOWLEDGEMENTS

This project would not have been accomplished without Allah Almighty's will. We humbly thank Him for His blessings and giving us the wisdom, knowledge and understanding, without which we would not have been able to complete this thesis research work.

Due extension of gratitude to our project supervisor, Col. Dr. Imran Rashid, without whose support and encouragement; it would not have been possible to complete this project.

We also thank our colleagues for helping us in developing the project and appreciate the people who have willingly helped us with their abilities.

Last but not the least; we are very thankful to our parents, who bore with us in times of difficulty and helped us overcoming obstacles. Without their consistent support and encouragement, we could not have accomplished our targets successfully.

TABLE OF CONTENTS

Page No.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AES:	Advanced Encryption Standard
FDE:	Full Disk Encryption
HDD:	Hard Disk Drive
LED:	Light Emitting Diode
SD:	Secure Digital
SPI:	Serial Peripheral Interface
USB:	Universal Serial Bus

CHAPTER 1

INTRODUCTION

A brief overview of development and implementation Hardware encrypted USB using AES 256 will be given in this chapter.

1.1 BACKGROUND OVERVIEW:

Nowadays, USB drives are frequently used today for data storage and transfer of data in offices and large organizations. Similarly, USB storage devices are also required for personal use. There is a high probability of data loss due to theft or loss of devices and security hack attacks in these devices. These effects are the main factors that make device highly inefficient for secure data transfer where integrity and security are of main concern. The importance of confidential data stored on hard disks cannot be ignored as data privacy is quite an issue in the modern world. Unfortunately, only few hard disk drives nowadays do include data security system to protect confidential data that may be critical for individuals and mandatory for business. There could be various security hazards, such as the pernicious data alteration, data leakage and lost hard-disk. Events may cause inestimable loss to some organizations such as military, governments and enterprises. Regarding the secured data importance, several risk protection schemes have been implemented. Those may vary in terms of:

- Strength of Security
- User friendliness
- Cost of implementation.

One of the major techniques used to achieve data security is encryption i.e. any data stored in the USB drive must be in encrypted form (software/hardware). Hardware encrypted USBs are safer and tend to be more cost effective.

1.2 PROJECT DESCRIPTION

The project aims at the design and implementation of a hardware encrypted USB device on an Arduino Mega 2560 board using 256 bit AES encryption algorithm. This USB is a plug and play device that provides the highest level of security provided by hardware encryption using the AES 256 encryption/decryption algorithm to all the data stored and transferred through this device. A numeric keypad is used to enter the code and the data can only be accessed if the code is correct. The data stored in the device is encrypted in the real time by the encryption algorithm stored within the device. The hardware encryption permits better protection against unauthorized access and malevolent threats in comparison to software encryption.

The device will be cost sufficient, hack immune and will provide secure data storage to keep confidential data safe, secure, portable and on the go. It basically involves the design of a hardware architecture, for a particular user, that will not only ensure the security of all the data present in the hard disk through hardware encryption of the data and secure login password access by an combined keypad but will also aim at overcoming the limitations in the processing power of the software based Full Disk Encryption systems. Development will be done using Arduino Mega 2560 kit which will be programmed in C language. The device at the user end will contain completely encrypted confidential data that shall be decrypted from the cipher text into its original form only upon providing the access password manually.

1.3 SALIENT FEATURES:

High capacity, compact USB storage drives enable workers to transport large amounts of classified company data anywhere. There are significant security hazards that could impact the organization if these drives are lost or stolen. Secure USB drives are the best way to stop the regeneration and insecure propagation of data against security breaches that have afflicted corporations and government agencies ever since unsecured flash drives became available.

For this reason we propose to target the following application areas to keep portable data safe, secure and on the go.

- **Keep Data Safe, Secure and Portable:**

All the information copied onto the Encrypted USB device will be encrypted and can only be read by authorized individuals. Built-in user access control and strong hardware data encryption keeps sensitive

- **Strong Hardware Encryption:**

All data on the USB device will be encrypted using AES-256 which is the strongest encryption algorithm available. Encryption keys are completely secure due to built-in hardware encryption and key generation. The key never leaves the USB drive and as a result, it cannot be obtained or copied.

- **Customized Solution:**

To access data on the USB users must authenticate themselves using a password via the number keys preventing unauthorized access to data.

- **High Performance and Compatibility:**

We target to provide a High Speed USB 2.0 Device that is compatible with any computer through the USB port.

- **Drive Reset Feature:**

The keypad is also used to perform other administrative actions, such as changing the PIN. After 10 failed login attempts, the drive resets erasing all data and PIN, but the AES 256-bit encrypting electronics remain intact.

- **Cost Effective Solution:**

Minimum computer and financial resources are utilized to provide a trustworthy solution to data leakage and confidentiality issues.

1.4 PROJECT SCOPE

A hardware encrypted USB device based on ARDUINO—AES system will be designed and implemented. Hardware will be programmed through C language. It is a plug and play device that encrypts 100% of your data in real-time and keeps your data safe even if the hard drive is removed from the enclosure.

1.5 OBJECTIVES:

The main goal of this project is to create a stand-alone device for providing secure communication technology for manual data transfer.

This project has the following highlighted objectives:

- 100% Hardware Platform
- Hardware Based Encryption/Decryption Methodology
- Hack Resistant

- Less Computer Utility Requirement
- Higher Security Safe Guard As Compared To Software Encryption
- Custom Solution (Limited User(s), Personalized Password Access)
- Easily Configurable
- Cost Effective Solution
- High Speed USB 2.0 Device
- High Compatibility with Any Computer through USB Flash Drive.

1.6 DELIVERABLES

This hardware encrypted USB device will provide secure storage and transfer of classified data with user friendly and portable on the go features. It is an excellent product for government, hospitals, insurance corporations, fiscal institutions, HR departments and management with sensitive data that needs the highest level of security and confidentiality.

1.7 FINALIZED APPROACH

The finalized approach shall be executed using Arduino Mega 2560 Kit. These are reconfigurable digital integrated circuits that in the past have proven to provide high performance and low cost for cryptographic applications. To achieve better speed, inbuilt parallelism is used in any algorithm put into operation. Also since the design is purely hardware based so it will relatively be immune to any hacking attacks. Once the password is entered, the data is transmitted from the PC to the Arduino board, encrypted at real-time and stored in the external memory interfaced with the kit.

Figure 1 below depicts the system model of our project:

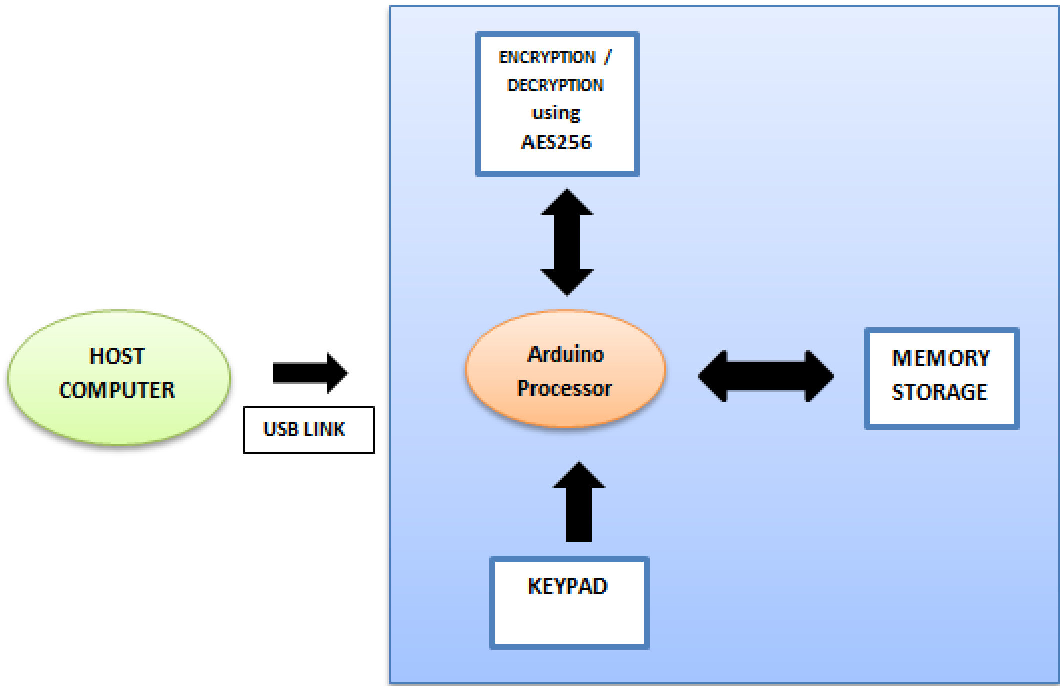

Figure 1: System Mode

Above diagram shows the modules of the project i.e.

1. Encryption/Decryption module
2. Flash memory module
3. Keypad interface module
4. USB to host computer link

Figure below shows the finalized approach of the project:

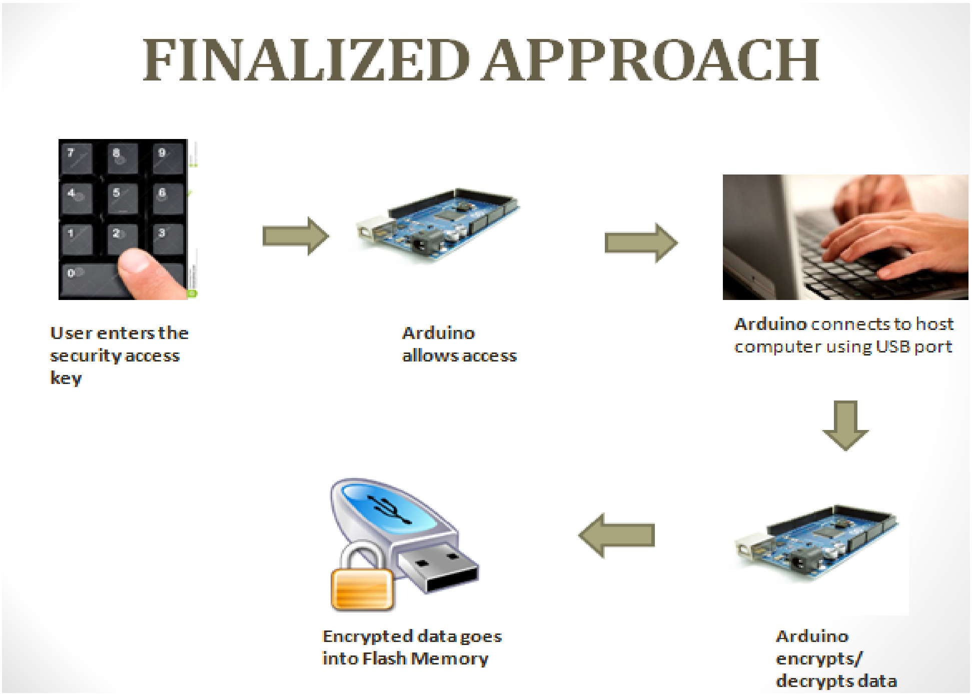

Figure 2: Finalized Approach of the Project

CHAPTER 2

LITERATURE REVIEW

Progress in information technology brings us the ease and competence together with new challenges on information security. Only by ensuring the safekeeping of information transmission, people can make better use of information services. Through the breakdown of the current situation and its existing problems, we understand the value of information security problem in the electronic document.

2.1 FULL DISK ENCRYPTION:

The significance of cryptography applied to security in electronic data communication has required a crucial relevance during the last few years. In cryptography, we deal with the protection of information from adverse individuals by changing it into an unfamiliar form while it is being saved and passed on. In a nutshell, data cryptography is the jumbling of the gist of data, such as text, audio, image, video and so forth to make the data incomprehensible, undetectable or unclear during communication or storage. This process of achieving data security is known as encryption. [1]

Several threat protection methods have been employed regarding the secured data importance.

BIOS and operating system passwords are usually utilized but these efforts often provide inadequate security. Amateur attackers can easily remove these security passwords and access private information. It is a bit difficult to crack or remove the hard drive protection password but the security level is still not strong enough.

Every bit of data is encrypted that goes through the disk either by hardware encryption or software encryption mechanisms using Full disk encryption (FDE or whole disk encryption. The FDE is appreciably stronger than the first two methods mentioned above. Cryptographic algorithm determines the security strength. Computer's CPU is used for encryption/decryption for software-based FDE method. This method has shown some disadvantages:

- A Trojan program can be easily used to scrutinize the encryption/decryption software.
- The instructions of the encryption/decryption are executed by the CPU. Obviously the processes consume more computer means.
- It is a very tough job to transfer the software used for encryption/decryption among different operating systems.

The advantage of hardware encryption is high speed while the advantage of software encryption is low cost. Less time is taken to encrypt, even when handing out large amounts of data. Hardware-based encryption is more secure, more concurring, and less time-consuming despite its higher cost. The highest level of data protection, hardware encryption will definitely fulfill the security requirements.

	Software Encryption	Hardware Encryption
Possibility of data leakage	Higher	Lower
Time needed for encryption	More	Less
Difficulty level of encryption & decryption	Lower	Higher
Compatibility with other software	Lower	Higher
Cost	Lower	Higher

Table 1: Software vs. Hardware Encryption

2.2 ADVANCE ENCRYPTION STANDARD:

On January 2, 1997 the National Institute of Standards and Technology (NIST) held a contest for a new encryption standard. The previous standard, DES, had been in use since November 23, 1976 and was no longer sufficient for security. Due to an increase in the computing power, the algorithm was no longer considered safe. In 1998, a specially made computer called the DES cracker was used to crack DES in less than three days. The contest went on for three years and NIST opted for an algorithm designed by two Belgian scientists, Joan Daemen and Vincent Rijmen. The name Rijndael was chosen for their algorithm. On November 26, 2001 the Federal Information Processing Standards Publication 197 declared a standardized form of the Rijndael algorithm as the new standard for encryption and named it as Advanced Encryption Standard. This standard is at present still the standard for encryption.

The AES Rijndael is a block cipher, which operates on different keys and block lengths: 128 bits, 192 bits, or 256 bits. The input to each round consists of a block of message called the state and the round key. It has to be noted that the round key changes in every round. The state can be represented as a rectangular array of bytes. This array has four rows; the number of columns is denoted by Nb and is equal to the block length divided by 32. The same could be applied to the cipher key. The number of columns of the cipher key is denoted by Nk and is equal to the key length divided by 32. The cipher consists of a number of rounds - that is denoted by Nr - which depends on both block and key lengths. Each round of Rijndael encryption function consists mainly of four different transformations: SubByte, ShiftRow, MixColumn and key addition. On the other hand, each round of

Rijndael decryption function consists mainly of four different transformations: InvSubByte, InvShiftRow, InvMixColumn, and key addition.[2]

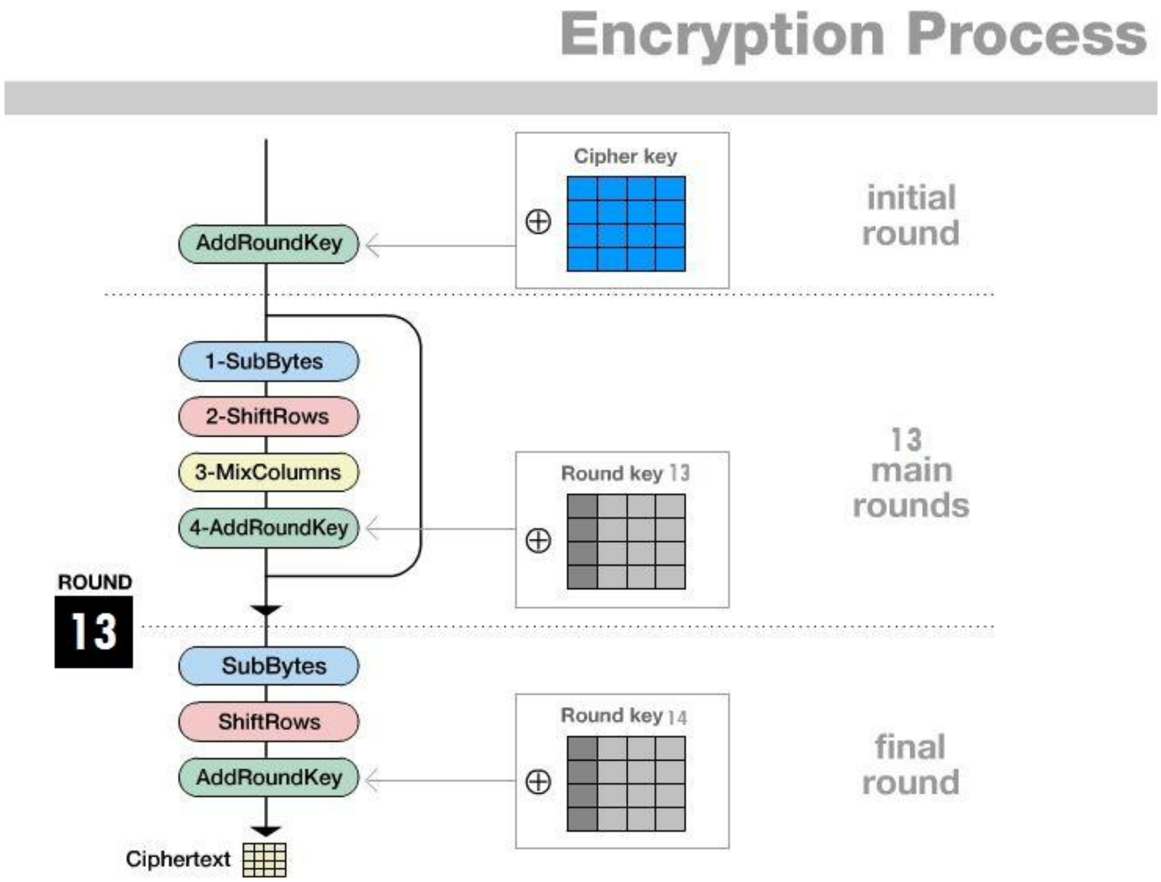

Figure 3: The Encryption Process

AES encryption/decryption algorithm consists of four steps:

2.2.1 SUB-BYTES TRANSFORMATION:

The Sub-Bytes transformation operates on each state byte independently in a non-linear byte substitution method. This transformation is done using a substitution table called S-box that is pre calculated. 256 numbers (from 0 to 255) are contained in the S-Box with corresponding resulting values.

The Sub-Bytes substitution is represented in the figure:

$$SB(State) = \begin{bmatrix} SB(d_{15}) & SB(d_{11}) & SB(d_7) & SB(d_3) \\ SB(d_{14}) & SB(d_{10}) & SB(d_6) & SB(d_2) \\ SB(d_{13}) & SB(d_9) & SB(d_5) & SB(d_1) \\ SB(d_{12}) & SB(d_8) & SB(d_4) & SB(d_0) \end{bmatrix}$$

2.2.2 SHIFT ROW TRANSFORMATION:

Through cyclic left shift over different offsets, the rows of the state are transformed in ShiftRows Transformation round. Row 0 doesn't shift; row 1 is shifted over one byte; row 2 and row 3 are shifted over two and three bytes respectively. The ShiftRows mechanism is shown in the figure below:

$$SR(SB\ (State)) = \begin{bmatrix} SB(d_{15}) & SB(d_{11}) & SB(d_7) & SB(d_3) \\ SB(d_{10}) & SB(d_6) & SB(d_2) & SB(d_{14}) \\ SB(d_5) & SB(d_1) & SB(d_{13}) & SB(d_9) \\ SB(d_0) & SB(d_{12}) & SB(d_8) & SB(d_4) \end{bmatrix}$$

2.2.3 MIX COLOUMNS TRANSFORMATION:

MixColoumns Transformation is a mixing operation which operates on the columns of the state, combining the four bytes in each column.

$$R = MC(SR(SB\ (State))) = \begin{bmatrix} '02' & '03' & '01' & '01' \\ '01' & '02' & '03' & '01' \\ '01' & '01' & '02' & '03' \\ '03' & '01' & '01' & '02' \end{bmatrix} \otimes \begin{bmatrix} SB(d_{15}) & SB(d_{11}) & SB(d_7) & SB(d_3) \\ SB(d_{10}) & SB(d_6) & SB(d_2) & SB(d_{14}) \\ SB(d_5) & SB(d_1) & SB(d_{13}) & SB(d_9) \\ SB(d_0) & SB(d_{12}) & SB(d_8) & SB(d_4) \end{bmatrix}$$

2.2.4 ADD ROUND KEY TRANSFORMATION:

AddRoundKey (AK) performs an addition (bitwise XOR) of the State with the RoundKey:

$$AK(R) = \begin{bmatrix} R_{15} & R_{11} & R_7 & R_3 \\ R_{14} & R_{10} & R_6 & R_2 \\ R_{13} & R_9 & R_5 & R_1 \\ R_{12} & R_8 & R_4 & R_0 \end{bmatrix} \oplus \begin{bmatrix} rk_{15} & rk_{11} & rk_7 & rk_3 \\ rk_{14} & rk_{10} & rk_6 & rk_2 \\ rk_{13} & rk_9 & rk_5 & rk_1 \\ rk_{12} & rk_8 & rk_4 & rk_0 \end{bmatrix}$$

The key expansions all utilize a few operations in Rijndael's Galois field. The operations are:

- An 8-bit circular rotate on a 32-bit word
- An Rcon operation that is simply 2 exponentiated in the Galois field.
- Rijndael's S-box operation
- A key schedule routine

The key expansion algorithm for AES 128 bit and AES 192 bit are same but AES 192 includes an extra application of the S-box.

Key length difference in terms of bits is the only difference between AES 128 bit and AES 256, in all the four steps before the data is fully gone through the encryption/decryption cycle and the key expansion algorithm. Repetitions of 10 and 14 cycles are followed orderly in AES 128 and AES 256 respectively. Hence the conversion of AES 128 to AES 256 occurs.

Counter (CTR) block mode operation also known as integer counter mode and segmented counter mode shall be used in our encryption algorithm. A block cipher is turned into a stream cipher in this mode. Generation of next keystream block by encrypting successive values of a "counter" is the key operation of this mode. A function which produces a sequence which is guaranteed not to repeat for a long time can be used as a counter. CTR mode has similar characteristics to Output feedback, but also allows a random access property during decryption.

CHAPTER 3

SYSTEM DESING AND DEVEPLOPMENT

3.1 TECHNICAL SPECIFICATIONS:

The Arduino Mega is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.

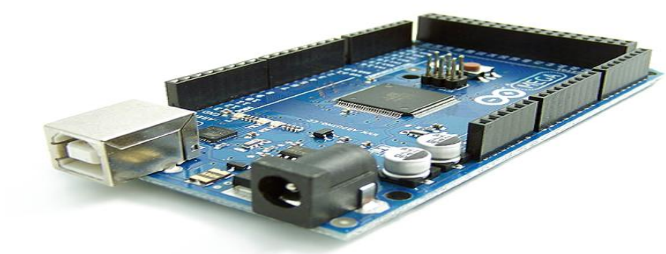

Figure4: Arduino Mega 2560

Other technical specifications of Arduino Mega 2560 include the following:

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54
PWM Digital I/O Pins	14
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB
Flash Memory for Bootloader	8 KB
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
Length	101.52 mm
Width	53.3 mm
Weight	37 g

Table 2: Specifications of Arduino Mega 2560

3.2 DESIGN REQUIREMENTS:

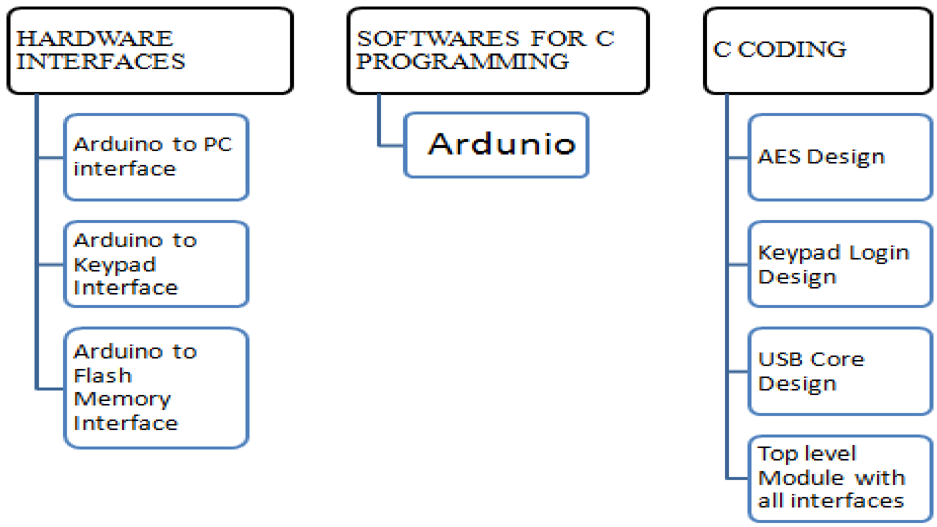

Figure 5: Design Requirements

The figure above shows the hardware interfaces, software used and the modules in the project

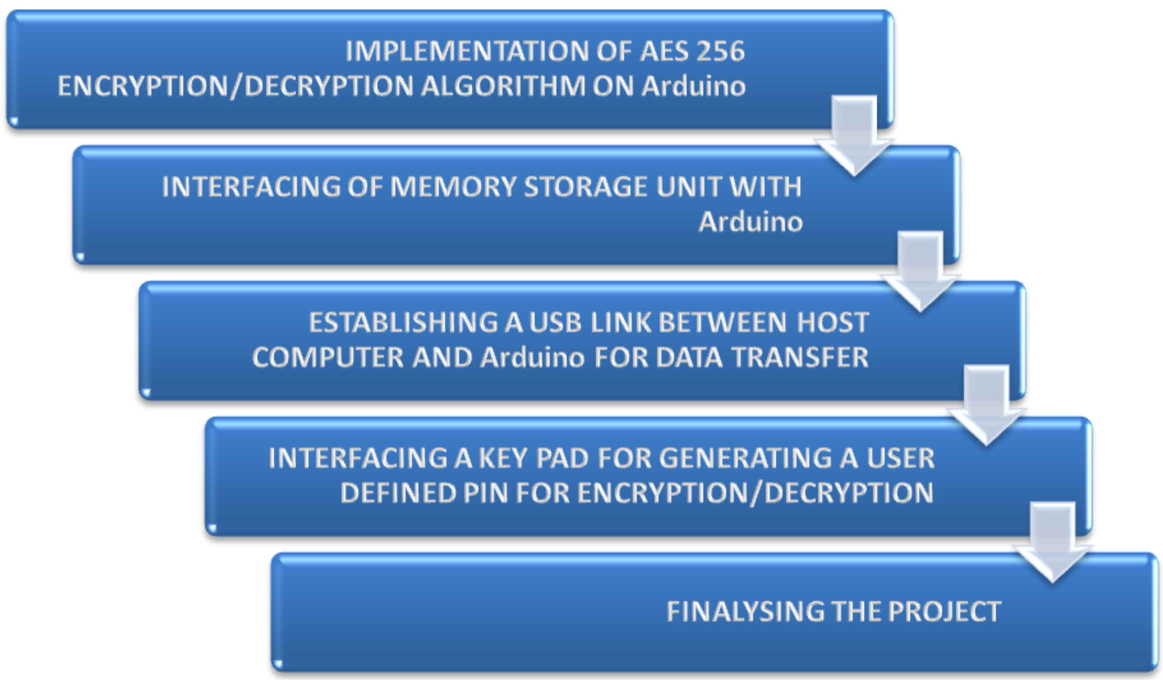

Figure 6: Implementation Procedure

The figure above shows our finalized approach towards the project and the implementation procedure that will be followed.

3.3 DESIGN SPECIFICATIONS:

- Hardware will be programmed using C Language.
- Real time data shall be used as an input from the user through a computer and stored in flash memory after the encryption/decryption process by the Arduino processor.
- Hardware implementation AES-256 bit with 14 cycles of repetitions of transformation rounds that convert the plaintext, into the final output, called the cipher text.
- User defined PIN required to access the flash memory in order to encrypt/decrypt data.

3.4 DETAILED DESIGN WITH JUSTIFICATIONS:

The figure below shows the detailed design of our project:

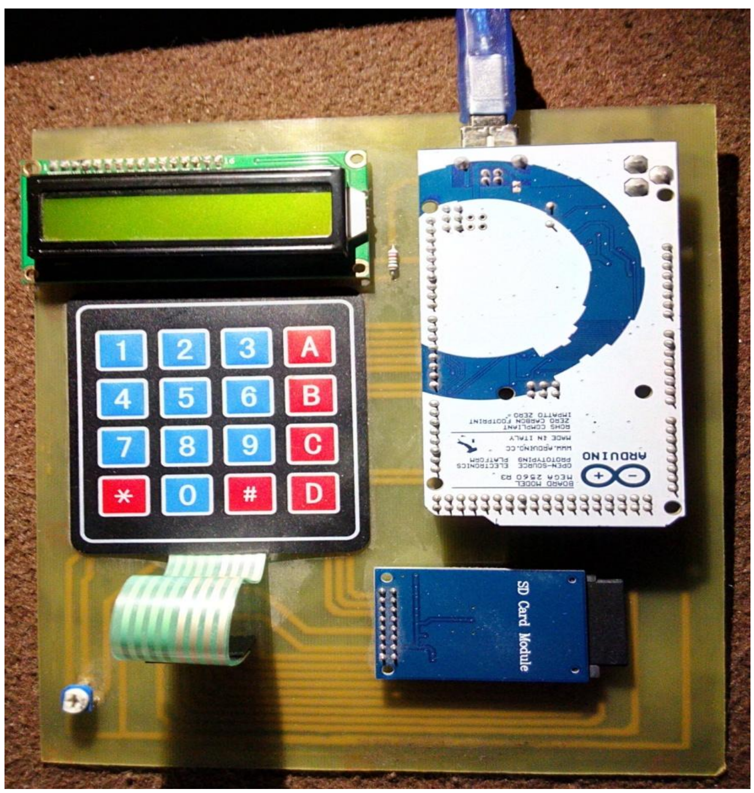

Figure 7: Detailed Design

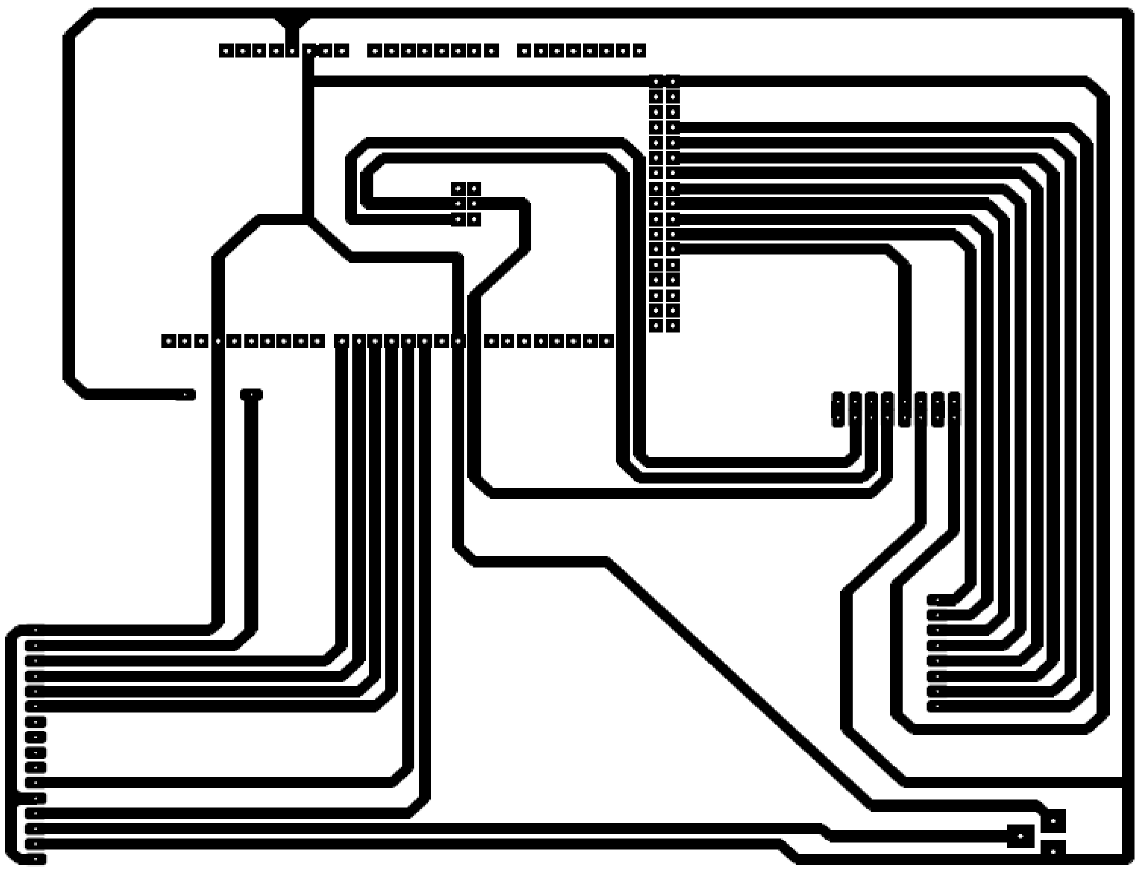

Figure 8: PCB Layout

Above diagrams show the implementation and PCB layout of the project. An Arduino Mega2560 board is interfaced to an SD Card reader module, a 16x2 LCD and a 4x4 keypad. A variable voltage regulator 7805 is used to control the voltage in the circuit. The details of the interfaces are given below:

3.4.1 ARDUINO TO SD CARD READER INTERFACE:

The SD Card Reader communicates with the Arduino using the SPI protocol. Serial Peripheral Interface (SPI) is a synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances. With an SPI connection there is always one master device (usually a microcontroller) which controls the peripheral devices. The pins of the SD card reader are connected to the following pins of Arduino:

- 53(SS) to CS
- 51 (MOSI) to D1
- 50(MISO) to D0
- 52(SCK) to CLK

3.4.2 ARDUINO TO LCD INTERFACE:

A 16x2 LCD is used for display purposes and consists of 16 pins. The pin configuration is as follows:

Pin 1	:	Vss	:	Function: GND.
Pin 2	:	Vdd	:	Function: +3V to +5V.
Pin 3	:	Vo	:	Function: Contrast/Brightness adjustment.
Pin 4	:	RS	:	Function: H/L Register Select Signal.
Pin 5	:	R/W	:	Function: Read/Write Signal.
Pin 6	:	EN	:	Function: H->L Enable Signal.
Pin 7	:	DB0	:	Function: H/L Data Bus Line.
Pin 8	:	DB1	:	Function: H/L Data Bus Line.
Pin 9	:	DB2	:	Function: H/L Data Bus Line.
Pin 10	:	DB3	:	Function: H/L Data Bus Line.
Pin 11	:	DB4	:	Function: H/L Data Bus Line.
Pin 12	:	DB5	:	Function: H/L Data Bus Line.
Pin 13	:	DB6	:	Function: H/L Data Bus Line.
Pin 14	:	DB7	:	Function: H/L Data Bus Line.
Pin 15	:	A/Vee	:	Function: 4.2V
Pin 16	:	K	:	Function: Power Supply for B/L

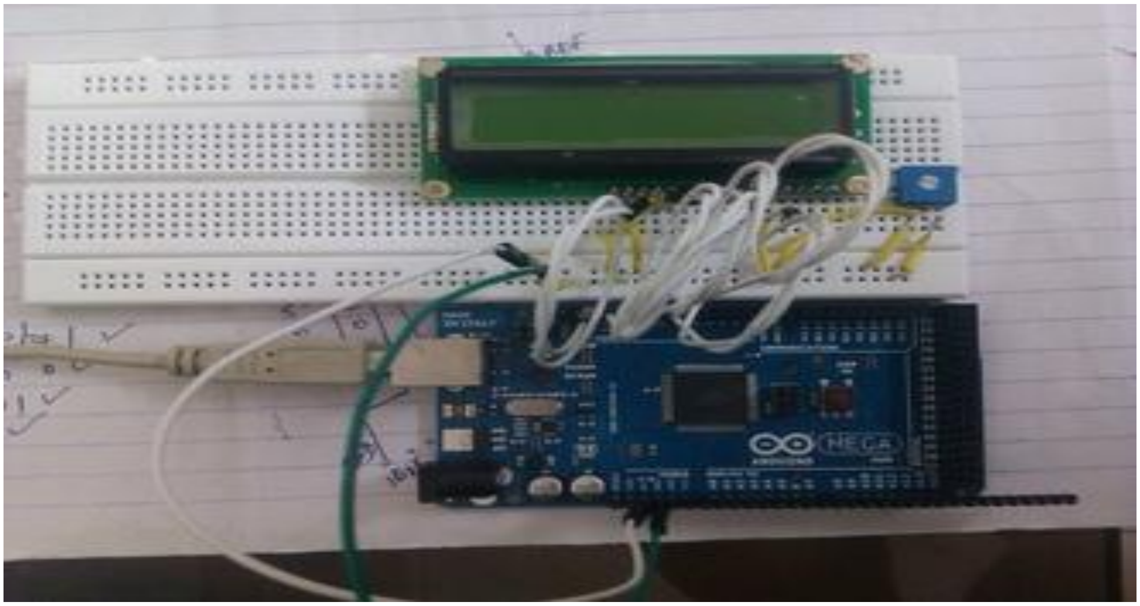

Figure 9: Arduino to LCD Interface

Data Pins DB4 to DB7 are used only and are connected to the pins 4, 5, 6 and 7 of the Arduino board. Pins 1,6 and 16 are grounded whereas Pin 2 and Pin 15 are connected to voltage supply of the Arduino board.

3.4.3 ARDUINO TO KEYPAD INTERFACE:

For getting access to the data inside the memory, a 4x4 keypad is used for entering the pin code. If the pin code is correct, Arduino allows access to the data stored. The rows of the keypad are connected to the pins 43, 45, 47 and 49 of the Arduino board. The columns of the keypad are connected to the pins 35, 37, 39 and 41 of the Arduino board.

CHAPTER 4

PROJECT ANALYSIS AND EVALUATION

Our aim of the project was to design a device that could be used as a safe data travelling mass storage device with all the data stored in the flash memory being in the encrypted form. The project was tested different types of text based files including notepad, MS excel and MS word files and the results of encrypted and decrypted data was correct.

Below is a screenshot of the application designed to access the contents of flash memory and encrypt/decrypt data:

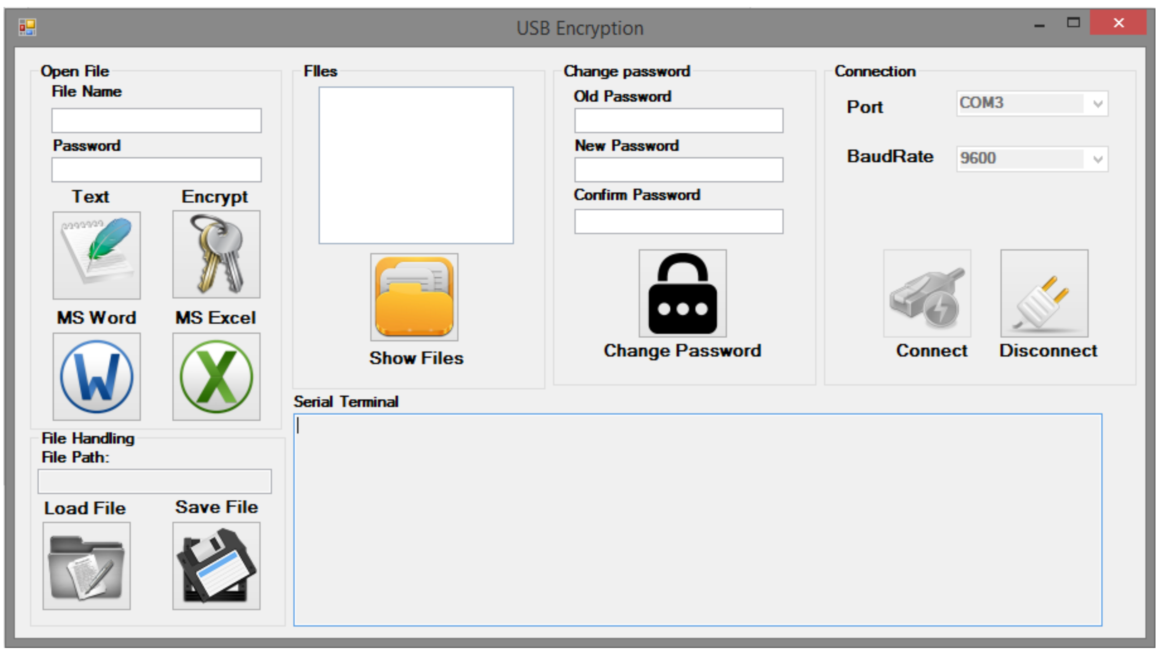

Figure 10: Application Evaluation

The functions of the buttons shown in the diagram are:

Show files:

Show the contents of flash memory.

Change password:

Change the user defined pin code which is used to access data.

Load file:

To save a file from computer to the flash memory.

Save file:

To create and save a file directly using the application

Encrypt:

To encrypt/decrypt data.

Text:

To open a file in notepad and to create a notepad file using the app.

MS Word:

To open a file in MS word and to create a Word file using the app.

MS Excel:

To open a file in MS Excel and to create an Excel file using the app.

CHAPTER 5

FUTURE WORK AND CONCLUSION

5.1 OVERVIEW:

The project aims at providing a secure way of using data travelers or mass storage devices using the highest standards of encryption i.e. AES 256. This project is a prototype of a hardware encrypted USB device for data travelling and mass storage using a keypad for user generated pin for more security. All the data stored in the device is in encrypted form and cannot be accessed without proper authentication that means the data is safe even in the case of theft or stolen devices.

5.2 OBJECTIVES ACHIEVED:

The objective of this project was to provide a safer way of travelling data using mass storage devise which is achieved using hardware encryption along with the AES 256 encryption standard. The project was thoroughly tested and it gives the required the required results satisfactorily. Objectives achieved regarding this project include complete implementation (software) of AES 256 algorithm and circuit design for interfacing Arduino with different modules of the project.

5.3 LIMITATIONS:

Due to hardware limitation of the Arduino Kit there are several limitations of the project regarding file handling of different types. As the processing power of the ATmega processor used in Ardunio is limited it is unable to handle to large size files that include media files and other application files.

5.4 FUTURE RECOMMENDATIONS:

Following are the recommendations made to be catered in the future projects:

- Overcoming of limitations regarding the handling of different file types to be encrypted. Media (Audio & Video) files to be encrypted.
- Incorporating different modes of encryption like Electronic codeback, Cipher-block chaining, Propagating cipher-block chaining, Cipher feedback and Output feedback other than the currently used CTR mode in the Cryptic application made.
- Handling of large sized files with faster and better speed and by allocating less computer resources.

CHAPTER 6

REFERENCES

6.1 LIST OF SIMILAR PROJECTS DONE AT MCS:

[1] Maj Babar Nawaz, Capt Muhammad Bilal, CaptKhurramRiaz, *"AES Encryptor/ Decryptor on FPGA"*, Military College of Signals, Aug 2013.

[2] Capt Muhammad Umair, Capt Muhammad Umar, Capt Ehtesham-ul-Haq, *"Secure SD CardReader"*, Military College of Signals, June 2014.

6.2 BIBLIOGRAPHY:

[1] Adnan Mohsin Abdulazeez, Farah Shleemon Khamo, "A Proposed Data Security Algorithm Based on Cipher Feedback Mode and its Simulink Implementation"from *Polytechnic University-Duhok City-Kurdistan Region of Iraq*

[2] Samir El Adib and Naoufal Raissouni, "AES Encryption Algorithm Hardware Implementation: Throughput and Area Comparison of 128, 192 and 256-bits Key" from National School for Applied Sciences of Tetuan, University Abdelmalek Essaadi.

[3] *Arduino-ArduinoBoardMega2560*. (n.d.) Retrieved from http://arduino.cc/en/Main/arduinoBoardMega2560

[4] How to Interface SD Card with Arduino: "*Arduino SD Card Project with Circuit Diagram".* (n.d.) Retrieved from http://www.engineersgarage.com/embedded/arduino/arduino/how-to-interface-sd-card-with-arduino-project-circuit

6.3 ONLINE HELP:

- http://arduino.cc/en/Main/ArduinoUSBHostShield
- http://www.circuitsathome.com/arduino_usb_host_shield_projects
- https://github.com/felis/USB_Host_Shield_2.0/tree/master/examples/testusbhostFAT
- https://github.com/felis/USB_Host_Shield_2.0/tree/master/
- http://www.atmel.com/Images/doc7631.pdf
- https://www.sparkfun.com/products/10155
- http://forum.arduino.cc/index.php?topic=88890.0
- https://github.com/qistoph/ArduinoAES256
- https://www.rivier.edu/journal/ROAJ-Fall-2010/J455-Selent-AES.pdf

CHAPTER 7

Appendix A
CODES

CODES FOR ARDUINO:

```
#include <Keypad.h>
#include<AESLib.h>
#include<SPI.h>
#include<SD.h>
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystallcd(2, 3, 4, 5, 6, 7);
File myfile, list;
boolean complete = false;
uint8_t key[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
30, 31};
uint8_tvari = 0, len = 0;
char encrypted[] = "";
char  enc[] = "";
intcnt = 0 , a = 0;
String check = "", val = "", final = "" ,encr = "", new_val = ""
, file = "", filename = "" , password = "", check_pass = "1234",
pass_send = "";
int i = 0 , inc = 0 , counter = 0;;
charvalu[] = "";
const byte ROWS = 4; //four rows
const byte COLS = 4; //three columns
char keys[ROWS][COLS] = {
                           {'1', '2', '3', 'A'},
                           {'4', '5', '6', 'B'},
                           {'7', '8', '9', 'C'},
                           {'*', '0', '#', 'D'}
                           };
//byte rowPins[ROWS] = {22, 24, 26, 28}; //connect to the row
pinouts of the keypad
//byte colPins[COLS] = {30, 32, 34 , 36}; //connect to the column
pinouts of the keypad
byterowPins[ROWS] = {49, 47, 45, 43}; //connect to the row
pinouts of the keypad
bytecolPins[COLS] = {41, 39, 37, 35};
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS,
COLS );

void setup()
{
      Serial.begin(9600);
      lcd.begin(16, 2);
```

```
      lcd.setCursor(0, 0);
      lcd.print("Please Wait.....");
      lcd.setCursor(0, 1);
      lcd.print("Initializing.");
      if (!SD.begin(33)) {
                 Serial.println("initialization failed!");
                 return;
                      }
Serial.println("initialization done.");
vari  = 50;
delay(2000);
}

void loop()
{
  //  if (Serial.available() )
  //  { //check = "";
  //
  //    while (Serial.available() )
  //    {
  //      char a = Serial.read();
  //      check += a;
  //    }
  //    //int le = check.length();
  //
  //  }

char key1 = keypad.getKey();
if (key1)
  {
if (inc == 0)
    {
lcd.clear();
    }
lcd.setCursor(0, 0);
lcd.print("Password:");
lcd.setCursor(9 + inc, 0);
lcd.print(key1);
delay(250);
lcd.setCursor(9 + inc, 0);
lcd.print("*");
inc = inc + 1;
if (key1 != '*' && key1 != '#' )
    {
pass_send += key1;
    }
if (key1 == '*')
{ if (check_pass == pass_send)
      {
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Correct Password");
```

```
vari = 50;
      }
else if (check_pass != pass_send)
      {
lcd.clear();
lcd.setCursor(3, 0);
lcd.print("Incorrect");
lcd.setCursor(2, 1);
lcd.print("Password...");
counter += 1;
        //Serial.println(counter);
if (counter == 10)
        {
format();
        }
      }
delay(1000);
Serial.print("^");
Serial.print(pass_send);
pass_send = "";
inc = 0;
vari = 50;
    }
else if (key1 == '#')
{ inc = 0;
pass_send = "";
Serial.print(pass_send);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Password:");
    }
  }
len = check.length();
if (complete == true)
{ complete = false;
if (len> 0)
{ int i = 0;
for (i = 0; i <len ; i++)
      {
if (check[i] == '!')
        {
vari = 1;
          //check = "";
        }
else if (check[i] == '$')
        {
vari = 2;
          //check[i] = ' ';
        }
else if (check[i] == '*')
        {
vari = 3;
```

```
        }
else if (check[i] == '%')
{ Serial.println(check);
vari = 4;
        }
else if (check[i] == '?')
        {
vari = 5;
Serial.print(check);

        }
      }
    }
  }
if (vari == 50)
  {
lcd.clear();
lcd.setCursor(4, 0);
lcd.print("Welcome..!");
vari = 100;
  }

if (vari == 1)
{ lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Showing Files!");
    //lcd.setCursor(2,1);
    //lcd.print("PassWord..");
delay(1000);
vari = 50;
    a = 0;
file = "";
check = "";
Serial.println("Files:");
list = SD.open("/");
printfiles(list, 0);
Serial.println("done!");
  }

else if (vari == 2)
{ lcd.clear();
lcd.setCursor(0 , 0);
lcd.print("DE-CRYPTING..!");
delay(1000);
encr = "";
vari = 50;

int i = 0 , j = 0;
for (i = 0 ; i <len - 2 ; i++)
    {
if (check[i] ==  '@')
      {
```

```
        j = 1;
      }
if (j == 0)
      {
filename += check[i];
      }
else if (j == 2)
      {
password += check[i];
      }
if (j == 1) {
        j = 2;
      }
    }
charfile_name[] = "";
    //Serial.println(filename);
    //Serial.println(password);
intlen_file = filename.length();
intpass_file = password.length();
check.toCharArray(file_name, len_file + 1);
check = "";
val = "";
    File open_file = SD.open(file_name);
    //Serial.print(":");Serial.print(filename);Serial.print(":");
if (password == check_pass)
{ //Serial.print("opening:");
pass_send = "";
int k = 0;
if (open_file)
      {//Serial.print("opened:");
while (open_file.available())
        {
char a = open_file.read();
val += a;
        }
open_file.close();
      }
final = "";
encr = "";
      k = 0 ;
      String aa = "";
int j = 0;
for (i = 0 ; i <val.length(); i++)
      {
final += val[i];
        k = k + 1;
if (k == 16 || i == val.length() - 1 )
        {

final.toCharArray(enc , 17);
          aes256_dec_single(key , enc);
```

```
encr += enc;

final = "";
          k = 0;
        }
if (i > 127)
        {
aa += val[i];
        }
      }
Serial.print(encr);
    }
else
{ pass_send = "";
if (open_file)
      {
        //Serial.print("Opened");
while (open_file.available())
{ char re = open_file.read();
Serial.write(re);
        }
      }
open_file.close();
 }
Serial.println("");
filename = "";
password = "";
  }
else if (vari == 3)
{ lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Password");
lcd.print(" Changed");
delay(1000);
Serial.print(check_pass);
vari = 50;
check = "";
  }
else if (vari == 4)
  {
vari = 50;
int le = check.length();
    String code = "";
    //Serial.println(le);
int i = 0;
for (i = 0 ; i < le - 2 ; i++)
    {
code += check[i];
    }
check_pass = "";
check_pass = code;
Serial.println("Changed");
```

```
check = "";
  }
else if (vari == 5)
{ lcd.clear();
lcd.setCursor(0, 0);
lcd.print("File");
lcd.print(" Saved.");
delay(1000);
encr =  "";
int le = check.length();
int i = 0 , k = 0;
    String data = "";
    String filename = "";
for ( i = 0 ; i <len - 2 ; i++)
    {
if (check[i]  == '>')
      {
        k = 1;
      }
if (k == 0)
      {
data += check[i];
      }
else if (k == 1)
      {
filename += check[i + 1];
      }
    }
Serial.println(filename);
val = data;
    k = 0 ;
for (i = 0 ; i <val.length(); i++)
    {
final += val[i];
      k = k + 1;
if (k == 16 || i == val.length() - 1 )
      {
final.toCharArray(enc , 17);
  aes256_enc_single(key , enc);
encr += enc;
final = "";
        k = 0;
      }
      //final.toCharArray(enc,17);
    }
val = "";
    //delay(1000);
charfinal_name[] = "";
intfil_len = filename.length();
filename.toCharArray(final_name, fil_len );
SD.remove(final_name);
    File my = SD.open(final_name, FILE_WRITE);
```

```
if (my)
    {
my.print(encr);
my.close();
Serial.print("done");
    }
check = "";
vari = 50;
  }
  //Serial.println("end");
  //delay(10);
}
voidprintfiles(File file , intnum)
{
while (true)
  {
    File valu = file.openNextFile();
delay(100);
if (! valu)
    {
break;
    }
for (uint8_t i = 0 ; i <num ; i++)
    {
Serial.print('\t');
    }
Serial.print(valu.name());
if (valu.isDirectory())
    {
Serial.println("/");
printfiles(valu , num + 1);
    }
else
    {
      //Serial.print("\t\t");
      //Serial.println(val.size(), DEC);
Serial.println("");
    }
valu.close();
  }
}
void format(void)
{ list = SD.open("/");
while (true)
  {
    File valu = list.openNextFile();
delay(100);
if (! valu)
    {
break;
    }
```

```
SD.remove(valu.name());
valu.close();
 }
}
voidserialEvent() {
while (Serial.available()) {
    // get the new byte:
charinChar = (char)Serial.read();
    // add it to the inputString:
check += inChar;
    // if the incoming character is a newline, set a flag
    // so the main loop can do something about it:
if (inChar == ')') {
complete = true;
    }
  }
}
```

CODES FOR VISUAL STUDIO APPLICATION:

```
Imports System.IO.Ports
Imports Microsoft.Office.Interop.Word


Public Class main

    Shared _continue As Boolean
    Shared _serialPort As SerialPort
    Dim WithEvents SP As New SerialPort
    Dim check As Integer = Nothing

    Dim open As Integer = Nothing
    Dim richtext As String = ""
    Dim val As Integer = 0
    Private Sub Form1_FormClosing(sender As Object, e As FormClosingEventArgs) Handles
Me.FormClosing
        If SP.IsOpen Then
            MsgBox("Disconnect before Closing")

        End If
    End Sub
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        check = 0
        GroupBox4.Enabled = False
        GroupBox3.Enabled = False
        GroupBox5.Enabled = False
        GroupBox2.Enabled = False
        btndisconnect.Enabled = False
        btnload.Enabled = False
        txtsend.Enabled = False
        Label8.Enabled = False

        Dim buadrate() As String = {"300", "1200", "2400", "4800", "9600", "14400", "19200",
"28800", "38400", "57600", "115200"}
        cmbbuad.Items.AddRange(buadrate)
        cmbbuad.SelectedIndex = 4

        Try
            GetSerialport()
            cmbport.SelectedIndex = 0
        Catch ex As Exception
            MsgBox("No Port Connected")
        End Try
    End Sub
    Private Sub GetSerialport()
        For Each spt As String In My.Computer.Ports.SerialPortNames
```

```
        cmbport.Items.Add(spt)

    Next
End Sub
Sub showstring(ByVal mystring As String)



    If mystring.Contains("^") = True Then
        check = 20
        txtpass.Text = ""
        richtext = ""
        Timer1.Enabled = True
    End If


    'txtIN.AppendText(mystring)
    richtext += mystring
    txtsend.Text = richtext




End Sub
Delegate Sub myMethodDelegate(ByVal [text] As String)
Dim mydelegate As New myMethodDelegate(AddressOf showstring)

Private Sub btnconnect_Click(sender As Object, e As EventArgs) Handles btnconnect.Click
    If (cmbport.Text.Length() > 0) Then


        GroupBox4.Enabled = True
        GroupBox3.Enabled = True
        GroupBox5.Enabled = True
        GroupBox2.Enabled = True
        Label8.Enabled = True
        txtsend.Enabled = True
        btnload.Enabled = True
        Try
            SP.PortName = cmbport.SelectedItem.ToString
            SP.BaudRate = cmbbuad.SelectedItem.ToString
            SP.Open()
            If (SP.IsOpen) Then
                btnconnect.Enabled = False
                cmbbuad.Enabled = False
                cmbport.Enabled = False
                btndisconnect.Enabled = True

            End If
```

```
        Catch ex As Exception
          SP.Close()
        End Try
      Else
        MsgBox("No Port Connected")
      End If

    End Sub



    Private Sub btndisconnect_Click(sender As Object, e As EventArgs) Handles
btndisconnect.Click
      GroupBox4.Enabled = False
      GroupBox3.Enabled = False
      GroupBox5.Enabled = False
      GroupBox2.Enabled = False
      txtsend.Enabled = False
      Label8.Enabled = False
      btnload.Enabled = False
      Try
        SP.Close()
        btnconnect.Enabled = True
        cmbbuad.Enabled = True
        cmbport.Enabled = True
        btndisconnect.Enabled = False
        Exit Sub

      Catch ex As Exception
        MsgBox("error while closing the prot!")
      End Try
    End Sub

    Private Sub SerialPort_DataReceived(ByVal sender As Object, ByVal e As
System.IO.Ports.SerialDataReceivedEventArgs) Handles SP.DataReceived
      Dim str As String = SP.ReadExisting()
      Invoke(mydelegate, str)


    End Sub


    Private Sub GroupBox2_Enter(sender As Object, e As EventArgs)

    End Sub

    Private Sub btfile_Click(sender As Object, e As EventArgs) Handles btfile.Click
      If SP.IsOpen() Then
```

```
        SP.WriteLine("!)")
        Timer1.Enabled = True
        richtext = ""
        txtIN.Text = ""
        check = 1


    End If
End Sub



Private Sub btnopen_Click(sender As Object, e As EventArgs) Handles btnopen.Click
    If SP.IsOpen Then
        Timer1.Enabled = True
        richtext = ""
        txtIN.Text = ""
        SP.Write("@")
        SP.Write(txtpass.Text)
        SP.Write("$)")
        SP.Write(TextBox1.Text)
        check = 2
        txtpass.Text = ""
    End If
End Sub



Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
    If check = 20 Then
        Dim num As Integer = 1
        For num = 1 To richtext.Length() - 1 Step 1
            txtpass.Text += richtext(num)
        Next
        check = 0
        richtext = ""
        Timer1.Enabled = False
    End If
    If check = 1 Then
        txtpass.Text = ""
        txtIN.Text = txtsend.Text
        check = 0
        Timer1.Enabled = False
        TextBox1.Text = ""
    End If
```

```
If check = 2 Then
    Timer1.Enabled = False
    If My.Computer.FileSystem.FileExists(TextBox1.Text) Then
        My.Computer.FileSystem.DeleteFile(TextBox1.Text)
    End If
    Dim file As System.IO.StreamWriter
    'richtext = ""
    file = My.Computer.FileSystem.OpenTextFileWriter(TextBox1.Text, True)
    file.Write(richtext)
    file.Close()
    txtpass.Text = ""
    'TextBox1.Text = ""
    check = 0

End If

If check = 3 Then
    txtsend.Text = richtext
    Timer1.Enabled = False
    If (txtoldpass.Text = richtext) Then

        If txtcnfirmpass.Text = txtpassnew.Text Then
            If SP.IsOpen Then
                txtpass.Text = ""
                SP.Write(txtcnfirmpass.Text)
                SP.Write("%)")
                txtcnfirmpass.Text = ""
                txtpassnew.Text = ""
                txtoldpass.Text = ""
                TextBox1.Text = ""
                check = 0
            End If
        Else
            MsgBox("Password Doesn't Match!", MsgBoxStyle.Exclamation, "Error")
        End If
    Else
        MsgBox("Old Password is Incorrect!", MsgBoxStyle.Exclamation, "Error")
    End If


End If

If check = 4 Then
    check = 0
    Timer1.Enabled = False
    Dim txt As String = ""
    If SP.IsOpen Then
        Dim valu As Integer = txtload.Text.LastIndexOf("\")
```

```
            Dim str As String = txtload.Text.Substring(valu + 1)



            If System.IO.File.Exists(txtload.Text) = True Then

                Dim read As New System.IO.StreamReader(txtload.Text)
                Do While read.Peek() <> -1
                    txt = txt & read.ReadLine

                    'txtsend.Text = txtsend.Text & read.ReadLine & vbNewLine
                    'SP.Write(txtsend.Text)
                Loop
                TextBox1.Text = ""
                txtsend.Text = txt
                txtpass.Text = ""
                SP.Write(txt)
                SP.Write(">")
                SP.Write(str)
                SP.Write("?)")

            Else
                MsgBox("File opening Error", MsgBoxStyle.Exclamation, "Error")
            End If
        End If
    End If

End Sub




Private Sub btnCreate_Click(sender As Object, e As EventArgs) Handles btnCreate.Click
    Process.Start("notepad", TextBox1.Text)
    'txtpass.Enabled = True
End Sub

Private Sub btnpassword_Click(sender As Object, e As EventArgs) Handles btnpassword.Click
    check = 3
    richtext = ""
    If SP.IsOpen Then
        SP.Write("*)")
        Timer1.Enabled = True
    End If
End Sub

Private Sub btnload_Click(sender As Object, e As EventArgs) Handles btnload.Click
    OpenFileDialog1.ShowDialog()
```

```
        txtload.Text = OpenFileDialog1.FileName
        Dim val As Integer = txtload.Text.IndexOf(".txt")
        '// If (val > 0) Then
        check = 4
        Timer1.Enabled = True


        '//Else
        'MsgBox("Please select a Text file!", MsgBoxStyle.Exclamation, "Error")
        'End If

    End Sub

    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
        Process.Start("winword", TextBox1.Text)
    End Sub

    Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
        Process.Start("excel", TextBox1.Text)

    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        If (TextBox1.Text.Length() > 0) Then


            txtload.Text = ""
            SaveFileDialog1.ShowDialog()
            txtload.Text = SaveFileDialog1.FileName()
            FileCopy(TextBox1.Text, txtload.Text)
            txtload.Text = ""
        Else
            MsgBox("Please Select a File to Copy",MsgBoxStyle.Question,"Warning")
        End If
    End Sub


End Class
```