Breaking an Animated CAPTCHA Scheme

Vu Duc Nguyen[1], Yang-Wai Chow[2], and Willy Susilo[1,⋆]

[1] Centre for Computer and Information Security Research
{dvn108,wsusilo,caseyc}@uow.edu.au
[2] Advanced Multimedia Research Laboratory
School of Computer Science and Software Engineering,
University of Wollongong, Australia

Abstract. CAPTCHAs have become a ubiquitous security countermeasure to protect online web-services against automated attacks. However, attackers have managed to successfully break many existing CAPTCHA schemes. Animated CAPTCHA schemes have been proposed as a method of producing CAPTCHAs that are more human usable and more secure. The addition of the time dimension is supposed to increase the robustness of animated CAPTCHAs. This paper investigates the robustness of HelloCaptcha, an animated text-based CAPTCHA scheme with a total of 84 different variations. In this paper, we show that simple techniques can be used to extract important information from the animation frames of an animated CAPTCHA. Our approach essentially reduces the animated CAPTCHA into a traditional single image CAPTCHA challenge. Furthermore, the methods presented in this paper can be generalized to break other animated CAPTCHAs.

Keywords: Animated CAPTCHA, character extraction, segmentation, optical character recognition.

1 Introduction

A CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is an automated challenge and response test to verify whether or not an online transaction is being carried out by a human [13]. At present, CAPTCHAs are used on many web-based services as a standard security mechanism for deterring automated attacks by bots or other malicious programs. Unfortunately, a large number of CAPTCHA schemes to date have been successfully broken. Many researchers and practitioners alike have shown that various design flaws can be exploited by automated programs to break these CAPTCHAs.

To increase the security strength and to confuse Optical Character Recognition (OCR) programs, traditional text-based CAPTCHAs rely on techniques like distorting the text and/or the overlaying of visual noise. However, this often makes the resulting CAPTCHA difficult for humans to use. A good CAPTCHA

⋆ This work is supported by ARC Future Fellowship FT0991397.

scheme must be both secure and human usable. It is for this reason that CAPTCHA developers have explored alternate paradigms in CAPTCHA design in an attempt to designing more secure and usable CAPTCHAs.

Animated CAPTCHAs have been proposed as a means of overcoming the limitations of traditional single image CAPTCHAs. One of the key principles behind the design of animated CAPTCHA schemes is that the information required to solve the CAPTCHA is not contained within a single image. As such, a human has to observe the animated CAPTCHA over its animation cycle in order to gather appropriate information to correctly solve the CAPTCHA challenge. This is assumed to be difficult for a computer because information is spread over multiple images. In addition, noise and other impediments can be added to the CAPTCHA challenge to deter automated attacks.

This paper addresses the question of whether animated CAPTCHAs really provide more security. In particular, this paper describes methods of breaking a representative animated text-based CAPTCHA scheme called HelloCaptcha [12], which at the time of writing has a total of 84 different variations. The results of this research show that we can successfully break all the variations of this animated CAPTCHA scheme with a high success rate using simple methods to extract information from the animation frames.

Our Contribution. In this paper we study the robustness of animated CAPTCHAs by breaking a representative animated CAPTCHA scheme named HelloCaptcha [12]. Our research shows that even though animated CAPTCHAs spread important information over multiple animation frames, it is possible to collect the relevant information from these frames in order to break the CAPTCHA. We present our approach of extracting key information from the animation frames and effectively reducing the animated CAPTCHA challenge into a single image. It is likely that the methods presented in this paper can be extended to break other animated CAPTCHA schemes. Our goal is to identify flaws in the design of animated CAPTCHAs that make them easy to break, so that future CAPTCHA schemes can be designed to avoid such pitfalls.

2 Related Work

2.1 Breaking CAPTCHAs

Over the year, many techniques have been proposed for breaking CAPTCHAs. These automated CAPTCHA solving methods are often based on pattern recognition, image processing, machine learning algorithms and so on. For example, Mori and Malik [9] developed an approach to break the Gimpy and EZ-Gimpy CAPTCHAs using object recognition techniques to identify words amidst background clutter. In their work, they presented a holistic approach of recognizing entire words at once, rather than attempting to identify individual characters in severe clutter. Li et al. [8] have also shown that image processing and pattern recognition algorithms, such as k-means clustering, digital image in-painting, character recognition based on cross-correlation, etc. have been successful in breaking a variety of e-Banking CAPTCHAs.

In a systematic study regarding the strengths and weaknesses of text-based CAPTCHAs, Bursztein et al. [3] observed that typical automated CAPTCHA solving processes can be divided into five generic steps, namely, pre-processing, segmentation, post-segmentation, recognition, and post-processing. While segmentation, the separation of a sequence of characters into individual characters, and recognition, the identification of those characters, are intuitive and generally understood, the additional pre-processing and post-processing steps are also included as part of a standard process. For example, pre-processing a CAPTCHA image can remove background patterns or eliminate other impediments that could interfere with segmentation, while a post-segmentation step can 'clean up' the segmentation output. After recognition, a post-processing step can used to improve accuracy by, for example, applying spell checking to any CAPTCHA that is based on actual dictionary words.

2.2 Segmentation Resistant

It is widely accepted that state of the art in text-based CAPTCHA design requires that a robust CAPTCHA be segmentation resistant. This segmentation resistant principle is based on the work by Chellapilla et al. [4] who have shown that automated programs can recognize single characters even better than humans. As such, if a text-based CAPTCHA can be segmented into its constituting characters, it is essentially broken.

This segmentation resistant principle required for robust CAPTCHA design, led a research team at Microsoft to develop a CAPTCHA scheme that was meant to be segmentation resistant. Unfortunately, it was shown that the Microsoft CAPTCHA could in fact be segmented, and thus broken, by a low-cost attack [15]. In addition, researchers have demonstrated the use of novel segmentation techniques to break various other CAPTCHAs [14]. Nevertheless, the success of these attacks does not negate the segmentation resistant principle in the design of robust CAPTCHAs. However, relying on segmentation alone does not provide reliable defense against automated attacks [3].

2.3 Animated CAPTCHAs

Animated CAPTCHAs have been proposed to overcome the limitations of traditional single image CAPTCHAs. Animated CAPTCHAs can be presented on webpages by means of three main formats; as an animated GIF (Graphics Interchange Format) image, a flash video or via video streaming. To date, animated CAPTCHA are not commonly used on websites yet. However, a number of animated CAPTCHA schemes have been proposed by the web community as well as the research community.

Athanasopoulos and Antonatos [2] contend that their proposed CAPTCHA, enhanced with animation technology, can resist sophisticated attacks better than standard CAPTCHAs based on static images with distorted text. Cui et al. [6] presented a sketch of an animated CAPTCHA approach based on moving letters amid a noisy background. An animated CAPTCHA method based on the idea of presenting distorted text on the face of a deforming surface was proposed by Fischer and Herfet [7]. Naumann et al. [10] suggest another animated CAPTCHA technique based on visual phenomena. The idea behind their approach is that by grouping different entities that move together, sketches or letters that are superimposed over a noisy background of the same color become visible once they are moving.

Chow and Susilo [5] devised an animated 3D CAPTCHA named AniCAP, that was designed with the segmentation resistant principle in mind. Their approach is based on motion parallax, the perception of depth through motion, where humans are supposed to distinguish the main characters located in the foreground from the background characters. NuCaptcha is another animated CAPTCHA designed to be segmentation resistant. The idea behind this CAPTCHA is that characters are joined together, but when seen to be moving, the user's mind sees the different parts and fills in the blanks; the parts that are moving together are grouped together, and user can clearly differentiate the letters. On the other hand, computers do not have this advantage and see a smear of pixels [11].

3 The Targeted Animated CAPTCHA Scheme

To investigate the robustness of animated text-based CAPTCHA schemes, the HelloCaptcha [12] was selected as a representative scheme. HelloCaptcha is an animated CAPTCHA scheme that has been made freely available via a web service with an easy to use embedding code. The developers of HelloCaptcha state that their CAPTCHA schemes are more readable, more secure and nice. The animation approach is intended to provide an additional time dimension in order to increase its difficulty against automated attacks [12]. However, whilst being usable and easy on the eyes, it is obvious that HelloCaptcha was not designed with the segmentation resistant principle in mind. Our purpose is to investigate whether this implementation of the time dimension alone really provides additional security to animated CAPTCHAs.

One of the key reasons why HelloCaptcha was selected as a representative animated CAPTCHA scheme for this study is because the developers have provided a variety of different variations to their scheme. At the time of writing, the developers of HelloCaptcha have provided 12 broad categories of different animated CAPTCHA schemes. Furthermore, each category has a number of different variations. In total, there are 84 different types, which have been grouped into the 12 broad categories. We have not found any other developer whom have provided a greater number of animated CAPTCHA types.

Please note that there too many HelloCaptcha variations to adequately describe in this paper. Thus, we encourage interested readers to visit the HelloCaptcha website at `http://hellocaptcha.com/` for further details and to view the different animated CAPTCHAs for themselves. In general, each animated CAPTCHA challenge provided by HelloCaptcha consists of a sequence of six letters and/or digits presented in an animated GIF image with the dimensions of 180×60 pixels. This paper will use the notation '*Category/Type*' to denote a specific HelloCaptcha ***type*** of a particular ***category***.

4 Breaking HelloCaptcha

In general, our attack can be divided into 4 stages. An overview of these stages is depicted in Fig. 1. At any time, only 1 of these 84 different types is randomly selected and presented to the user. Therefore, the first stage to breaking the CAPTCHA was to automate the process of identifying which of the 84 types is the current type being presented. This phase was called the type distinction stage. Upon successfully distinguishing the CAPTCHA type, the appropriate technique was then selected to extract a single image containing all the separate characters in the CAPTCHA challenge. The resulting image then underwent a pre-processing stage to remove noise and to increase the legibility of the characters. Subsequently, the final character recognition stage involves the image being passed through an OCR program to solve the CAPTCHA by recognizing the individual characters contained in the challenge.

4.1 Type Distinction

In order to automatically distinguish which of the 84 different types was currently being presented, a number of factors were considered. These factors are described as follows.

- **Number of frames.** Each HelloCaptcha challenge is displayed as an animated GIF that is constructed from a sequence of image frames. The total

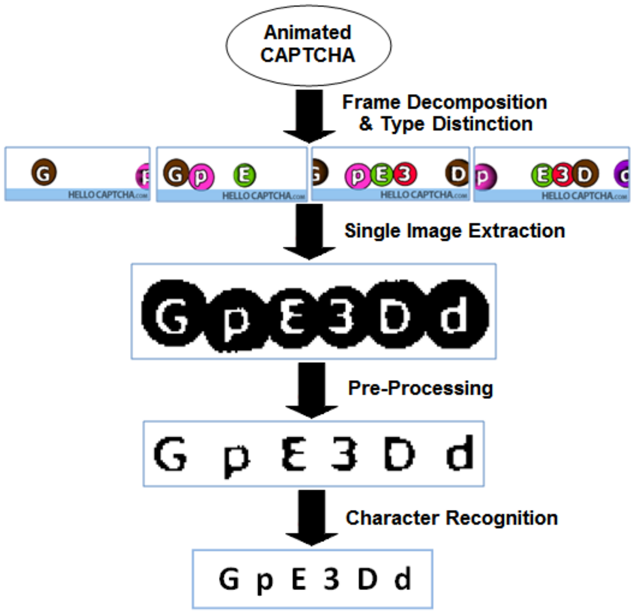


Fig. 1. Overview of the stages used to break HelloCaptcha

number of frames contained in the GIF file can be used to distinguish between some of the different types. For example, some HelloCaptcha types always contain a fixed number of frames (e.g. the types named *Noir Nois*, *Autumn Ant* and *Col Noi* of the *Noisy Mosaic* category are always fixed at 25 frames), while the total number of frames for other types may vary within a fixed range of frames (e.g. the number of frames in the *Contour* and *Redblue* types of the *TextFlood* category vary between 65 to 76 frames).
- **Maximum frame delay.** Animated GIFs display successive frames after a certain time duration, or frame delay time, which controls the speed of the animation. In other words, the frame delay informs the playback application how long to wait after displaying the current frame before it is to display the next frame in the animation. This information can be obtained from the GIF file and the maximum frame delay time can be used as a distinguishing factor between the different HelloCaptcha types. For instance, all types in the *Swapper* category consistently use a 100 ms frame delay whereas all type in the *Spring* category consistently use a value of 4 ms.
- **Number of blank frames.** We define blank frames as frames that contain only the background color. Some HelloCaptcha types always have a fixed number of blank frames over the entire animation. As such, this value can be used for type distinction. For example, there are always 17 blank frames in the *Smarties/Follower* type whereas the *Smarties/Negative* type always contains 7 blank frames.
- **Background color.** This is another value that is also very useful for type distinction. Many types are displayed using specific background colors. As an example, the background color used in the *Search Light/Stille Natch* type is made up of the following RGB color values Red=6, Green=38, Blue=66. The combination of these color values is unique in all the 84 types and it helps to easily distinguish the *Search Light/Stille Natch* type from other types. In practice, the background color can be ascertained by determining the most used pixel color in the first animation frame.

These simple pieces of information can easily be obtained from an animated GIF file. By using the different values of the number of frames, maximum frame delay, number of blank frames and background color, all the different HelloCaptcha types can be distinguished with a high degree of accuracy. Our experimental results show that we can accurately distinguish between the different types between 80%−100% of the time.

4.2 Single Image Extraction

One of the main security mechanisms provided by animated CAPTCHAs is that the information required to solve the CAPTCHA challenge is spread over multiple animation frames, unlike traditional CAPTCHAs where all information has to be presented in a single image. The same applies to the HelloCaptcha scheme, where the addition of this time dimension is intended to increase the difficulty of automated attacks [12].

In most animated CAPTCHAs, the challenge is obscured in individual frames where text characters may only be partially visible, joined together, overlapped with extra characters at random, and so on. While it is indeed impossible to solve most animated CAPTCHAs using individual animation frames, we show that it is possible to solve the CAPTCHA challenge by collecting information from multiple animation frames. Our strategy is to extract relevant information from the animation frames into a single image by using a set of simple techniques. We demonstrate that once this information is extracted, the animated CAPTCHA is in effect converted into a single image and the standard techniques used to break traditional CAPTCHAs can easily be applied to this resulting image.

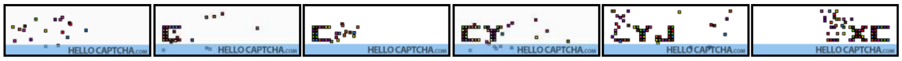
Extraction by Pixel Delay Map (PDM). This is the main method used to extract relevant information to break HelloCaptcha. A total of 61 of the 84 HelloCaptcha types can be broken using this approach. For the purpose of CAPTCHA usability, text characters required to solve the challenge in a text-based CAPTCHA often needs to be emphasized somehow. This is because the human brain needs to be able to distinguish and recognize the characters required to solve the challenge.

In many HelloCaptcha types (e.g. *Flitter/Colorful*, *H-Mover/Street*, *Search Light/Cyber*, etc.), in order to get the human user's attention, the text characters required to solve the challenge are displayed at certain fixed locations for longer periods of time compared to the noise elements and other peripheral components. This allows us to build what we call a Pixel Delay Map (PDM) from the animation frames. The PDM is an image resulting from the accumulation of the total amount of time that a pixel gets displayed in a color that is different from the background color. Given an animated CAPTCHA, *AC*, the following is a depiction of the basic algorithm used to construct a PDM.

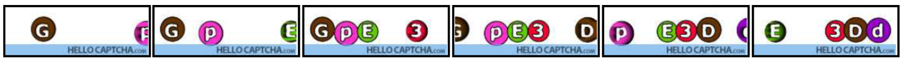
CalculatePDM(AC)

```
Initialize PDM(x,y) ← 0
begin
    for  i ← 1 to TotalFrames
       for  x ← 1 to TotalRows
          for  y ← 1 to TotalColumns
            if  Color(x,y) ≠ BackgroundColor  then
                PDM(x,y) ← PDM(x,y) + FrameDelay(i)
end
```

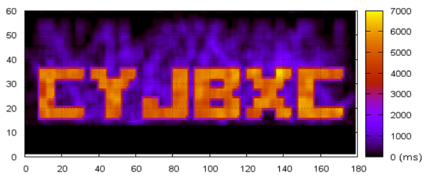
Fig. 2 shows examples of PDMs constructed from two different HelloCaptcha types, namely the *Flitter/Colorful* and the *H-Mover/Baller* types respectively. Fig. 2(a) and Fig. 2(b) depict six individual frames obtained at different times during the animation, with time increasing from left to right. In the *Flitter/Colorful* challenge that is shown in Fig. 2(a), small random colored squares gradually assemble to form the characters for a period of time, then disassemble



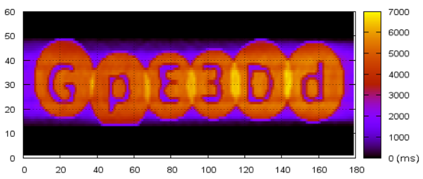
(a) Example frames from a *Flitter/Colorful* CAPTCHA challenge.



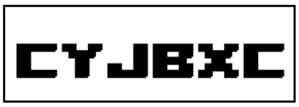
(b) Example frames from a *H-Mover/Baller* CAPTCHA challenge.



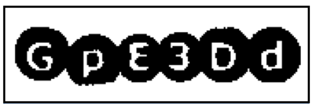
(c) PDM for challenge in Fig. 2(a).



(d) PDM for challenge in Fig. 2(b).



(e) Extracted from PDM in Fig. 2(c).



(f) Extracted from PDM in Fig. 2(d).

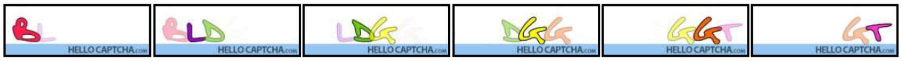
Fig. 2. Examples of character extraction using the Pixel Delay Map (PDM) approach

as if blown apart. For the *H-Mover/Baller* challenge in Fig. 2(a), the characters gradually slide in from the right at random speeds, pause for a short while, before sliding out from the left.

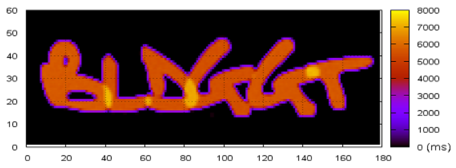
The resulting PDM images are shown in Fig. 2(c) and Fig. 2(d). The x-axis and y-axis denote the pixel positions, whereas the different colors in the PDM image illustrate the length of time in which the animated CAPTCHA's pixels were displayed in a color that was distinct from the background color. In particular, a PDM pixel with a higher value means that the color at that pixel position was different from the background color for a longer period of time compared to PDM pixels with lower values.

The extracted binarized images that are shown in Fig. 2(e) and Fig. 2(f) were obtained by simply thresholding the PDM image by $\frac{1}{2}$ the highest PDM value (i.e. 3500 ms). In other words, pixels in the PDM that had a value lower than this threshold were set to white whereas pixels with a value higher than the threshold were set to black. One can see that the CAPTCHA challenges in these images can easily be solved.

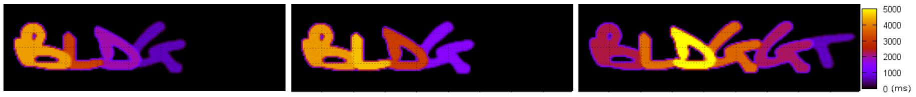
In practice, the PDM technique is flexible and can be used in other ways. For example, if the basic PDM approach was used for the *Smarties/Smarties* type where the pixel times were accumulated over all the frames, the challenge characters would significantly overlap and other techniques would have to be used to separate the characters. In the *Smarties/Smarties* challenge, characters gradually fade in and fade out, one by one from left to right. Some sample frames depicting this are shown in Fig. 3(a). Fig. 3(b) in turn shows a PDM that was



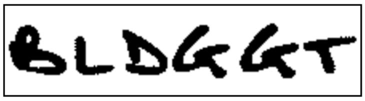
(a) Example frames from a *Smarties/Smarties* CAPTCHA challenge.



(b) Single PDM from all the frames.



(c) Three PDMs, each constructed from consecutive $\frac{1}{6}$ of the frames.



(d) Separate characters from six PDMs.

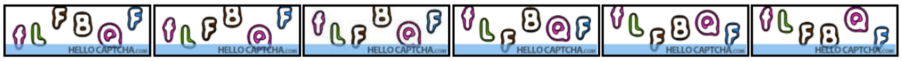
Fig. 3. Example of multiple PDMs from different portions of the animated frames

constructed using all the animation frames, one can see that the characters are joined together.

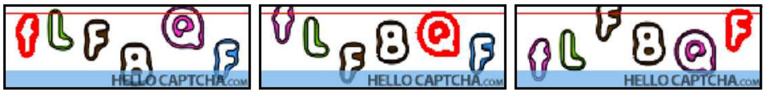
Nevertheless, in this situation we can construct six PDMs by splitting the animation frames into six distinct frame sequences. This is because challenges in HelloCaptcha always consist of six characters. In other words, a PDM is constructed by taking frames 0 to $\frac{1}{6}$ the total number of frames, another is constructed from frames $\frac{1}{6}$ to $\frac{2}{6}$ the total number of frames, another from frames $\frac{2}{6}$ to $\frac{3}{6}$ the total number of frames, and so on. The first three PDMs resulting from this approach are shown in Fig. 3(c). Consequently, the highest values from all the six resulting PDMs can be extracted giving the six separate characters that are depicted in Fig. 3(d).

For all the different HelloCaptcha types in the *Text Flood* and *Mass Flood* categories, a PDM can be constructed from the first $\frac{1}{3}$rd of the animation frames. This is because the relevant information to solve the challenge only appears during the first $\frac{1}{3}$rd of the animation then disappears and is no longer displayed again in the remaining frames.

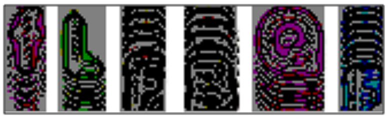
Extraction by Catching Line (CL). The *Spring/Jumpers* HelloCaptcha type is an example of a type where the Catching Line (CL) approach can be used to extract a single image. In the *Spring/Jumpers* type, the characters constantly move in a vertical direction. This is depicted in Fig. 4(a), where the characters appear to 'spring' or 'jump' up and down. The security is based on the fact that



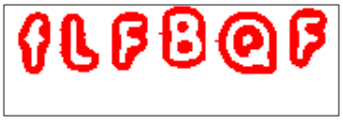
(a) Example frames from a *Spring/Jumpers* CAPTCHA challenge.



(b) Example of the virtual CL and 3 ‘caught’ characters (in red).



(c) Character vertical moving areas.



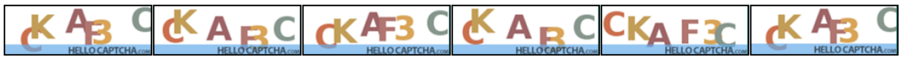
(d) ‘Caught’ characters.

Fig. 4. Example of character extraction using the Catching Line (CL) approach

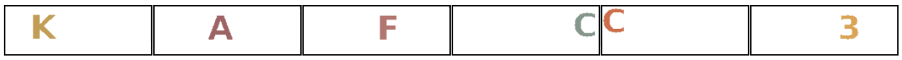
during individual frames, the upper or lower parts of certain characters may be hidden from view. However, the full characters appear when moving in the central area of certain frames. The idea behind the CL approach is to ‘catch’ (get an image of) full characters. This can easily be done by getting an image of the full character the very first time the individual character ‘touches’ a virtual line which is located at a position $\frac{1}{10}$th from the upper boundary of the animated GIF, as portrayed in Fig. 4(b). Since all the characters only move in the vertical direction and do not overlap, one can easily determine the image area to capture based on the characters’ moving areas, as shown in Fig. 4(c). Fig. 4(d) shows an example of an extracted image after catching all six characters.

Extraction by Color Selection (CS). Color can also be used to break a CAPTCHA scheme. The *Spring/Autumn Water* HelloCaptcha type is an example type where a Color Selection (CS) approach can be used to extract relevant information into a single image. The *Spring/Autumn Water* animated CAPTCHA challenge is composed of six moving characters on a white background, as depicted in Fig. 5(a). The characters move at random speeds and directions so they often overlap one another or are only partially visible if partly outside the image boundary. The color of each character is also random and they typically have different colors in a challenge.

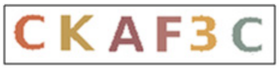
We simply use the distinct colors to extract individual character images from a ‘good’ frame. A good frame is defined as one where the total number of pixels which make up the character is at a maximum. In other words, because the characters have distinct colors we can count the number of pixels of a particular color contained in each frame. A frame where the maximum number of pixels for a particular color is displayed, indicates that the character is not overlapped by other characters nor is it partially outside the image boundary. Hence, we can do this separately for the different colors, as shown in Fig. 5(b). The resulting extracted image is shown in Fig. 5(c).



(a) Example frames from a *Spring/Autumn Water* CAPTCHA challenge.



(b) Characters separated based on color.



(c) Extracted characters.

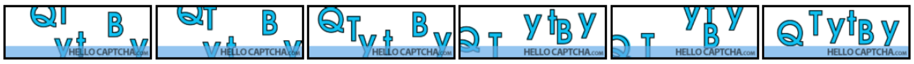
Fig. 5. Example of character extraction using the Color Selection (CS) approach

Extraction by Frame Selection (FS). Unlike the *Spring/Autumn Water* type where each character has a different color, the *Spring/Default* type uses the same color for all the characters. Therefore, to extract the characters a different approach is required. In this Frame Selection (FS) method, the 'best' frame is chosen as one that satisfies all, or at least two, of the following conditions:

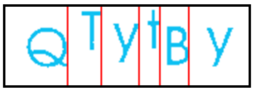
1. The characters can fully be separated into six distinct regions. To find this we use the Vertical Segmentation approach as described in Yan and Ahmad [15], in which a histogram representing the total number of pixels per column is constructed and the columns containing the characters can easily be identified.
2. None of the characters are in contact with the image boundaries. This condition is fulfilled as long as the border pixels all contain the background color.
3. Each character is made up of the maximum number of pixels. By counting the total number of connected character pixels, the total number of pixels that represents each of the six characters can be determined. The frame in which all characters are made up of the maximum number of pixels is selected.

An example of a sequence of frames from the *Spring/Default* type is provided in Fig. 6(a). Fig. 6(b) shows a frame where all three conditions are satisfied. On the other hand, Fig. 6(c) shows an example of a frame where only conditions 2 and 3 are met (condition 1 is not satisfied as the pixels of 'E' and 'T' overlap in the vertical direction). Nevertheless, it can be seen that the characters can still clearly be identified in that frame.

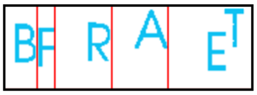
Extraction by Roller Selection (RS). The animation for all types in the *Roller* category is such that each character rotates around its center, starting from different rotation positions and rotating at different speeds. The *Roller/Rolling Clot* type is an example that uses this character rotation approach, as can be seen in Fig. 7(a). As the rotation progresses, at times characters may be connected or joined together. To obtain separate characters for



(a) Example frames from a *Spring/Default* CAPTCHA challenge.

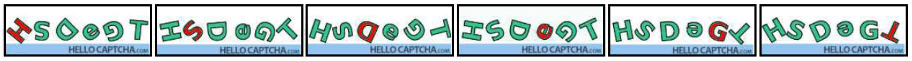


(b) All 3 conditions met.

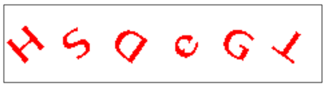


(c) 2 of 3 conditions met.

Fig. 6. Example of character extraction using the Frame Selection (FS) approach



(a) Example frames from a *Roller/Rolling Clot* CAPTCHA challenge.



(b) Extracted characters.

Fig. 7. Example of character extraction using the Roller Selection (RS) approach

these HelloCaptcha types, a Roller Selection (RS) method was used to extract the characters. The RS approach was done as follows:

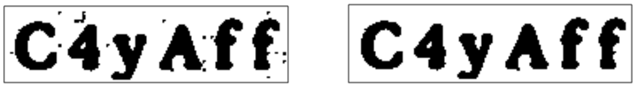
1. Perform flood-fill starting from the center pixel of each the six characters (each character always rotates around its center). The flood-fill algorithm will give us the connected pixels of the same color, hence, we can obtain six individual characters.
2. Select the character when its highest pixel reaches its maximum height, as shown in Fig. 7(a). This was done to reduce the character recognition training set required by the OCR program, discussed later in section 4.4 below.

The end result is a set of six separate characters, at rotation angles that produce their maximum heights. This is shown in Fig. 7(b).

4.3 Pre-processing and Character Recognition

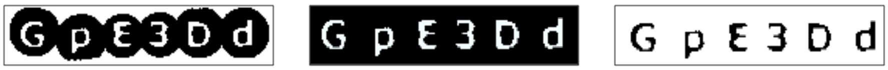
The image extraction stage produces a single image with all the characters. This image may contain noise and other impediments. Thus, to improve character recognition, a pre-processing stage is conducted to clean up the image. Depending on the HelloCaptcha type, different pre-processing steps may be used. The pre-processing methods that were used are described as follows.

- **Noise removal.** After using the PDM extraction method on types like the *Noisy Mosaic/Autumn Ant* type. The binarized image may contain unwanted noise, as can be seen in Fig. 8(a). This can be removed by performing



(a) Before noise removal. (b) After noise removal.

Fig. 8. Example of noise removal from extraction image

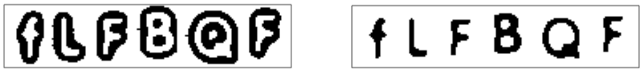


(a) Before circle removal. (b) Background filled in. (c) With colors inverted.

Fig. 9. Example of circle removal from extraction image

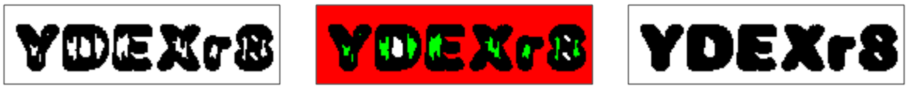
flood-fill on the different pixel clusters in the image and only saving the large areas. An example of the results can be seen in Fig. 8(b).

- **Circle removal.** In some types (e.g. *Roller/Cirque de Couleur*, *H-mover/Baller*, *Popup/Cirque*), the characters are encapsulated within circles, as shown in Fig. 9(a). We simple flood-fill the background starting from the image border, as depicted in Fig. 9(b) and invert the colors to obtain the image shown in Fig. 9(c).
- **Outline removal.** A similar process must be done for types where the outline of the characters are obtained, as shown in Fig. 10(a). The reason for this is that the OCR program typically achieves lower accuracy for characters displayed solely by their outlines. However, in these character outline cases care must be taken for characters such as the letter 'Q', in Fig. 10(a), which may contains a small internal white region. For these cases, we flood-fill the white internal regions of the characters to get the connected pixels and only save the large areas, resulting in the image shown in Fig. 10(b).
- **Refine by filling.** For certain types, the extracted image contains of characters with unwanted 'holes'. An example of this can be seen in Fig. 11(a). To remove these holes, we first obtain the character bounds by flood-filling the background (depicted in red in Fig. 11(b)), then flood-filling in the connected pixels within the characters which are less than 5 pixels from the character bounds (shown as green in Fig. 11(b)). This fills in the unwanted 'holes' whilst preserving the correct hollow regions for letters such as 'D' and digit '8' as can be seen in Fig. 11(c).



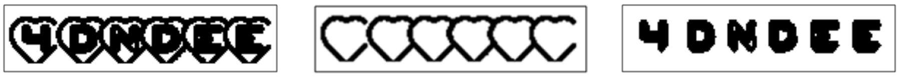
(a) Before outline removal. (b) Outlines removed.

Fig. 10. Example of removing the outlines surrounding the characters



(a) Before filling. (b) Removing 'holes'. (c) After filling.

Fig. 11. Example of refining the extracted image by filling in the unwanted holes



(a) Before shape removal. (b) Reference image. (c) After shape removal.

Fig. 12. Example of shape removal from the extracted image

- **Shape removal.** The *Popup/Love* type is an example of a HelloCaptcha type where each character is encapsulated within a heart shape, as can be seen in Fig. 12(a). To remove these shapes, we obtain a reference image containing the shapes, shown in Fig. 12(b), and subtract this from the extracted image. Fig. 12(c) shows the results with the shapes removed.

4.4 Character Recognition

After extracting the characters into a single image and cleaning up the image, the result is an image with six separate and distinct characters. All that remains is to perform character recognition. This is straightforward and can be done using any good OCR program. For this, we used the ABBYY FineReader 11 Professional Edition [1], which is one of the best OCR programs currently available on the market.

For more accurate results, the ABBYY FineReader uses a machine learning approach that can be trained from a training set of character samples. For some HelloCaptcha types, e.g. all the types in the *Flitter* category and for the *Search Light/Grun Ninja* type, we created a training set to be used in conjunction with the ABBYY FineReaders's existing embedded training database. For types such as types in the *Roller* category and *Swapper/3D*, we only used our own training set. Furthermore, for types that only used a selection of uppercase letters and digits, these were defined as the input language for the OCR.

5 Results and Discussion

An experiment was performed to test the accuracy of our methods. A total of 8,400 animated CAPTCHA samples were collected from the HelloCaptcha website [12] (i.e. 100 samples for each of the 84 different types). Overall, our approach achieves a high degree of accuracy. We can automatically distinguish between the different HelloCaptcha types between 80%−100% of the time. Most types can correctly be distinguished 100% of the time. In addition, the accuracy of breaking the different types (i.e. correctly recognizing all six characters in the animated CAPTCHA challenges) ranges between 16%−100% of the time. As stated in Bursztein et al. [3], a CAPTCHA scheme that can be broken more that 1% of the time is essentially broken. Our experiment was conducted on an Intel Core 2 Duo 3.33GHz PC and the average attack speed was around 4 seconds per challenge, which is well within the length of time that it would take for a normal human to solve the challenge.

5.1 Method of Attack

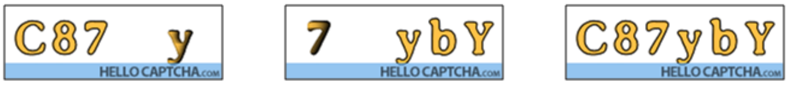
The PDM method that was presented in section 4.2, was the most commonly used approach to breaking HelloCaptcha. It was used to break 61 of the 84 different types. As CAPTCHAs have to be both usable and secure, designing a robust CAPTCHA scheme is a nontrivial task. From a usability standpoint, most CAPTCHAs are designed in a way where the important information is emphasized in order for the human brain to identify the main information required to solve the CAPTCHA. In a number of animated CAPTCHAs, the important information is emphasized by displaying it for longer periods of time. This is a fundamental problem that can be exploited using the PDM method. If one were to change the animated CAPTCHA design by displaying all information for equal lengths of time, this may severely impact the usability of the animated CAPTCHA, as no information really ‘stands out’ for a human user.

We observe that the PDM method can also be used to attack animated CAPTCHAs other than HelloCaptcha. Similarly the CL approach can also be used to break other animated CAPTCHA schemes which have properties whereby characters only move in the vertical direction. Other methods such as the CS, FS and RS methods, previously discussed in section 4.2, are possibly less general and may only be effective against specific animated CAPTCHAs.

5.2 Security Issues

One of the primary security issues in HelloCaptcha is that it was not designed with the segmentation resistant principle in mind. As such, individual characters can easily be extracted from the animation frames by exploiting a number of key features. Other than the segmentation resistant principle, several other security issues affect the overall robustness of a CAPTCHA scheme and should be considered when designing animated CAPTCHAs. We highlight a number of these issues in this section.

- **Number of characters and positions.** The PDM method is most effective for animated CAPTCHAs where the characters appear at fixed locations against a moving background/foreground. This is because the characters are displayed at relatively fixed locations for longer periods of time compared to the constantly changing noise. Furthermore, HelloCaptcha challenges always contain six characters which makes it predictable and easier to break. The issues of fixed length CAPTCHA challenges that appear at fixed locations are

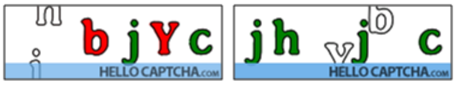


(a) Frame 45. (b) Frame 82. (c) Combined image.

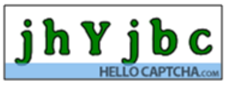
Fig. 13. Example where 2 halves of the challenge always appear in the specific frames

not unique to animated CAPTCHAs, it has previously been highlighted that this is a design issue that affects the robustness of CAPTCHAs in general [15].

- **The use of color.** The use of color is another common issue that has to be considered in the design CAPTCHAs, whether animated or otherwise. The incorrect use of color can negatively affect the security of a CAPTCHA, and may possible even affect the usability [16]. In a number of HelloCaptcha types, color played a negative role in the security of the CAPTCHA as important information could be obtained by identifying the different colors.
- **The number of frames.** The number of frames used in the animation is an issue that is unique to animated CAPTCHAs. A fixed number of frames can have a negative impact of the security of animated CAPTCHAs. In our study, we used the number of frames as one of the factors to differentiate between the various HelloCaptcha types. Furthermore, specific frames may be used to obtain certain pieces of information. For example, by evaluating 1,000 samples of the *H-mover/Default* type, we observed that it always uses a the total of 123 frames. In addition, frame number 45 always contains the first three characters and frame number 82 always contains the last three characters. This can be seen from the examples shown in Fig. 13(a) and Fig. 13(b) respectively. We can simply combine the first half of frame 45 with the second half of frame 82 to form the image shown in Fig. 13(c). Therefore, to improve the security of an animated CAPTCHA scheme, it is advisable to use a variable number of frames with information randomly distributed over the frames, even if this means increasing the complexity of generating the animated CAPTCHAs.
- **The frame delay.** If care is not taken, the frame delay can also be exploited to break animated CAPTCHAs. For example, in the *Swapper/Default* HelloCaptcha type, the correct solution is contained in the frame which has the longest delay time. Fig. 14(a) shows two frames with a frame delay time of 40 ms, note that the solution is only partially visible in these frames. However, Fig. 14(b) shows the frame with a frame delay time of 200 ms. One can see that the complete solution is clearly visible in this one frame. In such cases, the addition of the time dimension is completely redundant.
- **Method of delivery.** Animated GIF files are effective on the Web when used for small animation as such as animated CAPTCHAs because of its low implementation cost and it works on most browsers without the need for special plugin applications.However, it is also easy for attackers to download animated GIF files to extract and analyze all the animation frames. It has been argued that video streaming an animated CAPTCHA that is rendered on the client's browser is a more secure delivery mechanism [11].



(a) Examples of two 40 ms frames.



(b) A 200 ms frame.

Fig. 14. Example where frame delay time can be exploited

6 Conclusion

In this paper, we examined the robustness of an animated CAPTCHA scheme. The addition of the time dimension is meant to increase its robustness against automated attacks. However, this paper has demonstrated that if not designed well, the addition of the time dimension does not make the animated CAPTCHA more secure. We showed that simple but effective attacks can be used to break animated CAPTCHAs by exploiting certain characteristics in the CAPTCHA design. Our methods can successfully break all the different HelloCaptcha types with a high success rate. It is likely that our approach can be extended to break other animated CAPTCHA schemes.

References

1. ABBYY. ABBYY FineReader, `http://finereader.abbyy.com/`
2. Athanasopoulos, E., Antonatos, S.: Enhanced CAPTCHAs: Using Animation to Tell Humans and Computers Apart. In: Leitold, H., Markatos, E.P. (eds.) CMS 2006. LNCS, vol. 4237, pp. 97–108. Springer, Heidelberg (2006)
3. Bursztein, E., Martin, M., Mitchell, J.C.: Text-based CAPTCHA Strengths and Weaknesses. In: Chen, Y., Danezis, G., Shmatikov, V. (eds.) ACM Conference on Computer and Communications Security, pp. 125–138. ACM (2011)
4. Chellapilla, K., Larson, K., Simard, P.Y., Czerwinski, M.: Designing Human Friendly Human Interaction Proofs (HIPs). In: van der Veer, G.C., Gale, C. (eds.) CHI, pp. 711–720. ACM (2005)
5. Chow, Y.-W., Susilo, W.: AniCAP: An Animated 3D CAPTCHA Scheme Based on Motion Parallax. In: Lin, D., Tsudik, G., Wang, X. (eds.) CANS 2011. LNCS, vol. 7092, pp. 255–271. Springer, Heidelberg (2011)
6. Cui, J.-S., Mei, J.-T., Zhang, W.-Z., Wang, X., Zhang, D.: A CAPTCHA Implementation Based on Moving Objects Recognition Problem. In: ICEE, pp. 1277–1280. IEEE (2010)
7. Fischer, I., Herfet, T.: Visual CAPTCHAs for Document Authentication. In: 8th IEEE International Workshop on Multimedia Signal Processing (MMSP 2006), pp. 471–474 (2006)
8. Li, S., Shah, S.A.H., Khan, M.A.U., Khayam, S.A., Sadeghi, A.-R., Schmitz, R.: Breaking e-Banking CAPTCHAs. In: Proceedings of the 26th Annual Computer Security Applications Conference, ACSAC 2010, pp. 171–180. ACM, New York (2010)
9. Mori, G., Malik, J.: Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA. In: CVPR (1), pp. 134–144 (2003)

10. Naumann, A.B., Franke, T., Bauckhage, C.: Investigating CAPTCHAs Based on Visual Phenomena. In: Gross, T., Gulliksen, J., Kotzé, P., Oestreicher, L., Palanque, P., Prates, R.O., Winckler, M. (eds.) INTERACT 2009. LNCS, vol. 5727, pp. 745–748. Springer, Heidelberg (2009)
11. NuCaptcha Inc., NuCaptcha, `http://www.nucaptcha.com/`
12. Program Produkt, HelloCAPTCHA, `http://www.hellocaptcha.com/`
13. von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: CAPTCHA: Using Hard AI Problems for Security. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 294–311. Springer, Heidelberg (2003)
14. Yan, J., Ahmad, A.S.E.: Breaking Visual CAPTCHAs with Naive Pattern Recognition Algorithms. In: ACSAC, pp. 279–291. IEEE Computer Society (2007)
15. Yan, J., Ahmad, A.S.E.: A Low-Cost Attack on a Microsoft CAPTCHA. In: ACM Conference on Computer and Communications Security, pp. 543–554 (2008)
16. Yan, J., Ahmad, A.S.E.: Usability of CAPTCHAs or Usability Issues in CAPTCHA Design. In: Cranor, L.F. (ed.) SOUPS. ACM International Conference Proceeding Series, pp. 44–52. ACM (2008)