Fighting Spam May Be Easier Than You Think

Cynthia Dwork
Microsoft Research  SVC
Why?

- Huge problem
  - Industry: costs in worker attention, infrastructure
  - Individuals: increased ISP fees
  - Hotmail: huge storage costs, 65-85%
  - FTC: fraud, confidence crimes
  - Ruining e-mail, devaluing the Internet
Computational Approach [DN’92]

- If I don’t know you:
  Prove you spent ten seconds CPU time, just for me, and just for this message

- User Experience:
  Automatically and in the background
  Checking proof extremely easy

- All unsolicited mail treated equally
Principal Techniques

- Filtering
  - Everyone: text-based
  - Brightmail: decoys; rules updates
  - Microsoft Research: (seeded) trainable filters
  - SpamCop, Osirusoft, etc: IP addresses, proxies, ...

- Make Sender Pay
  - Computation [Dwork-Naor’92; Back’97]
  - Human Attention [Naor’96, DEC patent]
  - Money [eg, Gates’96, Hayes, McCurley]
Outline

- The Computational Approach
  - Overview; economic considerations
  - Burrows’ suggestion: memory-bound functions
- Turing Tests
- Architectures
  - Point-to-Point
  - Web-based mail
- Deeper Discussion of Memory-Bound Functions
  - Nascent Proposal (joint with A. Goldberg and M. Naor)
Economics

- \( (80,000 \text{ s/day}) / (10\text{s/message}) = 8,000 \text{ msgs/day} \)
- Hotmail’s billion daily spams:
  - 125,000 CPUs
  - Up front capital cost just for HM: circa $150,000,000
- The spammers can’t afford it.
Who Are the Spammers?

"Most of the spammers are not wealthy people," said Stephen Kline, a lawyer for the New York State attorney general's office.
Who Are the Spammers?

- Spamhaus ROKSO 100/90%
- F. Krueger, SMN: circa 300 people total; very top few spammers make a few million/year; resources spent “dealing with” ISPs, setting up shell companies
- Big Player: eUniverse (Yahoo! lists annual earnings of $6.1 Mil), has mailing list of 100,000,000 names, Mosaic Data Systems has list of 60,000,000
Cryptographic Puzzles

- Hard to compute; $f(m,S,R,t)$ can’t be amortized
  lots of work for the sender

- Easy to check “$z = f(m,S,R,t)$”
  little work for receiver

- Parameterized to scale with Moore's Law
  easy to exponentially increase computational cost, while
  barely increasing checking cost

- Can be based on (carefully) weakened signature
  schemes, hash collisions
Burrows’ Suggestion

- CPU speeds vary widely across machines, but memory latencies vary much less (10+ vs 4)
- So: design a puzzle leading to a large number of cache misses
Social Issues

- Who chooses f?
  One global f? Who sets the price?
  Autonomously chosen f’s?

- How is f distributed (ultimately)?
  Global f built into all mail clients? (1-pass)
  Directory? Query-Response? (3-pass)
Computation: Technical Issues

- Distribution Lists (!)
- Awkward Introductory Period
  - Old versions of mail programs; bounces
- Very Slow/Small-Memory Machines
- Cache Thrashing (memory-bound)
- The Subverters
Turing Tests: Payment in Kind [N’96]

- CAPTCHAs (Completely Automated Public T. test for telling Computers and Humans Apart)
  - Defeat automated account generation
  - 5-10% drop in subscription rate
  - Teams of conjectured-humans (8-hour shifts)

- Yes: Distorted images of simple word
- No: ``Find 3 words in image with multiple overlapping images of words"

- Others: subject classification, audio
- M. Blum: people have done preprocessing
Social and Technical Issues

Social (especially in enterprise setting)
- ADA, S.508 (blind, dyslexic)
- Not ``professional"
- Productivity cost: context switch (may be mitigated in architecture supporting pre-computation)
- Wrong direction: we should offload crap onto machines

Technical
- No theory; if broken, can’t scale.
- Idrive, AltaVista, broken [J]
Turing Test Economic Issues

- Human time in poor country has roughly same cost as computer time (even if 8/5 wk)
  - Also need computer, but it can be slow, cheap
  - 10s same ballpark as some Turing tests

- Suppose we’re wrong about cost, and really the price should be 1 cent
  - 1-cent challenge costs 2 minutes. No one not being paid would dream of submitting to this.
Cycle Stealing

- Stealing cycles from humans: Pornography companies require random users to solve a CAPTCHA before seeing the next image [vABL]
- Worse for computational challenges
- Economics: lots of currency means currency is devalued. Prices go up.
- There are lots of cycles, but anyone can buy them. Can go into business brokering cycles.
Point-to-Point Architecture

(Ideal Message Flow)
- Single-pass “send-and-forget”
- Can augment with amanuensis to handle slow machines
- Can add post office / pricing authority to handle money payments
Here to There (and There’) 

- Three e-mail messages
- R’s mail client caches m, h(m), S, R, t
- Bounce (viral marketing opportunity)
  - html attachment with Java Script for f (make it an intrinsic)
  - contains parameters for f (h(m), S, R, t)
  - clicking on link causes computation, sending e-mail
  - (optional) link for download of client software
Remark: Web-Based Mail

- Computation done by client (applet or JavaScript)
- Verification and safelist checking done by server
Memory-Bound Functions [ABMW’02]

- Devilish model!
  - assume cache size (of spammer’s big machine) is half the size of memory (of weakest sending machine)
  - must avoid exploitation of locality, cache lines (factors of 64 or even 16 matter)
  - watch out for low-space attacks in crypto literature
Incompressibility and RC4 [DGN’02]

- Big ($2^{22}$) Fixed-Forever Truly Random T
- Model walk on prg of RC4 stream cipher

Intuition:

Use the pr bit stream (seeded by message) to choose entries of T to access.
These, in turn, affect the stream (defending against pre-processing to order T so as to exploit locality).
RC4 (Partial Description)

Initialize $A$ to pseudo-random permutation of $\{1, 2, \ldots, 256\}$ using a seed $K$

- Initialize Indices:
  
  $i = 0$; $j = 0$

- PR generation loop:
  
  $i = i + 1$
  
  $j = j + A[i]$
  
  Swap ($A[i]$, $A[j]$);
  
DGN: Initializing A and Basic Step

- Fixed-Forever truly random A of size 256; 32-bit words
- Given m and trial number i, compute strong cryptographic 256-word mask M.
- Set $A = M \odot A$.
- $c = A \mod 2^{22}$

- Initialize Indices: $i=0; j=0$
- Walk:
  - $i = i + 1$
  - $j = j + A[i]$
  - $A[i] = A[i] \odot T[c]$
  - Swap $(A[i], A[j])$
Side by Side

Generate pseudo-random permutation of \{1, 2, \ldots, 256\} using a seed K

- Fixed-Forever truly random A of size 256; 32-bit words
- Given m and trial number i, compute strong cryptographic 256-word mask M.
- Set \( A = M \odot A \).
- \( c = A \mod 2^{22} \)
Initialize Indices:
  \( i = 0; \ j = 0 \)

PR generation loop:
  \( i = i + 1 \)
  \( j = j + A[i] \)

Swap \((A[i], A[j])\)

Initialize Indices:
  i=0; j=0

Walk:
  \( i = i + 1 \)
  \( j = j + A[i] \)
  \( A[i] = A[i] \odot T[c] \)
Swap \((A[i], A[j])\)
Full DGN

- **E**: factor by which computation cost exceeds verification
- **L**: length of walk
- **Trial i succeeds**: if hash of A after ith walk ends in \(1 + \log_2 E\) zeroes (expected number of trials is E).
- **Verification**: trace the walk (L memory accesses)
Parameters

- Want $E L t = P$, where
  - $L =$ path length
  - $P =$ target real time cost of proof of effort = 10s
  - $t =$ memory latency = 0.2 nanoseconds
- Choose $E < \frac{|\text{Cache}|}{|\text{Cache Line}|}$
- Reasonable choices:
  - $E = 32,000$, $L = 625$
Summary

- Discussed computational approach, Turing tests, economics, cycle-stealing
- Briefly mentioned two architectures
- Examined difficulties of constructing memory-bound pricing functions and proposed a new one designed to avoid these difficulties (no proofs yet)
Future Work and Open Questions

- Implement and test Outlook, Pine plug-ins (at Stanford); add signatures
- Further study of DGN algorithm
- Distribution Lists
- Good MB functions with short descriptions (will subset product work)?