International Journal of Innovative Computing, Information and Control
Volume **8**, Number **1(A)**, January **2012**

ICIC International ©2012 ISSN 1349-4198
pp. **387-402**

PASSWORD CRACKING BASED ON SPECIAL KEYBOARD PATTERNS

Hsien-Cheng Chou[1], Hung-Chang Lee[2], Chih-Wen Hsueh[1] and Fei-Pei Lai[1,3,4]

[1]Department of Computer Science and Information Engineering
[3]Graduate Institute of Biomedical Electronics and Bioinformatics
[4]Department of Electrical Engineering
National Taiwan University
No. 1, Sec. 4, Roosevelt Road, Taipei 10617, Taiwan
{ d96922034; cwhsheh }@csie.ntu.edu.tw; flai@ntu.edu.tw

[2]Department of Information Management
Tamkang University
No. 151, Yingjhuan Road, Danshuei, New Taipei 25137, Taiwan
hclee@mail.im.tku.edu.tw

Received September 2010; revised January 2011

Abstract. *Passwords are still the most commonly used mechanism for user authentication. However, they are vulnerable to dictionary attacks. In order to guard against such attacks, administrative policies force the use of complex rules to create passwords. One commonly used "trick" is to use keyboard patterns, i.e., key patterns on a keyboard, to create passwords that conform to the complex rules. This paper proposes an efficient and effective method to attack passwords generated from some special keyboard patterns. We create a framework to formally describe the commonly used keyboard patterns of adjacent keys and parallel keys, called AP patterns, to generate password databases. Our simulation results show that the password space generated using AP patterns is about* $2^{44.47}$ *times smaller than that generated for a brute-force attack. We also design a hybrid password cracking system consisting of different attacking methods to verify the effectiveness. Our results show that the number of passwords cracked increases up to 114% on average than without applying AP patterns.*

Keywords: Password cracking, Dictionary attack, Brute-force attack, Keyboard pattern

1. **Introduction.** Passwords for identity authentication or access control are still widely used means of ensuring system security despite the increased use of alternative techniques such as graphical passwords [1], smart-card or biometrics. However, these passwords are vulnerable to dictionary attacks [2]. In an attempt to force users into selecting strong passwords [3], system administration policies often regulate several complex rules for creating passwords. These rules might require some special characters, constrain minimum password lengths, and even forbid words from dictionaries. It is common users struggling to create passwords that meet these rules. In an effort to memorize these meaningless combinations, many users have resorted to alternative approaches such as typing the password based on the keyboard layout, i.e., the key positions and movements.

Take a password "!qaz@WSX#edc" for example. This 12-character sample password is at first glance a seemingly random string; it is actually generated utilizing the keyboard layout. It is clear that the user chooses an easy-to-remember approach forming a keyboard pattern starting with the "!" key down to the "z" key without the Shift key pressed, followed by a parallel same pattern starting with the "@" key with the Shift key pressed,

and then another similar parallel pattern starting with the "#" key down to the "c" key without the Shift key pressed.

In this work, we establish a framework to describe the commonly used keyboard patterns of adjacent keys and parallel keys, called AP patterns, to generate password databases. First, we provide a formal definition for AP patterns. Next, we build the AP patterns generating password databases in two major steps. The first step is the evolution of AP patterns based on various combinations of character types. The second step uses these patterns to construct password databases using the depth-first search. Finally, we design a hybrid password cracking system consisting of three sequential stages, i.e., Dictionary attack, AP-pattern attack and Brute-force attack, and successfully simulate and apply this system to crack UNIX and PC access passwords.

This paper is organized as follows. Section 2 describes related research on password attacks. Section 3 proposes AP patterns and password generation. Section 4 presents the simulation results. Section 5 demonstrates the effectiveness and comparison with other related researches. Section 6 contains conclusions and future work.

2. **Related Work.**

Techniques for cracking or acquiring passwords range from social engineering, phishing and shoulder surfing. However, most current identity authentication attacks (focusing on passwords) are still based on the dictionary attack or brute-force methods. Other attacks such as time-memory tradeoff [4,5] are gradually gaining importance as computing power and storage space increases. Nevertheless, the attack requires a lot of time and computing effort in order to build so-called Rainbow tables [6] and thus is not widely deployed. In addition, an effective defense has been mounted against Rainbow tables using the fact that some hash functions are combined with a random salt (for example, salted SHA-1), causing the same passwords to produce different hash values.

Actually, many companies such as Elcomsoft, Passware and Wwwhack, have developed powerful password recovery software for documents that use character strings for data encryption, such as Word, Excel, PDF, RAR and ZIP encryption files. However, most of these password recovery packages are still based on dictionary attack or brute-force attack.

Most keyboard-related attacks are limited to physical attack, i.e., an attacker analyzes the possible characters [7] by logging and analyzing the key stroke sounds. In 2001, Song et al. [8] presented a timing attack on the Secure Shell (SSH) network protocol. To perform this type of attack, a large amount of statistical training data is essential. Song performed a statistical analysis by timing the latencies between two keystrokes for all possible pairs of characters, and then predicted key sequences from the inter-keystroke timings. However, the need for pre-knowledge and user-specific statistical training models and data make the technique rarely practical.

Although the password database generated by keyboard patterns can be regarded as a kind of dictionary attack, there has been little discussion on the use of such a technique to crack passwords. In 2009, Schweitzer et al. [9] proposed an approach to collect and categorize keyboard patterns. By collecting a large number of users' passwords and analyzing keyboard patterns, they found that the keyboard patterns consisted of continuous 2-4 keys are most commonly used as passwords. Based on the analysis of Schweitzer, we define the frequently used adjacent and parallel keyboard patterns as AP patterns to generate password databases, and then successfully apply them to crack UNIX and PC access passwords.

3. **AP Patterns and Password Generation.**

AP patterns are composed of adjacent patterns and parallel patterns by the keys with adjacent relationships (adjacent keys) and

parallel relationships (parallel keys), respectively. The adjacent or parallel relationships for AP patterns are usually sensed visually. However, we provide formal definitions in the following sections. Although the relationships are discussed on a standard keyboard, we can extend them to any other input devices through two keyboard configuration files. After that, we describe the procedure to generate the passwords defined by AP patterns.

In general, all keys on a keyboard can be divided into the following three categories, printable keys (e.g., "a", "A", "1" and "!"), function transferring keys (e.g., Shift, Caps Lock and Ctrl) and other keys (e.g., arrow keys and Backspace) [10,11]. Without loss of generality, we focus only on printable keys because passwords are usually composed of only printable keys so that it is easier for users to type and remember and for the system to record.

Definition 3.1. *(Key coordinates) A key on a keyboard can be described as a polygon (usually square) and positioned with its upper-left most coordinate* (x_1, y_1) *and lower-right most coordinate* (x_2, y_2) *with reference to the keyboard's upper-left most coordinate:* $(0, 0)$.

For a standard keyboard, all printable keys are usually in a square shape of the same size. For convenience, we can define both the key length and width as 1, and ignore the horizontal and vertical gaps among keys. As shown in Figure 1, all these keys are arranged into four rows, with two continuous rows offset by δ $(0 < \delta < 1)$. Again, for convenience, we assume that these offsets for a standard keyboard are the same. This definition of key coordinates makes the following definitions of key relationships rather easy.

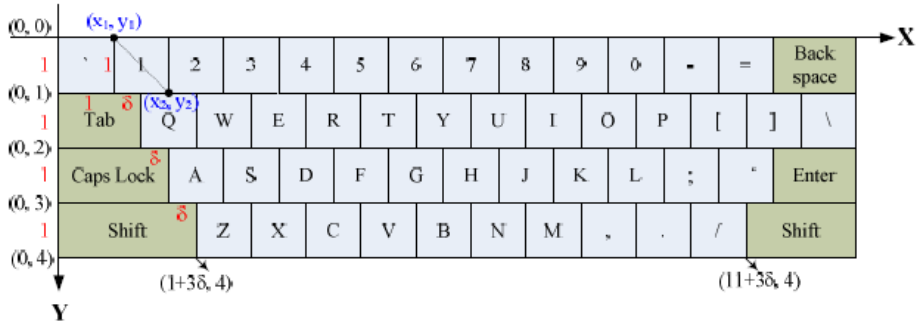


FIGURE 1. Key coordinates for a standard keyboard

3.1. **Adjacent relationship.** The first relationship considered visually among printable keys is the adjacent relationship to describe the adjacent keys. For any key, in addition to its obvious left adjacent and right adjacent keys, there are upper adjacent keys in the upper left and upper right directions, and lower adjacent keys in the lower left and lower right directions.

Definition 3.2. *(Adjacent keys) The adjacent keys of a key are the keys each of which and the key itself are adjacent.*

Using key coordinates, we can formally define the adjacent relationship as in the following example. Consider key K_1 with coordinates (x_1, y_1) and (x_2, y_2), and key K_2 with coordinates (x_3, y_3) and (x_4, y_4). If K_2 is upper adjacent to K_1, i.e., K_1 is lower adjacent to K_2, then $y_1 = y_4$ and $x_3 \leq x_1 \leq x_4$, or $y_1 = y_4$ and $x_3 \leq x_2 \leq x_4$, as shown in Figures 2(a) and 2(b). Note that any key is adjacent to itself.

We can obtain the center coordinates of each key by averaging the corresponding key coordinates. For example, the center coordinates of K_1 is $((x_1 + x_2)/2, (y_1 + y_2)/2)$, as shown in Figure 3(a). Therefore, the adjacent keys can be detected easily as in Lemma 3.1.

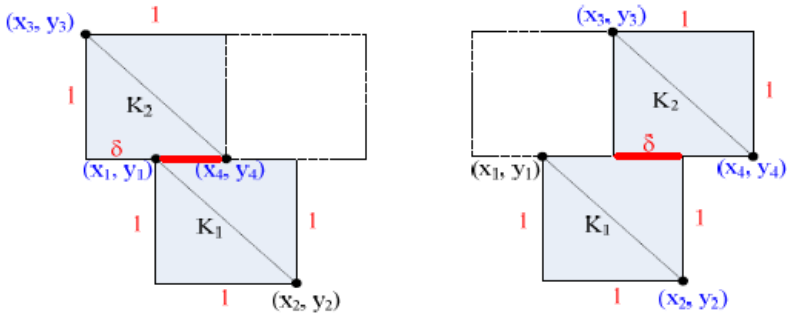


FIGURE 2. The upper adjacent and lower adjacent relationship of keys K_1 and K_2

Lemma 3.1. *Two keys are adjacent, if and only if the Euclid distance d between the centers of the keys is less than the length of key diagonal, $d < \sqrt{2}$, as shown in Figure 3(b).*

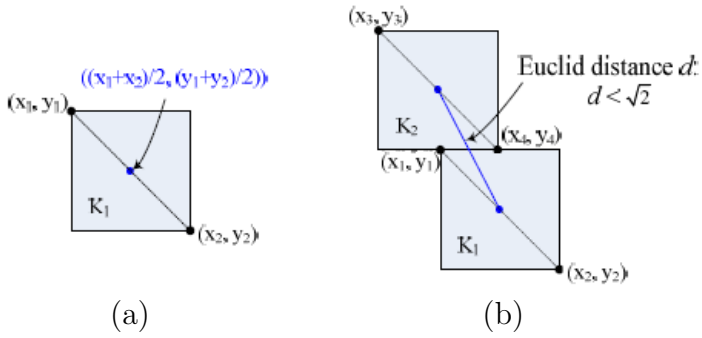


FIGURE 3. The center and Euclid distance for a standard keyboard

Lemma 3.2. *(Number of adjacent keys) The maximum number of adjacent keys for a key excluding itself is six.*

In despite of the value of offset δ, the maximum number of upper adjacent or lower adjacent keys is 2 because each key has the same size. Therefore, adding the unique left adjacent and right adjacent keys, the maximum number of adjacent keys for a key is 6. Taking a key "H" for example, keys "Y", "U", "J", "N", "B" and "G" are adjacent. However, for some keys as if in a border, their adjacent keys would be less than six. For example, the adjacent keys of "Z", are "A", "S" and "X", as shown in Figure 4.

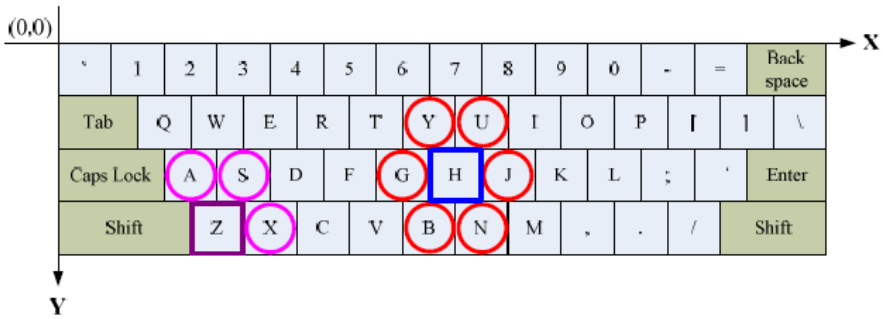


FIGURE 4. The adjacent keys of keys "H" and "Z"

3.2. **Parallel relationship.** Applying recursively the adjacent relationship defined above in the same direction, a special straight-line pattern of keys, key line, would form. By connecting the center of each key in the key line with a straight-line, the slope of the key line can be derived by the slope of the straight-line.

Definition 3.3. *(Key line) The keys adjacent in the same direction. The slope of the key line is the slope of the line connecting the centers of the keys in the key line. The length of the key line is the number of the keys in the key line.*

From Lemma 3.2, since a key might have 6 different adjacent keys, a key might branch out key lines at most in 6 different directions. Consider a standard keyboard: keys "1", "2", "3", "4" and "5" form a key line of zero slope and length 5 in the right direction; keys "Q", "A", "Z" form a key line of negative slope and length 3 in the lower-right direction; keys "0", "O", "K", "M" form a key line of positive slope and length 4 in the lower-left direction. The parallel relationship is considered among two key lines of the same slope. There are only three different slopes due to the same size of keys, as shown in Figure 5.

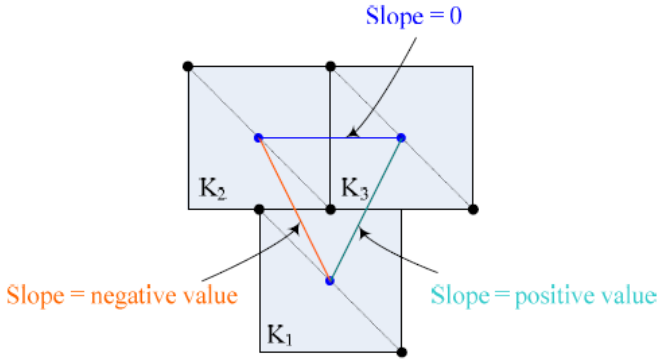


FIGURE 5. Three different slopes of key lines

Definition 3.4. *(Parallel keys) If keys are parallel, they are two parallel key lines of the same direction and length. The direction of the parallel keys is the direction of the key lines. The length of the parallel keys is the length of the key lines.*

Again, from Lemma 3.2, a key might branch out parallel keys at most in 6 different directions. For the parallel keys of negative, zero and positive slopes, we call them negative parallel, horizontal parallel and positive parallel, respectively. For example, parallel keys such as "456" and "ERT" are horizontal parallel, "QAZ" and "WSX" are negative parallel, and "UHB" and "OKM" are positive parallel. The parallel keys "654" and "TRE", "ZAQ" and "XSW", "BHU" and "MKO" are also horizontal parallel, negative parallel and positive parallel in the opposite direction, respectively, as shown in Figure 6.

The length of parallel keys varies according to the choice of its staring key. By the definitions above, for horizontal parallel, it ranges from 2 to 13 and, for negative parallel and positive parallel, it ranges from 2 to 4.

3.3. **Adjacent patterns and parallel patterns.** Modern consumer electronic devices, such as desktop computers, notebooks, PDAs and handsets, might have their own unique keyboard layout, although the positioning of keys on these keyboards might be only slightly different from each other. Since a printable key might generate different printable characters when it is typed together with other function transferring keys such as Shift key, before we establish AP patterns by the adjacent keys and parallel keys defined in

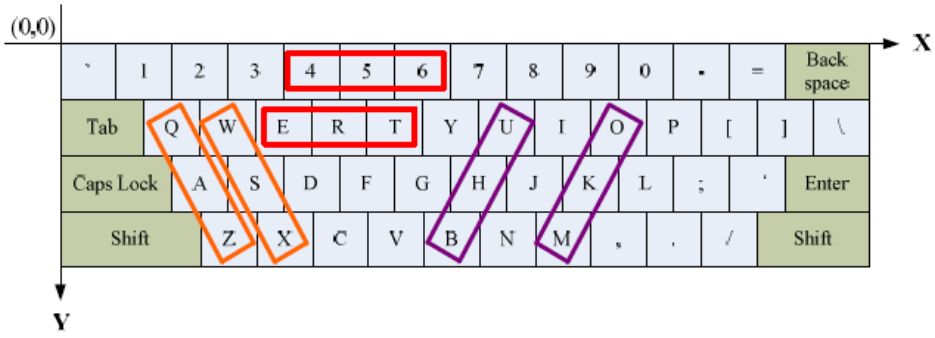


FIGURE 6. Parallel keys for a standard keyboard

previous sections, we need to formalize the mapping of a key and the characters it can derive.

To design a general system for generating passwords on these keyboards, we construct two configuration files to describe the key arrangement on a keyboard. A key-position file is used to record the relative position of each key, and a key-symbol file is used to record possible characters derived from each key. By using these two files, the system can then generate the AP patterns of adjacent keys and parallel keys, which will be described in Sections 3.3.1 and 3.3.2, respectively. The key-position and key-symbol files for a standard computer keyboard are shown in Tables 1 and 2.

TABLE 1. The key-position file for a standard keyboard

```
`1234567890-=
qwertyuiop[]\
asdfghjkl;'
zxcvbnm,./
```

Table 1 shows the layout of 47 printable keys arranged into four rows with representing characters when the keys are typed without Shift key pressed, where the offset δ is ignored by missing any space in the beginning of lower 3 rows.

TABLE 2. The key-symbol file

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

The 94 printable characters shown in Table 2 derived from the 47 printable keys can be classified into four commonly-used types, i.e., Numbers (N), Lowercases (L), Uppercases (U) and Others (O). Fifteen possible combinations of these types are listed in Table 3. With the keyboard layout defined by the two configuration files, each combination defines the set of characters which can be used with the adjacent keys or parallel keys to establish AP patterns. Actually, each combination represents a kind of user preferences. The AP patterns are the sets of possible next string to append as potential passwords indexed by current key or key line according to the adjacent keys or parallel keys of the combinations, respectively. Note that the length of the next string to append is 1 for adjacent patterns and the length of the parallel keys for parallel patterns.

TABLE 3. Combinations of printable character types

Index	Description of Combination	Number of Characters	Abbreviation
1	Numbers	10	N
2	Lowercases	26	L
3	Uppercases	26	U
4	Others	32	O
5	Numbers + Lowercases	36	NL
6	Numbers + Uppercases	36	NU
7	Numbers + Others	42	NO
8	Lowercases + Uppercases	52	LU
9	Lowercases + Others	58	LO
10	Uppercases + Others	58	UO
11	Numbers + Lowercases + Uppercases	68	NLU
12	Numbers + Lowercases + Others	68	NLO
13	Numbers + Uppercases + Others	68	NUO
14	Lowercases + Uppercases + Others	84	LUO
15	Numbers + Lowercases + Uppercases + Others	94	NLUO

3.3.1. *Establishing adjacent patterns.* Now we are ready to establish adjacent patterns based on adjacent keys and a set of the combinations in Section 3.3. Once the combinations are chosen, the adjacent patterns can be established as an array of adjacent keys indexed by each character in the combinations. For example, as shown in Figure 7 based on the NLUO combination, characters "w" and "W" derived by the same key "w/W" have adjacent keys including "2", "3", "e", "s", "a", "q", "w", "@", "#", "E", "S", "A", "Q" and "W", 14 possible next keys of Numbers, Lowercases, Uppercases, or Others types of printable characters. Table 4 and Table 5 show the adjacent patterns of simpler combinations L and NU indexed by 26 and 36 characters, respectively. We can see that the more complex character combinations adopted, the more complex adjacent patterns could be established.

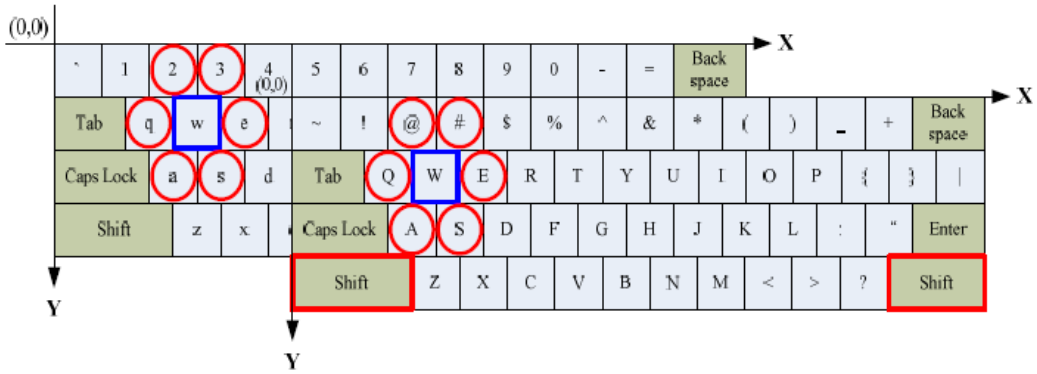


FIGURE 7. The adjacent keys of the key "w/W"

3.3.2. *Establishing parallel patterns.* Similar to establishing adjacent patters, we can establish parallel patterns based on parallel keys and a set of the combinations in Section 3.3. Once the combinations are chosen, the parallel patterns of a given length can be established. The parallel patterns are an array indexed by different parallel relationships

TABLE 4. The adjacent patterns of uppercases (U)

Key	Next Keys	Key	Next Keys	Key	Next Keys
A	A S Q Z W	J	J H K U M I N	S	S A D W X E Z
B	B V N G H	K	K J L I O M	T	T R Y G F
C	C X V D F	L	L K O P	U	U Y I J H
D	D S F E C R X	M	M N J K	V	V C B F G
E	E W R D S	N	N B M H J	W	W Q E S A
F	F D G R V T C	O	O I P L K	X	X Z C S D
G	G F H T B Y V	P	P O L	Y	Y T U H G
H	H G J Y N U B	Q	Q W A	Z	Z X A S
I	I U O K J	R	R E T F D		

TABLE 5. The adjacent patterns of numbers and uppercases (NU)

Key	Next Keys	Key	Next Keys	Key	Next Keys
0	0 9 P O	C	C X V D F	O	O I P 9 L 0 K
1	1 2 Q	D	D S F E C R X	P	P O 0 L
2	2 1 3 W Q	E	E W R 3 D 4 S	Q	Q W 1 A 2
3	3 2 4 E W	F	F D G R V T C	R	R E T 4 F 5 D
4	4 3 5 R E	G	G F H T B Y V	S	S A D W X E Z
5	5 4 6 T R	H	H G J Y N U B	T	T R Y 5 G 6 F
6	6 5 7 Y T	I	I U O 8 K 9 J	U	U Y I 7 J 8 H
7	7 6 8 U Y	J	J H K U M I N	V	V C B F G
8	8 7 9 I U	K	K J L I O M	W	W Q E 2 S 3 A
9	9 8 0 O I	L	L K O P	X	X Z C S D
A	A S Q Z W	M	M N J K	Y	Y T U 6 H 7 G
B	B V N G H	N	N B M H J	Z	Z X A S

(negative parallel, horizontal parallel and positive parallel) of parallel keys of the same length based on the combinations. Since parallel keys contain two parallel key lines, an array element in the parallel patterns contains all the possible key lines parallel to each other in the same direction, i.e., the possible next strings. Only the right, upper right and lower right directions are established. The other left, lower left and upper left directions can be established on-line by the same parallel patterns reversed, respectively. Some parallel key lines of length 3 with respect to "qwe" based on different character combinations are shown as follows.

Combination	Parallel Key Lines
L	qwe, wer, ert, ..., iop, asd, sdf, ..., jkl, zxc, xcv, ...
NL	123, 234, 345, 456, 567, 789, 890, qwe, wer, ert, ...
LU	qwe, wer, ert, rty, ..., bnm, Qwe, Wer, ..., Bnm, ..., QWE, ERT, ...
LO	qwe, wer, ert, rty, tyu, ..., op{, p{}, ..., bnm, ~!@, !@#, @#$, ...
NLO	`12, !23, @34, #45, $56, ^78, &89, * 90, (0–,)–=, qwe, wer, ..., 0–=, ...

Based on the character combination Uppercases, the whole parallel patterns with length 3 is shown in Table 6, where each string in its set of next key lines is indexed to the set itself. Based on the basic 15 character combinations, we establish the parallel patterns

with length 2-5 and list the numbers of next key lines for different parallel relationships in Appendix A.

TABLE 6. The parallel patterns of uppercases (U) with length 3

Parallel Relationship	Next Key Lines
horizontal parallel	QWE WER ERT RTY TYU YUI UIO IOP ASD SDF DFG FGH GHJ HJK JKL ZXC XCV CVB VBN BNM
negative parallel	QAZ WSX EDC RFV TGB YHN UJM
positive parallel	OKM IJN UHB YGV TFC RDX ESZ

3.4. **Generating password databases.** Once the AP patterns are established, we can then generate passwords one by one from an initial string as current string to find the possible next string to append on the password. The possible next stings can be found in the AP patterns in the set indexed by the current string. The next string will then be the new current string to apply on the AP patterns again to find the next string continuously until the length of the password is satisfied. The initial string should be empty or one of the next keys or next key lines in the adjacent patterns or parallel patterns, respectively. If the initial string is empty, all the next keys and next key lines are the possible next stings. If the initial sting is a single character for parallel patterns, all the next key lines starting with the single character are the possible next strings. All possible passwords can thus be found by depth-first search [12,13]. For example, the partial search tree for the adjacent patterns of Uppercases as in Table 4 is shown in Figure 8. Based on the search tree, we can generate passwords with length 3, i.e., AAA, AAS, AAQ, AAZ, AAW, ASS and ASA.

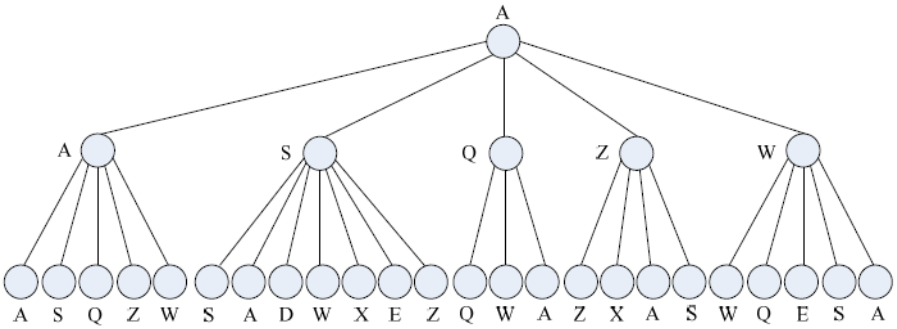


FIGURE 8. The partial tree by the adjacent patterns based on Uppercases (U)

To generate the password databases, the input parameters include:

A. Keyboard patterns: adjacent patterns or parallel patterns;
B. Password length: a fixed number or a range of numbers;
C. Initial string: a starting character or an empty string;
D. Output format: off-line databases or on-line message.

There are several guidelines in setting the parameters:

A. The choice of adjacent patterns or parallel patterns affects much the password space generated. It increases drastically if both are considered at the same time.
B. If users would know the possible password length or can guess a likely range, the password database can be generated by setting the password length or range; this can greatly reduce the password space, so does the search time and enhance attack efficiency.

C. If users would know some information that relates to the password to be cracked, such as the first character, this information can be utilized as the initial string to reduce the password space as well.
D. The passwords can be generated off-line as databases for future reuse or on-line for verification directly because the space is usually too large to store in memory.

4. Simulation Results.

4.1. Password databases generated using adjacent patterns. Based on the 15 basic character combinations in Section 3.3, we generate all passwords using adjacent patters with empty initial string and length range 5-16 off-line into databases of different lengths. We show the numbers of passwords generated of frequently used character combinations U, NU, NLU, NLO and NLUO in Table 7 and the same data in Figure 9 in the logarithm to base 2. As the password length increases, the resulting number of passwords also increases proportionally (in logarithm), especially when character combinations are more complex, such as the case of NLUO, the number of passwords also increase. For example, the number of passwords with length 15 generated is between $2^{39.36}$ and $2^{56.8}$.

TABLE 7. Numbers of passwords generated using the adjacent patterns

Password length	Numbers of passwords (Logarithm to base 2)				
	NLUO	NLO	NLU	NU	U
5	20.78	18.39	19.82	15.37	14.49
6	24.36	21.56	23.34	17.96	16.97
7	27.99	24.75	26.85	20.55	19.46
8	31.59	27.96	30.38	23.14	21.95
9	35.28	31.20	33.90	25.74	24.45
10	38.80	34.45	37.43	28.33	26.93
11	42.40	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

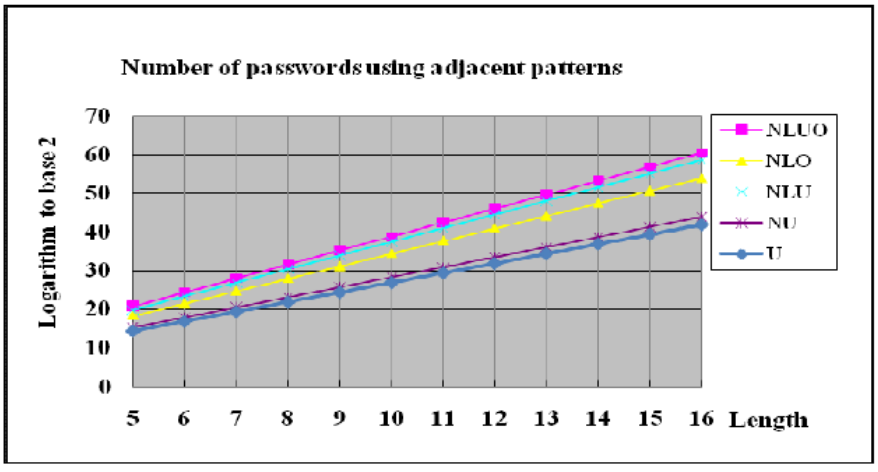


FIGURE 9. Number of passwords generated using the adjacent patterns

4.2. **Password databases generated using parallel patterns.** Similar to Section 4.1, we generate passwords using parallel patterns with empty initial string and length range 5-16 off-line into databases of different lengths. The same frequently used character combinations U, NU, NLU, NLO and NLUO are also chosen and the results are shown in Figure 10, where "parallel n" means parallel patterns with length n are adopted. When parallel 2 is adopted, only passwords with lengths 6, 8, 10, 12, 14 and 16 can be generated. So do parallel 3, parallel 4 and parallel 5 with lengths 6, 9, 12 and 15, lengths 8, 12 and 16, and lengths 5, 10 and 15, respectively. Noted that not all parallel relationships, horizontal parallel, negative parallel and positive parallel, are present in the passwords generated as shown in Appendix A. Therefore, the numbers of passwords generated with parallel 5 is smaller.

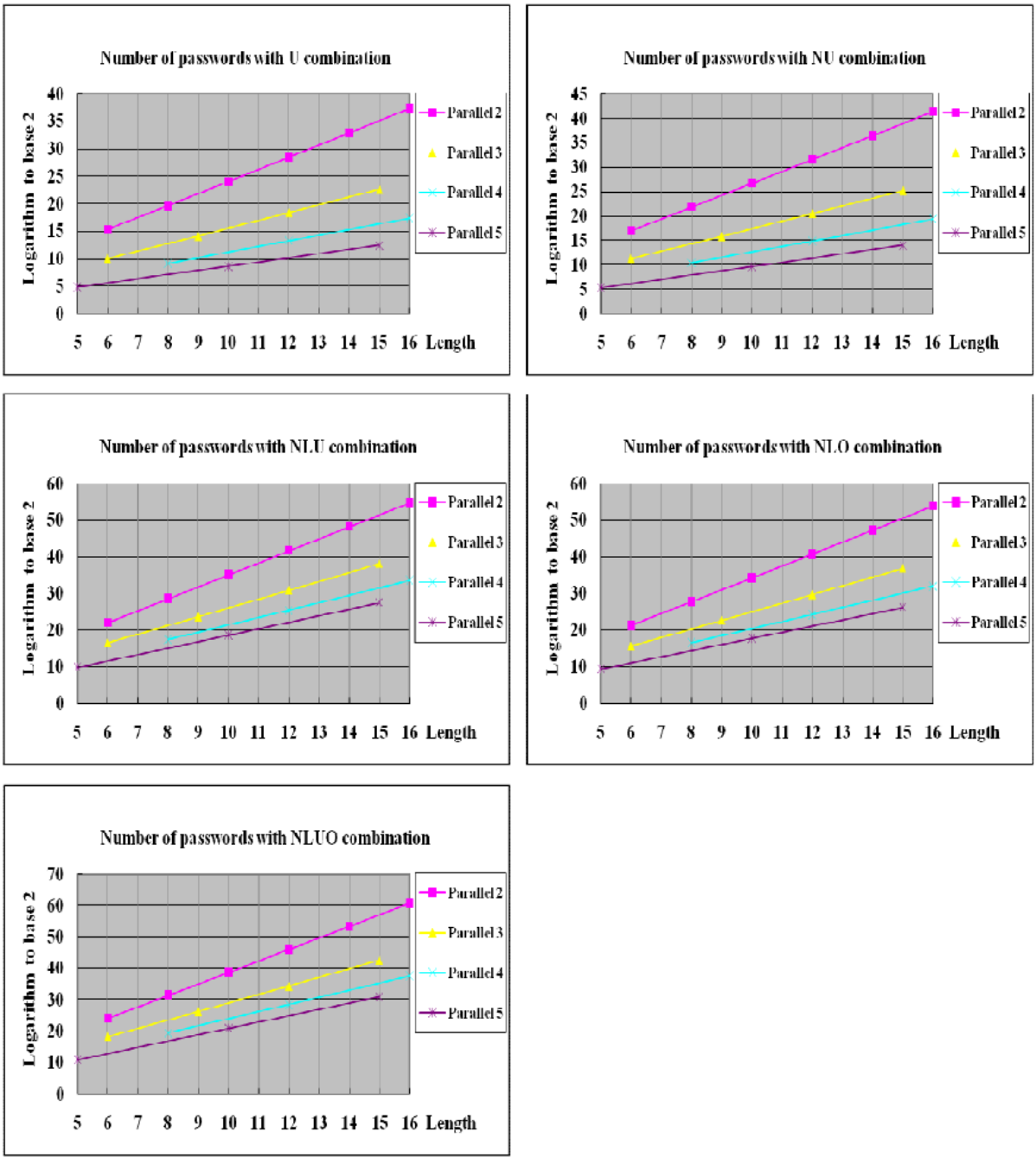


FIGURE 10. Numbers of passwords generated using parallel patterns

4.3. **Experimental observations.** As shown in Figure 11, we combine the experimental results of adjacent patterns (by legend Adjacency) and parallel patterns with all possible passwords generated by the brute-force method of the same character combinations. All the data for passwords with length 16 in Figure 11 are also compared in Figure 12. A few observations were made from Figure 11 and Figure 12 as follows.

1. As shown in Figure 11, the number of passwords generated using AP patterns is far smaller than that of the brute-force method. From Figure 12, for the passwords with length 16 based on character combination NLUO, the number of passwords is reduced from $2^{104.87}$ of brute-force method to $2^{60.4}$ of adjacent patterns by $2^{44.47}$ times. The number is even further reduced to $2^{37.53}$ by parallel 4 patterns.

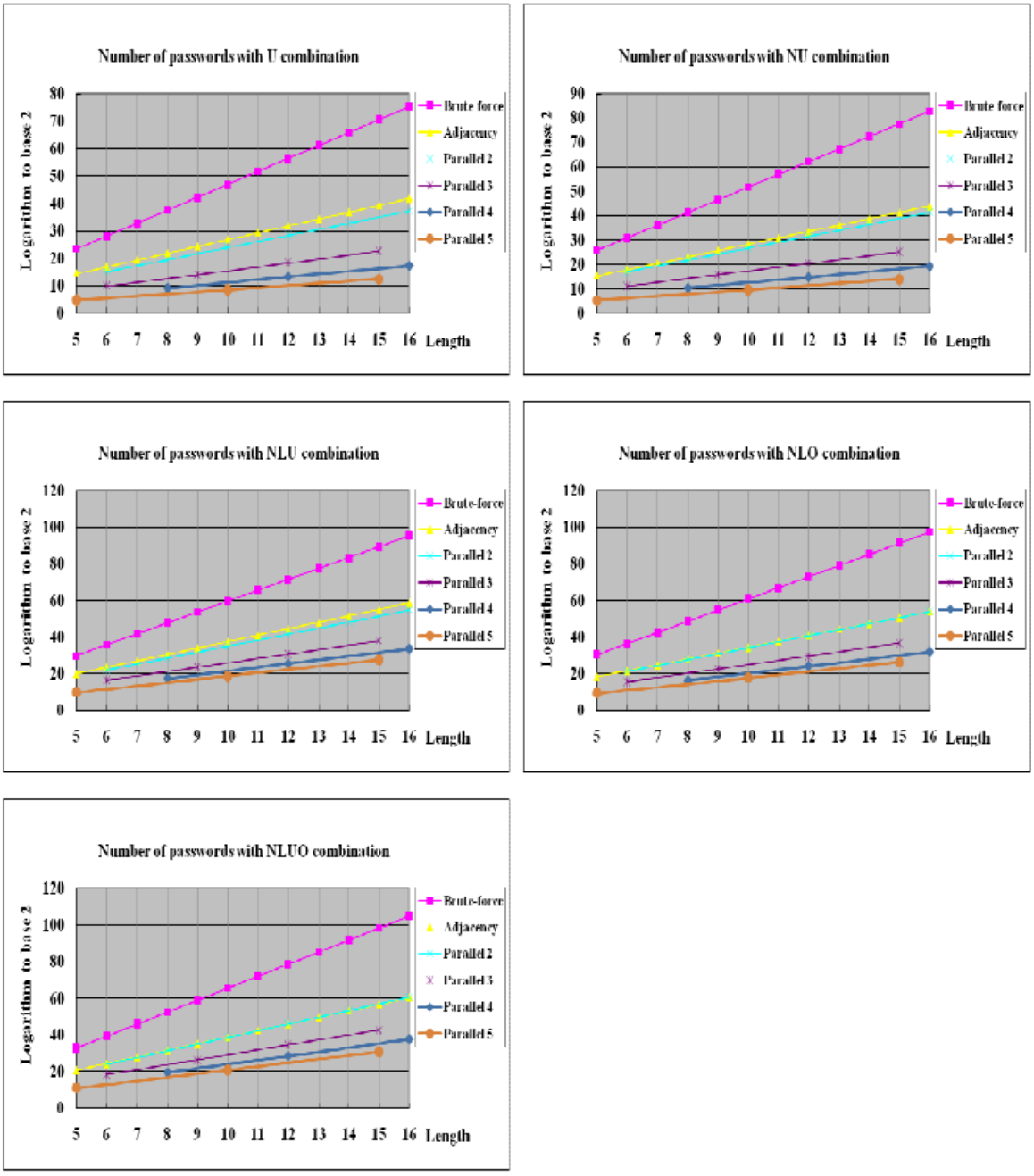
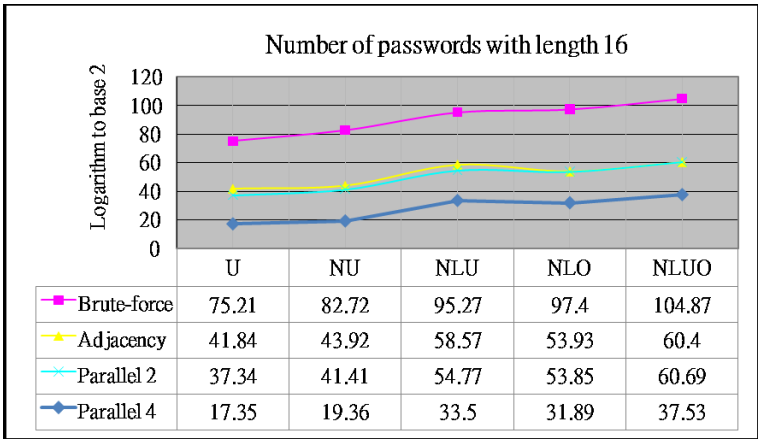


FIGURE 11. Numbers of passwords generated using AP patterns and brute-force

2. The numbers of passwords generated through adjacent patterns and parallel 2 patterns are almost the same because keys in parallel 2 patterns are also adjacent. The number of passwords generated through parallel 5 patterns is greatly reduced compared to that of parallel 3 and 4 patterns, as there is only the horizontal parallel relationship for parallel 5 patterns, while there are horizontal, negative, and positive parallel relationships for parallel 3 and 4 patterns.
3. There is a trade-off between the password length and character combinations, where they both affect password space in order to mount an attack efficiently. For example, if password length exceeds 10 and the combination is of three or more character types, the number of passwords generated exceeds 2^{40}. To avoid such a large search space, simple character combinations, such as N, U, L, O, NL and LO, with length no greater than 10 are suggested.



	U	NU	NLU	NLO	NLUO
Brute-force	75.21	82.72	95.27	97.4	104.87
Adjacency	41.84	43.92	58.57	53.93	60.4
Parallel 2	37.34	41.41	54.77	53.85	60.69
Parallel 4	17.35	19.36	33.5	31.89	37.53

FIGURE 12. Numbers of passwords generated with length 16

5. **Effectiveness and Comparison.** This section describes a hybrid password cracking system to crack access passwords collected from UNIX and PCs. Within this system, we introduce a new stage called AP-pattern attack, which uses the password databases generated by AP patterns.

5.1. **Effectiveness.** The hybrid password cracking system consists of three sequential stages of attack, i.e., Dictionary attack sequentially followed by AP-pattern attack and Brute-force attack, called the DAB password cracking system, as shown in Figure 13. The system can be in DAB mode if all stages are enabled and in DB mode if only the stage of AP-pattern attack is disabled.

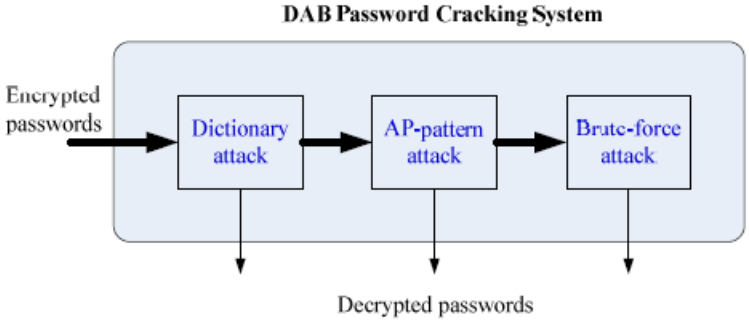


FIGURE 13. The system architecture of DAB password cracking system

UNIX Access Password Attack

We collected 382 encrypted access passwords for the UNIX system using the John Ripper cracker tool [14]. These passwords are of 13-byte values generated by the DES cryptosystem. Then, we proceeded to crack the passwords in DAB and DB modes. The results showed that, in the same 5 hours of experiments, 202 passwords and 155 passwords are cracked in DAB mode and DB mode, respectively. After the dictionary attack, in the same period of time, the number of passwords cracked with AP patterns increases 47 and up to 114% [= (51 + 37)/41]. The details are shown in Table 8, where the time taken to crack the passwords is in brackets.

Table 8. Number of UNIX passwords cracked

DAB Password Cracking System	DAB mode			DB mode	
	Dictionary attack	AP-pattern attack	Brute-force attack	Dictionary attack	Brute-force attack
Number of Passwords Cracked	114 (12min)	51 (27min)	37 (4hr 21min)	114 (12min)	41 (4hr 48min)
	202			155	

PC Access Password Attack

We also collected 196 encrypted access passwords for the PC system using the Pwdump software [15]. These passwords are of 128-bit hash values generated using the MD5 hash function. Again, we conducted the same experiments as for the UNIX passwords. The results showed that 123 passwords and 90 passwords are cracked in DAB mode and DB mode, respectively. After the dictionary attack, the number of passwords cracked with AP patterns increases 33 and up to 103% [= (36 + 29)/32]. The details are shown in Table 9, where the time taken to crack the passwords is in brackets.

Table 9. Number of PC passwords cracked

DAB Password Cracking System	DAB mode			DB mode	
	Dictionary attack	AP-pattern attack	Brute-force attack	Dictionary attack	Brute-force attack
Number of Passwords Cracked	58 (5min)	36 (13min)	29 (4hr 42min)	58 (5min)	32 (4hr 55min)
	123			90	

For both UNIX and PC systems, the number of passwords cracked with the AP-pattern attack outperforms obviously in the same period of time. The reason is that the passwords cracked fall into the password space generated for AP-pattern attack, which is far smaller than that for brute-force attack. Password cracking is thus more effective.

5.2. **Comparison.** For the keyboard-related password cracking methods, Schweitzer [8] focused primarily only on the analysis of keyword patterns, and found that the keyboard patterns of continuous 2-4 keys are the most commonly used. AP patterns formalized the commonly used adjacent and parallel keyboard patterns, and were verified effective in a password cracking system. Actually, AP patterns are a superset of the heuristic grouped patterns proposed by Schweitzer.

Moreover, Song [9] proposed a statistical analysis by timing the latencies between two keystrokes for all possible pairs of characters, and then predicted key sequences from the

inter-keystroke timings. Due to timing factors, this method is vulnerable to external influence. Passwords generated using AP patterns are as "dictionaries" developed by special keyboard patterns, and can be widely used in cracking password-based cryptosystems.

Table 10 shows a comparison of the methods of Schweitzer, Song and AP patterns. Nowadays, in order to resist various attacks, many password-based cryptosystems not only strengthen their cryptographic security, but also ask users to use "strong" passwords, at least combinations of two character types and of length 8. The strong passwords fall within the password space by AP patterns, and the cracking time is reduced.

TABLE 10. Comparisons of keyboard-related password cracking methods

Method	Schweitzer	Song	AP patterns
Feature	Classify and analyze keyboard patterns.	Predict key sequences from the inter-keystroke timings.	Focus on adjacent and parallel patterns, a superset of Schweitzer's patterns, verified in real cracking system.
Execution	Off-line.	On-line.	Off-line/on-line.
Flexibility	Standard keyboard only.	Sensitive to keyboard material.	Suitable for all well-defined input devices.
Limitation	Heuristic grouped patterns.	Statistic data required for specific keyboards.	Patterns are adjacent and parallel only.

6. **Conclusions and Future Work.** We establish a framework to formally describe the commonly used keyboard patterns of adjacent keys and parallel keys, called AP patterns, to generate password databases. The AP patterns are designed in general to be established for any well-defined keyboard-like input devices. All the printable characters on any keyboard can be classified into frequently used combinations. Based on the character combinations and AP patterns of a standard keyboard, various password databases are generated according to user preferences for password cracking, called AP-pattern attack. We also design a hybrid password cracking system consisting of dictionary attack, AP-pattern attack and brute-force attack to verify the effectiveness of AP patterns. The experimental results show that the password space is reduced drastically and cracking UNIX and PC access passwords can still be effective and outperform for up to 114% more with AP-pattern attack. Future work will focus on the formulation of more specific keyboard patterns such as triangle patterns in order to further strengthen the password databases with more flexibility and generality.

Acknowledgement. The authors gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] H. Gao, X. Liu, S. Wang, H. Liu and R. Dai, Design and analysis of a graphical password scheme, *The 4th International Conference of Innovative Computing, Information and Control*, Kaohsiung, Taiwan, pp.675-678, 2009.

[2] S. Delaune and F. Jacquemard, A theory of dictionary attacks and its complexity, *The 17th IEEE Computer Security Foundations Workshop*, 2004.

[3] J. Yan, A. Blackwell, R. Anderson and A. Grant, Password memorability and security: Empirical results, *IEEE Security & Privacy*, vol.2, pp.25-31, 2004.

[4] P. Oechslin, Making a faster cryptanalytic time-memory trade-off, *Advances in Cryptology – CRYPTO'03*, pp.617-630, 2003.

[5] V. L. L. Thing and H. M. Ying, A novel time-memory tradeoff method for password recovery, *Digital Forensics Research Conference (DFRWS)*, 2009.
[6] O. Billet and H. Gilbert, Cryptanalysis of rainbow, *Security and Cryptography for Networks*, vol.4116, pp.336-347, 2006.
[7] L. Zhuang, F. Zhou and J. D. Tygar, Keyboard acoustic emanations revisited, *ACM Transactions on Information and System Security*, vol.13, no.1, 2009.
[8] D. X. Song, D. Wagner and X. Tian, Timing analysis of keystrokes and timing attacks on SSH, *Proc. of the 10th Conference on USENIX Security Symposium*, 2001.
[9] D. Schweitzer, J. Boleng, C. Hughes and L. Murphy, Visualizing keyboard pattern passwords, *The 6th International Workshop on Visualization for Cyber Security Atlantic City*, NJ, USA, 2009.
[10] J. Noyes, The QWERTY keyboard: A review, *International Journal of Man-Machine Studies*, vol.18, pp.265-281, 1983.
[11] D. G. Alden, R. W. Daniels and A. F. Kanarick, Keyboard design and operation: A review of the major issues, *The Journal of the Human Factors Society*, vol.14, no.4, pp.275-293, 1972.
[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithm*, 2nd Edition, 2001.
[13] S. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd Edition, 2006.
[14] Openwall Project. John the Ripper Password Cracker. Retrieved on 13/3/2010.
[15] Pwdump7 by Andres Tarasco Acuna, Windows NT family, up through XP or Vista.

Appendix A. The numbers of next key lines of parallel patterns with length 2-5 established on different combinations of character types.

Length	2														
Character Combination	N	L	U	O	NL	NU	NO	LU	LO	UO	NLU	NLO	NUO	LUO	NLUO
Horizontal Parallel	9	23	23	37	32	32	68	92	66	66	101	97	97	141	172
Negative Parallel	0	16	16	16	26	26	16	64	48	48	84	58	58	112	132
Positive Parallel	0	16	16	20	25	25	20	64	48	49	82	58	58	110	128

Length	3														
Character Combination	N	L	U	O	NL	NU	NO	LU	LO	UO	NLU	NLO	NUO	LUO	NLUO
Horizontal Parallel	8	20	20	32	28	28	104	160	70	70	168	142	142	240	312
Negative Parallel	0	7	7	8	16	16	8	56	34	34	92	45	45	128	168
Positive Parallel	0	7	7	24	15	15	24	56	43	43	88	51	51	128	160

Length	4														
Character Combination	N	L	U	O	NL	NU	NO	LU	LO	UO	NLU	NLO	NUO	LUO	NLUO
Horizontal Parallel	7	17	17	15	24	24	160	272	66	66	279	211	211	415	560
Negative Parallel	0	0	0	0	7	7	0	0	15	15	56	30	30	80	160
Positive Parallel	0	0	0	16	7	7	16	0	27	27	56	34	34	88	144

Length	5														
Character Combination	N	L	U	O	NL	NU	NO	LU	LO	UO	NLU	NLO	NUO	LUO	NLUO
Horizontal Parallel	6	14	14	14	20	20	288	448	62	62	454	336	336	718	992
Negative Parallel	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Positive Parallel	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0