COE 485: Senior Design Project

Final Report.

3WSDS
3-Way Secure Data Splitting

Supervisor: Dr. Talal Alkharobi

Team Members:

- Abdul-Mohsin Al-Faraj(201081340)
- Hamed Al-Mehdhar (200925210)

Table of Contents

1 Introduction:

Nowadays all governments, national and international security departments of countries have secrets. Not only that, citizen individuals and groups have their own secrets too. These secrets vary in terms of cruciality, some are very important and critical and some are not. All of these secrets need to be saved securely where they can't be exposed. Encrypting secrets will secure them where there will be a key, if known then the secret can be known too. However, there are secrets if they are to be revealed, different pieces of information are needed to be known from different persons. Where every person provides his part of the key, or needed information, so that the secret can be revealed.

Whether it is simple cryptography or some secret sharing scheme, any device implementing any of these methods need to provide full security. Because if not, some crucial secrets can be known may cause problems on different scales: individual, national, and international. Not only that, such devices are not really common and can't be found in any computer store. They are mostly used by governments and specially developed for governments too. Very few of these secret protecting devices are available for normal people.

2 Problem Statement:

This project is supposed to solve part of the problem of the lack of availability of secret protecting devices in the free market. As it has been noted before, similar devices are mostly exclusively made for governments agencies and not for general people. More specifically this project will solve the problem of the lack of the availability of a secret sharing device in the free market. Such devices are important to keep secrets protected. So providing such a device will help in protecting secrets.

This project has no direct impact on the society. It may however have a positive impact where it will help in enhancing the protection of people's secrets. However it also may result in a negative impact which is that if the device is not well developed it may not protect secrets properly, hence, it will result in revealing secrets which may have negative impacts on the global and local society, safety, and environment.

3 Project Specifications

We will first list the preliminary user requirements and technical specifications which we had on our progress report. Then we will list the adjusted final user requirements and technical specifications of our design:

3.1 User requirements:

1- Device connects to PC through USB port.
2- 3 USB ports to connect the flash drives.
3- The device appears on the PC as a single flash drive.
4- When a file is transferred to the device, 3 different secret shares are constructed and distributed among 3 connected flash drives (following a 2 out of 3 secret share scheme).
5- Secret shares will automatically be taken from each drive and the original file will be constructed and viewed in the PC.
6- Only the secret shared files are viewed by the PC when the device is connected.

3.2 Technical specifications:

1- The device connects to the PC (USB host) and act as a USB device using USB.
2- 3 different USB ports which are used to connect the flash drives and the device should be the USB host of these memory sticks.
3- Programming the device such that it will interact with the PC as if it is a single flash drive.
4- Viewing the size of the flash drive seen by the PC as the minimum available size in the 3different flash drives.
5- Detecting the availability of secret shares in the connected flash drives and constructing the original file using the embedded microcontroller.
6- When a file is copied to the device, 3 different secret shares are created and copied to the attached flash drives.
7- When the pc asks for the available files in the device, the device will respond with information about the secret shared files only.

Now we list the adjusted final user requirements and technical specifications:

3.3 Adjusted user requirements:

1- Device is powered up by connecting through USB port to a PC or by having an internal battery.
2- 3 USB ports to connect the flash drives, where share builds will be distributed to and reconstructed from.
3- Fourth USB port to connect the flash drive containing the file to construct shares for, or to write a reconstructed file back to.
4- A button to perform partitioning of an original file to 3 secret shares when pressed.
5- A button to perform reconstruction of an original file using 2 secret shares when pressed.
6- An indicator which shows the start, progression, and end of a function when its button is pressed.

3.4 Adjusted Technical specifications:

1- Device is powered up by connecting through USB port to a PC or by having an internal battery.
2- 3 USB ports to connect the flash drives, where share builds will be distributed to and reconstructed from.
3- Fourth USB port to connect the flash drive containing the file to construct shares for, or to write a reconstructed file back to.
4- A partitioning button is pressed to partition the original file in main flash drive to 3 different secret shares which are distributed to the other three flash drives (following a 2 out of 3 secret share scheme).
5- A reconstruction button is pressed to reconstruct the original file in main flash drive by reading 2 out of 3 of the secret shares available in the other three connected flash drives.
6- When any of the buttons is pressed, an LED is lighted up to indicate ongoing function, which becomes off when the function ends.

These adjustment were made because we changed our design idea due to reasons which will be shown and discussed later in the report. Following are our project usage guidelines, which will explain how to correctly use our device:

3.5 Project usage guidelines:

-The flash drive containing the file to build the secret share for or to reconstruct the original file to, should be connected to port 0.
-The flash drive connected to port 0 must contain a file name.txt containing the name of the file to be partitioned(only for partitioning function). After constructing the 3 different secret share files which will be named WSDS.txt and written to the flash drives connected to ports 1,2 and 3. The main file will be deleted.
-the other 3 flash drives should be connected to ports2,3 and 4.

-after doing the above steps, any of the 2 functions buttons can be pressed were an LED will turn ON until the completion of that function and it will go off after that.
-now flash drives can be removed. They CANT be removed while the LED is on.

4 Teamwork

Before we start talking about the teamwork we had in our project, we will first show the table of our tasks, which shows all of our finished tasks and their owners:

Table 1: Task table.

Task ID	Owner	Description
1	Both	Literature Survey about USB in general.
2	Both	Literature Survey about USB hubs and try to relate it to our project.
3	Both	Research to decide on a design whether using microcontrollers or using hardware (FPGA).
4	Both	Research to be able to communicate with flash drives through USB ports using an Arduino microcontroller and USB host shields.
5	Both	Research to be able to communicate with the PC (USB Host) using an Arduino (USB Device) and make the Arduino act as a flash drive by communicating with the PC accordingly.
6	Both	Deciding on and buying our project parts.
7	Both	Writing pseudo code for our design.
8	Both	riting different c code's for the Arduino to test the USB host shields by simple reading, writing, and copying of files.
9	Both	Writing different c code for the Arduino to implement our design using the pseudo code.
10	Both	testing and debugging of our code.

Looking at the above table we can notice that all of our tasks where the responsibility of both of us(the 2 members of the project group). In the first 5 tasks where literature surveys and research were conducted, each one of us does his research on the required subject, and we share our findings with each other and make decisions accordingly. We have made decisions in tasks 3, 4 and 5 based on our extensive research on each subject which will be shown in decisions section later in the report.

As for the tasks 7, 8, 9 and 10, we have done each of those together to insure correctness and consistency of our code. In each task, we test and modify until we get to what we want. However we did that together, by trying to debug and solve any problems we faced by sharing our ideas of how to solve such problems and what may be the cause of such problems. With such collective and teamwork thinking we solved and debugged a lot of small and big faced problems. Where if such tasks were done individually, they would have required more time.

5 Engineering Design

5.1 Design Decisions

Throughout the 15 weeks of work to design our project we have taken several design decisions. Design decisions were mainly taken in the first research tasks were after we research a subject we decide accordingly. In the following paragraph we will talk about the design decisions we have taken.

The first decision was at the start when we were thinking how to design the required device. We knew that we needed to implement a 3-Way Secure Data Splitting device where we have to connect to 4 USB ports and do some secret sharing processing. So we had more than one approach to design such a device. The first was to design the device logic using pure hardware components, we couldn't go with such solution since we didn't have enough knowledge in the field nor the required tools to go with that design method. The second design option was to use an FPGA to handle the logic of the design and find an interface to connect the USB to the FPGA, however our knowledge is very limited so we didn't feel comfortable going with. The third design option which was the one we have choose, is to have a microcontroller to do the logic of design and have some kind of interface to connect between USB ports and the microcontroller. We have chosen the third design because we had enough knowledge about microcontrollers and we could visualize that the device could be done with such an approach.

The second decision was after we chose the microcontroller's approach. We had to choose between different microcontrollers. So we had a lot of options. However in the end we chose to go with an Arduino since we are more experienced with it than with other microcontrollers. In addition, Arduino is an open source platform and is very easy to use and deal with. We said that we didn't want to waste our time learning other stuff and an Arduino would do the job so we went with this design decision. We then chose the Arduino Uno as our Arduino microcontroller and such decision will be explained later in the report.

The third decision was to choose a USB host board to serve as an interface between a USB device and the Arduino. We have found several USB host shields, however, 2 were most appropriate. The first was Arduino USB host shield. Where it is inserted over the Arduino as a shield and communicated with using a library. It was not very straight forward, so we continued searching for alternatives. Until we have found the second USB host board with preinstalled flash drive software manufactured by hobbytronics as shown in figure 4. These boards were ready made to communicate with flash drives, which we needed in our design. So they were the perfect fit to our design, hence, we chose them. So by this point we have the components needed to communicate with flash drives using an Arduino.

The fourth decision we had to make was after we have researched for a solution to help us communicate with PC's using the Arduino and make the Arduino act as a flash drive. At that time the idea of the design was the device will be connected to a PC which will show a flash drive, and whenever a file is copied to the flash drive, automatically a secret share will constructed and distributed among the 3 flash drives connected to the Arduino on the other side. Also, when the PC views the flash drive it should display the secret files from connected flash drives. After our extensive research about how to implement such functionality we couldn't find any solution which makes us able to do such a thing with a microcontroller. So we had to make a choice to change the design idea. We changed the design such that instead of connecting the

main USB port of the device to PC we will connect it to a fourth flash drive. Where the different functions of our device are going be triggered by different buttons. This eased the project but it was the only available alternative to go with, since we have taken a lot of design decisions to get this far.

After the fourth design decision we had a full scope our design and the needed components. So we purchased what we needed and started working on connecting the components and writing the Arduino code which will implement our design logic.

5.2 Architecture

5.2.1 Sub-function identification:

The main function of our project design is to implement (2 out of 3) secret sharing scheme for a file on a group of flash drives. A file secret share will be constructed for each byte (ASCII character) following a secret sharing scheme function which depends on the slope of a line and its intersection with the y axis (y=mx+s). This linear function is used to partition and reconstruct a group of bytes (a file). Since the function will be applied to ASCII characters the equation Y = (MX +S)%256 will be used. There are two main sub functions of secret sharing implemented by our design which will be discussed in the following paragraphs.

5.2.1.1 Partitioning Sub Function:

This sub function is responsible for constructing and distributing 3 different shares of the main file. Where a file is read line by line from flash drive connected to port 0, and 3 different secret shares are constructed for that line and distributed among 3 flash drives connected to ports 1,2 and 3.
Each line's secret share will be computed by calculating the different share of each ASCII character of the line. Such computation will follow the described linear function above. Where the character read from Flash drive connected to port 0 will be S. The slope m will be a random number and there will be 3 different values of X (X1, X2, X3). By substituting S , X1, X2, and X3 and m in equations Y1=(M*X1+s)%256, Y2=(M*X1+S)%256 and Y3=(M*X3+S)%256, we will get characters Y1 Y2 and Y3 which will be written to each of the corresponding flash drives connected to ports 1, 2, and 3 respectively.

The previous operation will be done for all characters in a read line and for lines of a file, hence 3 secret shares for the whole file are computed and written to 3 flash drives. The last thing to do is to delete the main file and the partitioning sub function is completed.

5.2.1.2 Reconstruction Sub Function:

This sub function is responsible for the reconstruction of the original file using 2 out of 3 of the distributed secret shares. Where the 2 different secret share files are read line by line from the flash drives connected to ports 1 and 2. And each line is read character by character. We will read 2 lines at a time, one from flash drive connected to port 1 and the other from the flash drive connected to port 2. And then read 2 characters one from each line. These read characters will be Y1 and Y2 respectively. Since X1, X2 and X3 are constants, we will use the available information to calculate the slope (assuming we will use X1 and X2), so the slope will be M=(Y2-Y1)/(X2-X1). Now we have M, so we can use the inverse of the above equation which is

S=(Y1-M*X1)%256 (if S is negative add 256, else don't do anything) and such computation will give us S. Then an original character of the original file will be known and written to the flash drive in USB port 0.

Continuing the same operation for all characters in a line and all lines in a file will result in writing the original file in the flash drive connected to port 0. This triggers the end of this sub function and the secret share files will be deleted. Note that the original file name will be saved in the first line of the secret share file along with the constant number X1, X2 or X3 which will be used to generate the corresponding secret share, and will be read in the start of this sub function to act accordingly.

5.2.2 System architecture, interfaces and components:

Now we will talk about the architecture of our design. Figure 1 shows our system architecture. We can see from the figure the Arduno uno is in the middle, and connected to 3 USB host shields from the right (USB ports 1,2 and 3) and connected to 1 USB host shield from the left (USB port 0). And we have 2 different buttons connected to the Arduino. Before we talk about how these components are connected and interfaced, we will identify each component and explain its functionality.

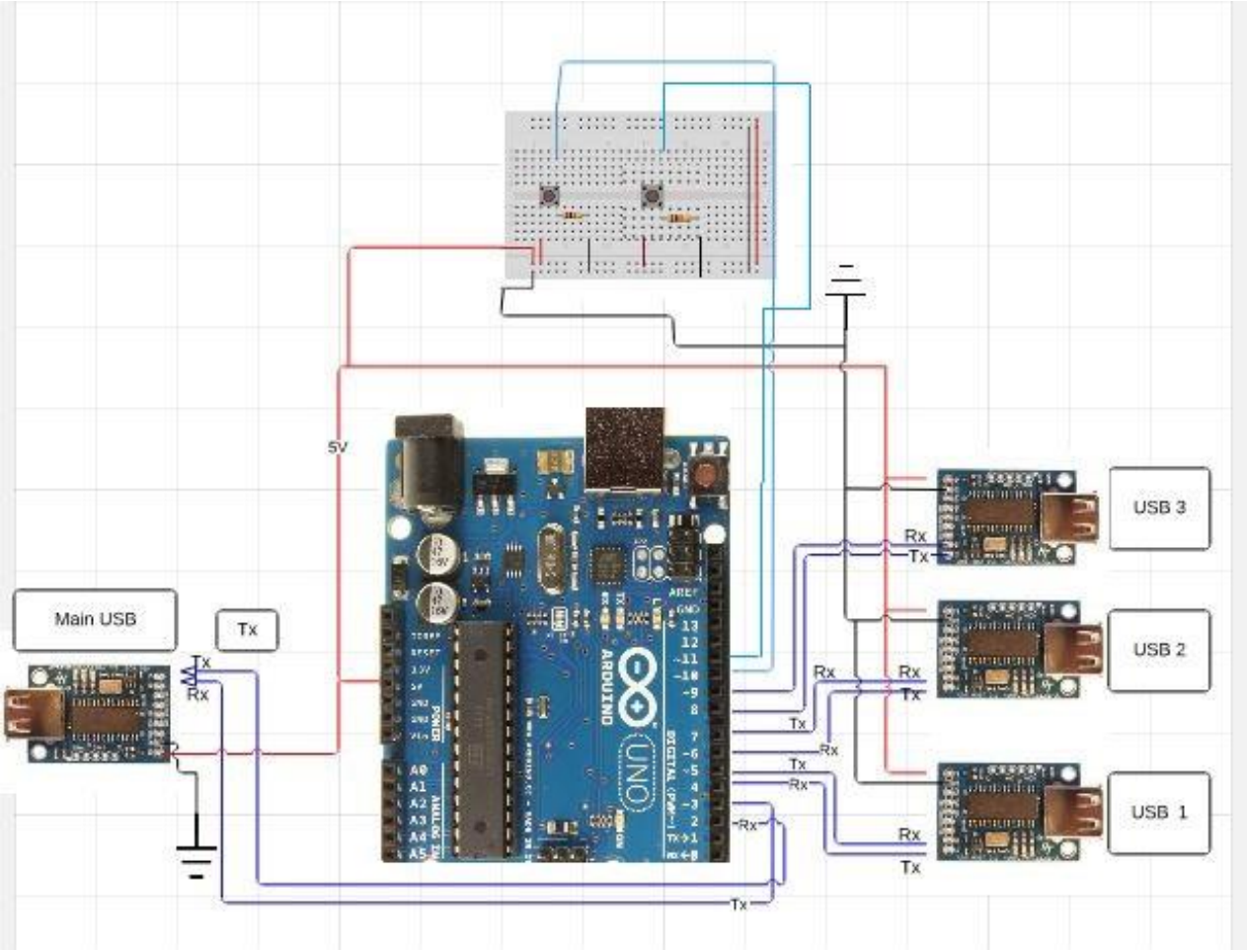

Figure 1; System Architecture.

5.2.2.1 System Components:

This sub section will show our design components and explain each component's function. Figure 2 shows all of our design components and how they are interconnected. From figure 2, we can see that we have the components: Arduino Uno, USB Host Shields, Buttons, secret partitioning software code, secret reconstruction software code, and SoftwareSerial Arduino library.

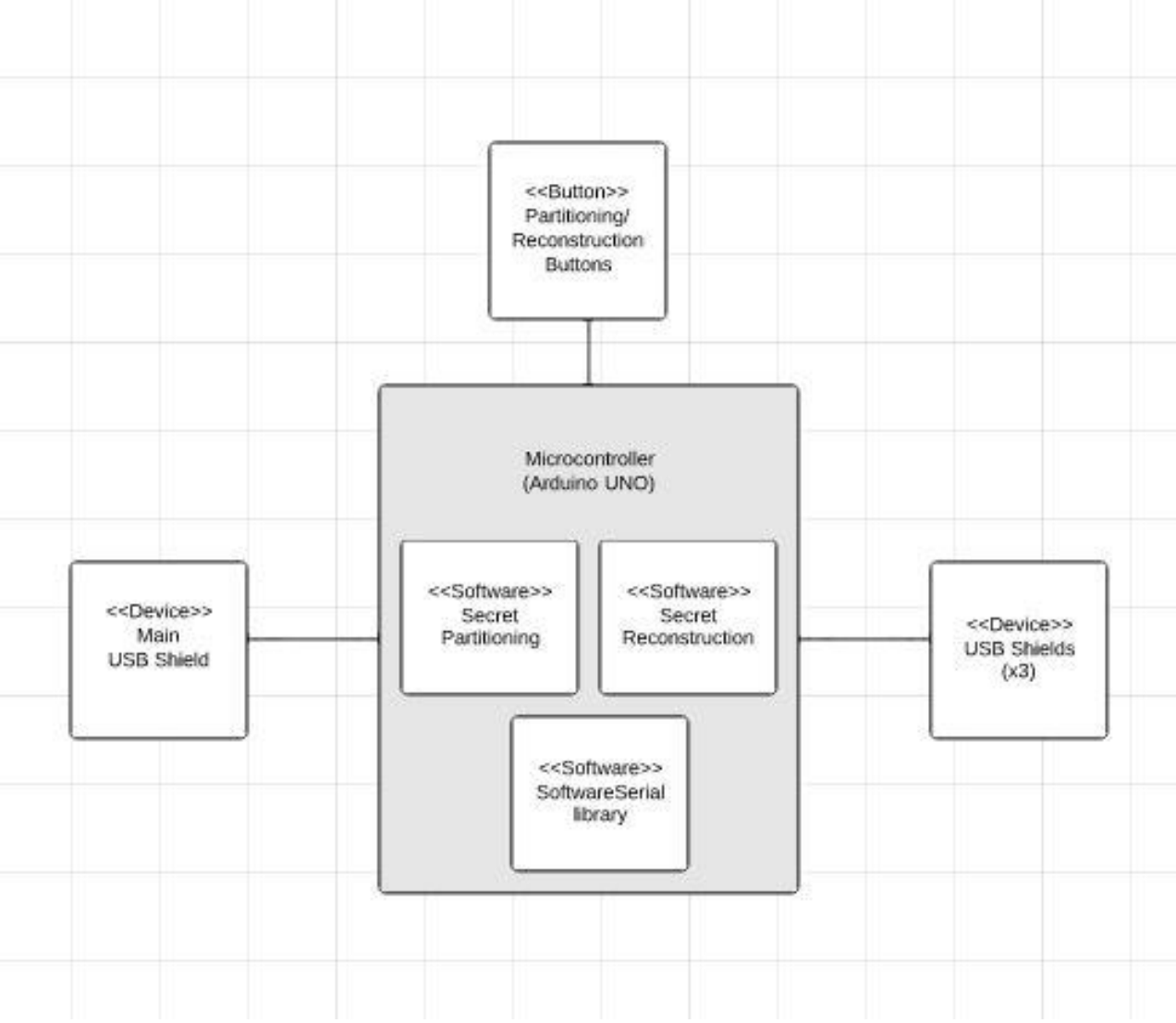

Figure 2; Component Diagram

In the following paragraphs we will distinguish hardware and software components and explain each component's function:

Hardware components:

Arudino Uno:

We have chosen Arduino Uno to be our microcontroller for our design. The Arduino role is to do the processing part of the design. Where its embedded code will be compiled and, hence, do the logic of our design. It is also responsible to power up USB host shields and sends commands to them to perform corresponding actions. It processes the logic of our design by communicating with USB shields and listening to button presses. It compiles our project code, which is the heart of the design.

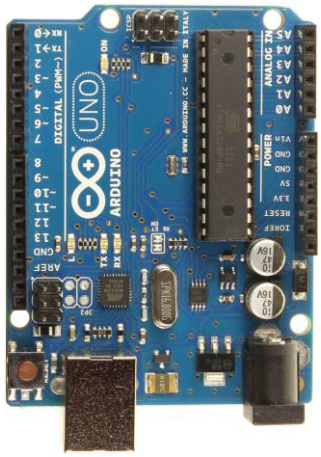

Figure 3: Arduino UNO

USB Host Shields:

These USB host shields are responsible for the communication with the USB flash drives. They come with preinstalled flash drive software. They are manufactured and distributed by HobbyTronics. The preinstalled software on the board allows us to connect a USB flash memory stick to the host board and read/write files via a Serial TTL (transistor-transistor logic) interface.

Figure 4; Hobbytronics USB Shield

It supports both FAT 16 and FAT 32 file systems, hence, can access large capacity memory sticks. The drive flash drive can be accessed via A PC terminal program or directly from a microcontroller using standard TTL Serial at different baud rates.

n our case) to the shield board via TTL serial communication. Commands are sent using ASCII characters and should be terminated by a carriage return character. Example of commands are: write <filename>, read <filename> [linenumber], and size <filename> [BYTE][LINE]. These commands are used to write a file by providing its name, read from a file by providing its file name and line number to read from, and get the size of a file in number of lines or bytes.

The shield has 2 modes of operation. First is the terminal mode, which is used for communication with PC's where commands are echoed back to the Serial interface. Second the silent mode, which is enabled by prefixing any command with a $ symbol. The silent mode is designated for microcontrollers since the command are not echoed back in this mode.

In our design we use the USB host shields as an interface between the Arduino Uno and the flash drives. Where whether we want to read or write to a file in a flash drive, we send a command to the USB host shield connecting the corresponding flash drive. So no direct communication happens between the Arduino Uno and the flash drives.

Buttons:

The buttons in our design are used to trigger events for the microcontroller. We have 2 buttons each triggers one of the 2 sub functions of our design which are: Partitioning sub function and Reconstruction sub function. When any of which is pressed, it sends a 5V signal to the Arduino and then the Arduino will trigger one the corresponding sub function code of the corresponding sub function button.

Software components:

Secret partitioning software code:

This is a software component, which is a piece of the embedded Arduino C code in the microcontroller. This part of the big main code is triggered when the partitioning button is pressed. This piece of code implements the function of the Secret Partitioning sub function which was explained in the sub functions section. This custom made software component will be further explained later in the report.

Secret reconstruction software code:

This is a software component, which is a piece of the embedded Arduino C code in the microcontroller. This part of the big main code is triggered when the reconstruction button is pressed. This piece of code implements the function of the Secret Reconstruction sub function which was explained in the sub functions section. This custom made software component will be further explained later in the report.

SoftwareSerial Arduino library:

This software component (SoftwareSerial.h) is an Arduino library which is used to make digital pins act as a Serial TTL RX and TX pins. It creates an SoftwareSerial object by providing 2 digital pins, where one will be RX and the other will be TX. For example, SoftwareSerial mySerial(2,3). This line of code will create an object, mySerial, which uses digital pin 2 as RX and 3 as TX. Using the library methods, we can send via this virtual serial pins using mySerial.write() and we can receive bytes using mySerial.read(). We needed to use this library since there are only 1 TX RX pins in the Arduino Uno, and we needed 4 of these. Whenever we want to send a command to the USB host shields we used mySerial.write() where mySerial corresponds to the digital pins connected to that USB shield. Same goes with reading.

5.2.2.2 Interfaces between hardware components and how they are connected to each other:

The Arduino is connected to the USB host shields using TTL serial communication interface, hence, communication between Arduino Uno and each USB host shield is done using TX and RX lines. Each TX and RX of a shield is connected to 2 digital pins of the Arduino Uno resulting in total of 8 connected digital pins. Each of the 2 digital pins, for example pins 2 and 3, will be defined as SoftwareSerial (will be explained more in components section) pins RX and TX respectively in the Arduino. Such that the TX and RX of the shield are connected to digital pins 2 (RX) and 3 (TX) respectively. So doing that for all 4 USB host shields we will have: USB host shield representing USB port 0 is connected to pins 2(RX) and 3(TX), USB host shield representing USB port 1 is connected to pins 4(RX) and 5(TX), USB host shield representing port 2 is connected to pins 6(RX) and 7(TX), and USB host shield representing port 3 is connected to pins 8(RX) and 9(TX). This is it to perform the communication between these components, but the shields and the Arduino need to be powered up, so the Arduino will get its power from a battery or by connecting it to a PC using USB cable, and then the shields will be powered by connecting their 5v pins to the 5v pin of the Arduino and by connecting their ground pins to gound pin of Arduino. This defines the interface between the Arduino and the USB Host shields and up to this point, we have the Arduino and the shields powered up and their communication interface is set up. The Arduino C code will be used to establish further successful communication between the Arduino and the USB host shields.

Now we move to the 2 buttons which are connected to Arduino at digital pins 10 and 11. The partitioning button will be connected to digital pin 10 and the reconstruction button will be connected to digital pin 11. The connections of the buttons are shown in figure 1.

Up to this point, we have shown our system architecture, its components, and the interfaces between components. Now we will move to component design and implementation.

5.3 Component Design and Implementation:

Off-the-shelf hardware and software components:

Four of our design components were off-shelf components, which were:

1- Arduino Uno.
2- USB host shields.
3- Buttons.
4- SoftwareSerial Librarys.

Custom software components:

Our design has only 2 custom software implemented components and no custom hardware component. The 2 custom software components were: Secret Partitioning code and Secret Reconstruction code. Each will be explained in the following paragraphs, but before that something's are needed to be clear. The Arduino sends and receives characters from the USB host shield using RX and TX which are define in the Arduino as a virtual serial port using SoftwareSerial library. Using the library we will have 4 objects, mySerial0, mySerial1, mySerial2, and mySerial3 to communicate with USB ports 0, 1, 2, and 3, respectively, each with its corresponding USB host shield. Both custom software components are included in single main code where there are 2 methods. One is setup() which is used to define and initiate variables and the other is loop() where this function loops forever. In the loop function there are the 2 partitioning and reconstruction codes, each will be triggered if its corresponding button is pushed. Now we can explain each of the 2 custom components.

Secret Partitioning code:

As it has been noted before, this partitioning code is triggered by the partitioning button. That is if the button is pressed (set to HIGH) then this code will be activated. This code implements the partitioning sub function which was explained in system sub functions. Now we will show the pseudo code of this piece of code to show how it actually implements the partitioning sub function.

Secret Partitioning Pseudo code:

```
if partitioning button is pressed (piece of code triggered)

get name of targeted file by using mySerial0.write("$ read name.txt 1")
//sends command to USB host shield representing USB port 0

save the name of targeted file in tfile variable

get size of tfile in number of lines by using mySerial0.write("$size tfile
line")

save it in linesnum variable

create 3 files names "WSDS.txt" in mySerial1, mySerial2, mySerial3

write tfile in the first line of "WSDS.txt"  in each mySerial1, mySerial2,
and mySerial3 (to save orginal file name )

write previously defined constant values x1, x2 and x3 in second line in each
file in mySerial1, mySerial2, and mySerial3 respectively (to know which
constant was used to calculate the secret share)

loop from i=1 up to i=linenume        //read all lines from original file

      get line i from tfile by using mySerial0.write("$read tfile (line
      number) i")

      save the line in array of characters lineChar[]

      calculate number of characters in lineChar and save it in variable
      length.

      define 3 arrays of characters y1, y2, and y3 ( will save secret share
      of each line)

      initialize m as a random number

      Loop from j=0 up to j=length     //calculate 3 secret shares of each
      character

            y1[j]=m*x1+lineChar[j]

            y2[j]=m*x2+lineChar[j]

            y3[j]=m*x3+lineChar[j]

            write y1, y2, and y3 in mySerial1, mySerial2, and mySerial3

            close the files in mySerial1, mySerial2, and mySerial3

            delete tfile in mySerial0

/* now all flash drives contain different secret shares of the file saved
with same file name WSDS.txt */
```

Secret Reconstruction code:

As it has been noted before, this Reconstruction code is triggered by the Reconstruction button. That is if the button is pressed (set to HIGH) then this code will be activated. This code implements the Reconstruction sub function which was explained in system sub functions. Now we will show the pseudo code of this piece of code to show how it actually implements the Reconstruction sub function.

Secret Reconstruction Pseudo code:

```
if Reconstruction button is pressed (piece of code triggered)

get name of original file by reading first line of WSDS.txt using
mySerial1.write("$ read WSDS.txt 1")

save it in a variable orignalFile

read second line of WSDS.txt by using mySerial1.write("$ read WSDS.txt 2")
which will determine which constant x1, x2, or x3 was used to build this
secret share

get size of WSDS.txt in number of lines by using mySerial1.write("$size
WSDS.txt line")

save it in linesnum variable

create 1 file with the name saved in variable original file in mySerial0

loop from i=3 up to i=linenume        //read all lines from 2 different
secret share files

      get line i from secret share 1 by using mySerial1.write("$read WSDS.txt
      (line number) i")

      save it in array of characters called y1[]

      get line i from secret share 2 by using mySerial2.write("$read WSDS.txt
      (line number) i")

      save it in array of characters called y2[]

      calculate number of characters in lineChar1[] and save it in variable
      length1

      calculate number of characters in lineChar2[] and save it in variable
      length2

      define an array of characters originalLine[] (to save the original line
      using secret shares)

      calculate m= (y2-y1)/(corresponding x of file in port1-corresponding x
      of file in port2)

      Loop from j=0 up to j=length     //calculate orginial line

            originalLine [j]=(y1-m*( corresponding x of file in port1))%256

            if originalLine [j] < 0

                        originalLine [j]= originalLine [j]+256
```

```
            close the file in mySerial0

            delete WSDS.txt files in mySerial1, mySerial2, and mySerial3
// now the flash drive in USB port 0 contains the original file
```

5.4 System integration

When we first received all of the components we didn't try to implement the final design directly. We had to test the USB host shields and to understand how they work since we already have experience with the other hardware components. We have done the testing of the USB host shields in several steps:

- Tested to read and write from a file using a single shield.
- Tested to copy a file from a flash drive connected to a shield to another flash drive connected to another shield.
- Tested using 4 shields to do the partitioning sub function.
- Tested using 4 shields to do the reconstruction sub function.

In each test step we noticed inconsistency in the behavior of the shields. For example we send a command and then command is echoed back. Another example is when we read a line from one the shields it reads the command concatenated along with the line itself. We tried to overcome the problem by creating a method which deletes the command from the String when it is received. However at each different testing step something odd happens, we tried to debug at the start and we were able to progress, however, things get more and more disturbing we couldn't understand what was happening. After continuous debugging we were able to partition a file to the different secret share, but when we want to save them on the flash drives we couldn't save different files, we had to save the same secret share in 2 shields. We couldn't understand what was causing such behavior although we spent a lot of time trying to debug this problem. As for the reconstruction part, we couldn't even read from a flash drive even though we used the same code from the partitioning part. Very weird behaviors were happening and in the end we figured out that the shield weren't reliable. More about the issues we faced will be discussed in the issues section.

6 Issues :

6.1 Issues:

When we first thought of this design, we thought that should be going so smoothly because everything was clear. However, throughout the progressions of our work we faced a lot of issues and challenges. Each will be discussed on its own in the following paragraphs following a chronological order:

1- The first issue we faced was how to communicate with flash drives using an Arduino. So we started researching for solutions. And after extensive research we found USB host shields with its different brands and kinds. Until we found the one we used for design, which has a preinstalled flash drive software.

2- The second issue we faced was how to make the Arduino act as a flash drive when it is connected to a PC. We spent a lot of time researching to find solutions to such a challenge. However, our attempt was unsuccessful. There may be a solution for such an issue but we were not able to find any. So we had to work around this issue and go with an alternative design idea. Where we will have a fourth USB host shield USB port 0 and we will connect a fourth flash memory to it. So now we don't have to take a file from the PC to partition, we have to take it from a flash memory. The functionality intended will be triggered by a corresponding button. The reason we had to go with this alternative is that from our research we only found USB host shields as a valid solution to communicate with USB devices using a microcontroller and was best fit for our project working time frame. And we couldn't use such shields to communicate with PC's because the Arduino has to act as device to the PC which is a host. And these shields work in the opposite manner. So we had to go with our alternative design to overcome this issue. Such change in the design cause a lot of things to change in our project.

3- The third issue we faced was with the testing of the USB host shields. We started by testing reading from and writing to a flash drive via a single USB host shield. For such test we used a ready-made code provided by the USB host shield manufacturer website, HobbyTronics. When we first tried to read and write we were successful, but when we re-tried after a couple of days we were not successful. Even though, we used the same code. We couldn't explain such behavior. So we started debugging the code, which should work, in weird ways until we were successful in reading and writing. After a couple of days we tried the same modified code and it didn't work again. So we thought that the problem may be in the Arduino we were using. So we decided to try another Arduino, and when we did, the code worked which made us believe that the problem was actually with the old Arduino. After that we started continuing in evolving our code to get the final design code even though we continued facing weird problems which we continued and were able to solve in weird ways.

4- The fourth issues was that when we read a line from a flash drive, for some reason the command we sent to read the line is also read along with line. We couldn't know the source of such a thing, so we had to overcome the problem by creating a method which takes the corrupted line, removes the command from it, and return the correct read line.

5- The fifth issue we faced was with one the USB shields, where while testing it suddenly stopped working. The blue LED in the shield which indicates that a flash drive is connected or not was always off. That is, it wasn't able to read that a flash drive is actually connected. So we had to change out prototype to show only 3 USB ports, where the fourth one was not actually crucial for a prototype.

6- The sixth issue we faced was again with USB host shields. We were able to write the partitioning sub function code but stopped at the point of writing the 2 different secret share files to the flash drives connected at USB ports 1 and 2. For some reason the first secret share file was successfully written to the flash drive in USB port 1 but the second secret share wasn't written on the flash drive connected in USB port 2. So we tried to write first secret share on both flash drives and it worked, however, we re-tried several times and got different slightly different files in each time. For some reason writing the same file was to certain extent applicable where writing 2 different files wasn't. We really couldn't understand what may have cause such a problem. We spent a lot of time trying to solve it, but in the end we couldn't. So we had to ignore the issues, which affected our final prototype.

7- The seventh issue we faced was with the reconstruction part of the design. We were able to write the full code, but we faced problems. Not surprisingly the problems were with reading from and writing to a flash drive. So we had to debug by returning to the HobbyTronics provided code for reading and writing, and still we were not successful. We spent a lot of time trying to overcome this problem… but apparently there were some problems with the USB host shields.

After finishing with testing and running through all of the above issues, we were not able to have a complete prototype due the weird inconsistent behavior of the USB host shields. The only possible explanation we came up with was that the USB host shields were not reliable and give different results each time. We have sent an email to the manufacturer explaining that there USB host shields may have not been well developed. There is another possibility for the issues we faced which is that the serial communication of the Arduino has some problems, which is much less likely to be the cause.

6.2 Limitations and constraints:

In this section we will talk about the limitations and constraints of our desing disregarding the issues we faced and assuming there is a working design prototype. Following are the limiations and constraints:

1- The flash drive shouldn't be removed while a function is being performed, or else the flash drive would be damaged.
2- The name of the file to be partitioned, have to be saved in the first line of a text file called "name.txt".
3- The file name is assumed to be provided correctly, while if not, the function of the device won't trigger.
4- The device can't show nor it can know the available size of flashes. So it is assumed that there is enough size in the flashes to perform a function.
5- The device function speed is relatively low since it is limited by the processing of the Arduino and the USB host shields and also by the embedded delays in the code to give enough time for the shields to process data and commands.

7 Engineering Tools and Standards

7.1 Used tools: which of the available tools are chosen, and why:

- **Arduino IDE to program and compile our code:**

We chose to use Arduino IDE because it is the Arduino open-source environment makes it easy to write our code and upload it to the Arduino UNO that we are using. It also makes the process of testing and debugging more efficient since the Arduino IDE is optimized to work with Arduino microcontrollers.

- **Arduino UNO board:**

We used Arduino UNO to be the microcontroller to run our code on because it supports every functionality we need for the device to work, it has enough pins to connect the USB shields to it and enough processing power to do run our software. Another reason why we used Arduino UNO is that it is easy to use and it makes the process of prototyping much more faster and easier.

- **Used programs for diagrams created above:**

To create the UML diagrams we used an online application called “Lucid chart”. Since our UML diagrams were not as complex as some other projects, we found that this software can support what we wanted to do. The software is very interactive, you can drag and drop the components you want and you can add pictures to show the wiring, which made it easier for us to better explain the technical side of our project.

7.2 Used standards: which of the relevant standards are used, and why:

- **USB protocol already used by flashes(have to use):**

The USB shields would communicate with the flash drives using the USB protocol, we had to use this protocol because it was specified by in the project and since USB was designed to standardize the connection of computer peripherals.

- **TTL protocol(the interface provided by shields and can be interfaced with Arduino):**

The serial TTL interface is the interface between the USB shields and the Arduino UNO. With TTL serial there are two unidirectional data lines each line is driven by the sender(Tx), it the line voltage = Vcc (5V in our case) this would represent a 1, else it would be a 0 bit. We had to use this protocol because the USB shields are ready made components, and they were design to communicate with the Arduino using TTL protocol.

8 Conclusion:

In conclusion, we have learned a lot of things throughout the weeks we spent on this project. We learned about the USB communication and USB hubs design. We also learned the steps of coming up with the design of a device and satisfying it requirements. At the start we knew we will make 3-WSDS device and knew its functionality, but we had no idea how to do it. We started thinking of possible design decisions and took the decision we have seen most appropriate for our case. In the end we were able to come up with a full design of such device even though it didn't satisfy all of the requirements. We also faced a lot of issues and solved most of them which was very useful for us as engineers. This has boosted our critical thinking and problem solving capabilities.

If in the future we were to work in a similar project we would do something's differently. First of all we will consider all design options even if we lacked experience in some of them. Because these solutions we lack experience with, may be much better and easier solutions. So we need to spend some time learning things we don't know in exchange of using what we already know if the thing we will learn will save our time eventually. But doing which, requires the availability of time. Another thing is that before considering a design option, we need to see if such an option would actually cover all of the requirements of the required design in the end.

At the end of this report we would like to note that we have spent a lot of time designing this device and researching its potential design solutions. We have never spent as much time with any other previous project which is expected since this is our senior design project. And we have worked really hard trying to get the final expected prototype, however, in the end we only succeeded in completing parts of it.