Final documentation

Lauri Kangassalo

June 16, 2013

Structure

First of all, there are two versions of this software (branches in github). Master contains a neatly written implementation, that supports passwords of varying lengths. This is the branch I want to be evaluated for the course. For the curious, the *threaded* branch contains an implementation that supports multithreading for table creation. However, this version supports only fixed-length passwords and the code is pretty ugly. You have been warned.

The program contains following packages:

hashtable A hash table implementation and all classes regarding it

helpers Helper classes that are used in many parts of the code

md5crack Classes used for creating rainbow tables and cracking hashes

The three most important classes of my project are TableCreator, MD5Crack and Reductor (a helper class), the first one generates rainbow tables, and the latter one cracks hashes, and Reductor is used by both to reduce a hash to a password.

All input/output stuff is done in the helper classes or in the class UI.

For a more detailed documentation of the classes, refer to the javadoc at: `http://www.cs.helsinki.fi/u/laka/tiralabra/javadoc/`

Performance

I have achieved my goals regarding the big O-notation.

The program generates rainbow tables in O(m*n) time, and cracks hashes in roughly O(n*n) time, where m stands for chains per table and n for chain length. The hash table implementation works in time O(1) for insert, search and contains operations. This has not been tested in any way, but should be easily readable from the code. Looping the hash table will take O(maxl,k) time, where l is the number of rows in the hash table (the number is fixed after creation) and k is the total number of nodes in the table.

How would I make the software better?

First of all, I would write the whole project in C or some other language with less overhead than Java. Secondly, I would make my program run on the GPU rather than on CPU. And finally, I would write it to support bigger rainbow tables. (i.e. tables so large, they won't fit in the memory)