Easy Does It: More Usable CAPTCHAs

Elie Bursztein
Google
elieb@google.com

Angelika Moscicki
Google
moscicki@google.com

Celine Fabry
contact@celine.im

Steven Bethard
University of Alabama at Birmingham
bethard@uab.edu

John C. Mitchell
Stanford University
jcm@cs.stanford.edu

Dan Jurafsky
Stanford University
jurafsky@stanford.edu

ABSTRACT

Websites present users with puzzles called CAPTCHAs to curb abuse caused by computer algorithms masquerading as people. While CAPTCHAs are generally effective at stopping abuse, they might impair website usability if they are not properly designed.

In this paper we describe how we designed two new CAPTCHA schemes for Google that focus on maximizing usability. We began by running an evaluation on Amazon Mechanical Turk with over 27,000 respondents to test the usability of different feature combinations. Then we studied user preferences using Google's consumer survey infrastructure. Finally, drawing on the insights gleaned during those studies, we tested our new captcha schemes first on Mechanical Turk and then on a fraction of production traffic. The resulting scheme is now an integral part of our production system and is served to millions of users. Our scheme achieved a 95.3% human accuracy, a 6.7% improvement.

Author Keywords

Security; CAPTCHA; World Wide Web; Empirical Methods; Quantitative Usability Testing and Evaluation; User Studies;

ACM Classification Keywords

K.6.5. Management of computing and information systems: Security and Protection

Introduction

Distinguishing computers from humans is a central issue for website security. For example, GMail must prevent abuse by automated spammers, eBay must stop bots from flooding its site with scams, and Facebook must block the proliferation of fake profiles used to send spam and cheat at games. Captchas [18] [1] allow websites to make this distinction automatically. Captchas usually take the form of distorted or rotated sequences of characters that are easy for humans to recognize, but are hopefully difficult for computers to process. Over the last years at Google, in collaboration with Stanford University, we have been working on designing usable and yet effective captchas schemes using a systematic design approach.

This work focuses on designing a high usability captcha for use in cases with low risk, where the captcha is used primarily as a means to elicit an interactive engagement with the end user. As with any security mechanism, captcha design involves a tradeoff between security and usability. An easier captcha can be used when the impact of abuse is low, or when it has been verified by other means that the user is unlikely to be an attacker (e.g., from a trusted IP address). However, even in the low risk case, designing a usable captcha is still a challenge.

We ran an extensive study of visual features that are used in captchas (e.g., adding a line) and how they interact in order to understand how they affect the difficulty of the captcha, and alter user perception. Our user preference study shows that these two are not in fact the same.

Based on the insights gleaned during this study, we were able to devise a new captcha scheme that increased our users' pass rate by 6.7% over our previous scheme. We were able to do this without compromising the security of our system, but our system does not rely on captchas in isolation.

We began by identifying a set of distinctive security features that are common across the thirteen most widely used captcha schemes [5, 6] or mentioned in previous work on captcha usability [8]. We supplemented them with visual features described in human legibility and readability litterature [3,14,17]. Our feature set includes basic text features such as character set, font size and color, anti-segmentation techniques such as overlapping characters, and anti-recognition techniques such as character rotation. We implemented an open source generator (`http://goo.gl/ZPejcY`), that allows us to create new captchas based on combinations of features.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CHI 2014, April 26–May 1, 2014, Toronto, Ontario, Canada.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2473-1/14/04..$15.00.
http://dx.doi.org/10.1145/2556288.2557322.

[1] We write this acronym in lower case for readability.

We conducted our study by presenting our generated captchas to users and measuring how the different features influenced solving time and accuracy. We examined the effects of each feature in isolation and analyzed the interaction between features by varying multiple features at the same time.

To conduct this study on a scale large enough to obtain meaningful results, we built an application for Amazon's Mechanical Turk[2] (AMT) and used it to record human performance on nearly a million captchas. To our knowledge, this study is the first large scale systematic study of captcha features.

Our work confirms some of the findings of previous work [8] on a large scale and in real-world conditions, while also expanding the set of features tested. We also explored how computer hardware (namely screen resolution) impacts human accuracy and solving time. We find that screen resolution does not have a significant impact on captcha accuracy or solving time.

Besides studying features in isolation, our study is also the first to systematically analyze how feature combinations (in particular pairwise interactions) affect user accuracy and solving time. We uncovered an important limit to systematic design: predicting the effect of interactions between features is currently intractable. Our analysis reveals that this limitation stems from the fact that the effect on accuracy of many feature interactions, 20.7% of the pairwise interactions, requires a nonlinear function to be described and there is no clear pattern to which interactions are nonlinear. This means that we can't reliably predict the outcome of a specific feature combination using statistical analysis or machine learning.

This led us to use an iterative approach with human testing to reach our final design. We offer multiple examples of how unexpected interactions impact accuracy, demonstrating how important iterative HCI methods are to the field.

Finally, our study of user preferences about captcha schemes reveals that perceived captcha difficulty is not necessarily aligned with their real difficulty. For instance, when words are used, the meaning and frequency of the words greatly affects user perception of difficulty. Negative words and low frequency words are perceived as harder to complete, whereas positive or high frequency words are perceived to be easier.

The accuracy we observed on AMT (97%) turned out to be very close to the accuracy we observed in production (95.3%), confirming that AMT is an effective way to approximate real world results. On the other hand, deploying our scheme on real world traffic allowed us to discover very subtle usability issues due to feature interactions that are only observable on extremely large sample size. This led us to conclude that AMT is a great tool for rapid iteration but it needs to be supplemented with large scale A/B testing on real traffic to polish captcha design.

The remainder of the paper is organized as follows: first we present our AMT experimental setup, then we report accuracy and solving time for each feature in isolation. Next we analyze feature combinations, and show why predicting feature interaction outcomes is currently intractable. Then we report the result of our user preferences study, and discuss how user perception of difficulty differs from the measured accuracy and solving time. Finally we discuss how we used what we learned to design Google's new captcha schemes, and conclude by discussing the lessons learned throughout this study and how they generalize beyond captchas.

[2] `http://www.mturk.com`

BACKGROUND AND RELATED WORK

The most similar previous work [8] mainly focused on testing a specific set of security features in a laboratory setting on a small population (76 participants). The authors studied the impact of a subset of the features studied in this paper, but they did not examine character collapsing, which is the most common security feature used today [6], or any aesthetic features. They also did not consider the interactions between features. In [2], the authors studied the usability of a specific captcha scheme called ScatterType, where parts of the characters are erased and the fragments are separated. They examined in a lab setting how different fonts affect solving accuracy. They also studied easily confusable characters such as 'c' and 'e' impact usability. In [4] the authors look at the usability issues in presenting audio captchas to humans and in [22] the authors study the effect of page aesthetics and feedback when running captcha tasks.

In [25], the authors discuss the usability of image-based captchas but didn't perform a user study. In [13] the authors study how to balance security and usability for video-based captchas. More recently, [23] revisited this topic and extended it to all sort of moving captchas. In [7], human performance on captchas was considered for both humans and computers but only for single-character recognition.

CAPTCHA FEATURES STUDY

The main goal of this study was to understand how each feature used in captchas affects user accuracy and solving time, both in isolation and in combination. First we established a baseline by measuring how well users solve captchas where the fonts, colors, etc. were all set to common web browser defaults. Then we measured how users responded to changes in individual features in isolation. Finally, we measured user response to variation in pairs of features. This set of experiments was run on AMT.

Users. We recruited participants for our study from Amazon's Mechanical Turk (AMT) [12], an online marketplace providing workers who solve Human Intelligence Tasks (HITs). We chose AMT primarily because it gave us access to a large number of participants, suitable for the scale of our study. AMT has also been used successfully for user studies in related fields [12, 19]. In all of our experimental settings, we presented AMT workers (colloquially,"Turkers") with a survey of basic demographic information like age, language, and education. Turkers were paid $.30 to solve a series of 20 captchas, one at a time, before answering the survey.

Turkers typed in their guesses for each captcha and we recorded their responses and solving time. 27,422 Turkers participated in the study according to Amazon's count.

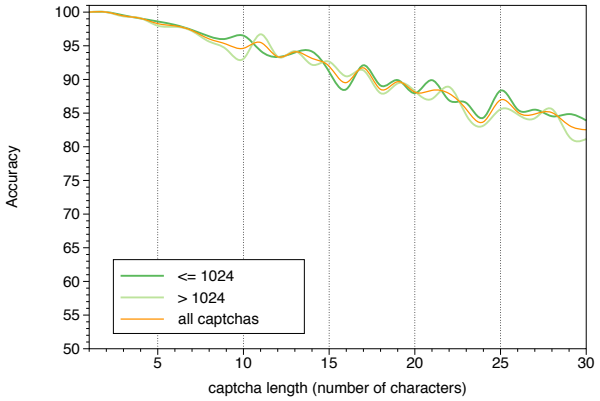


Figure 1. The non-impact of screen resolution on captcha solving accuracy.

Figure 2. Main set of captcha security features produced by our captcha generator.

As in previous studies using AMT [5, 19], most of our participants were between ages 18 and 35 (51%), though participants as old as 78 were reported. Many of our participants (35%) were also between the ages of 26-35, and the remaining participants were older than 36. Tamil was the most common native language of our participants (49%), with English a distant second (18%). There was a wide variety of other native languages reported. In order from most prevalent to least prevalent we observed: Malayalam, Hindi/Urdu, Telugu, Kannada, Marathi, Gujarati, and Bengali. About 7% of our participants reported native languages other than these. Finally, for most of our participants (58%) the maximum education level was a bachelors degree, though there were a moderate number of participants with only a high school education (20%), and a moderate number of participants with a masters degree (19%). Overall, these demographics are comparable to those observed in [5]. We do note that our Turker group is slightly younger and more Indian centric than in [19].

The non-impact of hardware. We were concerned by the impact of screen resolution, as captcha solving is a visual task. To understand whether screen resolution introduced a bias, we ran an experiment with Turkers solving our baseline captchas with length from 1 to 30 characters (1,000 samples for each length; 30,000 captchas total). We captured screen resolution programmatically using JavaScript.

We found that screen resolution did not impact the accuracy or the solving time of our participants. Figure 1 shows that the accuracy is similar for resolutions both greater than and less than 1024 by 768 pixels.

Captcha features

We identified a set of features based on the thirteen most widely used captcha schemes mentioned in [5] and from previous work [11] done in a small lab experiment. We divided our features into three categories:

1. *visual features* unrelated to security but relevant to usability such as text size and font color.
2. *anti-segmentation features* used to prevent programs from automatically separating the captcha into individual letters such as adding a line, collapsing characters.
3. *anti-recognition features* used to impede recognition of individual letters such as character rotation.

Visual features. The *visual* features are familiar parameters that can be relevant to usability but that are generally unimportant for automated captcha solvers.

- *Character sets*: lowercase letters (*a-z*), uppercase letters (*A-Z*), digits (*0-9*), lowercase letters and digits (*a-z0-9*), uppercase letters and digits (*A-Z0-9*), letters (*a-zA-Z*), letters and digits (*a-z0-9*), non-confusable letters and digits (*a-z0-9, except for 0, o, 1, l, i, 9 and g*), non-confusable letters (*a-z, except for o, l, i, g*), words (*a-z, but must be a valid English word*) and pseudo-words (*a-z, but derived from a character bigram model trained on English words*). Note that unlike the other character sets, for words and pseudo-words characters are not generated uniformly - there is a bias towards English-like character distributions.
- *Character counts*: between *4* and *15*. This matches the range of lengths observed for captchas in the wild.
- *Font sizes*: between *6px* and *36px*, in 2px increments.
- *Font families*: *Arial*, *Verdana, Times*, *Aescrawl*, *AfterShock*, *Atlandsketchesbbreg*, *BleedingCowboys*, *CardiffRegular*, *CrashcourseBBreg*, *DoktorTerror*, *Erthqake*, *FutureSallow*, *Grandesignneueserif*, *GreaseBalls*, *Imitation*, *Incubus*, *Jag*, *JaggaPoint*, *Jellykaestryahandwriting*, *Justusoldstyle*, *Phandv2*, *Slammertag*, *SlicedIron*, *Under*, *WhiteLineFever3D100*.
- *Foreground colors*: *black, white, gray, pink, red, orange, yellow, green, cyan, blue, purple*. The foreground color was used both for the font and for the defenses (e.g. lines, random noise) – using different colors would make the captcha insecure as the non-text color could be easily stripped away.
- *Background colors*: the same as foreground colors.

Anti-segmentation features. Some captcha features are specifically designed to make it more difficult for machines to automatically determine where characters begin and end. Preventing segmentation is central for captcha security. It has been shown in [24] that most of the captcha security comes from the inability of machines to segment captchas into individual characters. We refer to such features as *anti-segmentation* features, and consider the following variants:

- *Character overlaps*: gaps of *-15px* to *+5px* between successive characters.
- *Random dot sizes*: between *2px* and *10px*.
- *Random dot counts*: between *500* and *12,000* dots per captcha, in 500 dot increments.
- *Line types*: *straight* or *wavy*. Wavy lines are generated by randomly picking adjacent points.
- *Line counts*: between *1* and *10* lines.
- *Line widths*: between *1px* and *10px*.
- *Line positions*: lines start to the left of the first character, allowing the vertical position of the start to range from 10% below the first character to *10%* above, at 10% intervals. Similarly, lines end to the right of the last character, again allowing the vertical position of the endpoint to range from 10% below to 10% above.
- *Similar foreground/background colors*: the background color is selected by moving from the foreground color in small steps in each possible direction in RGB space.

Anti-recognition features. Some features are designed to make it more difficult for attackers to recognize individual characters after segmentation. We refer to such features as *anti-recognition* features and consider the following variants:

- *Rotated character counts*: between *1* and *6* characters are rotated
- *Rotated character degrees*: between *10* and *350* degrees, at 10 degree increments. All characters are rotated the same amount.
- *Vertical shifting sizes*: between *5px* and *60px* at 5px increments. How much the character is shifted within the defined range is selected at random.
- *Character size variations*: differences from the previous character size between *2px* and *20px*, in 2px increments.
- *Character distortions*: based on attractor points, the basic idea is to pick two random points on the image to create a distortion field. Random perturbations to the field are added to make sure the deformation is not predictable and then the field is used to stretch and translate pixels based on their position in the field.

Experimental setup

We ran several rounds of experiments, summarized in Table 1. We began by establishing a baseline, intended to characterize the least obtrusive captcha that looks as much like regular browser text as possible. The baseline captcha consisted of 6 lowercase letters and digits in 20 pixel (15 point) black Arial font on a white background, with no anti-segmentation or anti-recognition techniques. We had 1,000 different captchas annotated so that we could measure accuracy with 0.1% precision.

Next we considered the effects of individual features in isolation. We enumerated 496 possible variations of our feature set. For example, a variation might be a captcha containing one line. Another variation might be a captcha containing 3 lines, or 7 lines, etc., all with the same character set, length, font, and colors as the baseline. We had 200 captchas annotated for each variation in order to achieve 0.5% precision in our results. The order in which the captchas were presented was randomized in order to avoid a prediction bias.

Then we tested variations of two to four features simultaneously. This is intended to capture the observation in [5] that captchas in the wild rarely exhibit more than one or two anti-segmentation features and one or two anti-recognition features. There were too many possible feature combinations to test the space exhaustively, so we took a random sample of 110,950 combinations out of 115,315,858 possible combinations.

Finally, we used our insights to settle on a single captcha design, and ran a last round of tests to validate the quality of our design before testing on production traffic. A subsequent section is dedicated to this experiment and our production deployment results.

Individual feature analysis

We evaluated participant performance on captchas in two ways: solving accuracy and solving time. For solving accuracy, we compared the answers given by participants to the text from which we generated the captcha, ignoring differences in case or spacing. Solving time was measured using JavaScript by recording the time between when participants were presented with a captcha and when they submitted their response.

We filtered out our experiment results by removing responses that were done too slowly or too quickly using 3 standard deviations, and ones that took less than 4 seconds (19.8% of all responses). Measuring times on AMT can result in high variance because many Turkers do other things on their computer at the same time even if we explicitly ask them to work as fast as possible. Still, by averaging over large numbers of participants, we can get an idea of which captchas take more or less time.

We claim that Turkers' behavior closely reflects real user attention patterns while surfing, for example, from home in front of the TV, which makes Turkers a more accurate subject pool than students or manually recruited subject for Internet behavior related studies. This claim is supported by the fact, as reported later in the paper, that the accuracy results obtained with AMT match very closely the results we got on real world traffic

Round	Task	N possible	N sampled	N tests per sample	Total tests
1	Preliminary Experiment	30	30	1000	*30000*
2	Baseline (“Control”)	1	1	1000	*1000*
3	Features in isolation	496	496	200	*99200*
4	Feature interactions	115315858	110950	5-10	*804750*
5	new scheme design testing	8	8	2000	*16000*
	Total				*950.550*

Table 1. Overview of the experimentation rounds.

when we deployed our new capcha schemes and analyzed the results of tens of millions of captchas solved.

Character sets. Table 2 shows how participants performed on different character sets. Pseudo-words, words, and simple character sets like all digits, all lowercase letters and all uppercase letters were the easiest, with accuracies of *97%* or higher. Mixing letters, digits or uppercase and lowercase letters dropped accuracy down into the 80s, except in the case where we explicitly removed the easily confused letters.

Timing for the most part mirrored these trends, with words being the fastest, solved on average in only 4.3 seconds. While these results suggest that words and pseudo-words are the best choice, as we will discuss in the user preference study, word meaning and frequency greatly affect user perception and therefore should be used with caution. We note that *a-z0-9* can be used with minimal loss of speed or accuracy, as long as the confusable letters like *l, 1, o and O* are removed. However, as discussed in [6], using a larger key space does not increase captcha security, hence there is not a good reason to use a complex character set.

Number of characters We found that there is a slight decline in accuracy as more characters are presented. There is about a *1.5%* absolute drop for each additional character, and a moderate increase in time, about *1.0s* for each additional character. Thus, captcha designers can for the most part freely adjust the length of the captchas to achieve the desired level of security without harming the accuracy of their users, though there will be an additional time requirement.

Character Overlap Figure 3 shows that non-negative gap widths, including gaps of 0px, where characters touch all other characters, resulted in similar accuracies to the baseline *90-95%* range. Any amount of negative gap width however, caused dramatic drops in accuracy.

Charset	Time (s)	StdDev (s)	Accuracy
pseudo-word	5.0	2.2	99%
a-z	6.6	2.8	97%
0-9	6.1	2.5	98%
a-z0-9 (non-confusable)	6.6	2.6	97%
word	4.3	1.9	98%
a-z0-9	7.6	3.1	93%
a-zA-Z	8.2	3.6	88%
a-zA-Z0-9	8.3	3.4	82%

Table 2. Accuracy and solving time for various character sets

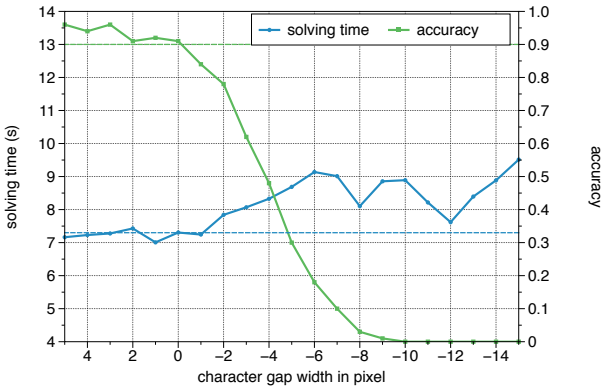


Figure 3. Accuracy and solving time vs. character overlap.

These results suggest that there is a hard limit to how much overlap between characters can be used to prevent captcha segmentation by automated solvers and still retain a tolerable user accuracy.

Font Size Participants performed similarly across different font sizes, with an accuracy standard deviation of only *1.7%*, and a time standard deviation of only *1.2s*. Thus, captcha designers can freely select whatever font size best fits their security design, without worrying about human performance issues.

Font Family Performance varied widely across fonts, with participants being most accurate on Arial and Verdana fonts, *97%* and *94%* respectively. There area some extreme differences between the other fonts – from an accuracy of *89%* on Grandesign Neue Serif to *1%* on Jellyka Estrya’s Handwriting. This makes the choice of font family difficult without user testing.

Foreground and Background Color Like font size, varying the foreground and background colors generally had only a minimal effect, with an accuracy mean of *91.5%* and standard deviation of *3.5%*. Timing had a mean of *8.2s* and a standard deviation of *1.0s*. The two noticeably worse combinations were yellow-on-white (*84%* accuracy and *8.5s*) and white-on-yellow (79% accuracy and 9.8s). So colors may be tuned for the most part as desired, except for a few bad combinations.

Random Dots Participant performance was no worse than baseline with up to 2,000 random 1 pixel dots. However, for every additional 1,000 dots after that, accuracy declined about *6%* and solving time increased by *0.7s*. Thus, the random dot noise can be tuned a bit, but after more than 2,000 dots, human performance will suffer.

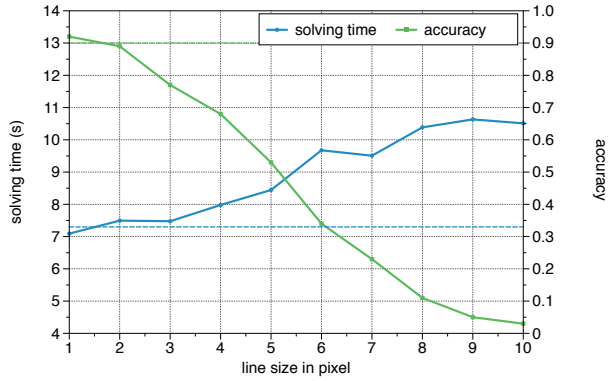


Figure 4. Accuracy and solving time vs. line size

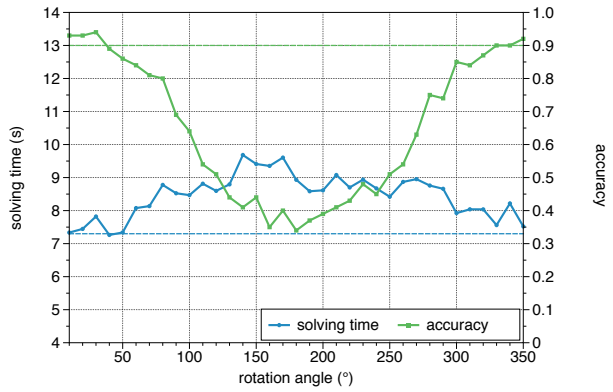


Figure 5. Accuracy and solving time vs. rotation angle

Lines We found that with up to three straight 1 pixel lines, participant performance was similar to that of no lines. Every additional straight line cost about 6.0% in accuracy and 0.5s in time. We also found that 1 or 2 pixel wide lines did not impair human performance, but thicker lines quickly hurt in both accuracy and time. See Figure 4. Wavy lines always impaired performance (figure not shown). These results suggest that a small number (1-3) of thin (1-2px) straight lines in any orientation can be added to captchas without noticeably impairing human performance.

Rotated Characters Angles of rotation within approximately 30 degrees of vertical had minimal effect on accuracy and solving time, as shown in Figure 5. The worst accuracies and slowest responses occurred with fully inverted characters (180 degrees). These results suggest that humans can tolerate small character rotations with minimal impact on performance. However, a larger rotation can have a much more dramatic impact on human accuracy.

Vertically Shifted Characters Shifting characters up and down in the captcha generally had only a small effect, and then only after shifts of 50 pixels or more. It is unclear how effective such shifting is as a security measure, but at least doing so has only a minimal effect on the readability of the captcha.

Character Size Variation Similar to vertically shifting characters, varying the size of characters generally did not impair our participants unless there was more than a 12 pixel difference in character size.Thus, character size variation is a reasonably safe feature to tune.

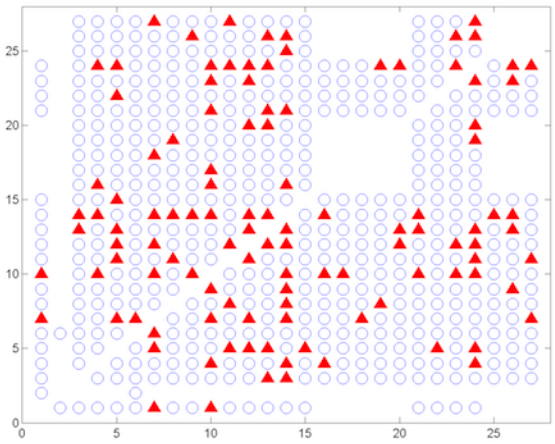


Figure 6. Scatter plot representing the type of interaction of pair-wise features (2d) accuracy. X and Y axis are features.

Feature Interactions

It turns out that many features are not wholly independent, thus it is also important to understand how features interact. One of our hopes was that we might be able to build a model to predict the accuracy and solving time of a captcha scheme given its features. However, in our experiments with various machine learning algorithms (e.g. SVM with various type of kernels) we were never able to beat the baseline.

Puzzled by how ineffective machine learning algorithms were on our dataset we shifted our goal to characterize the complexity of interactions between features. We used a function-fitting approach [16] that to find the simplest known function that explains a data distribution. For each pairwise interaction, we used the least squares method [20] and tested multiple functions: *additive, convolutive, exponential, logarithmic, square root, and polynomial (orders* 2 *to* n*)*. We kept the simplest one that had an overall error .001 or less. This is consistent with the Occam's razor principle that the best explanation is the simplest one.

We plotted the results of this procedure to show the types of functions needed to explain the resulting accuracy (Figure 6) for the various pairwise interactions. On this scatter plot the *blue circles* represent interactions that need a linear function to be explained. The *red triangles* represent the interactions that require a higher-order function to be explained. Typically this was the exponential function or a high order polynomial function. White space represents feature interactions that are meaningless as they create impossible to solve captchas, for example, black text on black background. Similar results were obtained for solving time. Figure 6 highlights two essential properties of our dataset that explain why it is so difficult to apply machine learning algorithms on it. First, higher order functions are needed to explain at least 20% of the feature

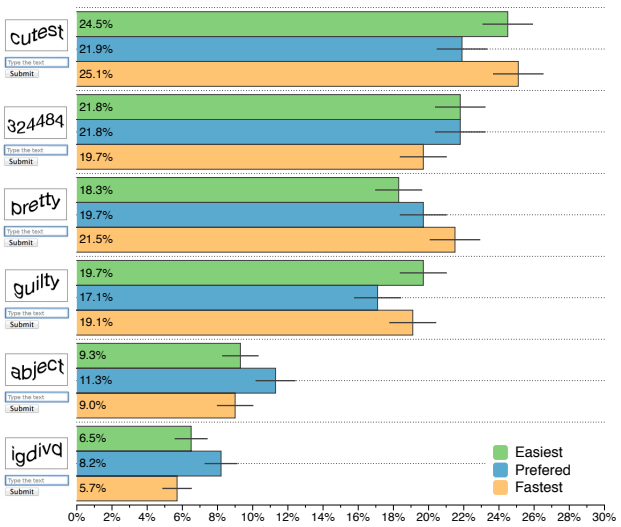


Figure 7. Word preferences

interactions (112/582 interactions for accuracy, 122/582 for solving time). Secondly, there is no discernible pattern that can be used to partition the set into features that require higher order functions and those that only need a linear function. As a result, building a predictive model that can explain this dataset accurately requires an extremely high order kernel function, which makes this problem intractable in practice. Remember here that we only analyzed pairwise interactions and that higher order interactions will require even larger higher order functions to be explained.

While there is still the possibility that deep belief networks or higher order machine learning algorithms might one day be able to predict the result of feature interaction, for now we are bound to rely on user testing and iterative design to validate captcha schemes.

USER PREFERENCES

We ran two Google consumer surveys [10] to understand how user perceived difficulty is related to solving time and accuracy. In each survey, we asked 5,000 respondents in the United States to answer three preference questions. We note that the Google consumer survey infrastructure ensures that the respondent population is representative of the US population [15]. The survey results are publicly available at `http://goo.gl/rkp4bZ`, `http://goo.gl/uYLaIj`.

Each question consisted of a handpicked set of captchas. In the first survey we show side by side 3 captchas where each captcha exhibits a distinct anti-segmentation feature: moderate noise, a single wavy line, and moderate character collapsing. The second survey consisted of 6 captchas with different character sets. Respondents were asked which captcha out of the group they thought was the *fastest*, which one was *easiest*, and which one they *preferred to solve*. The order in which the captchas were presented was randomized in order to avoid positional sampling artifacts.

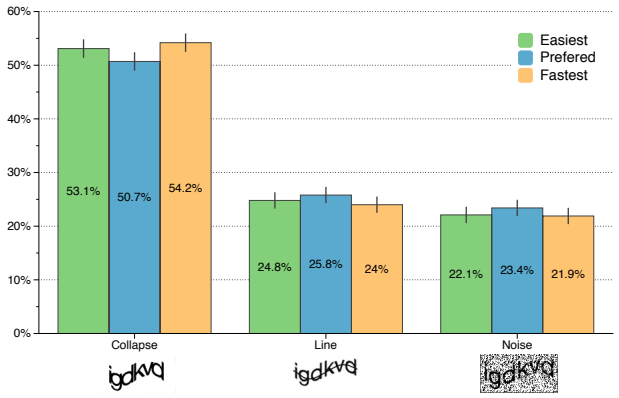


Figure 8. Security feature preferences

We found that word connotations play an important role in user perception. We chose the word *pretty* as an example of a high frequency positive word, and *cutest* as a low frequency positive word. Google search returned approximately 1.1 billion results for *pretty* and about 36 million results for *cutest*. We chose *guilty* as a high frequency negative word, and *abject* as a low frequency negative word. These words had 150 million and 4 million Google search results, respectively. Figure 7 shows a clear preference for positive words over negative words. It also suggests that low frequency words negatively impact user perception. The unconscious bias introduced by words led us to reject using words in our captcha schemes, since this bias makes it impossible to generate hundreds of millions of captchas with consistent user sentiment.

The lowercase meaningless string of letters is significantly ranked last in all three metrics considered, whereas strings of digits is users' second most preferred captcha, and second in terms of easiness, even though the result are not statistically significant when compared to positive words. We therefore chose to use digits as our character set.

The second survey reveals that users' preferences among-anti segmentation techniques, summarized in figure 8, align with their relative security strengths. According to [6], character collapsing is more secure than lines, which is in turn more secure than noise. That is, users happen to prefer more secure techniques. A possible explanation for this alignment is that users are more exposed to stronger security techniques used by prominent captcha schemes and are therefore more familiar with them.

These surveys show that accuracy and solving time are not good predictors of user preference. For example, our AMT results report in table 2 indicate that English words are solved the quickest and most accurately, whereas users perceive digits to be both faster and easier than words with negative connotations. Similarly, random lowercase strings have the same performance characteristics as random strings of digits, but users perceive digits to be faster, easier, and more likable.

PUTTING IT ALL TOGETHER: HOW WE DESIGNED OUR NEW CAPTCHA SCHEME

In this section we relate how we designed and deployed new, highly usable captcha schemes for Google using the knowledge gathered during our experiments and the guidelines for creating secure captchas devised in [6]. Overall our newly deployed captcha improved user accuracy by 6.7% (95.4% up to 88.5%). We also recount how the complexity of feature interaction caught us off guard several times and how we eventually overcame it.

Recall that our designs are for captchas for use in low risk situations, e.g., where they are combined with a comprehensive anti-abuse system. Therefore, we make design tradeoffs that aim to improve the user experience, and include only as many security features as are possible while still maintaining a very high accuracy rate.

Design choices

We decided to design two different schemes rather than just one; a main one, and a backup one in case the main one is broken.

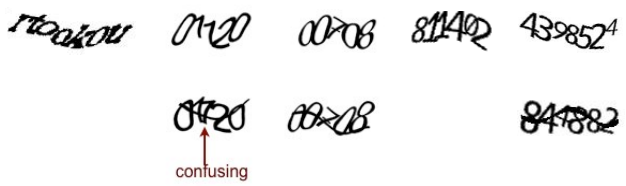
For the main scheme we chose to retain as many visual features as possible from our current captcha in order to give users a sense of continuity. We kept the captcha size, the text color (black) and the background color (white).

We used a different font though, for reasons recounted in the next subsection. We also simplified the character set by moving from lowercase letters to digits. This decision was motivated by the data in [6] which shows that a complicated character set does not increase security, since machine learning algorithms are equally good on all common character sets. On the other hand, as discussed above, humans are more accurate on simpler character sets. While humans are also very good with words, we chose to avoid them due to the sensitivity of user perception to word meaning as discussed in the user preference section.

Anti-segmentation techniques. Line and character overlap are the two recommended, but imperfect, techniques to make captchas resistant to automated attacks [6]. Following the insights provided by the user preference study, we decided to use character overlap for the main scheme and combine it with a line for the backup scheme. This strategy also fits well with our wish to maintain continuity with our previous captcha scheme which also used character collapsing.

Extra security features. To thwart state-of-the-art attacks against overlap and line-based defenses [1, 6], we added the following extra security features to our captcha schemes:

- **Length randomization**: We randomize the length of the captcha between 6 and 8 to prevent the attacker from knowing how many segments are present [6]. As discussed earlier, longer captchas have little impact on actual difficulty.
- **Text-size randomization**: Similarly we randomize the character size to prevent the attacker from guessing how many characters there are in the captcha by looking at its width [6].



Original	Mturk v1	Mturk v2	1% test	New
88.6%	89.2%	97%	94.3%	95.4%
	82.6%	92.2%		93.2%

Figure 9. Comparison between our previous captcha, the AMT experiments we conducted (v1 and v2) and our new scheme (1% test and current one). The accuracy is reported at the bottom of the figure. As shown in the v1 column, when the collapsing and waving features are combined with the digit-charset the 1 and 7 are easily confused.

- **Tilting**: We also randomly tilt each character up to 20 degrees to provide a third layer of defense against length guessing attacks.
- **Waving**: Finally we use sinusoidal based waving to prevent the attack where the segmentation is inferred by looking at various shape invariants (such as the S pattern) as explained in [1].

While none of these features guarantee that our captcha scheme is secure against highly sophisticated attacks [9], they ensure that our captcha is not trivially breakable and that it fulfills its role of keeping unsophisticated attackers at bay.

Early Experiments

As digit character set measured accuracy is 98% (table 2), we were expecting close to that number for our main scheme. Based on our experiments, we were also anticipating the backup captcha to be around 5% less accurate than the main one because of the line addition.

Similar to the previous experiment, we used AMT to quickly test our new designs before running a field trial on a small fraction of our production traffic. We asked Turkers to solve 2,000 captchas of our two new schemes. As reported in Figure 9, accuracy results did not live up to our expectations.

After reviewing Turkers' answers, it became clear that the poor performance was due to the unexpected interaction between digits, overlap and waving: in many cases it was impossible to tell apart 7 and the 1 as shown in Figure 9. To address this issue, we decided to use a font that had an extra bar on the 7. We ended up testing using the sans-serif open fonts *tenor-sans* that included a bar on the 7. We achieved an accuracy pretty close to what we expected from our earlier experiments: 97% accuracy for the main scheme and 92.7% for the backup one. This anecdotally supports the conclusion of our feature interaction analysis: feature interactions are nonlinear and therefore can lead to unpredictable effects. We confirmed that our performance issue was due to the confusion between 1 and 7 by re-testing with another font that didn't have a bar on the 7 and ended up with results that were even worse than our baseline font (Arial): 86.2% accuracy.

Real world results

We decided to deploy the main scheme on a small fraction of production traffic in a series of field trials on a particular Google web page. We note that Google's captcha is used in many other places than the page we consider in this paper. Each trial consisted of approximately 100,000 captchas. We chose to use a version of the *Oxygen-Mono* font that we modified by adding bars on the 7 and 0 glyphs to avoid the confusion identified in the earlier AMT experiment. We excluded empty responses from the results reported in this section, and responses that contained the word 'captcha' (we noticed an interesting phenomenon where users type in what the image semantically is).

As reported in figure 9, we were able to achieve a 94.3% accuracy rate with our main scheme. This is slightly lower than the AMT experiment, but within a reasonable margin of error. To our knowledge, this is the first large scale empirical verification where results obtained via AMT are reproducible in real world conditions. Analysis of the errors that users made revealed another unintended interaction of combining features that was not detected in our AMT screening: users infrequently confused 0 with 8 due to the distortion combination. To further improve accuracy, we removed 1, 7, and 0 from our character set and repeated the experiment. We improved the accuracy to 95.3%. We believe that the confusion between 0 and 8 did not appear in the AMT screening due to its low prevalence - less than 1% of captchas - and therefore requires a very large sample to clearly stand out. Overall however, we found that AMT (or any other crowd computing service) is a very valuable tool to iterate quickly on captcha design and get it mostly right before experimenting on production traffic. Finally, we also field tested the backup variant that includes a varying width curve drawn across the digits. We found that adding this line cost 2.2% accuracy.

Following these experiments, we fully deployed our main scheme on production traffic in early September 2013. Since then we have not observed a significant change in abusive activity. However, our anti-abuse defenses do not solely rely on captchas, so our results in this regard may not generalize to other companies. Overall we observed a consistent 95.3% accuracy rate over a sample of approximately 20 million captchas. This is a 6.7% improvement over an equivalent traffic sample taken immediately prior to the new scheme.

Our captcha infrastructure also has a feature for users to skip difficult captchas and get new ones. Since deployment, we observe that users request new captchas 55% less often. This, combined with the accuracy improvement, resulted in approximately 20% fewer incorrect captcha solutions submitted to the page.

LESSONS LEARNED

Balancing security and usability. Captchas were originally intended to solve a security problem: automatic verification of humanity. As automatic text recognition software (OCR) improves, solving this problem with distorted text in isolation is becoming increasingly difficult, as well as burdensome for users. While captchas are still an effective security tool [21], we find that it is much more effective to use the captcha as a medium for engagement with the end user, and examine the interaction holistically. We believe that this is a positive development for usability, as shifting the security burden away from the captcha image itself makes it possible to use less aggressive distortions. Deploying the new captcha scheme did not result in decreased security at Google, but as mentioned earlier, captchas are just one component of Google's anti-abuse system. It may be interesting future work to extend the analysis of captcha features by acquiring captcha breaking software and measuring the break rate for each feature along side the usability.

MTurk results generalize to real world. Some of the results in this work are applicable beyond designing new captcha schemes. In particular one of our key results is the first large scale empirical verification that results obtained via AMT are reproducible in real world conditions with the caveat that fine-tuning at very large scale is still necessary. Overall the accuracy of our new scheme is very close (1.4%) to what our MTurk experiments predicted. On the other hand, our MTurk evaluation failed to reveal infrequent confusion between 0 and 8, due to the relatively low prevalence of captchas where the confusion might occur (< 1%). Exposing such infrequent biases may be prohibitively expensive using paid services like AMT.

User perception of the task difficulty is affected by unrelated variables. Another lesson learned is that user perception of a task difficulty is significantly influenced by cues (e.g content) that is unrelated to the actual task difficulty. The difficulty of solving the captchas in figure 7 is technically identical - they are all the same length, same character set (except digits), distorted the same way. Yet, users preferred the top example over two and a half times more than the bottom one (21.9% vs. 8.2%).

CONCLUSION

In this work we recount how we designed new captcha schemes for Google using a systematic design approach. We deployed it in production without adversely affecting security, and achieved a 95.3% accuracy, which is a 6.7% improvement over the previous captcha scheme.

To design these new schemes we first selected a comprehensive list of features used in captchas from prominent real world captcha schemes and previous work. We then produced captchas with different combinations of those features, and ran a usability study on Amazon's Mechanical Turk on nearly a million captchas. This allowed us to identify the limits of how much distortion users can process and maintain a tolerable accuracy. A key finding was that results obtained on AMT generalize well to the real world; the user accuracy measured on AMT is very close to our production result. Another key finding is the nonlinearity of interactions between features, which prevents using machine learning or statistical models to predict the accuracy of a given capcha scheme design. This finding emphasizes the importance of using an iterative approach to remove unexpected feature interactions that may hurt accuracy.

We also ran a user perception study to understand which factors influence perception of captcha difficulty. The study revealed that user sentiment towards captcha schemes is not aligned with real difficulty. In particular user perception is highly sensitive to the connotation and frequency of the content of the captcha.

REFERENCES

1. A. S. E. Ahmad, J. Yan, and M. Tayara. The robustness of google captchas. Technical report, New Castle, 2011.
2. H. S. Baird and T. P. Riopka. Scattertype: a reading captcha resistant to segmentation attack. In *Electronic Imaging 2005*, pages 197–207. International Society for Optics and Photonics, 2005.
3. M. Bernard, C. H. Liao, and M. Mills. The effects of font type and size on the legibility and reading time of online text by older adults. In *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*, pages 175–176, New York, NY, USA, 2001. ACM.
4. J. P. Bigham and A. C. Cavender. Evaluating existing audio captchas and an interface optimized for non-visual use. In *ACM Conference on Human Factors in Computing Systems*, 2009.
5. E. Bursztein, S. Bethard, J. C. Mitchell, D. Jurafsky, and C. Fabry. How good are humans at solving captchas? a large scale evaluation. In *Security and Privacy*, 2010.
6. E. Bursztein, M. Martin, and J. Mitchell. Text-based captcha strengths and weaknesses. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 125–138. ACM, 2011.
7. K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski. Computers beat humans at single character recognition in reading based human interaction proofs (hips). In *CEAS*, 2005.
8. K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski. Designing human friendly human interaction proofs (hips). In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 711–720. ACM, 2005.
9. C. Cruz-Perez, O. Starostenko, F. Uceda-Ponga, V. Alarcon-Aquino, and L. Reyes-Cabrera. Breaking recaptchas with unpredictable collapse: heuristic character segmentation and recognition. In *Pattern Recognition*, pages 155–165. Springer, 2012.
10. Google. Google consumer surveys. `http://www.google.com/insights/consumersurveys/home`.
11. P. S. K Chellapilla, K Larson and M. Czerwinski. Designing human friendly human interaction proofs. In ACM, editor, *CHI05*, 2005.
12. A. Kittur, E. H. Chi, and B. Suh. Crowdsourcing user studies with mechanical turk. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 453–456, New York, NY, USA, 2008. ACM.
13. K. A. Kluever and R. Zanibbi. Balancing usability and security in a video captcha. In *SOUPS '09: Proceedings of the 5th Symposium on Usable Privacy and Security*, pages 1–11, New York, NY, USA, 2009. ACM.
14. K. Larson, M. van Dantzich, M. Czerwinski, and G. Robertson. Text in 3d: some legibility results. In *CHI '00: CHI '00 extended abstracts on Human factors in computing systems*, pages 145–146, New York, NY, USA, 2000. ACM.
15. P. McDonald, M. Mohebbi, and B. Slatkin. Comparing google consumer surveys to existing probability and non-probability based internet surveys. Technical report, Google, 2012.
16. H. Motulsky and L. Ransnas. Fitting curves to data using nonlinear regression: a practical and nonmathematical review. *The FASEB journal*, 1(5):365–374, 1987.
17. T. Mustonen, M. Olkkonen, and J. Hakkinen. Examining mobile phone text legibility while walking. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1243–1246, New York, NY, USA, 2004. ACM.
18. M. Naor. Verification of a human in the loop or identification via the turing test. Available electronically: `http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps`, 1997.
19. J. Ross, L. Irani, M. Silberman, A. Zaldivar, and B. Tomlinson. Who are the crowdworkers?: shifting demographics in mechanical turk. In *CHI'10: 28th international conference on Human factors in computing systems*, pages 2863–2872. ACM, 2010.
20. T. Strutz. *Data Fitting and Uncertainty: A Practical Introduction to Weighted Least Squares and Beyond*. Vieweg and Teubner, 2010.
21. K. Thomas, D. McCoy, C. Grier, A. Kolcz, and V. Paxson. Trafficking Fraudulent Accounts: The Role of the Underground Market in Twitter Spam and Abuse. In *Proceedings of the USENIX Security Symposium*, August 2013.
22. M. Toomim, T. Kriplean, C. P "ortner, and J. Landay. Utility of human-computer interactions: toward a science of preference measurement. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, pages 2275–2284. ACM, 2011.
23. Y. Xu, G. Reynaga, S. Chiasson, J.-M. Frahm, F. Monrose, and P. van Oorschot. Security and usability challenges of moving-object captchas: Decoding codewords in motion. In *Usenix Security*, 2012.
24. J. Yan and A. S. E. Ahmad. A low-cost attack on a microsoft captcha. `http://bit.ly/nfpEis`, 2008.
25. J. Yan and A. S. El Ahmad. Usability of captchas or usability issues in captcha design. In *SOUPS '08: Proceedings of the 4th symposium on Usable privacy and security*, pages 44–52, New York, NY, USA, 2008. ACM.