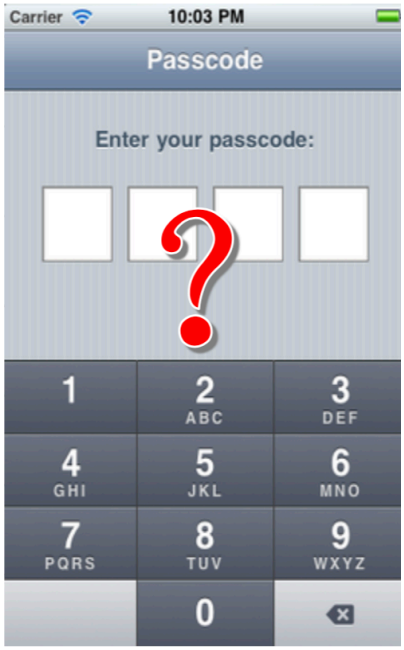
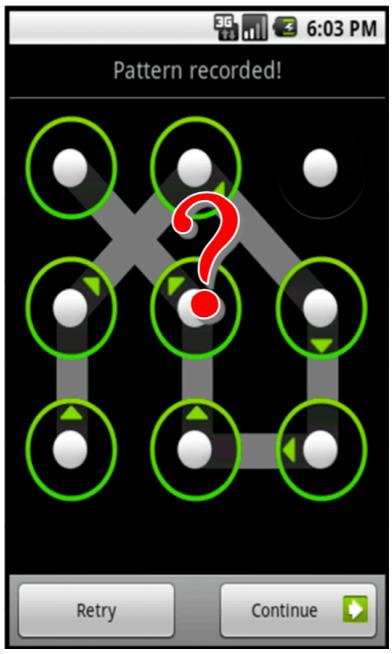
Combinatorics Project 1.
Student ID: **2018280070** Name: **Peter Garamvoelgyi**

Problem statement

In this project, our task is to compare two common mechanisms for unlocking smart phones: the 4-digit PIN (a) commonly used in Apple products, and the *swipe screen* (b) popular with Android users.



(a)



(b)

Let us first follow the assumption that the security of a login mechanism is proportional to the number of possible passcodes. This makes sense for brute-force attacks, as the attacker needs more trials to find the correct code.

Enumeration of all possibilities

For *(a)* we can directly calculate the number of possible PINs. We have 4 digits and all of them can take on values from 0 to 9, i.e. 10 distinct values:

$$\overline{10} \quad \overline{10} \quad \overline{10} \quad \overline{10}$$

Using the *multiplication principle*, the number of all possibilities is

$$N_{apple} = 10^4 = \mathbf{10,000}$$

Enumerating the number of possible swipe codes for *(b)* is trickier, as we have to follow certain constrains:

1) The passcode must contain at least 4 dots.
2) The passcode must contain distinct dots, i.e. cannot use the same dot twice. (Passing over a dot is a separate case, see 4.)
3) The line formed by the segments must be continuous (no breaks).
4) For each move, if the line segment formed by the two dots passes over a third dot, then this third dot must have been linked up previously for the move to be valid.

We could get an upper bound of the number of possibilities by enumerating all permutations of size 4, …, 9 of the numbers 1, …, 9:

$$N_{android} < \widetilde{N}_{android} = \sum_{i=4}^{9} P(9, i)$$

But many of these violate constraints 3) and 4). The best way to enumerate all valid possibilities is to write a program. The program should go through all $\widetilde{N}_{android}$ permutations and check if they satisfy the constraints. The implementation is attached as **android_code.cpp**.

Enumerating the possible swipe passcode with the above program, we find that the number of possibilities is 389,112.

$$N_{android} = \mathbf{389,112}$$

Discussion

From the above calculations, we can calculate the probabilities of a random guess being right:

$$p_{apple} = \frac{1}{10{,}000} = 0.01\% \quad p_{android} = \frac{1}{389{,}112} \approx 0.00026\%$$

This suggests that attacks for Android require more steps[1], thus it is safer.

However, given that most users choose weak passwords on both platforms, these attack probabilities are significantly higher in reality. For example, using someone's birth year or birthdate as a PIN makes it much easier to guess. Using 4-5-digit swipe passcodes also results in a much smaller number of possibilities.

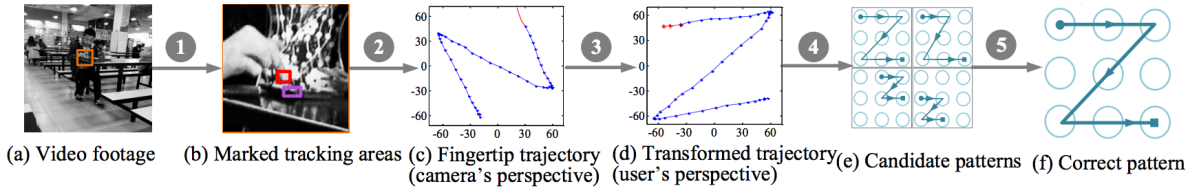
It is important to mention that recently, Apple PINs are required to have 6 digits instead of 4. With this change, the probability of a successful guess becomes

$$p'_{apple} = \frac{1}{10^6} = 0.0001\%$$

Which is on par with Android.

[1] worst-case estimation

In reality, guessing passcodes with such brute-force attacks is rarely possible, as most devices either limit the number of trials, or use time-delay between trials thus rendering such attacks prohibitively time-consuming. A hacker would probably rather revert to *social engineering*[2] or other methods. For instance, researchers from Northwest University in China have been able to reconstruct passcodes from video footages using fingertip tracking[3]:



(a) Video footage (b) Marked tracking areas (c) Fingertip trajectory (camera's perspective) (d) Transformed trajectory (user's perspective) (e) Candidate patterns (f) Correct pattern

[2] "*Social engineering, in the context of information security, refers to psychological manipulation of people into performing actions or divulging confidential information.*" Wikipedia

[3] Ye, Guixin et al. "Cracking Android Pattern Lock in Five Attempts." NDSS (2017).