Rodney Beede  
Neelam Agrawal

Yogesh Virkar

CSCI5576 – High Perf Sci Computing

Due: 2011-02-23

**Project Proposal – Parallel Zip Archive Password Recovery (P-ZAPR)**

Our project will focus on the problem of recovery of data stored in encrypted zip archive files for which the password has been forgotten. When a user forgets a password to their zip archive the solutions available are usually limited to brute force attempts to guess the correct password. The number of possible passwords makes brute force a very time consuming task. Our project will provide a parallel solution that can utilize large clusters of computing power to attempt to find the password in a shorter time versus trying one at a time in a serial fashion.

There are businesses that provide a password recovery service using their own propriety solutions for a fee (pwfinder.com). They typically use large dictionaries of possible passwords and attempt brute force methods of finding the correct password. Our project will provide an open source solution to this problem and demonstrate the importance of having strong passwords for protecting data. In addition we will be able to provide benchmarks and formulas for estimating the amount of time it would take to discover a password by brute force based on the password length and the number of nodes in a cluster.

The zip format (aka PKzip) has an older encryption scheme known as the PKZIP stream cipher which has been defeated with a technique which uses a small amount of known plain-text to decipher the encrypted zip archive (Biham, 1994). It has been shown that this type of encryption can be easily cracked in a serial manner on a single node machine.

Our interest will be in using brute force techniques of guessing passwords without knowing any plain-text to decipher a zip archive compressed with the newer AES-256 algorithm. Due to the way the AES-256 algorithm is used in the zip format it will not be possible to pre-compute large rainbow tables (aka lookup tables) of possible passwords (APPNOTE.TXT - .ZIP File Format Specification, 2007) therefore our implementation will have to rely on attempting to decrypt the archive with a large amount of passwords at run-time. We do plan on, however, only attempting to decrypt the first few kilobytes of a giving archive to speed up the process.

The program will take a given zip archive as input data along with either a dictionary of possible passwords or a list of all possible case sensitive alphanumeric passwords of a given length.

The dictionary will be taken from a combination of online sources and auto-generated variants of words. The size of the dictionary will be in the hundreds of thousands of words, but it will only have to be generated one time.

The alphanumeric password list will be generated based on 62 possible characters (a-z, A-Z, 0-9) with varying length of 1 to 7 characters for a total of 3,579,345,993,194 possible combinations. The goal is to attempt to achieve 41,427,616 passwords per second across all nodes on a cluster such that the entire key space could be attempted in 24 hours maximum. This password type was chosen because it represents a common space for passwords chosen by users and the zip archive doesn’t enforce any particular password length or complexity requirements. If this number proves unfeasible we can revert to shorter lengths.

We will benchmark the number of attempts per second for a varying amount of nodes. In addition we will time how long it takes to find a correct solution based on varying password length. We will include tests against zip archives with known solutions in the dictionary as well as tests against zip archives that do not have a known solution in the dictionary for recording times. With this data we will be able to derive a formula for how long it would be expected to discover a password or arrive at no solution for a password based on the number of nodes and hardware along with the password space size.

We will also investigate ways to parallelize the AES-256 algorithm for a single password attempt and whether doing so would provide any advantage over a serial implementation of AES-256. The division of dictionary words or alphanumeric passwords across nodes will be the primary aspect of the parallelization for this project.

# References

(n.d.). Retrieved February 15, 2011, from pwfinder.com: http://www.pwfinder.com/

*APPNOTE.TXT - .ZIP File Format Specification*. (2007, September 28). Retrieved February 15, 2011, from pkware.com: http://www.pkware.com/documents/casestudies/APPNOTE.TXT

Biham, E. a. (1994, December). A Known Plaintext Attack on the PKZIP Stream Cipher. Fast Software Encryption 2, Proceedings of the Leuven Workshop, LNCS 1008.