IET Information Security

Research Article

IET Journals
The Institution of Engineering and Technology

Robustness of text-based completely automated public turing test to tell computers and humans apart

ISSN 1751-8709
Received on 5th August 2014
Revised on 18th March 2015
Accepted on 21st May 2015
doi: 10.1049/iet-ifs.2014.0381
www.ietdl.org

Haichang Gao ✉, *Xuqin Wang, Fang Cao, Zhengya Zhang, Lei Lei, Jiao Qi, Xiyang Liu*

School of Software, Institute of Software Engineering, Xidian University, 710071 Xi'an, Shaanxi, People's Republic of China
✉ *E-mail: hchgao@xidian.edu.cn*

Abstract: Text-based completely automated public turing tests to tell computers and humans apart (CAPTCHAs) have been widely deployed across the Internet to defend against undesirable or malicious bot programmes. In this study, the authors provide a systematic analysis of text-based CAPTCHAs and innovatively improve their earlier attack on hollow CAPTCHAs to expand applicability to attack all the text CAPTCHAs. With this improved attack, they have successfully broken the CAPTCHA schemes adopted by 19 out of the top 20 web sites in Alexa including two versions of the famous ReCAPTCHA. With success rates ranging from 12 to 88.8% (note that the success rate for Yandex CAPTCHA is 0%), they demonstrate the effectiveness of their attack method. It is not only applicable to hollow CAPTCHAs, but also to non-hollow ones. As their attack casts serious doubt on the viability of current designs, they offer lessons and guidelines for designing better text-based CAPTCHAs.

1 Introduction

CAPTCHA stands for 'completely automated public turing test to tell computers and humans apart'. It is used as a defence mechanism to prevent automated registration, spam and malicious bot programmes. The process usually involves a computer asking a user to complete a simple test, designed to be difficult for a computer to solve, but easy for a human. If the correct solution is received, it can be presumed to have been entered by a human [1, 2]. The success rate for humans of a good CAPTCHA should reach to 90% or more and for computer programmes should be <1% [3].

The most common CAPTCHAs are text-based CAPTCHAs [4], which show some digits, and upper and lower case letters in an image. Compared with audio-based CAPTCHAs and image-based CAPTCHAs, text-based CAPTCHAs have the following advantages [5, 6]: being intuitive to users world-wide (users' task is simple character recognition); having few localisation issues (users world-wide recognise English letters and Arabic numerals); having good potential to provide strong security (e.g. when properly designed, the space a brute-force attack has to search can be huge).

The robustness of text-based CAPTCHAs relies on the difficulty of detecting 'where' each character is rather than 'what' it is. Computers perform better than humans in recognising individual characters, even under severe distortion [7]. Therefore text-based CAPTCHAs should be designed to be segmentation-resistant. If a scheme is vulnerable to a segmentation attack, then it is effectively broken. Many attack programmes have worked well when focused on a particular CAPTCHA scheme, such as colour filling segmentation (CFS), even cut [7] and detecting distinctive character shape patterns [3]. A clear limitation of all those methods is that they are neither generalisable nor robust in detecting slight changes in the CAPTCHA.

On the basis of the attack framework used for attacking hollow CAPTCHAs we related in our early work [8], we propose an improved attack framework that is applicable to non-hollow schemes as well. This improved attack keeps the attack essence by first, extracting character components, then trying different combinations and finally selecting the most likely partition using the graph search algorithm we designed. In the procedure of partition and recognition, the graph search algorithm proposed in [8] is improved in this paper by removing the useless nodes from the graph and, which will reduce to the calls to classifier, so as to the time complexity of our attack. The major innovation of this attack is that the method of extracting character components is applicable to most of the text-based schemes, whether the characters in CAPTCHA are hollow or solid, crowded together or isolated.

Through extensive evaluation of the CAPTCHAs adopted by the top 20 web sites in Alexa [9] in August 2013, our attack can break most schemes with a success rate of at least 12% (except the success rate of the Yandex CAPTCHA is 0%). This suggests that our attack works at a fundamental level and is especially applicable to ReCAPTCHA, the Google scheme and widely used by millions of users of Facebook, Twitter and other Internet services on a daily basis (used by over 100 000 web sites). Our attack was implemented for a ReCAPTCHA version that was active in August 2013. To our surprise, this attack, without any significant change, also works well on the latest ReCAPTCHA version. To the best of our knowledge, this is, to date, the most effective attack against these two versions of this scheme.

This improved attack provides key insights on security vulnerabilities that text CAPTCHAs should account for. It can be used to examine a CAPTCHA's robustness, as well as to guide the design of next generation CAPTCHAs. Therefore in this paper we summarise general principles for the design of robust text-based CAPTCHAs after analysing attack results and evaluating our improved attack.

The structure of this paper is as follows. In Section 2, we describe related works. In Section 3, we analyse the features of the three categories of CAPTCHAs. Section 4 provides an overview of our attack and Section 5 describes the attack in details and applies it to the CAPTCHAs of top 20 web sites. We analyse our attack results and propose some new advices for CAPTCHA design in Section 6. Section 7 provides concluding remarks.

2 Related works

There have been many methods used to attack text-based CAPTCHAs. Mori and Malik [10] used sophisticated object recognition algorithms to attack Gimpy which used clutter interference and EZ-Gimpy which used texture-background CAPTCHAs with a success rate of 33 and 92% in 2003. In 2005, the machine learning algorithms created by Chellapilla [11] in

Stanford University broke some types of CAPTCHA with a success rate ranging from 4.89 to 66.2%. Yan broke most visual schemes provided at Captchaservice.org, a publicly available web service for CATPCHA generation, with a success rate of nearly 100% by simply counting the number of pixels of each segmented character, although these schemes were all resistant to the best optical character recognition software on the market [12]. More character segmentation techniques designed to attack a number of text-based CAPTCHAs, including the earlier mechanisms deployed by Microsoft, Yahoo! and Google were developed in 2008, and these achieved a segmentation success rate of 92% against Microsoft CAPTCHA [7]. In 2011, by applying a systematic evaluation methodology to 15 current CAPTCHA schemes from popular web sites, Bursztein *et al.* [13] demonstrated that 13 of them were vulnerable to automated attacks. Recently, Xu [14] presented an attack that defeated the moving-object CAPTCHA, involving dynamic text strings called 'codewords' with a 77% success rate. In our early work, concentrating on crowding characters together (CCT), CAPTCHAs such as Google, Baidu and Yahoo! were attacked with a total of more than 30% success rate using 'divide and conquer' [15].

The early framework for breaking CAPTCHAs can be divided into three steps: pre-processing, segmentation and recognition. Within that process, the challenge must be segmented into individual characters. However, if the background clutter consists of shapes similar to letter shapes, or the characters are connected together, it becomes difficult to segment the CAPTCHA into single characters. In addition, the segmentation methods mentioned in [7, 12, 14, 15] are mostly *ad hoc* and not applicable to a wide range of text-based CAPTCHAs.

3 Text-based CAPTCHAs: classification

Excluding the hollow character CAPTCHA, text-based schemes include solid character CAPTCHA as well, and according to the positional relationship between characters, solid character CAPTCHA can be classified into two categories: character isolated CAPTCHA and CCT CAPTCHA.

Detailed information of our target schemes for the top 20 web sites is shown in three categories in Fig. 1. Some web sites have used the same CAPTCHA mechanism. For example, Google, Facebook, YouTube, LinkedIn, Twitter, Blogspot, Google India and WordPress all use ReCAPTCHA. Since ReCAPTCHA only tests the correctness of the control word when user tries to access the site, so we demonstrate the CAPTCHA image with control word only.

Given these facts, our task is to apply the attack framework used for attacking hollow CAPTCHAs to attacking the two solid category CAPTCHAs. There will be some small improvements in this attack and we will give an overview in the next section.

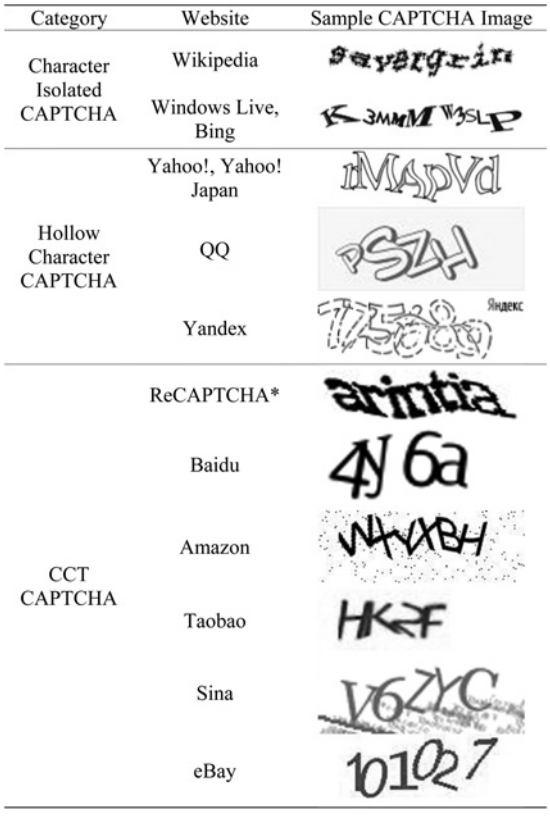

Fig. 1 *Categories of top 20 web sites*

*ReCAPTCHA was used by Google, Facebook, YouTube, LinkedIn, Twitter, Blogspot, Google India and wordpress

4 Our improved attack: an overview

The key insight behind our attack is very similar to attack for the hollow scheme, and the high-level flow of this attack is shown in Fig. 2. It includes three main sequential steps: (i) pre-processing prepares each challenge image with standard techniques; (ii) extracting character components; and (iii) partition and recognition with the convolutional neural network (CNN) engine assisted graph search.

The biggest difference between the two attacks lies in the procedure of extracting character components. For character isolated CAPTCHAs, the components can be found by CFS, similar to finding hollow characters (Note that there is no need to remove noise components.). Although for CCT CAPTCHAs, these components are extracted using stroke connection point segmentation or column segmentation, which will be described in detail in the next section.

The other difference lies in pre-processing. For hollow schemes, pre-processing may include repairing the contour line. Although for the solid schemes, the main task of pre-processing is to remove interferences.

5 Our attack: technical details

5.1 Pre-processing

As a necessary procedure in attacking CAPTCHAs, pre-processing generally involves: (i) convert the challenge image to a

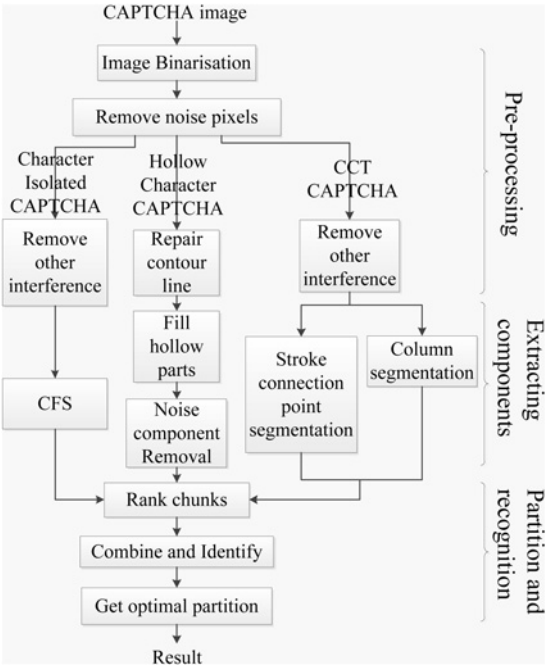

Fig. 2 *Framework of our improved attack*

IET Inf. Secur., 2016, Vol. 10, Iss. 1, pp. 45–52

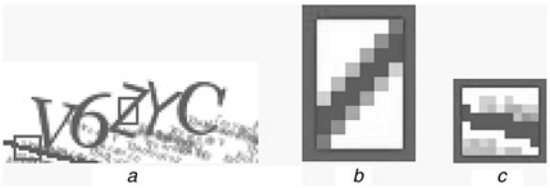

Fig. 3 *Sina CAPTCHA*

a Sina CAPTCHA
b Character part
c Noise arc part

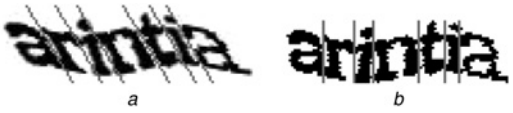

Fig. 4 *ReCAPTCHA image after pre-processing*

a Original image
b Image after rotation

black-and-white one. (ii) Remove background patterns or eliminate other additions to the image that could interfere with following work. (iii) Modify the characters to facilitate any following work, if necessary.

5.1.1 Image binarisation: This is to covert a colour or grey-scale image into black-and-white. This binarising process is done via the standard Otsu's threshold method [16] in our attack.

5.1.2 Remove interference: One form of background interference is noise pixels. They can be removed using the small pixel count.

As an alternative interference measure, noise arcs, appear in CAPTCHAs in all three categories. For hollow character CAPTCHAs, it is not necessary to remove these arcs since they do not have any effect on extraction. In the schemes adopted by the top 20 web sites, only Sina CAPTCHAs have added noise arcs to the CAPTCHA image. Analysis shows that these noise arcs are dissimilar to the characters since the characters are surrounded by light colour pixels in both up and down directions while for most of the noise arcs, there are no light colour pixels surrounding or these pixels only appear in up or down direction (Fig. 3). Since this characteristic is similar to that of early Baidu CAPTCHA [15], the noise arcs are removed using the same method.

5.1.3 Modify the characters: For the CAPTCHA whose characters are obviously rotated and the rotation angle of each character is almost the same, we rotate the CAPTCHA to make the characters upright (Fig. 4). First, set several guidelines with different slopes, ranging from the largest rotation angle to the smallest rotation angle of object scheme; the range can be derived statistically. Then use the guideline projection technique [15] to find the most likely rotation angle. Finally, rotate the image using this rotation angle.

5.2 Extracting character components

In this step, aimed at finding components in solid character CAPTCHAs, we present three different extraction methods. These three methods are applicable to different CAPTCHAs. As used in hollow schemes, CFS is used for extraction in character isolated CAPTCHAs. For CCT CAPTCHAs, stroke connection point segmentation and column segmentation are proposed to extract components.

5.2.1 CFS

CFS is used to detect the connected components [7]. For the hollow and character isolated CAPTCHAs, a simple CFS algorithm can achieve the goal of extracting the character components.

Similar to hollow character CAPTCHAs in which each filled hollow part is a component, the character isolated CAPTCHAs has each character as a component. There are no noise components to be removed in character isolated CAPTCHAs and all components will be used in the following partition and recognition steps. Fig. 5 provides an extraction sample.

5.2.2 Stroke connection point segmentation: Unlike hollow and character isolated CAPTCHAs, characters are more or less connected or overlapped in CCT CAPTCHAs as a defence measure. Stroke connection point segmentation is suitable for CAPTCHAs in which the adjacent characters are not closely overlapped and each of the connection regions between the two adjacent characters is small (Fig. 6*a*). In this CAPTCHA, the

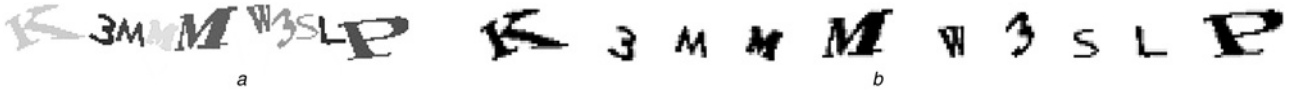

Fig. 5 *CFS in character isolated scheme (Microsoft CAPTCHA)*

a Image after CFS
b Components created by CFS (each component has been set black)

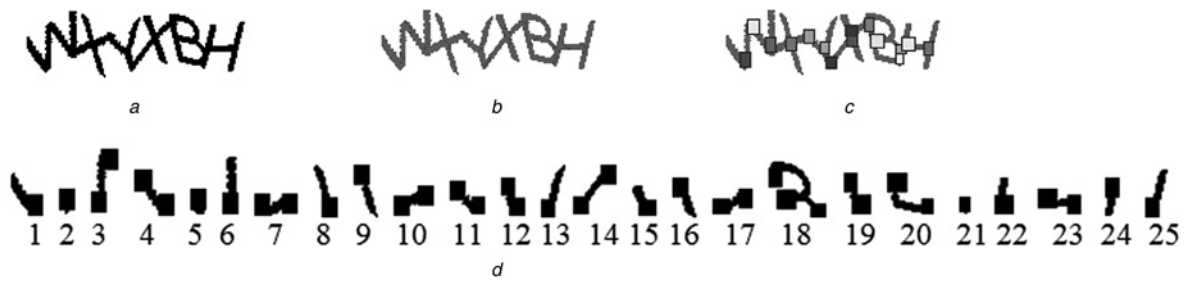

Fig. 6 *Stroke connection point segmentation in Amazon CAPTCHA*

a After pre-processing
b Sign the connection point
c Rectangle surrounding
d Components

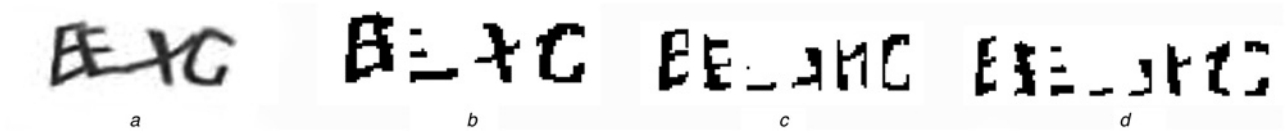

Fig. 7 *Dynamic segmentation in Taobao CAPTCHA(m = 3 and S = {4, 6, 8})*

a Original image
b Segment into four components
c Segment into six components
d Segment into eight components

connection point of strokes is similar to a junction in a maze. Considering these junctions as segmentation points can achieve the goal of extracting components from CAPTCHAs and each component belongs to a character.

The seed-filling algorithm [17] is used to find the stroke connection point. It is used to fill colour (we choose green in our attack) along the character stroke until reaching a junction at which there are two filling directions. The junction is exactly the stroke connection point and is marked with red, and then a random decision will be made about the next direction to follow. This process continues until all characters are filled (Fig. 6*b*). The final step is to draw a rectangle surrounding each marked junction (Fig. 6*c*), the rectangular region is considered to be public which will be merged into each of its adjacent components (Fig. 6*d*).

5.2.3 Column segmentation: An obvious limitation of the 'stroke connection point segmentation' method is that it is not suitable for schemes with many connection points between two adjacent characters (Fig. 7*a*). Another method, 'column segmentation', is proposed to break these orderly CAPTCHAs.

Select $m(m \geq 1)$ numbers, each no less than the CAPTCHA length, forming a set S, then for each number c in S, adopting the 'even cut' method to segment target challenges [7], that is, divide the challenge evenly into c components, each with the same width. Therefore for each challenge, there will be m different segmentations (Fig. 7). Then for each segmentation, apply the following partition and recognition algorithm (detailed in Section 5.3) to the components, figure out the final recognition result and the corresponding highest confidence sum v. Select the optimal segmentation with the biggest v from all the m segmentations, and the corresponding result is obtained as well. Specially, for the challenge with a fixed length (denoted by l) and the character width is almost the same, to reduce time consumption, it can be directly segmented into l chunks.

For the CAPTCHAs with flexible lengths, and each of the character widths is varied, to improve the success rate, one-pixel column segmentation might be a viable alternative. Each pixel column is regarded as one component. This segmentation creates a more accurate partition. The drawback of this segmentation process is the extended time required. The longer the challenge length, the more time is required.

5.3 Partition and recognition

This step is similar to that of the hollow scheme and we chose the Amazon scheme as representative to explain key techniques. Note that the techniques are generically applicable to all the schemes (except Yandex), as evidenced by our implementation and evaluation (details see attack summary in Section 5.4 and result in Section 6).

As described in [8], first, all components in an image are numbered in an incremental order from the upper left to lower right (Fig. 6*d*). Then an $n \times n$ (n is the total number of components) table is built for each image. A cell at the intersection of row i and column k (i, k), if feasible, is marked. That is, the cell will be marked if it meets the following two conditions: (i) its row index i is smaller than column index k and (ii) the width of the combination is smaller than the largest possible character width and larger than the smallest one. As depicted in Table 1, the feasible combinations are marked by '●'.

Fig. 8 gives the equivalent graph of Table 1. Our task now is to find the best segmentation and thus the most likely recognition result. This is equivalent to finding a path starting from node 1 and ending at node $n + 1$ whose length is equal to the CAPTCHA length.

Some nodes in Fig. 8 are redundant, that is, there is no feasible segmentation among all the paths passing through this node. To

Table 1 Initial $n \times n$ table for the sample

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1				●	●	●	●																			
2					●	●	●																			
3					●	●	●																			
4								●	●																	
5										●	●															
6										●	●															
7										●	●															
8											●	●														
9												●	●	●												
10												●	●	●												
11													●	●												
12															●	●	●									
13															●	●	●									
14																	●	●								
15																		●								
16																										
17																										
18																			●	●	●	●	●			
19																					●	●	●			
20																					●	●	●			
21																								●	●	●
22																										●
23																										●
24																										
25																										

IET Inf. Secur., 2016, Vol. 10, Iss. 1, pp. 45–52

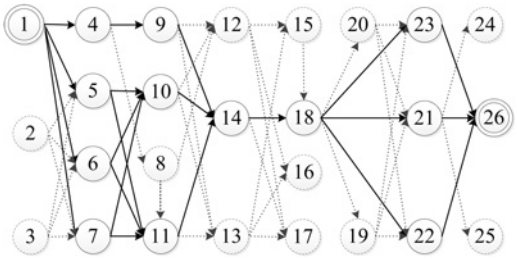

Fig. 8 *Equivalent graph of Table 1*

reduce the search space and save the time loss in classifier, the graph needs to be pruned. For each node i (excluding node 1 and node $n+1$), determine the shortest path from node 1 to node i and the shortest path from node i to node $n+1$. If the length sum of these two paths is larger than the largest possible CAPTCHA length, the node i will be removed. This algorithm repeats recursively until no further nodes are removed after a traversal.

The shortest path is computed by dynamic programming. In finding the shortest path from node 1 to i, first set that of node 1 to 0, then for each node (excluding node $n+1$), traverse all the edges whose head node is i and the length of the corresponding path is that of the tail node of this edge (the shortest path from node 1 to the tail node) plus 1. Select the smallest of all the values and this is the shortest path from node 1 to node i. In the same way, the shortest path from node i to node $n+1$ can be determined. Note that if there is no path from node 1 to node i or no path from node i to node $n+1$, we set the length of corresponding shortest path to infinity.

After this process, the redundant nodes and their connecting edges are marked by the dotted line in Fig. 8, and they are removed. Obviously, this removal process greatly facilitates our following work.

Then CNN is used to decide which character each remaining component is likely to be, and the cell (i, k) stores the neural network's recognition result and the corresponding confidence level. The final $n \times n$ table is shown in Table 2 and its equivalent graph is shown in Fig. 9. The rows and columns corresponding to the removed nodes are no longer listed in Table 2.

By the graph search algorithm we proposed in [8], all likely partitions and the corresponding sum of confidence levels are shown in Table 3. The items highlighted indicate the optimal partition that has the highest sum of confidence level and that matches a legitimate length of CAPTCHA strings. In this sample case, 'WXYXBH' is our recognition result and it is recognised correctly.

5.4 Attack summary

The partition result of object schemes are all shown in the following tables. Fig. 10 shows the output of each attack procedure in isolated character and hollow character CAPTCHAs.

In CCT CAPTCHAs, it is possible that not all the adjacent characters are crowded together, only some of the characters connected, as in the eBay and Baidu schemes. Under this condition, we first separate the text into several parts using CFS, then extract the components and apply partition and recognition, and finally find the recognition result according to the position of these separated parts in the original CPATCHA image (Figs. 11 and 12).

6 Discussion

In this section we briefly summarise our attack results for the top 20 web sites in Alexa in August 2013, to provide a comprehensive assessment of the current state of the art. We demonstrate that our attack also works for the latest ReCAPTCHA version and then provide an evaluation on our attack. Finally, some defence

Table 2 Final $n \times n$ table for the sample. The highlighted values indicate optimal partitions.

0	3	4	5	6	8	9	10	13	16	19	20	21	25
1	7/−0.33	W/0.47	W/0.54	***W/0.92***									
4					W/0.34								
5						A/-0.08	X/-0.21						
6						A/-0.05	X/-0.2						
7						A/-0.24	***X/0.43***						
9								N/-0.21					
10								A/-0.12					
11								***Y/0.68***					
14									***X/0.84***				
18										***B/0.56***	E/−0.44	B/−0.30	
21													***H/0.66***
22													H/0.43
23													H/0.32

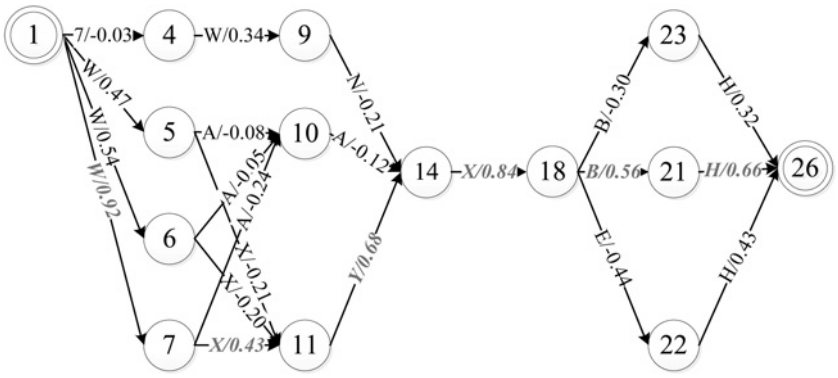

Fig. 9 *Equivalent graph of Table 2*

Table 3 Every likely result and its confidence sum for the sample. The highlighted result and value indicate the final result and value.

Result	Value	Result	Value
7WNXBH	2.16	WAAXBH	1.23
7WNXEH	0.94	WXYXBH	3.08
7WNXBH	0.96	WXYXEH	1.86
WAAXBH	2.32	WXYXBH	1.88
WAAXEH	1.1	WAAXBH	2.62
WAAXBH	1.13	WAAXEH	1.39
WXYXBH	3	WAAXBH	1.42
WXYXEH	1.78	***WXYXBH***	***4.1***
WXYXBH	1.8	WXYXEH	2.88
WAAXBH	2.43	WXYXBH	2.9
WAAXEH	1.21		

measures are suggested to design more secure text-based CAPTCHAs based on the lessons learned while doing this extensive evaluation.

6.1 Results

Table 4 shows the results of our attack on the target schemes. The results demonstrate that our attack is effective against common popular text-based CAPTCHAs.

Only Yandex CAPTCHA is resistant to our attack attempts, and we have reached some informative understanding of why we could not break it. In this CAPTCHA, the contour line is badly broken, and there are many break points marked after using Lee's algorithm. This will lead to many connection failures, filling failures and noise components removal failures.

The average speed shows our attack is efficient. Most CAPTCHAs are broken within 6 s which is close to human reaction time on CAPTCHAs. In the case of Wikipedia, Microsoft, QQ and Sina schemes, our time consumption was nearly 1 s, which is faster than human recognition. The average speed for ReCAPTCHA is 118.44 s, because the one-pixel column segmentation produces more than 100 components so that there are thousands of combined components needing to be classified. Compared with

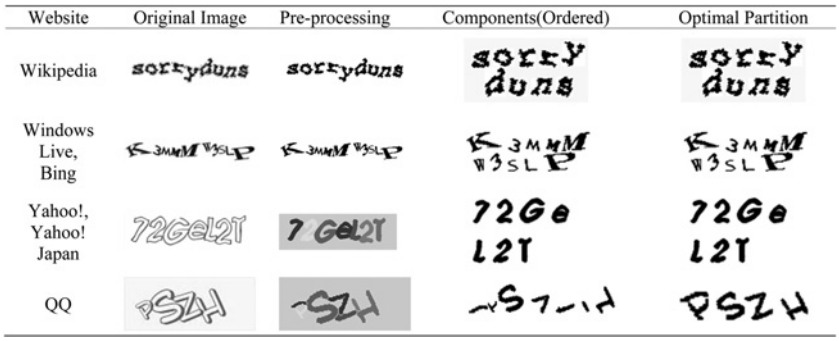

Fig. 10 *Result of CAPTCHAs using CFS*

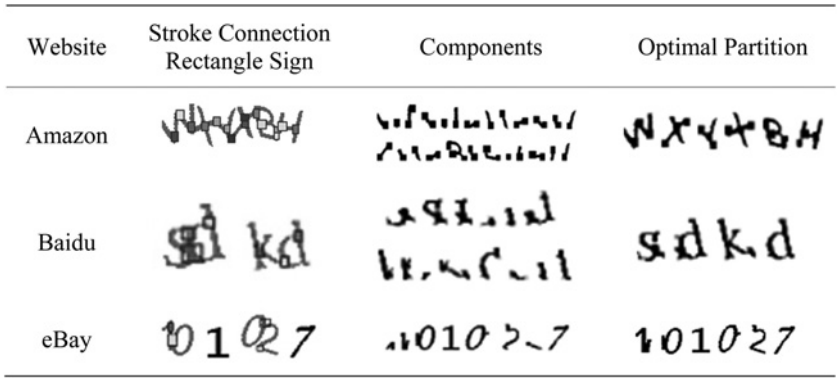

Fig. 11 *Result of CAPTCHAs using connection point segmentation*

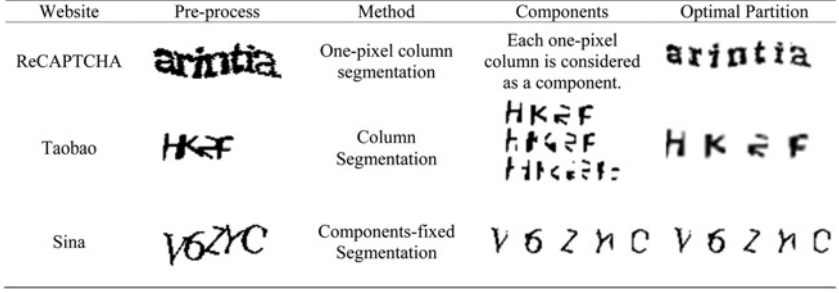

Fig. 12 *Result of CAPTCHAs using pixel column segmentation*

Table 4 Results of our attack

Category	Web site	Success rate	Average speed, s	Extraction method
character isolated CAPTCHA	Wikipedia	70.4%	1.11	CFS
	Windows Live, Bing	88.8%	0.96	CFS
hollow character CAPTCHA	Yahoo!, Yahoo! Japan	56%	3.82	CFS
	QQ	93%	1.08	CFS
	Yandex	0%	–	CFS
CCT CAPTCHA	ReCAPTCHA	12%	118.44	column segmentation
	Baidu	62%	2.6	stroke connection point segmentation
	Amazon	19.5%	4.51	stroke connection point segmentation
	Taobao	42.8%	5.38	column segmentation
	Sina	20%	0.93	column segmentation
	eBay	49.8%	3.87	stroke connection point segmentation

other CAPTCHAs which are usually <30 components after extraction, ReCAPTCHA consumes more time in our attack.

6.2 Attack applicability to latest ReCAPTCHAs

In early October 2013, we noted that ReCAPTCHA rolled out new protection for a series of numbers to increase their robustness to attacks. We downloaded 500 random samples from the Internet and tested our attack. The results are amazing: our programme achieved a success rate of 23%; that is, 115 out of 500 samples were completely recognised. On average, it took 4.76 s for our computer to break a challenge.

6.3 Comparisons with related works

Decaptcha has also broken the Wikipedia, Baidu and eBay using the traditional attack which contains consecutive steps of segmentation and recognition, with success rates of 25, 5 and 43%, respectively [13]. Our attack results are 70.4, 62 and 49.8%, respectively, demonstrating that our attack is more effective. Furthermore, we successfully broke ReCAPTCHA while they do not. Moreover Decaptcha cannot break hollow CAPTCHAs, either, but ours can. Besides, their work is more like a toolbox combining a variety of algorithms.

In [18], Bursztein's team uses a brute-force similar approach with five steps included to break many popular CAPTCHAs. The main idea is to use a machine learning algorithm to score all possible ways to segment a CAPTCHA and decide which combination is the most likely to be the correct one. In their second procedure, an exponential number of segments or cuts are produced. From the comparisons in Table 5, we can see that both our method and [18] can break all the illustrated schemes. However, our method performs better than [18] in most of the schemes, and our method is much simpler and can deal with hard schemes as well.

For Claudia's work, their attack approach aims at breaking ReCAPTCHA only. Therefore they do lots of *ad hoc* methods to segment the CAPTCHA and use morphological features to tell different characters. Their attack results are indeed better than us to break ReCAPTCHA, but ours aim at a generic method, ReCAPTCHA is just one kind.

From the compare with prior art Decaptcha [13], Bursztein [18] and Claudia's [19] in this section, it is clear that generality and simplicity are the main advantages of our method. Moreover, our method performs better than previous works in most of the schemes.

Table 5 Attack results (success rate and time) for different methods

	Decaptcha [13]	Bursztein [18]	Claudia's [19]	Our method
ReCaptcha	0%/–	19.22%/4.59 s	40.4%/2 s	12%/118.44 s
Wiki	25%/–	28.29%/–	–	70.4%/1.11 s
eBay	43%/–	47.92%/2.31 s	–	49.8%/3.87 s
Baidu	5%/–	33.42%/3.94 s	–	62%/2.6 s

6.4 Evaluation

6.4.1 Evaluation on the attack-resistance features: Table 6 summarises the features of each target scheme. As these schemes represent different designs, each with distinctive features, our attack is of some generic value.

Referring to the results shown in Table 4 and the features shown in Table 6, we will discuss which features significantly contribute to a CAPTCHA's security and which do not.

String length: As described in [8], first, the more characters used in a CAPTCHA image, the more components remain after pre-processing, and the larger the solution space will be, which will decrease an attack's success and speed. Second, the more characters used, the harder for brute-force guessing. Besides, it is good to use a varied length, which does not give away any useful information to aid attackers. Attackers have to try multiple possible lengths, which increase the search space for our graph algorithm, and could decrease its success and speed.

Varied fonts: Compared with others, this feature contributes little to security. As long as the varied fonts do not affect the components to be extracted and our CNN classifier is strong enough to classify characters with different fonts, this feature seems weak for attackers.

Alphabet size: The larger the alphabet size, the larger a candidate set faces in our CNN classifier, and the more likely it is to give inaccurate results. For example, the alphabet size of ReCAPTCHA is 42, which is relatively large, and our attack success rate is only 12%. However, what calls for special attention is that if confusing characters are included in the alphabet, it will have a negative impact on CAPTCHA's usability at the same time. For Taobao CAPTCHA, the usage of '0' and 'O' reduces its usability greatly.

Width difference of thinnest and fattest characters: The larger the gap is, the larger a solution set our graph search algorithm is required to go through. This might significantly slow down our attack.

Noise arcs: With arcs cutting across characters, more components will be extracted. As noted in Section 5.1, for character isolated and CCT CAPTCHAs, it is necessary to remove these arcs. This considerately increases the difficulty level of designing and implementing an effective attack and may lead to a lower success rate. Take Sina (with noise arcs) and Taobao (with no noise arcs) schemes as example, the attack success rate of Taobao scheme is 22.8% higher than that of Sina.

6.4.2 Evaluation on our general method: An important procedure of our attack is extracting character components. The three extraction methods we proposed are effective in most CAPTCHAs. CFS is a fast and efficient method to extract components from character isolated CAPTCHAs and hollow character CAPTCHAs. Stroke connection point segmentation is appropriate for the CAPTCHAs where the connection pixel points between two adjacent characters are few. Column segmentation has a good performance on the CAPTCHAs in which there are many connections between two adjacent characters (such as Taobao and ReCAPTCHA).

The components are combined according to their location, recognised by CNN and the optimal partition is obtained by graph search. If the number of components is large, the time complexity of algorithm will be high, and the success rate may be reduced.

6.5 Defence

Clearly, CAPTCHA security comes from having a sound and coherent design at the core design, anti-recognition and anti-segmentation levels. Considering these two perspectives, we

Table 6 Main attack-resistance features of the top 20 web sites

Category	Web site	CAPTCHA length	Varied fonts	Noise arcs	Alphabet size	Width difference of fattest and thinnest character (pixels)
character isolated	Wikipedia	varied (8–10)	no	no	26	14
CAPTCHA	Windows Live, Bing	varied (8–10)	yes	no	30	41
hollow character	Yahoo!, Yahoo!	varied (6–8)	yes	no	28	55
CAPTCHA	Japan					
	QQ	fixed (4)	no	no	25	19
	Yandex	fixed (6)	yes	yes	10	23
CCT CAPTCHA	ReCAPTCHA	varied (6–8)	yes	no	42	53
	Baidu	fixed (4)	no	no	23	25
	Amazon	fixed (4)	yes	no	47	32
	Taobao	fixed (4)	yes	yes	37	30
	Sina	fixed (6)	no	no	10	28
	eBay	fixed (6)	no	no	30	29

derived the following core set of design principles for CAPTCHA designers to follow to create schemes resilient to state-of-the-art attackers to mitigate our attack:

- Increase the length of each CAPTCHA string.
- Increase the variation in character widths.
- *Increase the alphabet size:* This is a simple but effective solution, and with little negative impact on usability (if confusing characters are excluded from the alphabet).
- *Expiry:* Set a relatively short expiry time for each image, with a rapid refreshment rate. The result shows our attack on some CAPTCHAs may consume a long time, such as ReCAPTCHA. We advise setting an expiry refresh mechanism so that it is not long enough for an automated attack, but humans can still react in this period of time.
- *Background confusion:* There are three main ways to achieve this: using a complex image background, having a background that has similar colours to the text and adding noise to the CAPTCHA. Clearly, these techniques can resist extraction, but may reduce the recognition rate of humans at the same time.

We note that careful studies are needed to establish how well these measures will work, and more importantly, whether these measures may decrease recognition success for humans. It is important to maintain balance between security and usability. It remains an open problem when a design will be both secure and usable, and whether this design is mission impossible. Nonetheless, our discussion offer practical suggestions for improving the state of the art of text-based CAPTCHA designs.

7 Conclusions

In this paper, we proposed an improved attack frame that is applicable for most text-based CAPTCHAs. The CAPTCHAs adopted by the top 20 web sites in Alexa were selected as the representatives for attack. As used in the hollow scheme, our attack methods included pre-processing, extracting character components, partition and recognition. CFS, stroke connection point segmentation and column segmentation methods are proposed to extract components from CAPTCHA schemes of different styles in CCT CAPTCHAs. Partition and recognition were adopted to combine components into original characters and gain the result. Apart the Yandex CAPTCHA, we achieved a success rate of at least 12% in these schemes. Overall, our attack contributed to furthering current understanding of the design of better CAPTCHAs, in particular the design and implementation of segmentation and recognition resistance mechanisms.

By comparing representative designs of popular text-based CAPTCHAs, we have identified good design features for better security. We have also discussed how to create the next generation of better designs. However, it remains an open problem how to design CAPTCHAs that are both secure and usable, and this is our ongoing work. Overall, our work contributes to advancing the current collective understanding of CAPTCHA design.

8 Acknowledgments

The authors thank the reviewers for their careful reading of this paper and for their helpful and constructive comments. This project is supported by the National Natural Science Foundation of China (61472311) and the Fundamental Research Funds for the Central Universities.

9 References

1 von Ahn, L., Blum, M., Langford, J.: 'Telling humans and computer apart automatically', *Commun. ACM, Aug.*, 2003, **46**, pp. 57–60
2 von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: 'CAPTCHA: using hard AI problems for security' (Eurocrypt, 2003)
3 El Ahmad, A.S., Yan, J., Tayara, M.: 'The robustness of Google CAPTCHAs' (Newcastle University, Bericht, 2011)
4 Yan, J., El Ahmad, A.S.: 'Usability of CAPTCHAs or usability issues in CAPTCHA design'. Proc. of the Fourth Symp. on Usable Privacy and Security, 2008, pp. 44–52
5 Chellapilla, K., Larson, K., Simard, P.Y., *et al.*: 'Building segmentation based human-friendly human interaction proofs (HIPs)', in Baird, H.S., Lopresti, D.P., (Eds.): 'Human interactive proofs' (Springer, Berlin, Heidelberg, 2005), pp. 1–26
6 Chellapilla, K., Larson, K., Simard, P.Y., *et al.*: 'Computers beat humans at single character recognition in reading based human interaction proofs (HIPs)'. Second Conf. on Email and Anti-Spam (CEAS) 2005, Stanford University, USA, July 2005
7 Yan, J., El Ahmad, A.S.: 'A low-cost attack on a Microsoft CAPTCHA'. Proc. of the 15th ACM Conf. on Computer and Communications Security, 2008, pp. 543–554
8 Gao, H., Wang, W., Qi, J., *et al.*: 'The robustness of hollow CAPTCHAs'. Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security, 2013, pp. 1075–1086
9 Chellapilla, K., Larson, K., Simard, P., *et al.*: 'Designing human friendly human interaction proofs (HIPs)'. Proc. SIGCHI Conf. on Human Factors in Computing Systems, 2005, pp. 711–720
10 Mori, G., Malik, J.: 'Recognizing objects in adversarial clutter: 'breaking a visual CAPTCHA'. Proc. 2003 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, 2003', 2003, vol. 1, pp. 1: I-134–I-141
11 Simard, P.Y.: 'Using machine learning to break visual human interaction proofs', *Adv. Neural Inf. Process. Syst.*, 2005, **17**, pp. 265–272
12 Yan, J., El Ahmad, A.S.: 'Breaking visual CAPTCHAs with naive pattern recognition algorithms'. 23rd Annual Computer Security Applications Conf., 2007. ACSAC 2007, 2007, pp. 279–291
13 Bursztein, E., Martin, M., Mitchell, J.: 'Text-based CAPTCHA strengths and weaknesses'. Proc. 18th ACM Conf. on Computer and Communications Security, 2011, pp. 125–138
14 Xu, Y., Reynaga, G., Chiasson, S., *et al.*: 'Security and usability challenges of moving-object CAPTCHAs'. Decoding Codewords in Motion[C]//USENIX Security Symp., 2012, pp. 49–64
15 Gao, H., Wang, W., Fan, Y., *et al.*: 'The robustness of 'connecting characters together CAPTCHAs', *J. Inf. Sci. Eng.*, 2014, **30**, (2), pp. 347–369
16 Hoel, J.H.: 'Some variations of Lee's algorithm', *IEEE Trans. Comput.*, 1976, **100**, (1), pp. 19–24
17 Bruce, J., Balch, T., Veloso, M.: 'Fast and inexpensive color image segmentation for interactive robots'. Proc. 2000 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2000. (IROS 2000), 2000, vol. 3, pp. 2061–2066
18 Bursztein, E., Aigrain, J., Moscicki, A., *et al.*: 'The end is nigh: generic solving of text-based CAPTCHAs'. Proc. Eighth USENIX Conf. on Offensive Technologies. USENIX Association, 2014, pp. 3–3
19 Cruz-Perez, C., *et al.*: 'Breaking ReCAPTCHAs with unpredictable collapse: heuristic character segmentation and recognition', in Carrasco-Ochoa, J.A., (Eds.): 'Pattern recognition' (Springer, Berlin, Heidelberg, 2012), pp. 155–165