Asirra: A CAPTCHA that Exploits Interest-Aligned Manual Image Categorization

Jeremy Elson, John R. Douceur, Jon Howell
Microsoft Research
{jelson,johndo,howell}@microsoft.com

Jared Saul
Petfinder, Inc.
jared@petfinder.com

ABSTRACT

We present Asirra (Figure 1), a CAPTCHA that asks users to identify cats out of a set of 12 photographs of both cats and dogs. Asirra is easy for users; user studies indicate it can be solved by humans 99.6% of the time in under 30 seconds. Barring a major advance in machine vision, we expect computers will have no better than a 1/54,000 chance of solving it. Asirra's image database is provided by a novel, mutually beneficial partnership with Petfinder.com. In exchange for the use of their three million images, we display an "adopt me" link beneath each one, promoting Petfinder's primary mission of finding homes for homeless animals. We describe the design of Asirra, discuss threats to its security, and report early deployment experiences. We also describe two novel algorithms for amplifying the skill gap between humans and computers that can be used on many existing CAPTCHAs.

1. INTRODUCTION

Over the past few years, an increasing number of public web services have attempted to prevent exploitation by bots and automated scripts, by requiring a user to solve a Turing-test challenge (commonly known as a *CAPTCHA*[1] or *HIP*[2]) before using the service. Because the challenges must be easy to generate but difficult (for non-humans) to solve, all CAPTCHAs rely on some secret information that is known to the challenger but not to the agent being challenged. For our purposes, we can divide CAPTCHAs into two classes depending on the scope of this secret. In *Class I* CAPTCHAs, the secret is merely a random number, which is fed into a publicly known algorithm to yield a challenge, somewhat analogous to a public-key cryptosystem. *Class II* CAPTCHAs employ both a secret random input and a secret high-entropy database, somewhat analogous to a one-time-pad cryptosystem. A critical problem in building a Class II CAPTCHA is populating the database with a sufficiently large set of classified, high-entropy entries.

Class I CAPTCHAs have many virtues. They can be concisely described in a small amount of software code; they have no long-term secret that requires guarding; and they can generate a practically unbounded set of unique challenges. On the other hand, their most common realization—a challenge to recognize distorted text—evince a disturbingly narrow gap between human and non-human success rates. Optical character recognition algorithms are competitive with humans in recognizing distinct characters, which has led researchers toward increasing the difficulty of segmenting an image into distinct character regions [11]. However, this increase in difficulty affects humans as well. Although laboratory experiments suggest that humans can segment text characters accurately [2], CAPTCHAs deployed on commercial public web sites continue to use cleanly segmented challenges (e.g., Fig. 2a), some of which are dialed-down versions (Figs. 2d and 2e) of CAPTCHAs with difficult segmentation challenges (Fig. 2c). The owners of commercial web sites have apparently decided that a user's success at navigating a CAPTCHA depends not only on whether they are *able* to solve the challenge, but also on whether they are *willing* to put forth the effort. Informal discussions with MSN and other web site owners suggest even relatively simple challenges can drive away a substantial number of potential customers.

Class II CAPTCHAs have the potential to overcome the main weaknesses described above. Because they are not restricted to challenges that can be generated by a low-entropy algorithm, they

Figure 1: An Asirra challenge. The user selects each of the 12 images that depict cats. As the mouse is hovered over each thumbnail, a larger image and "Adopt me" link appear. "Adopt me" first invalidates the challenge, then takes the user to that animal's page on Petfinder.com.

[1]"Completely Automated Public Turing test to tell Computers and Humans Apart." CAPTCHA is a trademark of Carnegie Mellon University.
[2]"Human Interaction Proof"

CCS'07, October 29–November 2, 2007, Alexandria, Virginia, USA.

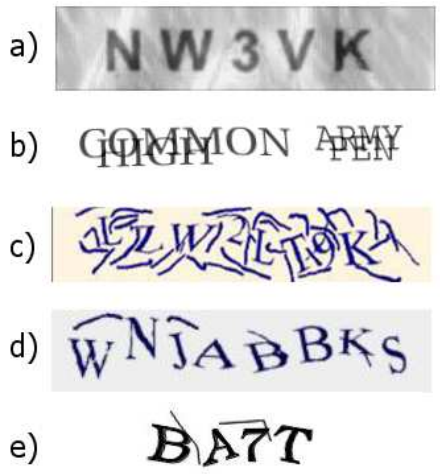

Figure 2: A gallery of text CAPTCHAs. Simple text challenges, such as *a* (register.com), are still common despite recent defeat by optical character recognition. Researchers have begun to focus on schemes that make letter segmentation difficult, as seen in *b* (Carnegie Mellon [13]) and *c* (Microsoft Research [2]). Webmasters, wary of what users will tolerate, dial back researchers' noise parameters, seen in *d* (Microsoft Hotmail) and *e* (Yahoo! Mail).

can exercise a much broader range of human ability, such as recognizing features of photographic images captured from the physical world. Such challenges evince a broad gulf between human and non-human success rates, not only because general machine vision is a much harder problem than text recognition, but also because image-based challenges can be made less bothersome to humans without drastically degrading their efficacy at blocking automatons.

A significant issue in building a Class II CAPTCHA is populating the secret database. Existing approaches take one of two directions: (a) mining a public database or (b) providing entertainment as an incentive for manual image categorization. Examples of the first group include the seminal work by Chew and Tygar [3], which used Google Image Search [6]; hotcaptcha [1], which references the HotOrNot database [7]; and KittenAuth [16], which draws images from Wikimedia Commons [17]. A problem with these approaches is that the public source of categorized images is small or available to attackers, so a small, fixed amount of effort spent reconstructing the private database can return the ability to solve an unbounded number of challenges. The second direction was pioneered by the ESP-PIX CAPTCHA [13], whose database is populated as a deliberate side effect of playing the ESP Game [15], a very clever mechanism for enticing people to label images accurately. Although potentially powerful, it is not yet clear whether this approach will yield a sufficiently large set of categorized images. Furthermore, many of the images in the current implementation are rather abstract, which may make the challenge difficult enough to drive away users.

In this paper, we present a new direction for populating image databases for Class II CAPTCHAs, namely re-purposing a large, continually evolving, private database of images that are manually categorized. Although this may seem trivial, it is not *a priori* clear why the owner of such a database would be willing to release the images for use in Turing-test challenges. The answer is that there can exist—and, in at least one instance, does exist—an *alignment of interests* between a database owner and web-service owners wishing to secure their sites. Both parties can benefit from selective, wide-scale display of categorized images: the latter for security and the former for advertising.

We present **Asirra**[3], a CAPTCHA that asks users to categorize photographs depicting either cats or dogs. An example is shown in Figure 1. Asirra's strength comes from an innovative partnership with Petfinder.com [9], the world's largest web site devoted to finding homes for homeless animals. Petfinder has a database of over three million cat and dog images, each of which is categorized with very high accuracy by human volunteers working in thousands of animal shelters throughout the United States and Canada. Petfinder has granted ongoing access to its database, which grows by nearly 10,000 images daily, to the Asirra project. In exchange, Asirra provides a small "Adopt me" link beneath each photo, promoting Petfinder's primary mission of exposing adoptable pets to potential new owners. This partnership is mutually beneficial, and also produces the dual social benefits of improving computer security and animal welfare.

This paper describes Asirra and an analysis of its strengths and weaknesses. We also report our deployment experience, and the results of two user studies involving 332 test subjects.

Asirra is easy for users; it can be solved by humans 99.6% of the time in under 30 seconds (Section 6, Table 1). Barring a major advance in machine vision or compromise of our database, we expect computers will have no better than a 1/54,000 chance of solving it (Section 6, Table 2). Anecdotally, users seem to find the experience of using Asirra much more enjoyable than a text-based CAPTCHA that provides equal security.

The organization of this paper is as follows. In Section 2, we review related work in more detail. In Section 3, we describe the design of Asirra. §3.1 describes user experiments we performed to quantify humans' performance. §3.2 explores the other side of the equation—potential attacks on Asirra, and how they can be resisted. We developed two algorithms that can be used to improve virtually all CAPTCHAs, including those that are text-based; these improvements are described in Section 4. In Section 5 we describe our scalable Asirra implementation. Finally, in Section 6, we summarize our contributions and offer conclusions.

Asirra is available free at `www.asirra.com`.

2. RELATED WORK

Since the concept of a CAPTCHA was widely introduced by von Ahn in 2000 [14], hundreds of design variations have appeared. By far, most are text-based: The computer generates a challenge by selecting a sequence of letters, rendering them, distorting the image, and adding noise. Text CAPTCHAs are popular because they are simple, small, and easy to design and implement. Challenges as short as four characters are robust against random guessing; there are $36^4 \approx 1.7$ million possible four-character challenges consisting of case-insensitive letters and digits.

Unfortunately, computers can do far better than guess randomly. Simard et al. showed that Optical Character Recognition (OCR) can achieve human-like accuracy, even when letters are distorted, as long as the image can be reliably segmented into its constituent letters [11]. Mori and Malik demonstrated that von Ahn's original GIMPY CAPTCHA [13] can be solved automatically 92% of the time [8].

Consequently, recent text-based CAPTCHAs have focused on making image segmentation difficult. Figure 2c shows a challenge designed by Chellapilla et al., who claim it is hard for OCR because the noise confounds known segmentation techniques [2]. Microsoft's Hotmail (free email service) deployed it; however, due

[3]"Animal Species Image Recognition for Restricting Access"

to usability concerns, they later selected noise parameters demonstrated in Figure 2d. Yahoo's current CAPTCHA, shown in Figure 2e, seems to have suffered a similar fate. The noise is not sufficient to make automatic segmentation unreliable. Text CAPTCHAs seem either too easy to be secure, or too difficult to be tolerated by users.

2.1 Image Classification CAPTCHAs

Text-based CAPTCHAs seem to universally suffer from an unfortunate property: Making them hard for computers also makes them hard for humans. This has led some researchers to use photographs as CAPTCHAs instead. Because general machine vision is a much harder problem than character recognition, there are opportunities to find and exploit larger gaps in the capabilities of humans and computers.

Chew and Tygar [3] were among the first to describe using labelled photographs to generate a CAPTCHA. They generated a database of labelled images by feeding a list of easily-illustrated words to Google Image Search [6]. Unfortunately, this technique does not yield well-classified results due to Google's method of inferring photo contents based on surrounding descriptive text. To use Chew and Tygar's example, the word *pumpkin* may refer to either a large vegetable or someone's pet cat Pumpkin. Because of these errors, they manually cull bad images from their collection. This is devastating to the security of the scheme. A database small enough to be manually constructed by researchers is also small enough to be manually *re*-constructed by an attacker.

Of course, applying automation to database construction is inherently problematic. If a researcher can populate a database by writing a program to automatically classify images, an attacker can write a program to beat the CAPTCHA by performing the same classification.

A novel solution to this problem is described by von Ahn *et al.*: They were able to entice humans to manually describe images by framing the task as a game. Their "ESP Game" awards points to teams of non-communicating players who can both pick the same label for a random image, encouraging them to use the most obvious label [15]. Their PIX CAPTCHA displays four images from the ESP Game database that have the same label, then challenges the user to guess the label from a menu of 70 possibilities.

PIX is clever, but has several potential problems. First, its scale seems insufficient. By solving PIX repeatedly, it is not hard to get repeated images, making the database easy to reconstruct by an attacker. However, perhaps more fundamental, it has a fixed menu of only 70 object classes. This makes it a potential target for brute force attacks (though potentially defensible using our token bucket scheme; see Section 4.2). Even with a large number of categorized images, it may be difficult to add a large number of classes. As the number of classes goes up, so does the number of words that could reasonably be used to describe a set of photos. Finally, PIX photos are sometimes abstract, making it potentially difficult or frustrating as a CAPTCHA.

A fascinating use of a large-scale human-generated database is the site HotCaptcha.com. HotCaptcha displays nine photographs of people and asks users to select the three which are "hot." Its database comes from HotOrNot.com, a popular web site that invites users to post photos of themselves and rate others' photos as "hot" or "not." HotCaptcha is clever in its use of a pre-existing motivation for humans to classify photos at a large scale. However, humans may have difficulty solving it because the answers are subjective and culturally relative; beauty has no ground truth. It is also offensive to many people, making it difficult for serious web sites to deploy.

Finally, worthy of mention is the similar-seeming KittenAuth project [16]. Like Asirra, KittenAuth authenticates users by asking them to identify photos of kittens. However, this is a coincidental and superficial similarity. KittenAuth is trivial to defeat because it is has a database of less than 100 manually selected kitten photos. An attacker can (indeed, already has [12]) expose the database by manually solving the KittenAuth challenge a few dozen times. An arbitrary number of challenges can then be solved using an image comparator robust to simple image distortions.

3. ASIRRA

Asirra surmounts the image-generation problem in a novel way: by forming a partnership with Petfinder.com [9], the world's largest web site devoted to finding homes for homeless pets. Asirra generates challenges by displaying 12 images from a database of over three million photographs that have been manually classified as cats or dogs. Nearly 10,000 more are added every day by volunteers at animal shelters throughout the United States and Canada. The size and accuracy of this database is fundamental to the security provided by Asirra; it is what differentiates our work from previously proposed image-based CAPTCHAs.

In exchange for access to Petfinder's database, Asirra provides an unobtrusive "Adopt me" link beneath each photo. This promotes Petfinder's primary mission of exposing adoptable pets to potential new owners. To maximize the probability of successful adoptions, Asirra will employ IP geolocation to determine the user's approximate region, and preferentially displays pets that are nearby. The security implications of this feature are discussed in Section 3.2.1.

Asirra has several attractive features:

- Humans can solve it quickly (§3.1.2) and accurately (§3.1.3).
- Computers can not solve it easily (§3.2).
- Unlike many image-based CAPTCHAs which are abstract or subjective, Asirra's challenges are concrete, inoffensive (cute, by some accounts), require no specialized or culturally biased knowledge, and have definite ground truth. This makes Asirra less frustrating for humans. Some beta-testers found it fun. The four-year-old child of one asked several times to "play the cat and dog game again."
- It promotes an additional social benefit: finding homes for homeless animals.

Asirra also has several disadvantages:

- Most CAPTCHAs are implemented as stand-alone program libraries that can be integrated into a web site without introducing external dependencies. In contrast, Asirra, like PIX, is both an algorithm and a *database*; there is only one instance of it. Therefore, Asirra must be implemented as an administratively centralized web service that is used to generate and verify CAPTCHAs on-demand for every site that wishes to use it. (Our scalable implementation is described in Section 5.)
- Asirra may abruptly lose its security if the database is compromised. For example, an attacker may hire cheap labor to classify all three million images. If this does happen, we may not even be aware of the attack.
- A typical Asirra challenge requires more screen space than a traditional text-based CAPTCHA.
- Like virtually all other CAPTCHAs, Asirra is not accessible to those with visual impairments (§3.3).

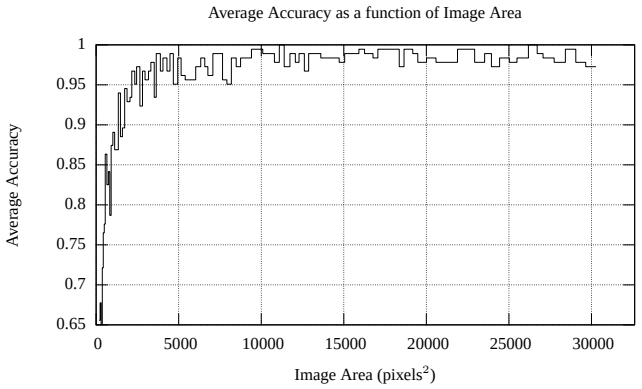

Figure 3: Size of an individual image vs. classification accuracy in our test population.

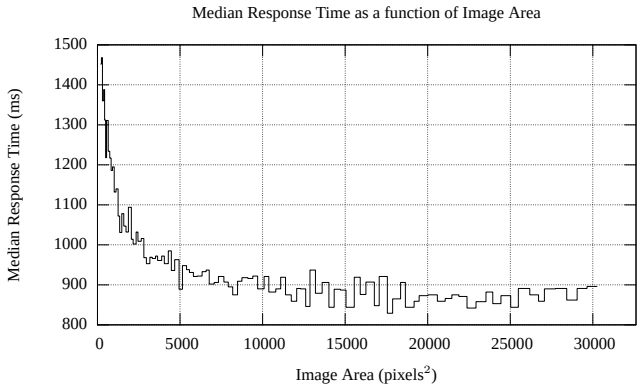

Figure 4: Size of an individual image vs. time required to classify it in our test population.

3.1 Usability

To quantitatively evaluate Asirra, we performed several user studies. We also collected data from our live Asirra deployment, which is described in Section 5.

The user studies, in total, displayed 23,208 cat and dog images to 332 test subjects. The subjects were Microsoft employees, friends, and family, recruited via postings to internal mailing lists. Our experiment displayed a random sequence of cat and dog images, one at a time. Users were asked to identify the species depicted in each image as it appeared. We used 300 random images from the Petfinder database, each scaled to a random size as described in the next section. The users' response time and accuracy were recorded. To match the conditions of a real CAPTCHA, we implemented the experiment in the web browser using HTML and JavaScript, and asked users to participate from their own home or office computers.

The real Asirra web service went live in March of 2007. In the subsequent 6 weeks, it served about 100,000 challenges. Most of these were from our own web page that demonstrates Asirra. However, several dozen web sites (blogs, free services, etc.) have experimented with integrating Asirra and were responsible for about 13,000 "real" challenges.

In following sections, we characterize various aspects of Asirra's performance, based on both the outcome of our user experiments and data from our deployment.

3.1.1 Best Image Size

Our first design question was, "What is the best image size to display?" We expected larger images to exhibit higher accuracy and faster response time. However, smaller images have the advantage of taking up less screen space, making Asirra easier to integrate visually with the rest of a web page. Smaller images are also faster to download, which especially important for users with slow Internet connections.

To find the best image size, our first user experiment randomly varied the size of the images from a minimum of 225 total pixels (about 15x15 pixels square) to a maximum of 30,000 pixels (about 175x175 pixels square). We used total pixels instead of linear dimension as our metric because most images are not square. We collected data from 18,311 displays of images to 185 users.

Figure 3 plots image size vs. average accuracy. Each graph segment depicts the average of 1/100th (183) of our data points. 10,000 pixels seems to be the sweet spot: larger images show no improvement. (We quantify classification accuracy in Section 3.1.3).

3.1.2 Typical Response Time

We also used our first experiment to evaluate the effect of image size on response time. Figure 4 shows the results. We plot the median response time because it is robust against outliers (e.g., when a subject receives a phone call during the experiment). 10,000 pixel images are typically classified in about 900msec. There does seem to be a slight (50msec) speed benefit to displaying images larger than 10,000 pixels. It is also interesting to note from Figures 3 and 4 that smaller images cause degradation in both response time *and* accuracy. That is, users spend longer looking at an image but still end up getting it wrong.

Budgeting 900msec per image, plus a few extra seconds to understand the task, we expect most users will spend about 15 seconds solving a single, 12-image Asirra challenge.

3.1.3 Classification Accuracy

After our first experiment made the best image size clear, we ran a second user experiment focusing exclusively on images scaled to 10,000 pixels. Our goal was to collect a large number of samples at our target image size. We collected data from 4,717 displays of images to 147 users. The overall accuracy rate was 98.5%.

The Asirra challenge has 12 images. Based on a 98.5% per-image accuracy, 83.4% of users should be able to pass Asirra after one challenge, 97.2% after two challenges, and 99.5% after three. However, Asirra uses a novel scheme called the *Partial Credit Algorithm*, described in Section 4.1, which significantly increases the probability of users passing Asirra while only marginally improving the yield of a bot. With PCA, we expect 99.6% of users will solve Asirra after two challenges.

Extracting a good estimate of users' error rates in the wild is somewhat more difficult. In our controlled experiment, users were focused exclusively on image categorization, so errors were clearly attributable to task difficulty. In contrast, the deployed CAPTCHA is integrated into web forms with many other fields and information. Any click of the form's "Submit" button causes Asirra to score the challenge, even if the user had some other intent in mind; 26% of scoring requests received by our server had no response at all (no cats selected). Of non-null responses scored, 66% were scored as correct. Note that in this accounting, a user who successfully solves Asirra on her second try counts as 50% accuracy (one right plus one wrong response). In addition, since Asirra is new, users may sometimes be tinkering: "Oh, what happens if I get one wrong?"

Figure 5: Though rare, some images at Petfinder.com are confusing to humans. *(left)* A photo depicting a cat and dog together. *(right)* A dog-like cat.

3.1.4 Automatic Database Pruning

The classification errors made by users are not uniformly distributed: some images are confusing, even to humans. Figure 5 shows two examples.

If Asirra achieves only modest popularity, our accuracy figures are likely to stay close to those described in the previous section. However, if Asirra receives widespread use (tens of thousands of challenges generated per day), we will have enough data to make automatic, statistically significant judgements about confusing images. There will be cases when a user ultimately succeeds in passing Asirra, but may have made errors along the way. Once Asirra decides a user is human, it will be able to mark the previously-incorrect images as "possibly confusing." Images that are marked beyond a threshold can then be removed from the database.

3.1.5 Interest in the "adopt me" link

In both user experiments, we placed a small "Adopt me" link below the cat or dog displayed in each trial. We did not give users any instructions regarding this link or tell them what it did. Our goal was to test how often Internet users involved in some unrelated task would be motivated (by curiosity, cuteness of a photo, etc.) to click it. This test is important because it suggests whether our partnership with Petfinder is viable.

In our controlled experiments, 27 users (7.5% of the test population) clicked "Adopt me". One of our beta-testers adopted a beagle.

The public's interest in the link has been lower. In the 6 weeks following Asirra's release, we issued 13,334 "real" challenges—that is, integrated with pages other than our own demonstration pages. 279, or 2.1%, led to clicks on "Adopt me".

We discuss the security implications of "Adopt me" in §3.2.1.

3.2 Security

Before discussing the security of Asirra, it is useful to review the threat model. CAPTCHAs are an unusual area of security in that we are not trying to provide absolute guarantees, only slow down attackers. By definition, anyone can "break" a CAPTCHA by devoting a small amount of human effort to it. A CAPTCHA therefore is successful, essentially, if it forces an automated attack to cost more than 30 seconds worth of a human's time in a part of the world where labor is cheap. The generally accepted figure in the literature [13, 2] seems to be that CAPTCHAs should admit bots less than with less than 1/10,000 probability.

3.2.1 Attacks on "Adopt Me"

The most common first question we get when people see Asirra is, "Doesn't the adopt-me link defeat your security?" This misconception is understandable; each pet's page on Petfinder.com describes it as being either a cat or dog. However, this is not actually a security hole. The adopt-me links are not direct links to Petfinder.com; they lead to the Asirra web service. Asirra provides a redirection to Petfinder.com only after it has marked the challenge as invalid. (A new challenge is then fetched and displayed.) Asirra rejects attempts to solve invalidated challenges. In addition, Asirra only permits redirection for a single adopt-me link per 12-image challenge. The number of allowed redirections per IP address per day is also limited, to prevent adopt-me from becoming a vector for revealing large portions of the database.

Adoption links also have a second, more subtle effect on security. Currently, the images shown as a challenge are selected at random from the entire database of pet images. To maximize its utility to Petfinder, we would ultimately like to restrict the challenge to pets that are close to the user based on IP geolocation, and currently available for adoption. These constraints reduce the usable database for a given user to just a few thousand images, small enough to be easily exploitable. Our plan, therefore, is to allow the first few challenges per day from an IP address to use the restricted database; subsequent challenges will be drawn from the complete collection. We have not yet implemented this feature, so our challenges currently draw images from Petfinder's complete image pool, including the history of pets previously available for adoption.

3.2.2 Brute Force

The simplest attack on Asirra is brute force: Give a random solution to challenges until one succeeds. If an attacker has no basis for a decision (that is, a 50% probability of success for each image) brute force will succeed with probability 1/4,096 for a 12-image challenge. This is a large enough slowdown that it becomes easy to detect and evade such an attack. We use a token bucket scheme, described in detail in Section 4.2. Briefly, the scheme penalizes IP addresses that get many successive wrong answers by forcing them to answer two challenges correctly within 3 attempts before gaining a ticket. With this scheme in place, attackers can expect only one service ticket per 5.6 million guesses.

3.2.3 Machine Vision Attacks

While random guessing is the easiest form of attack, various forms of image recognition can allow an attacker to make guesses that are better than random. Asirra's strength, however, comes not only in the size of its image database, but its diversity. Photos have a wide variety of backgrounds, angles, poses, lighting, and so forth—factors that make accurate automatic classification difficult. As Figure 6 demonstrates, the variations between photos are large, but visual differences between cats and dogs are often subtle.

Based on a survey of machine vision literature and vision experts at Microsoft Research, we believe classification accuracy of better than 60% will be difficult without a significant advance in the state of the art. For example, the best known algorithms for facial recognition can achieve in excess of 90% accuracy under controlled conditions, but are not robust to occlusions and variations in pose similar to those seen in our database [5, 18]. The 2006 PASCAL Visual Object Classes Challenge [4] included a competition to identify photos as containing several classes of objects, two of which were Cat and Dog. Although cats and dogs were easily distinguishable from other classes (e.g., "bicycle"), they were frequently confused with each other.

One attack on our database proposed by vision researchers was color histogram analysis. We implemented and tested this technique. We first characterized each of 150 random images as a 15-feature color vector, computing histograms of each fifth of the red,

Figure 6: The differences between cats and dogs are immediately obvious to humans. In many cases, species look similar, with only subtle cues to distinguish them. This makes it a hard vision problem.

green, and blue channels of each image. We then computed the best least-squares fit coefficients of these 15 features to the two (cat and dog) image classes. The coefficients were then used to predict the class of another 150 images. The result was 56.9% accuracy.

If we assume an attacker can build a 60% accurate classifier, the probability of solving a 12-image challenge improves from 1/4,096 to 1/459. However, with our token bucket scheme in place (§4.2), this is reduced to one ticket per about 70,000 guesses.

3.2.4 Database Attacks

One way to attack any image-based CAPTCHA is to reconstruct its underlying database. In a sense, this attack is unavoidable: A portion of the database is revealed each time a challenge is displayed. By solving enough challenges manually, the entire database is eventually revealed to the attacker. The key question is not "is database reconstruction possible?" but rather "is database reconstruction economical?"

A CAPTCHA can be considered secure if the most cost effective way to defeat it is for a human to solve it once per unit of service gained. In other words, our goal is to make the most efficient attack an on-demand one (e.g., pay people to sign up for one Hotmail account at a time). This tradeoff is directly informed by the size and stability of the underlying database.

Oli Warner's KittenAuth provides an instructive example. It originally launched with only 42 images; within days, someone posted a map of those images' MD5 hashes to a cat-or-dog bit [12]. Classification of 42 images does not take much investment, so this form of attack was an easy choice. Asirra's three million images moves the break-even point significantly. An attacker would expect to solve about 750,000 12-image challenges to reveal 95% of the database. Reconstruction of our database is not economical unless there is a financial incentive to solve millions of Asirra challenges.

Attackers might also try to reconstruct Asirra's database by attacking its source—that is, automating a long series of queries to Petfinder.com. However, Petfinder's public interface only displays pets currently available for adoption, which represents less than 10% of their total historical database. In addition, there is no efficient way to track database changes using Petfinder's public interface. The private API provided to Asirra by Petfinder is not available to the public.

Note that in this section we do not consider techniques for reusing an individual image such that it appears to be multiple distinct images. There are robust image comparator functions (e.g., image hashes, color histograms) that are insensitive to many simple image distortions. Warping an image sufficiently to fool a computer will likely also be troublesome to a human—the arms race found in text CAPTCHAs that we are trying to avoid.

3.2.5 Implementation Attacks

Many CAPTCHAs suffer from weak implementations. For example, some allow a session ID authorized by a single successful challenge to be re-used repeatedly to gain access to a protected service. Some assume trustworthy clients.

To avoid such weaknesses, we have taken two approaches. First, we made the Asirra web service as conceptually simple as possible, so that we can verify its correctness via code review. (The web service is about 700 lines of Python code.) Second, as we discuss further in Section 5.1, we designed the Asirra interface so as to minimize the implementation burden on the webmaster. For example, we do not require sites that use Asirra to keep track of user sessions or any other form of state. Our example service is, in fact, completely stateless.

3.3 Accessibility

Virtually all visual CAPTCHAs, including Asirra, are not accessible to visually impaired users. The problem of CAPTCHA accessibility is an important one, but beyond the scope of this paper. Accessible web sites typically allow users to switch between a visual challenge (e.g., distorted letters) and an auditory one (e.g., numbers being recited over noise). Asirra is only meant to replace visual CAPTCHAs. The audio-accessible version of a site's CAPTCHA is valuable, but orthogonal to Asirra.

However, it is worthwhile to note that Asirra's core idea of interest alignment is potentially usable in an audio CAPTCHA. For example, online music stores typically provide free samples of millions of songs in their catalogs. These might be used with a "buy me" link.

4. IMPROVEMENTS TO ALL CAPTCHAS

In the course of our work on Asirra, we developed two algorithms that can be used to improve virtually all CAPTCHAs, including those that are text-based. These techniques share a common theme. Imagine that users' responses to Asirra challenges were scored manually, by a human judge, instead of automatically by computer. Even in this seemingly straightforward task, the flexibility of human judgement would be a valuable asset. For example, our human judge could see the same user try to solve three challenges, getting 11 out of 12 images correct each time. She might say, "That looks like a human who is just having a little trouble. I'll let him pass." Conversely, our judge might see the same IP address submitting thousands of random, incorrect answers. Even when finally given a correct answer, she'd still suspect the user was a bot.

In the next sections, we describe two algorithms designed to make CAPTCHA scoring a little more "human": the Partial Credit Algorithm (§4.1) and CAPTCHA Token Buckets (§4.2).

4.1 The Partial Credit Algorithm

Traditionally, computers score CAPTCHA responses with one bit of output: right or wrong. The intuition behind the Partial Credit Algorithm (PCA) is that *a user's response contains much more than one bit of information*. Two "almost right" answers are strong evidence that a user is human; two random answers are not. Without PCA, this distinction is lost.

The idea of an "almost right" response is meaningful in the context of a wide variety of CAPTCHAs. For example, in a text-based CAPTCHA, typing five out of six characters correctly can be considered almost-right. In Asirra, we consider a response to be almost-right if 11 out of the 12 images presented are identified correctly.

Asirra awards Partial Credit to a user who gets a challenge almost right. If the user gets a subsequent challenge almost right—that is, while already holding Partial Credit—we judge the response as if it were completely correct. (Asirra challenges all take place within a *session*, making it easy to associate a single user's responses.)

The appeal of PCA is that its use of previously-ignored information allows us to significantly improve the pass rates for humans while minimally improving the pass rates for bots. For example, in Asirra, PCA reduces the number of humans rejected after 2 challenges from 2.8% to 0.4%: a 7-fold reduction in unhappy users (Section 6, Table 1). However, the bot yield improves from 1/4,096 to 1/3,952: only a 3% improvement (Section 6, Table 2).

In comparison, simply scoring every almost-right answer as correct (i.e., passing users who get 11/12) has a devastating effect on security: a random-guess bot's success rate improves from 1/4,096 to 1/315, a 13-fold increase.

We can mathematically model the effect of PCA as follows. We consider a new user to arrive in the unverified (u) state, and by solving a challenge the user moves into the verified (v) state. PCA introduces an intermediate (i) state, which the user moves into when getting a challenge almost right. From the intermediate state, if the user almost—or completely—solves a subsequent challenge, the user moves to the verified state; otherwise, the user is returned to the unverified state.

Let α represent the probability of solving a challenge, β be the probability of getting close enough to enter the intermediate state, and γ be the probability of getting close enough to become verified when in the intermediate state. After n steps, the probability that the user is in each state is given by the following recurrence relation:

$$\begin{aligned} u_n &= (1-\alpha-\beta)\,u_{n-1} + (1-\gamma)\,i_{n-1}, & u_0 &= 1 \\ i_n &= \beta\,u_{n-1}, & i_0 &= 0 \\ v_n &= v_{n-1} + \alpha\,u_{n-1} + \gamma\,i_{n-1}, & v_0 &= 0 \end{aligned}$$

The expected number of trials until verification is:

$$E = \sum_{1 \le n < \infty} n\,(v_n - v_{n-1}) = \frac{1+\beta}{\alpha+\beta\,\gamma}$$

In Asirra, the user moves to the intermediate state if exactly one image (out of 12) is misclassified; from the intermediate state, the user moves to the verified state if zero or one image is misclassified. Thus, using the binomial distribution function b, the PCA probabilities are:

$$\begin{aligned} \alpha &= b(0; 12, 1-p) \\ \beta &= b(1; 12, 1-p) \\ \gamma &= b(0; 12, 1-p) + b(1; 12, 1-p) \end{aligned}$$

This model can be used to compute the effect of PCA on any CAPTCHA that can define a solution as "almost right." The effect on 12-image Asirra challenges is shown in Tables 1 and 2.

4.2 CAPTCHA Token Buckets

Asirra, like virtually every CAPTCHA, is subject to brute-force attacks. Since guessing is nearly free, a success probability of 1/4,096 may not be a strong enough assurance that attackers can't automatically collect service tickets en masse.

Our countermeasure is based on the supposition that a bot can be identified as a small number of IP addresses that submit a very large number of incorrect responses, interspersed with a much smaller number of correct responses. (Assume for the moment that each user has a unique IP address; we will relax this constraint shortly.)

The scheme works by assigning a token bucket to every IP address that requests an Asirra challenge. Each bucket is initialized with *TB-Max* tokens. Every time a client submits *any* response to a challenge, a single token is removed from the client IP's bucket, down to a minimum of 0. Every time a client submits a *correct* response, *TB-Refill* tokens are added, up to a maximum of *TB-Max*. In our implementation, *TB-Max* is 100 and *TB-Refill* is 3. (The bucket is also re-initialized to *TB-Max* tokens after 24 hours of disuse, but this is not strictly required.) If a user submits a response while its token bucket is empty, the user is told the answer is incorrect, regardless of their response.

A brute-force bot trying to earn a large number of service tickets from Asirra will quickly empty its token bucket, as its incorrect guesses will outnumber its correct guesses by more than a factor of *TB-Refill*. In this steady-state, the empty-bucket policy will force

bots to correctly answer *two* challenges within *TB-Refill* attempts before they earn credit for a correct response.

Intuitively, the effect of this policy is to amplify the skill difference between humans and bots. Modelling it mathematically is straightforward. A bot has some probability p of getting a single challenge correct. With an empty token bucket, the bot must also get a second challenge right within *TB-Refill* tries; this happens with probability $1-(1-p)^{TB-Refill}$. That is, token buckets reduce the bot success probability to

$$p \times \left(1-(1-p)^{TB-Refill}\right)$$

For example, if a bot has a 1/4,096 chance of getting a single challenge correct, and *TB-Refill* is 3, our scheme reduces the probability of getting a ticket to one per about 5.6 million attempts per bot IP address.

4.2.1 Preventing denial-of-service against humans

Our scheme, as described so far, makes the assumption that each user has a unique IP address. That assumption is both strong and incorrect. Users share IP addresses when they access the Internet via a web proxy or firewall that performs network address translation. Such configurations are common in large corporations and some Internet Service Providers.

This is a problem for the token bucket scheme we described previously. When a human shares an IP address—and, thus, a token bucket—with a bot, the human is denied service while the bot is attacking. Since bots are much faster than humans, the human is likely to encounter an empty token bucket every time she submits an answer, and thus will always be marked wrong.

We address this problem by defining a slightly more complex scheme that uses two token buckets. The first is a per-IP bucket, as described before. The second is a per-*session* bucket. (All interactions with Asirra must take place within a session, created when a client first begins to interact with Asirra's web service.) The rules governing the buckets are:

- Per-IP buckets are given an initial value of *TB-Max* tokens, as before.
- When a new session is created, the per-session bucket is given an initial value equal to the current value of the client's per-IP bucket. Creation of a new session also deducts a token from the per-IP bucket.
- When any response is submitted, a token is deducted from *both* the per-IP and per-session bucket.
- When a correct response is submitted, *TB-Refill* tokens are added to *both* the per-IP and per-session bucket.
- A user is only told their response is correct if their *session* bucket is non-empty.

Intuitively, this scheme blacklists IP addresses, but allows humans to restore service to individual sessions by solving one extra challenge correctly.

Using a single, per-session bucket would not be effective because clients are untrusted. A bot does not play by the rules, and would simply create a new session before every guess. Therefore, the per-IP bucket is still necessary. However, legitimate users *will* create a single session and use it persistently, allowing us to distinguish them from bots sharing their IP address. A user sharing an IP with an attacking bot will initially get an empty per-session bucket, but will add *TB-Refill* tokens after her first correct response. These per-session tokens are not consumed by the bot, so the user is granted access after a second correct response.

5. DEPLOYMENT

Most CAPTCHAs are *algorithms*; their realization is typically source code that can be shared freely and integrated into web sites without external dependencies. In contrast, Asirra is both an algorithm and a *database* that must be kept secret. To protect the database, we had no option but to implement Asirra as a web service. In this section, we briefly describe our service's interface, implementation, and deployment.

The service is divided into the trusted web service, run by us, and the untrusted client component, implemented by us but run in the end-user's browser.

5.1 Interface

Our goal in designing the interface to the web service was to make it as easy as possible for the webmaster. Overly complex interfaces tend to be mis-used in ways that compromise security.

Suppose a web site wishes to use Asirra to protect a sign-up form. Our site has instructions for the webmaster describing how to include a few lines of JavaScript in the HTML form she wishes to protect. This code loads the bulk of Asirra's front-end implementation from the Asirra server.

Once the end-user views, completes, and submits the form—including solving the Asirra CAPTCHA—our JavaScript will set a hidden input field in the HTML form called `Asirra_Ticket`. The ticket is a bit string that the webmaster then must validate against the Asirra web service to ensure the client did not forge or re-use it. This validation typically happens as a step of processing the client's form on the webmaster's back-end server.

Ticket validation is simple: the ticket is passed to an Asirra validation URL. Our web service returns "Pass" only if the ticket is valid, recent, and has never been validated before. The JavaScript that manages user interactions on the client prevents the form from being submitted until a valid ticket has been received by passing the Asirra challenge. Thus, webmasters typically only see validations fail when their service is under attack.

5.2 Implementation

Asirra manages its own interaction with the user inside an embedded DIV. The interactions between client and web service are performed using "AJAX" (asynchronous HTTP requests executed from within JavaScript). When our JavaScript is loaded, it first creates the visual elements of the challenge in the user's browser. It then sends requests to the Asirra web service to create a new Asirra session, retrieve one or more challenges, and submit user responses for scoring. The client earns a service ticket for a correct response (or, in some cases, an almost-correct response; see Section 4.1). The user can choose to see a new challenge if they are asked to solve one that contains a confusing image.

Though administratively centralized, the web service's implementation is distributed to ensure scalability. Load balancing is achieved by mapping our web service's DNS name to multiple IP addresses; clients choose among them randomly. The first action performed by a client is session creation. Whichever machine is randomly selected by the client to execute this action becomes the permanent custodian of that session's state. The custodian stores the session state locally and returns a session ID to the client. The session ID has the custodian server's ID embedded in it.

If later operations that are part of the same session arrive at a different Asirra service machine, the request is forwarded to the session custodian by finding its identity embedded in the session ID. (The client is not trusted; a forged session ID will simply fail to find any expected state on the incorrectly indicated custodian server.) Our forwarding scheme ensures that at most two machines

Challenges Solved	Users Passed, No PCA	Users Passed, With PCA
1 (≈15 sec)	83.4%	83.4%
2 (≈30 sec)	97.2%	99.6%
3 (≈45 sec)	99.5%	99.96%

Table 1: Expected user population that will pass Asirra after 1, 2, and 3 challenges, with and without PCA (§4.1). Assumes a 12-image challenge, 98.5% classification accuracy for an individual image (§3.1.3), and 15 seconds required per challenge (§3.1.2).

Image Classifier Accuracy	Bot Success Rate		
	No PCA	With PCA	PCA + Tokens
50% (random guessing)	1/4,096	1/3,952	1/5.2 million
60% (known techniques)	1/459	1/404	1/54 thousand
70% (major AI advance)	1/72	1/54	1/990

Table 2: Expected success rates of bots with three hypothetical image classifiers. Assumes a 12-image challenge. The effects of PCA (§4.1) and Token Buckets (§4.2) with *TB-Refill* of 3 are also considered. A detailed threat analysis can be found in §3.2.

are ever involved in servicing a single request: the machine which receives the request from the client, and the machine that owns the session state and receives the sub-request. The Asirra service is therefore readily scalable; the overhead of parallelization will never be more than 2x regardless of the total size of the farm.

In practice, we have observed lower overhead; request forwarding is not the common case. Many combinations of browser and operating system continue to use the same IP address for a given domain name for the duration of a browser session. This may be an aspect of many browsers' "DNS Pinning" policy meant to reduce cross-site scripting attacks.

The web service is about 700 lines of Python code. It is currently deployed on Amazon's EC2 [10] compute cloud platform, making it easy to add resources as the service's popularity increases.

6. SUMMARY AND CONCLUSIONS

In this paper, we presented Asirra, a CAPTCHA that asks users to identify cats out of a set of 12 photographs of both cats and dogs. Our image database is provided by a novel, mutually beneficial partnership with Petfinder.com, which has a database of over three million images of pets looking for new homes. In exchange for the use of these images, Asirra displays an "adopt me" link beneath each one, promoting Petfinder's primary mission of finding homes for homeless animals. (Security implications of Adopt-Me are discussed in Section 3.2.1.)

Asirra is attractive because humans can solve it quickly and accurately (Table 1), but it provides significant protection from bots (Table 2). Asirra provides a social benefit: improving animal welfare. And, anecdotally, users report that Asirra is enjoyable: Spending a few moments looking at cute kittens is generally preferred to squinting at deformed letters.

Asirra is a free web service, available at `www.asirra.com`.

Our contributions in this paper also include two techniques that we use in Asirra, but can be added to virtually any CAPTCHA. The Partial Credit Algorithm (§4.1) significantly improves the pass rates of humans while only marginally improving the yield of bots. Conversely, our use of CAPTCHA Token Buckets (§4.2) significantly decreases the yield of brute-force bots while having a minimal effect on humans.

CAPTCHA design seems to be both an art and a science. They will never be strong in the sense of a cryptosystem or a proof; by definition, they can be broken with nothing more than a few moments of casual human effort. In what may be an endless arms race, perhaps the best we can do is ensure the latest weapon *du jour* is fun for users. In this goal, at least, we hope Asirra succeeds.

7. REFERENCES

[1] Frozen Bear. hotcaptcha. http://www.hotcaptcha.com.

[2] Kumar Chellapilla, Kevin Larson, Patrice Simard, and Mary Czerwinski. Designing human friendly human interaction proofs (HIPs). In *Proceedings of ACM CHI 2005 Conference on Human Factors in Computing Systems*, volume 1 of *Email and security*, pages 711–720, 2005.

[3] Monica Chew and J.D. Tygar. Image recognition CAPTCHAs. In *Proceedings of the 7th International Information Security Conference (ISC 2004)*, pages 268–279. Springer, September 2004.

[4] Mark Everingham, Andrew Zisserman, Chris Williams, and Luc Van Gool. The PASCAL visual object classes challenge 2006 (VOC2006) results. Technical report, University of Oxford, 2006.

[5] Ralph Gross, Jianbo Shi, and Jeff Cohn. Quo vadis Face Recognition? Technical Report CMU-RI-TR-01-17, Carnegie Mellon University Robotics Institute, June 2001.

[6] Google Images. http://images.google.com.

[7] Eight Days Inc. Hot or not. http://www.hotornot.com.

[8] Greg Mori and Jitendra Malik. Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In *Conference on Computer Vision and Pattern Recognition (CVPR '03)*, pages 134–144. IEEE Computer Society, 2003.

[9] Jared Saul. Petfinder. http://www.petfinder.com.

[10] Amazon Web Services. Ec2 scalable compute cloud. http://aws.amazon.com/ec2.

[11] Patrice Simard, David Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *International Conference on Document Analysis and Recognition*, pages 958–962. IEEE Computer Society, 2003.

[12] Digg.com user DoubtfulSalmon. http://tinyurl.com/2stwu3, April 2006.

[13] L. von Ahn, M. Blum, N.J. Hopper, and J. Langford. The CAPTCHA web page. http://www.captcha.net.

[14] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. CAPTCHA: Using hard AI problems for security. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 294–311. Springer, 2003.

[15] Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In Elizabeth Dykstra-Erickson and Manfred Tscheligi, editors, *Proceedings of the 2004 Conference on Human Factors in Computing Systems, CHI 2004, Vienna, Austria, April 24 - 29, 2004*, pages 319–326. ACM, 2004.

[16] Oli Warner. Kittenauth. http://www.thepcspy.com/kittenauth.

[17] WikiMedia Foundation. http://commons.wikimedia.org.

[18] Wen-Yi Zhao, Rama Chellappa, P. J. Phillips, and Azriel Rosenfeld. Face recognition: A literature survey. *ACM Comput. Surv*, 35(4):399–458, 2003.