A Distributed Algorithm for Brute Force Password Cracking on n Processors

Roman Bahadursingh
bahadursingh4@yahoo.com

Abstract

A Password **H**, can be defined as a hash of x symbols .A brute force password cracking algorithm will go through every possible combination of symbols from 1 – x symbols. This form of password cracker takes O(l(**H**)) time to solve, where n is the number of possible combinations, achieved by s^x where s is the number of symbols available for a password. Having a password cracker with multiple processors, having the processors instead of all checking from symbol 0 to the last symbol, using a more decentralized approach can greatly improve the speed of this computation, letting the original algorithm having a speed of O (n) to O(n/2) for two processors, O(n/3) for three processors and $O(n/n_p)$ as a generalized formula. This algorithm also allows for multiple processors of different clock speeds to also crack a password in more optimal time.

1 Introduction

The Brute Force Password Cracking Algorithm (BFP from now on for simplicity), uses an approach to crack a password, in which, it will check from the first available symbol and continue computation until it has reached the password. BFP has a computation time of O(n) and is greatly inefficient when multiple processors are being used. When multiple processors are being used, it will start from the first symbol and continue to the last with each processor adding diminishing returns in computations time, as each must check the other processors, fetch the current password value and then create their own and continue this cycle until the password is found. BFP, however can be optimized by the Generalized Brute Force Password Cracking Algorithm (GBFP).

1.1 Generalized Brute Force Password Cracking Algorithm (GBFP)

To calculate the password, GBFP will begin by having the processors organized by processor clock speed, with the most powerful being named Processor 1, Processor 2 for the second – most powerful, etc. If two processors are the same clock speed, they will be named randomly. First, the algorithm will take all of the available symbols in an order (0 - %). GBFP will then assume that there are at most five symbols which create **H**. Then, for p processors, **H** will be divided into p password ranges to be checked. Processor 1, will then begin at the first symbol of the password, and if in a two processor machine, processor 2 will begin halfway through the available symbols **H** could be and the algorithm will then begin BFP, with different starting points for both processors and the ending points begin either the end of the l(**H**) or the starting point of processor p + 1, whichever comes first. After all of the symbols within a processors range have been exhausted, the processor shall begin from %%%%% and depending on the number of processors in the system, check from here, to $1/n_p$ the amount of available symbols. When processor is finished it also moves to this password space and checks the end of p -1's password range and continues to the beginning of processor p + 1's password range. This algorithm continues until quit or until a password is found.

1.2 Computational Complexity Comparisons

BFP takes O(n) time to be completed and to find the hash, with diminishing returns for each processor added to the system running BFP, as each processor must take the current password values of every other processor and then continue forward with this. GBFP with multiple processors of the same clock speed take $O(n/n_p)$ time to be completed as each processor added performs its own calculations and letting r_p be the password range (range of possible symbols to be checked by a processor) of processor p, when $r_p \cap r_{p+1}$, it moves on to a different set of symbols to be checked, effectively optimizing BFP for n processors.

2 Glossary of Symbols

l – Length Function

H – Password Hash

n – l(**H**)

n_p – Number of Processors

p – Processor Number

s – Number of Available Symbols

BFP - Brute Force Password Cracking Algorithm

GBFP - Generalized Brute Force Password Cracking Algorithm

References

1. S. Arora,B.Barak.,"Computational Complexity: A Modern Approach," 2017

2.A.P.Black.,"Exhaustive Search Algothms",2019