

POW! Webmail

Effective Disincentives Against Spam

Wu-chang Feng (wuchang@cs.pdx.edu)

Ed Kaiser (edward.l.r.kaiser@gmail.com)



Supported by:



Motivation

- Spammers targeting webmail systems to send spam
 - Creating new accounts on free webmail sites (GMail, Hotmail, Yahoo! Mail)
 - Hijacking legitimate accounts via spear phishing attacks to send spam (Horde/IMP, SquirrelMail)
 - Webmail spam 5% of all spam sent in 2008 [IronPort08]

CAPTCHAs to the rescue?

- Use a hard AI problem for security
 - Force users to solve a problem that is hard for a computer, but easy for a human
 - Does not require special client software
- Widely used
 - Google / Gmail
 - Microsoft Live / Passport / Hotmail
 - Yahoo!

A handwritten word in blue ink, appearing to be "thalia".A handwritten alphanumeric string in blue ink, appearing to be "JH U D T 5 N 8".A handwritten alphanumeric string in black ink, appearing to be "3st5wk".

CAPTCHA Problem #1

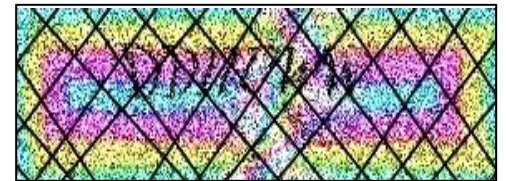
- User experience is frustrating, annoying, and aesthetically unappealing
 - Inaccessible to many
 - Not suitable for frequent transactions



Blogger



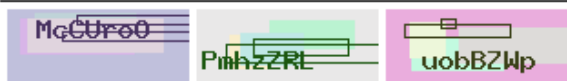





Facebook



TicketMaster

CAPTCHA Problem #2

- Adversaries solving the hard OCR problem
 - Strong financial incentive to break them
 - Yahoo!, Windows Live, Google all broken in early 2008
 - PWNtcha CAPTCHA solving library

Origin	Samples	Efficiency
linuxfr.org		100%
LiveJournal		99%
Paypal		88%
phpBB		97%
SCode and derivatives		100%
Slashdot		89%

CAPTCHA Problem #3

- Economics broken
 - Fixed workload priced at 10 seconds of human time
 - Outsourced for under 1¢ per CAPTCHA

GEI a FREELANCER.com

Status:	Closed
Average bid:	\$ 35
Bid count:	14
Description:	<p>I need a big team for this project, 10+ people at the very least. The job requires entering captchas for a social networking site to create accounts. I am paying \$3 US for 1,000 captchas. With my program, 1,000 captchas (1,000 accounts) can be entered/made within an hour easily if you are proficient at typing. I will require delivery of 20 THOUSAND accounts or more PER DAY. Please understand that this is a big undertaking, only serious bidders.</p> <p>Report Violation</p>
Job Type:	◆ Data Entry

- CAPTCHA pricing does not work when value of what is being protected is more than 1¢

Outsourcing Example

HOME PAGE	MY TIMES	TODAY'S PAPER	VIDEO	MOST POPULAR	TIMES TOPICS
-----------	----------	---------------	-------	--------------	--------------

The New York Times

Business

WORLD	U.S.	N.Y. / REGION	BUSINESS	TECHNOLOGY	SCIENCE	HEALTH	SPORTS	OPINION
-------	------	---------------	----------	------------	---------	--------	--------	---------

MEDIA & ADVERTISING WORLD BUSINESS SMALL BUSINESS YOUR MONEY DEALBOOK MARKETS RESEARCH

DIGITAL DOMAIN

Hannah Montana Tickets on Sale! Oops, They're Gone

By RANDALL STROSS
Published: December 16, 2007

HANNAH MONTANA has made 2007 a very bright year for various business interests, but especially for StubHub, the online ticket exchange site.

- E-MAIL
- PRINT
- SINGLE PAGE
- REPRINTS

RMG answered Ticketmaster's Captchas — the visual puzzles of distorted letters that a customer must type before buying tickets— not with character recognition software, he said, but with humans: “We pay guys in India \$2 an hour to type the answers.”

Need a variable workload to price out adversaries!

Proof-of-Work (PoW)

- a.k.a. Client Puzzles
 - Alternative to CAPTCHA
 - Clients solve a cryptographic puzzle to get access
- Addresses CAPTCHA problems
 - No user interface issues
 - Adversary must break a cryptographic problem
 - Adjustable difficulty that can be arbitrarily increased
- But....
 - Landscape littered with unused PoW schemes!
 - Hashcash, TLS puzzles, TCP puzzles, IP puzzles, Public work

Problems with Proof-of-Work

- Ineffective puzzle algorithms
- Inability to protect legitimate clients
- Deployment

Ineffective puzzle algorithms

- Ideal puzzle algorithm
 - Fast issuing
 - Fast verification
 - Fine-granularity work given to client
 - Deterministic work given to client
 - Non-parallelizable
- Current puzzle schemes based on breaking a weakened cryptographic problem
 - Hash-reversal
 - Time-lock

Hash-reversal puzzles

- Brute-force search on the input space of a cryptographic hash (H) to generate a specific output

- Hash-reversal [Juels99]

$$H(\text{input}) = P$$

- Give client P and high-order bits of input, client solves for input
- Number of bits given determines difficulty

- Hint-based hash-reversal [Feng05]

$$H(\text{input}) = P, \text{Hint} = \text{input} - u(0, D), D = \text{difficulty}$$

- u = uniform distribution over $0, D$
- Given P and Hint, client solves for input

- Targeted hash-reversal [Kaiser08]

- N = random number generated by server, D = difficulty
- Client finds any input such that

$$H(N \parallel \text{input}) = 0 \pmod{D}$$

Analyzing hash-reversal puzzles

- The good
 - Fast issuing
 - Single hash or random number to issue
 - Fast verification
 - Single hash to verify
- The bad
 - Can be coarse-grained
 - Powers of two [Juels99]
 - Non-deterministic
 - Probabilistic run-times
 - Parallelizable
 - Hash searches easily parallelized across machines

Time-lock puzzles

- Based on repeated modular squaring [Rivest96]
 - Server generates
 - $n = p * q$ where p and q are two large primes
 - Random number a and Difficulty t
 - Server sends client a, n, t
 - Client calculates $A = a^{2^t} \bmod n$
 - Server validates answer via short-cut
 - $\phi = (p-1) * (q-1)$
 - $r = 2^t \bmod \phi$
 - $A = a^r \bmod n$

Analyzing time-lock puzzles

- The bad
 - Slow issuing
 - Generating large primes per puzzle prohibitively expensive
 - Slow verification
 - Must keep track of all puzzles a, n, t to subsequently verify
- The good
 - Fine-grained
 - Granularity down to a single modular squaring operation
 - Deterministic
 - Exact number of squarings given
 - Non-parallelizable
 - Repeated squaring not easily parallelized

kaPoW's modified time-lock puzzle

- Modify algorithm to efficiently issue and verify
 - Re-use modulus n across clients
 - Generate a as a hash of client request parameters and server nonce to make server constant state and prevent replay
$$a = H(K \parallel f_c)$$
$$K = \text{server random number}$$
$$f_c = \text{client request parameters (URL, POST data, client IP, etc.)}$$
 - n and K must be refreshed periodically to prevent attacks

Comparison

PUZZLE TYPE	ISSUING METHOD	VERIFICATION METHOD
Time-Lock [31]	RSA key generation	modular exponentiation
Hash-Reversal [19]	cryptographic hash	cryptographic hash
Hint-Based Hash-Reversal [13]	cryptographic hash	cryptographic hash
Targeted Hash-Reversal [12]	minimal	cryptographic hash
kaPoW Modified Time-Lock	cryptographic hash	modular exponentiation

PUZZLE TYPE	NON-PARALLELIZABLE	DETERMINISTIC	WORST-CASE GRANULARITY
Time-Lock [31]	Yes	Yes	1 modular squaring
Hash-Reversal [19]	No	No	2^{n-1} cryptographic hashes
Hint-Based Hash-Reversal [13]	No	No	1 cryptographic hash
Targeted Hash-Reversal [12]	No	No	1 cryptographic hash
kaPoW Modified Time-Lock	Yes	Yes	1 modular squaring

Problems with Proof-of-Work

- Ineffective puzzle algorithms
- Inability to protect legitimate clients
- Deployment

Inability to protect legitimate clients

- Resource disparity gives adversary a huge advantage
 - Statically priced proof-of-work cannot stop adversaries with massive resources [Laurie04]
 - Must control difficulty across users to achieve adequate separation of adversaries from legitimate clients [Liu06]

Prior PoW systems

- Simplistic difficulty settings easily bypassed
 - Static, uniform difficulties
 - [Juels99], [Aura00], [Dean01], [Goodman04]
 - Dynamic, uniform difficulties
 - [Back02], [Wang03]
 - Usage-based difficulty
 - [Kaiser08]
 - Content-based difficulty
 - [Zhong05]

kaPoW's defense-in-depth approach

- Set difficulty using comprehensive metrics
 - Time
 - Time of day, time since last e-mail transmission, time since account creation
 - Usage
 - Number of messages recently sent, number of recipients in message
 - Location
 - Geographic distance from server, distance from prior transmissions
 - Reputation
 - Presence on distributed IP address blacklists
 - Content
 - Spam score of message, reputation of embedded URLs
 - Social network
 - Whether recipient has sent messages to sender or is in sender's social network

Problems with Proof-of-Work

- Ineffective puzzle algorithms
- Inability to protect legitimate clients
- **Deployment**

Deployment

- Prior systems require software modifications
- kaPoW implemented with PHP and Javascript
 - No protocol changes
 - No web browser changes
 - No web server changes
- PHP script
 - Analyzes client and content metrics to determine difficulty
 - Dynamically embeds PoW challenges with client-specific difficulty into form submission and URL tags
 - Attaches 9KB JavaScript solver for client to run
 - Verifies subsequent solutions
- JavaScript solver
 - Client browser runs solver to calculate answers
 - Attaches answers to subsequent requests

Prototype

- <http://kapow.cs.pdx.edu/mail>

kaPoW webmail - Mozilla Firefox

File Edit View History Bookmarks Tools Help

kaPoW webmail

From: Spammer: Buy Viagra!
VIAGRA VIAGRA VIAGRA

Distance from Portland, OR: 0 miles

Your IP:

Your Name:

To:

Subject:

Body:

Good sender: Old account, Few messages sent recently, Recipient has e-mailed sender before.

Suspicious sender: New account, Many messages sent recently, Recipient has not e-mailed sender before.

It is expected to take your browser 1048576 units of work to submit it.

Done

Prototype

- <http://kapow.cs.pdx.edu/mail>
 - Modulus generation via OpenSSL
 - Content analysis via SpamAssassin, URIBL, SURBL
 - Reputation lookup via Spamhaus, SpamCop, and Project Honeypot DNS blacklists
 - Geographic location resolution via MaxMind's GeoIP
 - Leverages BigInteger library for Javascript solver

Evaluating modified time-lock

- Modulus generation

Key size (bits)	Generation time (seconds)
400	0.165
800	1.13
1200	3.90
1600	9.88

- Per-request generation prohibitive
- Periodic generation feasible

- Puzzle issuing

- 5.32 μ s

- Puzzle verification

Key size (bits)	Verification time (milliseconds)
400	0.184
800	1.16
1200	2.35
1600	4.01

Evaluating adaptive difficulties

- No access to production webmail service so...
- Simulate a hijacked account on a university webmail service with a simplistic difficulty algorithm
 - $\text{score} = S_T + S_U + S_L + S_R + S_C + S_S$
 - $S_T =$ Time component
 - 1 if during an 8-hour “active” period, 0 otherwise
 - $S_U =$ Usage component
 - 1 if user sent a message in past 5 minutes, 0 otherwise
 - $S_L =$ Location component
 - 1 if user is within 500 miles, 0 otherwise
 - $S_R =$ Reputation component
 - 1 if client IP is on a blacklist, 0 otherwise
 - $S_C =$ Content component
 - 1 if SpamAssassin considers message spam, 0 otherwise
 - $S_S =$ Social network component
 - 1 if recipient is in user’s address book, 0 otherwise
 - $t = 20 * \text{score}^6$

Simulation experiments

- Spam bot with a hijacked user account
 - Greedy sender sending messages to random e-mail addresses continuously throughout the day
 - $S_C=1$ for 80% of messages
 - $S_S=1$ for all messages
- Legitimate sender using the same account
 - Sends one message during the middle of the day at a local location from a “good” IP address to a friend
 - $S_T=0$
 - $S_L=0$
 - $S_R=0$
 - $S_C=0$
 - $S_S=0$
- Full-day simulation across 1000 trials

Results

- Legitimate sender minimally impacted
 - Worst-case greedy bot in local location on a “good” IP address sends 160 messages per day on average

Bot type	Average messages sent by bot during the day	Average message delay for legitimate client
Local bot, good IP	159.7 ± 5.6	0.400 ± 0.000
Local bot, bad IP	30.3 ± 2.2	0.116 ± 0.040
Remote bot, good IP	30.4 ± 2.3	0.104 ± 0.041
Remote bot, bad IP	8.4 ± 1.2	0.000 ± 0.000

Conclusion

- kaPoW Webmail addresses key problems in proof-of-work systems
 - Ineffective puzzle algorithms
 - Modified time-lock algorithm
 - Inability to protect legitimate clients
 - Defense-in-depth approach for determining difficulties
 - Deployment
 - PHP, Javascript implementation requiring no changes to browser or web server
- Future work
 - More effective difficulty algorithms
 - Evaluation on a deployed webmail service
 - Applying techniques to other web applications
 - Commercialization

Questions?

<http://kapow.cs.pdx.edu>

<http://kapow.cs.pdx.edu/mail>

Extra slides

Issuing Challenges

- Example HTML content on disk

```
<INPUT TYPE='button' VALUE='Submit'  
      AA=13F75ABE24C  
      NN=A2972AACCC37BE6F6BF5CA01282B  
      TT=1048576 ONCLICK='Solve(this);' >
```

- Javascript solver

```
while ( tag.cnt < tag.tt ) {  
    squareMod_(tag.a,tag.n);  
    tag.cnt += 1;  
}  
// Update the tag to indicate success and POST the form.  
tag.A = bigInt2str(tag.a,16);  
document.getElementById("answer").value = tag.A;  
document.getElementById("do").value = "submit";  
document.forms[1].submit();
```

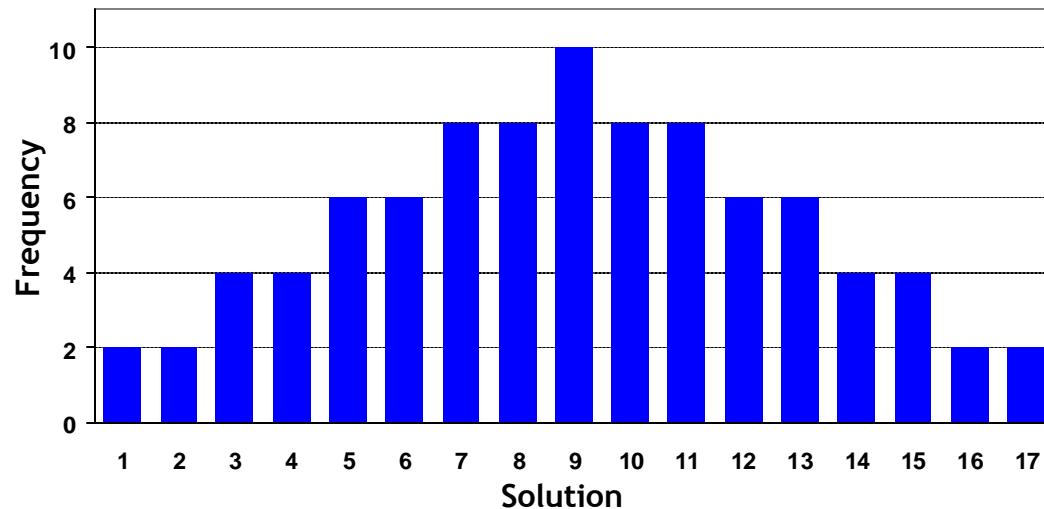
Sometimes Not Even A Hard Problem

- Poor homebrew CAPTCHAs
 - eg. Scranton Times Tribune

“Sum the two flashing numbers:



- Small biased solution space



- Trivial character isolation



... Scripted in 15 Minutes

- Test pixel color at fixed locations

```
wget $1 -O temp.gif
giftopnm -image=all temp.gif > temp.ppm
./solve
```

```
int main(int argc, char *argv[]) {
    FILE *fp = fopen("temp.ppm", "rb");
    unsigned char buf[81028];
    fread(buf, 81028, 1, fp);
    int answer = 0;
    #define t(x,v) if ((row[((x)*3)  ] < 240) || \
                    (row[((x)*3)+1] < 240) || \
                    (row[((x)*3)+2] < 240)  ) answer += v
    unsigned char *row = &(buf[14 + (30*270*3)]);
    t( 44, 1); t( 72, 2); t(100, 3); t(123, 4); t(149, 5);
    t(162, 6); t(193, 7); t(213, 8); t(243, 9);
    row = &(buf[28 + (80*270*3)]);
    t( 44, 1); t( 72, 2); t(100, 3); t(123, 4); t(149, 5);
    t(162, 6); t(193, 7); t(213, 8); t(243, 9);
    printf("CAPTCHA answer: %d\n", answer);
}
```

Addressing economics

- How do you construct a pricing system that works?
 - What is the cost of unattended (idle) CPU cycles?
 - Can costs be controlled to create sufficient disincentives for botnets of 20,000 idle machines?
 - How much is it worth to keep bots hidden?
 - How do you cope with price limits to legitimate users?