REPLICATED SECRET SELF-DESTRUCT USB

Alex Clemens

Varun Siva

Danny Metzger

Final Report for ECE 445, Senior Design, Spring 2025

TA: Michael Gamota

7 May 2025

Team 69

Abstract

This project presents a custom USB flash drive designed with hardware based security features in order to protect sensitive data. Instead of relying on traditional passwords, the custom PCB system utilizes Replicated Secret Sharing to split an AES-Encryption key across three separate authentication cards. Access to data held on the USB is only available after a user has inserted ⅔ authentication cards. The hard drive includes a tamper detection system that also safeguards from hardware tampering. Whenever the USB is attempted to be accessed by malicious parties (by failed authentication or tampering) the data present will automatically be erased to ensure security. A microcontroller coordinates secure communication between the USB interface, NAND flash memory, and secure elements. The device was successfully built and tested for encryption, data storage, and authentication, although tamper-triggered data destruction was not fully functional during the final demo. This system offers an innovative approach to hardware secure storage.

Contents

1. Introduction

Traditional Flash drives, while often convenient for basic data storage, pose significant security issues and risks when being used to store sensitive information [1]. Despite the possibility and availability of software based encryption tools, many of these remain vulnerable to brute force attacks, password theft, or hardware forensic recovery. Hardware-encrypted devices often bypass some of these issues but still use passwords, which can often be hacked in under a few seconds and still lack a tamper response [2]. This presented us with a critical need for a more secure, user independent solution for security of classified or private data on a USB.

To address these concerns, our team developed a secure USB flash drive that implements a combination of hardware based encryption two factor authentication, and the addition of an automated data erasure sequence in the event of unauthorized access attempts. This system utilizes replicated secret sharing to distribute parts of a 256-bit encryption key across three different physical authentication cards. Two of these cards will be required to access the data present on the USB. A tamper detection system has also been put in place to protect from physical intrusion, also resulting in data erasure.

This report details our full design and implementation of the system, which includes physical hardware, embedded firmware, cryptographic protocols, and power systems. Each subsystem will be discussed in its design, along with its testing and verification of operation as intended. We conclude afterwards with an evaluation of the system's success, limitations, and future improvements.

1.1 High Level Requirements

The high level requirements for this secure USB flash drive are laid out as follows:

1. The flash drive must allow a maximum of 5 failed authentication attempts before triggering the self-destruct.
2. The flash drive must require at least 2 out of 3 physical authentication cards to decrypt the hidden partition.
3. The flash drive’s various modes of encryption should all utilize at least 256-bit keys.

2 Design

The design of our system centers on a custom USB flash drive that prioritizes hardware-based security and physical authentication. It consists of multiple subsystems working together to store, protect, and securely erase encrypted user data. Each subsystem, including authentication, storage, tamper detection, crypto control, and power, was carefully engineered to ensure reliability, interoperability, and resilience against physical and software-based attacks.

2.1 Block Diagram

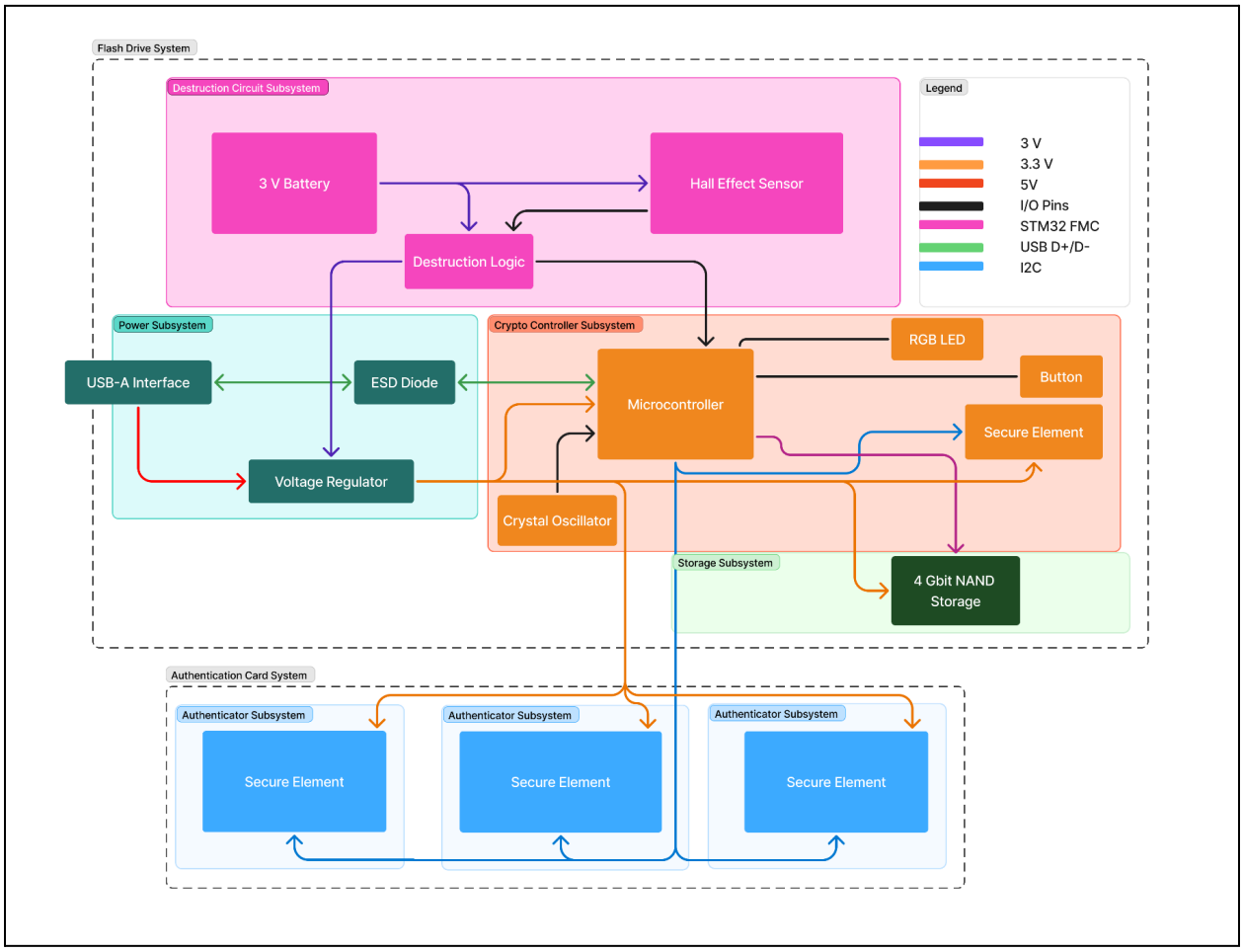


Figure 1: Replicated Secret Shared Self-Destruct USB Block Diagram

2.2 Authentication Card Subsystem

The Authentication Card Subsystem allows for secure, hardware-based authentication before the USB grants access to the NAND storage data on the USB. This subsystem is located on three entirely separate PCBs that connect to the main USB PCB for the authentication process. There were many of these authentication card PCBs created, but only three of them are initialized for each USB PCB. This subsystem communicates with the Crypto Controller Subsystem, sending or withholding the key shares that ultimately enable (or deny) access to encrypted data when ⅔ of the authentication cards are plugged into the USB via GPIO pins. This subsystem mainly consists of a ATECC608B Secure Element chip placed on each Authentication Card, which holds a cryptographic key pair. This key pair is split with a system described below.

Replicated Secret Sharing stems from multi-party computation and involves a key, K, being split into three components K_0, K_1, and K_2 [6]:

$$K = K_0 \oplus K_1 \oplus K_2$$

Since XOR is a secure operation, it ensures that all three shares are needed to reconstruct K. This is a 256-bit key and each Secure Element holds a combination of two of the keys. For example, Card 1 would hold K_0 and K_1. That way, K can be reconstructed with any two cards being plugged in. No other device should be able to extract these keys from the Secure Element. The MCU can verify a reconstructed K by since it holds a SHA-256 hash of K in secure storage. This can be easily compared to check if the right authentication cards are plugged in. K can then be used for AES-256 on the storage subsystem.

When plugged into the USB via GPIO pins, the secure element will send its key-pair with I2C communication, which travels to the microcontroller and either be confirmed or denied as the correct value. This subsystem is driven by the 3.3V output from the voltage regulator on the main USB PCB as well as the USB's common ground that both come through the GPIO pins.

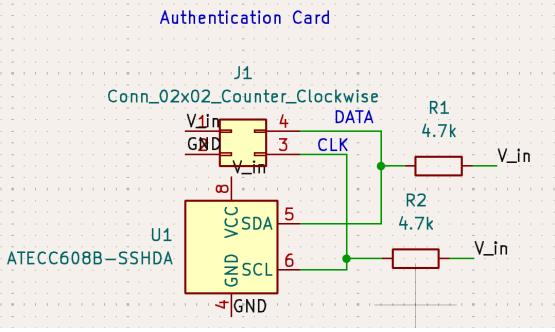


Figure 2: Authentication card schematic

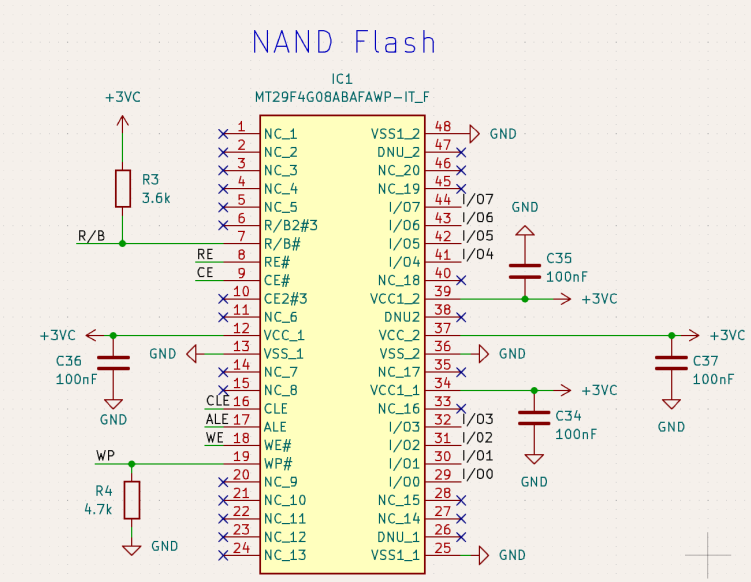
2.2.1 Authentication Card Subsystem Decisions

Since this subsystem was not too complex on a hardware level, there were a limited amount of design considerations that went into our development process. We always knew we wanted to use some sort of chip to store data values, but we wanted to ensure that storage was also secure, which prompted us to utilize a secure element. There were a multitude on the market, and for design purposes we wanted one that would hold an encryption of the two key elements that would be stored on each authentication card. We specifically chose the ATECC608B-SSHDA-T as the secure element since it allowed for this functionally as well as locking of data slots to ensure the authentication cards could not be reinitialized with another USB.

Looking back it may have been worth considering including a less restricting secure element, since the ATECC608B-SSHDA-T was so secure that it was difficult to design and test with. Additionally the datasheet for this part was difficult to acquire as you have to sign an NDA at a local Microchip office.

2.3 Storage Subsystem

There are multiple options for a storage device on any type of USB flash drive. The cheapest, and most common form factor is NAND flash. We went with a small NAND flash of 0.5GB because the storage size was not the focus of this project. We considered eMMC and NOR since eMMC has better erasure properties and NOR is simpler to work with. In hindsight, we should have chosen NOR, as it would have made writing firmware much easier throughout this project.



2.3.1 Storage Subsystem Decisions

This subsystem is responsible for handling all read, write, and erase operations to user data. This subsystem simply consists of the MT29F4G08ABADAWP-IT NAND Flash chip in order to hold and interact with the USB data [3]. Decryption of the data occurs after the Crypto Controller Subsystem authorizes it, at which point the microcontroller can securely read and write to NAND Flash. The storage contains 0.5GB of NAND flash storage and is powered from the 3.3V bus from the rest of the device. The microcontroller handles all the encryption and decryption of data, so the storage subsystem is simply responsible for storing, reading, writing, and erasing arbitrary data. In the event of tampering, the storage subsystem uses a *BLOCK ERASE* function over all the blocks of data within the flash [4].

2.4 Tamper Detection Subsystem

The Tamper Detection Subsystem ensures tight protection against physical attacks to tamper with the physical USB PCB. This subsystem is located on the USB PCB and consists of a A1125LLHLT-T Hall Effect Sensor(HES) that detects when the USB enclosure is tampered or removed and initiates the sequence to erase the data present on the NAND memory. The Hall Effect Sensor sends a signal based on the presence or absence of magnetic fields to a series of diodes and a dual ideal diode referred to as the Destruction Logic in the Block Diagram above [5]. The enclosure we had designed for the USB has a small magnet attached on its inside ceiling, which will constantly cause the Hall Effect Sensor's output signal to be ignored. Once the enclosure is tampered and removed, the magnetic field is absent from the Hall Effect Sensor's radius and triggers its output signal, which is fed into the Destruction Logic. The output of the Destruction Logic will indicate to the Microcontroller that the USB has been tampered with, initiating the data erasure sequence. This Tamper Detection is powered by the CR2477X-HO 3V coin battery within the subsystem, so that the Hall Effect Sensor is able to detect and communicate while the USB is unplugged, when an attack such as this is likely to occur. The Destruction Logic switches the Microcontroller and NAND memory to be powered by this coin battery when the Hall Effect Sensor detects tampering so they are able to carry out the data erasure whilst the USB is unplugged. The battery powers a supercapacitor that allows for sustained power delivery to the system that the coin battery itself

was not able to provide. In order to avoid premature tampering signals by the Hall Effect Sensor before the USB enclosure and magnetic field are secured, there is switch logic using diodes to disconnect the communication between the Destruction Logic and the Microcontroller until the Cryptographic keys have been initialized. A circuit diagram of this system can be seen below in Figure 4. Unfortunately, we did not have time to fully test and develop this subsystem, which we discuss further in section 3.3

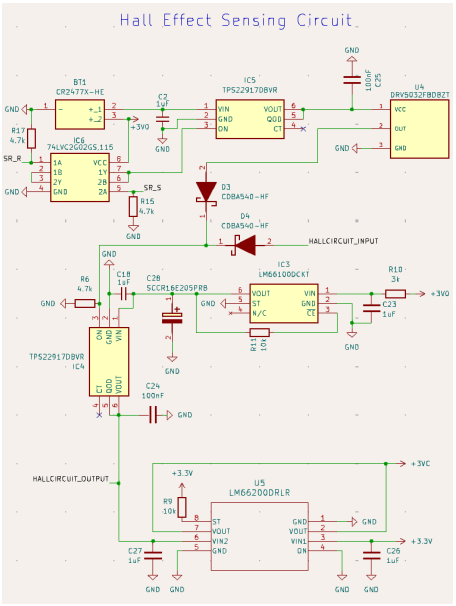


Figure 4: Tamper Detection Subsystem schematic

2.4.1 Tamper Detection Subsystem Decisions

The Tamper Detection subsystem required lots of time and effort to design, specifically with the destruction logic. The first challenge was figuring out a way to correctly send signals from the hall effect sensor to the microcontroller using a series of diodes and ideal diodes. We did not want this signal to turn on until after the initialization, so a signal from the microcontroller had to control part of the logic as well.

Another challenge was finding a way for the 3V coin battery to power this circuit as well as the microcontroller and NAND whenever this hall effect sensor was triggered. This included even more diode logic and the need for a power switch in the case that the USB power was plugged in while the coin battery was attempting to power everything.

Lastly, the coin battery was unable to give us the sustained current draw we needed to power the entire system during this process, introducing the need for a super capacitor that would allow for sustained power output when this tamper-caused data erasure sequence was necessary.

2.5 Crypto Controller Subsystem

The Crypto Controller subsystem functions as the central processing unit of the entire PCB, managing and coordinating all other components. It comprises an STM32U5A9ZJT6Q microcontroller, a TS02-66-55-BK-100-LCR-D button, a 551-0207-004F LED array, an ABM8AIG-12.000MHz-8-D4Z-T

crystal oscillator, and an ATECC608B-SSHDA-T secure element. The microcontroller interfaces with three primary peripherals. First, it connects to a USB port, with the crystal oscillator providing the stable, dedicated clock required for reliable USB communication. Second, it interfaces with a NAND flash chip using its on-chip Flexible Memory Controller (FMC). Third, it communicates with authentication cards over the I²C bus. The on-board secure element that is part of this subsystem is used to hold the secret key of the cryptography subsystem, and is also communicated with over the same I²C bus as the authentication cards. The LED array is connected to the microcontroller over some 150-ohm current limiting resistors. This LED is used to display the state of the FSM we have designed to traverse the states of the decrementing authenticating attempts remaining, and the open and cleared state. In Figure 6 below, a success is defined as two or more authenticated authentication cards being present, while a failure is defined as one or less. The button is used for user interrupt, and upon being pressed it raises an interrupt which is serviced by the RTOS running on the chip.

Speaking of the on-chip RTOS, we are using four main software libraries which are running on the microcontroller. ThreadX is STM's implementation of the Azure RTOS for their STM32 microcontrollers. We used this library to coordinate all of our firmware running on the microcontroller, and it provided us with a ready-made implementation of threads and interrupts. USBX is STM's library for all usb connections. This library managed our USB stack and allowed us to enumerate our device as a mass storage controller (MSC) usb class, which is what all flash drives present themselves as when connected to a host computer. Finally, to manage the NAND flash we used two libraries, LevelX and FileX. LevelX was the low-level driver we used to manage accessing the NAND flash as well as wear-leveling. FileX was used to format the storage of the NAND flash and provide a file system the computer could interface with.

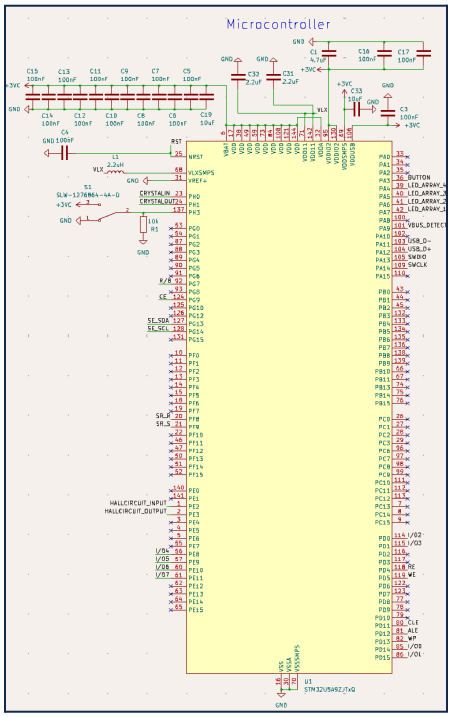


Figure 5: Crypto Controller Subsystem Schematic

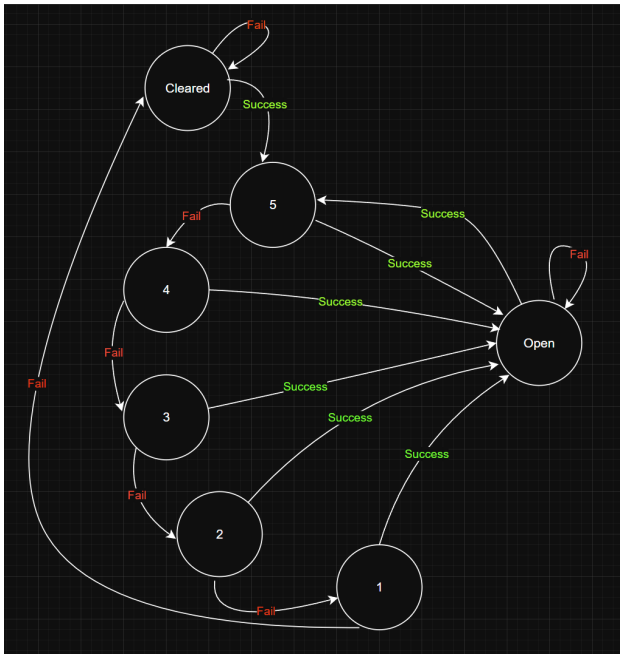


Figure 6: FSM used to control states of USB

2.5.1 Crypto Controller Subsystem Decisions

The most consequential design decision of the crypto controller subsystem, and maybe the entire project as a whole, was the choice of our microcontroller, the STM32U5A9ZJT6Q. This microcontroller was chosen because it had all of the peripherals we needed, a USB High-Speed interface with an internal PHY controller, a flexible memory controller for interfacing with our NAND flash, and multiple I²C connections. Additionally, it had plenty of on-chip storage, a hardware SHA-256 implementation, and multiple low-power modes. However, integrating this microcontroller was more difficult than it needed to be. Firstly, the smallest package of this microcontroller that had everything we needed was the LQFP-144 package, which has 144 pins that are very close together. This resulted in a very difficult solder, and made the long feedback loop of hardware development even longer. Additionally, a full RTOS was definitely overkill when we could have achieved our desired functionality with a simple event loop. The reason we were forced into using a RTOS was because the USBX, LevelX, and FileX libraries all required ThreadX to work properly. There are legacy libraries that achieve the same thing as USBX, LevelX, and FileX, but because this microcontroller is so cutting-edge, they are not supported. If we were to do this project again we would probably have used a simpler microcontroller that didn't have an on-chip PHY for USB communication or a FMC for the NAND flash, opting instead to use a NOR flash which can use the more readily available octo-SPI.

2.6 Power Subsystem

Power is drawn directly from the USB port to power all of our components during normal use. This is done by taking 5V from the USB and using a regulator to step it down to 3.3V. In addition, the USB data lines must be protected from electrostatic discharge.

2.6.1 Power Subsystem Decisions

This flash drive's power is very similar to other flash drives, aside from reverse current protection to the battery source from the tamper detection subsystem. The voltage regulator accepts a 5V input and outputs 3.3V at up to 500mA, which is enough to power the secure elements, NAND flash, microcontroller, and all peripherals [7]. The original design did not account for the microcontroller's second source, which needed 1.1V through an internal switching mode power supply. A simple inductor circuit was added to power the microcontroller thoroughly. A dual ideal diode was added to switch source between the battery and the USB connector. The USB connector also includes a differential data pair, which is routed through an ESD diode to protect the microcontroller from static discharge. The data lines then connect to the microcontroller's embedded high-speed USB 2.0 PHY, enabling 480 Mbps data transfer.

3. Design Verification

The final revision of our PCB had multiple functional subsystems, including USB mass storage, real-time encryption and decryption capabilities. The device successfully operated as a standard flash drive with AES-256 encryption through the authentication cards. However, due to time constraints, we were unable to build or test the tamper detection subsystem.

3.1 Authentication Card Subsystem Verification

The authentication card subsystem had multiple requirements we had to ensure worked properly before our final secure USB was operational. First of all, we were able to ensure that initialization only happens properly when all 3 authentication cards are plugged into the USB PCB via GPIO pins. After this initialization process, it was confirmed that the USB data was not available to the user when at least ⅔ authentication cards were not used for authentication. Additionally, it was proven that after the initialization process, other dummy authentication cards were not able to unlock the USB besides those that had been initialized. Finally, our tests showed that cards were not able to set new K-pair values on the authentication cards, and reinitialization of authentication cards on the same or different USBs were possible. All of these verifications and tests that we detail further in the Appendix proved our authentication card subsystem to be operational.

3.2 Storage Subsystem Verification

The storage subsystem contained multiple requirements traditionally associated with any other encrypted flash drive. We verified reads and writes to the flash drive with authentication cards connected, and were able to store a large file on a computer, unplug and plug the device, and see that the data was still securely stored on the flash storage. Since it is not possible to mount an encrypted drive, we used a debugger to verify that the contents of the NAND flash were encrypted when not authenticated. There was no readable plaintext and we were able to confirm that the ciphertext was associated with the encrypted plaintext by printing AES-256 outputs. For the erasure requirement, we tested and confirmed low level block erases [8], but did not have time to write the firmware to fully format the drive under the right conditions. This requirement was not verified.

3.3 Tamper Detection Subsystem Verification

Unfortunately, we did not have time to connect and fully test the functionality of this subsystem. We ran multiple simulations on the power switching and capacitor current flow using MultiSim and are confident in the parts we chose. However, we could not test or verify any signals from the Hall effect sensor and data deletion.

3.4 Crypto Controller Subsystem Verification

The main verifications of this subsystem were communication with all other peripherals. After writing firmware for the NAND flash, we tested reading and writing a range of block addresses to the flash, ensuring that the read matched the write using a debugger. To test I2C communication, we used an oscilloscope and sent multiple commands to check if the binary matched the command that should be sent. With the secure elements connected, we were able to probe the I2C addresses and ensure each secure element could communicate with the MCU. We loaded simple firmware to test all LEDs and observed toggled states for multiple states in the finite state machine. This was tested multiple times with software debouncing to check that one button press always moved exactly one state. To verify correct

communication between the host computer and the USB we viewed the packets being sent in wireshark. The final test of this system was the device enumerating as a mass storage controller class device on our computer, being able to mount it as a drive, and being able to create and destroy files from the host computer.

3.5 Power Subsystem Verification

The USB power provided 5V, routed to a regulator stepped down to 3.3V. We connected the regulator and ensured that the 3.3V output was within tolerance of 3.0 to 3.6V. On a breadboard, multiple loads were added to test consistent 3.3V output. In addition, a custom variable source was provided instead of USB and we verified a tolerance of +/- 1V safely regulated down to 3.3V. The data lines required ESD protection and this was accomplished with an ESD diode. To verify this, we applied a small discharge to the USB housing via a metal tool and verified the data lines were unaffected with an oscilloscope. In addition, we plugged and plugged the flash drive many times to ensure that data lines were not affected. Although not an original requirement, we also verified cross-interference between the battery and USB by testing that only one source could be active at a time using a multimeter.

4. Costs

4.1 Parts

Table X Parts Costs

Part	Manufacturer	Description	Retail Cost ($) (Also Actual Cost)	Bulk Purchase Cost ($)
STM32U5A9ZJT6Q	STMicroelectronics	Microcontroller	$16.50	$10.24
ATECC608B-SSHDA-T (4)	Microchip Technology	Secure Element	$0.90	$0.725
USB1061-GF-L-A	GCT	USB Port	$0.86	$0.47
DRV5032FBLPGM	Texas Instruments	Hall Effect	$0.37	$0.156
MT29F4G08ABAFAWP-IT:F	Micron	NAND Flash	$3.56	$2.75
LD39050PU33R	STMicroelectronics	Voltage Reg.	$0.99	$0.484
ABM8AIG-12.000MHz-8-D4Z-T	ABRACON	Crystal	$0.81	$0.416
CR2477X-HE	Murata Electronics	Battery	$2.78	$1.47
USBLC6-2SC6	STMicroelectronics	ESD diode	$0.36	$0.077
SLW-1276864-4A-D	Same Sky	Switch	$0.84	$0.456
30 pf capacitors	ECEB Lab	30 pf capacitors	Free	Free
0.1 uf capacitors	ECEB Lab	0.1 uf capacitors	Free	Free
10 uf Capacitors	ECEB Lab	10 uf Capacitors	Free	Free
4.7k Resistors	ECEB Lab	4.7k Resistors	Free	Free
150 Resistors	ECEB Lab	150 Resistors	Free	Free
GRM21BR61A225KA01 (2)	Murata Electronics	2.2 uf Capacitors	$0.18	$0.038
DFE201612E-2R2M=P2	Murata Electronics	2.2 uH Inductors	$0.25	$0.124
TS02-66-55-BK-100-LCR-D	Same Sky	Button	$0.10	$0.055
74LVC2G02GS,115	Nexperia	Dual NOR Gates	$0.31	$0.109
LM66100DCKT	Texas Instruments	Ideal Diode	$0.83	$0.384
SCCR16E205PRB	KYOCERA AVX	2 F Supercapacitor	$2.03	$0.879
551-0207-004F	Dialight	LED Array	$3.96	$1.92
LM66200DRLR	Texas Instruments	Dual Ideal Diode	$0.46	$0.197
TPS22918DBVT (2)	Texas Instruments	Power Switch IC	$1.06	$0.508
ALC 2020	Alliance Magnets	Case Magnet	$4.00	$2.50
Total			$45.09	$26.371

Table 1: Parts Cost Table

4.2 Labor

Each member contributed an estimated 200 hours to this project. With 3 members at an ideal salary rate of $40/hr and the 2.5 times multiplier our total labor costs are an estimated:

$$3 \times \$40 \times 200 \times 2.5 = \$60,000$$

5. Conclusion

5.1 Accomplishments

We believe that we succeeded in creating a cryptographically secure functioning flash drive. The data stored on our PCB will be uncorrupted, retrievable, and inaccessible to the majority of attackers. Additionally, our PCB functioned on its own and did not need any supporting breadboards.

5.2 Uncertainties

While this system is much more secure than a typical flash drive, we did fall short of our goal of absolute security. Because of the lack of a functioning tamper detection subsystem, it is technically possible for an attacker to solder wires onto the correct pins and brute force the encryption. While we did use very solid encryption throughout this project, the increasing computational power of systems would in theory be able to break our encryption sometime in the future.

5.3 Ethical considerations

Our device includes a data erasure procedure, which necessitates transparency in its ability and usage to any user. Per IEEE ethics, we must "be honest and realistic in stating claims or estimates" (IEEE CoE Section 5). Users must be clearly warned of potential data loss. According to the ACM, technology should "contribute to society and human well-being" (ACM CoE Section 1.1). Our device protects sensitive data but must also remain safe and usable.

Additionally, we must avoid harm through accidental deletions of data (IEEE CoE Section 9).This is why the USB requires 5 failed authentication attempts before deletion. Following ACM guidelines (ACM CoE Section 2.1), we tested the USB device extensively to ensure its successful operation as intended.

Furthermore, under Illinois law (815 ILCS 530), we ensure that data is erased with no possibility of recovery. We also upheld ECE445 student standards, such as honesty, documentation, proper attribution, and team respect.

5.4 Future work

An obvious candidate for future work is to finish up the Tamper Detection subsystem and design a case to hour our PCB in. However, there are multiple other improvements that can me made to this project. The footprint of the PCB can be reduced, as under the battery there was a fair bit of unused space. Additionally, we only had four LEDs to display seven FSM states, so an improvement to the LED user interface would make our project easier to use.

6. References

[1] Volle, A. (2025, February 14). *USB flash drive*. Encyclopædia Britannica. https://www.britannica.com/technology/USB-flash-drive

[2] Stouffer, C. (2023, June 27). *139 password statistics to help you stay safe*. Official Site. https://us.norton.com/blog/privacy/password-statistics

[3] Bigelow, Stephen J., and Margaret Jones. "What Is NAND Flash Memory?: Definition from TechTarget." *Search Storage*, TechTarget, 12 May 2023, www.techtarget.com/searchstorage/definition/NAND-flash-memory.

[4] "Erasing Data on a Flash Storage Device Is Not as Easy as It Seems." *How to Securely Erase Flash Storage*, www.atpinc.com/blog/secure-erase-ssd-data-sanitization. Accessed 6 Mar. 2025.

[5] Soltero, M. (n.d.). *What is a hall-effect sensor?*. SSZT164 Technical article | TI.com. https://www.ti.com/document-viewer/lit/html/SSZT164

[6] Foundation, P. B. (2021, June 4). *MPC techniques series, part 1: Secret sharing*. Medium. https://medium.com/partisia-blockchain/mpc-techniques-series-part-1-secret-sharing-d8f98324674a

[7] Datasheet - STM32U5Axxx - Ultra-low-power arm< ... (n.d.). https://www.st.com/resource/en/datasheet/stm32u5a5aj.pdf

[8] *Global leaders in semiconductors*. Micron Technology. (n.d.). https://www.micron.com/?login=&returnUrl=%2F-%2Fmedia%2Fclient%2Fglobal%2Fdocuments%2Fproducts%2Fdata-sheet%2Fnand-flash%2F70-series%2Fm70a_4gb_3v_nand_spi.pdf%3Frev%3D14a6ca3df4e046d09f2db87b126018ed

Appendix A Requirement and Verification Table

Table 1 System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
All 3 authentication cards are able to be plugged into the USB via GPIO pins and initialized at the same time.	1. Before any authentication cards have been initialized with the USB, connect all 3 of the Authentication Card PCBs into the USB. 2. Next, press the button present on the USB to initialize the cryptographic keys for the system. 3. Measure the encrypted values of the cards once initialized for testing purposes. To do this, connect the 3 Authentication Cards again and have the USB microcontroller send the encrypted values to the NAND memory to be viewed as verified during production.	Y
Once initialized, the K-pair held in each authentication card can not be altered or changed. Additionally, no further authentication cards can be initialized.	4. Insert one initialized authentication card and two uninitialized cards into the USB. 5. Press the USB button to attempt reinitialization of cryptographic keys. 6. Test the two newly initialized cards by inserting them into the USB and pressing the button—ensure access is denied. 7. Measure the encrypted values of the previously initialized card after reinitialization, then insert it along with another original card. Send the encrypted values to NAND memory and confirm they remain unchanged for production verification.	Y
When connected to the USB PCB, the Authentication Card Secure Element is automatically prompted to send its K-pair via I2C communication.	8. Insert the authentication card into the USB PCB and monitor I²C lines (SDA/SCL) with an oscilloscope or logic analyzer.	Y

	9. Verify that the secure element powers on and responds to an I²C request from the USB PCB. 10. Confirm that the secure element transmits the K-pair via I²C. 11. Cross-check the received K-pair against expected values stored in the secure element.	
The NAND Flash correctly reads and writes data when the correct Authentication Cards are utilized.	12. Insert a correct authentication card into the USB PCB and attempt to write test data to the NAND flash. Record the operation status via serial debugging. 13. Remove the USB and place it back in with the correct Authentication Cards. Read the stored data from and compare it to the original input to confirm accuracy. 14. Remove the correct authentication card and attempt to write or read data again. Verify that access is denied.	Y
The NAND Flash contains only encrypted data, nothing that would be understandable without an encryption key.	1. Write test data to the NAND flash using the correct authentication card and encryption process. 2. Read back the stored data directly from the NAND flash using debugging tools or a memory reader. 3. Verify that the retrieved data appears as encrypted, with no readable plaintext or identifiable patterns. 4. Attempt to decrypt the stored data using the correct encryption key and confirm that it successfully restores the original input.	Y
All the valid data blocks stored on the NAND Flash are deleted once the destruction sequence is enacted with the	1. Ensure the NAND flash contains valid encrypted data by writing test data using a correct authentication card.	N

physical tampering or incorrect Authentication Cards.	2. Trigger the destruction sequence by either physically tampering with the device (e.g., removing the USB casing/magnet) or using an incorrect authentication card. 3. Attempt to read data from the NAND flash after the destruction sequence and verify that all valid data blocks have been erased. 4. Confirm that only erased or randomized data remains by checking for unreadable or zeroed-out memory regions with the microcontroller.	
The signals sent from the Destruction Logic are not able to interface with the Microcontroller until the Cryptographic Keys are initialized.	1. Plug in the USB without initializing the cryptographic keys and monitor the connection between the Destruction Logic and the microcontroller. 2. Utilize a multimeter to verify that no signals from the Destruction Logic are received or processed by the microcontroller, as they are not connected yet. 3. Initialize the cryptographic keys and check that the microcontroller now recognizes and connects to signals from the Destruction Logic after a switch has been closed.	N
Once the USB casing and magnet are removed, the Hall Effect Sensor stops sending its signal to the Destruction Logic.	1. During testing and before implementing the firmware to delete the USB's data after tampering, connect an LED to the signal sent out by the Hall Effect sensor. 2. Repeatedly put on and take off the USB enclosure with the magnet attached. Ensure that the signal lights up the LED when the case is closed, and turns the LED off once the case and magnet are removed.	N
The Destruction Logic sends signals to the microcontroller to initiate data deletion and	1. Ensure the system is in an idle state with the USB casing and magnet in	N

connect routing of the 3V coin battery to power the microcontroller and NAND memory.	place. Record the signal from the Destruction Logic to the microcontroller via serial debugging. 2. Before implementing data deletion firmware, test by removing the USB casing and magnet to trigger the Hall Effect Sensor. Record the signal from the Destruction Logic to the microcontroller via serial debugging. Confirm the signal is now present. 3. Also affirm that voltage has now been routed from the 3V coin battery to both the microcontroller and the NAND memory. 4. Once implemented the data deletion firmware, repeat the process and see that previous files on the NAND storage have been deleted once plugged back into the computer with the correct authentication cards.	
The microcontroller must be able to successfully transmit data in and out of the USB port	1. Connect the device to a PC using the USB port. Transfer a 1 MB test file from the PC to the device. 2. Read back the file stored on the device to ensure the data matches the original file by comparing checksums. 3. Transfer that same 1MB test file back to the PC, and ensure that the file received on the PC is identical to the one originally sent. 4. Monitor USB status logs (through microcontroller's debug interface) to confirm no errors occur during read/write operations	Y
The microcontroller must communicate with the NAND flash using the Flexible Memory Controller (FMC)	1. Load a test firmware onto the microcontroller that writes a recognizable pattern to a specific range of addresses in the NAND flash. 2. Read the same range of addresses back into microcontroller memory	Y

	3. Verify that the data read from NAND matches what was written, indicating proper FMC operation.	
The microcontroller must communicate with the secure elements using I2C	1. Power on the system and run a diagnostic routine that attempts to detect all secure elements on the I2C bus 2. Ensure the microcontroller successfully identifies each secure element by its unique address 3. Write a test value to one secure element and confirm it is not written to any others 4. Read back the value to verify that the correct secure element received it 5. Repeat the write/read process for each secure element to ensure reliable communication across all devices.	Y
The LED must display the correct status when the button is pushed.	1. Load a firmware sequence that changes the LED status when the button is pressed (e.g., from “off” to “blue,” or from “green” to “red,” depending on the intended system state). 2. With the system powered, observe the LED while pressing the button: 3. Confirm that pressing the button once switches the LED to the expected color or mode (e.g., authentication mode). 4. Release the button and confirm that the LED returns to its previous state or moves to the next intended state as specified by the system design. 5. Repeat the test multiple times to ensure consistent behavior and no software/hardware debounce issues	Y
Must be able to regulate USB power to power components throughout the duration of connectivity to the computer.	1. Measure the voltage regulator output with a multimeter while plugged into a computer. Ensure it consistently provides 3.0 - 3.6V to all	Y

	components under multiple load conditions. 2. Use an oscilloscope to measure power draw upon startup and exit, ensuring all components maintain a stable 3.0 - 3.6 V before any data transactions begin.	
Proper ESD protection on USB Data Lines	1. Apply a controlled electrostatic discharge to the USB housing and verify that the microcontroller is functional. 2. Perform multiple USB plug/unplug cycles to confirm data lines remain stable. 3. Use an oscilloscope to monitor signals before and after the ESD diode, ensuring that signals are protected.	Y
Must be able to protect against variable changes in USB power input, as it may overvolt or draw too much current.	1. Test with a custom USB power source and introduce a small overvoltage to verify that the voltage regulator safely regulates to 3.3V. 2. Apply a load that exceeds the current draw and verify the regulator limits current appropriately.	Y