# Putting the 'PoW' in Proof-of-Work!

Ed Kaiser & Wu-chang Feng

## The Problem

Automated attacks on the web remain a problem.

***CAPTCHA*** is a Turing Test defense that uses distorted images to distinguish humans from computers.

- \+ No special software needed
- − Inaccessible and frustrating
- − Economics are broken; outsourced for under 1¢ per CAPTCHA on sites like GetAFreelancer
- − Many have been completely broken

***Proof-of-Work*** is a defense that prioritizes service requests based on the clients' willingness to solve computational challenges.

- \+ No user interaction needed
- − Requires installing special client software to solve the challenges
- − Clients without the software are rejected

**Can their strengths be combined?**

1

## The Goals

***Transparency*** so that no user input is needed.

***Backwards-compatibility*** so that clients do not need to install software.

***Flexibility*** **to bind** work functions to the client, server, and time and then t**ailor** the challenge with a client-specific difficulty to prioritize clients based on their past behavior.

***Efficiency*** to minimize overhead.

2

## Our Solution

Embed the Proof-of-Work functions and responses ***within the URLs*** of protected web content.

Clients use ***JavaScript*** to solve the work functions.

The server uses an ***Apache*** module to prioritize HTTP requests based on the solution in the URL;

| | | |
|---|---|---|
| valid solution | → | high priority |
| missing solution | → | low priority |
| expired solution | → | low priority |
| invalid solution | → | low priority |

3

## The Work Function

Find an answer **A** that satisfies:

$$H(D_C, N_C, URL, A) \circ 0 \bmod D_C \quad (1)$$

**H** is a one-way hash function (i.e. SHA1) with uniformly distributed output

$D_C$ is a client-specific server-assigned difficulty

$$N_C = IP_C \text{ Å } N_S \quad (2)$$

$IP_C$ is the client's network identity

$N_S$ is a frequently-updated server-generated random number

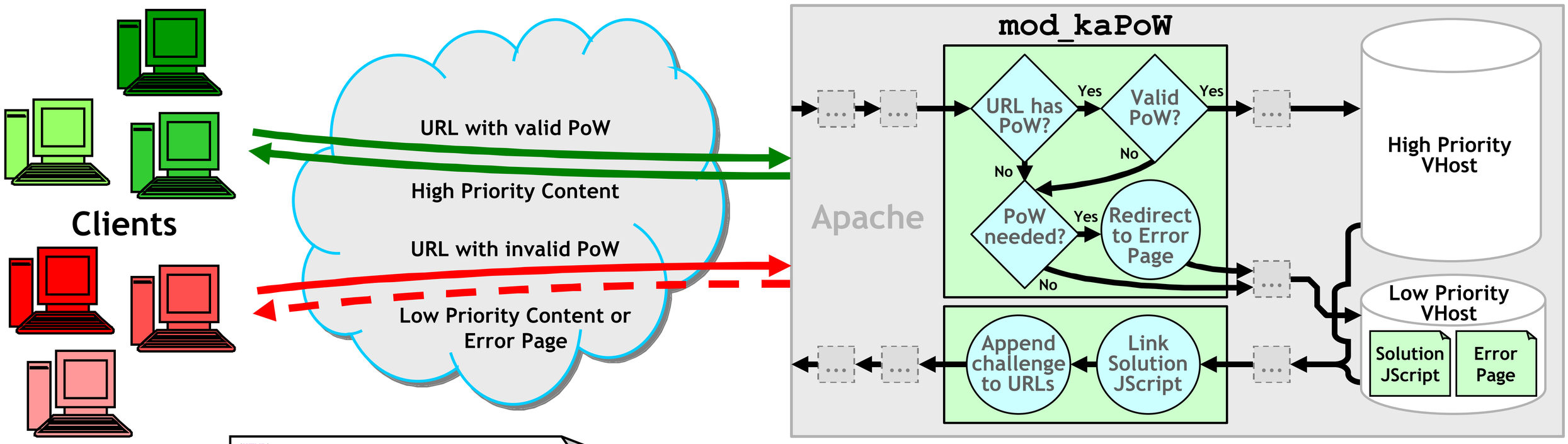
4

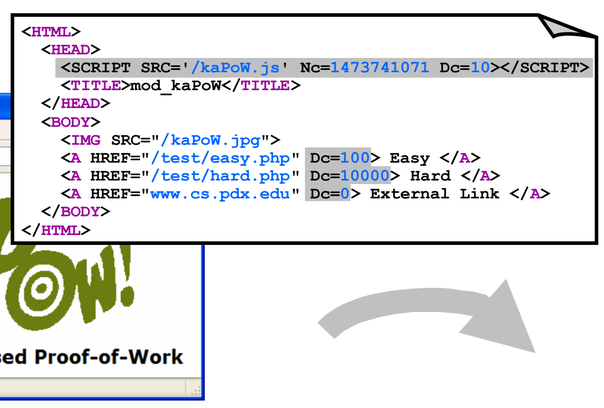
## Challenge Difficulty

The client-specific difficulty $D_C$ is assigned based upon the client's load history, which is stored efficiently using a counting Bloom Filter indexed by the client's identity $IP_C$.

Each entry measures a client's cumulative load from ***successful*** requests (i.e. those with valid solutions) and is periodically decayed.

5




**User clicks a PoW-protected link.**
**HTML modifications are highlighted.**

**Their browser solves the work function.**
**The progress bar is only shown for very difficult challenges.**

**A valid answer yields the content.**

## Transparency

Scripts solve the challenges
- user input is not needed

Challenges solved just in time
- image URLs solved as DOM is loaded
- hyperlinks solved after user click

Error page automatically solves the work function and refreshes using the correct URL
- error page is not seen by users

6

## Backwards-Compatibility

Clients lacking *JavaScript* can access content via low priority virtual host
- all clients can access the content

No changes to content are necessary
- whether statically or dynamically generated
- the module handles the necessary modifications to outgoing webpages

7

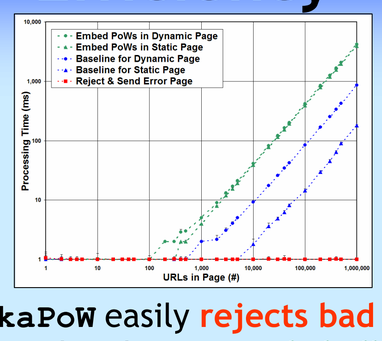
## Flexibility

Answer **A** becomes invalid if the $N_C$, $D_C$, or **URL** changes (Equation 1 fails).
- client $IP_C$ changes → $N_C$ changes
  - **A** is bound to the client & server
- $N_C$ frequently expires → no replay
- $N_C$ is random → no pre-computation
  - **A** is bound to a specific time

Difficulty $D_C$ tailors client workloads proportional to their past behavior.
- economizes work-for-content

8

## Efficiency




- **mod_kaPoW** easily **reject bad URLs**
- small overhead to **append challenges**
  - only noticeable for files with hundreds of URLs

9

PORTLAND STATE UNIVERSITY

Project Webpage: http://kapow.cs.pdx.edu

Funded by: National Science Foundation, intel