Client Puzzles
A Cryptographic Defense Against Connection Depletion Attacks

Most of slides come from
Ari Juels and John Brainard
RSA Laboratories

The Problem

How to take down a restaurant

Restauranteur
Saboteur
Saboteur vs. Restaurateur

Table for four at 8 o'clock. Name of Mr. Smith.

O.K., Mr. Smith

Saboteur

Restaurateur

No More Tables!

An example: TCP SYN flooding

"TCP connection, please."

"O.K. Please send ack."

Buffer

O.K., Mr. Smith
TCP SYN flooding has been deployed in the real world
- Panix, mid-Sept. 1996
- Others
Similar attacks may be mounted against e-mail, SSL, etc.

Some defenses against connection depletion

Problem: Legitimate clients must keep retrying

Throw away requests
IP Tracing (or Syncookies)

Hi. My name is 10.100.16.126.

Problems:
- Can be evaded, particularly on, e.g., Ethernet
- Does not allow for proxies, anonymity

Digital signatures

Problems:
- Requires carefully regulated PKI
- Does not allow for anonymity

Connection timeout

Problem: Hard to achieve balance between security and latency demands
Our solution: *client puzzles*

Intuition

Suppose:
- A puzzle takes an hour to solve
- There are 40 tables in the restaurant
- Reserve at most one day in advance

A legitimate patron can easily reserve a table
Intuition

Would-be saboteur has too many puzzles to solve

The client puzzle protocol

Client → Service request \( M \) → Server

O.K.

Buffer

What does a puzzle look like?
Puzzle basis: partial hash inversion

Puzzle basis: (Cont’d)
- Only way to solve puzzle \((X', Y)\) is brute force method. (hash function is not invertible)
- Expected number of steps (hash) to solve puzzle: \(2^k / 2 = 2^{k-1}\)

Puzzle construction
Puzzle construction

Server computes:
- Secret $S$
- Time $T$
- Request $M$

hash

pre-image

hash

image $Y$

Sub-puzzle

- Construct a puzzle consists of $m$ $k$-bit-hard sub-puzzles.
- Increase the difficulty of guessing attacks.
- Expected number of steps to solve: $m \times 2^k$.

Puzzle properties

- Puzzles are stateless
- Puzzles are easy to verify
- Hardness of puzzles can be carefully controlled
- Puzzles use standard cryptographic primitives
Client puzzle protocol (normal)

Client puzzle protocol (under attack)

Where to use client puzzles?
Some pros

Avoids many flaws in other solutions, e.g.:
- Allows for anonymous connections
- Does not require PKI
- Does not require retries -- even under heavy attack

Practical application

- Can use client-puzzles without special-purpose software
  - Key idea: Applet carries puzzle + puzzle-solving code
- Where can we apply this?
  - SSL (Secure Sockets Layer)
  - Web-based password authentication

Conclusions
Contributions of paper

- Introduces idea of client puzzles for on-the-fly resource access control
- Puzzle and protocol description
- Rigorous mathematical treatment of security using puzzles -- probabilistic/guessing attack

Questions?