Marcel Bieganski

Program written in Python 3

Security Engine:

Digits are entered one by one into the engine. After each digit is entered, the new digit is added to the current sequence of digits, which is then skimmed of non-integer values and **scanned** for either the unlock or lock code. After either of the codes are detected, the machine’s state is changed with respect to the found code.

**Scan** function:

The scan function contains two strings: a sequence for display, and a ‘functional’ sequence for parsing. The display sequence is a compilation of all entered digits which is output after each digit entered. The functional sequence contains only the digits since the last state change. Only the functional sequence is scanned by the engine for a code, and if one is found it is reset. This allows the engine to avoid complications caused by previous portions of the sequence involving the proper code that have already caused a state change.

Diagram

Description automatically generated

Brute force testing:

Playing the role of someone attempting to break into the security engine using a random method of digit inputting, we can estimate the time it would take and simulate the breaking of the engine.

Assuming the intruder knows only that the sequence contains digits 0-9, we can calculate the probability of getting the sequence entirely correct.

The probability of guessing the correct digit in a 1 number long sequence would 0.111 (or 1/9). The intruder must make that correct guess 6 times in a row, making the probability (0.9)^6.

To simulate this, a random number is generated and entered into the engine instead of a user typed digit. Each time a random digit is entered, it is compiled into a sequence with all previous digits and scanned for the unlock code. If that code is not present, another digit is generated and entered. This continues until the proper unlock code is found inside the sequence.

Sample output using 10 trials:

