Systems and Internet Infrastructure Security

Network and Security Research Center
Department of Computer Science and Engineering
Pennsylvania State University, University Park PA

Scalable Integrity-Guaranteed AJAX

Patrick McDaniel,
14th Asia-Pacific Web Conference (APWeb)
April 12, 2012
Kunming, China

When systems fail ...

The big question ...

- What guarantees does a secure web session provide?
  - SSL: The content comes from a system that possesses a private key that somebody paid to have vouched for. More directly, the authenticity of the source.

- What do you want to know?
  - The content source was generated by *legitimate sources* running *legitimate software* from *legitimate data* ...

SSL does not give you a secure web any more than an armored car gives a secure banking system.

Integrity Guaranteed Documents

- *Integrity guaranteed documents a provable binding of the data to the system that generated it.*
  - shows the data was generated or delivered by a identifiable system that (e.g., not compromised).
  - Is the system good?
    - For some value of good ...

- Note: If the customers knew that that bank server was running compromised (logger) code, they would not have been tricked into giving up all their personal data.

Problem

How can we provide integrity guarantees data for commercial grade web servers*?

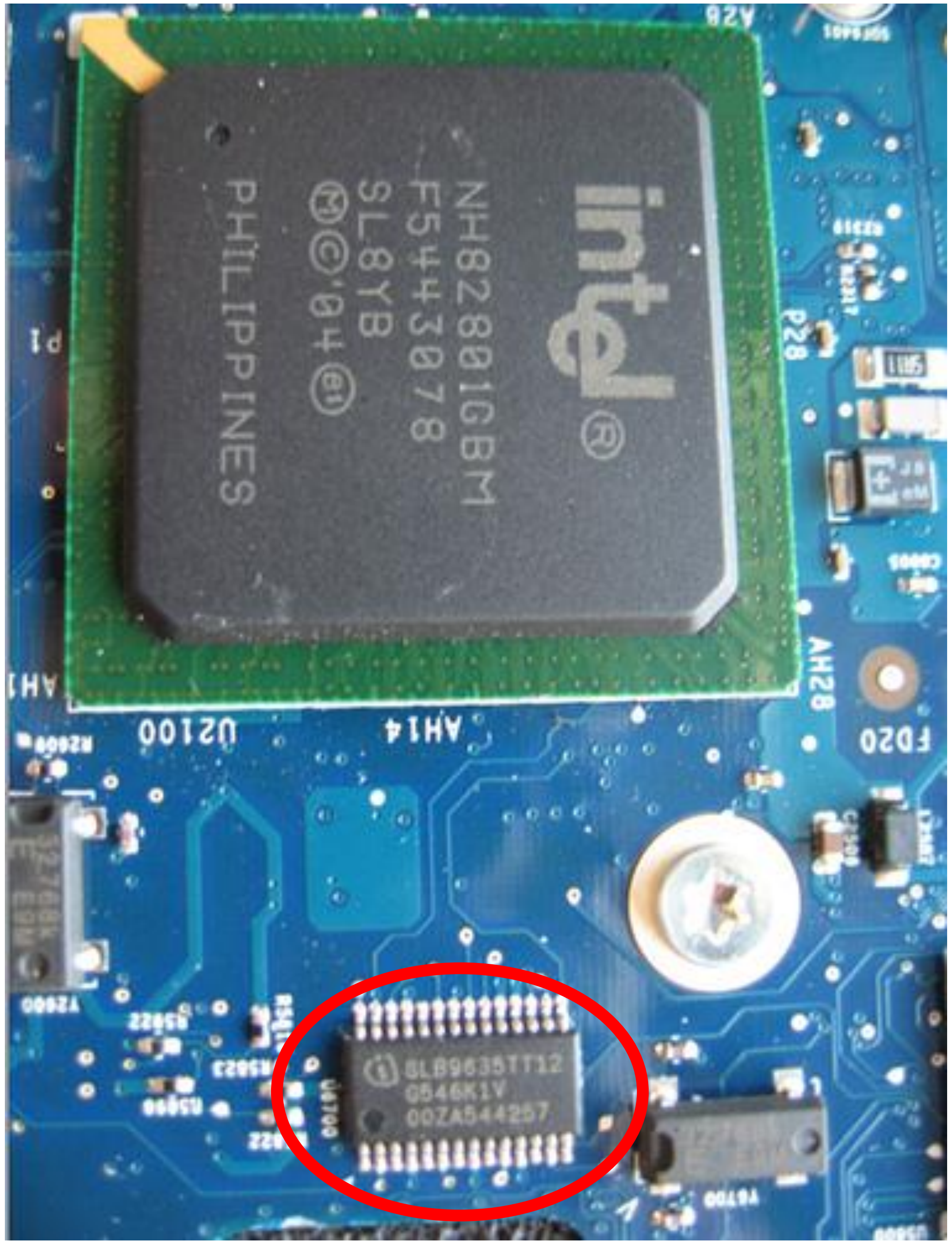
* Web systems are focus of this talk, but the work applies to other domains.

Integrity Measurement

- *Integrity measurement* is a sub-field of systems security that aims to certify software running on a computer system.
  - The system uses hardware support to *measure* software.
  - Remote parties request proof of the certification using an *attestation protocol*
  - Failure indicates *untrusted software*-system is compromised in ways that are otherwise undetectable
    - e.g., root-kit, trojans, ...
- Genesis: *secure boot*

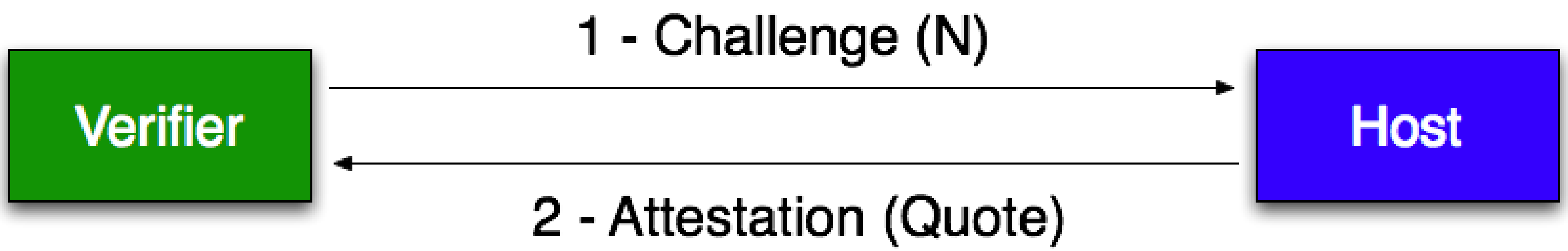
TPM

- The Trusted Platform Module is a tamper resistant secure crypto-processor.
    - Manages cryptographic keys and functionality it uses to support security relevant operations.
    - Measures the code loaded by the system (firmware, BIOS, OS kernel, device drives, application processes, ...)
        - Measurements are hashes of loaded code (PCRs)



Integrity Measurement

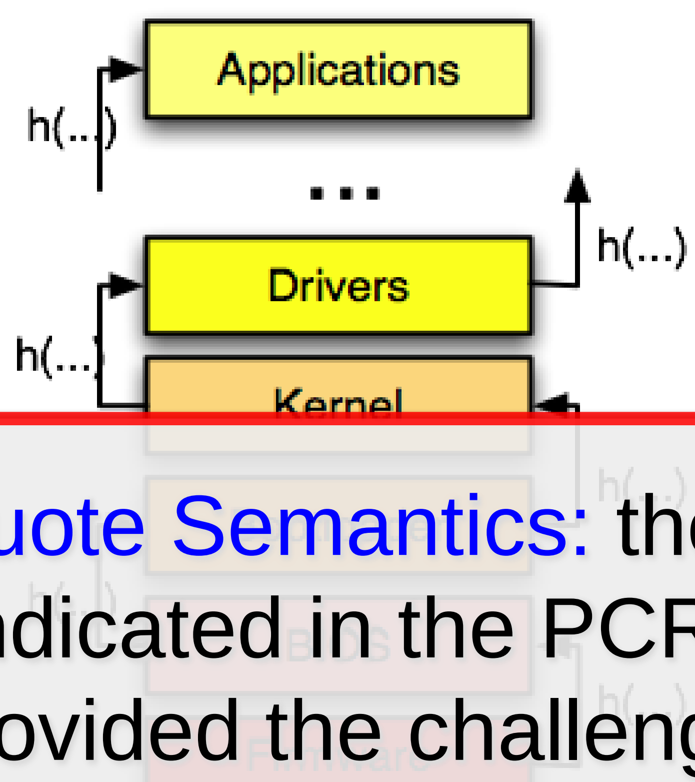
- Each system has a unique public key pair called the attestation identity key (AIK)
  - The AIK is (indirectly) certified by the manufacturer at the time the system is built - private key only visible to TPM
  - This key AIK^- is used to sign attestation operations
  - The verifier validates the quote partially using the AIK^+ at the time the quote is received



The Integrity Quote

AIK challenge

$$Quote(H_w, PCR_{H_w}, n)$$



- The full quote contains:
  - The signature on the quote:

    $$\{PCR_{H_w}, n\}_{H_w}$$

  - A *measurement list*
  - AIK+ and validating certificates
- The verifier
  1. Validates the keys/certs
  2. Validates the signatures
  3. The measurement list
  4. The challenge (nonce)

Quote Semantics: the system H_w is running known software (indicated in the PCR register) at or about time the verifier provided the challenge nonce n.

STOP: All this machinery does is identify what software is running on a system.

Three Challenges of IM

- Key management
  - Largely an issue of certification (specification)
- Measurement List Management
  - Where do these lists come from?
  - How do you know what is the “correct” code?
- Performance
  - How do you do all of this in a timely manner?
  - *This question is the focus of this talk ....*

Problem

How can we supply integrity-guarantees for commercial grade web servers*?

* Web systems are here, but the approach applies to other domains.

An observation ...

PENNSTATE 1855

- Why not use the TPM to tie the content to the software running on a host?
  - Hash the document as TPM

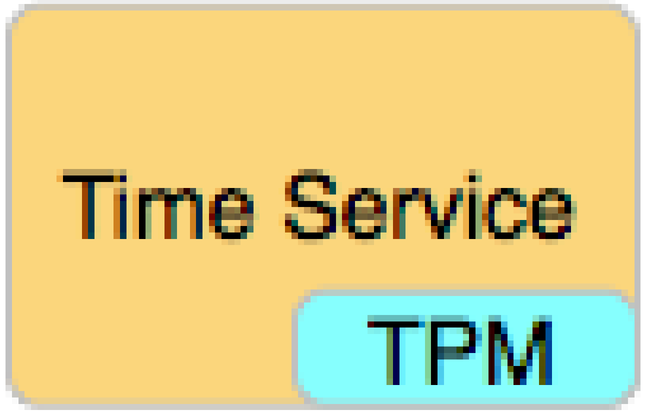
$$\text{Quote}(\text{H}_w, pcr_{H_w}, h(p_i))$$

AIK PCR state web page

  - The TPM attests the system code and content by performing a normal system quote
  - Proof verified using existing TPM validation approach
  - Content proof delivered with web page
- Resulting proof represents *integrity guarantee*

Quote Semantics: the system H_w running known software (indicated in the PCR register) delivered document p_i.

Semantic limitation: time



- Problem: the proof does not indicate
- when the content was generated.

- Sln: *time service (TS)* content/time binding.
  - ROT/TS provides periodic attested time quote:

$$Quote(H_{TS}, pcr_{H_{TS}}, h(t_i))$$

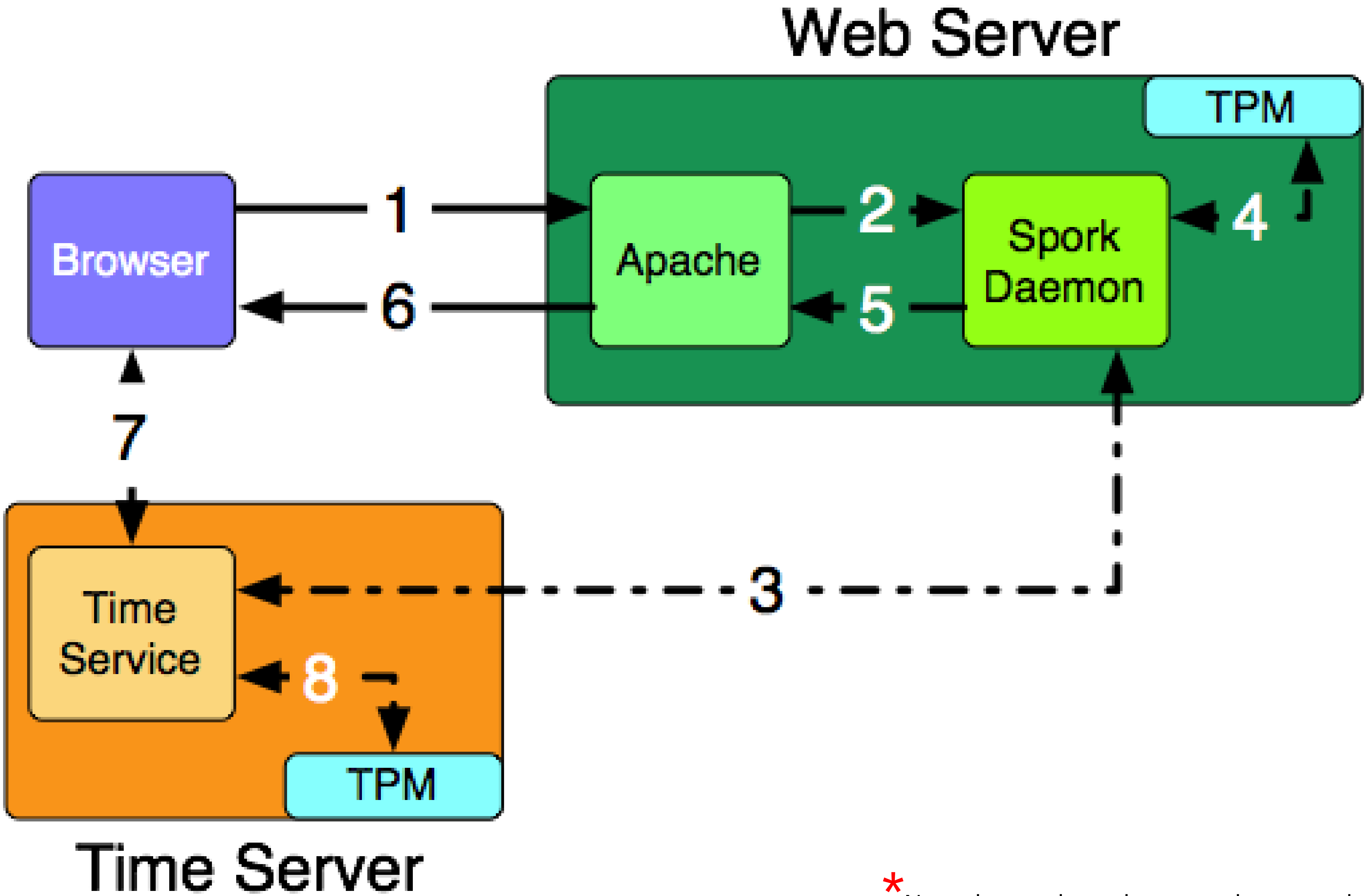
  - Webserver obtains periodic time quotes (push or pull)
  - Integrate directly into the content proof:

$$Quote(H_w, pcr_{H_w}, h(h(p_i) | Quote(H_{TS}, pcr_{H_{TS}}, h(t_i)))) \, | \, Quote(H_{TS}, pcr_{H_{TS}}, h(t_i)) \, | \, t_i$$

time server quote — time

Quote Semantics: the system H_w running known software (indicated in the PCR register) at or about time t_i delivered document p_i.

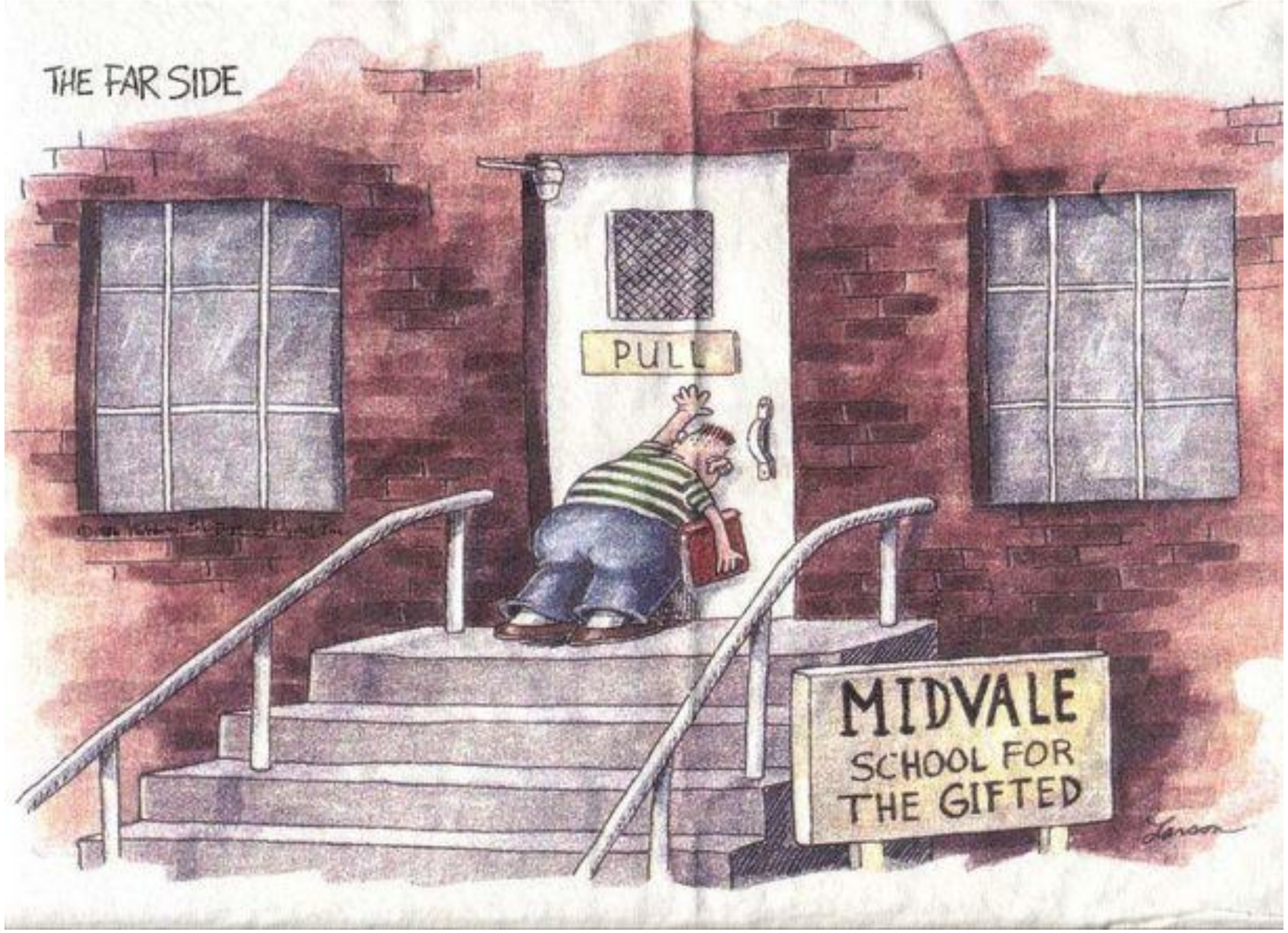
Spork* Web System



*Not quite a web service, not quite a security service

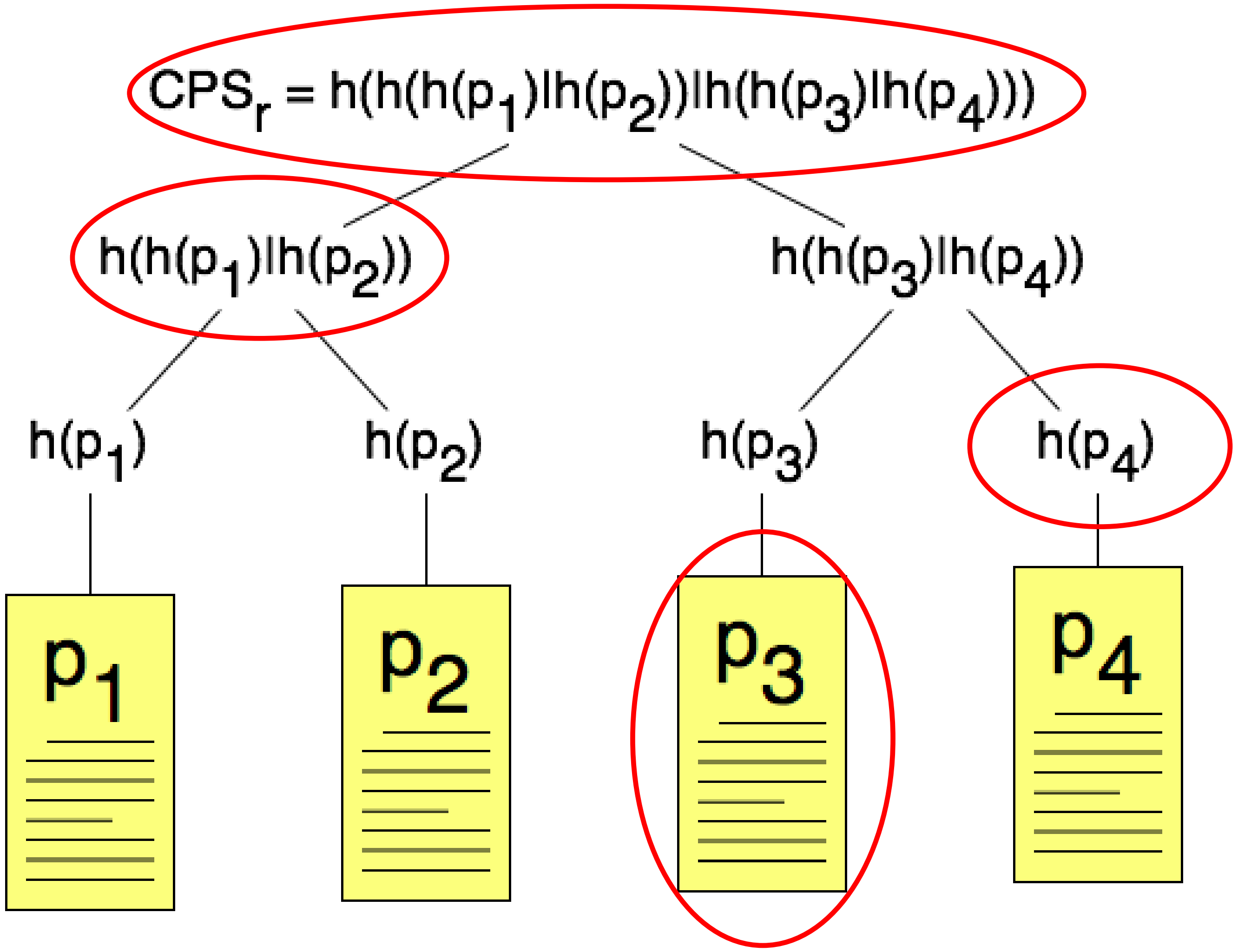
But wait ...

- If a single TPM quote takes 900+ msec, *how is this ever going to work in a real system*?



Cryptographic Proof Systems

- *Cryptographic Proof Systems* (CPS) amortize verification costs using a small number of crypto operations.
  - A Merkle Hash Tree is the canonical CPS
  - A succinct *proof* is a page and its siblings on the path to the root



- *Others*: authenticated dictionaries, skip lists, revocation trees

TECHNIQUE: Use cryptographic constructions to amortize computation to create small “proofs” over all documents served in an epoch.

Amortized Proofs

- Using CPS:

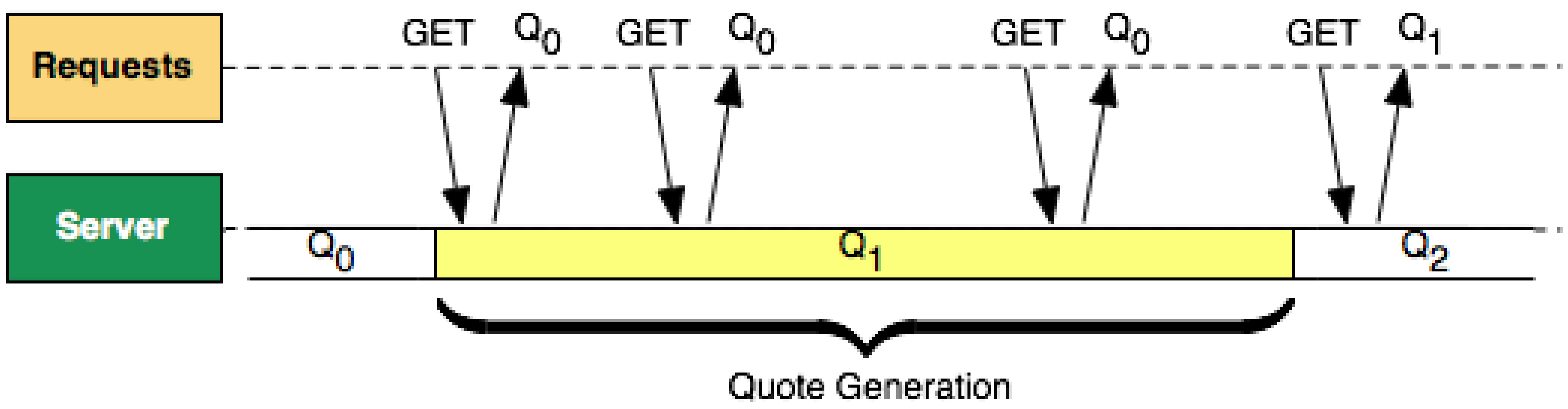
$$\underbrace{\underbrace{Quote(H_w, pcr_{H_w}, h(CPS_r}\,|\,\underbrace{Quote(H_{TS}, pcr_{H_{TS}}, h(t_i))))}}_{\text{web server quote (content proof + time server quote)}}\,|\,\underbrace{CPS_r}_{\substack{\text{proof}\\ \text{sys. root}}}\,|\,\underbrace{Quote(H_{TS}, pcr_{H_{TS}}, h(t_i))}_{\text{time server quote}}\,|\underbrace{Pf(p_i)}_{\substack{\text{page}\\ \text{proof}}}|\,\underbrace{t_i}_{\text{time}}$$

- Advantages:
  - Web server only needs one TPM quote for many pages.
  - Browser needs to perform only one expensive signature validation per one CPS.
  - Proofs can be cached with content, e.g., in squid cache.

- Q: Which pages do you include in a proof system?

Static Proof Scheduling

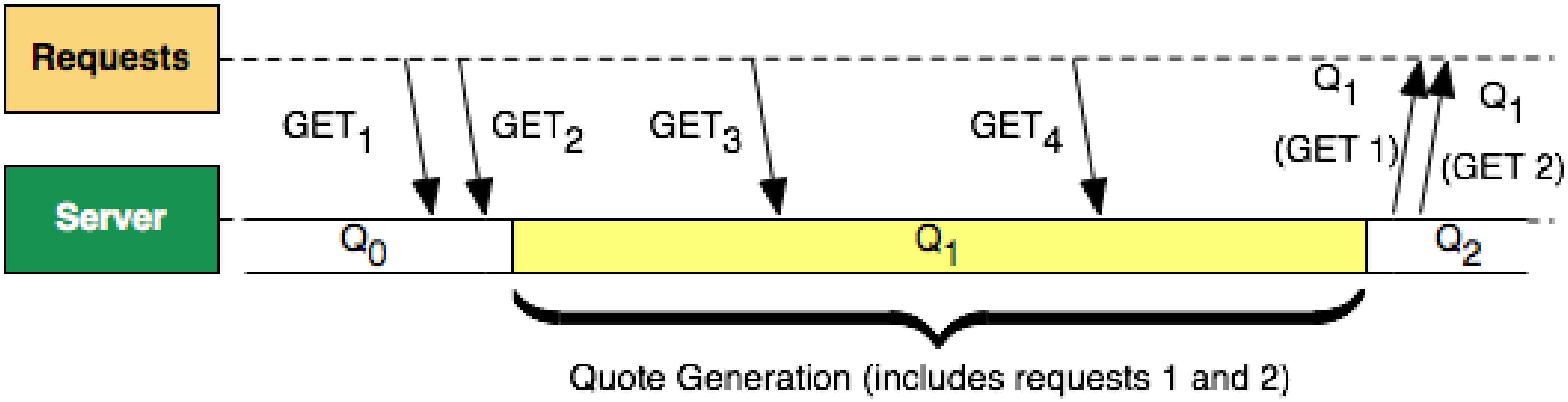
- Create MHT for all static pages *periodically*
  - at the rate of the TPM quote mechanism
  - provide the most recently completed proof (*the page is guaranteed to be in it because all pages in each proof*)
  - proof latency seen by browser is bounded by request RTT because there will always be a valid proof available



Dynamic Proof Scheduling

PENNSTATE 1855

- Create MHT for all dynamic pages for request received since the last proof generation began
  - begin the TPM quote for all dynamic content “batched”
  - respond with quote when the associated proof is available
  - proof latency seen by browser is bounded by 2 * TPM quote (or on average 1.5 * TPM assuming uniformly distributed inter-arrival times, i.e., 1350 msec)

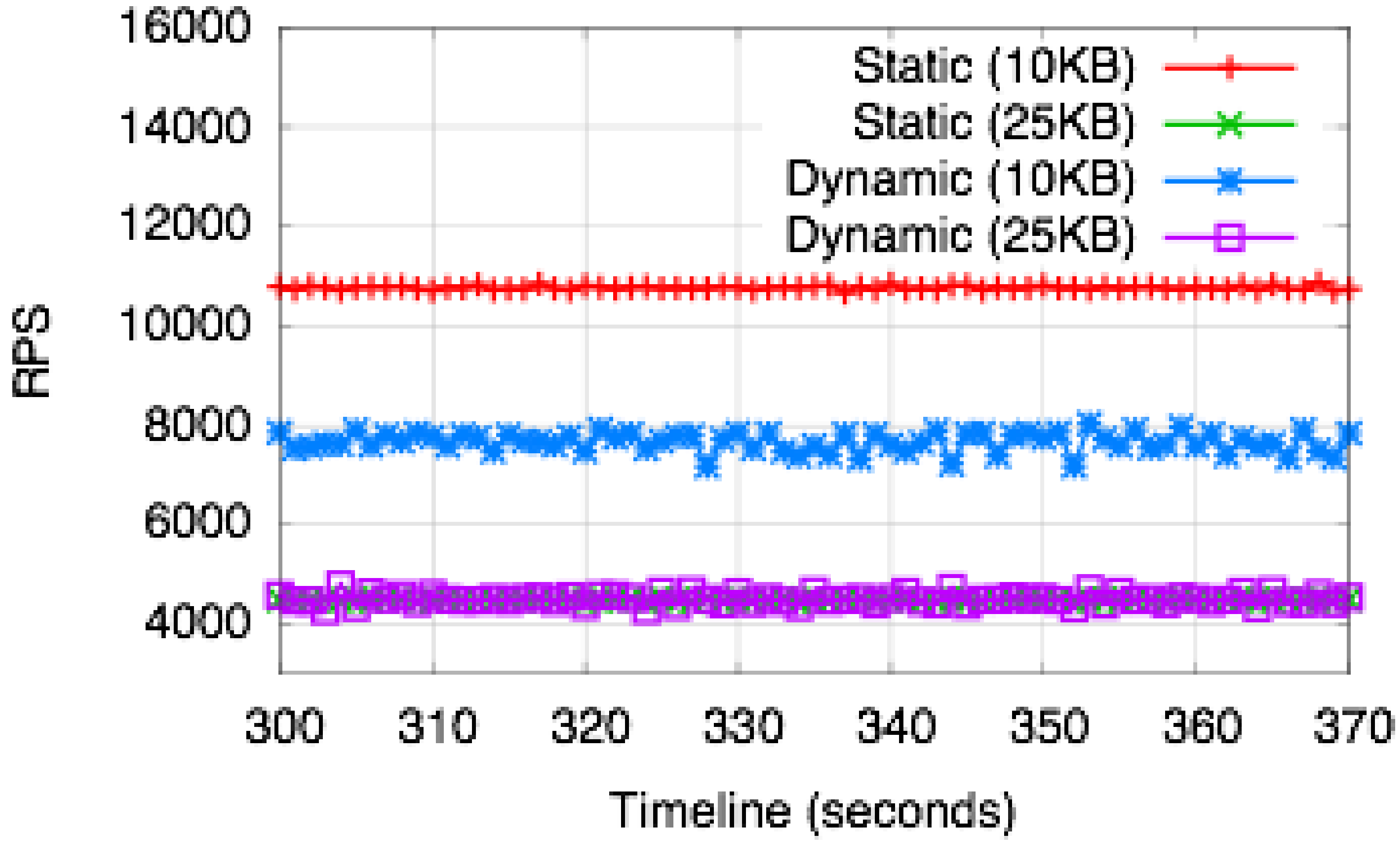


Evaluation

PENNSTATE 1855

- *Question*: what are the costs on real traffic?
- Setup:
  - Apache 2.2.8
  - Ubuntu Linux 8.0.14, kernel 2.6.24
  - 6 Dell M650 blades (8 core, 2.3 Ghz, 16GB RAM)
    - 1 web server, 1 time server, 4 clients (Apache JMeter)
  - Gigabit Ethernet (quiescent network)
  - 5,000 LOC
    - Python (web services)
    - C (custom TPM integration code)
    - Firefox client extension

Baseline Performance

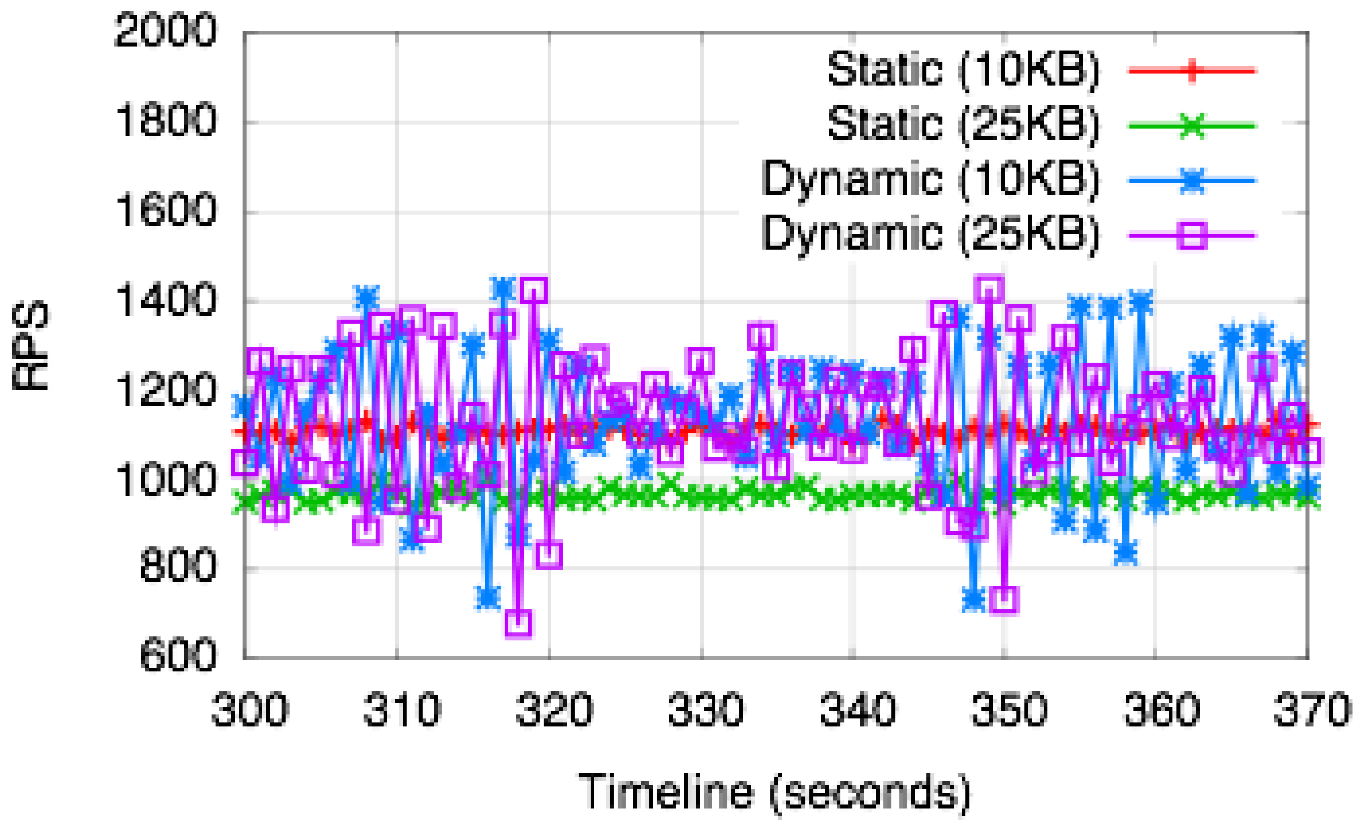


- Static content is bound by network *bandwidth*

$$10,770 * 10 = 107,700KB/s \approx 4,485 * 25 = 112,125KB/s$$

- Dynamic content is bound by *computation*, where the RPS throughput is independent of content size

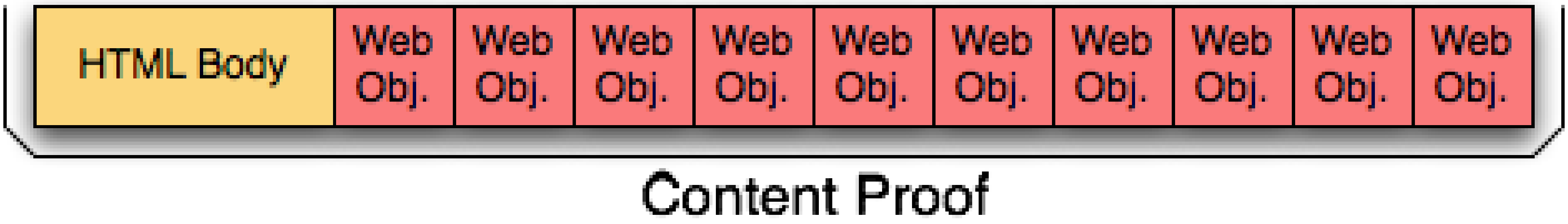
Naive IM Performance



- The IM web server shows bottlenecks similar to the dynamic case, and *substantial* overheads associated content generation.
  - Largely because each content get requires two HTTP GETS, the document itself and a “dynamically generated” proof
  - The proofs are quite large (106KB)
  - Compression may help, increasing throughput by as much as 20%

Real Web Traffic

- Recent studies have shown that the average web page is a composite of many objects
  - has 25kb base HTML document and 10 (non-flash) embedded objects of 10kb each.
    - .gifs, .jpgs, scripts, style sheets, etc.
  - *Observation*: Wouldn't it be efficient to create proofs over the entire body of page elements and retrieve one proof
    - thereby amortizing the proof acquisition over the entire rendered page



Estimating Throughput

Hence, the expected throughput $\mathcal{P}$ (in RPS) would be:

$$\mathcal{P} = \frac{1}{\left(10 * \frac{1}{\mu}\right) + \frac{1}{\epsilon}}$$

where:

μ is the baseline service time for a 10kb web object

ϵ is the baseline service time for a 25kb web object

- *Note*: content service time is calculated by dividing the RPS throughput by 1 second. For example, the throughput for the baseline static 10kb content is 10,770 RPS, so the service time for a single acquisition is:
- 10,700 RPS : 1/10,769 = 0.00009286 seconds = 92.86 usec

Experimental Results

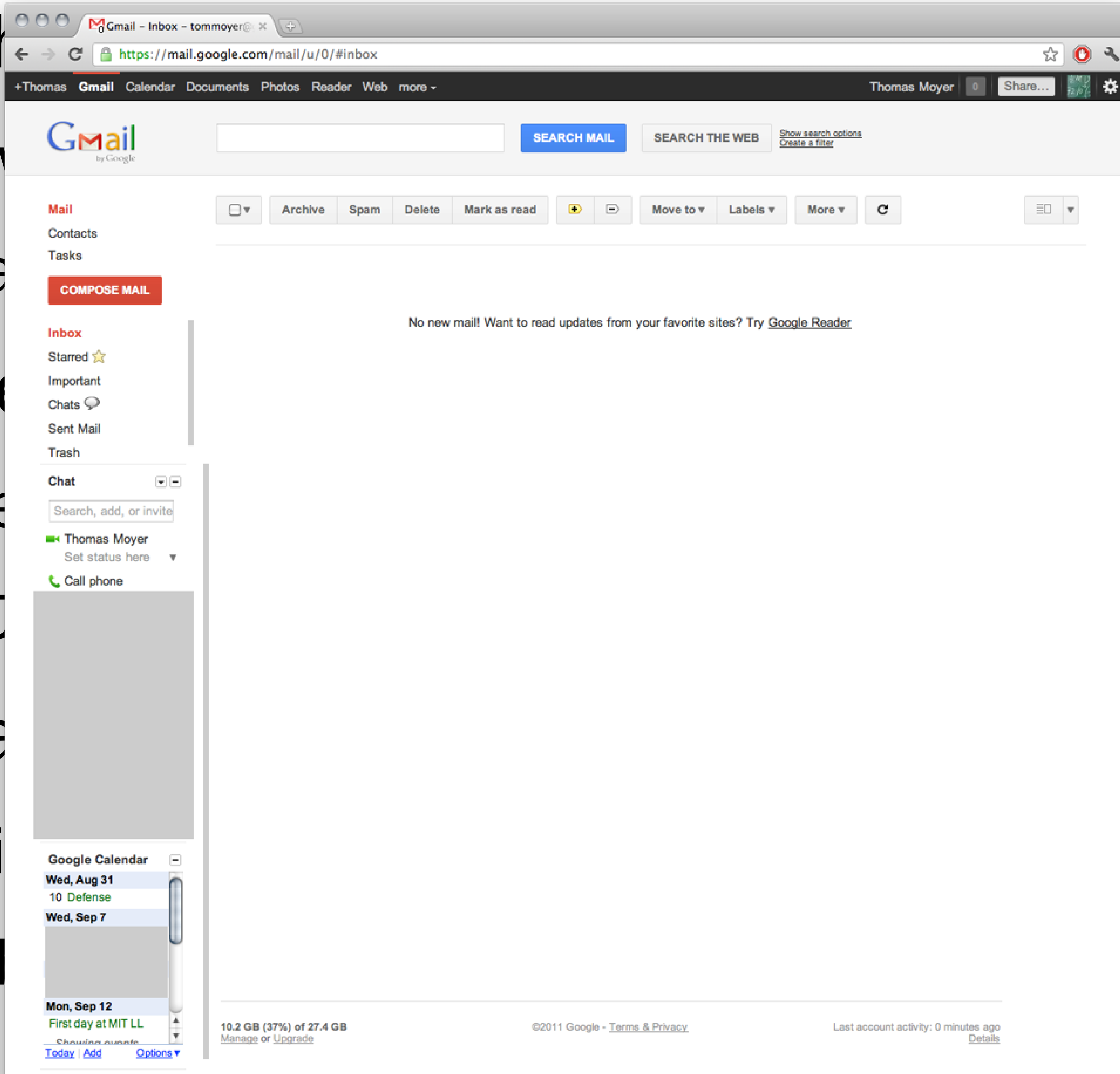
PENNSTATE 1855

	μ	ϵ	**Expected**		**Actual**	
			$\mathcal{P}$	Web Objects	$\mathcal{P}$	Web Objects
Baseline with Static Root Page	10769	4485.5	868.4	9552.5	867.4	9541.5
Baseline with Dynamic Root Page	10769	4507.8	869.2	9561.7	745.9	8204.8
Integ. Measured Static Root (Full IMA)	10769	968.1	509.8	5607.8	494.9	5444.4
Integ. Measured Static Root (Comp. PRIMA)	10769	1526.8	631.5	6946.4	724.3	7967.4
Integ. Measured Dynamic Root (Full IMA)	10769	1130.7	551.6	6067.3	494.4	6438.3
Integ. Measured Dynamic Root (Comp. PRIMA)	10769	1127.2	550.7	6058.1	650.5	7155.1

- Note that an unmodeled interleaving effect on content delivery caused the metric to often *underestimate* throughput
  - In this case, *avoids* underutilization of the network
- Static content is delivered within 17% of line speed.
- Static/Dynamic content can be delivered at almost 8,000/7,000 RPS, well within acceptable rates of commodity web servers.
  - These costs will improve with content size, e.g., large web pages with many objects (possibly more appropriate for workloads with provenance needs).

AJAX Applications

- Asynch
  - New
  - Intera
- Browse
  - No ne ... nged
  - Use J ... arrives
- Popula
  - Gmai
  - Facel



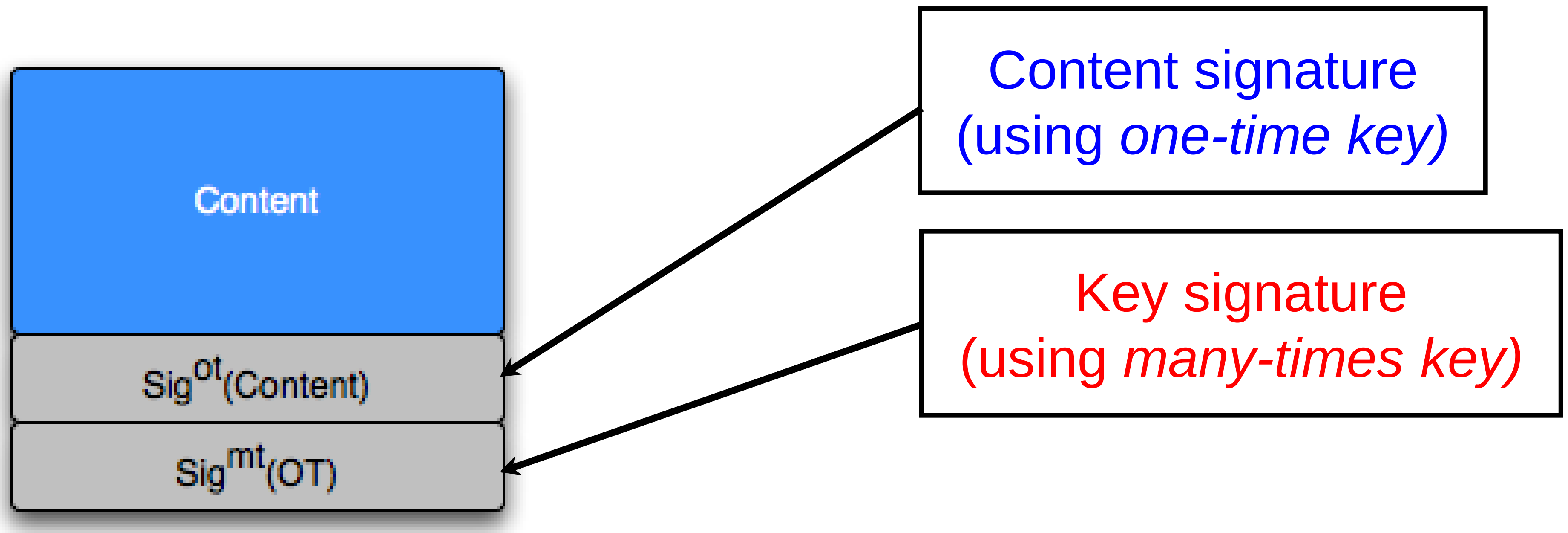
Integrity Guaranteed AJAX?

Latency is the problem in Web 2.0.

Off-line/on-line Signatures

PENNSTATE 1855

- New cryptographic constructions, relying on two types of digital signatures
  - *Many-times* signature schemes, e.g. RSA
  - *One-time* signature schemes, e.g. Lamport
- *Intuition*: Use many-times key to sign one-time keys (*slow*), use one-time keys to sign content (*fast*)



Binding AJAX Requests

PENNSTATE

- Use off-line/on-line scheme to bind AJAX content to system integrity state
  - Generate one-time keys *before* content generation
  - Bind one-time keys to system integrity proof

Off-line phase

  - Sign *dynamic content* with one-time keys when content is generated

On-line phase

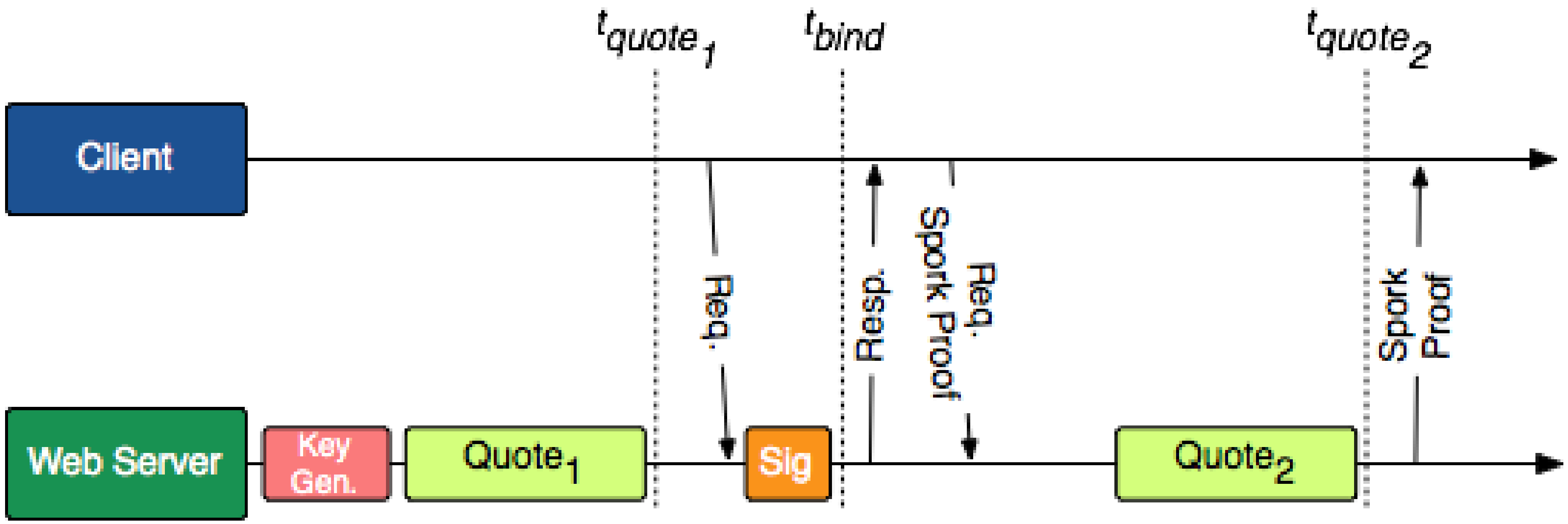
$$\underbrace{Q(H_{ws}, pcr_{ws}, h(CPS_r|VK|Q(H_{ts}, pcr_{ts}, h(t_i))))}_{\text{Web server quote (content proof + time quote)}} |\underbrace{CPS_r}_{\text{Proof sys. root}}|\underbrace{Pf(\pi)|\pi|\sigma|VK|VK^{ot}}_{\text{Dynamic content proof}}|\underbrace{Q(H_{ts}, pcr_{ts}, h(t_i))|t_i}_{\text{Time server quote}}| \underbrace{M_{ws}|M_{ws}}_{\text{IMA measurement lists}}$$

Web server quote (content proof + time quote) — Proof sys. root — Dynamic content proof — Time server quote — IMA measurement lists

  - Cryptographic proof system (hash tree) is now
    - Static content tree
    - *Tree of one-time keys*

TECHNIQUE: Create "fast signing" keys beforehand using TPM and sign as AJAX responses are served.

Spork Proof



- Proof only shows measurements through quote (t_{quote_1})
  - Binding occurs after quote (t_{bind}), not at the same time
- Bind content to next quote (t_{quote_2})
  - Rollback on failure

Weaker Quote Semantics: the system H_w running known software (indicated in the PCR register) created key k at time ti delivered document p_i signed with k.

Macrobenchmarks

	Content		Proof	
	Thpt.	Lat.	Thpt.	Lat.
Baseline	6134.4	80.8	-	-
GHR-DL	384.2	358.9	381.1	316.1
GHR-DL2	390.7	558.6	387.6	256.2
CS-DL	270.8	984.1	266.8	531.7
CS-DL2	274.5	713.6	270.9	415.1

- AJAX updates are 2.5KB per request
- Baseline latency is 80.8 milliseconds
- Sporf latency is between 360 and 1000 milliseconds
  - Neilsen [Nie99] describes a “usable” web application as one that responds in *under a second*

Summary

- Fundamental misconception: security provided by SLL and server administration do not provide the security needs for high-value systems.

- Bottom line: we are moving towards broader definition of web security that encompasses the *authenticity* and *integrity* of documents.

- Lesson: hardware assistance is coming.

Thanks!

Patrick McDaniel (mcdaniel@cse.psu.edu)

http://www.patrickmcdaniel.org/

SIIS Laboratory (http://siis.cse.psu.edu)