Improving the Security of Session Management in Web Applications

Philippe De Ryck, Lieven Desmet, Frank Piessens, Wouter Joosen

Philippe.deryck@cs.kuleuven.be

Session Management

Server ties multiple requests together

Enables consecutive requests

Allows storage of session information

- E.g. **Authentication status**

Known by a ***session identifier***

Session identifier (SID) included in every request

Allows lookup of correct session state

SID effectively acts as a ***bearer token***

Session Identifiers as Bearer Token

Multiple deployment scenarios

`Cookie: PHPSESSID=a8914ka`

`http://example.com?`**`PHPSESSID=a8914ka`**

Flawed by design

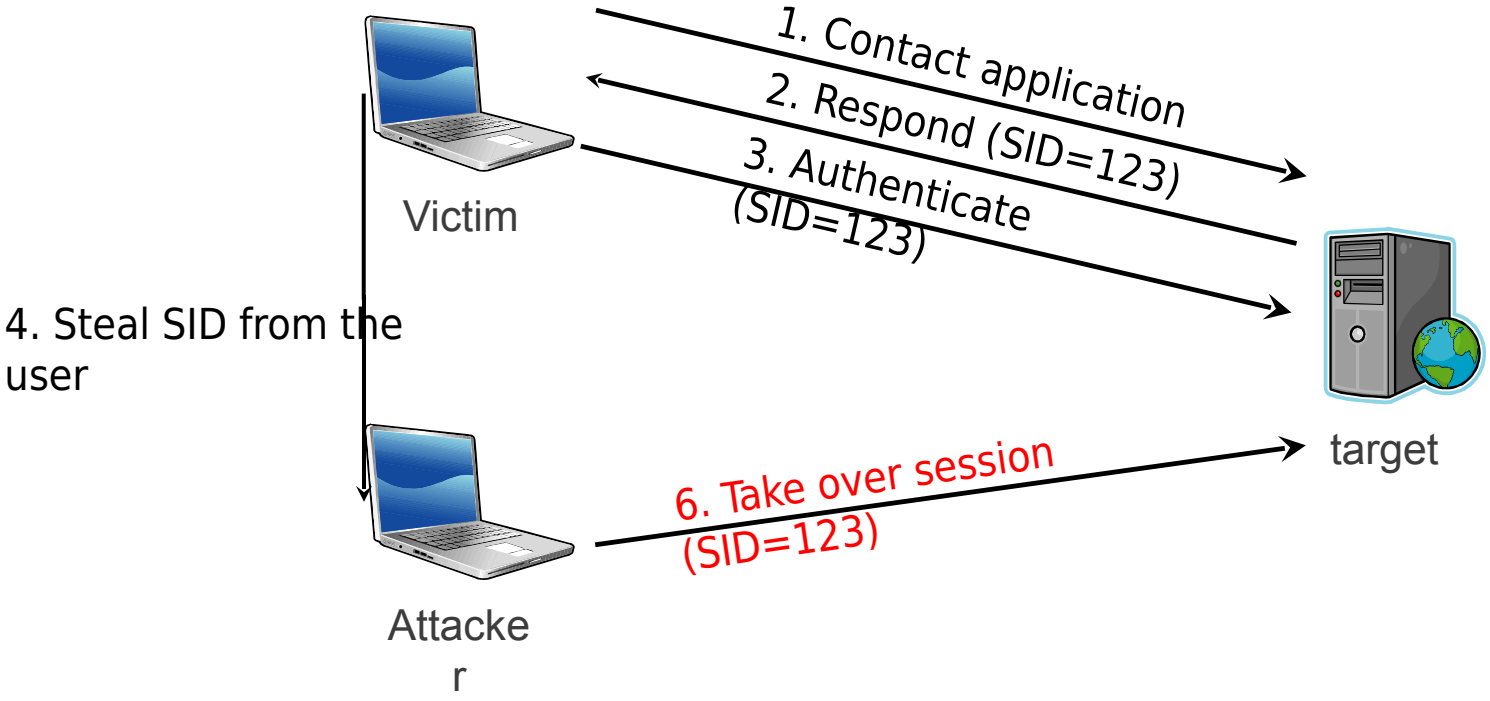
- Holder of the token controls the session
- SIDs are typically easy to obtain in a web context
- #2 in OWASP top 10 (2013)
- Illustrated by numerous attacks

Attacks on Session Management

Session Hijacking

- Full takeover of user's session
- Especially powerful after user authentication
- Multiple attack vectors (JavaScript, Eavesdropping)

Session Hijacking



Attacks on Session Management

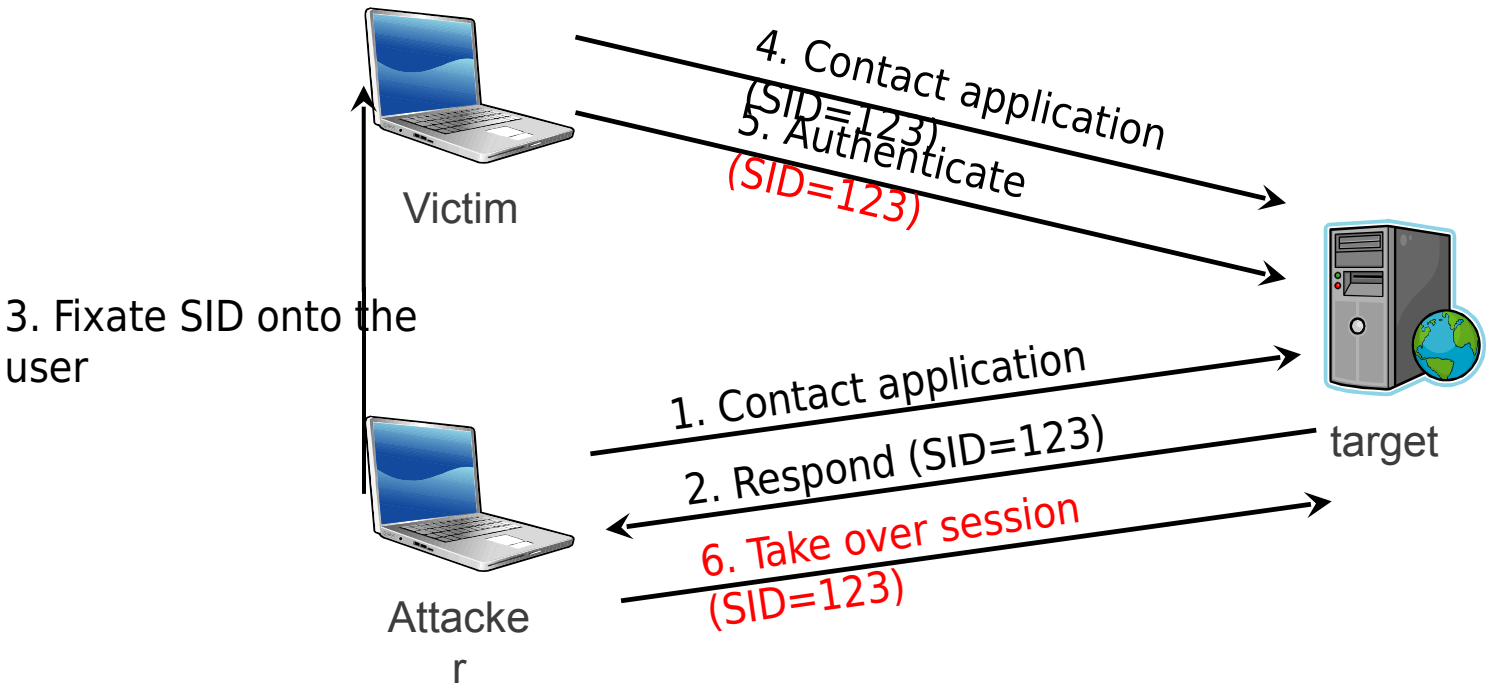
Session Hijacking

- Full takeover of user's session
- Especially powerful after user authentication
- Multiple attack vectors (JavaScript, Eavesdropping)

Session Fixation

- Force user to work with attacker's session
- Less known than session hijacking
- Multiple attack vectors (JavaScript, Meta-tag, related domain, simple URL)

Session Fixation



Protecting Session Management

General advice

- Strong, unique session identifiers
- Rotate after privilege change
- Deploy your site over HTTPS

Specific for cookie-based systems

- Use ***HttpOnly*** flag
- Use ***Secure*** flag
- Limit lifetime

So Problem Solved …?

Limited deployment

HttpOnly and Secure not often used

HTTPS deployments are also limited

- Often in insecure combination with HTTP

Many reasons, mainly speculation

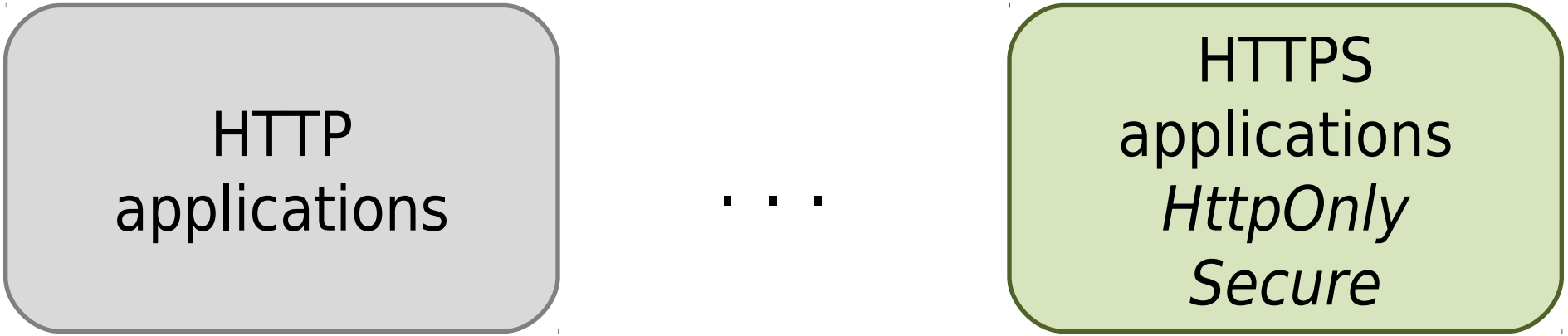
Uninformed developers?

Certificate complexity?

Interaction with middleboxes (caches, IDS, …)

Not needed (e.g. when using an authentication provider)

Problem Statement



Ideally

Secure session management

HTTPS deployment for further security guarantees

- Entity authentication, confidentiality, integrity

Running Example

Web application using 3rd party authentication

OpenID, Facebook, Google, …

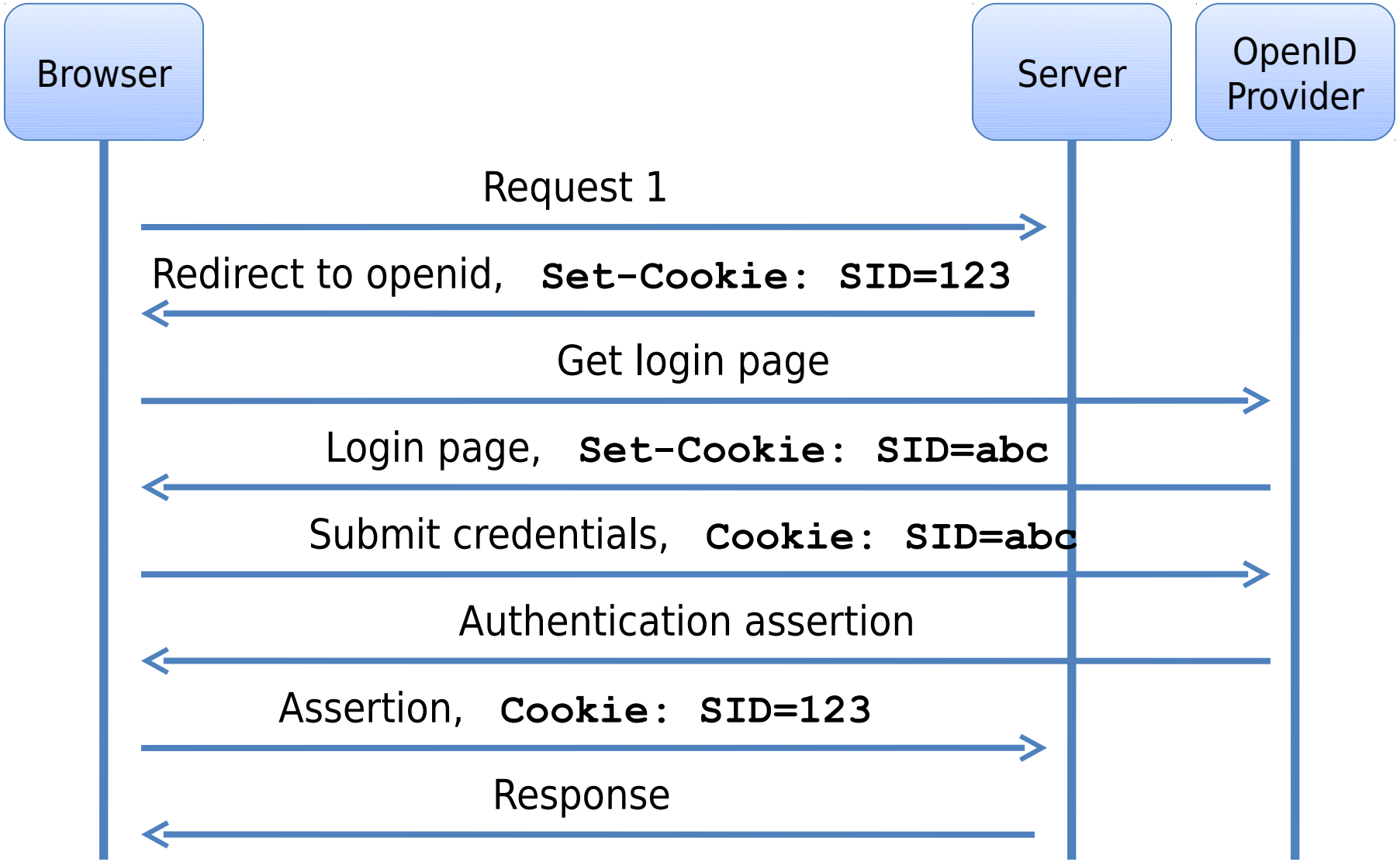
Outsources security-sensitive part

Remaining part has no need for confidentiality

- HTTPS deployment may be deemed unnecessary

Vulnerable session management

Running Example



Our Proposal

Secure session management

Ensure that a session remains between both initiators

Be resilient against

- Eavesdropping
- In-application attacks

Support scenarios with 3rd party authentication

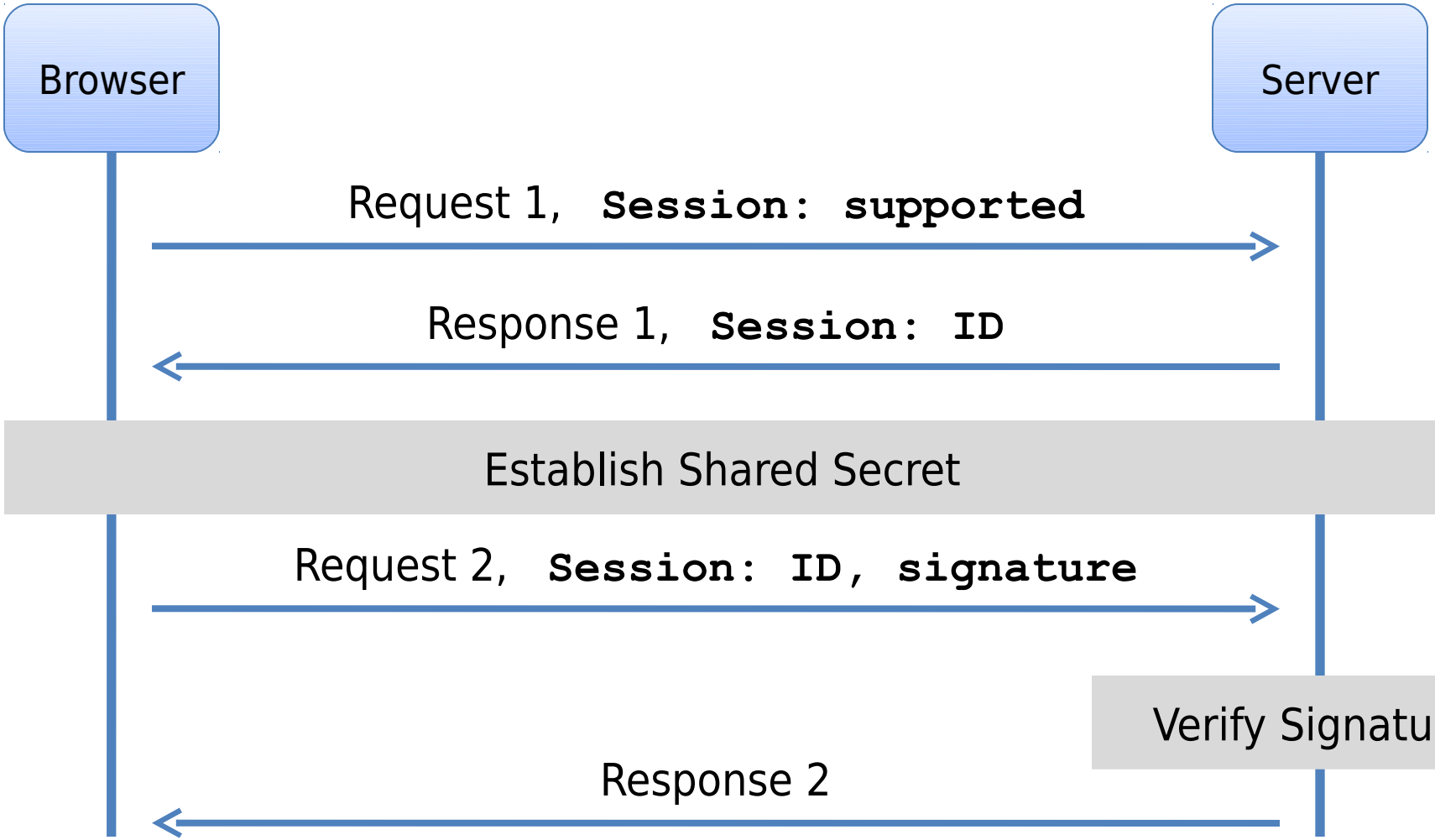
High-level overview

Establish shared secret using Hughes variant of DH

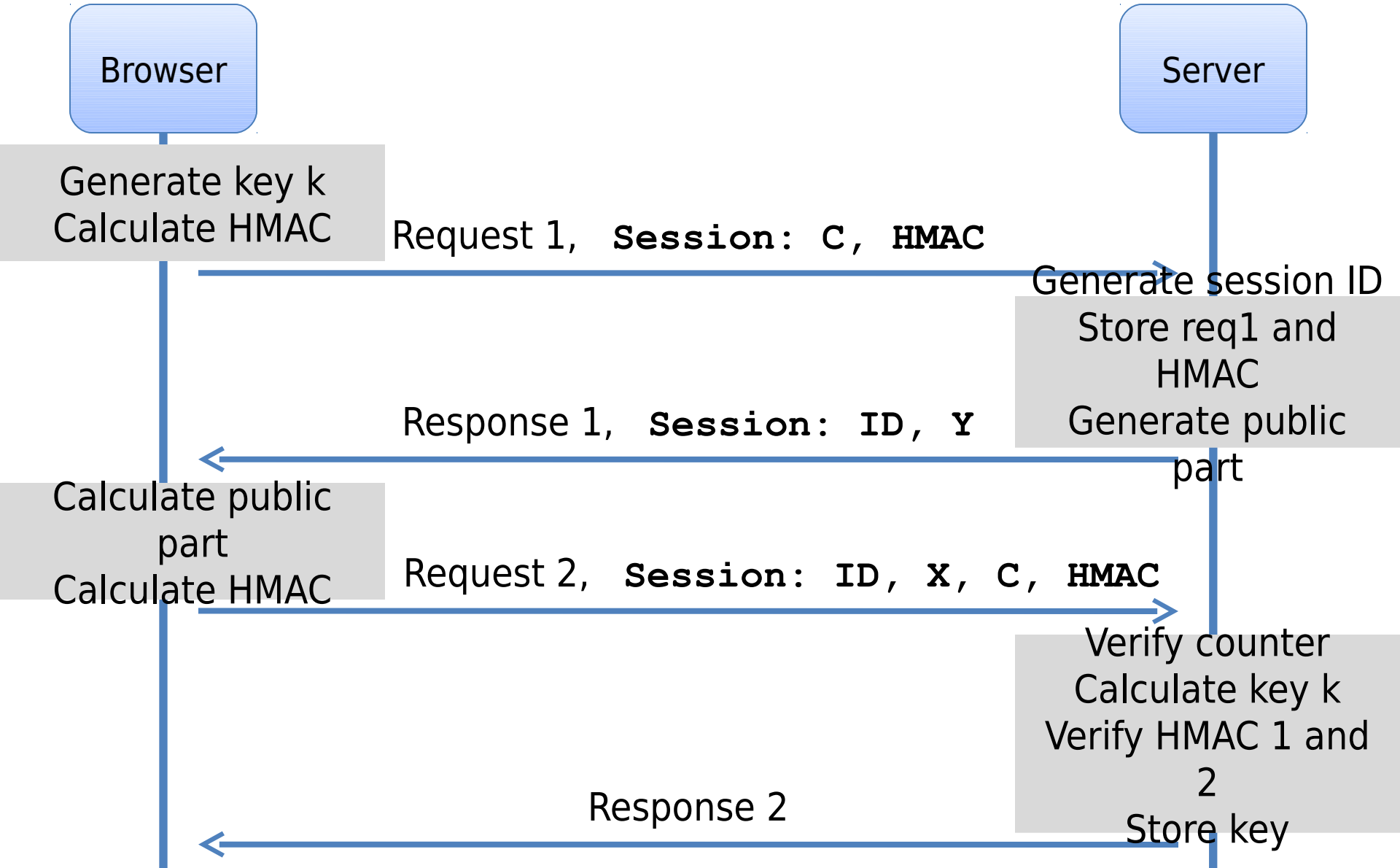
Use ***session*** header with signature

Secret locked in browser, so unreachable

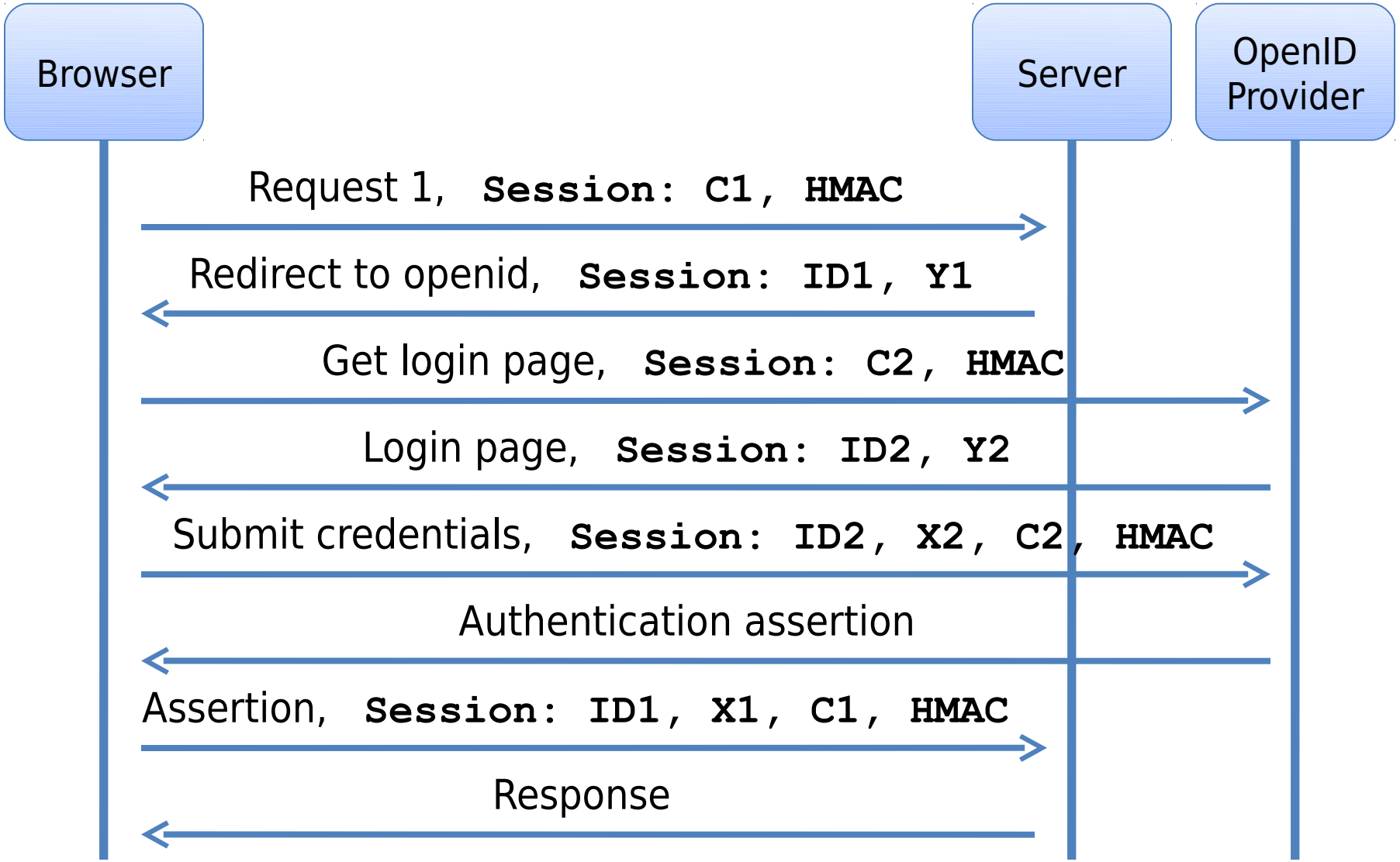
Secure Session Management



Secure Session Management (2)



Running Example Revisited



Required Infrastructure Support

Add support for *Session* header field

- Default session management mechanism in frameworks
- Browser support required
- Cookie-based session management as fallback

Legacy applications

- Server-side proxy translates cookies to *Session*

Scenarios for Secure Session Management

Combining HTTP and HTTPS

- Web app switches to HTTPS for sensitive operations
- **Bearer token of shared session becomes vulnerable**
- No problem for secure session management

HTTP Only applications

- Beyond vulnerable, but occurs in practice
- Protecting against Active Network Attackers
- Combine secure session management with TLS

Related Work

SessionLock

- Secret fetched from server or calculated with DH
- Depends on JS in the page to protect requests
- Incompatible with complex applications

HTTP Integrity Header

- Establishes secret key over TLS or with traditional DH
- Adds integrity to selected parts of message

Related Work

BetterAuth

- Calculates secret locally based on password
- Incompatible with third-party authentication
- Depends on HTTPS for initial setup

TLS Origin Bound Certificates

- TLS extension enabling browser certificates
- Supports binding of tokens to channel (i.e. cookies)

Conclusion

Secure session management

- Inherently fixes session management
- Replaces bearer token with signature
- Compatible with third party authentication providers
- Backwards compatible with legacy applications

Thank you

philippe.lederyck@cs.kuleuven.be

Acknowledgements

With the financial support from the Prevention of and Fight against Crime Programme of the European Union.