New Client Puzzle Outsourcing Techniques for DoS Resistance

- Paper by Waters, Juels, Halderman, and Felten
- Presenter: Michael Peck

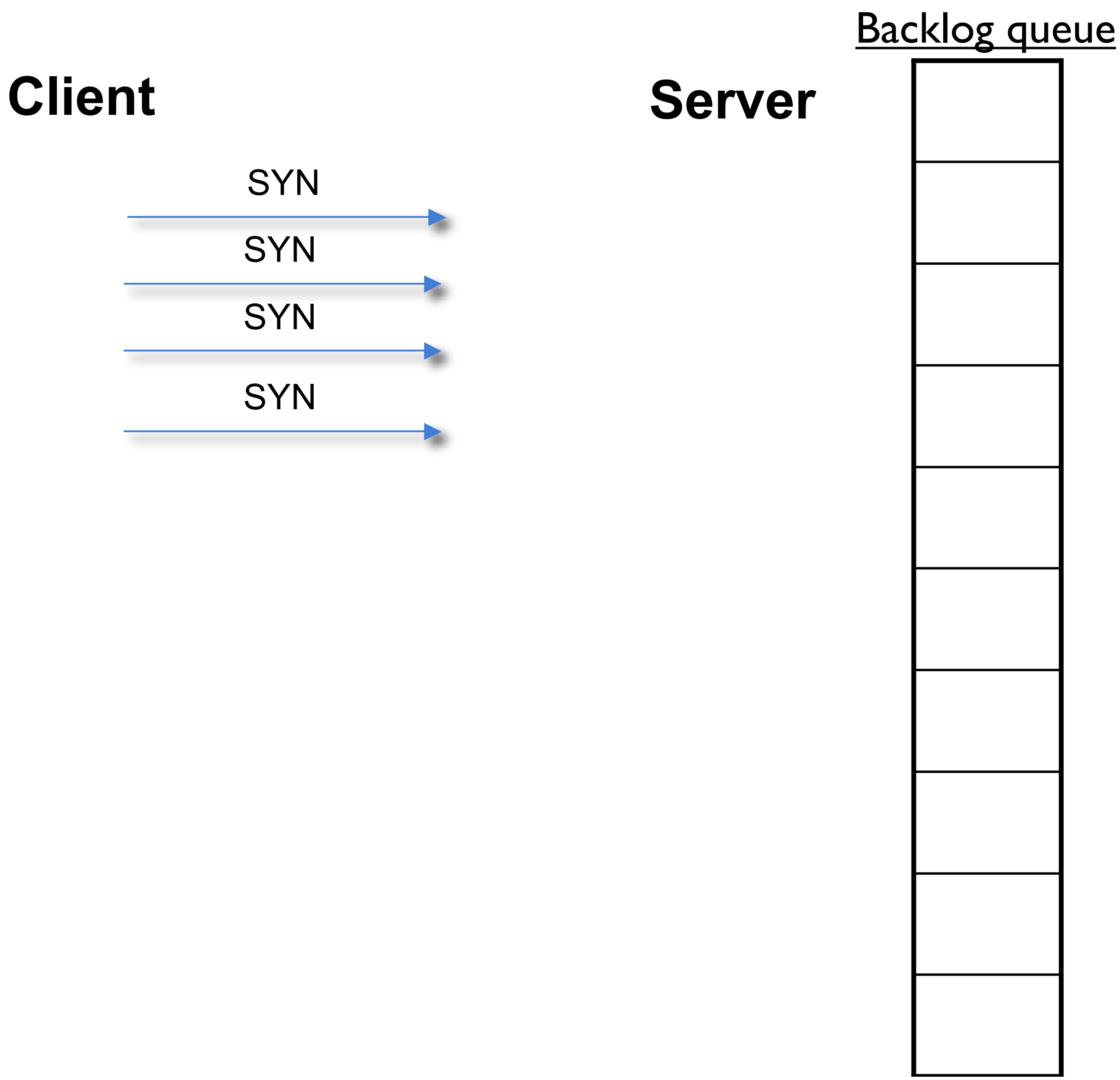
Outline

- Need for client puzzles
- Basic idea of client puzzles
- Related ideas
- Juels/Brainard paper
- Waters et al. Client Puzzle Outsourcing paper

Need for client puzzles

- Fight DoS/DDoS attacks
  - SYN floods and other connection depletion attacks
  - Attackers consume all server resources, leaving none for legit clients
- Fight spam

SYN floods



Hashcash (Adam Back)

Ver	# bits	Date	Resource	Ext	Rand	Counter

- Two sample tokens:
  - 1:20:050323:mpeck@jhu.edu::sa0D5ybM1AMVmoJ6:00007EOW
  - 1:20:050323:mpeck@jhu.edu::nyRy2TzXOSGMVRIR:00000twV
- Verification:
  - echo -n "1:20:050323:mpeck@jhu.edu::sa0D5ybM1AMVmoJ6:00007EOW" | openssl sha1
    000003ec0cda9b5f640cdd1caaf6081bad65dfa0
  - echo -n "1:20:050323:mpeck@jhu.edu::nyRy2TzXOSGMVRIR:00000twV" | openssl sha1
    000008f39f027ce00891554f2620c7563245c672

Basic idea

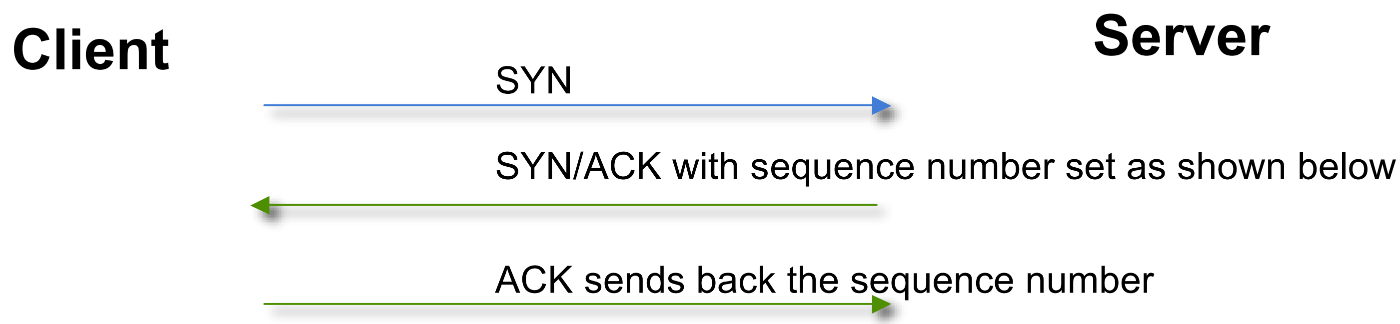
- Force client to commit resources (CPU or memory) before the server commits resources on the client’s behalf
  - Workstations have more power than they really need, might as well use some of it.
  - Makes client put in some effort of its own

Alternative strategies

- “Postage” - client sticks on some kind of proof that it paid a nickel or other small amount - proposed for e-mail.
- CAPTCHAs - client proves that there’s a human on its end actively participating
  - Widely used - especially for registering for free e-mail accounts (Gmail, Yahoo!, Hotmail, etc.)

Alternative strategies

- SYN cookies
  - Don’t keep any state on the server until the connection is established.
  - Minor eavesdropping weaknesses



SHA1(

Src Addr	Dst Addr	Src Port	Dst Port	Time	Secret

)

Attack Model

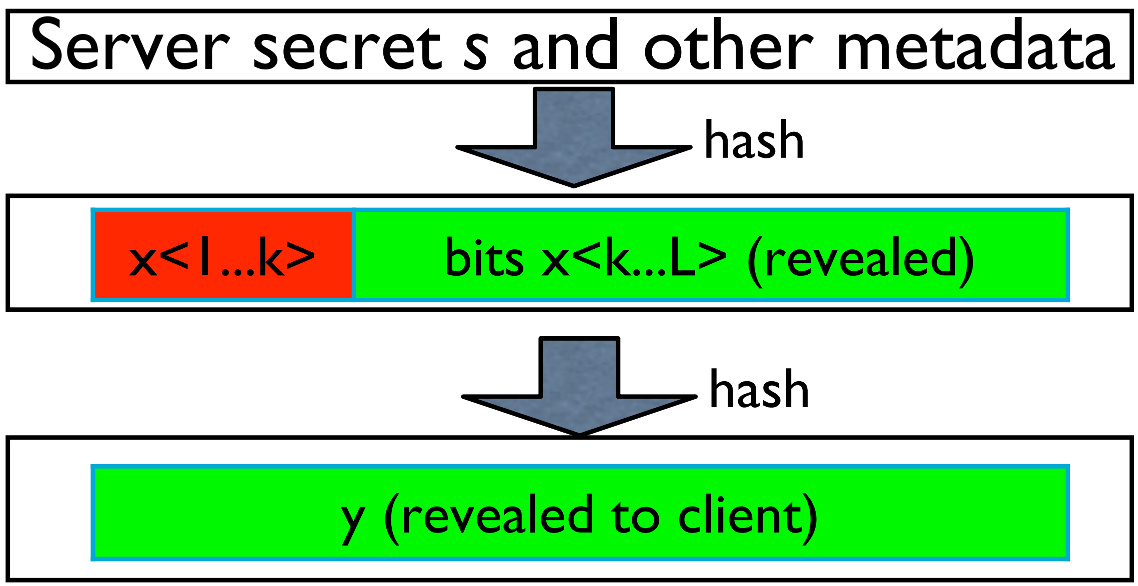
- Attacker can’t modify packets between clients and servers
- Attacker can’t significantly delay packets
- Attacker can’t saturate server, network, or any port
- Attacker can perform IP spoofing
- Can attacker eavesdrop?
  – Juels paper and Waters paper disagree

Properties of good client puzzles

- Stateless on server until client provides valid solution
- Server can verify solution quickly
- Client takes time to compute solution
  – But not too much or too little
  – Hard to account for varying CPU speeds

Juels/Brainard Paper (NDSS ‘99)

- Server hands out puzzles to clients when under attack.
- Puzzle made up of n independent subpuzzles each of difficulty k to solve



Juels/Brainard Paper

- $m * 2^k$ complexity for client to solve puzzle
  - m = number of subpuzzles
  - k = # of bits of x not revealed to client

Juels/Brainard Paper

- Improvement suggested to make subpuzzles dependent, for quicker verification on server.

Puzzle auctions

- Wang, Reiter - 2003
  - Client decides puzzle difficulty (bids)
  - Server allocates resources first to client who solved most difficult puzzles
- Somewhat backwards-compatible
- More on this tomorrow

Shortcomings

- Existing schemes themselves can be subject to DoS attack
  – Puzzle creation/verification requires hash computations in Juels/Brainard scheme
- Existing solutions require on-line computation by clients - wastes users' time
  – On-line computation doesn't hurt attackers as much since they're not interactive users

Waters et al.

- Outsource puzzle creation and distribution to a bastion
  - Same puzzles can be used by clients for multiple, unrelated servers
    - Bastion can be mirrored
    - Servers don’t have to worry about creating puzzles
    - Servers & bastion need to stay in sync

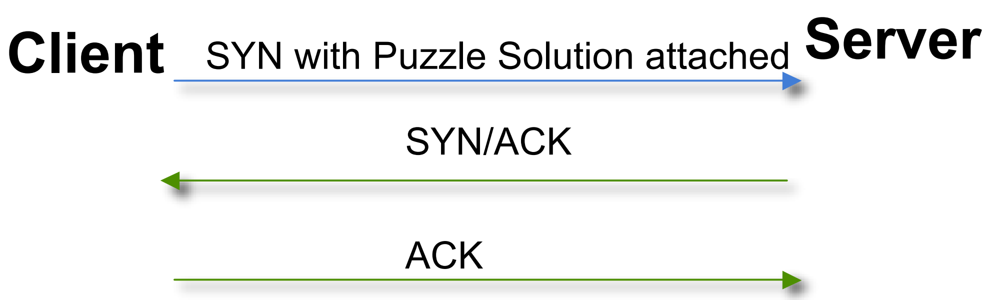
Approach

- Outsource puzzle creation to bastion
  - Servers can all share the same puzzles
- Solution verification only requires a table lookup
- Clients can solve puzzles slightly ahead of time
- Solving puzzle only gives client access to a small slice of the server’s resources (virtual channels)

Virtual channels

- Each puzzle solution is only valid for a specific channel - but, the solution can be used for ANY server
- Server limits how many connections are accepted per channel
- Channels designed to separate attackers & regular users

Virtual channels



Channels

1	Solution 1
2	Solution 2
3	Solution 3
4	Solution 4
...	
N	Solution N

Puzzle construction goals

- Unique puzzle solutions (needed for lookup)
- Per-channel puzzle distribution
- Per-channel puzzle solution
- Random-beacon property
- Identity-based key distribution
- Forward secrecy
- But, make sure a server can’t compute another server’s solution

Hash function inversion won’t work

- Doesn’t meet the per-channel puzzle distribution property
- So, use Diffie-Hellman based scheme for constructing puzzles
  - Bastion creates puzzles, distributes to clients & servers
  - Servers adapt puzzles to themselves (compute puzzle solution using backdoor)

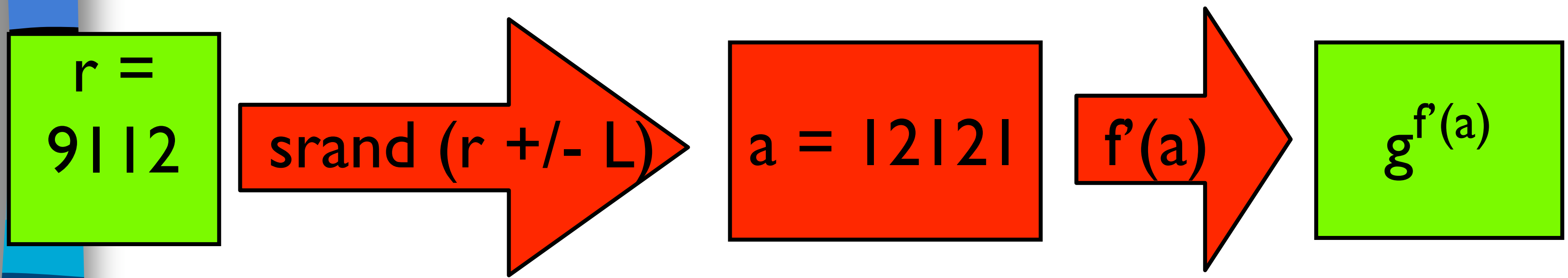
DH based construction

- Each server has a D-H secret key x_1 and a D-H public key g^{x_1}
  - Public keys distributed to clients
- Bastion selects a random integer $r_{c,T}$

DH Construction

- Bastion uses first random number as a range for seed to generate a second number a
- Second D-H secret key set to f'(a).
  - $g^{f'(a)}$ (f' is a one-way function)
- Bastion publishes $g^{f'(a)}$ and r

DH Example



Bastion publishes range for the seed, and a D-H public key

Steps taken by server

- Server precomputes each puzzle solution by doing one modular exponentiation.
  - But, has to do this once for each channel
- Stores solutions in a table for quick lookup
- Cost: (calculated with BouncyCastle)
  - Modular exponentation (768 bit): 10ms
  - SHA-1 hash computation (448 bit): 0.4 ms

Steps taken by client

- Client brute-forces the seed
  1. Guess a candidate a’
  2. Apply one-way function to a’
  3. Compute $g^{f'(a')}$
  4. If matches published value, save, and combine with server’s public key as needed
- Requires an average of L/2 modular exponentiations

Server public key distribution

- Could use identity-based public keys
  – Server's public key derived from a string representing the server & public parameters.
- Trusted dealer gives servers their private keys
- Not used for prototype implementation due to inefficiencies

Time-lock puzzles

- Proposed by Rivest, Shamir, Wagner (1996)
- Achieves random-beacon property
  - Puzzles can be based on stock index quote or some other widely distributed value
- May not achieve per-channel puzzle solution property
  - Client has to compute a solution for each individual server that it wants to access

System description

- Each server has *n* virtual channels
  - n is fixed for all servers using bastion
- Each solution to a channel is valid for time period *t* (several minutes).

System description

- T_i denotes the *i*th time period.
- At beginning of T_i, bastion publishes puzzles whose solutions will be valid during T_{i+1}.
  – Each server computes all puzzle solutions for all channels and stores in table for easy lookup to have ready by T_{i+1}
  – Each client solves puzzles for randomly chosen channels to have ready by T_{i+1}

How many channels?

- More channels are better
  - Decreases chance that a legitimate client is using same channel as an attacker
- Server's memory & CPU power limit the number of channels
- Unlike other client puzzle schemes, this scheme directly benefits from technological advances
  - Hopefully advances benefit server more than attacker

Prototype Implementation

- Client puts token into an option field of TCP SYN packet
- Server uses token to put client in a channel
- Each channel only accepts a new connection every *n* seconds.
- Bastion:
  - Creates/distributes new puzzles at regular interval via HTTP

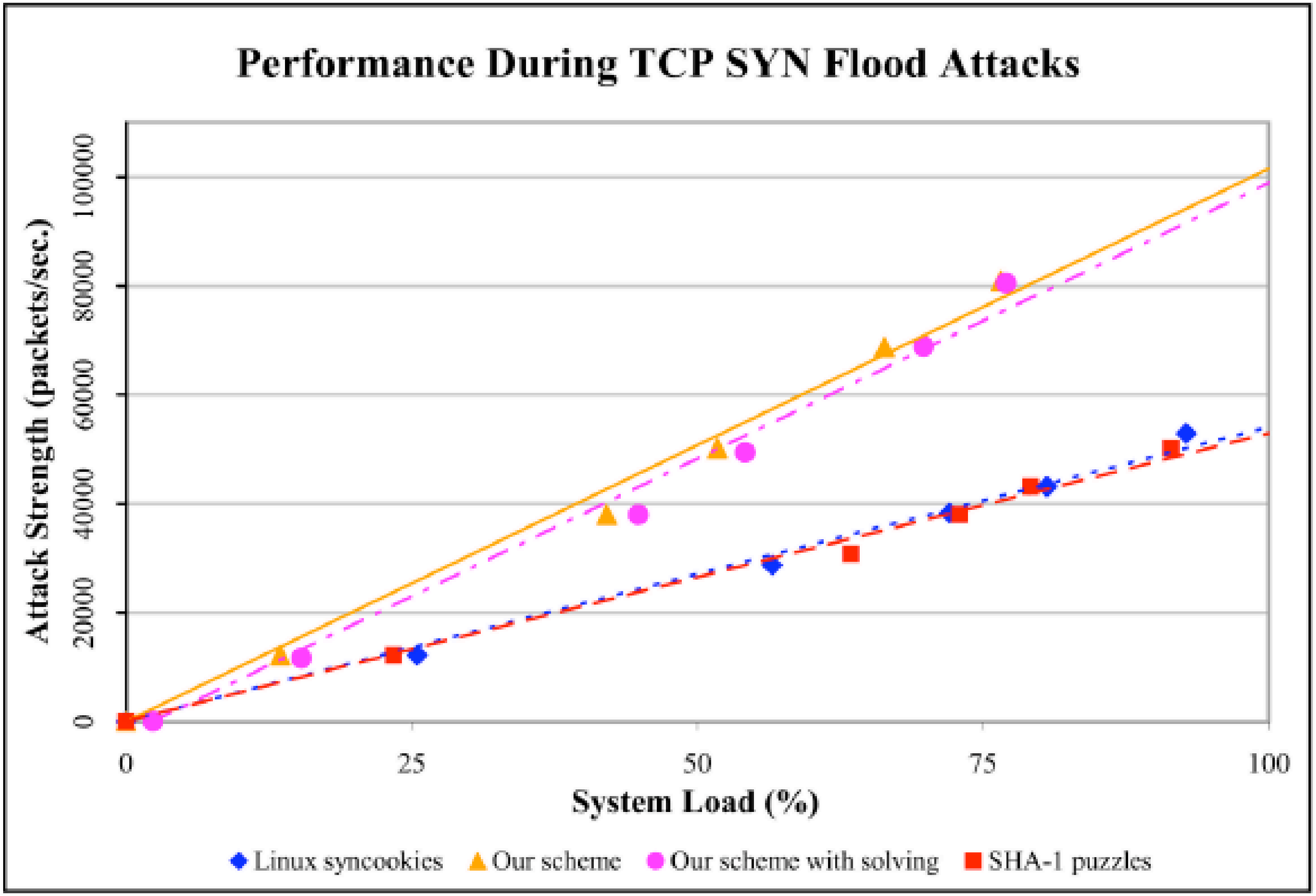
Prototype Implementation

- Server: Two applications
  - User space: Retrieve new puzzles from bastion & precompute solutions using D-H private key
  - Kernel space: Filter incoming SYN packets, rate limit virtual channels

Experiment

- Compared implementation to simulated conventional hash puzzles and Linux syncookies
  - Simulated conventional hash puzzles:
    - Server computes a SHA-1 hash in place of puzzle verification, then drops packet
      - Juels/Brainard use MD4, does this matter?
- 10,000 virtual channels
  - Approximately 100 seconds needed for server to precompute solutions

Performance



Experiment limitations

- We'll cover these tomorrow

Extensions

- Flexible number of channels per server
  - Servers have varying needs / processing capabilities
  - Secondary puzzles
    - Solutions to secondary puzzles encrypted with solutions of primary puzzles

Extensions

- Deploy at IP level instead of TCP level
  - Implement in routers
  - “Biggest challenge” - where to put the token in IP packet?
- Fight eavesdropping attacks (even though out of scope of the attack model)
  - Problem: Eavesdroppers can steal channels from legitimate clients by replaying tokens
  - Proposed solution: Create an IPSec tunnel?

Summary

- Client puzzle creation/distribution can be outsourced, to prevent DoS attacks on the client puzzle scheme itself
- Client puzzle verification can be done with a simple table lookup, once again to prevent DoS attacks on the client puzzle scheme itself