Proc. of the Second Intl. Conf. on Advances in Electronics, Electrical and Computer Engineering -- EEC 2013

ISBN: 978-981-07-6935-2

Cracking MD5 hashes by simultaneous usage of Multiple GPUs and CPUs over multiple machines in a network

Jega Anish Dev

***Abstract*—Cryptographic Hash functions find ubiquitous use in various applications like digital signatures, message authentication codes and other forms of security. Their associated vulnerabilities therefore make them a prevalent target for cyber criminals. General means of defeating hash based authentication involves both discovering and exploiting inherent weakness in cryptographic hash functions or by straight forward brute force attack on the hash Digest. The latter in most cases presents to be an extremely time intensive task. Recent times have however seen usage of GPUs for brute forcing hashes thus significantly accelerating the rate of brute forced hashes. This has further been extended to use simultaneous usage of multiple GPUs over multiple machines. Usage of CPUs for brute forcing have more or less lost importance over the advent of GPU based cracking. This paper presents an efficient method of maximally using computing resources by simultaneously using Graphics Processing Units (GPUs) and Central Processing Units (CPUs) of machines over a network and compares its efficiency against using networked GPUs alone. This also discusses the possibility of currently unseen exploitation of computing power of botnets.**

***Keywords*— MD5 Hash cracking, CUDA based cracking, combined usage of CPU and GPU for hash cracking, Botnet computing power**

I. Introduction

Cryptographic hash functions find its uses in most of the security related fields involved in day to day life. They are designed to take a string of any length as input and produce a fixed-length hash value called the Digest. Digital finger printing, verifying authentication of messages/data and password verification are some of the most common uses of the cryptographic hashes. The three main requirements of a hash function are Pre-image resistance, Second pre-image resistance and Collision resistance. These features prevent a malicious adversary from being able to replace or modify the input data without changing its Digest. Exploit based attacks involve defeating one or more of the 3 resistance features of the algorithm. These exploits are considered to be the greatest vulnerabilities of a hashing algorithm as opposed to brute forcing which almost completely depended on the power of machines used. Brute forcing involves generating a look up table on the fly consisting of all possible strings, referred to as Messages, and their hashed equivalents. The hash required to be broken is compared with the hashes generated on the fly. A generated hash found equal to the hash to be cracked indicates discovery of the Message. Usage of CPUs alone was considered to be largely inefficient as this would yield not more than a dozen million hashes per second. CUDA technology granted access to hundreds of cores within the GPU and enabled generation of approximately 300 to 400 million hashes per second. This figure grew manifold with the GPUs being used in parallel across a network. Modern day high performance desktop GPUs can generate around a billion hashes a second.

This paper aims at harnessing the combined power of modern multi core CPUs along with GPUs over multiple computers in a network. This also aims to present a decentralized solution to integrating multiple computers to work on concurrently breaking a hash. This is demonstrated in breaking an MD5 hash Digest. The proposed concept can be divided into 2 modules:

1. Platform Frontend
2. Brute Forcer Backend

The rest of the paper is organized as- Section II discusses the current standards in GPU based hash password cracking. Section III describes the proposed methodology. Section IV analyses the performance results obtained and discusses its scalability. Section V concludes the paper with future work and also discusses possible malicious uses of large scale distributed CPU and GPU computing power.

II. Literature Survey

From the references [1] to [6], I arrived at various methods of using CUDA to brute force hash Digests both by single and multiple GPUs. This also showed a number of ways to distribute the cracking Job amongst the nodes involved in cracking the hash Digest

David Apostal et al 2012, [1] suggested an HPC (high performance computing framework), MPI (message passing interface) to minimize the amount of latency and handle communication between multiple GPUs. This allows for a course-grained division of the Job using MPI where each device applies a fine-grained division of the problem using CUDA to perform the actual calculations to brute forcing hashes. The paper also specifies 3 dictionary based password recovery algorithms.

Tomosuke Murakami et al, 2010, [2] describe performance of GPGPU (general purpose computing on graphics processing unit) to parallelize cryptographical hash processing

Jega Anish Dev
Department of Computer Science and Engineering,
College of Engineering, Guindy. Anna University
India

Proc. of the Second Intl. Conf. on Advances in Electronics, Electrical and Computer Engineering -- EEC 2013

ISBN: 978-981-07-6935-2

of a password cracking tool, John the Ripper. The paper describes parallelizing the 3 modes supported by John The ripper, “Singe mode”, “Word list mode” and “incremental mode”. The paper specifically exemplifies cracking MD5 hashes

Anh-Duy Vu et al, 2011, [3] described a homogeneous parallel brute force cracking algorithm that performs all the works on the GPU side.

BarsWF, (3.14.by/en/md5), [4] is currently the most efficient and fastest MD5 hash brute forcer by Svarichevsky Mikhail. It utilizes both the CPU and GPU in a machine to brute force a hash Digest. A modified version of this is used as the Brute Forcer Backend module.

Feng Wang et al, 2012, [5] explore constant memory in CUDA architecture and achieve 44.6% improvement by allocating constant memory to a padding array. It presents a highly scalable implementation of Brute Force Attack Algorithm of MD5 Crypt on Tianhe-1A.

Duc H. Nguyen et al, [6] discuss an approach using a cluster of modern multi-core graphic processing units (GPU Cluster) as computing devices for recovering lost passwords of MD5. The paper also discusses the rate of increase of hashes generated per second with addition of more GPU devices.

Adrew D. Zonenberg, [7] discusses distributed hash cracking system capable of running on all major platforms, using nVidia GPU acceleration where available. The cracker is modular and allows precompiled hash algorithms to be added with no modification to the existing application binaries, in order to add support for new algorithms or make use of hardware acceleration.

[8] States the number of computers constituting the largest botnets ever recorded.

[9] States the number of computers in an average botnet. This data is used to theorize the potential average computing power provided by a botnet.

III. Proposed Work

This section presents the methodology proposed to setup a decentralized network of computer nodes whose combined computing powers of both CPUs and GPUs can be used for cracking a hash. From hence forth, all references to cracking a hash are implied to use both the CPU and GPU. All references to cracking a hash Digest by splitting up the work among several computers are called Jobs. A Job pool is referred to as the total number of hashes required to be tested before the hash Digest has been cracked.

A. *Platform Frontend*

The main aim of this platform is to provide a stable platform on which the combined computing powers of all participating machines can be pooled together and be made ready for simultaneous use. The outcome of this module is essentially a single program that can act at as both a client and a server in a network of available machines. It is deployed on machines that are required to be added to the computing pool. It can work as a client when a Job is initiated by another node. In this case, the program receives the hash Digest, the Minimum Key Length, Character Set used and its quota of the Job pool and then begins brute forcing the Digest according to the quota received. If the program works as a server, it first receives from the user: the hash Digest to be cracked, the minimum key length for the brute Forcer to start working on and the character set used in the brute force. It then distributes the Job pool among the available machines and also to itself. This system ensures the availability of nodes that are on the network and ranks them based on their processing power taking into consideration the combined strengths of their GPUs and CPUs. It splits the Job in such a way that the machines with greater computing power receive larger Job pools as compared to the machines of intermediate power. The ranking of a node is based on the total number of hashes it can generate per second. Once the Job has been deployed to the computing pool, each node brute forces the Digest as per the Job Split and individually attempts to find out the Message. On successful discovery of the Message, the computer that finds it informs other computers about the discovery and the brute forcing stops. It then sends the Message to the machine acting as the server and displays it to the user initiating the Job. The overall architecture is given in Fig. 1.

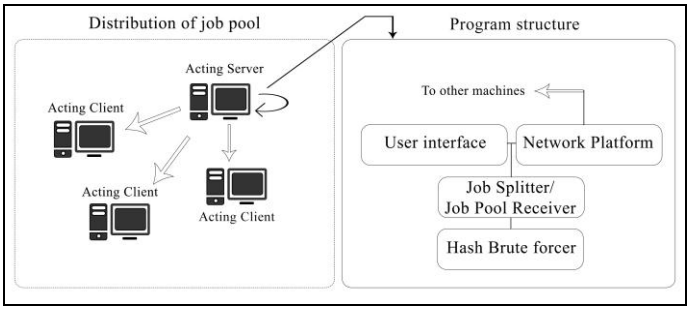


Fig. 1. Proposed Architecture

1) Network Sub Module

Multicast sockets are used instead of traditional sockets. These sockets so do not transmit messages on single channels to the recipient machines. Instead, they broadcast messages to all the computers in a particular “Multicast Group”. All program instances share a common “Multicast Group” and therefore, all messages from the programs in a network reach every other instance of the program. This is to avoid user given details such as Server IP address and so on. The general aim is to make all nodes mutually aware of each other and to be able to automatically detect nodes spontaneously leaving/dropping from the network. This is achieved by usage of a repeatedly pulsing message dubbed “I’m here” and the usage of countdown timers. The “I’m here” message is continuously pulsed in a time period of one second by each and every node. The message consists of the sender’s IP Address, Information detailing its hardware power and state of program, i.e. whether it is currently running a Job or not. Since this message is broadcasted, every node receives this message from every other node. Other instances of the program receive the message and finds out if it has previously received the

Proc. of the Second Intl. Conf. on Advances in Electronics, Electrical and Computer Engineering -- EEC 2013

ISBN: 978-981-07-6935-2

message from the sender. In case of a new sender, the sender is registered as a new computing element and a countdown timer linked to that particular node is initialized. An appropriate time-out period, preferably of 2 seconds is set for this timer. Upon timeout, it is programmed to remove the computing node associated with it and the timer then disposes itself. In case of an already existing sender, the countdown timer associated with that particular sender is looked up and then reset. This system enables all nodes in a network to be simultaneously aware of the presence of every other node in the network. A figurative diagram is shown in figure 2 and 3

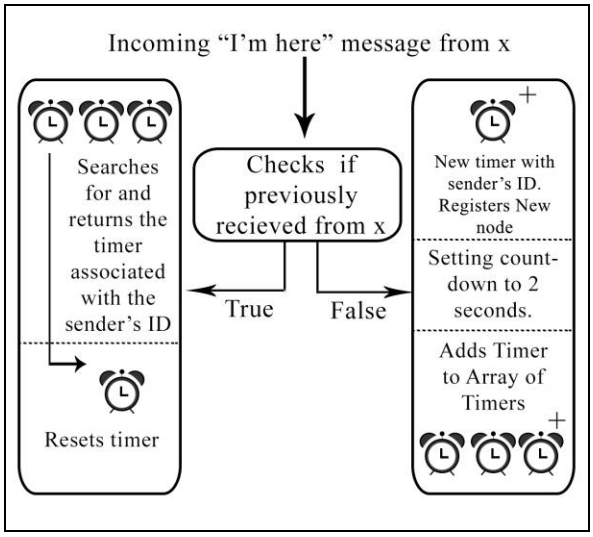


Figure 2: System keeping track of availability of nodes – Handling "I'm here message"

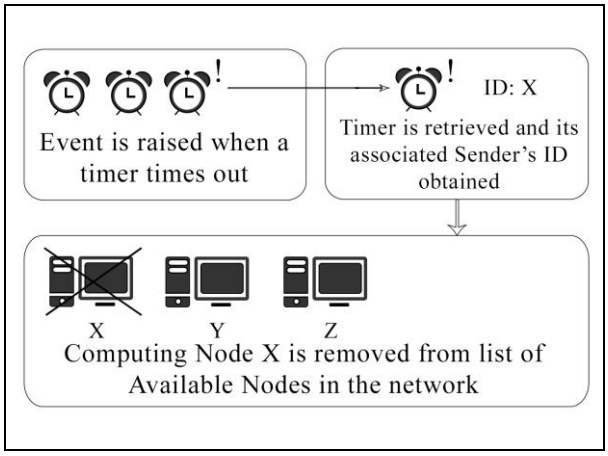
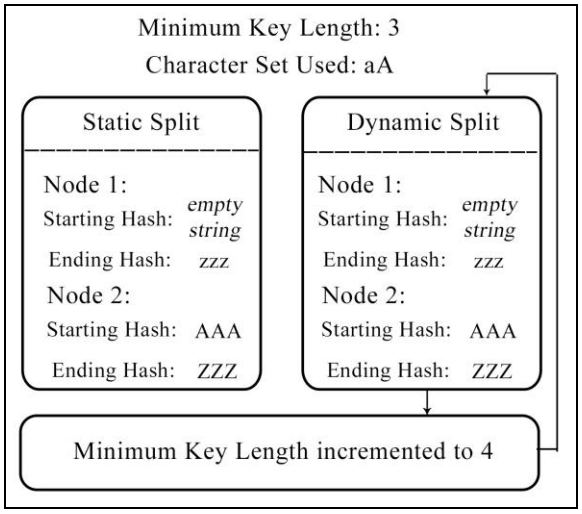


Figure 2: System keeping track of availability of nodes – Handling countdown Timers' Timeouts.

2) Job Splitter/Job Receiver

As previously mentioned, a Job is referred to as the process of cracking a hash Digest. Since the Job is split among the available computing nodes, the split is required to be equally distributed among them taking into consideration the difference in their computing powers. The basic parameters in determining the Job size are the minimum Key Length of the brute Forcer and the Character Set used. Furthermore, the split is decided to be either a Static or a Dynamic split. A static split is when the maximum key length is specified. In this case, the total Job size is a definite quantity and the Job can be directly split among the available nodes. A dynamic split is when the maximum Key Length is not specified. In this case, a temporary maximum key length is assumed and the split is carried out as usual. When each node reaches the end of the quotas of their Jobs, they assume a new incremented maximum Key Length. They then compute the next quota of their Jobs and begin brute forcing again. Examples of both splits are given in Figure 3.



3) User Interface

The User Interface of the program presents an integrated system for the user to be able to both prepare Jobs for processing or to view details on any other currently running Job requested by another computer. Parameters for a new Job can be altered before beginning Job processing. Once a Job is in progress, the state of that particular node is set to "Busy". This is to notify other nodes of its unavailability. These Nodes may have joined the network after a currently running Job has started processing. Nodes which may have joined after a Job has begun can start processing a separate Job amongst them.

4) Configuration File Editor

The brute Forcer in each node has a configuration file from which it reads the parameters: Key length, Character Set and Character Index. The Character Index is a numeric field which one uses to specify the starting Key to the Brute Forcer. This is used by the Job Splitter to specify the range of the Job Size for each node to Brute Force.

5) Brute Forcer backend

As aforementioned, The Brute Forcer back end is a modified version of BarsWF [4]. The modification involves stopping the Job at the end of a specified range instead of auto incrementing the Key Length.

The Brute Forcer is executed by the Platform Frontend upon reception of Job details and the Job split for the particular Node. It then reads the configuration file as written

Proc. of the Second Intl. Conf. on Advances in Electronics, Electrical and Computer Engineering -- EEC 2013

ISBN: 978-981-07-6935-2

by the Platform Frontend, taking in details of the starting Character Key Index, Minimum Key Length and Character Set used. If the split is static, the process goes on till either the key is found or the quota of Job assigned to the node is complete. In case of dynamic split, the Job process goes on perpetually until the Message is found or until the Job is cancelled. A diagrammatic representation of interaction between the Frontend and the Brute Forcer back end is shown in Figure 4.

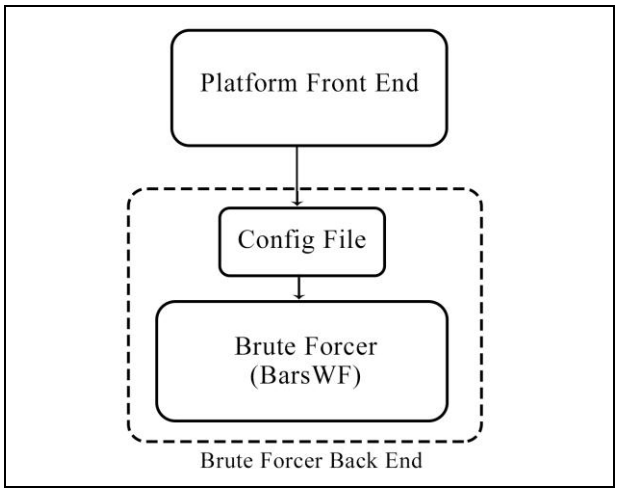


Figure 4. Interaction of Platform Frontend and Brute Forcer Backend

Once the Message is found, the Backend reports the Message to the Frontend which forwards it to the Acting Server. All Brute Forcers in other machines are ceased by their respective Frontends upon discovery of the Message.

IV. Results

Initially, the user has to deploy the program on networked computers. Once running, the program automatically detects all other machines running the program and registers all active Nodes. In case of a Node's resignation from the network, every other Node becomes immediately aware of the drop out and removes the computer from its list of available computers.

The user can use any one of the computers to specify Job details and its associated parameters. The user's computer becomes the Acting Server and distributes the Job details along with the Job split to each of the available computers in the network including itself. Computers with greater computing power are given a larger Job Split. The process goes on until the Message is discovered by a Node or until the user cancels the Process.

Tests conducted on a single machine using BarsWF yielded hash generation rates shown in Table 1. Specifications of a Test Machine used are shown in Table 2.

When networked with 2 computers of same specification, the total hash generation rate, as expected, was found to be approximately twice times the rate obtained on a single machine.

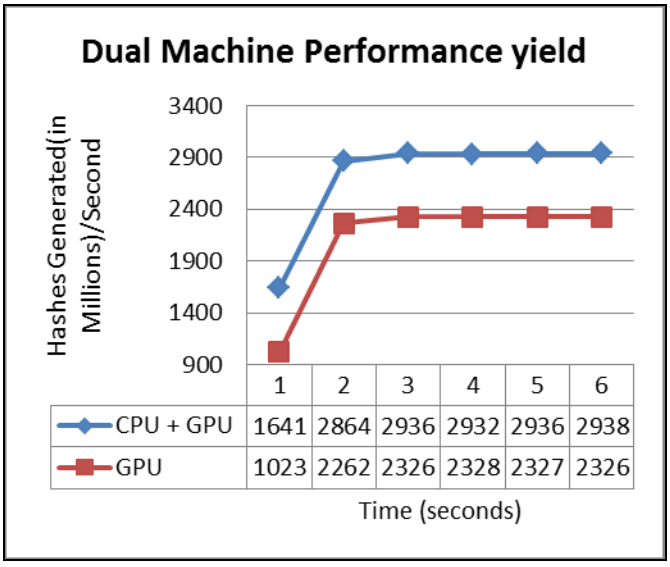
Most importantly, simultaneous usage of multiple CPUs combined with GPUs over multiple machines in a network was found to generate 20.79% more hashes than when tested by conventional use of networked GPUs alone.

TABLE I: TEST COMPUTER SPECIFICATIONS

Specification	Detail
Processor	Intel I7-2600K @ 3.3 GHz 4 Cores 8 logical processors
GPU	NVIDIA GTX 550 TI CUDA Cores: 192 1024MB GDDR5 Graphics Clock (MHz): 900 Processor Clock (MHz): 1800
RAM	8 GB DDR3
Operating System	Microsoft Windows 7 Professional

TABLE II: DETAILED SINGLE MACHINE PERFORMANCE

Computing Device	Hashes Generated (in Million hashes/Second)
CPU0: CPU1: CPU2: CPU3: CPU4: CPU5: CPU6: CPU7: CPU Total:	38.82 37.58 38.01 38.24 38.11 38.43 38.28 38.03 305.50
GPU	1163.16
Combined Total:	1468.66



Graph 1: Overall Hash Rate Generation using 2 Test Machines

Proc. of the Second Intl. Conf. on Advances in Electronics, Electrical and Computer Engineering -- EEC 2013

ISBN: 978-981-07-6935-2

V. CONCLUSION AND FUTURE WORK

The 20.79% speed boost in hash brute forcing is a sizeable amount considering that the CPU is generally much less expensive than a specialized gaming graphics card. Moreover, performance graphics cards are used only by gaming enthusiasts are not very widely used as compared to Intel I7 and I5 processors which are standard specifications of the general computer. Furthermore, building custom machines having multiple graphics cards, say more than two, involve critical technical know-how and large amounts of funds for purchase and construction of the equipment.

This establishes the fact that simultaneous usage of CPUs and GPUs across a network is not only more simpler, requiring absolutely no custom equipment, but is also more easily obtainable as one need not specifically buy extra specialized graphics cards but could just as well rent or use multiple standard office or home machines having standard graphics cards coupled with intermediately powerful CPUs to achieve the same or greater rates of hash generation and therefore faster hash Digest cracks as opposed to using a few expensive high performance computers.

Future work would involve extending the Frontend to support computers or networks of computers across the internet. In this way, a user could use any Node to simply add a computer or a network of computers, which may be connected to even more computers and so on, to the current network. This significantly increases the ease of physically networking computers and eliminates the requirement of using computers to be within the same locality, area or subnet.

This brings about a possibility of misuse of massive computer resources. As of present times, botnets, which are large number of compromised computers that are used to generate spam, harvest credit card details, etc., are used by malicious hackers for only directly harmful purposes. I theorize that future times could see hackers harnessing the computing power of botnets instead of their internet resources. An average botnet possesses about 20,000 computers [9] and occasional Super botnets possess computers by the millions [8]. Such large number of computers would consist of not only many high performance gaming machines but also powerful office or home computers. For approximation calculations, tests were conducted on an average Intel I5 processor @ 2.5 GHz. They yielded an average of 132 Million hashes/second. When excluding powerful machines and taking a bare minimum average of 100 Million hashes/second per computer, an average sized botnet of 20,000 computers would yield 200 Billion hashes/second. Inclusion of high performance computers could put this at greater figures of up to 3 or 4 times the current one. An attacker could use this mean of computing power to crack critical authentication messages, high security digital fingerprints, complex passwords etc. which cannot otherwise be cracked by even custom built high performance cluster machines. All the aforementioned predictions can easily hold true if the attacker manages to efficiently distribute the Job among such a large number of computers.

References

[1] David Apostal, Kyle Foerster, Amrita Chatterjee, Travis Desell: "Password Recovery Using MPI and CUDA", 19th Annual International Conference on High Performance Computing, 2012.

[2] Tomosuke Murakami, Ryuta Kasahara and Takamichi Saito: "An Implementation and its Evaluation of Password Cracking Tool Parallelized on GPGPU", Communications and Information Technologies (ISCIT), 2010 International Symposium on, pp 534 - 538

[3] Anh-Duy Vu, Jea-Il Han, Hong-An Nguyen, Young-Man Kim, Eun-Jin Im, "A Homogeneous Parallel Brute Force Cracking Algorithm on the GPU", 2011 IEEE

[4] BarsWF, Svarichevsky Mikhail, http://3.14.by/en/md5

[5] Feng Wang, Canqun Yang, Qiang Wu, Zhicai Shi , "Constant Memory Optimizations in MD5 Crypt Cracking Algorithm on GPU-Accelerated Supercomputer Using CUDA", The 7th International Conference on Computer Science & Education (ICCSE 2012)

[6] Duc H.Nguyen, Thuy T.Nguyen, Tan N.Duong, Phong H.Pham, "Cryptanalysis of MD5 on GPU Cluster", in The 2010 International Conference on Information Security and Artificial Intelligence (ISAI2010), Chengdu, China, 17-19 December, 2010.

[7] Andrew D. Zonenberg, 2009, Rensselaer Polytechnic Institute, 27

[8] James Wray and Ulf Stabe (2010-10-28). "Researchers: Bredolab still lurking, though severely injured (Update 3) - Security". Thetechherald.com

[9] "Hackers Strengthen Malicious Botnets by Shrinking Them". Computer (IEEE Computer Society). April 2006.

About Author (s):

I'm an outgoing Student of the dept. of Computer Science and Engineering, CEG, Anna University. I've been actively interested in researching and implementing novel ideas, tools and software since pre college days. I am interested in digital security, study and beating of reverse engineering, network programming and game programming