Scientific Research Publishing

Journal of Information Security, 2019, 10, 45-68
http://www.scirp.org/journal/jis
ISSN Online: 2153-1242
ISSN Print: 2153-1234

Security of Password Hashing in Cloud

Parves Kamal

Department of Information Systems Assurance, St. Cloud State University, St. Cloud, MN, USA
Email: pakaml@stcloudstate.edu

How to cite this paper: Kamal, P. (2019) Security of Password Hashing in Cloud. *Journal of Information Security*, **10**, 45-68.
https://doi.org/10.4236/jis.2019.102003

Received: January 23, 2019
Accepted: February 25, 2019
Published: February 28, 2019

Abstract

Though the History of using password in computing can be traced back to as far as mid of last century little focus has been implied on how to securely store and retrieve password to authenticate and authorize services to the end users. In this paper the current security of various password hashing schemes that are in use today will be investigated through practical proof of concept-GPU based, password hash dump cracking using the power of cloud computing. We will be providing comparison on different password hashing cracking time using the cloud GPU power in AWS. The focus of this paper is to show the possible use of cloud computing in cracking hash dumps and the way to countermeasures them by using secure hashing algorithm and using complex passwords.

Keywords

Index Terms-Hash, Collision, GPGPU, CUDA, OpenCL

1. Introduction

The most common means of authentication scheme are password-based authentication system [1]. An employee uses multiple passwords daily for all the applications and systems that he/she might be working on for the employer. Businesses spend a tremendous amount of money for not only storing these passwords but also for securing the storage of these passwords. Especially when organization deals with a huge number of customers; it's very hard for them to create, maintain and distribute these passwords across the network for authentication, authorization or accounting purposes. Thus, passwords based authentication system possesses many security problems into rather relatively secured existing infrastructures [2]. To overcome the possible security concerns with storing and distributing the password across the network, the password is often run against the cryptographic hash function to get the equivalent digest of the password which is stored along with the user's other credentials in the database. When users try to login with the password, the input is calculated by the same hash function to compare with the digest of the same password that has been stored in the databases. One of the properties of the cryptographic hash function is its irreversible one-way function which means it's nearly impossible to get back the password from the digest itself. Then again, many of the commonly used hashing functions like MD5, SHA-1, etc. have been developed during the mid-nineties. One of the weaknesses of most widely used hash function MD5 is that the attacker can create two identical digests for two different inputs which in cryptography field is called Hash collision [3]. In fact, the possibility of an adversary of finding the password from the hash dump is proportional to the amount of the work he/she puts in and the ability to predict the password characteristics distribution. Moreover, with the advent of the cloud computing and 8 new powerful Graphics Processing Unit (GPU), the attacker is now well equipped than ever to decipher the passwords from the hash at their will. Since last decades there has been significant development going on in the field of Graphics Processing Unit (GPU). The GPU is very suitable for performing parallel tasks as well as calculating floating point related problems. Modern GPU based, password cracking is ten times faster than Central Processing Unit (CPU) based password cracking [4]. In our paper, we will be showing how an attacker can leverage existing cloud services to crack passwords from sample password hash dumps using the high-performance-based computing resources. We think our work is one of the few studies that have been done on the power of GPU computing using cloud computing services like Amazon web services and the sheer breadth of test that is performed. Our paper is structure in way that first few sections we will be providing the reader some background on all the possible password related attacks possible where in the middle section we set up our test environment in AWS with sample password hash file of different popular hash algorithm that are in use today to crack. In the end we compare the results and we retrieve of the cracking time from our test scenario. We then compare the test results and suggest the use of password generator script and other possible means of creating strong passwords for using online services.

2. Nature and Significance of the Problem

Due to the recent hacking and public disclosure of private information (User's passwords) from several big profile organizations like LinkedIn, E-harmony and Yahoo within last 5 years raises the serious questions of not only the security of the authentication systems in these high profile organizations but also the security aspects of their password storing techniques in their databases.

Hence, in this paper the work presented is to show how effective the GPU based, password cracking technique is against the hashing techniques and provides insight on why choosing strong, complex passwords along with slow computers hashing function will keep the attackers at bay while leveraging GPU processing power of the cloud computing resources. We believe our model can be applied to any form of online GPU/CPU resources and produce the similar results for identifying the strength of the password hash of any type. Previous work has been done usually based on single instances of computing resources, but we believe our environment provides the more up to date distributed power of GPU that is available currently in cloud. Here are the few terms that should help understand some of the acronyms better is shown in **Table 1**.

3. Related Work

Thompson in his paper has shown how GPU's can be used for usual computing task other than graphically intensive work [5] whereas Cook in his paper shown GPU's are good for solving cryptography related work too [6]. In the beginning, it was hard to make any application that could take advantage of GPU's because of the lack of API's and supports. Now with the advent of CUDA, OpenCL platform, APIs for GPU's become widely available. Using GPU's performance on cryptographic computation has been under review by many researchers. Yang and his colleague have shown how GPU's can outperform high-performance CPUs in symmetric cryptographic computations [7]. In Asymmetric encryption strength has also been reviewed by many researchers suing GPU's [8] [9] [10]. Hash functions like MD5 and Blowfish have also been tested with GPU processing power, outperforming the CPU's significantly [11] [12].

Graphical Processing Unit or GPU is now lots used for general purpose computing or better known as GPGPU than rather using it as to drive graphics.

With the advent of CUDA and OpenCL framework researcher are putting the hash security to the test by exploiting the power of the GPU to crack them using parallel processing power. R. Zhang and his colleagues showed the method to crack MD5 hash using CUDA and reached the speed of 223 Mbps [13]. Another researcher compared decryption software John, the Ripper against cracking software based on OpenCL and found 17-times faster speed [14]. In other research, the authors implemented MD5 decryption methods using Tianhe-1A

Table 1. Definition of terms.

Acronym	Description
GPU	Graphics Processing Unit
CPU	Central Processing Unit
MD5	Message Digest Algorithm 5 (1992)
Hash	Cryptographic fixed-size value from arbitrary input
Collision	Two different input yields to the same output hash value
GPGPU	General-Purpose Graphics Processing Unit
GFLOPS	(Giga Floating Point Operations Per Second)
OpenCL	Open Computing Language
CUDA	Compute Unified Device Architecture

using CUDA to reach calculation speeds up to 18 billion keys per seconds [15]. Oclhashcat happens to be the multiplatform world's fastest password cracker which is GPUGPG based open source free hash cracker with speeds of up to 8511 mc/s and 2722 mc/s for MD5 and SHA-1 hash respectively. In one study researcher, implemented MD5 decryption algorithm using GPU cluster and gain 100 times faster performance in comparison to CPU [16]. Many of the researchers used open source cracker like Oclhashcat on GPU platforms like NVidia or AMD using CUDA or OpenCL. In our paper, we will also exploit GPU power that is on offered in the cloud to crack sample password hash dump using Oclhashcat.

4. What Is a Hash

The process of taking an arbitrary length input and converting it to a fixed-length output value by integrating through a cryptographic hash function is called hashing, and the output value is called the hash value as shown in **Figure 1**. The property of cryptographic hash functions is they are a one-way function, and there is no way of deducing the input value by reverting from the output hash value [17].

Because of its irreversible features, hashing is very useful for storing the password in the authentication server. This allows not only protecting customer's privacy but also allows the server to authenticate user's login information without storing the user's password. Let's say for example the cryptographic hash value of the word "parves" using MD5 algorithm is:

MD5 (parves) = cf7cccd2e366698244ac5891da31bb82

4.1. Different Hashing Algorithm

In the following table the function of a different hashing algorithm that is in use at large is shown (**Figure 2**).

4.2. Password Hash Cracking Techniques

Password hash cracking is a method of attacking dumped hash to find flaws in the underlying secured hash characteristics that we discussed in the previous section to find the message that computer to the same hash value. The attacker, once they get hold of the hash file by getting the unauthorized access to the network, usually copies the hashed password file and perform one of the following attacks.

- Exhaustive Attack:

This attack involves trying every possible combination of characters within character sets. Since it looks for every possible combination, the success rate is 100% of finding the correct combination given the time, and the cost is out of consideration. Also, the most cryptographic system in use today uses very large space of time, making the exhaustive search impractical to perform against. The exhaustive attack is used in two of the following ways [20].

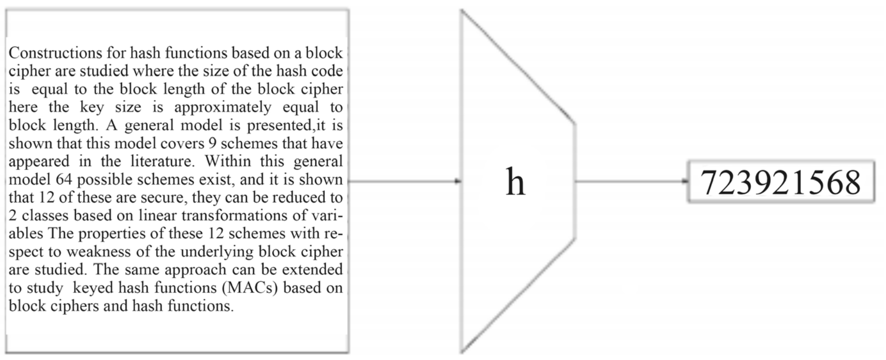
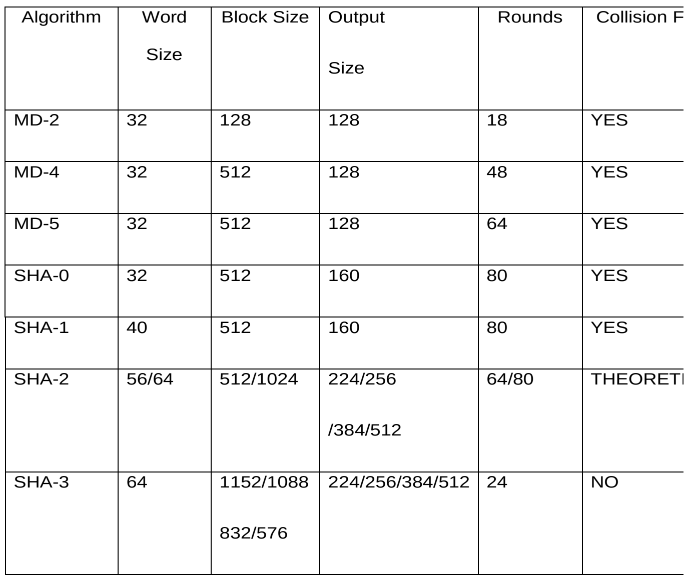

Constructions for hash functions based on a block cipher are studied where the size of the hash code is equal to the block length of the block cipher here the key size is approximately equal to block length. A general model is presented,it is shown that this model covers 9 schemes that have appeared in the literature. Within this general model 64 possible schemes exist, and it is shown that 12 of these are secure, they can be reduced to 2 classes based on linear transformations of variables The properties of these 12 schemes with respect to weakness of the underlying block cipher are studied. The same approach can be extended to study keyed hash functions (MACs) based on block ciphers and hash functions.

Figure 1. Hashing (h) Function in operation [18].



Algorithm	Word Size	Block Size	Output Size	Rounds	Collision F
MD-2	32	128	128	18	YES
MD-4	32	512	128	48	YES
MD-5	32	512	128	64	YES
SHA-0	32	512	160	80	YES
SHA-1	40	512	160	80	YES
SHA-2	56/64	512/1024	224/256 /384/512	64/80	THEORETI
SHA-3	64	1152/1088 832/576	224/256/384/512	24	NO

Figure 2. Comparison between different hash algorithms [19].

Known-plaintext only attacks: In this attack, only the perpetrator knows the plaintext and ciphertext both. Then he tries to discover the key that encrypts the plaintext for example.

Ciphertext-only attacks: The attacker only knows the ciphertext, and he tries to find the corresponding key or plaintext by going through every combination of the keys.

Password hashing crack technique is only possible in cipher text the only method as every hash function is a one-way function. However, with a single digit increase in password length makes the exhaustive search iteration increase exponentially, making the exhaustive search attack impractical for the attacker, especially when long and complex alphabetical characters are chosen [21].

- Dictionary Attack:

A Dictionary attack is a very effective attack if the user uses some human-memorable password for their login credentials and the attacker tries a list of common words and expressions used in any language for example in English to find the password. Users tend to use common or simple passwords [22] across many platforms, so that can be easily recalled and if it's the case then performing dictionary attacks is highly successful. But some simple modification to those common words can make a dictionary attack highly unsuccessful too [13].

- Rainbow table Attack:

Rainbow table Attacks involve looking up the precomputed hash tables and the corresponding key value to find any matching hash. This type of attack, though, has limited combination to look up for but when working within its constraints, it takes less time decrypting hashed password than those two other attacks mentioned earlier [23].

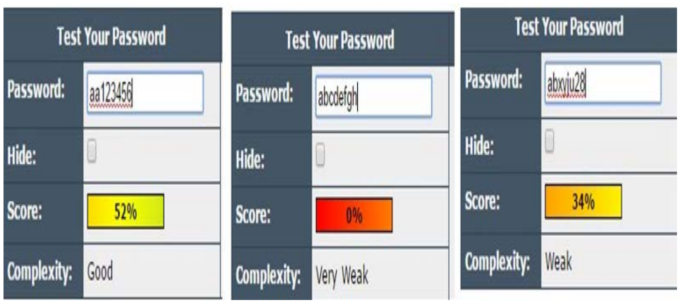
4.3. Characteristics of Strong Passwords

There are many characteristics that make a password easy to crack using exhaustive attacks. Some of the characteristics are:

- Passwords are based on common dictionary words.
- Passwords are easily guessable.
- Passwords are relatively short in length, making it possible to brute force attack easily.
- Passwords have some sorts of the pattern which is easy to deduce. e.g. abc123, XYZ, 46824682 etc.
- Passwords have been repeating characters like abab11 or xy12x. People Use the passwords with repeating characters so that they can remember.

To better understand, let's show it more graphical way. The following pictures are some of the chosen passwords with a combination of characters, numbers, and symbols to check the strength of the passwords based on the characteristics of the characteristics above of strong passwords.

So, as we have seen even if all the passwords above eight characters were long, using memorable words like a watch or if using characters that come in sequences significantly makes the password strength very low while using random sequences of characters makes passwords relatively stronger. It's very hard to get real random numbers, and for humans, we are not well equipped to remember random numbers. In fact, a human can only remember random numbers up to 7 ± 2 characters [25]. Having passwords with a random sequence of numbers with at least one capital letter, number, a Special character from large key space makes password very strong as shown in **Figure 3**.



Test Your Password

Password:	aa123456
Hide:	☐
Score:	52%
Complexity:	Good

Test Your Password

Password:	abcdefgh
Hide:	☐
Score:	0%
Complexity:	Very Weak

Test Your Password

Password:	abxyju28
Hide:	☐
Score:	34%
Complexity:	Weak

Figure 3. Example of some of the weak and strong passwords [24].

One of the methods used to check the strength of the password is based on the entropy. Entropy depends on the length of the passwords and the key space used. Entropy can be calculated using the formula, log2(n) [26] where n represents the number of characters in key space.

4.4. CPU-Based versus GPU-Based Password Cracking Performance

The CPU has traditionally been used for general purpose computing. Usually, the CPU has limits on how many processing cores it can accommodate. To overcome these CPU's is hyper threading technology to compensate whereas the GPU has many more cores compared to its similarly priced CPU counterpart.

GPU also works as Single instruction, but Multiple Data computations or known as (SIMD) whereas CPU's work as Single Instruction, Single Data computations or known as (SISD). This makes GPU largely suited for password cracking. Part of the reason behind the GPU'S power over CPU is for the recent increase in GPU'S power in comparison with the CPU'S one. AS we know from Moore's law CPU'S power doubles once in 18 months, whereas GPU'S power is doubling four times at the same time [27]. The difference between CPU'S and GPU'S can regard performance based on single precision floating point number is clearly shown in Figure 4.

4.5. Cloud Computing

Cloud computing is rapidly provisioned on demand configurable resources that can be shared with minimal management effort or service overhead [22]. There are two types of cloud—public and private. Public cloud is a cloud service offered by the cloud. A Private cloud is exclusively provisioned for a certain group of users. Because of the flexibility and the cost effectiveness every company is

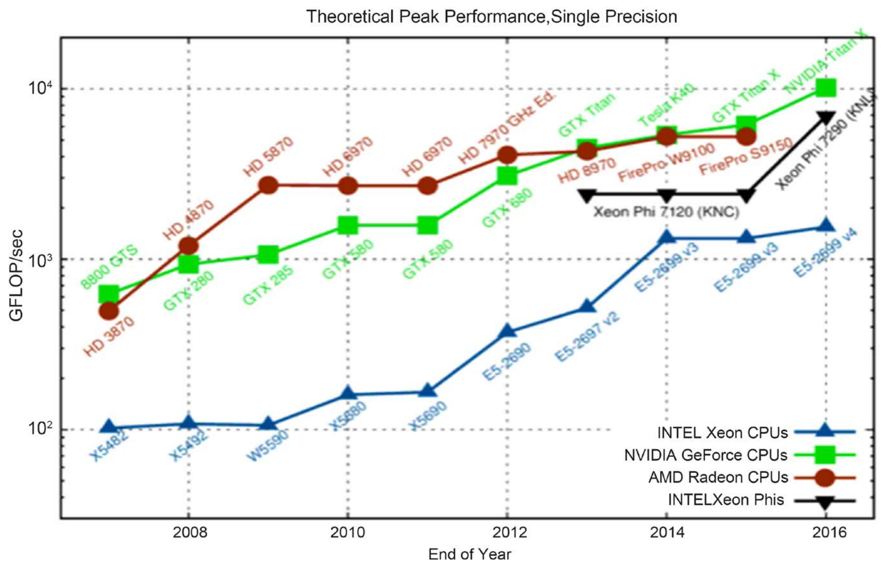


Figure 4. Theoretical peak GFLOP/Sec between CPU'S vs. GPU'S [28].

moving their infrastructure to the cloud now. There are many public cloud companies like AWS, Rackspace, Google Cloud, Microsoft Azure provide (**Figure 5**).

Spinning up a virtual machine in the Amazon AWS with Intel Xeon-based G series computing along with NVidia's K520 (1536 cores) cost about $0.02 to $2.87 per hour making it very feasible for someone to run parallel GPU-based, password cracking on the password hash dump. In my paper I will demonstrate how effective is GPU based, password cracking using cloud platform.

5. Test Environment

Test Environment was set up to test the following characteristics:

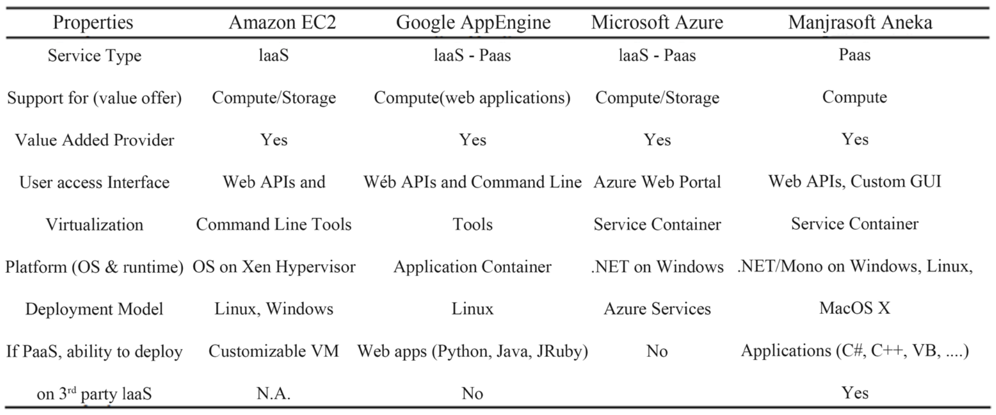
- To identify the characteristics of strong hashing algorithm as well as the passwords with the data gathered from the proof of concept—password hash cracking in the cloud.
- To show the performance of the GPU on password hash cracking using cloud we will implement GPU based, hash cracking on cloud and we will compare this with CPU based cracking.
- Compare the analyzed data to compare the GPU based, and CPU based, password cracking performance as well as the effect of using secured password hashing algorithm on the cracking performance. Also analyze the reason behind the password hash cracking effectiveness based on password strength.

5.1. Data Collection

The data will be collected once the test is performed and the benchmark report, as well as the generated passwords, will be the source of the data which will later be analyzed.

5.2. Tools and Techniques

Multi-GPU based Oclhashcat on Amazon AWS EC2 on CUDA based NVIDIA Tesla GPU to crack sample password hash of MD5, SHA-256, and Bcrypt.



Properties	Amazon EC2	Google AppEngine	Microsoft Azure	Manjrasoft Aneka
Service Type	laaS	laaS - Paas	laaS - Paas	Paas
Support for (value offer)	Compute/Storage	Compute(web applications)	Compute/Storage	Compute
Value Added Provider	Yes	Yes	Yes	Yes
User access Interface	Web APIs and	Wéb APIs and Command Line	Azure Web Portal	Web APIs, Custom GUI
Virtualization	Command Line Tools	Tools	Service Container	Service Container
Platform (OS & runtime)	OS on Xen Hypervisor	Application Container	.NET on Windows	.NET/Mono on Windows, Linux,
Deployment Model	Linux, Windows	Linux	Azure Services	MacOS X
If PaaS, ability to deploy	Customizable VM	Web apps (Python, Java, JRuby)	No	Applications (C#, C++, VB, ...)
on 3rd party laaS	N.A.	No		Yes

Figure 5. Cloud computing offered solutions and their comparison [29].

5.3. Hardware and Software Environment

The following hardware and software will be used in conducting my research. The setup was done in AWS (Amazon Web Service) cloud. The exact software and Hardware details are in the table. The test will be conducted in cracking the password hash on both GPU and CPU instances (Machine). The specifications for both GPU Machines as shown in Table 2 and CPU machines are given in Table 3.

5.4. Testing Environment Diagram

The testing Environment of this research is shown (Figure 6).

5.5. Wordlist Selected

For the test, different filters and combination of uppercase letters, lowercase letters, digits, special characters will be used. The test will be done using the hybrid attack at first before applying brute force attacks where every combination of the characters will be tried.

5.6. Installing Oclhashcat

First, the installation of the oclhashcat multi-GPU based, hash cracker with the following steps after ssh into our cloud Linux machine on both CPU and GPU test machine (Figure 7).

Table 2. GPU test machine.

Software	Hardware
Ubuntu 16.04 with NVIDIA GRID and TESLA GPU Driver	Intel Xeon E5-2670 (Sandy Bridge) Processors
Oclhashcat is a GPGPU-based multi-hash cracker	P2 instances provide up to 16 NVIDIA K80 GPUs, 64 vCPUs and 732 GiB of host memory, with a combined 192 GB of GPU memory, 40 thousand parallel processing cores, 70 teraflops of single precision floating point performance, and over 23 teraflops of double precision floating point performance
	vCPU-32 Ram-488 GPU-8 GB SSD-300 GB

Table 3. CPU test machine.

Software	Hardware
Ubuntu 16.04—with Updates HVM-1602	Intel Xeon E5-2666 v3 2.9 GHz
Oclhashcat is a CPU-based multi-hash cracker	High-frequency Intel Xeon E5-2666 v3 (Haswell) processors optimized specifically for EC2 EBS-optimized by default and at no additional cost Ability to control processor C-state and P-state configuration on the c4.8xlarge instance type
	VCPU-8 RAM-15 GB SSD-30 GB

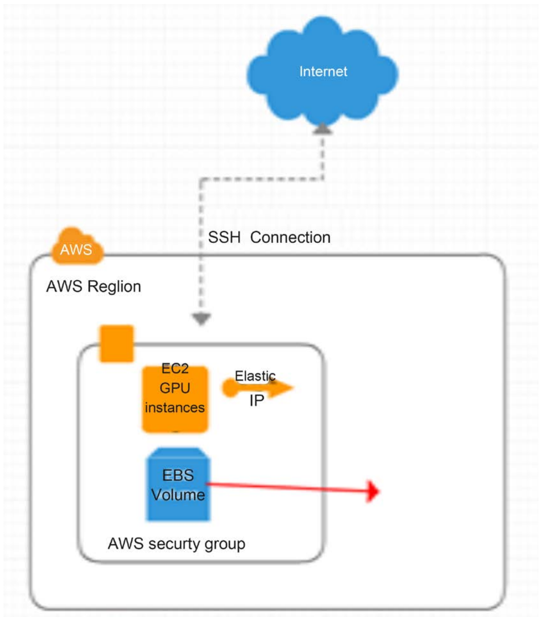


Figure 6. Testing environment.

```
sudo apt -y update sudo apt -y upgrade
sudo apt install —y p7zip—fu11 build—essential linux-image-extra-virtual linux-source

echo  options nouveau modeset=O      I sudo tee —a / etc/ madprabe . d/nouveau—kms . canf
sudo   update—initramfs —u

#to activate latest kernel
sudo    reboot

s sh     -i keyfile .pem  ubuntu@<ip>
sudo    apt install linux-headers-  'uname -r '

sudo    mkclir -p / data
sudo    chown ubuntu / data
cd / data

#  latest driver here: http: // www.nvidia.cam/zbject/unix.html
wget http: // us . download.nvidia. cam/XFree8E/Linux—x86 64/375.26/NVIDIA-
Linux-x86 64-375.26.run
chmod 755 NV*. run
sudo . /NV*. zun
# accept license, install, enter,    enter

1 smacl I nvidia
# if not Izaded, do extra reboot

# optimizations from ÄWS blog
sudo nvidia—smi              -pm 1
sudo nvidia—smi              -acp O
sudo nvidia—smi              --auto-boost-permission=0
sudo nvidia—smi              -ac 2505, 875

# Download and install Hashcat
wget http: // hashcat . net/ files/ hashcat—3.40.7z
7za x hashcat—3.            40.7z
Cd hashcat—3 . 40

/ hashcatE4 . bin     -b
```

Figure 7. Installing oclhashcat.

5.7. Sample Password Hashed Dump File

To conduct the test some common password up to 8 characters in length characters' length and will hash it with MD5 and SHA1 as well as more cryptographically strong hashing algorithm bcrypt. The following sample password hash has been selected.

Sample password list:
Password
HELLOO
MYSECRET
test1234
password!
You9can!
./?';,<>
Mysecret

The MD5 and SHA1 hash of the generated password from the above is generated by the following code (**Figure 8**).

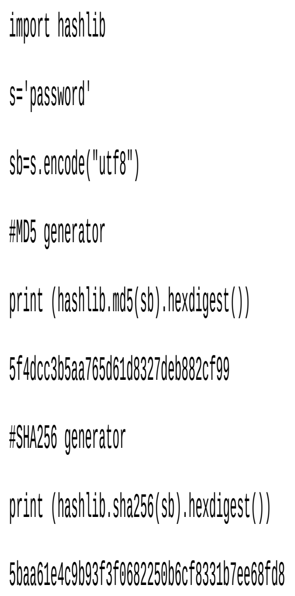
- MD5 Hash:

Password-**dc647eb65e6711e155375218212b3964**
HELLOO-**16454bd041c46012e31778eb94b8111a**
MYSECRET-**958152288f2d2303ae045cffc43a02cd**
test1234-**16d7a4fca7442dda3ad93c9a726597e4**
password!-**49f24c0c152b2375431210f9443d176f**
You9can!-**b64b0e1165a77bd90a1673469f1af0e1**
./?';,<>-**7185ad7f0851780a2db24edc8347b12a**
Mysecret-**06c219e5bc8378f3a8a3f83b4b7e4649**

- SHA256 Hash:

Password-**e7cf3ef4f17c3999a94f2c6f612e8a888e5b1026878e4e19398b23bd38ec221a**
HELLOO-**2d32d2db26a9a8e8b3a69a5739a17981a5a064a5c3037a8d3891ab3e41f57246**
MYSECRET-**3fcdbc4a0ed38df8d4bd234e2c8ad3b2623fa5265f31763d1e91a848471a8a9b**
test1234-**937e8d5fbb48bd4949536cd65b8d35c426b80d2f830c5c308e2cdec422ae2244**
password!-**c075349b9b6f6b3e41b34e4e71ac22a685102b0b2246c5f84d67c5eed3ad39fb**
You9can!-**4824033be89e919a06ac33255b06761f706edc1ac8fc37b86574892ab7c3248d**
./?';,<>-
9070906f306d5d34c301b9f4cda9f71c2a19543ccea44b6b08c18b3d76941936
Mysecret-**652c7dc687d98c9889304ed2e408c74b611e86a40caa51c4b43f1dd5913c5cd0**

Bcrypt hash of the following password with the following code.



```
import hashlib

s='password'

sb=s.encode("utf8")

#MD5 generator

print (hashlib.md5(sb).hexdigest())

5f4dcc3b5aa765d61d8327deb882cf99

#SHA256 generator

print (hashlib.sha256(sb).hexdigest())

5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8
```

Figure 8. Python MD5/SHA password hash Generator sample code.

Sample Password:
HELLO
1!Su
Pass
pass
1234
:/?<

Here the bcrypt hash of sample password "Pass" was created with 14 rounds (2^{14} = 16,384) of generation via gensalt() function. For the sake of our test, we will be keeping it to minimal, less than 10 (**Figure 9**).

The Resulting generated Bcrypt hash of the above sample password was:

HELLO-**$2y$10$vHvY3cA252u/68KesxmOg.WIjkWOHQcFqXc.KRSV4aLn/pIC1D5ZC**

1!Su-**$2y$10$eVaCxfaaMDXKS5LTu1uX2O6k3/lUpGE83luvxfoGdkM6HAMVWirvW**

Pass-**$2y$10$K8nQaEFpiZkSdkjKlXfjveu44pTKD/lvpOU2Cu/8INh2vPDUgS29e**

Pass-**$2y$10$wacXM0HGg/26pzRGvIEE4OMg4jGIHjhptHKMdowmdr4zpTyu0triC**

1234-**$2y$10$FwaS9uiu7mkCIJ.d9fCYI.PsF8qY5I1ZdrbJ0qBBcHdDLhours3gxPLq**

:/?<-**$2y$10$kjGTFQVsc7eislYriw0ibu1GZi1rUod0unnpY0rT9eeSIYGBJRiui**

6. Test Results

We ran the CPU crack and GPU cracking on the sample MD5, SHA-256, and Bcrypt hash dump. We applied the different filter as follows:

```
import bcrypt

password = b"Pass"

# Hash a password for the first time, with
a certain number of rounds

hashed = bcrypt.hashpw(password,
bcrypt.gensalt(14))

print(hashed)

b'$2b$14$4tWHOXsyYtuVD8CbzjMUbeNqfMKGKiECOD
TkQ4Zcf11wJ6nJAD7XW'
```

Figure 9. Python Bcrypt password hash Generator sample code.

u = uppercase letters only—total 26 characters

l = lowercase letter only—total 26 characters

ul = uppercase and lowercase—total 52 characters

d = digits only—total 10 digits

s = special characters only—total 33 characters

ls/us = lowercase/uppercase with special characters—total 59 characters

usld = lowercase, uppercase, special character, and digits—total 95 characters

We also conducted an experimental run where we applied fixed characters in certain positions to observe any improvements in timing. All our CPU/GPU/Experimental test results are shown in the following tables.

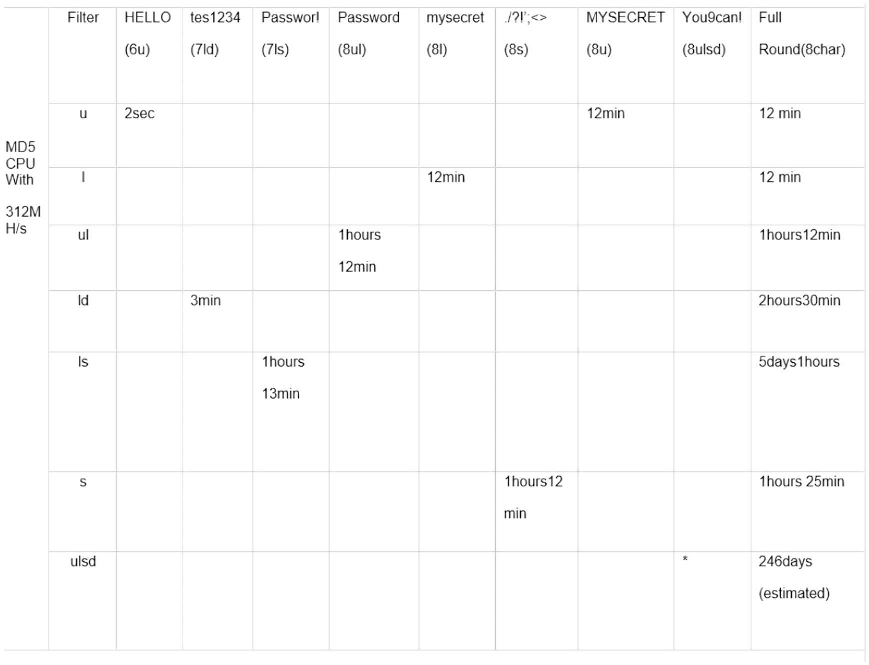
In Figure 10 as the result is as follows:

- Using uppercase only character (u) it takes 2sec for 6-character long password, HELLO to crack where 8-character long password "MYSECRET" takes 12 mins and it finishes checking all the combination of 8 characters roughly at the same time too.
- Using lowercase only character (l) it takes 12 min to crack 8-character long password "mysecret" as well as finishes checking all the combination of 8 characters roughly at the same time too.
- Using a combination of uppercase and lowercase character (ul) it takes 1 hour 12 min to finish cracking 8 char long "Password" as well as finishes checking all the combination of 8 characters roughly in the same time too.
- Using lowercase and digits (ld) it takes 3 mins to crack 7 char long "tes1234" password while it takes 2 hours 30 mins to finish checking all the combination of the 8-char field.
- Using lowercase and a special character (ls) it takes it takes 1hours 13 min to find the 7-char password "Passwor!" while it will take approximately 5 days and 1hours to finish checking all the combination.

The way we estimated 8 char long password cracking time with lowercase and special characters are as follows:

Total characters: lowercase (26) and special characters (33) = 33 + 26 = 59

Total combination possible $[59]^8$ = 1.4683044e+14



	Filter	HELLO (6u)	tes1234 (7ld)	Passwor! (7ls)	Password (8ul)	mysecret (8l)	./?!';<> (8s)	MYSECRET (8u)	You9can! (8ulsd)	Full Round(8char)
MD5 CPU With 312M H/s	u	2sec						12min		12 min
	l					12min				12 min
	ul				1hours 12min					1hours12min
	ld		3min							2hours30min
	ls			1hours 13min						5days1hours
	s						1hours12 min			1hours 25min
	ulsd								*	246days (estimated)

Figure 10. MD5 cracking performance with CPU machine.

MD5 CPU cracking speed = 312 MHS = 312,000,000 H/s

Cracking time in days = 1.4683044e+14/312,000,000 = 470610.376937 s/3600 = 130.725104705 hours/24 = 5 days 1 hours

- Using the special characters only it takes 1 hours 12 min to finish cracking 8 char long password "./?!';<>" while it takes 1 hours 25 min to finish checking all the special character combination of the 8char field.
- At last we try all the lowercase, uppercase, special characters and digits (usld) for all the 8 filed of the password and though we could not find out 8 char long password "You9can!" we did find to estimate how long it will take to look up all the (USLD) combination of each 8-char filed with our CPU machine with the following calculation.

Total characters: lowercase (26) and special characters (33), uppercase (26) and digits (10) = 33 + 26 + 26 + 10 = 95

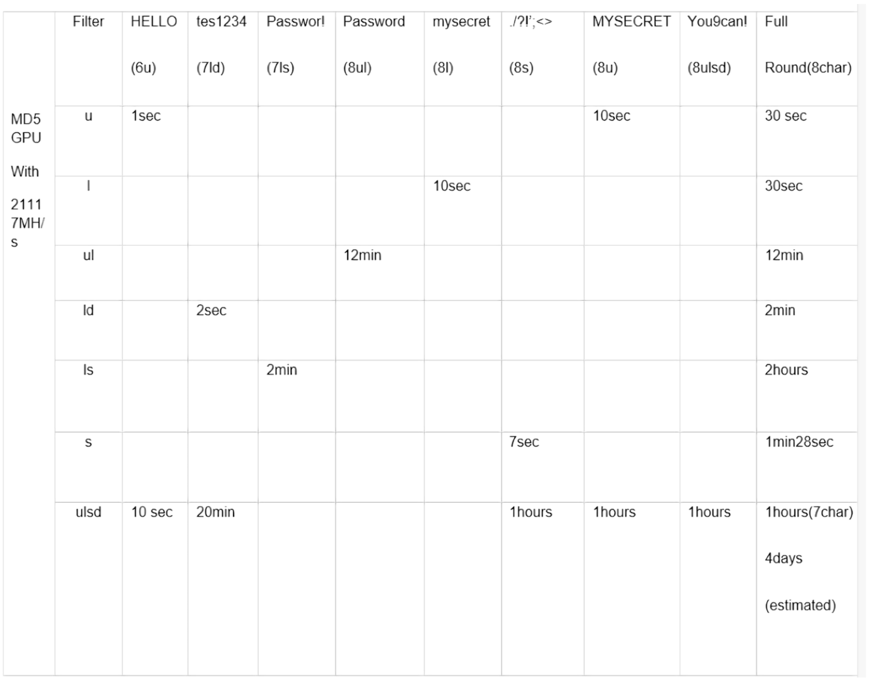
Total combination possible $[95]^8$ = 6.6342043e+15

MD5 CPU cracking speed = 312 MHS = 312,000,000 H/s

Cracking time in days = 6.6342043e+15/312,000,000 H/s = 21263475.3618 s/3600 = 5906.52093384 hours/24 = 246 days

In Figure 11 the result is as follows:

- Using uppercase only character (u) it takes 1sec from 6-character long password, HELLO to crack where 8-character long password "MYSECRET" takes 10 sec and it finishes checking all the combination of 8 characters roughly in 30 secs.
- Using lowercase only character (l) it takes 10 secs to crack 8-character long password "mysecret" as well as finishes checking all the combination of 8 characters roughly in 30 secs.



	Filter	HELLO (6u)	tes1234 (7ld)	Passwor! (7ls)	Password (8ul)	mysecret (8l)	./?!';<> (8s)	MYSECRET (8u)	You9can! (8ulsd)	Full Round(8char)
MD5 GPU With 21117MH/s	u	1sec						10sec		30 sec
	l					10sec				30sec
	ul				12min					12min
	ld		2sec							2min
	ls			2min						2hours
	s						7sec			1min28sec
	ulsd	10 sec	20min				1hours	1hours	1hours	1hours(7char) 4days (estimated)

Figure 11. MD5 cracking performance with GPU machine

- Using a combination of uppercase and lowercase character (ul) it takes 12 min to finish cracking 8 char long "Password" as well as finishes checking all the combination of 8 characters roughly in the same time too.
- Using lowercase and digits (ld) it takes 2 secs to crack 7 char long "tes1234" password while it takes 2 mins to finish checking all the combination of the 8-character field.
- Using lowercase and a special character (ls) it takes it takes 2 min to find the 7-char password "Passwor!" while it will take approximately 2hours to finish checking all the combination.
- Using the special characters only it takes 7 sec to finish cracking 8 char long password "./?!';<>" while it takes 1 min 28 sec to finish checking all the special character combination of the 8char field.
- At last, we try all the lowercase, uppercase, special characters and digits (ulsd) for all the 8 field of the password and for the 8-char long password (./?!';<>, MYSECRET, You9can!). We could not finish the all the combination of 8 chars, but we estimated it will take around 4 days to finish checking all the possible combination of 8-char password using our GPU machine. The way we estimated the cracking time as follows:

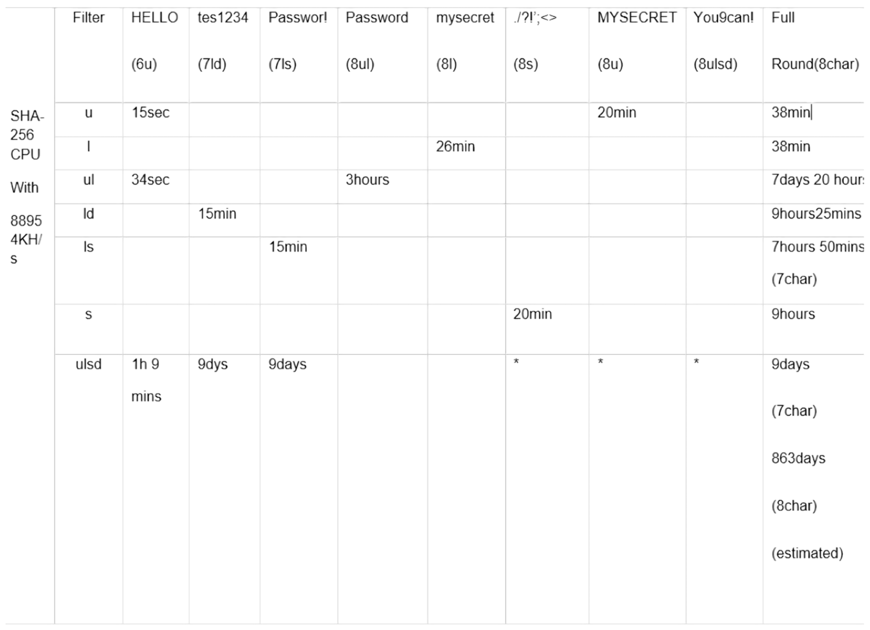
Total characters: lowercase (26) and special characters (33), uppercase (26) and digits (10) = 33 + 26 + 26 + 10 = 95

Total combination possible $[95]^8$ = 6.6342043e+15

MD5 GPU cracking speed = 21117 MHS = 21,117,000,000 H/s

Cracking time in days = 6.6342043e+15/21,117,000,000 H/s = 314164.14798 s/3600 = 87.2678188833 hours/24 = 4 days

In **Figure 12** the result is as follows:



	Filter	HELLO (6u)	tes1234 (7ld)	Passwor! (7ls)	Password (8ul)	mysecret (8l)	./?!';<> (8s)	MYSECRET (8u)	You9can! (8ulsd)	Full Round(8char)
SHA-256 CPU With 88954KH/s	u	15sec						20min		38min
	l					26min				38min
	ul	34sec			3hours					7days 20 hour
	ld		15min							9hours25mins
	ls			15min						7hours 50mins (7char)
	s						20min			9hours
	ulsd	1h 9 mins	9dys	9days			*	*	*	9days (7char) 863days (8char) (estimated)

Figure 12. SHA-256 cracking performance with CPU machine.

- Using uppercase only character (u) it takes 15sec for 6-character long password HELLO to crack where 8-character long password "MYSECRET" takes 20 mins and it finishes checking all the combination of 8 characters roughly in 38 mins.
- Using lowercase only character (l) it takes 26 min to crack 8-character long password "mysecret" as well as finishes checking all the combination of 8 characters roughly in 38 mins.
- Using a combination of uppercase and lowercase character (ul) it takes 3 hours to finish cracking 8 char long "Password" as well as finishes checking all the combination of 8 characters will take estimated 7 days 20 hours to finish.

  The way we estimated the cracking time as follows:

  Total characters: lowercase (26), uppercase (26) = 26 + 26 = 52

  Total combination possible $[52]^8$ = 5.3459729e+13

  SHA-256 CPU cracking speed, 88,954 KH/s = 88,954,000 h/s

  Cracking time in days = 5.3459729e+13/88,954,000 = 600981.726864 s/3600 = 166.939368573 hours/24 = 7 days

- Using lowercase and digits (ld) it takes 15 mins to crack 7 char long "tes1234" password while it takes 9 hours 25 mins to finish checking all the combination of the 8-char field.
- Using lowercase and special character (ls) it takes it takes 15 min to find the 7-char password "Passwor!" while it will take 7 hours 50 mins to check 7 chars field.
- Using the special characters only it takes 20 min to finish cracking 8 char long passwords "./?!';<>" while it takes 9 hours to finish checking all the special character combination of 8 char field.

- At last we try all the lowercase, uppercase, special characters and digits (usld) for all the 8 field of the password and though we could not find out 8 char long password "You9can!" we did find to estimate how long it will take to look up all the (USLD) combination of each 7 char and 8-char filed with our CPU machine with following calculation.

Time Estimation for 7-char password

Total characters: lowercase (26) and special characters (33), uppercase (26) and digits (10) = 33 + 26 + 26 + 10 = 95

Total combination possible $[95]^7$ = 6.983373e+13

SHA-256 CPU cracking speed, 88,954 KH/s = 88,954,000 h/s

Cracking time in days = 6.983373e+13/88,954,000 h/s = 785054.405753 s/3600 = 218.070668265/24 = 9 days

Time Estimation for 8-char password

Total characters: lowercase (26) and special characters (33), uppercase (26) and digits (10) = 33 + 26 + 26 + 10 = 95

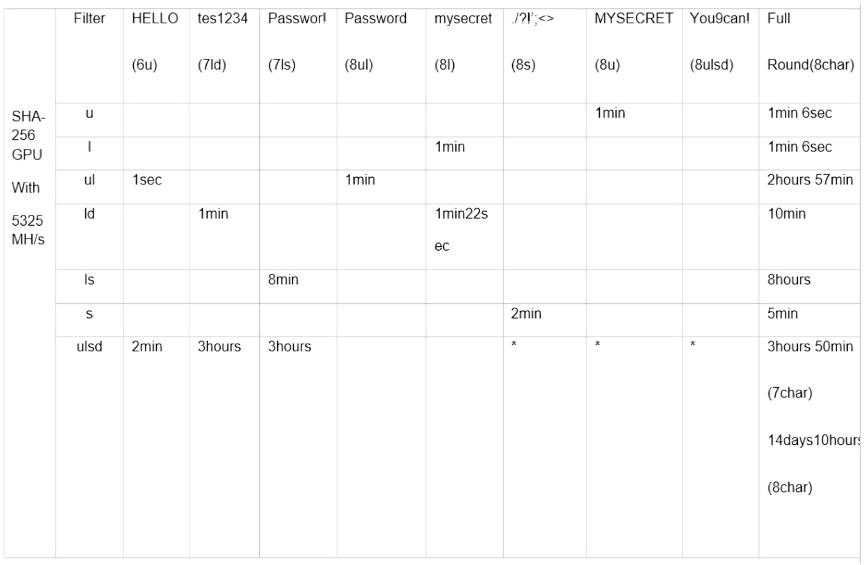
Total combination possible $[95]^8$ = 6.6342043e+15

SHA-256 CPU cracking speed, 88,954 KH/s = 88,954,000 h/s

Cracking time in days = 6.6342043e+15/88,954,000 h/s = 74580168.5466 s/3600 = 20716.7134852 hours/24 = 863 days

In Figure 13 the result is as follows:

- Using uppercase only character (u) to crack 8-character long password "MYSECRET" takes 1 min and it finishes checking all the combination of 8 characters roughly in 1 min 6 secs.
- Using lowercase only character (l) it takes 1 min to crack 8-character long password "mysecret" as well as finishes checking all the combination of 8 characters roughly in 1 min 6 secs.
- Using a combination of uppercase and lowercase character (ul) it takes 1 min to finish cracking 8 char long "Password" as well as finishes checking all the combination of 8 characters roughly in 2 hours 87 min.
- Using lowercase and digits (ld) it takes 1min to crack 7 char long "tes1234" password and 1 min 22 sec for 8-char password "mysecret", while it takes 10 mins to finish checking all the combination of the 8-char field.
- Using lowercase and special character (ls) it takes it takes 8 mins to find the 7-char password "Passwor!" while it will take approximately 8hours to finish checking all the combination.
- Using the special characters only it takes 2 min to finish cracking 8 char long password "./?!';<>" while it takes 5 min to finish checking all the special character combination of 8 char field.
- At last we try all the lowercase, uppercase, special characters and digits (usld). It took around 3 hours to finish checking 7 char length passwords while for all the 8 field of the password and for the 8-char long password (./?!';<>, MYSECRET, You9can!). Though we could not finish 8 char passwords, but we estimated it will take around 10 days and 10 hours to finish checking all the combination of the 8 char field. The estimated time is calculated as follows:



	Filter	HELLO (6u)	tes1234 (7ld)	Passwor! (7ls)	Password (8ul)	mysecret (8l)	./?!';<> (8s)	MYSECRET (8u)	You9can! (8ulsd)	Full Round(8char)
SHA-256 GPU With 5325 MH/s	u							1min		1min 6sec
	l					1min				1min 6sec
	ul	1sec			1min					2hours 57min
	ld		1min			1min22sec				10min
	ls			8min						8hours
	s						2min			5min
	ulsd	2min	3hours	3hours			*	*	*	3hours 50min (7char) 14days10hour (8char)

Figure 13. SHA-256 cracking performance with GPU machine.

Time Estimation for 8-char password

Total characters: lowercase (26) and special characters (33), uppercase (26) and digits (10) = 33 + 26 + 26 + 10 = 95

Total combination possible $[95]^8$ = 6.6342043e+15

SHA-256 GPU cracking speed, 5325 MH/s = 5,325,000,000 h/s

Cracking time in days = 6.6342043e+15/5,325,000,000 h/s = 1245859.96486/3600 = 346.0722 hours/24 = 14 days

In Figure 14 the result is as follows:

- Using uppercase only character (u) it takes 21 min to go through all combinations of 3 char length, whereas it takes 8 hours to finish 4 char long.
- Using uppercase and lowercase (ul) it takes 2 hours 21 min to finish 3 char length passwords. We could not finish cracking 4 char length passwords "Pass" as we estimated it will take 3 dys 17 hours to finish checking all the combination. The estimation is calculated as follows:

Time Estimation for 4-char password

Total characters: lowercase (26), uppercase (26) = 52

Total combination possible $[52]^4$ = 7,311,616

Bcrypt CPU machine cracking speed = 90 h/s

Cracking time in days= 7,311,616/90 = 81240.1777778 s/3600 = 2031.004 hr/24 = 22 hours 50 mins

- Using lowercase only character (l) it takes 3 hours 20 min to crack 4-character long password "Pass" and to finish all the combination of 4 chars it takes 8 hour where for 3 chars it takes 2 hour.
- Using only digits (d) it takes 9 min to crack 4 char passwords "1234" as well as roughly going through all the combination of 4-char field.
- Using only special characters(s) it takes 36 mins to finish checking all the combination of 3 char length password whereas it found 4 char length passwords ",/?<(" in 3 hours while taking 4 hours to go through all the combina-

tion of the 4-char length password.

- At last we try all the lowercase, uppercase, special characters and digits (usld). It takes 15 hours 4 mins to finish going through all the combination of the 3-char length. We could not finish checking for our 4-char length password "1!Su" as we estimated it will take 10 days 10 hours to check all the possible combinations using following formulas:

Time Estimation for 4-char password

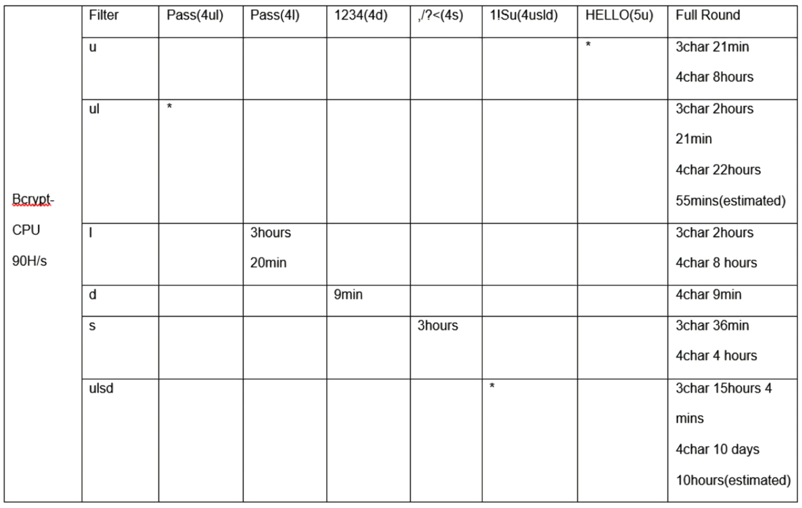
Total characters: lowercase (26), uppercase (26), Special characters (33), digits (10) = 95

Total combination possible $[95]^4 = 81{,}450{,}625$

Bcrypt CPU machine cracking speed = 90 h/s

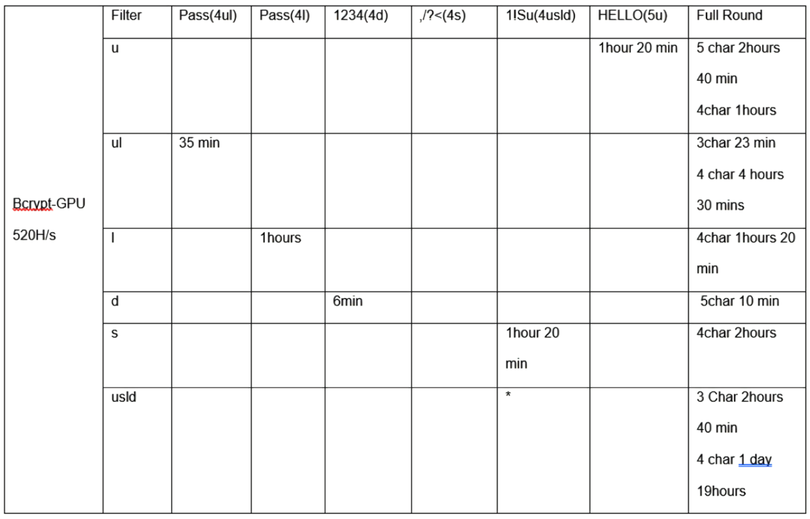
Cracking time in days = 81,450,625/90 = 905006.944444/3600s = 251.390817901 hr = 10 days

In Figure 15 the result is as follows:



	Filter	Pass(4ul)	Pass(4l)	1234(4d)	,/?<(4s)	1!Su(4usld)	HELLO(5u)	Full Round
Bcrypt-CPU 90H/s	u						*	3char 21min 4char 8hours
	ul	*						3char 2hours 21min 4char 22hours 55mins(estimated)
	l		3hours 20min					3char 2hours 4char 8 hours
	d			9min				4char 9min
	s				3hours			3char 36min 4char 4 hours
	ulsd					*		3char 15hours 4 mins 4char 10 days 10hours(estimated)

Figure 14. Bcrypt cracking performance with CPU machine.



	Filter	Pass(4ul)	Pass(4l)	1234(4d)	,/?<(4s)	1!Su(4usld)	HELLO(5u)	Full Round
Bcrypt-GPU 520H/s	u						1hour 20 min	5 char 2hours 40 min 4char 1hours
	ul	35 min						3char 23 min 4 char 4 hours 30 mins
	l		1hours					4char 1hours 20 min
	d			6min				5char 10 min
	s					1hour 20 min		4char 2hours
	usld					*		3 Char 2hours 40 min 4 char 1 day 19hours

Figure 15. Bcrypt cracking performance with GPU machine.

- Using uppercase only character (u) it takes 1hour to go through all combinations of 4 char length, whereas it takes 2 hours to finish 5 char length passwords. It found our 5-char length password "HELLO" in 1 hour 20 mins.
- Using uppercase and lowercase (ul) it takes 35 mins to crack 4 char length password "Pass". To go through all the combination of 3 char length passwords it takes 23 mins while to 4 char length passwords it takes 4 hours 30 mins only.
- Using lowercase only character (l) it takes 1hour to crack 4-character long password "Pass" and to finish all the combination of 4 chars it takes 1hour 20 mins only.
- Using only digits (d) it takes 6min to crack 4 char passwords "1234". It finished checking all the combination of 5 char length passwords in about 10 mins.
- Using only special characters(s) it took 1 hour 20 mins to crack 4 char passwords "1!Su" and going through all the combination in roughly about 2 hours.
- At last we try all the lowercase, uppercase, special characters and digits (usld). It takes 2 hours 40 mins to finish going through all the combination of the 3-char length. We could not finish checking for our 4-char length password "1!Su" as we estimated it will take 1-day 19 hours to check all the possible combinations using following formulas.

Time Estimation for 4-char password

Total characters: lowercase (26), uppercase (26), Special characters (33), digits (10) = 95

Total combination possible $[95]^4 = 81,450,625$

Bcrypt CPU machine cracking speed = 520 h/s

Cracking time in days = 81,450,625/520 = 156635.817308/3600 s = 43.5099492521 hr =1 day 19 hours

6.1. Experimental Run

- MD5 GPU with 1st character set as uppercase (U) and last character set as special characters(s) while all other character is combination of lowercase (l), uppercase (u), special character(s), digits (d) takes only 2 mins to finish whole 8-character set.
- MD5 GPU with last character fixed as special characters(s) and trying all other combination (lowercase (l), uppercase (u), special character(s), digits (d)) in first 7 character takes only 8 hours to finish.
- MD5 GPU trying all first 7 char as lowercase (l) and special character(s) whole last character fixed as special character(s) makes the cracking time of 8-character set to only 1 min.
- SHA-256 GPU machine cracking 8 characters with combination of lowercase (l), uppercase (u), special character(s), digits (d) in 2nd to 7th character while making the 1st character fixed for special characters(s) and 8th character

fixed for uppercase (u) brings the cracking time to only 6 mins while just making the 1st character fixed for uppercase(s) letters makes the cracking time around 10 hours.

6.2. Analyzing Result

- Using CPU instances with the combination of characters like uppercase, lowercase, special characters and digits a password length of 8 using MD5 hash takes 246 days to decrypt while using GPU it takes only 3 days.
- Similarly, the same password length using more secured SHA-256 hashing algorithm takes 863 days for our CPU machine to crack where with our GPU its only 10 days.
- Using more secured and computationally intensive Bcrypt hashing algorithm a password length of 4 characters only with a combination of characters like uppercase, lowercase, special characters and digits it takes our CPU instances 84 days to crack whereas with our GPU only 12 days.
- GPU instancing took only 5 min to crack SHA-256 password length of 8 with special characters only, whereas it took almost 3 hours to crack password length of 4 with Bcrypt hashing algorithm.

7. Recommendations

- Use password random generator to make a strong Radom password. One such sample random password generator script is given in Figure 16.

```
import random

Import string

str = [ ]

chars = string.ascii_letters + string.punctuation + string.digits

num = int (input(‘How long do you want the string to be?’))

for k in range(1,num+1):

str.append(random.choice(chars))

str = “”.join(str)
```

Figure 16. Alphanumeric Password generator script.

- Never reuse the same password for different accounts.
- As a security administrator or developer tries to use a modern hashing algorithm like Bcrypt which is slow in computing using GPU or CPU.
- Never store password without hashing.
- Always add salt to the hashed password for added security.
- Password length should be at least 10 characters in length and use combination of characters like uppercase, lowercase, special characters and digits and never use dictionary words.
- Avoid using words from dictionary which can be easily brute-force by dictionary attack.

8. Future Work

In the future, possible plan to test whether adding salt (random number) with MD5 and SHA-256 increase the cracking time it takes with GPU or not. Also, in this paper the effect of fixed certain character types in the password field with Bcrypt hashing algorithm like SHA-256 and MD5 hash was not tested. So, in the future endeavor test run with Bcrypt and compare results with SHA-256 and MD5.

9. Conclusion

In the end the aim of the paper was to compare the effectiveness of a GPU based, password cracking over the CPU as well the weakness in contemporary password hashing algorithm used (SHA-256, MD5) in today and why one should use a modern hashing algorithm like Bcrypt over SHA-256 and MD5 and why should use more complex passwords. We also came into conclusion that using salt with a hashed password adds more computational cost to crack even with highly capable GPU. We also presented a test bed scenario where a normal user can leverage the power of cloud computing to crack relatively complex password relatively easily.

Acknowledgements

Special acknowledgement to university of ST. Cloud State University and the guidance of Dr. Guster, Dr. Jim Chen and Dr. Mahbub from Information Assurance Department for there through support with this project.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] O'Gorman, L. (2003) Comparing Passwords, Tokens, and Biometrics for User Authentication. *Proceedings of the IEEE*, **91**, 2021-2040. https://doi.org/10.1109/JPROC.2003.819611

[2] Adams, A., Sasse, M.A. and Lunt, P. (1997) Making Passwords Secure and Usable. In: Thimbleby, H., *et al*., Eds., *Proceedings of HCI on People and Computers* XII, Springer-Verlag, London, 1-19. https://doi.org/10.1007/978-1-4471-3601-9_1

[3] (2016) MD5 Message Digest Algorithm Hash Collision Weakness. Securityfocus.com. http://www.securityfocus.com/bid/11849/discuss

[4] Qiu, W.D., Gong, Z., Guo, Y.D., Liu, B.Z., Tang, X. and Yuan, Y. (2016) GPU-Based High-Performance Password Recovery Technique for Hash Functions. ResearchGate. https://www.researchgate.net/publication/292761539_GPU-Based_High_Performance_Password_Recovery_Technique_for_Hash_Functions

[5] Thompson, C.J., Hahn, S. and Oskin, M. (2002) Using Modern Graphics Architectures for General-Purpose Computing: A Framework and Analysis. *Proceedings of the* 35*th annual ACM/IEEE International Symposium on Microarchitecture*, Istanbul, 18-22 November 2002, 306-317. https://doi.org/10.1109/MICRO.2002.1176259

[6] Cook, D.L., Ioannidis, J., Keromytis, A.D. and Luck, J. (2005) CryptoGraphics: Secret Key Cryptography Using Graphics Cards. *Cryptographers' Track at the RSA Conference*, 334-350.

[7] Yang, J. and Goodman, J. (2007) Symmetric Key Cryptography on Modern Graphics Hardware. *Advances in Cryptology*, 249.

[8] Di Biagio, A., Barenghi, A., Agosta, G. and Pelosi, G. (2009) Design of a Parallel AES for Graphics Hardware Using the CUDA Framework. *IEEE International Symposium on Parallel & Distributed Processing*, Rome, 23-29 May 2009, 1-8. https://doi.org/10.1109/IPDPS.2009.5161242

[9] Manavski, S.A. (2007) CUDA Compatible GPU as an Efficient Hardware Accelerator for AES Cryptography. *IEEE International Conference on Signal Processing and Communications*, Dubai, 24-27 November 2007, 65 p.

[10] Harrison, O. and Waldron, J. (2007) AES Encryption Implementation and Analysis on Commodity Graphics Processing Units. *Cryptographic Hardware and Embedded Systems CHES* 2007, 209 p. https://doi.org/10.1007/978-3-540-74735-2_15

[11] Bernstein, D., Chen, H.C., Cheng, C.M., Lange, T., Niederhagen, R., Schwabe, P. and Yang, B.Y. (2010) ECC2K-130 on NVIDIA GPUs. Progress in Cryptology-Indocrypt, 328-346.

[12] Hu, G., Ma, J. and Huang, B. (2010) High Throughput Implementation of MD5 Algorithm on GPU. *Proceedings of the* 4*th International Conference on Ubiquitous Information Technologies & Applications*, Fukuoka, 20-22 December 2009, 1-5.

[13] Mukherjee, R., Rehman, M.S., Kothapalli, K., Narayanan, P.J. and Srinathan, K. (2009) Presenting New Speed Records and Constant Time Encryption on the GPU. 3.

[14] Zhang, R. and Wang, X. (2011) MD5 Crack Method Based on Compute Unified Device Architecture. *Computer Science*, **38**, 302-305.

[15] Weng, J., Wu, Q. and Yang, C. (2011) OpenCL-Based MD5 Decryption Algorithm. *Computer Engineering*, **37**, 119-121.

[16] Nguyen, D.H., Nguyen, T.T., Duong, T.N. and Pham, P.H. (2010) Cryptanalysis of MD5 on GPU Cluster. *Proceedings of International Conference on Information Security and Artificial Intelligence*, Vol. 2, Chengdu, 17-19 December 2010, 910-914.

[17] Bauspiess, F. and Damm, F. (1992) Requirements for Cryptographic Hash Functions. *Computers, and Security*, **11**, 427-437. https://doi.org/10.1016/0167-4048(92)90007-E

[18] Anon (2016) Computing.dcu.ie.

http://www.computing.dcu.ie/~hamilton/teaching/CA642/notes/Hash.pdf

[19] Jose, R.T. and Thomas, C.G. (2015) A Comparative Study on Different Hashing Algorithms. *International Journal of Innovative Research in Computer and Communication Engineering*, **3**, 170-175.

[20] Menezes, A.J., Van Oorschot, P.C. and Vanstone, S.A. (1997) Handbook of Applied Cryptography. CRC, Boca Raton, 8, 14, 15.

[21] Reid, D. and Knipping, C. (2010) Proof in Mathematics Education: Research, Learning and Teaching.

[22] Florencio, D. and Herley, C. (2007) A Large-Scale Study of Web Password Habits.

[23] Hellman, M. (1980) A Cryptanalytic Time-Memory Tradeoff. *IEEE Transactions on Information Theory*, **26**, 401-406. https://doi.org/10.1109/TIT.1980.1056220

[24] Password Strength Checker (2016) Passwordmeter.com. http://www.passwordmeter.com/

[25] Mariger, H. (2016) Cognitive Disabilities and the Web: Where Accessibility and Usability Meet. Ncdae.org. http://ncdae.org/resources/articles/cognitive/

[26] Yang, Y., Lindqvist, J. and Oulasvirta, A. (2014) Text Entry Method Affects Password Security. Computing Research Repository. http://arxiv.org/abs/1403.1910

[27] Liu, Y. and Wu, E. (2008) Emerging Technology about GP-GPU. *Circuits and Systems Asia Pacific Conference*, Macao, 30 November-3 December 2008, 618-622.

[28] Rupp, K. (2016) CPU, GPU and MIC Hardware Characteristics over Time. https://www.karlrupp.net/2013/06/cpu-gpu-and-mic-hardware-characteristics-over-time/

[29] Vecchiola, C., Pandey, S. and Buyya, R. (2009) High-Performance Cloud Computing: A View of Scientific Applications. 10*th International Symposium on Pervasive Systems*, *Algorithms*, *and Networks*, Kaohsiung, 14-16 December 2009, 4-16.