

The Robustness of CAPTCHAs

Jeff Yan

School of Computing Science
Newcastle University, UK
(Joint work with **Ahmad El Ahmad**)



Roadmap

- Introduction
- How did we break CAPTCHAs?
- Why did we break CAPTCHAs?
- Does CAPTCHA have a future?

CAPTCHA is almost everywhere ...



Why was it invented?



- Often desirable to make sure a real human is behind a computer ...

CAPTCHA: what is it?

- An automated Turing test
 - that computers cannot pass, but human should
 - Often making use of a hard, open AI problem
- The most widely deployed: **text CAPTCHAs**
 - users are asked to solve a text recognition task
 - Recognition of distorted texts

CAPTCHA: what is it?

- History
 - First proposed by Moni Noar [1996]
 - Early prototypes by Alta Vista [1997]
 - Made widely known by a CMU team (Luis von Ahn et al since 2000)

Security of CAPTCHAs

- Protocol-level attacks:
 - (porn site based) oracle attack
 - outsourcing attack: cheap human solvers
- **Robustness: strength of resisting automated solvers**
 - Common design goal: bot's success rate < .01%

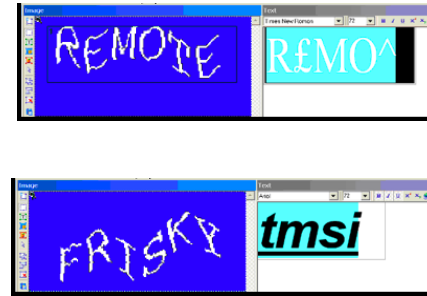
How did we break CAPTCHAs?

Captchaservice.org [HIP'06]



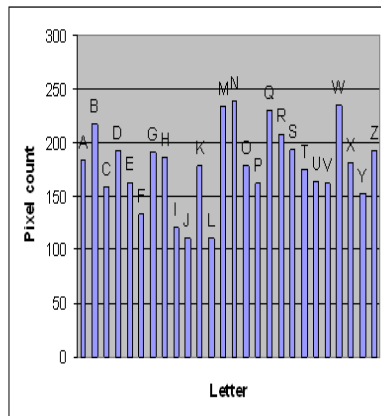
- Distortion: random shearing (both vertically and horizontally)

OCR (FineReader) results

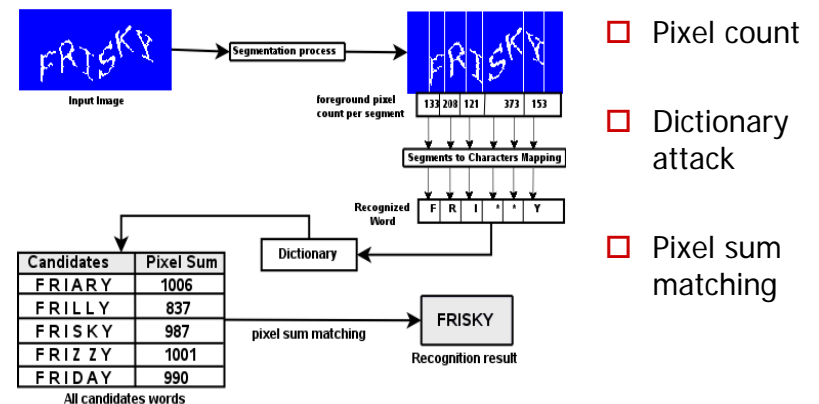


Character <-> pixel count

Letter	Pixel Count	Letter	Pixel Count
A	183	N	239
B	217	O	178
C	159	P	162
D	192	Q	229
E	163	R	208
F	133	S	194
G	190	T	175
H	186	U	164
I	121	V	162
J	111	W	234
K	178	X	181
L	111	Y	153
M	233	Z	193



Our simple attack



- Pixel count
- Dictionary attack
- Pixel sum matching

Results

- ❑ Success rate: 92~94%
- ❑ Attack speed: ~50 ms per challenge
- ❑ Many more schemes vulnerable to the **pixel count attack!**

Microsoft CAPTCHA



- ❑ Designed by the MSR team
- ❑ deployed since 2002 for Hotmail, MSN, Live

Design principle 1: easy to recognise

Characters under typical distortions	Recognition rate
	~100%
	96+%
	100%
	98%
	~100%
	95+%

Chellapilla et al, [CEAS'05]:

Computers do better than humans!

Design principle 2: hard to segment



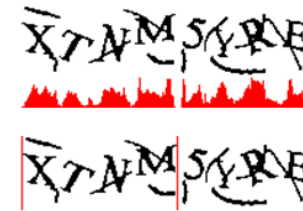
- ❑ Segmentation: to **locate where each letter is**
 - **Computationally expensive**
 - ❑ every position must be tested for a potential candidate
 - **Space of input is huge**
 - ❑ as it includes all non-valid characters

Microsoft CAPTCHA



- Main trick: random arcs of different thicknesses
- arcs are good candidates for false characters

Key trick 1: vertical segmentation



- image → histogram → chunks

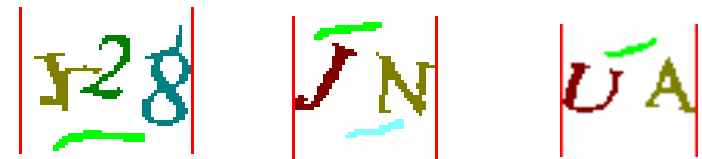
Key trick 2: color filling segmentation

- Chunk by chunk



- The number of color = the number of objects in each chunk

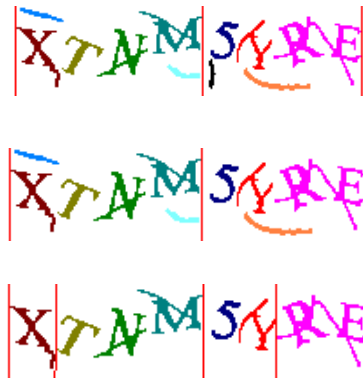
Key trick 3: arc removal



- Chunk by chunk
- **Relative position analysis**
 - Characters juxtapose horizontally, but never vertically

Key trick 3

- Arc removal sample
 - P.c. < 50, go
 - Relative position checking
 - Updating vertical histogram



Key trick 4

□ Locating connected characters

- MS helped us: 8 characters all the time!
- A simple game of throwing 8 balls into c bins ($c=1, \dots, 8$)

□ example: we know there are 4 chunks



Key trick 4



- All possibilities for distributing 8 chars (balls) into 4 chunks (bins) – bin order ignored
 - | @@ | @@ | @@ | @@ |
 - | @@ @ | @@ | @@ | @ |
 - | @@ @@ | @@ | @ | @ |
 - | @@ @ | @@@ | @ | @ |
 - | @@ @@@ | @ | @ | @ |

Key trick 4



- All possibilities for distributing 8 chars (balls) into 4 chunks (bins) – bin order ignored
 - | @@ | @@ | @@ | @@ | X (one chunk has 3 objects)
 - | @@ @ | @@ | @@ | @ |
 - | @@ @@ | @@ | @ | @ | X (three wide chunks)
 - | @@ @ | @@@ | @ | @ | X (three wide chunks)
 - | @@ @@@ | @ | @ | @ | X (three wide chunks)

Key trick 4



- All possibilities for distributing 8 chars (balls) into 4 chunks (bins) – bin order ignored
 - | @@ | @@ | @@ | @@ | X (one chunk has 3 objects)
 - | @@ @ | @@ | @@ | @ |
 - | @@ @@ | @@ | @ | @ | X (three wide chunks)
 - | @@ @ | @@@ | @ | @ | X (three wide chunks)
 - | @@ @@@ | @ | @ | @ | X (three wide chunks)
- Most possible answer: option 2
 - Wider chunk #4 has 2 chars (8 – 1 – 3 – 2)

Last step

- “Even cut”



Results

- Segmentation success: 92%
- Speed: ~80ms/s
 - PC with 1.86 GHz Intel Core 2 CPU and 2 GB RAM
- Overall success: 61% ($\approx .92 * .95^8$).
 - MS claim: “extremely difficult and expensive for computers to solve”
- Applicability: yahoo, google

Why did we break CAPTCHAs?

- Identify and fix vulnerable designs
- Inform better future design
- Bad guys were doing the cracking for profit already, good to stay a step ahead of them!

Do CAPTCHAs have a future?

- Huge gap of perceptual capabilities between human and machine
- Not invented to differentiate different people, but sweatshop can be mitigated using other means
- Better text CAPTCHAs and new types will emerge
 - Game as captcha: “Magic Bullet”, “Cat or Dog”

MAAWG, Amsterdam (June 11, 2009)

(29)

Take-home message

- Although not a panacea, CAPTCHA is useful, and better design will emerge.
- The devil is in the details. **Careful robustness evaluation is needed before deployment.**
 - We have both expertise and tools to help

MAAWG, Amsterdam (June 11, 2009)

(30)

“Secure and usable CAPTCHAs”

Project website:

<http://homepages.cs.ncl.ac.uk/jeff.yan/>

Thank You!

Q?

Jeff.Yan@ncl.ac.uk