I created a security device program using python that is in FSA format. I finished the first part by using a lot of if else statements, and functions to take inputs and generate the outputs. I found that it was quite difficult at first on how to make it function like an FSA, but once I figured it out, it wasn't too difficult. For part 2, my initial guess was somewhere around 6 and 3/4 or so days (1 /(0.1*0.1*0.1*0.1*0.1*0.2) = 500,000. 500,000 seconds to days is around 6 ¾ days) I figured the best way to generate random numbers was to just use the random module in python. To figure out how much time it took to brute force, I first kept track of how many guesses it took to unlock/lock. I let it brute-force it 1000 times, so I could get a good sample size. I then calculated the min, max, and average in the program itself. I then took that data and converted it to where 1 guess = 1 second. Here is the data I got:

Min is: 38
Max is: 4089595
Average is: 544495.252
Count: 1000

Minimum time: 38 seconds
Maximum time: 47 days, 7 hours, 59 minutes, 55 seconds
Average time: 6 days, 7 hours, 14 minutes, 55 seconds (Average in seconds rounded to nearest integer)

I was surprised to see that the minimum time it took was just 38 seconds! The average time was also pretty close to my prediction, which feels good.

I also added a few features so the user could choose between running it the standard way and running it where the digits are random. The user can also choose how many times to brute force the code.

For testing, I just made a unittest that makes sure that all state and single digit combinations output the correct combination, as well as other test cases such as when it is not a digit.

The regular expression is ^[0-9]$, and if you wanted to use it in python, you'd use a function like this:

```
def use_regex(text):
    pattern = re.compile(r"^[0-9]$", re.IGNORECASE)
    return pattern.match(text)
```

FSA Diagram: