****

**CE4024 Cryptography and Network Security**

**Assignment 2**

**Dictionary Attack Program Documentation**

**Submitted By: SEET YONG SONG KENNY**

**TAY JIN HENG**

**YEOH KENG WEI**

**Group: 12**

**SESSION 2013/2014**

**SEMESTER 2**

**COMPUTER ENGINEERING PROGRAMME**

**SCHOOL OF COMPUTER ENGINEERING**

**NANYANG TECHNOLOGICAL UNIVERSITY**

# Assignment Overview

In this assignment, we are tasked to perform a dictionary attack on a list of hashed salted passwords and recover the passwords corresponding to each hash. The dictionary attack involves using an exhaustive list of words, called the dictionary. The hashes of each corresponding words in the dictionary are computed and compared with a list of hashes that we are trying to crack. A password is only discovered when a collision occurs.

# 3 Phases of the Program

The program is made up of three stages. The first stage is the *Load Phase* which runs sequentially. After the *Load Phase*, the *Password Transform Phase* and the *Password Validation Phase* runs concurrently via multi-threading.

# Load Phase

The *Load Phase* is the first stage of the program, in charge of loading all the words in the dictionary from a .txt file into the program. It also loads the list of hashes to crack from a .txt file as well.

During the Load Phase, all uppercase letters were transformed to lowercase and all words which contained symbols, (e.g*. $, #, @* etc.) were omitted. This is due to the passwords specifications used in this assignment is only made up of only lower case letters (*a-z*) and digits (*0-9*). These words then go through various transforms, processed by the *Transformation Thread*.

# Password Transform Phase

*Password Transform Phase* takes inputs from the words in dict.txt and outputs a new transformed word, which will be used as input for the *Password Validation Phase*. The transformation is processed by the *Transformation Thread*. The transformation pattern found in the passwords of hash.txt is as follows:

1. No Transformation
2. Character Substitution (Leet speak)
   1. ‘i’ – ‘1’
   2. ‘e’ – ‘3’
   3. ‘a’ – ‘4’
   4. ‘s’ – ‘5’
   5. ‘t’ – ‘7’
   6. ‘o’ – ‘0’
3. 1 or 2 digit suffix/prefix
   1. Digits from (0-99)
4. 2-words combination
5. 1 digit suffix and word combination (double transform)

The output of each transformation are queued into 5 separate queues, according to the transformation subjected. The outputs in the queues will be used as inputs for the *Validation Threads* in the *Password Validation Phase*. This is illustrated in the diagram below.

# 

# Password Validation Phase

In the *Password Validation Phase*, 4 threads (or more depending on the available CPU processors, 4 being the minimum as 4-core CPU will be used stated in the specifications) will be spawned to consume input from the queues. Each thread will be assigned their own input queue. When the threads finished processing their own input queue, they will “help” the other threads by processing their input queues.

In each validation thread, the salt of each un-cracked password hash are used with each word to generate the corresponding MD5 hash. The MD5 hash is then compared with the un-cracked password hash. If they are not the same string, the process is continued with the same word and another un-cracked password hash. Once a collision occurs (both hashes match each other), the username and password are printed out and the corresponding hash is removed from the un-cracked password hash list.

This process is continued with all the words generated by the *Password Transformation Phase*. Eventually, the program stops when all the hashes are cracked or if there are no more inputs.