Jurnal Masyarakat Informatika, 15(1), 2024
DOI:10.14710/jmasif.15.1.62545
ISSN: 2777-0648

Hash Value Security Improvement of PCS Password using Signed Binary Operation

M. Haikal Rahmadi*, Ary Bayu Nurwicaksono
Data Optimizer Team, Kecilin.id, Yogyakarta, Indonesia
* *Corresponding author: rahmadiihaikal@gmail.com*

Abstract
Convenience and security have always been inversely related requirements in data protection systems. Users want a short and simple password that is easy to remember. On the other hand, the system that is widely used in securing user data, especially passwords, is using a one-way message digest. In addition, users are also required to use complex passwords through a combination of letters, numbers, and symbols. It aims to increase security but a complex password will make it difficult for users to remember their passwords. Even though a complex password does not necessarily make it secure because it's still on a PCS which is vulnerable to hacking. Moreover, in the current development of cybersecurity science where password hacking systems are very easy to obtain and can be used by anyone to find hash value of password on PCS quickly. A preliminary test that has been carried out proves that even complex passwords can be hacked easily. This study proposes the use of a code extension system for passwords before the hashing process is carried out through two simple schemes C1 and C2 through bitwise xor and addition operators respectively. The code from the password data is mapped out of PCS by using a unique value of data. Experimental results show that the C1 scheme is able to thwart hacking attempts by 80%, while C2 is able to increase the security of alphanumeric passwords by up to 90%. The proposed method is able to make a simple but strong password system.

Keywords : *data security, password, code extension, message digest, hash value*

1 Introduction

Data security system using hashing encryption has become a major requirement in maintaining confidentiality, especially for user password data in a database. The popular algorithms used today are Message Digest (MD) and Secure Hash Algorithm (SHA) which are able to map a string into a hash value of a certain length. This method is one-way where the hash value cannot be decrypted, so that the password data cannot be known by anyone including the system owner. In general, users are required to use a long password using a combination of capital letters, numbers, and symbols to strengthen the security system [1]. On the other hand the use of such passwords will make it difficult for users to remember their passwords [2].

In fact, a security system with hashing and using complex passwords does not guarantee data security. The password entered by the user will still be in the Printed Character Set (PCS) which has a very limited number of characters that would be easy to crack through a password guessing such as Hashcat [3], [4] which are easily available on the internet. The password guessing method has 3 categories, namely brute force, dictionary and rainbow table [5]. Brute force is the method with the lowest effort from the people's point of view. The third party just needs to create a simple program to

Received: 8 March 2024; Revised: 8 May 2024; Accepted: 14 May 2024; Published: 31 May 2024.

try all possible password protecting data until he gets the correct password to unlock the data encryption. Another scheme called a dictionary attack, is almost similar to brute force, but a dictionary attack uses the help of a dictionary that contains a collection of words that are likely to be the password being searched for.

Many studies have explained the dangers of hash-solving systems such as Hashcat because it can be widely applied such as open source website, instant messaging applications [6], smart phone devices [7], cryptographic currency accounts [8] which can be done in a distributed manner [9]. Research on intelligent password cracking systems is also continuously being developed [10] which is supported by the speed of computer devices that will continue to increase twice every couple years. Brute force systems such as Hashcat use Printable Character Sets (PCS) as the main target in finding the hash value of data with low entropy values efficiently, it can even be done using only desktop or laptop computers that are widely owned most of people. Therefore, this article proposes the use of a pre-hash coding system by utilizing a unique id (UID) to encode the data before the hash process is carried out on the database. The purpose of using the system is to turn the data out of the PCS value so that it can increase the entropy value and can reduce the success rate of the brute force system on Hashcat significantly in low computational cost.

2 Password Security

Hashcat is one of wordlist based password guessing tools [11] which exploit both CPU and GPU [12]. It has a lot of hash mode operation to find the correct hash value of data and can be operated using a brute-force mode to find the uncommon or unusual password in PCS. Generally there are 15 type of pattern in password [12] which has different security level as shown in Table 1.

Table 1 Password Patterns

Pattern	Lower	Upper	Number	Symbol	Example
1	√				paswd
2		√			PASWD
3	√	√			PasWd
4			√		12345
5	√		√		pas12
6		√	√		PAS12
7	√	√	√		Pas12
8				√	&@$|)
9	√			√	pa$|)
10		√		√	PA$|)
11	√	√		√	Pa$|)
12			√	√	&@S12
13	√		√	√	pa$12
14		√	√	√	PA$12
15	√	√	√	√	Pa$12

Users can only use these patterns for their password. Even though the patterns are on the PCS which can be solved easily. Hashcat has 5-character sets that can be used to solve 15 existing password patterns through brute force mode as shown in Table 2.

Table 2 Character Sets

Set	Pattern Solved	Length
L	1	26
U	2	26
D	4	10
S	8	33
A	3, 5, 6, 7, 9 - 15	95

Hashcat is intended to help users to find their password when they forget their password and to test the security system that is being developed. However, it is mostly used by third parties to hack passwords stored in databases. This is usually done for malicious purposes such as personal data theft or another purpose such as learn or practice hacking. Hashcat has tremendous benefits as well as a very dangerous threat. System owners can't necessarily block it for security purposes because they also need it for system testing. The best solution is to take advantage of the unique value of each data stored in the database. A Unique value is generally used to find data index in database. This paper proposes using this value to prevent data hacking which will discuss in the next section.

3 The Proposed Method

This paper proposes a scheme to improve security of password data through a hash coding system with a UID value before the data is stored in the database as shown in Figure 1. The coding is done to map the PCS which have 95 values into Extended Character Set (ECS) within 256 values in 8-bit as follows:

1. Determination of the value of the initial number (seed) s that can be taken from the unique value of each user
2. Get the character length n from data/plaintext P
3. Generation of unique *U* code along n
4. Data encoding to get the *C* pre-hash code value with 2 alternatives as follows:

$$C1_n = P_n \oplus U_n \tag{1}$$

$$C2_n = P_n + U_n \tag{2}$$

The hash coding models in equations (1) and (2) are carried out through the bitwise xor operator and the addition mathematical operator respectively to the unique value *U*. Both coding systems have a low order so that they are able to map values to a maximum range of 8-bit values within low computational process and data storage. The encoded data will then be stored in an MD5 encrypted database. The security level test will be carried out using a brute-force attack with Hashcat on the P and C values and then compare the results to calculate the success rate of the proposed model. In addition, a comparison of the processing speed of the two variables will also be carried out to ensure that the proposed model is able to run fast.

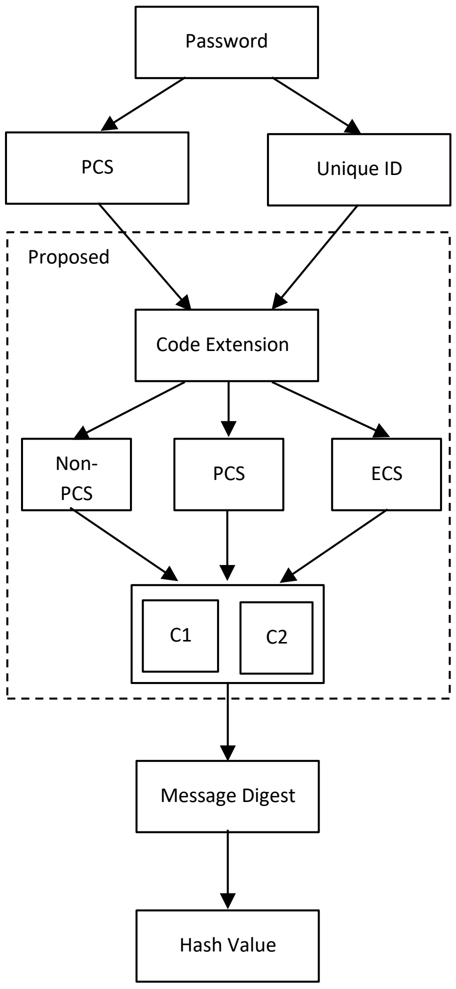

Figure 1 The Proposed Scheme

4 Results and Discussion

In the case of data hacking, hackers or sniffer generally copy the encrypted database through data stream or data sniffing and then perform the hack on the local device. This is done because it is almost impossible to do it directly into online systems that have been tightly protected, including firewalls and captcha. This research was conducted with a mid-end device with following specifications:

- Type	: MSI GL62M 7RDX
- Processor	: Core(TM) i7-7700HQ CPU @2.80GHz
- GPU	: GTX 1050
- RAM	: 4GB DDR4 @2400MHz
- OS	: Windows 10 Education 20H2

The first experiment is conducted to show the capability of Hashcat in term of cracking hash value of random password using pattern 15 which is mostly used in password requirement and considered as the strongest pattern today. Table 3 shows that all of passwords were cracked with the fastest time of 3.84 seconds, the slowest time was 9.03 seconds, and the average time was 5.92 seconds. The set of A requires large resources and slower computing times up to 3 times when compared to the LUD. However, the average time it takes to solve the hash value of a password is very short. This proves that the use of combinations of letters and numbers does not guarantee the security of data.

Table 3 Test Result on Pattern 15

Pass	**Hash**	**Status**	**Time**
^#px`	d8efce8c4ce9fe3d706ed963bc7182c0	Cracked	9.03
gfE^0	86e7768e52ed082ef8fa72cc5d9713a0	Cracked	5.90
c#:$)	b3f7408f4320b95f9f93613d99a17794	Cracked	5.97
nb>z#	dbcb37337e47e7a9bdd4ff81b97fde19	Cracked	5.90
IDhk1	a37609454de3ad188e83568a1c3bebea	Cracked	3.84
NJ]cg	29a14cc984edaf1d43bd2bec4ace49fc	Cracked	5.03
:`^/+	efacd0838bbd314e5444c75f24c96a99	Cracked	5.89
O{@W5	b10c89eef0ce68f31d1925137268e3b7	Cracked	6.46
g8Pbt	65911f939dd428dbfbe287423759a68c	Cracked	5.08
{S-.8	1bb4ad0148601dacb5614035b19fbbb7	Cracked	6.11

The next experiment was conducted to compare the performance of the C1 and C2 models in securing passwords. Table 4 shows the experiment result using 10 samples of password which has 5 character long. There are two samples that can be cracked and they are in alphanumeric character. Meanwhile the sample which include symbol tend to be secure. The use of bitwise xor operations with unique values will further map 6-bit values to 8-bit values. The use of a pre-hash system with bitwise xor operators will work better for passwords that include symbol characters. This is because 66% of symbol characters are in the 6-bit range while in alphanumeric characters only 15% are in that range.

Table 4 Test Result on Proposed Scheme C1

Pass	**Hash**	**Status**	**Time**
kunci	8b606753d26c92d6652aa63f4fb47ff2	Cracked	8.21
bunda	2a95870a1f1a19f24e94c561e7138eb4	Failed	9.02
13mei	71d1119e65858b9e2ca34cb183bb25ef	Failed	8.94
undip	abd41f112f7384d09cfea622c798e9ff	Failed	8.84
masuk	094b85a2de8f2fadce734064293d783b	Failed	8.82
US$10	1c1367a7e25d0921da132805993e1325	Failed	8.95
AIK21	b3bf7f10f3572719d3ebd0a77467316f	Cracked	5.78
i-l-u	09eefcce0b54eefa353d5681c46d7126	Failed	8.92
20/21	2157f9a1cafb96f3a7b4916d65ac7568	Failed	8.84
10_06	8a57b91f508382cef029482766d03ddb	Failed	8.95

In the second scheme, the test is done using a mathematical operator addition with equations (2) and obtained a better result as shown in Table 5. There is only one sample that can be cracked and it is symbol character while the entire sample with alphanumeric are failed to crack. This is in accordance with the workings of the addition operator to a unique value where the value in the 6-bit range will be more mapped to the 7-bit range that is still on the PCS while the value in the 7-bit range will be more mapped to 8-bit which is outside the PCS. 85% of alphanumeric characters are in the 7-bit range while in symbol characters only 33% of values are in that range. The comparison results of the two proposed methods are in Figure 2. shows that the performance of C2 is outperform C1. The test results also show that the threshold value between cracked and failed is in the range of 8.22 to 8.84 with an average time of 8.53. Generating a hash value that is longer than 8.5 seconds indicates that the value is approaching the initial limit of the ECS so that it has tend to be failure.

Table 5 Test Result on Proposed Scheme C2

Pass	Hash	Status	Time
kunci	c240f9ebc8e0de30307673bc05248cdd	Failed	11.16
bunda	49972a98d56b7a80889e3d7770ced008	Failed	9.19
13mei	84e8289aea4f26bcf13a87e4b60ddd71	Failed	8.87
undip	bf17d5dda706bac7e54c924921790ec1	Failed	8.86
masuk	983f4f10a003dccbf2e3d7d2f8d0c912	Failed	8.85
US$10	532da16868ff91f75a4eb0c97f00ceec	Failed	8.87
AIK21	beef6ddfd87a222aa6955b55205ea8d1	Failed	8.96
i-l-u	530209cff0d1d06f274613bcbcd19654	Failed	8.96
20/21	3980f5d8330e8cb8cfde0ec5d5f5c94d	Cracked	4.84
10_06	4d9223b5f03280c61eeaf2da388bb8cb	Failed	8.96

Table 6 Test Result on Proposed Scheme C2 using 6 Character Set

Pass	Hash	Status	Time
091298	47b1ed0ce923ebc0c76137431988cafb	Failed	297.37
haikal	00a07260a9152bc4948d7f1debbe33f6	Failed	291.63
999999	ef899ce05960dc1ecb84ded866175eef	Failed	289.43
undip1	cee526b1902f2b882e5ec5f22aeeaf42	Failed	289.38
123456	abe3c88315fd864f3877ef17d5dd34e7	Failed	291.50
IF2017	668f6408c489e1cb1b61e2d555970a8b	Cracked	56.33
mantab	ead0849165d86ae222ae93608fad150e	Failed	296.37
Kuat12	0652339865827a8e7005c9920ef7fd2b	Failed	324.30
pass	e801fb9e72b48c074278d439962747c9	Cracked	300.50
test_!	fc977c5debc570c07629bb0310c69fa9	Cracked	142.01

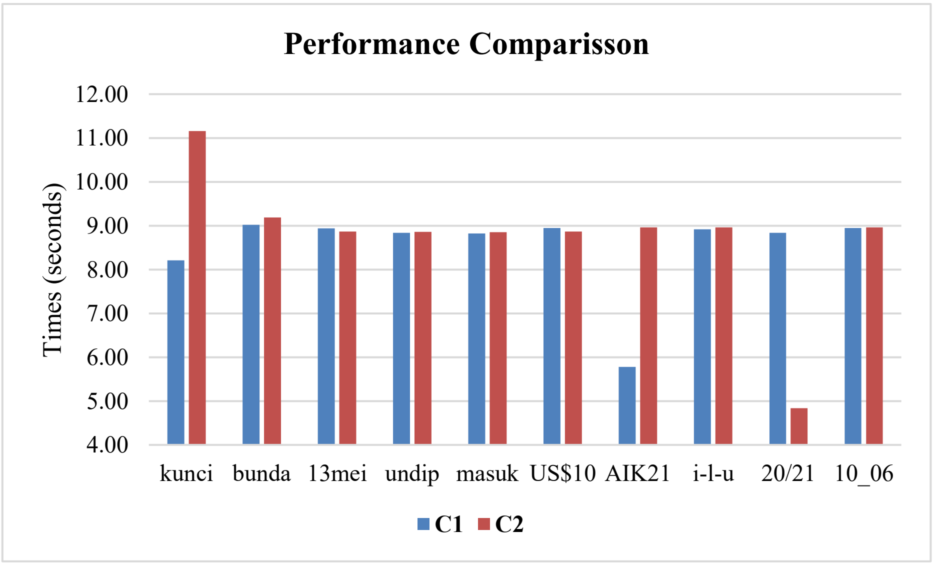

Figure 2 Performance Comparisson of the Proposed Methods on 5 Character Set

The last test was carried out to ensure the capability of the proposed method on longer password samples as shown in the Table 6. It is shows that one of alphanumeric and two samples with symbol are cracked. This result is in line with the previous test where C2 scheme can work better on alphanumeric passwords. A significant difference occurs in the processing time where a sample with 5 characters can be cracked in less than 10 seconds while for 6 characters it takes about 5 minutes.

5 Conclusion

Most of the user need a password that short and easy to remember. Meanwhile the system requirement forces the users to create a complex password for security reason. Even though complex passwords are not necessarily able to overcome current threats which often happens because it is easy to get a password guessing tool on the internet. This paper proposes a method that able to improve the security of short password. The password is mapped using the unique value outside the PCS, so it will be more difficult to crack. The experiment result shows that the proposed scheme C1 is able to improve the password security by 80% while scheme C2 successfully thwarted hacking on alphanumeric characters by 90%. The alphanumeric characters are used mostly current password because they are easy to remember. The experiments result shows that the proposed method is able to make a short and simple password more secure.

References

[1] Q. Guo *et al.*, "PUFPass: A password management mechanism based on software/hardware codesign," *Integration*, vol. 64, no. July 2018, pp. 173–183, 2019, doi: 10.1016/j.vlsi.2018.10.003.

[2] Y. Guo, Z. Zhang, and Y. Guo, "Optiwords: A new password policy for creating memorable and strong passwords," *Comput. Secur.*, vol. 85, pp. 423–435, 2019, doi: 10.1016/j.cose.2019.05.015.

[3] R. R. Asaad, "Penetration Testing: Wireless Network Attacks Method on Kali Linux OS," *Acad. J. Nawroz Univ.*, vol. 10, no. 1, p. 7, 2021, doi: 10.25007/ajnu.v10n1a998.

[4] B. Hitaj, P. Gasti, G. Ateniese, and F. Perez-Cruz, *PassGAN: A deep learning approach for password guessing*, vol. 11464 LNCS. Springer International Publishing, 2019.

[5] C. Ntantogian, S. Malliaros, and C. Xenakis, "Evaluation of password hashing schemes in open source web platforms," *Comput. Secur.*, vol. 84, pp. 206–224, 2019, doi: 10.1016/j.cose.2019.03.011.

[6] G. Kim, S. Kim, M. Park, Y. Park, I. Lee, and J. Kim, "Forensic analysis of instant messaging apps: Decrypting Wickr and private text messaging data," *Forensic Sci. Int. Digit. Investig.*, vol. 37, p. 301138, 2021, doi: 10.1016/j.fsidi.2021.301138.

[7] M. Park, G. Kim, Y. Park, I. Lee, and J. Kim, "Decrypting password-based encrypted backup data for Huawei smartphones," *Digit. Investig.*, vol. 28, pp. 119–125, 2019, doi: 10.1016/j.diin.2019.01.008.

[8] S. F. Dyson, W. J. Buchanan, and L. Bell, "Scenario-based creation and digital investigation of ethereum ERC20 tokens," *Forensic Sci. Int. Digit. Investig.*, vol. 32, p. 200894, 2020, doi: 10.1016/j.fsidi.2019.200894.

[9] R. Hranický, L. Zobal, O. Ryšavý, and D. Kolář, "Distributed password cracking with BOINC and hashcat," *Digit. Investig.*, vol. 30, pp. 161–172, 2019, doi: 10.1016/j.diin.2019.08.001.

[10] A. Kanta, I. Coisel, and M. Scanlon, "A survey exploring open source Intelligence for smarter password cracking," *Forensic Sci. Int. Digit. Investig.*, vol. 35, p. 301075, 2020, doi: 10.1016/j.fsidi.2020.301075.

[11] S. Mamonov and R. Benbunan-fich, "Computers in Human Behavior The impact of information security threat awareness on privacy-protective behaviors," *Comput. Human Behav.*, vol. 83, pp. 32–44, 2018, doi: 10.1016/j.chb.2018.01.028.

[12] A. Kanta, S. Coray, I. Coisel, and M. Scanlon, “Forensic Science International : Digital Investigation How viable is password cracking in digital forensic investigation ? Analyzing the guessability of over 3 . 9 billion real-world accounts,” *Forensic Sci. Int. Digit. Investig.*, vol. 37, p. 301186, 2021, doi: 10.1016/j.fsidi.2021.301186.