International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:4, No:10, 2010

How Efficiency of Password Attack Based on a Keyboard

Hsien-cheng Chou, Fei-pei Lai, Hung-chang Lee

***Abstract*—**At present, dictionary attack has been the basic tool for recovering key passwords. In order to avoid dictionary attack, users purposely choose another character strings as passwords. According to statistics, about 14% of users choose keys on a keyboard (Kkey, for short) as passwords. This paper develops a framework system to attack the password chosen from Kkeys and analyzes its efficiency. Within this system, we build up keyboard rules using the adjacent and parallel relationship among Kkeys and then use these Kkey rules to generate password databases by depth-first search method. According to the experiment results, we find the key space of databases derived from these Kkey rules that could be far smaller than the password databases generated within brute-force attack, thus effectively narrowing down the scope of attack research. Taking one general Kkey rule, the combinations in all printable characters (94 types) with Kkey adjacent and parallel relationship, as an example, the derived key space is about 2^{40} smaller than those in brute-force attack. In addition, we demonstrate the method's practicality and value by successfully cracking the access password to UNIX and PC using the password databases created

***Keywords*—**Brute-force attack, dictionary attack, depth-first search, password attack.

I. INTRODUCTION

A. Background

CURRENTLY the so-called brute-force attack is the most powerful method of attack among the various types of cryptanalytic attack, because brute-force attack is able to find out what the password is regardless of how it was set previously; however, the increase in password length means that the key space which brutal force attack must search through also grows proportionally to an exponential form, rendering this method inefficient. For example, the number of combination for a password length of 4 digits ranging from 0 to 9 is only 10^4; but once the length is increased to 8 digits ranging from 0 to 9, the number of combinations becomes 10^8, which is 10^4 times larger than the original 10^4. Based on this, one must try and reduce the size of password space in order to reduce the computational complexity in cryptanalysis.

H. C. Chou is with the Department of Computer Science and Information Engineering, National Taiwan University, Taiwan (e-mail: d96922034@csie.ntu.edu.tw).

F. P. Lai is with Graduate institute of Biomedical Electronics and Bioinformatics, National Taiwan University, Taiwan (e-mail: flai@ ntu.edu.tw).

H. C. Lee is with Department of Information Management, Tamkang University, Taiwan (e-mail: hclee@mail.im.tku.edu.tw).

B. Motivation

Dictionary attack [1, 2, 3, 4] can be considered as a simplified method of brute-force attack. It is based on the principle that people generally have a habit of using meaningful characters such as English names, nicknames, date of birth, and ID number in setting a password. By using the extensive amount of information collected on these types of characters during password cracking, it will greatly reduce the size of password space in dictionary attack and, therefore, have the better chance of finding the correct password.

This paper can also be regarded as another simplified version of brute-force attack. Although it shared a common feature of reduced password space with the dictionary attack, our scheme differs from the dictionary attack in that rather than using meaningful continuous characters as a search criteria, this method uses each key on a keyboard(Kkey, for short) and its adjacent/parallel relationship as search criteria. As meaningless continuous characters is not easily remembered and provide an necessity to prevent dictionary attack, the general users are bound to come up with other methods of generating meaningless continuous characters; key position on a keyboard provides a good choice, as one can generate a long list of seemingly meaningless character groups for as long as one remembers the starting position and movement trajectory.

This paper is designed around the concept of generating a password space using a keyboard, so that the generated password space will include the actual password that is created based on a keyboard, thus fulfilling the objective of simplifying the recovery password process. In order to achieve this objective, two major steps are designed in this paper. The first step involved the definition of keyboard rules that contains the various character combinations and the Kkey position among them. In Kkey position, two most commonly seen relationships are discussed, namely adjacent relationship and parallel relationship. Adjacent relationship refers to lowercase characters that are adjacent to one character, such as "r", "t", "y", "f", "h", "v", "b", and "n" to "g", while parallel relationship refers to character strings that are parallel to each other, such as "wsx", "edc", "`rfv", and so on that are parallel to "qaz". The second step used the keyboard rules defined in the first step to generate the required password databases according to user requirements.

C. Structure

The content of this paper is structured and organized as follows: Section II describes the relevant research done on

password attack. Section III explains the principles and methods of our scheme. Section IV contains the implementation and analysis of our scheme. Section V uses examples to describe the practical application of our scheme. Section VI contains the conclusions and future work.

II. Related Research

In recent years, the use of passwords for identity authentication or data encryption is becoming the most commonly used means of information security [5, 6, 7]. However, according to statistics from hacker websites, about 30% of users prefer to use their own familiar numbers or characters as a password, such as identity card numbers, date of birth, phone numbers, license plate numbers, and so on. There are even as high as 14% of users who use keys on a keyboard as passwords. In principle, if a user sets a password for the sake of convenience by following one of the above methods, it is very possible that he/she may be under the threat of hacker attacks.

Currently most identity authentication attacks through passwords are still based on either dictionary attack or brute-force attack; although other attack methods such as time-memory tradeoff [8] are gradually gaining importance as the computing performance and hard disk space continue to expand. At present, most identity authentication mechanism will first produce a 128-bit hash value through hash function, and the most famous hash functions are MD5 and SHA1. In time-memory tradeoff attacks, all possible passwords generated beforehand are stored in a hard disk using hash values produced by hash function. When the attacker guesses a password, all he/she has to do is comparing whether the hash value is correct. This will not only speed up the attack speed, but will also improve the chances of cracking a password. The hash values produced using this method is the famous Rainbow tables [9].

In addition, many commercial software development companies such as Elcomsoft [10], Passware [11], wwwhack [12], have developed powerful recovery password software against documents that use character string as data encryption, such as Word, Excel, PDF, RAR, ZIP encryption files. However, most of this recovery password software still bases its means of attack on dictionary attack or brute-force attack.

Most of the current dictionary attacks use a collection of commonly used characters in a variety of languages as the dictionary database, which contains mostly single type characters such as the lowercase characters, and rarely combinations of two or more characters, such as numbers plus lowercase characters. In view of the increasing importance attached to information security, users are bound to become more and more cautious in the choice of password, posing more and more challenges on the conventional dictionary-based means of attack.

Although the dictionary database generated by a keyboard can be described as a kind of dictionary attack, but the study has found that thus far there is virtually no trace of discussion on how to take advantage of this technique in guessed passwords in any paper or explored by any scholar. Most of the studies on a keyboard are confined to physical attack, namely an attacker analyzes the possible characters [13] the user entered by logging the key stroke sound.

For exploring the relevant techniques based on a keyboard, this paper uses adjacent and parallel relationships on a keyboard to generate password databases, and successfully applied them in the attached research on UNIX and PC access passwords.

III. Principles and Methods

A. Keyboard relationships

"Password attack based on a keyboard" is a methodology of guessed passwords using key position on a keyboard as a way in formatting password. The most common key position on a keyboard can be divided into adjacent and parallel relationship, which are explored and described in this paper as follows:

A.1 Adjacent relationship among keys on a keyboard

Definition 1. Adjacent relationship among keys means that two consecutive characters of keys corresponding to key position on a keyboard are adjacent.

Character strings such as *hjki* is the example of this adjacent relationship, as shown in Fig. 1.

Lemma 1. According to the definition 1, the characters of each key on a keyboard can contact with their adjacent characters.

For example, the character "s" contacts with "w", "e", "d", "x", "z", "a" characters which are adjacent to it, as shown in Fig. 1.

Fig. 1 Adjacent relationship among keys on a keyboard.

A.2 Parallel relationship among keys on a keyboard

Definition 2. Parallel relationship among keys means that *n* character strings (n>1) of keys corresponding to key position on a keyboard are parallel.

For example, the relationship between character string *qwe* and *asd* is a parallel one, and so is the relationship between character string *uhb* and *okm*, as shown in Fig. 2.

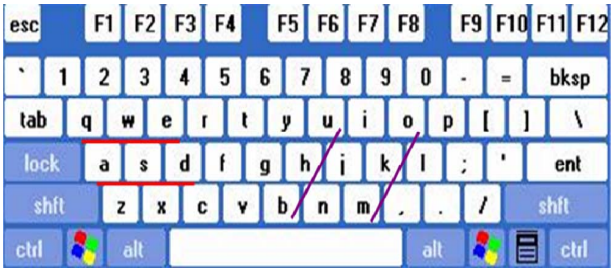

Fig. 2. Parallel relationship among keys on a keyboard.

B. Establishing key adjacent and parallel rules

Modern consumer electronics device such as desk-top computer, notebook, PDA and handset, have their own unique keyboard edition function, although the position of keys on those keyboards differ from each other slightly. To design a general system for these keyboards, we use two definition files to mark key information on a keyboard. The position file is to record relative position of each character and the symbol one is to record characters derived from each key. By using these two files, the system can then generate the key adjacent and parallel rules as described above. We use standard computer keyboard as an example, and the defined files (specifically, keyposition, keysymbol) as shown in TABLE 1 and TABLE 2.

TABLE 1 THE KEYPOSITION FILE

`1234567890-= qwertyuiop[]\ asdfghjkl;' zxcvbnm,./ 1qaz 2wsx 3edc 4rfv 5tgb 6yhn 7ujm 8ik, 9ol. 0p;/ -[' =]]'/ =[;. -pl, 0okm 9ijn 8uhb 7ygv 6tfc 5rdx 4esz 3wa 2q

The first row records right-left top-bottom characters. The second row records upper-left lower-right parallel characters. The third row records upper-right lower-left para-llel characters.

TABLE II THE KEYSYMBOL FILE

` ~	1 !	2 @	3 #	4 $	5 %	6 ^	7 &	8 *	9 (	0)	- _
= +	q Q	w W	e E	r R	t T	y Y	u U	i I	o O	p P	[{
] }	\ |	a A	s S	d D	f F	g G	h H	j J	k K	l L	; :
' "	z Z	x X	c C	v V	b B	n N	m M	, <	. >	/ ?	

B.1 Generating key adjacent rules according to adjacent relationship

Once we are in control of the adjacent relationship between each character on a keyboard, we can then go ahead and establish all possible password combinations based on this relationship. For example, the character "w" has two types on a keyboard, namely "w" and "W", and its adjacent characters include "2", "3", "q", "e", "a", "s", "@", "#", "Q", "E", "A", and "S" 12 possible characters. In terms of adjacent relationship among keys, if the previous character in the password is "w" or "W", then the next character may be one of the mentioned above characters, that is, if □ in character string aqw□ is the next character of "w", then its possible characters will include password combinations made up of numbers, lowercase characters, uppercase characters, and special printable characters, as shown in Fig. 3. It can be seen that, the more complex the user's password combination is, the more complex the relative rules based on key adjacency will be.

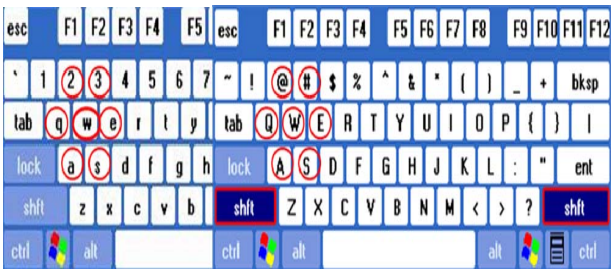

Fig. 3 Key adjacent rules on a keyboard

B.2 Generating key parallel rules according to parallel relationship

As far as parallel relationship on a keyboard is concerned, because at least two characters of relationship are required, one needs to move forward and look up at least one more character. For example, if the character before "w" is "q", then all horizontally arranged two characters from left to right are likely to be the next character group to emerge. Taking the 3-digit horizontal character string of *qwe* as an example, other likely character strings parallel to *qwe* include various combinations made up of numbers, lowercase characters, uppercase characters, and special characters, partial character strings as shown below.

123, 234, 345, 456, 567, 678, 789, 890	(numbers)
qwe, wer, ert, rty, tyu, yui, uio, iop	(lowercase characters)
asd, sdf, dfg, fgh, ghj, hjk, jkl	(lowercase characters)
zxc, xcv, cvb, vbn, bnm	(lowercase characters)
!@#, @#$, #$%, $%^, %^&, ^&*, &*(, *(), ()_,)_+	(special characters)
QWE, WER, ERT,RTY, TYU, YUI, UIO, IOP	(uppercase characters)
ASD, SDF, DFG, FGH, GHJ, HJK, JKL	(uppercase characters)
ZXC, XCV, CVB, VBN, BNM	(uppercase characters)

In addition to horizontal parallel, other parallel relationships on a keyboard include upper-left lower-right parallel and upper-right lower left parallel. Examples of 4-digit upper-left lower-right parallel characters include *1qaz*, *2wsx*, and *3edc*, etc, while *0okm*, *9ijn*, and *8uhb* are 4-digit upper-right lower-left parallel characters as shown in Fig. 4.

Fig. 4. Key parallel rules on a keyboard.

B.3 Combinations of character types on a keyboard

In Sections *A* and *B*, we introduced how to establish key rules through the method of key adjacent and parallel relationship, but in practice, there is still one issue that we need to consider when building key rules, that is combinations of character types. In standard computer keyboard, character types include numbers, lowercase characters, uppercase characters, and special characters. There are a total of 94 characters excluding the space key.

In order to improve flexibility in the building of key adjacent and parallel rules, the decision of combinations of character types has been given to the users, that is they are permitted to choose either one or more than two types of the above characters for the combination. In particular, if users have already observed a user encryption habit, it is then possible to build key rules by selecting the possible character types. Fifteen possible combinations of character types are listed in TABLE 3.

TABLE III COMBINATIONS OF CHARACTER TYPES ON A KEYBOARD

Item	Character combinations	Number
1	Number	10
2	Lowercase characters	26
3	Uppercase characters	26
4	Special characters	32
5	Numbers + lowercase characters	36
6	Numbers + uppercase characters	36
7	Numbers + special characters	42
8	Lowercase characters + uppercase characters	52
9	Lowercase characters + special	58
10	Uppercase characters + special	58
11	Numbers + lowercase characters + uppercase characters	68
12	Numbers + lowercase characters + special characters	68
13	Numbers + uppercase characters + special characters	68
14	Lowercase characters + uppercase characters + special characters	84
15	Numbers + lowercase characters + uppercase characters + special characters	94

C. Generating password databases

In Section *B*, we introduced how to create Kkey rules through key adjacent and parallel relationship. Once the rules are established, we can then generate password databases from the rules. Since the established rules can be regarded as a search tree, we may use depth-first search [14,15] or breadth-first search [14,15] to generate password databases.

In order to generate a set of password database that can best meet the actual needs of the users, the choice of combination for generating the password database will be given to the users as much as possible to maximize the flexibility of this method of attack. The choice items include:

a. Key relationship: adjacent or parallel relationship;
b. Password length: fixed or non-fixed password length;
c. First character: the starting character or an empty string;
d. Output format: the output to a file or directly contact callback function.

There are several features in allowing the users to set the mentioned above four options:

1. Key adjacent or parallel relationship will affect the size of password space created. According to Section 3.2, under same password length, the password space established through adjacent relationship is far greater than the one established through parallel relationship. It will increase the password space drastically if both are taken into consideration at the same time. Therefore, the password database is generated by letting the users to select either the adjacent or parallel relationship according to the flexibility of time and space.
2. If the users are familiar with the password length or likely range for guessing, password database can then be established through the method of setting password length or interval, as this can greatly reduce the password space and enhance the efficiency of an attack.
3. If users are aware of some information that relates to the password to be guessed, such as the password's first character is known, then such information can be utilized to reduce the complexity of the password database created and to reduce password search time.
4. Once a password database is created based on the user choice, it can either be exported to a specific file or feed directly into the target for verification. The benefit of the former is that one will only need to create the database once for the same set of conditions and such database can be reused continuously. The later is suitable to when the generated database is too large for the memory, but the database has to be regenerated each time.

IV. IMPLEMENTATION AND ANALYSIS

In Section III, we introduced the principles and methods in establishing password database through key adjacent and parallel relationship. In this section, the focus will be on actual program exploitation and experiment test. First of all, key rules must be created according to key adjacent and parallel relationship, which will then be used to generate password databases. The entirely diagram is shown in Fig. 5.

A. Establishing key rules according to the adjacent and parallel relationship

According to the adjacent and parallel relationship on a keyboard and combinations of character types, we exploit the program to establish the key rules. The algorithm is as follows:

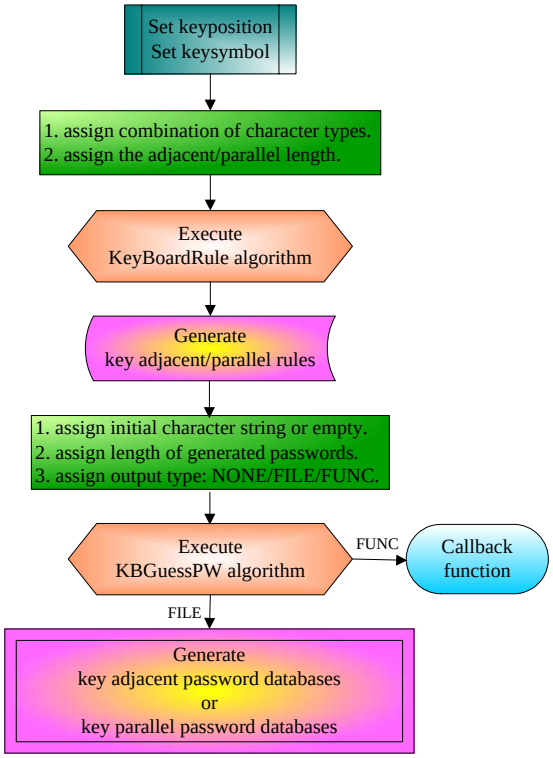

Fig. 5. The design diagram of our scheme.

KeyboardRule Algorithm. Establish the key adjacent/parallel rules.

Input:
- keyposition and keysymbol file.
- adjacent/parallel relationship.
- character length.
- combinations of character types:
  - 1: uppercase characters (uppercases, abbreviated as U), a total of 26 characters.
  - 2: lowercase characters (lowercases, abbreviated as L), a total of 26 characters.
  - 4: numbers (numbers, abbreviated as N), a total of 10 characters.
  - 8: other visible characters (others abbreviated as O), a total of 32 characters.

Output: the key adjacent/ parallel rules

```
KeySymbol ← keysymbol file
KeyPosition ← keyposition file
if (character length ==1) then
  Inference, InferenceSize ← key adjacent relationship in KeySymbol and KeyPosition
else
  Inference, InferenceSize ← key parallel relationship in KeySymbol and KeyPosition
end if
for i← 0 up to InferenceSize do
  while (Inference !=0)
    generate the key adjacent/parallel rules
  end while
end for
```

Let us use two examples to describe how key rules are generated through key adjacent or parallel relationship:

1. KeyboardRule.exe -n -f 7 -r rule.ne_NLU
   Use key adjacent relationship, numbers, lowercase characters, and uppercase characters to generate the key adjacent rule rule.ne_NLU.

2. KeyboardRule.exe -p 3 -f 1 -r rule.pa3_U
   Select parallel length of three characters together with uppercase characters according to key parallel relationship to generate the key parallel rule rule.pa3_U.

B. Generating password databases according to the key rules

When the key adjacent/parallel rules are established, we can then use these rules to generate password databases. The algorithm is as follows:

KBGuessPW Algorithm. Generate the password databases.

Input:
- key adjacent or parallel rules
- initial character string
- length of passwords generated
  - w: length of passwords generated equals to w.
  - W: length of passwords generated is less than or equal to W.
- type of exporting passwords.
  - NONE: do not export and use to calculate number of password set.
  - FILE: export to specified file.
  - FUNC: directly contact callback function.

Output: the password databases

```
rulefile ← key adjacent or parallel rule
initPat ← initial character string
PwdLen ← length of passwords generated
RulePattern, RuleSize ← rulefile
for i=0 up to RuleSize do
    run depth-first search by recursive function ← Rulepattern
end for
```

Let us also use two examples to describe the method of generating password databases through adjacent or parallel relationship:

1. KBGuessPW.exe -r rule.ne_NLU -W 8 –o FILE GuessPW_ne_NLU_W8
   Use the adjacent rule generated with numbers, lowercase characters, and uppercase characters to generate password database that contains length of equal or less than 8 characters, and output to the file GuessPW_ne_NLU_W8.

2. KBGuessPW.exe -r rule.pa3_U -W 5 –o FILE GuessPW_pa3_U_W5
   Use the 3-character parallel rule generated with uppercase characters to generate password database that contains length of equal or less than 5 characters, and output to the file GuessPW_pa3_U_W5.

C. Experiment results

C.1 Generating password databases through key adjacent relationship

To better explain the experiment results in the generation of password databases through key adjacent relationship, this paper has chosen 5 of character combinations out of 15 to generate a list of password databases, and these are U, NU,

NLU, ONL, and ONLU, with password length of between 5 to 16 characters. For better observation and comparison, the logarithm to the base 2 is used to represent the list of passwords as shown in TABLE 4, while a bar graph Fig. 6 is also prepared, where the horizontal axis indicates character length and the vertical axis indicates the number of passwords. From the analysis, it is found that as the password length increases, the resulted number of passwords also increases proportionally, especially when the character combinations are more complex, such as the case of ONLU, the derived number of passwords is also bigger. Take password length of 15 characters as an example, the number of passwords generated is between $2^{39.36}$ and $2^{56.8}$. Fig. 7 is a list of passwords generated using arbitrary relationship, namely the brute-force method. When we again take the password length of 15 characters as an example, the number of passwords generated is between 2^{70} and 2^{98}. As a result, it became obvious that even the number of passwords generated through key adjacent relationship remains large, it is already far smaller than the number of passwords generated through brute-force method.

TABLE IV THE NUMBER OF PASSWORDS GENERATED THROUGH KEY ADJACENT RELATIONSHIP

Pwd length	log 2 (number of guessing password combination)				
5	20.78	18.39	19.82	15.37	14.49
6	24.39	21.56	23.34	17.96	16.97
7	27.99	24.75	26.85	20.55	19.46
8	31.60	27.96	30.38	23.14	21.95
9	35.21	31.20	33.90	25.74	24.45
10	38.81	34.45	37.43	28.33	26.93
11	42.41	37.69	40.95	30.93	29.42
12	46.01	40.94	44.47	33.53	31.90
13	49.61	44.19	48.00	36.13	34.39
14	53.20	47.44	51.52	38.72	36.87
15	56.80	50.68	55.04	41.32	39.36
16	60.40	53.93	58.57	43.92	41.84

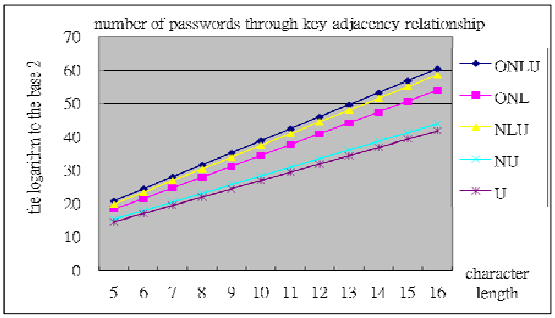

Fig. 6. The number of passwords generated through key adjacent relationship.

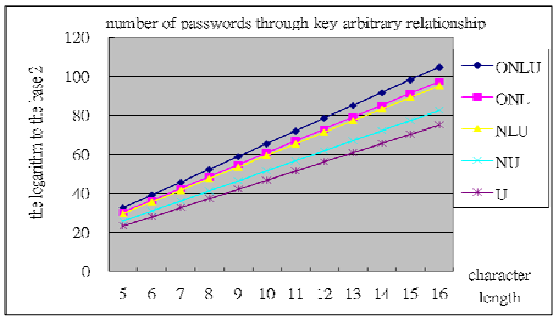

Fig. 7. The number of passwords generated through key arbitrary relationship.

C.2 Generating password databases through key parallel relationship

In our experiment, key parallel relationship of 2 parallel characters, 3 parallel characters, 4 parallel characters, and 5 parallel characters are chosen and used to establish parallel relationship key rules in character combination ONLU, which are then used to generate password databases respectively, as shown in Fig. 8. It should be noted that, 2 to 4 parallel characters include horizontal parallel, upper-left lower-right parallel, and upper-right lower-left parallel, while there is only horizontal parallel in 5 parallel characters. Therefore, the number of passwords generated is also relatively much less.

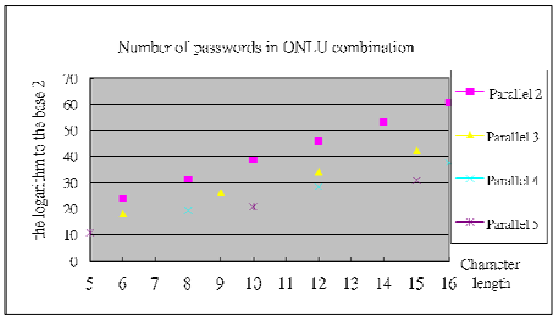

Fig. 8. The number of passwords generated with 2 to 5 parallel characters.

D. Comparative Analysis

As to the experiment results in Section *C*, the number of password database generated through key adjacent and 2 to 5 parallel characters relationship is compiled together with considerations of all extreme cases of ONLU combination of character of which the length is between 5 and 16 as shown in TABLE 5. A bar graph is also prepared as shown in Fig. 9. The figures also listed the number of resulted passwords under the conditions of no assumption, namely the so-called brute-force method. A few characteristics can be summarized by analyzing TABLE 5 and Fig. 9:

1. The number of password database established through key adjacent or parallel relationship is far smaller than that of the brute-force method. It is found that when character length equals 16, the number of passwords is reduced from $2^{104.87}$ to $2^{60.4}$; this figure can even be reduced further to $2^{37.53}$ if viewed from the point of 4 characters in parallel.
2. The number of passwords generated through adjacency and 2 characters in parallel are almost the same due to the fact that 2 characters in parallel can also be regarded as an adjacent relationship. The number of passwords generated through 5 characters in parallel is greatly reduced compared to 3 and 4 characters in parallel, as there is only horizontal parallel in 5 characters in parallel, while there are horizontal, upper-left lower-right, and upper-right lower-left parallel types in 3 and 4 characters in parallel.

If the extreme case in character combination ONLU is taken into consideration, the number of passwords generated through adjacency or 2 characters in parallel will have already exceeded 2^{40} when character length is longer than 10 characters, and even so, they are still far less than the number generated through brute-force method. Nevertheless, the number is significant. In order to reduce the number, one may choose the relatively simple character combinations, such as the single character U, N, L, and O, or the double characters such as UN, UL, LO and so on. In principle, there is a trade-off relationship between character length and character combinations, which may be a choice left for the users to make according to the actual needs.

TABLE V THE NUMBER OF PASSWORDS GENERATED BY ADJACENT AND PARALLEL RULES WITH ONLU COMBINATION, LENGTH IS BETWEEN 5 AND 16

ONLU	None (brute-force)	Adjacency	Parallel 2	Parallel 3	Parallel 4	Parallel 5
5	32.77	20.78				10.95
6	39.32	24.36	24.17	18.20		
7	45.88	27.99				
8	52.43	31.59	31.43		19.45	
9	58.99	35.28		26.22		
10	65.54	38.80	38.715			20.90
11	72.10	42.40				
12	78.65	46.00	46.01	34.34	28.44	
13	85.20	49.60				
14	91.76	53.20	53.34			
15	98.31	56.80		42.53		30.869
16	104.8	60.40	60.69		37.53	

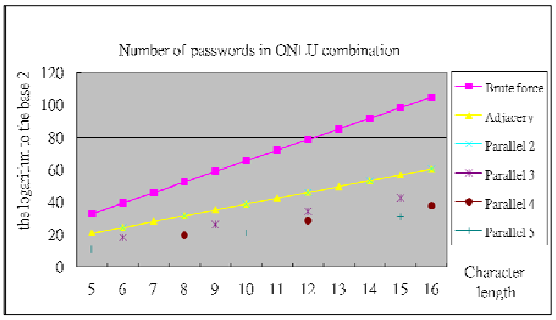

Fig. 9. The number of passwords generated by adjacent/parallel rules with ONLU combination, length is between 5 and 16.

V. PRACTICAL APPLICATION

The section describes how the password database generated through key position is used by this paper to successfully crack the universal passwords of UNIX and PC.

A. *UNIX universal password attack*

The unshadow function in John the Ripper software [16] is used to gather user account numbers and encrypted account passwords in UNIX, and the syntax is as follows:

```
unshadow /etc/passwd /etc/shadow > Usrpasswd
```

Usrpasswd content type is as follows:

```
root:xXjU16uYYiBVc:0:0:Super-User:/:/sbin/sh
test:j1iruP0Atc3E.:64002:64001:test:/home/test:/bin/bash
```

If semi-colon ":" is used as a separator symbol, the first field as the account number, such as root and test, and the second field as the encrypted account passwords, such as xXjU16uYYiBVc and j1iruP0Atc3E., this password is then encrypted through DES into a 13-byte encrypted password.

For better explanation, a set of functions provided by John the Ripper software is incorporated into the study for the attack, and syntax is as follows:

```
John.exe --wordlist=Word_p4_nl_W8 --rules Usrpasswd
```

Word_p4_nl_W8 is the password file generated through our method, meaning the 8-character password file generated through the combination of 4 characters in parallel, numbers, and lowercase characters, and the user account numbers and encrypted account passwords gathered by Usrpasswd. Execute the above syntax, and the program will stop execution and output the password as soon as the password is cracked. Through the above method, the account password for user account "test" is found to be *1qaz2wsx*, which is a password based on key parallel relationship and four characters.

B. PC universal password attack

The type of results received from using Pwdump software [17] to gather user's account numbers and encrypted account passwords in personal computers is as the following, represented in Usrpasswd name.

```
Administrator:500:8E4CC2363EA418C7AAD3B435B5
1404EE:302A4E27EEF7199FA75574956B9F8BA4:::
Joseph:1009:8E4CC2363EA418C7AAD3B435B51404E
E:302A4E27EEF7199FA75574956B9F8BA4:::
Test:1010:49D58563113416EBAAD3B435B51404EE:9
B77377DEE674B5106D13FCB626D5C40:::
```

The user passwords in a personal computer are 128-bit hash values produced through hash function. Once the user account name and encrypted account password of a certain PC are obtained, one can also try and guess the password using our method introduced in this study. Working with the John the Ripper software, the attack syntax is as follows:

```
John.exe --wordlist= Word_p3_nl_w6 --rules Usrpasswd
```

Word_p3_nl_W6 is the password file generated through our method, meaning the 6-character password file generated through the combination of 3 characters in parallel, numbers, and lowercase characters, and the user account names and encrypted account passwords gathered by Usrpasswd. Through the above method, the account password for user account "Joseph" is found to be *qweasd*, which is a password based on key parallel relationship and three characters.

VI. Conclusions and Future Work

Many papers on password attack have classified key arrangement as a method of guessed passwords and a type of dictionary attack. However, they have not discussed its technical methods, and nor have they produced any specific and usable password database except theoretical discussions.

This paper explores the actual techniques and experiment results in password attack based on a keyboard. By first establishing a corresponding key rule through key adjacent and parallel relationship and uses the rule to generate password database, this paper has successfully applied the method in cracking UNIX and PC account passwords. The contributions made by this paper can be grouped into three points:

1. Practical research: This paper proposes a complete technique for password attack based on a keyboard, through which one can establish the required key rules and provide the user with a flexible way of selecting and generating the needed passwords according to practical needs.
2. Design and implementation: According to our experiential results, we have confirmed that the password database generated is far smaller than that of brute-force method, effectively reducing the password space. Taking the combinations of all printable characters as an example, the password database generated in this study is about 2^{40} smaller than the one by brute-force method.
3. Application: The success achieved by this paper in cracking UNIX and PC passwords using the password databases generated based on a keyboard is a demonstration of this method's practicality.

At present, this password attack based on a keyboard developed by this paper can already be applied in cracking various passwords. In our experience, when password length exceeds 10 characters, the established password space will increase substantially. The future study will focus on the possibility of adding restriction rules in key adjacency in order to reduce the password space.

References

[1] Http://www.tech-faq.com/dictionary-attack.shtml.
[2] Password Cracking Wordlist. http: //www.openwall. com/wordlists/.
[3] Password Safe, http://passwordsafe. sourceforge.net/.
[4] Yahoo News. Favorite passwords: ‘‘1234’’ and ‘‘password’’, http:// news. yahoo.com, Feb 2009.
[5] Alain Forget, Robert Biddle, Memorability of persuasive passwords, CHI '08 extended abstracts on Human factors in computing systems, April 05-10, 2008, Florence, Italy.
[6] Mohammad Mannan, P. C. van Oorschot, Digital objects as passwords, Proceedings of the 3rd conference on Hot topics in security, p.1-6, July 29, 2008, San Jose, CA.
[7] Lorrie Faith Cranor, A framework for reasoning about the human in the loop, Proceedings of the 1st Conference on Usability, Psychology, and Security, p.1-15, April 14-14, 2008, San Francisco, California.
[8] Vrizlynn L. L. Thing, Hwei-Meng Ying, A novel time-memory tradeoff method for password recovery, June 2009.
[9] Project RainbowCrack website, http://project-rainbowcrack.com/.
[10] ElcomSoft password recovery tools, http://www.elcomsoft.com/.
[11] Password recovery software, http:// www.lostpassword.com/.
[12] Password recovery software, http:// www.wwwhack.com/.
[13] Lizuang, Feng Zhou, and J.D.Tygar, University of California, Berkeley, Keyboard Acoustic Emanations Revisited, ACM Transactions on Information and System Security, Vol. 13, No. 1, Article 3, October 2009.
[14] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, Introduction to Algorithm, 2nd Ed, 2001
[15] S. Russel and P. Norvig, Artificial Intelligence, a modern approach, 2nd Ed, 2006
[16] John the Ripper. Password cracker, http://www.openwall.com. LCPSoft. Lcpsoft programs, http://www.lcpsoft.com.
[17] Pwdump7 by Andres Tarasco Acuna, Windows NT family, up through XP or Vista. http://passwords.openwall.net/microsoft-windows-nt-2000-xp-2003-vista.

Hsien-Cheng Chou was born in Taipei, Tainan, R.O.C., in 1970. He received Master’s degree from the Department of Computer Science and Information Engineering, National Taiwan Normal University, Taipei, Taiwan in 2003. He is currently working toward the Ph.D. degree at the Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan.

He is currently an Engineer in National Central University. His current research interests include cryptography, cryptanalysis, parallel computing and software system.