Tong Wu(tw445), Xinyu Lyu(xl422), Xun Tang(xt63)

Project Pre-report:
Android Unlock Patterns Security Analysis

Motivation

Unlock gesture is a common used unlock pattern on Android smartphones. Normally, it contains 9 plots(3x3) and the user can use a specific gesture which contains some of the plots in a particular order to unlock the phone.

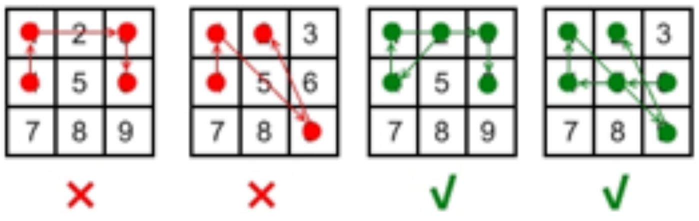
But how secure is it? Unlike the common unlock pattern using 4 digits as password, which has exactly 10000 different combinations, the security of this unlocks pattern is not sure for us. Therefore, we are going to figure out how many tries are needed when someone else wants to unlock the phone without knowing the gesture and how's the performance compared to other ways like PIN and password.

In this project, we will investigate on the Android Unlock Patterns to analyze patterns for unlocking an android smartphone and elaborate on different algorithms to compare the performance. The preliminary project plan consists these four parts:

1. Determine valid gestures;
2. Investigate algorithms that can solve the problem;
3. Compare and analyze performances of different algorithms;
4. Expand the problem to 4x4 and 5x5 situations and find the most secure gesture.

1. Determine Valid Gestures

Basically, given an Android 3x3 key lock screen, a valid unlock gesture must contain at least 2 but no more than 9 plots. The gesture must be consecutive and the keys must be distinct. In each valid gesture, the visited plot cannot be visited twice, except that it can be covered in a routine and get visited again. Additionally, every plot in the jumping curve must be marked as a visited plot. And the orders of plots used matters.We give some valid and invalid gestures below for convenience.



Invalid move: 4 - 1 - 3 - 6

Line 1 - 3 passes through key 2 which had not been selected in the pattern.

Invalid move: 4 - 1 - 9 - 2

Line 1 - 9 passes through key 5 which had not been selected in the pattern.

Valid move: 2 - 4 - 1 - 3 - 6

Line 1 - 3 is valid because it passes through key 2, which had been selected in the pattern

Valid move: 6 - 5 - 4 - 1 - 9 - 2

Line 1 - 9 is valid because it passes through key 5, which had been selected in the pattern.

2. The Available Algorithms

1. Brute Forth Mathematical Way:

The first way we come up with is using the number of all combinations minus the number of invalid gestures. We use permutation to calculate the total number of gestures that cover 4 to 9 points—permutation[4,9](A[9,4] + A[9,5] + A[9,6] + A[9,7] + A[9,8] + A[9,9]). Then we count out how many permutations are invalid. Last, we get the result by using total number minus the invalid permutations. However, this method may be complicated because we have to count out all of the invalid permutations. And it is easy to mix up and miss some invalid situations.

2. Depth First Search:

Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. We can firstly define a two-dimensional array to store the information whether there is an intermediate plot in the curve between other two plots. Then we are going to define an one-dimensional array to store that the plot has been visited or not. We attempt to visit the next plot if it has not been visited before by calling the recursive function. And if there is an intermediate plot covered by the curve, we visit it only if it has been not visited before. Last, we use symmetry to optimize the algorithm to 3 occasions: Starting from center(5), Starting from corner(1) and Starting from center of edge(2).

3. Back Tracking:

Using a hash set to store the visiting. We judge each situation if it is valid. Then we add the element into the hash set and count the gesture. Last, we go back and remove element from hash set.

4. A Smarter Way:

“used” is the 9-bit bitmask telling which keys have already been used
and (i1,j1) and (i2,j2)are the previous two key coordinates. A step is valid if:

- I % 2: It goes to a neighbor row or
- J % 2: It goes to a neighbor column or
- used2 & (1 << (I/2*3 + J/2))): The key in the middle of the step has already been used.

(i2,j2) are the coordinates of the previous key, (i,j) are the coordinates of the new key. So the new line goes from (i2,j2) to (i,j). The middle point of the line is at ((i2+i)/2, (j2+j)/2). My Iand J are those middle coordinates, except I didn't divide by 2 yet, so I can stay in integers. Now if I % 2 isn't zero, then that means I/2 and thus (i2+i)/2 is no integer. Which means the middle point of the line is not a key. Same with the other coordinate. If both those checks fail, then the middle point of the new line is a key, and thus I need to check that that key has been used already.

3. Performance Analysis

For the problem, we will implement them and make comparisons on the performance of different algorithms. Analyze the order of the growth on running cost on average, best and worst cases. And we will attempt to find the optimization methods on it to reduce the algorithm complexity.

4. Expand the problem and find the most secure gesture.

We will expand the scale of the problem to the 4X4 matrix and 5X5 matrix which need more consideration towards different scenarios, if possible.

We are also trying to provide a most secure gesture which has the longest path.