μcaptcha: Human Interaction Proofs Tailored to Touch-Capable Devices via Math Handwriting

Luis A. Leiva[1,*] and Francisco Álvaro[2,†]

[1] PRHLT Research Center, Universitat Politècnica de València (UPV)
Camí de Vera, s/n – 46022 Valencia, Spain

[2] WIRIS math
Plaça de Gal·la Placídia, 1 — 08006 Barcelona, Spain

Abstract

Online services are often protected with captchas that typically must be solved by typing on a keyboard. Now that smartphones and tablets are increasingly being used to browse the web, new captchas best suited to touch-capable devices should be devised, since entering text on soft keyboards is usually uncomfortable an error-prone. We contribute to solving this issue with μcaptcha, a novel captcha scheme to tell humans and computers apart by means of math handwriting input. Instead of entering text with a keyboard, the user retypes a mathematical expression on a touchscreen using e.g. the finger, a stylus, or an e-pen. Further, as a byproduct of solving μcaptcha challenges, a valuable labeled dataset of online handwritten math expressions is collected. Our studies reveal that μcaptcha is accurate, fast, and easy to perform, and that users find it to be both useful and enjoyable. Ultimately, this work informs our understanding of designing better web security measures.

Keywords: CAPTCHA; HIP; Math Handwriting; Web Security Measures; Challenge Response Protocol; Touch-based Interaction

1 Introduction

A captcha (Completely Automated Public Turing Test to Tell Computers and Humans Apart) is a program that protects online services against bots by generating and grading challenges that humans can pass but computers cannot.

*Corresponding author. Email: `llt@acm.org`
†Work conducted while affiliated with the UPV.

Captcha belongs to the set of protocols called HIPs (Human Interactive Proofs), which allow a person to authenticate as belonging to a select group (Rusu and Govindaraju 2004); e.g., human as opposed to machine, adult as opposed to a child, etc. Captchas are used on the web for many purposes, such as to prevent massive email account creation, avoid spam comments in blogs and forums, or verify financial transactions. The main advantage of captchas is that they operate without the burden of passwords, biometrics, mechanical aids, or special training (Baird and Popat 2002). However, the usability of captchas is a subject of intense debate (Bursztein et al. 2014, Chellapilla et al. 2005, Yan and El Ahmad 2008).

Historically, users have had to deal with captchas in the form of images of distorted text, such distortions based on the weaknesses of Optical Character Recognition (OCR). However, as OCR software improves, solving captchas is becoming increasingly difficult. This places a burden on users (Bursztein et al. 2014), who are progressively reluctant to solve them (Shirali-Shahreza and Shirali-Shahreza 2006). Moreover, the majority of captchas are designed for use on computers and laptops, which do not align well with the interaction style of mobile users; see Figure 1.

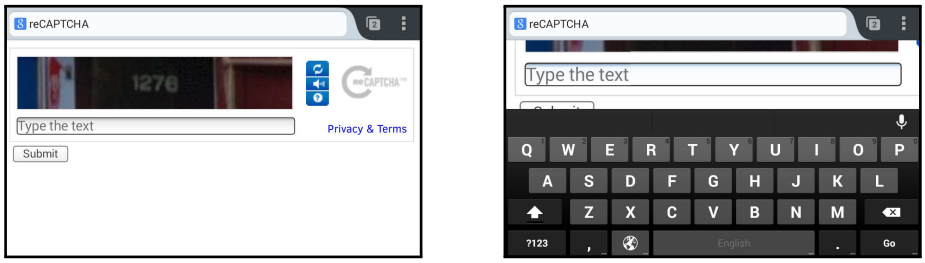

Figure 1: Solving captchas on a mobile device is rather uncomfortable. For instance, focusing on a text field causes zooming and field positioning which do not allow for the captcha to be read properly.

According to the international telecommunication union, there are more than 2 billion active mobile-broadband subscriptions worldwide.[1] An independent study by comScore[2] confirmed that in 2013 smartphones and tablets surpassed desktop PCs to become the leading platform in terms of total time spent online, either via web browsers or apps that make use of web services. These figures urge for a prompt revision on the design of HIPs in general and captchas in particular, since entering text in soft keyboards is uncomfortable and error-prone (Chen et al. 2010). To this end, drawing is presumably easier and quicker than typing on a mobile device (Kienzle and Hinckley 2013).

In this paper we introduce μcaptcha, a novel way to tell humans and computers apart by handwriting mathematical expressions on a touch-capable device. The main advantage of μcaptcha is that it is language-independent, so it is

[1] http://www.itu.int/ITU-D/ict/statistics

[2] http://www.comscore.com/mobilefutureinfocus2013

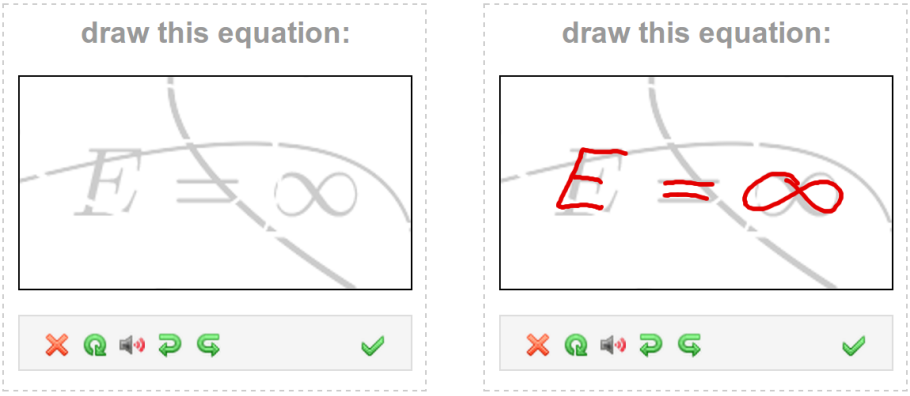

Figure 2: μcaptcha interface. A math expression is shown to the user (left), who has to draw it on a canvas (right). Buttons from left to right: clear strokes, request a new challenge, listen challenge (to write it in plain text), undo last stroke, redo last stroke, submit challenge.

equally easy to learn for everyone. Another potential advantage is that μcaptcha uses a controlled vocabulary of math symbols, which provides high recognition accuracy. Most important, the spatial relations between symbols (e.g., superscripts, subscripts, fractions, etc.) allow for a large number of different expressions to be generated. Furthermore, very few OCR software can recognize mathematical expressions. Together with its segmentation-resilient foreground noise technique (Figure 2), μcaptcha should keep regular attackers at bay. Finally, as a byproduct of solving a μcaptcha challenge, a labeled handwritten mathematical expression is obtained. Therefore, μcaptcha contributes to building valuable machine learning datasets. Ultimately, this work informs our understanding of designing better web security measures.

The contributions of this work include:

- Introduction of μcaptcha as a novel captcha scheme, together with a thorough description of the technology involved in its development.
- A web service that allows others to include μcaptcha on their websites or mobile apps, independently of the technology used to develop them.
- An open source web-based application that integrates our web service, which can be reused, studied, analyzed, or improved by both researchers and practitioners.
- Validation of μcaptcha in a series of experiments and in detailed analyses of reliability, accuracy, and user acceptance.

2 Related Work

Many captchas are known to be broken, so their general strength is an area of increasing concern. The next generation of captchas are likely to be more difficult and awkward for human users; e.g., Rusu et al. (2010) combined handwritten text images with a random tree structure and random test questions that leveraged unique features of human cognition. This approach was found to be hard to solve by machines, but also by regular users.

2.1 Text-based Captchas

There is a large body of work on text-based captchas that will not be discussed here because they do not bear direct relevance to this paper. The reader is redirected to comprehensive surveys in this area (Baird and Popat 2002, Basso and Bergadano 2010, Hidalgo and Alvarez 2011, Roshanbin and Miller 2013). In short, text-based captchas obfuscate OCR by introducing image degradations; the harder ones provide better performance but are also harder for humans to solve (Chew and Baird 2003, Coates et al. 2001). Among others, reCAPTCHA (von Ahn et al. 2008) stands out as the most popular solution. It makes positive use of human effort by channeling the time spent solving captchas into digitizing text, annotating street imaginery, or building machine learning datasets. It also can be broken with 99% of accuracy (Goodfellow et al. 2014).

2.2 Image-based Captchas

Since most text-based captchas are vulnerable, researchers have proposed alternatives to character recognition. A popular one is the form of image recognition, which requires users to identify simple objects in the images presented. However, the need for a human to label the pictures in a large database is mandatory, so that answers can be verified. Examples in this regard include: Sketcha (Ross et al. 2010), Implicit CAPTCHA (Baird and Bentley 2005), GOTCHA (Blocki et al. 2013), Asirra (Elson et al. 2009), or CAPTCHA-Zoo (Lin et al. 2011).

In visualcaptcha[3] the user has to click on the icon that best represents the word given. Currently, 37 icons and 20 audios are used as input stimuli. KittenAuth[4] also uses relatively small image databases. An image database small enough to be manually constructed is also small enough to be manually reconstructed by an attacker.

An interesting image-based captcha requires the user to rotate images to the correct orientation (Gossweiler et al. 2009). An advantage of this idea is that it uses unlabeled images. One drawback is that images have to be carefully selected since certain images can have several correct orientations. Also, if the images display faces, they can be automatically detected and rotated.

[3] `http://visualcaptcha.net`

[4] `http://thepcspy.com/kittenauth/`

2.3 Captcha Alternatives

The Math CAPTCHA (Hernandez-Castro and Ribagorda 2010) presents the user with an equation that must be solved. The number of different math challenges is very few and the answer for them is a single digit, so the challenge can be passed with trial and error. Further, this captcha is too complicated for the general public. There are other math-based captchas much simpler than this, relying on basic arithmetic operations shown in plain HTML, which can be automatically solved using regular expressions.

Another alternative is presenting the user with a short video, who can either describe it using tags (Kluever and Zanibbi 2009) or select the most appropriate answer from an option list (Shirali-Shahreza and Shirali-Shahreza 2008). Apart from their limited usability, these approaches do not scale well, as they require human intervention to enlarge the challenge database. In NuCaptcha[5] the user has to type moving letters, an approach that has been recently defeated (Xu et al. 2012).

Another option is to make the user play a game. PlayThru[6] provides different possibilities in this regard. Further, they claim to adopt a sophisticated mechanism to differentiate human game playing activity from automated activity (Mohamed et al. 2014). However, there is evidence that these games can be easily spoofed.[7]

2.4 Captchas for Mobile Devices

A number of research efforts are aimed to simplify and speed-up captcha solving on mobile devices. TapCHA (Jiang and Tian 2013) presents the user with a set of geometric shapes (square, triangle, etc.), and the challenge consists in dragging one of the shapes. Chow et al. (2008) propose to click on 3 valid English words out of a grid of 3x4 captchas. Some companies like Uniqpin[8] and ConfidentCAPTCHA[9] use this technique with images. Drawing CAPTCHA (Shirali-Shahreza and Shirali-Shahreza 2006) presents the user with a large number of squares randomly drawn, and the user must connect three diamonds via taps. QapTcha[10] is a draggable component for web forms; users just have to move a slider to confirm that they are humans. All of these approaches can be broken with machine learning techniques and client-side scripting (Chellapilla and Simard 2005, Lin et al. 2011).

SeeSay and HearSay (Shirali-Shahreza et al. 2013) allow the user to solve a captcha by submitting audio instead of text. There are, however, a number of situations where it is inappropriate to use speech-based input, and dictation difficulties are magnified when external conditions deteriorate (Price et al. 2009).

[5] http://nucaptcha.com
[6] http://areyouahuman.com
[7] http://spamtech.co.uk/software/bots/
[8] http://uniqpin.com
[9] http://confidentcaptcha.com
[10] http://myjqueryplugins.com/jquery-plugin/qaptcha

Moreover, recognition errors are one of the major concerns for users to accept these captchas (Shirali-Shahreza et al. 2013).

2.4.1 Handwriting Captchas

One option to ease text entry on mobile devices is by means of handwriting input. In Highlighting CAPTCHA (Shirali-Shahreza and Shirali-Shahreza 2011) the user must trace an obfuscated word with a stylus. This method is similar in spirit to ours; however it has two drawbacks. First, it requires the user to *precisely* trace each character, which is difficult to perform on a mobile device due to the inaccuracy of user input (Commarford 2004, Hourcade and Berkel 2008). Second, the handwriting recognition that validates user input relies on heuristics that do not achieve competitive accuracy.

Regular text handwriting could be further explored as a means to solve captchas on mobile devices. However, recognition of cursive hand-made text is language-dependent and challenging both for computers and humans (Rusu et al. 2010). Isolated handwritten character recognition could be used instead, but the number of symbols should be reduced in order to remove ambiguities (e.g., c, C, o, O, 0, etc.), resulting in few combinations available. Further, this is actually a subset of math symbols, which are language-independent and can be controlled together with the type of expressions shown to the user.

Another work closely related to ours is MotionCAPTCHA.[11] It presents the user with an image of a unistroke gesture that the user must draw. This is actually a proof of concept, and so it has a number of important security flaws. For example, the gesture vocabulary is very small (16 gestures, 4 bits of information entropy per challenge) and the solution to the captcha is available in the HTML source code. Nevertheless, we consider this approach worth of comparison, since today gestures are becoming ubiquitous on mobile devices (Leiva et al. 2014).

To conclude this section, Table 1 provides an overview of the relevant approaches to our work for the reader to have a good understanding of the technology landscape.

3 System Design

The specific implementation of μcaptcha reflects a number of design principles. We believe that understanding these will be useful to others trying to build similar systems. Ideally, desirable properties of captchas are the following (Hernandez-Castro and Ribagorda 2010):

- Automation: the challenge should be automatically generated and graded by a computer program.
- Easy of use: the challenge should be taken quickly and easily by human users.
- High reliability: the challenge should accept virtually all human users.

[11] `http://josscrowcroft.com/demos/motioncaptcha/`

Category	Name	Reference	Pros	Cons
General	reCAPTCHA	von Ahn et al. (2008)	supports different types of challenges	difficult to perform on mobile devices
Image	Rotating CAPTCHA	Gossweiler et al. (2009)	uses unlabeled data	images must be carefully selected
Math	Math CAPTCHA	Hernandez-Castro and Ribagorda (2010)	tailored to authenticating technical users	too complicated for regular users
Video	VideoCAPTCHA	Kluever and Zanibbi (2009)	large video dataset	limited usability
GamePlay	PlayThru	footnote 6	enjoyable to use	not accessible
Speech	SeeSay/HearSay	Shirali-Shahreza et al. (2013)	users enter audio instead of text	prone to recognition errors
Mobile	TapCHA	Jiang and Tian (2013)	drag and drop interaction	not scalable
Mobile	Drawing CAPTCHA	Shirali-Shahreza and Shirali-Shahreza (2006)	tap-based interaction	not accessible
Handwriting	Highlighting CAPTCHA	Shirali-Shahreza and Shirali-Shahreza (2011)	simple to use	difficult to perform on mobile devices
Handwriting	MotionCAPTCHA	footnote 11	works well on touchscreens	very insecure
Handwriting	μcaptcha	this work	easy to use and learn	uncomfortable for mouse interaction

Table 1: Summary of captcha schemes relevant to this article, including their main strengths and weaknesses.

- Low false negatives: the challenge should reject very few human users.
- Low false positives: the challenge should reject virtually all machine users.
- High reliability: the challenge should resist automatic attack for many years, even as technology advances.

However, no matter the challenge, the whole system must be secure. Concretely, μcaptcha complies with the following requirements:

- Protection against brute-force attacks. It must be resilient to requiring another captcha after a few invalid responses.
- Protection against replay attacks (Mohamed et al. 2014). A captcha ID generated once cannot be reused, or at least should not be valid in other active session.
- Protection against side-channel attacks (Zelkowitz 2001). The captcha solution must not be stored on the client side. Otherwise, it would be trivial to bypass the challenge.

- Server-side validation. The captcha system must validate what is submitted by the user.

3.1 Architecture Overview

Figure 3 provides an overview of our system architecture. At a high level, the user draws a mathematical expression, generating thus a sequence of handwritten strokes that are submitted to a math recognizer. For a challenge to be passed, the ID of the recognition result must match the ID of the mathematical expression presented to the user.

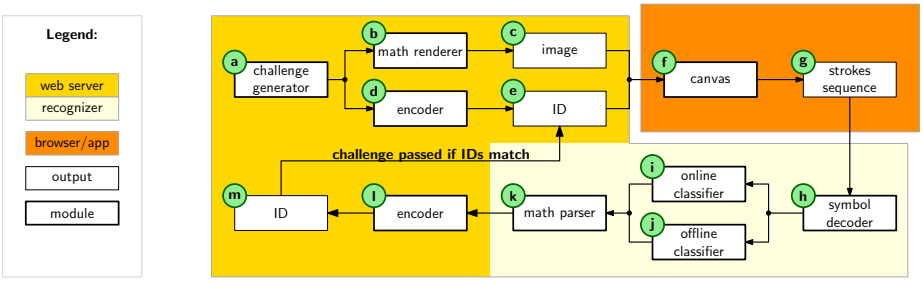

Figure 3: Requesting a μcaptcha generates a math expression in TEX format (a). The expression is encoded (d, e) and rendered as an image (b, c). The user must draw (f) the presented math expression, generating thus a sequence of strokes (g) that is submitted to a symbol decoder (h). The identified handwritten symbols are submitted to two classifiers (i, j) for later recognition within a math parser (k). The output of the parser is a math expression in TEX format, which is encoded (l) in the same way as (d). The challenge is solved if the final ID (m) matches the challenge ID (e).

A μcaptcha ID is a one-way "salted" hash of a TEX equation generated by a probabilistic context-free grammar (PCFG). The server-side encoder knows how to de-salt the hash, which depends on the session and a number of additional factors such as the website requesting the challenge or the checksum of the non-distorted challenge image. Therefore this ID scheme not only makes the backend stateless, but also makes it impossible to invert the hash. So, even if a malicious attacker could mimic our encoder algorithm, these measures would deter a potential attack.

3.1.1 Math Recognizer

μcaptcha is built upon SESHAT,[12] a state-of-the-art open source recognizer of handwritten math expressions. The recognizer operates in a two-step procedure. First, isolated symbol recognition is performed using both online and offline recurrent neural network classifiers. Second, the most likely math expression is

[12] `http://github.com/falvaro/seshat`

parsed according to a 2D-PCFG equivalent to the grammar we use to generate the challenges.

3.1.2 Challenge Generation

Figure 4 shows some μcaptcha examples. Special consideration has been put into making our challenges reasonable for humans to solve. On the one hand, successful captchas rely on segmentation problems, as they are computationally expensive (Simard et al. 2003). μcaptcha images use black and white foreground arcs, since these are easily recognized for humans and yet remain difficult for computers to distinguish (Chellapilla et al. 2005). On the other hand, PCFGs are defined at the symbol level (Álvaro et al. 2014, Zanibbi and Blostein 2012). However, because in μcaptcha the user has to enter strokes, our PCFG is defined at the stroke level. Thus, a "longer" challenge means that the user has to write more strokes, not necessarily that the expression has more symbols. For instance, $\boxed{2-3}$ (3 symbols, 3 strokes) is much faster to write than $\boxed{F \neq \pi}$ (3 symbols, 9 strokes).

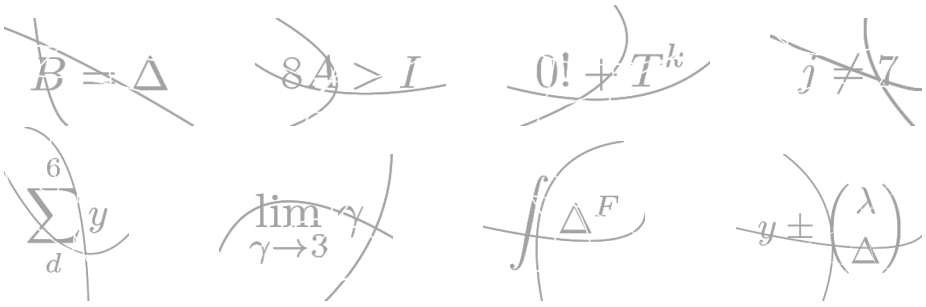

Figure 4: Examples using between 4 and 9 strokes per math expression.

3.1.3 Math Vocabulary Training

There are 3 different sources of error in μcaptcha: (1) the challenge is too hard or unclear, which may be skipped; (2) the user draws an incorrect symbol; (3) the math recognizer does not accept the answer. The latter is worth of consideration, as it introduces some degree of indeterminism due to recognizing hand-made input. Therefore, it is important for the math recognizer to reach high accuracy results.

Initially, we considered the full set of 100 symbols used in CROHME,[13] a popular competition on recognition of online handwritten mathematical expressions. The number of samples provided for each class is quite unbalanced, therefore we manually labeled new symbols until we obtained 500 samples per class. Then, for each symbol class we used 400 samples for training, 50 samples for validation, and 50 samples to form the test set. As a result, we trained the online and offline

[13] http://www.isical.ac.in/~crohme/

classifiers with 40,400 symbols. The error rate of isolated symbol recognition was 6.97%. This was not sufficient for μcaptcha, so we removed the symbols that caused most of the errors according to the classifiers' confusion matrices. After this, the error rate was as low as 0.88% considering 66 symbols. Figure 5 shows the symbol sets before and after removal of conflicting classes.

0	1	2	3	4	5	6	7	8	9
A	a	α	b	B	β	C	c	cos	$+$
d	$/$	Δ	$\div$	$\cdot$	$\ldots$	E	e	$=$	$\exists$
F	f	$\forall$	G	g	γ	$\geq$	$>$	h	H
i	I	$\in$	∞	$\int$	j	k	l	L	λ
$\{$	$[$	$\leq$	lim	log	$($	$<$	m	M	μ
n	N	$\neq$	o	P	p	ϕ	π	$\pm$	$\prime$
q	R	r	$]$	$\}$	$\rightarrow$	$)$	S	s	σ
sin	$\sqrt{}$	$\sum$	T	t	tan	θ	$\times$	u	V
v	w	X	x	y	Y	z	$-$	!	$\vert$

Figure 5: Original set (100 symbols) and reduced set (66 symbols), by removing those symbols indicated in gray background color.

Reducing the set of math symbols resulted in an isolated symbol classification error of near zero, but the structure of the math expressions has to be recognized as well. Thus, aiming at a better system, we limited our PCFG so that structural ambiguities were removed. For instance, we decided that a letter followed by a number can only be subscript or superscript. With these improvements, the error of our math recognition system should be very low, while providing a large number of math expressions. Concretely, with a vocabulary of m math symbols and r spatial relations, there are approximately $m^N r^{N-1}$ math expressions of N symbols. μcaptcha uses 66 symbols and 5 spatial relations, so a challenge with 3 symbols has about 7M of possible combinations, which represents 22.7 bits of information entropy. This is actually an upper bound, since not all symbols can use all spatial relations. Taking into account that μcaptcha generates expressions of 6 symbols on average, there are approximately $7.1 \cdot 10^{54}$ combinations (182.2 bits of information entropy). By way of comparison, a 4-digit PIN has 10^4 combinations and 13.2 bits of information entropy.

3.2 Web Service

μcaptcha provides a JSON-based REST API to become backend-agnostic. This way, developers can use their preferred technology stack to deploy μcaptcha on websites or native mobile apps. The API provides two endpoints: one for

requesting a challenge and other for solving the challenge (Figure 6). Both API endpoints require a registered user token to be submitted on each request. Otherwise the server responds with a "403 Forbidden" HTTP status code, which indicates that the server can be reached and understood the request, but refuses to take any further action. The API is available at `http://api.mucaptcha.com`.

When a challenge is requested, the developer can point to PNG and MP3 files by concatenating the response ID with `.png` or `.mp3` extensions. To solve the challenge, the user must submit a sequence of online strokes in the following format:

```
[
  [ [x_1, y_1, t_1], ..., [x_N, y_N, t_N] ],    // First stroke
  ...,
  [ [x_1, y_1, t_1], ..., [x_M, y_M, t_M] ]    // Last stroke
]
```

where x and y are coordinates and t is their timestamp.

A `status` code informs the application interfacing with our web service whether the result was successfully processed (0: no error) or not (code > 0 otherwise). A challenge is passed when the value of `msg` equals `"success"`. All passed challenges are periodically fed back to our math recognizer so that it can learn different writing styles and improve accuracy over time.

Request	`GET http://url/user_token/challenge`
Response	`data:{ "status":0, "id":"http://url/hash" }`

Request	`POST http://url/user_token/solve/id` `data:[ strokes array ]`
Response	`data:{ "status":0, "msg":"success" }`

Figure 6: REST API. We have developed an accompanying web-based prototype that interfaces with our web service.

3.3 Web Application

We have developed a web-based prototype that interfaces with our web service. The application is released as open source software so that others can reuse, study, analyze, or improve it. In order to save valuable touchscreen space, we decided to put the challenge as a background image in a web canvas, as shown in Figure 2. However, a developer implementing other μcaptcha interfaces may decide to present the user with the image and the canvas separately (Figure 7), as the user is strictly not required to trace the symbols of the math expression. It is required, however, to preserve the spatial relations between symbols; i.e., an expression with highly overlapping symbols might not be recognized reliably. Our prototype is available at `http://mucaptcha.com`.

Similar to other captchas, to account for challenges that are unreadable or too difficult to write, our application allows users to request a new challenge.

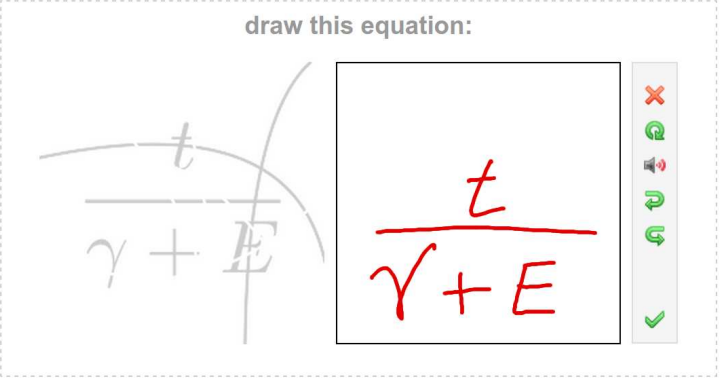

Figure 7: Alternate UI design. Eventually we opted for the compact version (Figure 2) in order to save screen space.

Also, when a μcaptcha is wrongly solved, the user is presented with a different challenge; there is no option to retry the same μcaptcha once submitted. Further, in order to provide an accessible option for all users, our application features the possibility of hearing the mathematical expression and letting the users to write it down in plain text. In this case, the web canvas is replaced by a regular text field. Under the hood, the audio synthesis is performed with Festival,[14] an open source speech synthesis system. English and Spanish voices are currently available, as they are shipped with the latest Festival version.

4 Evaluation

We performed 3 studies in order to investigate the possibilities of μcaptcha. The first one analyzed the strengths of μcaptcha against math-based OCR. The second one was a longitudinal user study, aimed at fine-tuning our recognizer for real-world use. The third one was an in-lab user study, aimed at comparing μcaptcha against two alternatives.

4.1 Attacking μcaptcha

We simulated a fundamental attack consisting in scanning μcaptcha challenges offline with math OCR software. We generated thousands of challenges and tried to automatically solve them with InftyReader,[15] the most competitive math-based OCR to date.

[14] `http://festvox.org`
[15] `http://www.inftyproject.org`

4.1.1 Results and Discussion

μcaptcha images are rendered in lightgray color, which caused InftyReader to misrecognize all expressions. Images were thus binarized and submitted again to InftyReader, but it could not recognize any expression because of the foreground noise. Therefore, we sought a more sophisticated attack.

Aimed at removing the foreground noise while preserving the math symbols, we applied erosion and dilation operations with different operator sizes. Figure 8 shows some examples of the images resulting after these morphological operations. Even so, none of these attempts allowed InftyReader to properly recognize any expression because the lines and arcs used as noise have similar width to that of the math symbols, thus either the noise was not successfully removed or parts of the symbols disappeared afterward.

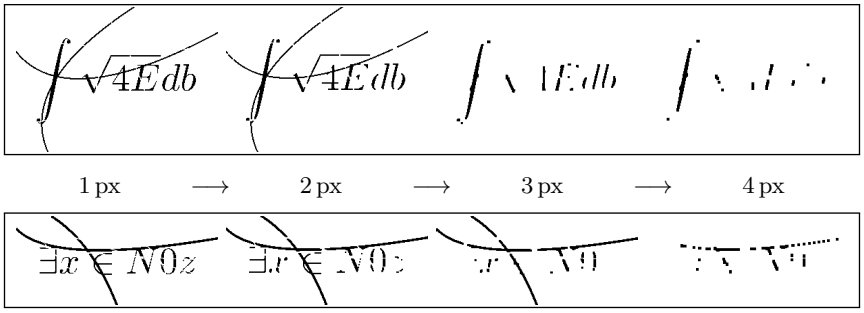

Figure 8: Attacking μcaptcha. Noise reduction examples use morphological operators (erosion plus dilation) of variable size.

We conclude that μcaptcha cannot be broken with out-of-the-box math OCR or basic image preprocessing techniques. This is because offline recognition relies on connected components analysis, which in our case is obfuscated by using foreground noise as lines and arcs. Even in the case that noise could be removed with a sophisticated preprocessing technique, at least two further issues should be addressed to break a μcaptcha. First, the denoised image should be scanned with a competent math OCR. Then, it still should be required to generate the online sequence of strokes that represents the math expression. As a two-dimensional problem, not only must the symbols be correct, but also properly arranged.

4.2 Longitudinal Pilot Study

We were interested in examining the actual performance of our recognizer, since the error rate achieved in training was at the symbol level. As we had constrained the set of math symbols and the grammar, we could not assess recognition rate at the expression level with the CROHME test set; roughly just 3% of the provided expressions could be accounted for by our system. Furthermore, we were interested in examining the users' performance as well as their acceptance toward μcaptcha. For these reasons, we decided to use a Wizard of Oz (WoZ)

system, which informed users only that they advanced to the next challenge. The study was conducted during 5 working days.

4.2.1 Participants

We advertised the study via local mailing lists and bulletin boards to get a sample of participants with diverse backgrounds, preferably not computer experts. Eventually we recruited 10 right-handed participants (4 females) aged 25–34 (M=28.3, SD=6.1) that were either students or university staff members with no technical background in computer security or captcha design. Participants were gastronomically compensated for their efforts the week after the study.

4.2.2 Design

The experiment was a within-subjects repeated measures design. Error rate and solving time were set as dependent variables. The collected data were analyzed on a daily basis.

4.2.3 Apparatus

Participants were given an iPad mini to solve μcaptchas within our web application. Using the same apparatus for all participants eliminates the effect of device and the effect of screen size. The math recognizer was tuned each day, using the accumulated data provided by the participants. Retraining the symbol classifiers was unfeasible, since each recurrent neural network usually takes up to 3 days. All challenges randomly ranged between 3 and 9 strokes per expression, which is approximately 2–9 symbols per expression. The web canvas was 300 px wide.

4.2.4 Procedure

Participants were briefly described the μcaptcha interface. Once they were ready, participants were given 10 minutes to solve as many challenges as they could. Participants were instructed to perform as fast and accurate as possible, so they operate at their own speed-accuracy tradeoff point. These instructions were repeated each day. Being a WoZ setting, participants were not told whether they successfully solved the captchas or not as they went.

4.2.5 Results

Participants solved 2,780 μcaptcha challenges, at a rate of 556 challenges (SD=58.1) per day, which amounts to M=55.6 daily challenges on average per participant.

Analysis of Recognition Accuracy μcaptcha achieved a mean error rate of 8%; see Figure 9. We believe this is good enough for production use: approximately 1 out of 15 "legitimate" challenge solutions would be rejected. Moreover, it is worth noting that these errors may be either because the user draws an

incorrect symbol or because the math recognizer does not accept the answer. A deeper examination of the challenges that were wrongly solved revealed that approximately 20% of the time the error was caused by sloppy handwriting, the remaining errors being just due to system errors. Considering that no participant had tried μcaptcha before, we find these results to be particularly promising. We shed more light in this regard in a later section.

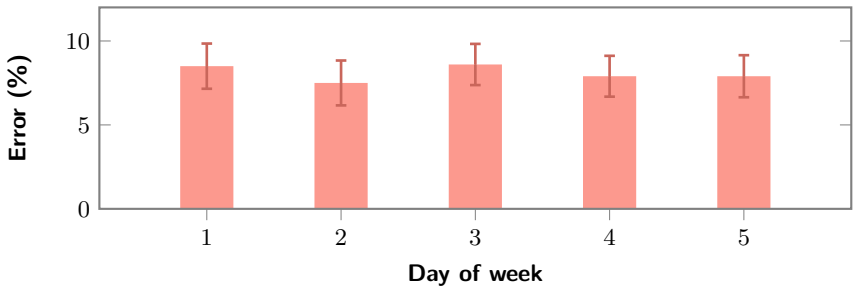

Figure 9: Error rate. Error bars denote 95% confidence intervals.

Analysis of Solving Time Overall, participants took around 6 seconds to solve each challenge (M=6.1, SD=3.6); see Figure 10. Compared to other captcha systems, we observed that μcaptcha is fast enough; e.g., solving a regular captcha takes between 5 and 10 seconds to complete (Bursztein et al. 2014, Lin et al. 2011), whereas reCAPTCHA takes 13 seconds on average (von Ahn et al. 2008). In the second user study we shed more light in this regard.

Qualitative Observations Arguably more important than performance-related results, participants found μcaptcha to be useful and particularly engaging. They stated that they do not see themselves using it with a regular computer mouse, but put in the context of mobile devices is perceived as a smart approach. Indeed, the mouse is less dexterous than the finger or an e-pen. Interestingly, some participants made the observation that *"some symbols were harder to enter because they require more writing strokes"*; and wished that expressions were slightly shorter overall. These statements encouraged us to push μcaptcha's development forward. For instance, as a result, we decided to set the upper bound of 9 strokes per expression to be at most 6 strokes.

4.3 In-Lab User Study

This study was aimed at comparing μcaptcha with reCAPTCHA (currently the most popular captcha system) and MotionCAPTCHA (a similar alternative to ours, from a user interaction's perspective). Our hypothesis was that drawing is better than typing on a mobile device in several ways.

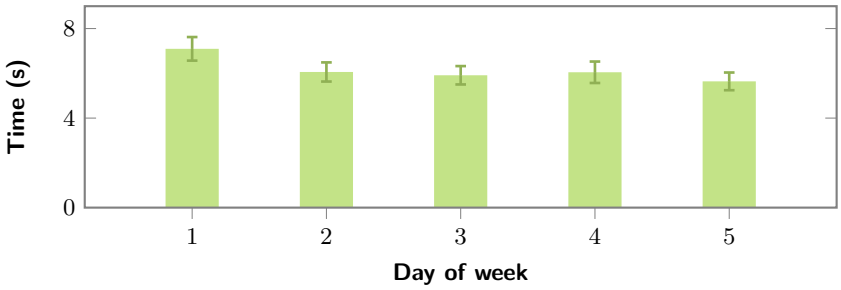

Figure 10: Solving time. Error bars denote 95% confidence intervals.

4.3.1 Participants

Again, we advertised the study widely to get another sample of participants with diverse backgrounds. We required participants to have no prior exposure to the previous study. Eventually we recruited 10 participants (3 females) aged 25–35 (M=30.4, SD=3.4). Two of them were left-handed. All participants had submitted a text-based captcha at some time while browsing the web, though nobody had technical background in computer security or captcha design. Each participant was given a gift voucher of $10 at the end of the study.

4.3.2 Design

The experiment was a within-subjects repeated measures design with 3 conditions: reCAPTCHA (Figure 1), MotionCAPTCHA (Figure 11), and μcaptcha. We measured error rates and 3 time-related measures: processing time (time between showing up the captcha and start entering the solution), execution time (time spent solving the captcha), and overall solving time (until clicking on the submit button).

4.3.3 Apparatus

All challenges were solved on a Nexus 4 smartphone (Android 4.4.4) with the Chrome mobile browser 36. We used the same web application as in the previous study with the same configuration. Now μcaptcha challenges were set to randomly range between 3 and 6 strokes per expression, which is approximately 2–6 symbols per expression. In addition, some adjustments were made to the other captcha systems:

- reCAPTCHA currently displays Google street view images by default. Then, after solving exactly 5 challenges they become the usual two words (a verification word, to which reCAPTCHA knows the answer, and an unknown word which comes from an old book). We ensured that the two-word version was always shown to the participants, so that everyone tested it under the same settings.

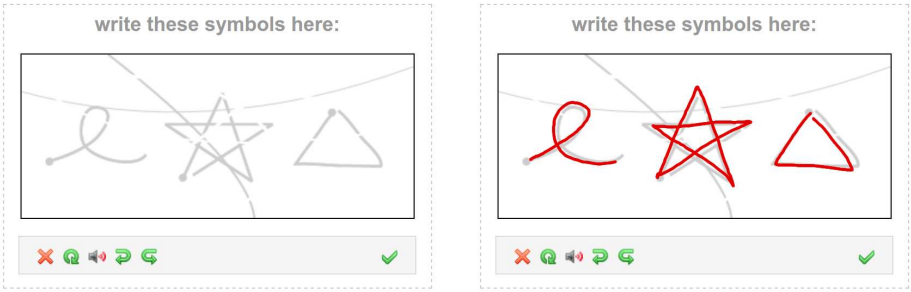

Figure 11: MotionCAPTCHA example, using our UI's look and feel.

- MotionCAPTCHA is an insecure option for many reasons. However, it is similar to μcaptcha in terms of interaction, c.f. drawing instead of typing. Therefore, we instrumented MotionCAPTCHA to reach the level of its peers. The prototype uses the same interface as μcaptcha, the same foreground noise technique, and a similar hashing function to encode the challenges. In addition, 3 unistroke gestures are always shown to the user, which increases the challenge combinations from 16 to 16^3. The server-side recognizer is a 1 nearest-neighbor classifier with Euclidean distance (Wobbrock et al. 2007).

4.3.4 Procedure

Participants were briefly introduced the captcha systems they would use and could test each for at most 5 minutes. They had to hold the smartphone in portrait position and were told to go at their normal working pace. They had to solve 25 challenges with each system; resulting thus in 250 observations per condition. This time participants were informed whether they successfully solved a challenge or not as they went, as it is a better simulation of real-world use cases. All systems automatically advanced to the next challenge, regardless it was correctly solved or not. Conditions were counterbalanced and presented to the participants in randomized order, in order to reduce the chance of learning effects.

In order to get a better picture of the study, participants filled in the SUS and NASA-TLX questionnaires on a nearby computer. They were also asked to score each captcha system in a 5-point Likert scale in terms of: usefulness, ease of execution, ease of understanding, and ease of learning.

4.3.5 Results

Analysis of Recognition Accuracy As shown in Table 2, μcaptcha was found to be more accurate than reCAPTCHA and MotionCAPTCHA. A Kruskal-Wallis one-way analysis of variance test was statistically significant $[\chi^2_{(2,N=30)} = 18.49, p < .001, \eta^2 = 0.68]$. Post-hoc pairwise comparisons using

the Wilcoxon signed rank test (Bonferroni corrected) indicated no significant difference between reCAPTCHA and μcaptcha, whereas MotionCAPTCHA performed significantly worse than its peers. We conclude that μcaptcha is accurate enough for production use. The bad performance of MotionCAPTCHA is explained by the fact that a single challenge entails 3 unistroke gesture recognition tests, so if just one of these tests is unsuccessful, the challenge is not passed.

System	Accuracy (%)	95% CI*
reCAPTCHA	88.4	[83.84 – 91.80]
MotionCAPTCHA	65.6	[59.52 – 71.21]
μcaptcha	90.8	[86.57 – 93.79]

* Wilson interval estimation for binomial distributions.

Table 2: Recognition accuracy results.

Analysis of Solving Time Overall, participants spent considerably more time solving a reCAPTCHA in comparison to MotionCAPTCHA and μcaptcha; see Table 3. Differences between systems were statistically significant according to the ANOVA test [$F_{2,27} = 112.16, p < .001, \eta_p^2 = 0.89$]. Effect size suggests a high practical significance. Post-hoc pairwise comparisons using the t-test (Bonferroni corrected) revealed that reCAPTCHA was significantly slower than its peers. μcaptcha was significantly 1 second slower than MotionCAPTCHA, however this was unsurprising because μcaptcha challenges had almost twice the number of strokes (M=5.4, SD=1.6).

Time (s)	reCAPTCHA	MotionCAPTCHA	μcaptcha
Processing	5.0 (2.9)	0.9 (0.4)	1.3 (0.7)
Execution	10.2 (3.6)	4.4 (1.2)	5.1 (1.9)
Overall	15.2 (4.5)	5.3 (1.5)	6.5 (2.1)

Table 3: Mean solving time. SDs are denoted in parentheses.

Similarly, execution time results showed that participants spent considerably more time typing a reCAPTCHA than drawing a MotionCAPTCHA or a μcaptcha. Differences between systems were statistically significant [$F_{2,27} = 80.54, p < .001, \eta_p^2 = 0.86$]. Effect size suggests a high practical significance. Post-hoc pairwise comparisons (Bonferroni corrected) revealed that participants spent a significantly high amount of time typing as compared to drawing. As expected, μcaptcha was significantly slower than MotionCAPTCHA, since μcaptcha challenges require entering more strokes.

We observed that reCAPTCHA takes up to one third of the total time to process the challenge, i.e., the user first reads the challenge and then focuses on the text field to start typing. In contrast, participants were considerably faster with the other systems. Differences between systems were statistically

significant $[F_{2,27} = 36.77, p < .001, \eta_p^2 = 0.73]$. Effect size suggests a high practical significance. Post-hoc pairwise comparisons (Bonferroni corrected) revealed that participants were significantly slower at processing reCAPTCHA challenges. MotionCAPTCHA and μcaptcha performed statistically similar, which suggests that our math challenges are both easy to read and understand.

Analysis of Usability and Workload In terms of SUS, reCAPTCHA scored lower than its peers (higher is better); see Table 4. Differences between systems were statistically significant $[F_{2,27} = 19.08, p < .001, \eta_p^2 = 0.59]$. Effect size suggests a moderately high practical significance. Post-hoc comparisons (Bonferroni corrected) revealed that reCAPTCHA was found to be significantly less usable than the other systems. MotionCAPTCHA and μcaptcha performed statistically similar. It is worth noting that SUS scores below 50 imply serious usability issues (Bangor et al. 2008).

Score	reCAPTCHA	MotionCAPTCHA	μcaptcha
SUS	49.7 (16.1)	77.0 (11.5)	82.0 (8.8)
TLX	61.1 (19.1)	40.5 (13.6)	39.6 (12.2)

Table 4: Mean usability (SUS) and workload (TLX) scores, both $\in [0, 100]$. SDs are denoted in parentheses. Higher SUS is better, lower TLX is better.

In terms of TLX, reCAPTCHA scored higher than its peers (lower is better); see Table 4. Differences between systems were statistically significant $[F_{2,27} = 6.31, p < 0.01, \eta_p^2 = 0.32]$. Effect size suggests a moderate practical significance. Post-hoc comparisons (Bonferroni corrected) revealed that reCAPTCHA was found to be more cognitively demanding than the other systems. MotionCAPTCHA and μcaptcha performed statistically similar. It is worth noting that TLX scores above 60 are cause for concern (McCabe 2002).

Analysis of Perceived Utility Our participants found reCAPTCHA to be less likable than the other systems, though they acknowledged that reCAPTCHA pursues the considerate goal of digitizing old books. This explains why it scored higher in terms of usefulness but was around a neutral score for the rest of the assessed statements; see Table 5. Overall, differences between systems were statistically significant according to the ANOVA test $[F_{2,27} = 17.10, p < .001, \eta_p^2 = 0.56]$. Effect size suggests a high practical significance. Post-hoc pairwise comparisons using the t-test (Bonferroni corrected) revealed that reCAPTCHA was perceived to be significantly less valuable than its peers. It was interesting to note that MotionCAPTCHA, despite its low accuracy, was rewarded by the participants as being statistically similar to μcaptcha. This suggests that mobile users are eager to try captchas that are best suited to mobile devices. Also interestingly, participants called into question the utility of MotionCAPTCHA.

Perception	reCAPTCHA	MotionCAPTCHA	μcaptcha
Usefulness	4.0 (0.8)	3.8 (0.7)	4.5 (0.5)
Execution	2.9 (1.3)	4.0 (0.8)	4.4 (0.5)
Understandability	3.2 (1.3)	4.2 (0.7)	4.3 (0.5)
Learnability	3.2 (1.2)	4.3 (0.7)	4.3 (0.5)
Overall	3.2 (1.2)	4.2 (0.7)	4.3 (0.5)

Table 5: Mean subjectivity scores (higher is better) $\in [1, 5]$. SDs are denoted in parentheses.

Qualitative Observations Participants were concerned about the fact that unistroke gestures must be performed in a unique way: *"even though it is very easy to reproduce the gestures, the recognizer fails too often."* In contrast, in μcaptcha it is possible to write math symbols in different ways, which provides the user with more flexibility. In this regard, one participant stated: *"I'm not very good at writing... I'm surprised how accurate is the math recognizer!"* Also, we observed that users tend to read the entire math expression prior to start writing, whereas for MotionCAPTCHA the user can read and enter one gesture at a time. Finally, other participant made an interesting observation: *"Initially I considered μcaptcha to be more complex than MotionCAPTCHA, but then I realized that you are just drawing what you see in the background."*

Labeled Dataset As a result of the challenges solved by our participants, 2,787 online handwritten math expressions have been automatically annotated. The math recognition system outputs a TEX transcription together with an InkML[16] file describing the recognition result. Therefore, when a challenge is solved we obtain a valid sample that is annotated at the stroke level with symbol segmentations, symbol classes, and the structure of the expression. This is a valuable asset helpful for machine learning in math expression recognition, and avoids the tedious manual annotation process. μcaptcha is thus channeling the user effort into building valuable machine learning datasets. For instance, the data can be useful for training handwritten character recognition systems, for which the annotation process is usually tedious and time-consuming.

4.4 Follow-up Study: Improving Recognition Accuracy

Overall, the accuracy rate of μcaptcha is competitive enough for production use, however we wondered if it could be further improved. One way to do so is by improving the math recognizer, i.e., either the symbol classifiers or the PCFG parser. Nevertheless, this is a non-trivial task that requires expert knowledge in pattern recognition or machine learning procedures. A more plausible option consists in introducing a confidence measure, so that the outcome of the recognizer is not as hard as a true/false binary decision. In fact, reCAPTCHA intentionally

[16] `http://www.w3.org/TR/InkML/`

tolerates some errors depending on how much they trust the user giving the solution. This is in line with a question raised by one of our participants: *"most of the time I did not understand why reCAPTCHA accepted my answer because I had to do a very big guess on one of the words."* Eventually we adopted the following solution.

For mathematical expressions, EMERS (Sain et al. 2011) is a well-defined tree edit distance evaluation measure. EMERS is not a normalized distance, but it calculates the set of edit operations to transform a tree into another such that if both trees are identical EMERS is equal to zero. EMERS is motivated by our desire to test the value of handwriting input (while realistically acknowledging that perfect recognition may in fact be unattainable) to determine whether it is worthwhile even to pursue better math recognizers for this task. Figure 12 shows the gain achieved in accuracy incurred by different EMERS thresholds.

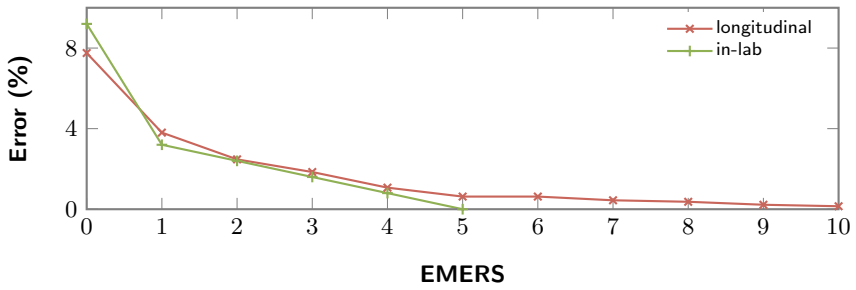

Figure 12: Improving accuracy by using different EMERS thresholds.

As observed, EMERS-1 (one tree-edit distance) leads to 50% of improvement in current recognition rates (Figure 12). However, this comes at the cost of increasing the chance of false positives. Thus, it is important to seek a balance between using a binary decision as recognition result (hard constraint) or being a bit fault-tolerant in order to improve the user's acceptance. Together with the distributions shown in Figure 13, we have decided to use EMERS-1 in case the recognition result does not match the challenge submitted to the user. This should increase the user experience without impacting security. Regarding the automatic annotation of math expressions, only those expressions that have been perfectly recognized (EMERS-0) are actually saved as ground truth data.

5 General Discussion

What fundamental aspect of mathematical expressions makes them suitable as captchas? For instance, could a simpler, non-math-based scheme like letters-only but in normal, superscript, and subscript positions work just as well? Letters-only is actually a subset of math expressions. Therefore, math expressions allow for more combinations. Furthermore, letters-only with subscripts/superscripts is a one-dimensional (left-to-right) problem that would be easier to crack than

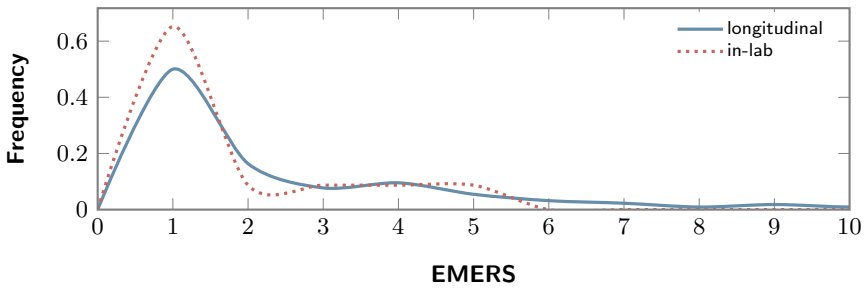

Figure 13: Relative frequency of EMERS error distributions.

two-dimensional expressions. Even more, detecting subscripts/superscripts from regular text is error-prone (Álvaro et al. 2014) and usually relies on language models (Zanibbi and Blostein 2012). We constrained our grammar in order to solve spatial ambiguities and improve accuracy.

Bursztein et al. (2014) analyzed the interactions of visual features used in today's captchas, in order to understand how they affect captcha's difficulty and user perception. Unexpectedly, it was found that accuracy and solving time are not good predictors of user preference. Instead, it is more effective to use the captcha as a medium for engagement with the user, and examine the interaction holistically. This explains why our participants scored MotionCAPTCHA and μcaptcha similarly; see Table 4 and Table 5. However, our evaluation tasks were artificial and therefore we suspect that in a real-world situation this perception toward MotionCAPTCHA might change due to its relatively low accuracy.

HIPs, and by extension captchas, are based on open problems in artificial intelligence (AI) that induce security researchers, as well as otherwise malicious programmers, to work on advancing the field of AI. HIPs are thus a win-win situation (von Ahn et al. 2004): either it is not broken and there is a way to differentiate humans from computers, or it is broken and an AI problem is solved. While μcaptcha does not guarantee that is secure against highly sophisticated attacks, it is not trivially breakable. However, like every captcha system, μcaptcha could be defeated by human manpower. For example, spammers could pay a developer to aggregate our challenges and feed them one by one to a human operator.

5.1 Limitations and Future Work

Our current implementation of μcaptcha has a number of limitations that are intended to be addressed in future work:

- It would be desirable to incorporate more math symbols, in order to further increase μcaptcha's information entropy. However, our preliminary studies suggest that doing so may increase the error rate of the recognizer. A plausible option would be balancing EMERS and expression length; e.g.

EMERS-2 would not be tolerable for use with expressions comprising 3 symbols, but it may be adequate for use with expressions having 6 symbols.

- The lines and arcs of the foreground noise are drawn in an uncontrolled fashion. It may happen that they occlude parts of the expression, such as a minus sign, which could lead the user to enter a wrong solution to the challenge presented. In fact, one of our participants complained about this issue. In any case, if the user cannot distinguish what symbols are being shown, it is possible to reject the challenge by requesting another one. We log all unsolved challenges for later analysis.
- It may be the case that a mathematical expression does not make sense; e.g. $\boxed{2x\ \forall y}$. This may confuse users with advanced math knowledge, though we have not received such a complain. We believe that a mechanism to generate "mathematically correct" expressions would be desirable.

On another line, we would like to explore more sophisticated security attacks. One of these could be that of template attack (Yan and Ahmad 2008), where pre-computed templates of math symbols would be used to try breaking μcaptcha. However, because the foreground noise makes it hard to isolate symbols, a template attack would probably detect wrong symbols. Further, we could use different math renderers, as TEX equations may have different font styles such as face, size or weight. Even if such attack were successful, handwritten strokes should be artificially generated and arranged according to both the structure and the relations of the math expression. Another possibility would be creating online strokes for the connected components of a math image. Nevertheless, this is really challenging. The foreground noise has different thick sizes so there are no expected wider or higher strokes. Therefore, removing components would either delete symbol parts or add noisy strokes.

Taking a different tack, it is worth noting that our web service can be integrated in native mobile apps; however we expect μcaptcha to be largely used on websites. This was the main reason why we developed a web-based application to interface with our web service. In this context, we should mention that the application is tailored to HTML5 browsers. Therefore, drawing on a web canvas might not work in old devices or old browsers. Also, drawing does not work at all in browsers that have disabled JavaScript.

Lastly, if μcaptcha were largely used, it would create a vast dataset of labeled online handwritten math expressions. This resource would help the advancement in math research, which is lately bringing more attention to students as a result of the rapid growth of mobile devices. For instance, education is moving toward computer-based solutions at the expense of traditional (paper-based) books, so handwritten input is likely to play an important role in the classrooms in the near future. Another increasingly important field is that of math information retrieval (Zanibbi and Blostein 2012), for which μcaptcha would contribute in creating worthwhile educational materials on math.

6 Conclusion

We have introduced μcaptcha, a novel method to tell humans and computers apart by handwriting mathematical expressions on a touch-capable device. Our studies suggest that μcaptcha is appreciated by the users, however we believe the results presented here are part of a more general idea. It is important to have captchas tailored to different platforms and devices, since there is a large number of them accessing the web today. μcaptcha is especially useful on mobile devices where soft keyboards are difficult to accommodate or use. Furthermore, the effort spent solving μcaptchas is channeled into creating labeled handwritten datasets. Looking forward, we believe our work suggests future research opportunities in current web security measures.

Acknowledgements

We are grateful to Mara Chinea for lending us her iPad mini. We would like to thank Vicente Bosch, Mauricio Villegas, and Vicent Alabau for fruitful discussions.

Funding

This work is part of the Valorization and I+D+i Resources program of VLC/CAMPUS and has been funded by the Spanish MECD as part of the International Excellence Campus program. This work has also been partially supported by the Spanish MINECO (grant TIN2014-37475), the FPU (grant AP2009-4363), the GVA (grant PROMETEOII/2014/030), and the EU 7th Framework Program (grant 600707).

References

Álvaro, F., Sánchez, J.-A., and Benedí, J.-M. (2014). Recognition of on-line handwritten mathematical expressions using 2D stochastic context-free grammars and hidden Markov models. *Pattern Recognition Letters*, 35(1):58–67.

Baird, H. S. and Bentley, J. L. (2005). Implicit CAPTCHAs. In *Proc. SPIE/IS&T Conf. on Document Recognition and Retrieval*, pages 507–518.

Baird, H. S. and Popat, K. (2002). Human interactive proofs and document image analysis. In *Proc. Document Analysis Systems (DAS)*, pages 507–518.

Bangor, A., Kortum, P. T., and Miller, J. T. (2008). An empirical evaluation of the system usability scale. *Intl. Journal of Human-Computer Interaction*, 24(6):574–594.

Basso, A. and Bergadano, F. (2010). Anti-bot strategies based on human interactive proofs. In *Handbook of Information and Communication Security*, pages 273–291. Springer Berlin Heidelberg.

Blocki, J., Blum, M., and Datta, A. (2013). GOTCHA password hackers! In *Proc. ACM Workshop on Artificial Intelligence and Security (AISec)*, pages 25–34.

Bursztein, E., Moscicki, A., Fabry, C., Bethard, S., Mitchell, J. C., and Jurafsky, D. (2014). Easy does it: More usable CAPTCHAs. In *Proc. SIGCHI Conf. on Human Factors in Computing systems (CHI)*, pages 2637–2646.

Chellapilla, K., Larson, K., Simard, P., and Czerwinski, M. (2005). Designing human friendly human interaction proofs (HIPs). In *Proc. SIGCHI Conf. on Human Factors in Computing systems (CHI)*, pages 711–720.

Chellapilla, K. and Simard, P. (2005). Using machine learning to break visual human interaction proofs (HIPs). In *Proc. Neural Information Processing Systems (NIPS)*, pages 265–272.

Chen, T., Yesilada, Y., and Harper, S. (2010). What input errors do you experience? typing and pointing errors of mobile web users. *Intl. Journal of Human-Computer Studies*, 68(3):138–157.

Chew, M. and Baird, H. S. (2003). BaffleText: a human interactive proof. In *Proc. Document Recognition and Retrieval (DRR)*, pages 305–316.

Chow, R., Golle, P., Jakobsson, M., Wang, L., and Wang, X. (2008). Making CAPTCHAs clickable. In *Proc. Workshop on Mobile Computing Systems and Applications (HotMobile)*, pages 91–94.

Coates, A. L., Baird, H., and Faternan, R. (2001). Pessimal print: A reverse turing test. In *Proc. Intl. Conf. on Document Analysis and Recgnition (ICDAR)*, pages 1154–1158.

Commarford, P. M. (2004). An investigation of text throughput speeds associated with Pocket PC input method editors. *Intl. Journal of Human-Computer Interaction*, 17(3):293–308.

Elson, J., Douceur, J. R., Howell, J., and Saul, J. (2009). ASIRRA: A CAPTCHA that exploits interest-aligned manual image categorization. In *Proc. ACM Conf. on Computer and Communications Security (CCS)*, pages 366–374.

Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. (2014). Multi-digit number recognition from street view imagery using deep convolutional neural networks. In *Proc. Intl. Conf. on Learning Representations (ICLR)*.

Gossweiler, R., Kamvar, M., and Baluja, S. (2009). What's up CAPTCHA? a CAPTCHA based on image orientation. In *Proc. WWW Conf.*, pages 841–850.

Hernandez-Castro, C. J. and Ribagorda, A. (2010). Pitfalls in CAPTCHA design and implementation: The Math CAPTCHA, a case study. *Comput. Secur.*, 29(1):141–157.

Hidalgo, J. M. G. and Alvarez, G. (2011). CAPTCHAs: An artificial intelligence application to web security. *Advances in Computers*, 83(1):109–181.

Hourcade, J. P. and Berkel, T. R. (2008). Simple pen interaction performance of young and older adults using handheld computers. *Interacting with Computers*, 20(1):166–183.

Jiang, N. and Tian, F. (2013). A novel gesture-based CAPTCHA design for smart devices. In *Proc. British Human Computer Interaction Conf. (BCS HCI)*, page 49.

Kienzle, W. and Hinckley, K. (2013). Writing handwritten messages on a small touchscreen. In *International Conf. on Human-Computer Interaction with Mobile Devices and Services (MobileHCI)*, pages 179–182.

Kluever, K. A. and Zanibbi, R. (2009). Balancing usability and security in a video CAPTCHA. In *Proc. Symposium on Usable Privacy and Security (SOUPS)*, pages 14:1–14:11.

Leiva, L. A., Alabau, V., Romero, V., Toselli, A. H., and Vidal, E. (2014). Context-aware gestures for mixed-initiative text editing UIs. *Interacting with Computers*. In Press. `http://dx.doi.org/10.1093/iwc/iwu019`.

Lin, R., Huang, S.-Y., Bell, G. B., and Lee, Y.-K. (2011). A new CAPTCHA interface design for mobile devices. In *Proc. Conf. on Australasian User Interface (AUIC)*, pages 3–8.

McCabe, P. T., editor (2002). *Contemporary Ergonomics*. CRC Press.

Mohamed, M., Sachdeva, N., Georgescu, M., Gao, S., Saxena, N., Zhang, C., Kumaraguru, P., van Oorschot, P. C., and Chen, W.-B. (2014). A three-way investigation of a game-CAPTCHA: Automated attacks, relay attacks and usability. In *Proc. Symposium on Information, Computer and Communications Security (ASIACCS)*, pages 195–206.

Price, K. J., Lin, M., Feng, J., Goldman, R., Sears, A., and Jacko, J. (2009). Nomadic speech-based text entry: A decision model strategy for improved speech to text processing. *Intl. Journal of Human-Computer Interaction*, 27(7):692–706.

Roshanbin, N. and Miller, J. (2013). A survey and analysis of current CAPTCHA approaches. *Journal of Web Engineering*, 12(1-2):1–40.

Ross, S. A., Halderman, J. A., and Finkelstein, A. (2010). Sketcha: A CAPTCHA based on line drawings of 3D models. In *Proc. WWW Conf.*, pages 821–830.

Rusu, A., Docimo, R., and Rusu, A. (2010). Leveraging cognitive factors in securing WWW with CAPTCHA. In *Proc. USENIX Conf. on Web Application Development (WebApps)*, pages 5–5.

Rusu, A. and Govindaraju, V. (2004). Handwritten CAPTCHA: Using the difference in the abilities of humans and machines in reading handwritten words. In *Proc. Intl. Conf. on Frontiers in Handwriting Recognition (ICFHR)*, pages 226–231.

Sain, K., Dasgupta, A., and Garain, U. (2011). EMERS: a tree matching-based performance evaluation of mathematical expression recognition systems. *Intl. Journal on Document Analysis and Recognition*, 14(1):75–85.

Shirali-Shahreza, M. and Shirali-Shahreza, S. (2006). Drawing CAPTCHA. In *Proc. Intl. Conf. on Information Technology Interfaces (ITI)*, pages 475–480.

Shirali-Shahreza, M. and Shirali-Shahreza, S. (2008). Motion CAPTCHA. In *Proc. Conf. on Human System Interaction (HSI)*, pages 1042–1044.

Shirali-Shahreza, S., Penn, G., Balakrishnan, R., and Ganjali, Y. (2013). SeeSay and HearSay CAPTCHA for mobile interaction. In *Proc. SIGCHI Conf. on Human Factors in Computing systems (CHI)*, pages 2147–2156.

Shirali-Shahreza, S. and Shirali-Shahreza, M. (2011). Multilingual highlighting CAPTCHA. In *Proc. Intl. Conf. on Information Technology New Generations (ITNG)*, pages 447–452.

Simard, P. Y., Szeliski, R., Benaloh, J., Couvreur, J., and Calinov, I. (2003). Using character recognition and segmentation to tell computer from humans. In *Proc. Intl. Conf. on Document Analysis and Recgnition (ICDAR)*, page 418.

von Ahn, L., Blum, M., and Langford, J. (2004). Telling humans and computers apart automatically. *Communications of the ACM*, 47(2):56–60.

von Ahn, L., Maurer, B., McMillen, C., Abraham, D., and Blum, M. (2008). re-CAPTCHA: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468.

Wobbrock, J. O., Wilson, A. D., and Li, Y. (2007). Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In *Proc. Annual ACM Symposium on User Interface Software and Technology (UIST)*, pages 159–168.

Xu, Y., Reynaga, G., Chiasson, S., Frahm, J.-M., Monrose, F., and Van Oorschot, P. (2012). Security and usability challenges of moving-object CAPTCHAs: Decoding codewords in motion. In *Proc. USENIX Conf. on Security Symposium*, pages 4–20.

Yan, J. and Ahmad, A. S. E. (2008). A low-cost attack on a Microsoft CAPTCHA. In *Proc. ACM Conf. on Computer and Communications Security (CCS)*, pages 543–554.

Yan, J. and El Ahmad, A. S. (2008). Usability of CAPTCHAs or usability issues in CAPTCHA design. In *Proc. Symposium on Usable Privacy and Security (SOUPS)*, pages 44–52.

Zanibbi, R. and Blostein, D. (2012). Recognition and retrieval of mathematical expressions. *Intl. Journal on Document Analysis and Recognition*, 15(4):331–357.

Zelkowitz, M. (2001). *Security on the Web*. Academic Press.

About the Authors

Luis A. Leiva, Ph.D., is a research staff member of the PRHLT Research Center at the Universitat Politècnica de València and the co-founder of Sciling. His research interests include Web Technologies, Human-Computer Interaction, Intelligent User Interfaces, Machine Learning, and the intersection between the four.

Francisco Álvaro, Ph.D. candidate, works at WIRIS and previously was with the PRHLT Research Center at the Universitat Politècnica de València, where he obtained an MSc in Artificial Intelligence, Pattern Recognition, and Digital Imaging. His research interests are Mathematical Expression Recognition and Machine Learning.