**KANTIPUR ENGINEERING COLLEGE**

**(Affiliated to Tribhuvan University)**

**Dhapakhel, Lalitpur**



**[Subject Code: CT…]**

**A MAJOR PROJECT PROPOSALON**

**“Distributed Password Cracking System”**

**Submitted by**

**Bhargab Acharya [BCT-54-2070]**

**Subek Acharya [BCT-83-2070]**

**Nischal Khadka [BCT-63-2070]**

**Anurag Poudel [BCT-49-2070]**

**Submitted to**

**Department of Computer and Electronics Engineering**

**December, 2016**

**Distributed Password Cracking System**

**Submitted by**

**Bhargab Acharya [BCT-54-2070]**

**Member 2 [BCT-83-2070]**

**Member 3 [BCT]**

**Member 4 [Roll No.]**

**Submitted to**

**Department of Computer and Electronics Engineering**

**Kantipur Engineering College**

**Dhapakhel, Lalitpur**

**Month, Year**

# ABSTRACT

In every secure system passwords are stored as hash values. Plain text passwords are passed to a hashing function and resulting hash values are stored. Plain text can be easily converted into hash values but hash values cannot be converted to its original text. But, what can be done is hash every possible combinations of text and check each one of them. So a good hash function is very complex and takes very long time to crack. But, if we can divide this work among network and try to crack it in parallel it would save lot of time.

**Key Words:** Hashing, Parallel Computing, General Purpose GPU, OpenCL, Brute Force, Client, Server, Worker

TABLE OF CONTENTS

[ABSTRACT i](#_Toc467032598)

[LIST OF FIGURES iv](#_Toc467032599)

[LIST OF TABLES v](#_Toc467032600)

[CHAPTER 1: INTRODUCTION 1](#_Toc467032601)

[1.1 Background 1](#_Toc467032602)

[1.2 Problem Statement 1](#_Toc467032603)

[1.3 Objectives 1](#_Toc467032604)

[1.4 Applications 1](#_Toc467032605)

[1.5 Project Features 1](#_Toc467032606)

[1.6 Feasibility Analysis 1](#_Toc467032607)

[1.6.1 Economic Feasibility 1](#_Toc467032608)

[1.6.2 Technical Feasibility 2](#_Toc467032609)

[1.6.3 Operational Feasibility 2](#_Toc467032610)

[1.7 System Requirement 2](#_Toc467032611)

[1.7.1 Software Requirement 2](#_Toc467032612)

[1.7.2 Hardware Requirement 2](#_Toc467032613)

[CHAPTER 2: LITERATURE REVIEW 3](#_Toc467032614)

[2.1 ………Topics as per ur project 3](#_Toc467032615)

[2.1.1 Subtopics as required 3](#_Toc467032616)

[2.1.2 ……… 3](#_Toc467032617)

[2.2 ……… 3](#_Toc467032618)

[2.2.1 ……… 3](#_Toc467032619)

[2.2.2 ……… 3](#_Toc467032620)

[CHAPTER 3: METHODOLOGY 4](#_Toc467032621)

[3.1 Topics as per ur project……… 4](#_Toc467032622)

[3.1.1 Subtopics as required 4](#_Toc467032623)

[3.1.2 ……… 4](#_Toc467032624)

[3.2 ……… 4](#_Toc467032625)

[3.2.1 ……… 4](#_Toc467032626)

[3.2.2 ……… 4](#_Toc467032627)

[CHAPTER 4: EPILOGUE 5](#_Toc467032628)

[4.1 Expected Output 5](#_Toc467032629)

[4.2 Budget Analysis 5](#_Toc467032630)

[4.3 Work Schedule 5](#_Toc467032631)

[REFERENCES 6](#_Toc467032632)

[APPENDICES 7](#_Toc467032633)

# LIST OF FIGURES

**FIGURE PAGE**

Figure 1.1: -------------------------------------------------------------------------- 2

Figure 1.2: --------------------------------------------------------------------------- 3

Figure 2.1: --------------------------------------------------------------------------- 7

….

# LIST OF TABLES

**TABLE PAGE**

Table 1.1: ---------------------------------------------------------------------------- 2

Table 1.2: ---------------------------------------------------------------------------- 3

Table 2.1: ---------------------------------------------------------------------------- 7

# INTRODUCTION

## Background

Every secure systems store only the hash of a user’s password. When you type your password to login, the system hashes the password you type and compares it against the stored hash value. In this way, an attacker who takes over a computer and steals the hashed password still cannot get real password. But what s/he can do is generate every possible combination of characters, hash them and check against the list of hashed password s/he got. This would still take thousands of years if password length is 9,10 characters long if it is done sequentially.

## Problem Statement

With today’s (drastically) faster computers, incredibly optimized password cracking code can attempt over three million hashes per second on a single core. While the hash function is still one-way, we can ﬁnd a lot of passwords using a brute-force approach: generate a string, hash it, and see if the result matches the hash of the password we’re trying to ﬁnd. Still, three million per second isn’t all that fast. Using just the upper, lowercase, and numeric characters, there are 628 possible eight character passwords. That would take 842 days to crack using a single core. Using GPU we can compute 2 billion hashes per second. Using clusters of multiple GPUs we can increase this number upto hundreds of billions. But why stop there? There must be lots of idle computer sitting around in internet. Lot of them have GPU(s).

## Objectives

1. To optimize Brute Force attack
2. To use/learn extreme parallel computing power of GPU
3. To use/learn concept of distributed systems

## Applications

* Penetration Testing
* Password Recovery

## Project Features

* Client-Server-Worker Architecture
* Use of GPU for parallel computation
* Supports multiple vendors of GPU/CPUs

## Feasibility Analysis

### Economic Feasibility

….

### Technical Feasibility

….

### Operational Feasibility

……

## System Requirement

### Software Requirement

* Java Runtime Environment

### Hardware Requirement

* Any processor which supports OpenCL (List of supported hardwares: <http://www.khronos.org/conformance/adopters/conformant-products#opencl>)

# LITERATURE REVIEW

## Cryptography

### Password Cracking

[1] In cryptanalysis and computer security, password cracking is the process of recovering passwords from data that have been stored in or transmitted by a computer system. A common approach (brute-force attack) is to try guesses repeatedly for the password and check them against an available cryptographic hash of the password.

The purpose of password cracking might be to help a user recover a forgotten password (installing an entirely new password is less of a security risk, but it involves System Administration privileges), to gain unauthorized access to a system, or as a preventive measure by system administrators to check for easily crackable passwords. On a file-by-file basis, password cracking is utilized to gain access to digital evidence for which a judge has allowed access but the particular file's access is restricted

…

### Hashing vs Encryption

[2] A hash function is any function that can be used to map data of arbitrary size to data of fixed size. The values returned by a hash function are called hash values, hash codes, digests, hash sums, or simply hashes. They are also useful in cryptography. A cryptographic hash function allows one to easily verify that some input data maps to a given hash value, but if the input data is unknown, it is deliberately difficult to reconstruct it (or equivalent alternatives) by knowing the stored hash value. This is used for assuring integrity of transmitted data, storing passwords, rapid data lookup etc.

Encryption in other hand can be decrypted using a key. Encryption is used to securely send data across untrusted network.

## Parallel Computing

### CPU vs GPU

CPU is a sequential device i.e it does all of its works one instruction at a time. It is great for normal computing but for graphics rendering same instructions has to be done over and over again. GPU is a dedicated processor for doing same instructions many times on different data sets. GPU was originally designed for 3D game rendering but it can also be used to do general purpose tasks other than graphics. CPUs are very fast(usually 1-2 GHz) where as GPU clock rates are in MHz. But CPU has only few cores unlike GPU which has about 100 cores.

### CUDA vs OpenCL for GPGPU

General purpose use of GPUs are increasing. Deep Learning, financial modelling, cutting-edge scientific research, oil and gas exploration, data centres etc are few such examples. So modern GPUs are programmable to function as general purpose GPU (GPGPU). CUDA is a parallel computing platform and programming model invented by NVIDIA. It works only in NVIDIA but better optimized and mature than OpenCL.

OpenCL in other hand is vendor independent. Both CUDA and OpenCL are written in their respective C like language.

### Lightweight Java Game Library (LWJGL)

[3] LWJGL is a Java library that enables cross-platform access to popular native APIs useful in the development of graphics (OpenGL), audio (OpenAL) and parallel computing (OpenCL) applications. This access is direct and high-performance, yet also wrapped in a type-safe and user-friendly layer, appropriate for the Java ecosystem. We write OpenCL code and compile and load it to GPU using LWJGL.

# METHODOLOGY

## Concept

C:\Users\Bhargab\Downloads\Untitled Diagram.png

Client requests server to crack some password and send it the hash value. Server checks its online workers and divide them work. It sends them hash value, starting string and end string. Workers may have NVIDIA , AMD or no GPU at all. They get their respective share of work. If a worker find a match, it informs server and server stops all running workers.

## Technologies To Use

Since we are targeting multiple computers across the network, we will use OpenCL rather than CUDA in worker application. We are using LWJGL library as a java wrapper because Java cannot directly access GPU. For Server and Client side applications, we use pure Java.

## Implementation

When a worker gets its allocated job, it further divides it into its GPU/CPU cores. String is generated in GPU itself because generating string in CPU and feeding it to GPU is too expensive and causes bottleneck. This whole algorithm should be implemented in OpenCL C.

### …

Include system developments, system designs, block diagrams, other diagrams and materials that explain about working principles of ur project…

# EPILOGUE

## Expected Output

………what do you expect after completion of project… you may include diagrams or tables etc………….

## Budget Analysis

………cost estimation of ur project development and running………….

## Work Schedule

……what u plan to do when… include gantt chart…………….

Figure .1: Gantt Chart

# BIBILOGRAPHY

[1]"Password cracking", *En.wikipedia.org*, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Password\_cracking. [Accessed: 05- Dec- 2016].

[2]"Hash function", *En.wikipedia.org*, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Hash\_function. [Accessed: 05- Dec- 2016].

[3]"LWJGL - Lightweight Java Game Library", *Lwjgl.org*, 2016. [Online]. Available: https://www.lwjgl.org/. [Accessed: 05- Dec- 2016].

# 

# APPENDICES

<Other contents that you want ur readers to read ….. >