# Secure Lockbox with Binary Tree-Based Algorithmic Access

## Overview

This project is a secure lockbox that opens only after a correct sequence of button presses, following a binary tree-based authentication algorithm. The system is enhanced with LED indicators, a servo-controlled dual deadbolt locking mechanism, and reinforced with metal components using carbon fiber PLA prints. It integrates hardware and software principles from embedded systems, data structures, and physical security.

## Project Workflow

### 1. Lockbox Design and Construction

- Design and laser cut or 3D print the lockbox panels (dimensions: 9in x 7in x 9in).
- Reinforce the box with embedded steel plates or rods in the frame.
- Use carbon fiber PLA for strength and durability in printed components.

### 2. Locking Mechanism: Dual Deadbolt Rods

- Two steel or reinforced carbon fiber rods slide sideways to lock/unlock the box.
- Controlled by two servo motors for strength and redundancy.
- Rods slide into metal brackets embedded in the side walls.
- Servo horns and printed sliders push/pull the rods via guides.
- Optional: add spring-loading for fail-safe locking when unpowered.

### 3. Binary Tree-Based Input Panel

- Construct a physical binary tree with momentary pushbuttons as nodes.
- Arrange buttons in a branching layout (3-4 levels deep).
- Each press lights up the next two buttons (via LEDs).
- Only one correct path through the tree (e.g., Node 0 → 2 → 5 → 6) will unlock the safe.
- Incorrect sequence resets the input state.

### 4. Electronics and Microcontroller Setup

- Use an Arduino Nano, Uno, or ESP32 microcontroller (ESP32 recommended).
- Connect each button to a digital input pin with a pull-up or pull-down resistor.
- LEDs are connected to output pins to show progress or reset states.
- Servos connected to PWM-capable digital pins.

• Optional: add a buzzer or display for error/success feedback.
• Use clean wire routing and soldered headers for reliability.

### 5. Programming Logic

• Store the valid input sequence as an array of node/button IDs.
• Track progress using a counter.
• If the user presses the correct next button, increment the counter.
• If the sequence is complete, activate the unlock function.
• If an incorrect button is pressed, reset to the beginning and signal with LEDs.
• Debounce all inputs in code to avoid false triggers.

### Sample Arduino Pseudocode

```
const int path[] = {0, 2, 5, 6};
int currentStep = 0;

void loop() {
  for (int i = 0; i < totalButtons; i++) {
    if (digitalRead(buttonPins[i]) == HIGH) {
      if (i == path[currentStep]) {
        currentStep++;
        lightUpLED(i);
        if (currentStep == sizeof(path)/sizeof(int)) {
          unlockSafe();
        }
      } else {
        currentStep = 0;
        resetLEDs();
      }
    }
  }
}
```

### 6. Final Assembly

• Mount electronics and servo locking system securely inside the box.
• Route wires safely and test each component individually.
• Embed steel or aluminum support into critical components.
• Final test of sequence logic and mechanical locking.
• Optionally enclose electronics in a separate compartment.

### Next Steps

- Finalize microcontroller selection.
- Design and fabricate the binary tree panel.
- Prototype and test button matrix with LEDs.
- Assemble locking mechanism with metal rods.
- Test full integration of input logic and servo output.
- Ensure enclosure is tamper-resistant.

## 7. Wiring Diagrams

This section outlines the connections between the microcontroller, buttons, LEDs, and servos. Use proper wire management and color-coded wires where possible.

- Each button is connected to a digital input pin with a pull-down resistor (~10kΩ).
- Each LED is connected to a digital output pin with a current-limiting resistor (220Ω–330Ω).
- Servos are connected to PWM digital pins with separate power (5V from external supply recommended).
- All components share a common ground.

NOTE: Create the schematic using Fritzing or Tinkercad Circuits for visual reference.

## 8. Sample Arduino Code

Below is example Arduino code for the binary tree authentication sequence.

```
const int path[] = {0, 2, 5, 6};
const int pathLength = sizeof(path) / sizeof(int);
const int buttonPins[] = {2, 3, 4, 5, 6, 7, 8};
const int ledPins[] = {9, 10, 11, 12, 13, A0, A1};
int currentStep = 0;

void setup() {
 for (int i = 0; i < 7; i++) {
  pinMode(buttonPins[i], INPUT);
  pinMode(ledPins[i], OUTPUT);
 }
}
```

```
void loop() {
  for (int i = 0; i < 7; i++) {
    if (digitalRead(buttonPins[i]) == HIGH) {
      delay(50); // Debounce
      if (i == path[currentStep]) {
        digitalWrite(ledPins[i], HIGH);
        currentStep++;
        if (currentStep == pathLength) {
          unlockSafe();
        }
      } else {
        resetSystem();
      }
      while(digitalRead(buttonPins[i]) == HIGH); // Wait for release
    }
  }
}

void resetSystem() {
  currentStep = 0;
  for (int i = 0; i < 7; i++) {
    digitalWrite(ledPins[i], LOW);
  }
}

void unlockSafe() {
  // Add servo unlock logic here
}
```