WSKE: Web Server Key Enabled Cookies

Chris Masone
with
Kwang-Hyun Baek and Sean W. Smith
Department of Computer Science
Dartmouth College

Outline

- Motivation
- WSKE
- Design
- Implementation
- Evaluation
- Related work
- Conclusions

Motivation

- Web app designers want to improve
  - Authentication usability
  - Phishing resistance
- One strategy: secure cookies
  - Disclosure resistant
  - "Same origin policy"
  - Set, released only over SSL/TLS
  - Usually encrypted w/site specific secret

Secure Cookie Issues

- Subject to replay attacks
- Cross-Site Scripting (XSS)
  - can be prevented by proper site construction
  - addressed in other work
- Pharming
  - Attacker can spoof DNS
- IP attacks (BGP)
  - Attackers can cause re-routing of IP traffic
  - Yes, this is seen in the wild

Secure Cookie Issues

- Subject to replay attacks
- Cross-Site Scripting (XSS)
  - can be prevented by proper site construction
  - addressed in other work
- Pharming
  - Attacker can spoof DNS
- IP attacks (BGP)
  - Attackers can cause re-routing of IP traffic
  - Yes, this is seen in the wild

...Cookies are in use, we should protect them!

Server-Side SSL

- SHOULD protect against DNS, IP spoofing
- A myriad of dialog boxes
  - mismatched domain name
  - unknown issuer for server certificate
  - makes secure cookies less usable for authentication
- Users trained to click through
- If warning, then no cookies
  - ~60% of SSL servers misconfigured
  - Sites cannot choose to go self-signed
  - Ideal solution avoids "breaking the web"

Properties of a Solution

- Leverage crypto
- Users shouldn't need to understand
- Limit impact on deployed sites
- Avoid server-side config changes
- Minimize user-side requirements

WSKE

After cookies set via SSL, WSKE binds them to server of origin and server's public key

- No user interaction
- Web apps don't need to change
- Misconfigured SSL OK
- Covers a network-based attacker
- Key expiration potentially an issue

WSKE: Note...

- WSKE does not address registration
- Registration hard, addressed elsewhere

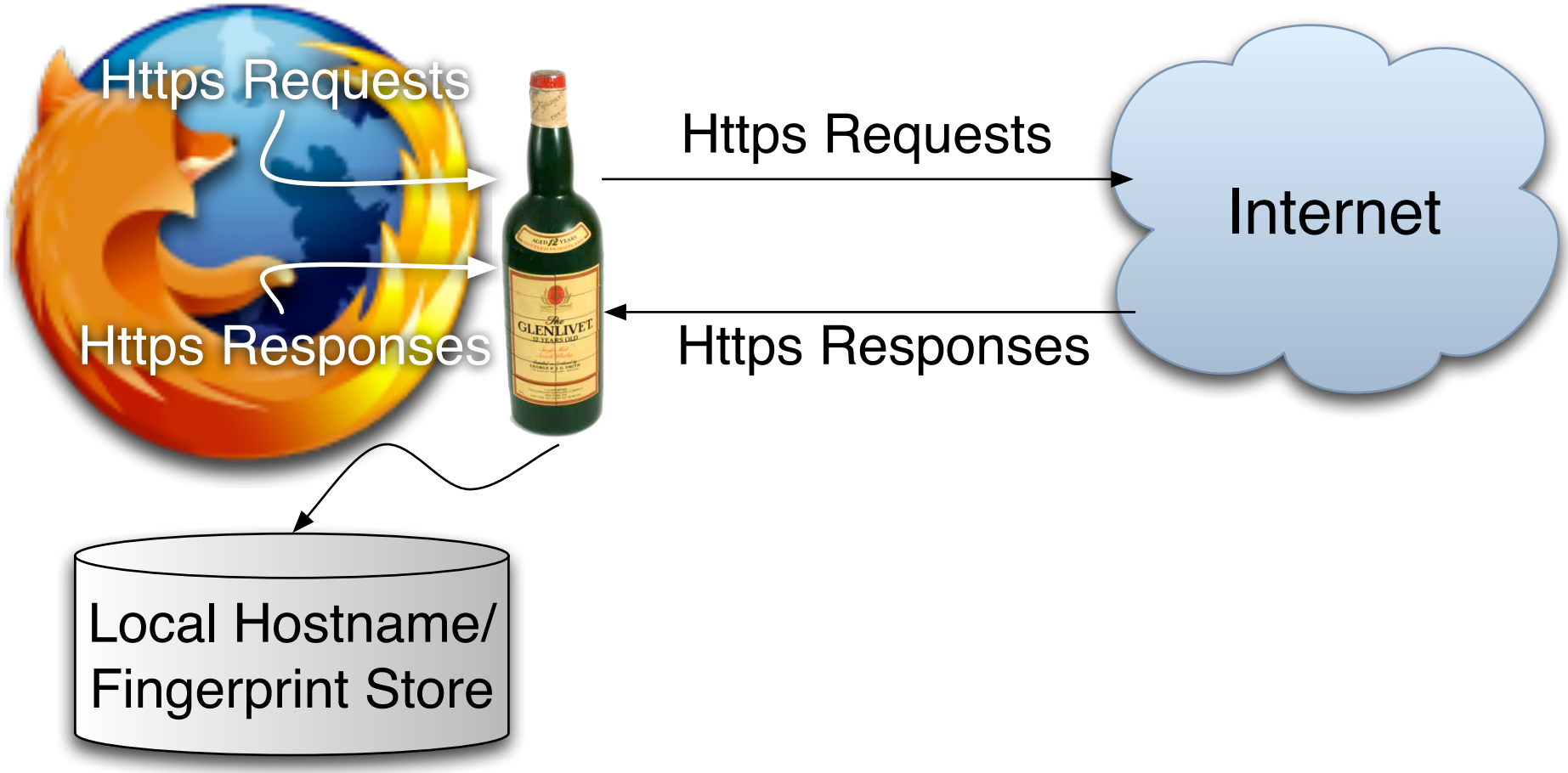
- WSKE simple, deployable now
  - Users careful about SSL signals once, then protected
  - Same trust model as SSH
  - Combine with more complex registration method

Prototype Design

- Man-in-the-middle at client
- When cookies are set:
  - Remember hostname
  - Remember server's SSL key fingerprint
  - Bind cookie to these values
- Just before cookie release:
  - Verify hostname (browsers do this already)
  - Check current SSL key against stored fingerprint
  - Release cookies only if key matches

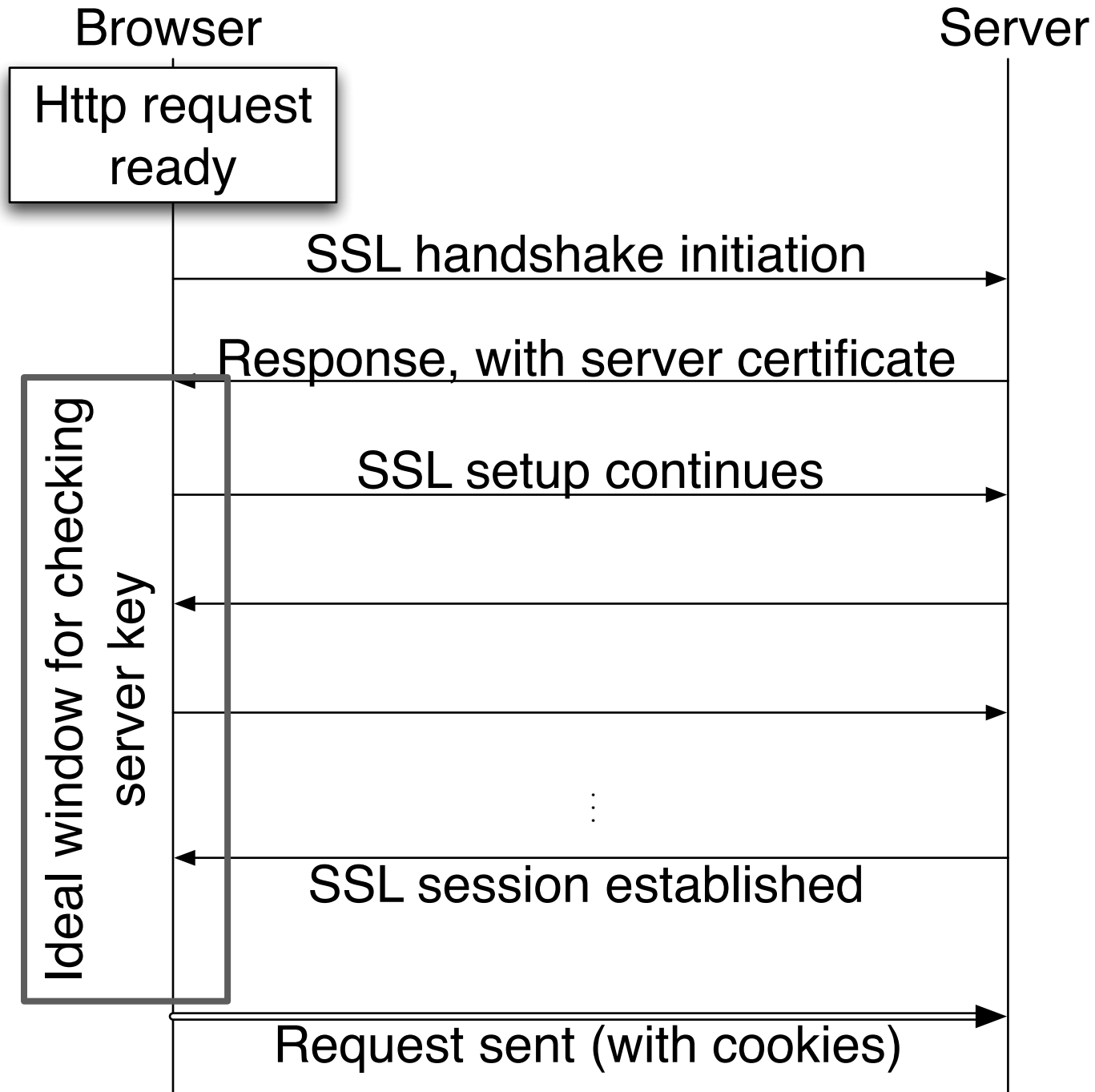
Prototype Implementation

- Firefox extension

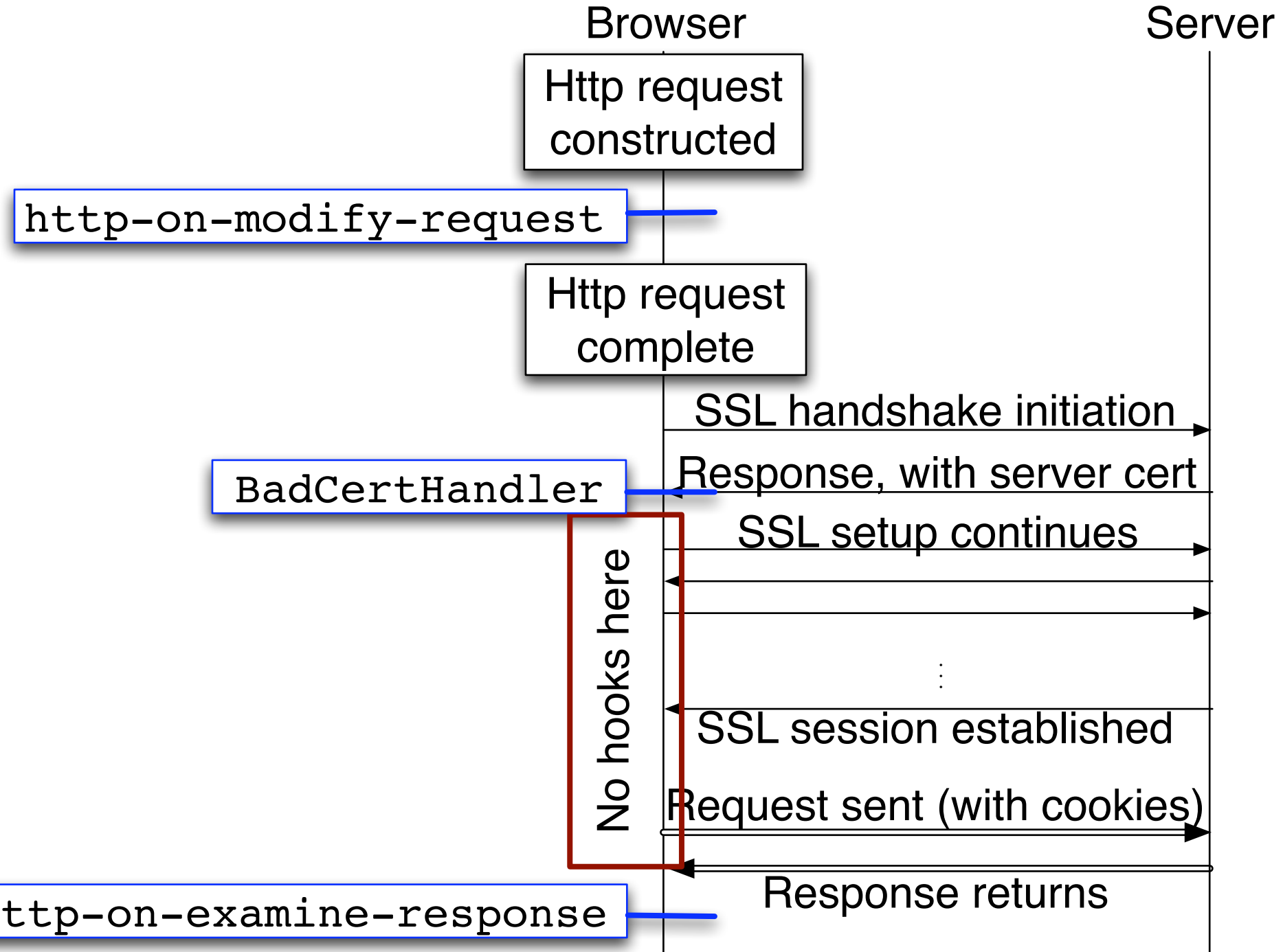


- JavaScript cookie access left for future work

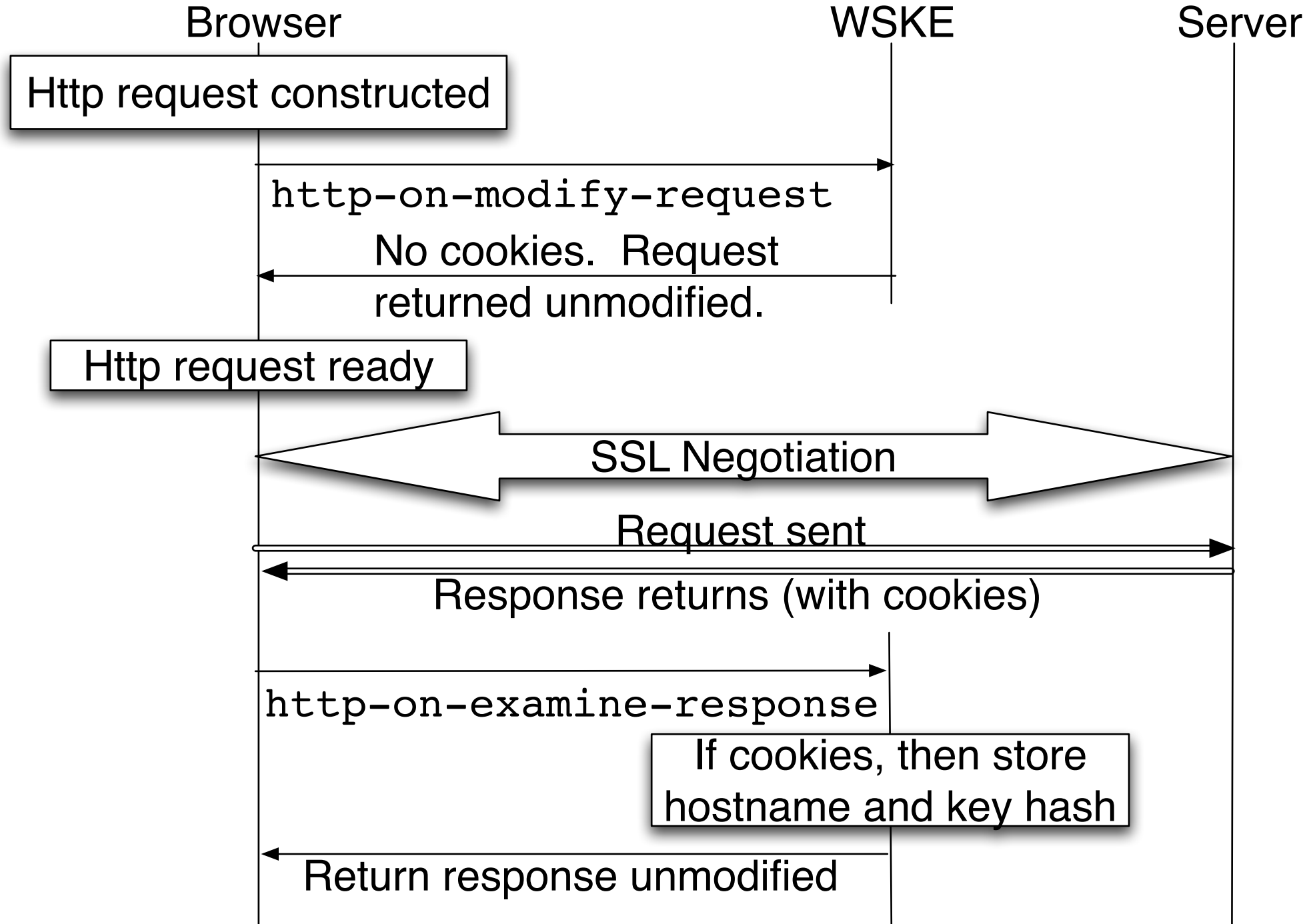
Prototype Implementation



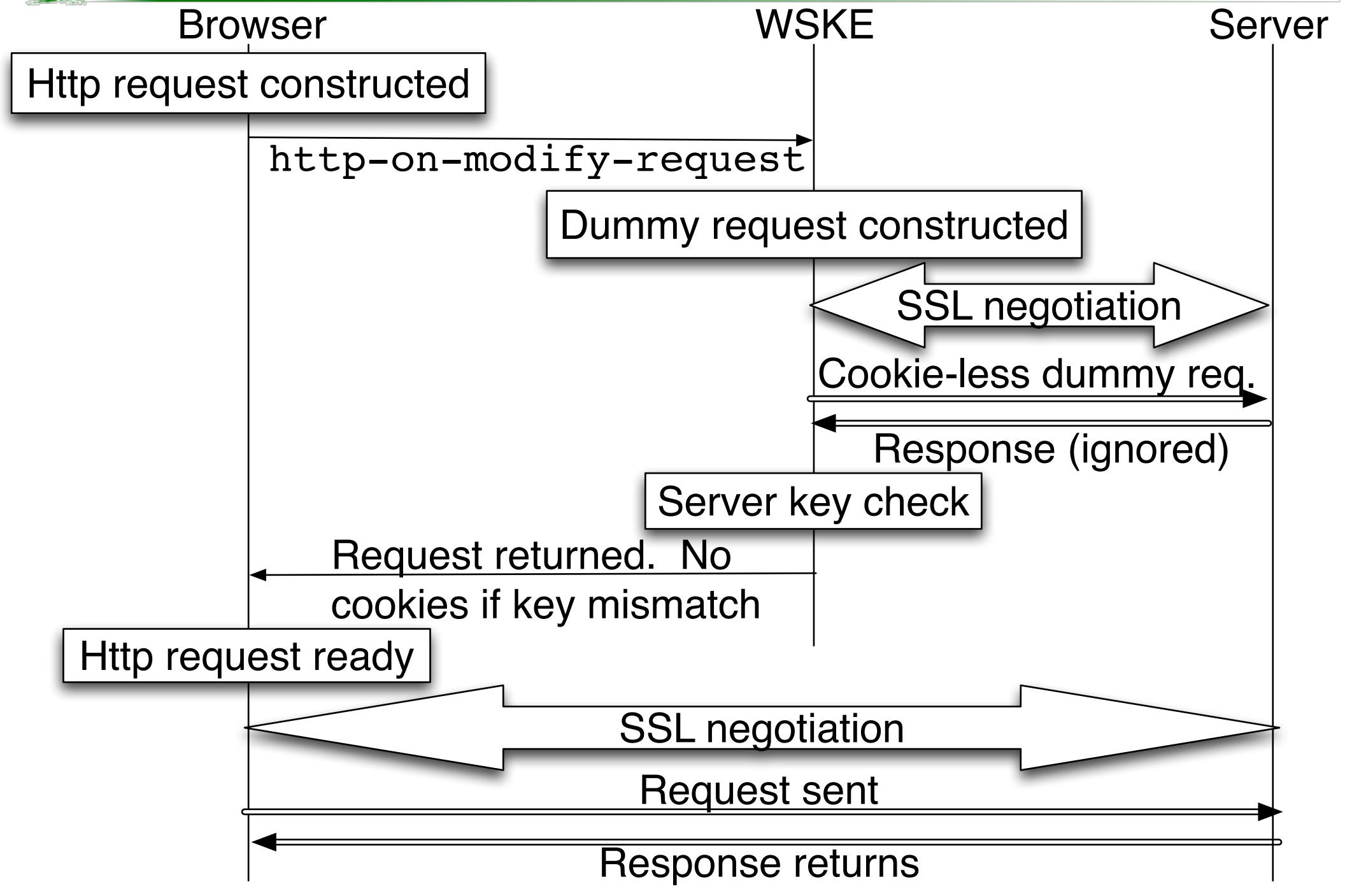
Prototype Implementation



Prototype Implementation



Prototype Implementation



Evaluation

- Attack resistance
  - Testbed: 2 webservers, BIND, and a client
  - Cookies blocked in simulated DNS attack
  - Cookies blocked in simulated IP-spoof attack
- Deployability
  - Web Apps need not know about WSKE
  - Load-balancing, new server keys could be problem
  - Possibly bind to CA key instead of server key
- Usability
  - Users only need to look at SSL cues once
  - If spoofing, credentials *cannot* be released
  - Is there a re-registration attack?

Related Work

- Locked Cookies
  - Contacted by authors after WSKE accepted to USEC
  - Same concept, implementation modifies binary
  - Published as a tech report
- Active Cookies
  - Requires server-side changes, no client-side code
  - Binds cookies to numeric IP addresses
  - Vulnerable to IP-based attacks
- Phone-based schemes
  - Phoolproof, Mannan & van Oorschot
  - Require an external device, server and client changes
  - Perhaps overkill for some sites

Conclusions

- WSKE could be deployed today
- Server-side SSL made more usable
- Cookie-based auth made more secure
- Prototype works, but could be cleaner

- More rigorous usability evaluation?

Thanks!

Questions?