



THE COMPUTER SECURITY GROUP AT UC SANTA BARBARA

Protecting Web-based Single Sign-on Protocols against Relying Party Impersonation Attacks through a Dedicated Bi-directional Authenticated Channel

Yinzhi Cao `yinzhi.cao@eecs.northwestern.edu`

Yan Shoshitaishvili `yans@cs.ucsb.edu`

Kevin Borgolte `kevinbo@cs.ucsb.edu`

Christopher Kruegel `chris@cs.ucsb.edu`

Giovanni Vigna `vigna@cs.ucsb.edu`

Yan Chen `ychen@cs.northwestern.edu`

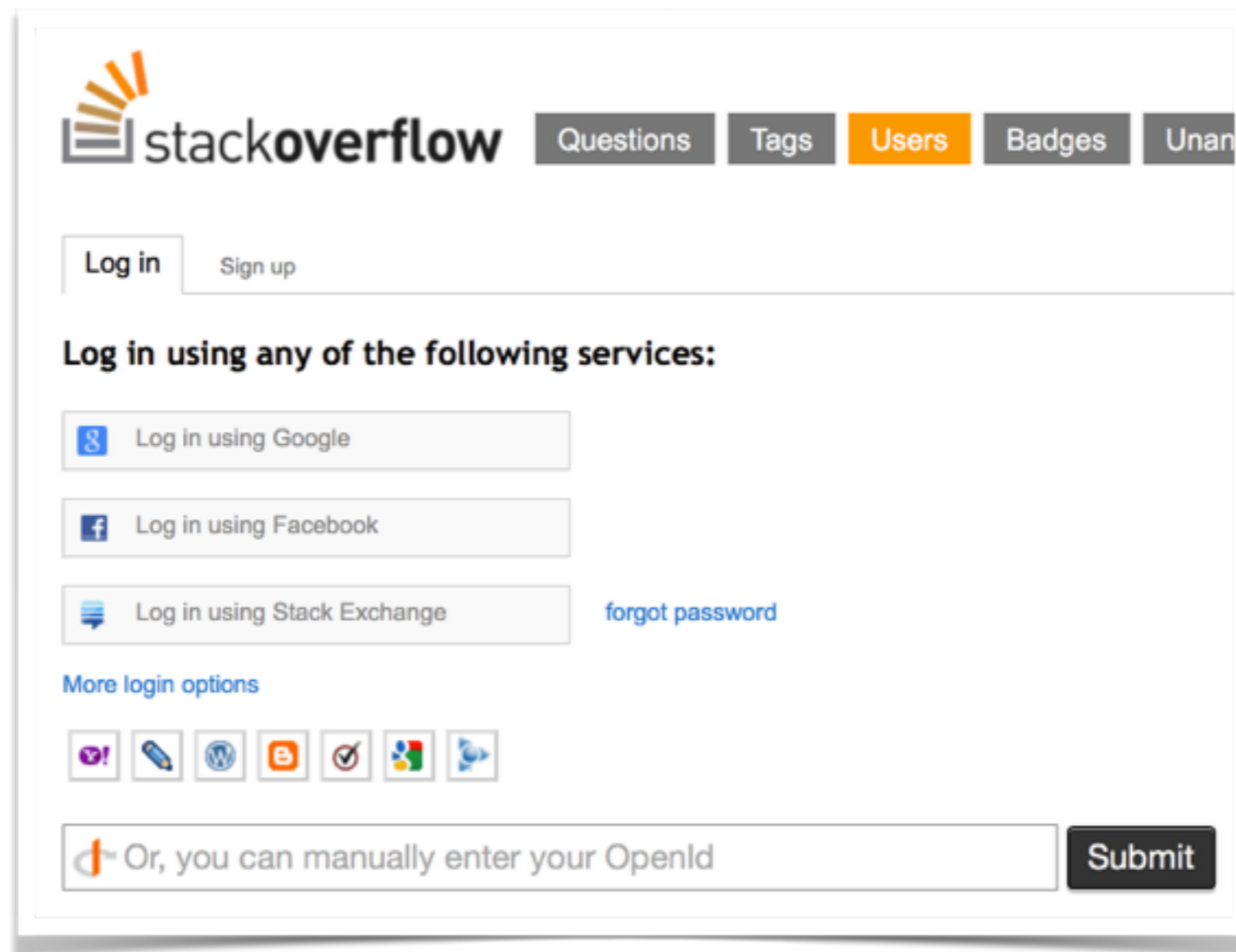
University of California, Santa Barbara
Northwestern University

September 17th, 2014

- Single Sign-on
- Threat Model
- Problems with Existing Designs
- Our Design
- Evaluation

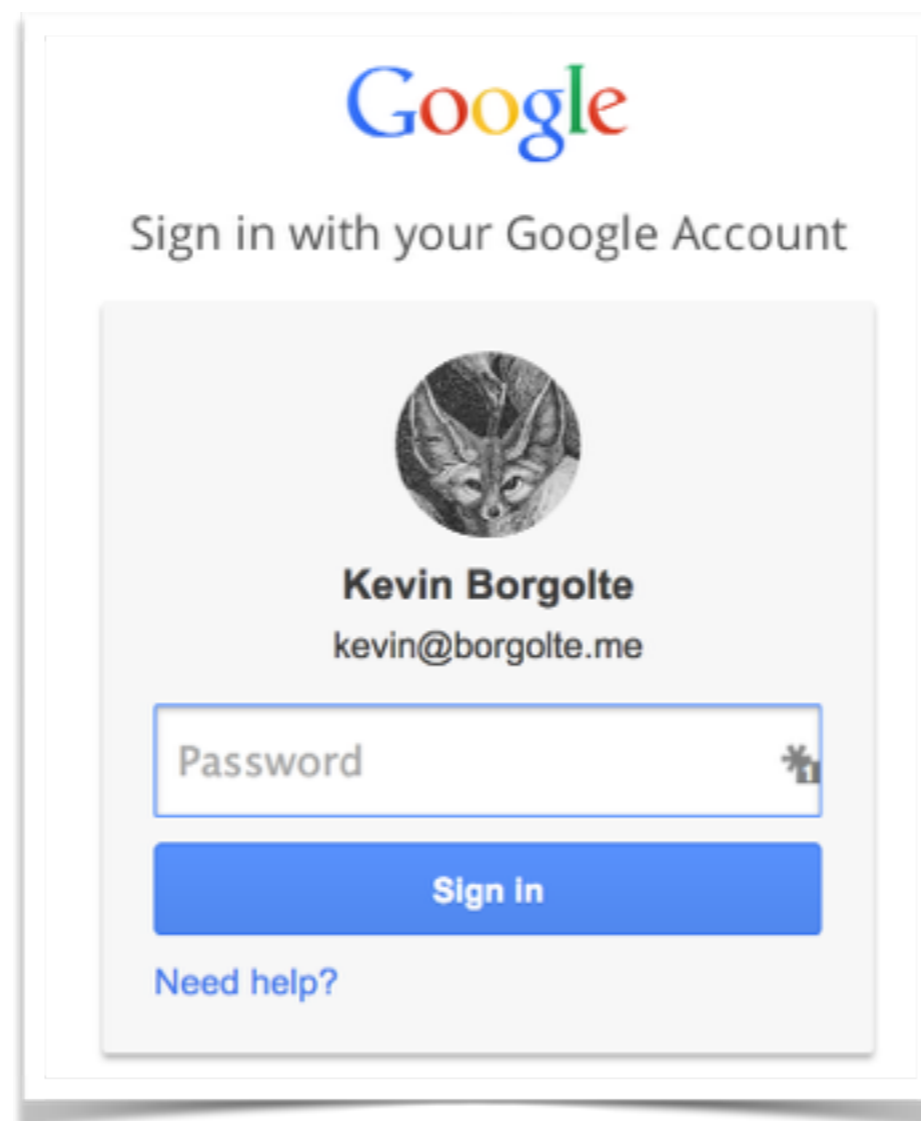
- Idea: log in to a website with your Facebook, Google, etc. account

- Idea: log in to a website with your Facebook, Google, etc. account

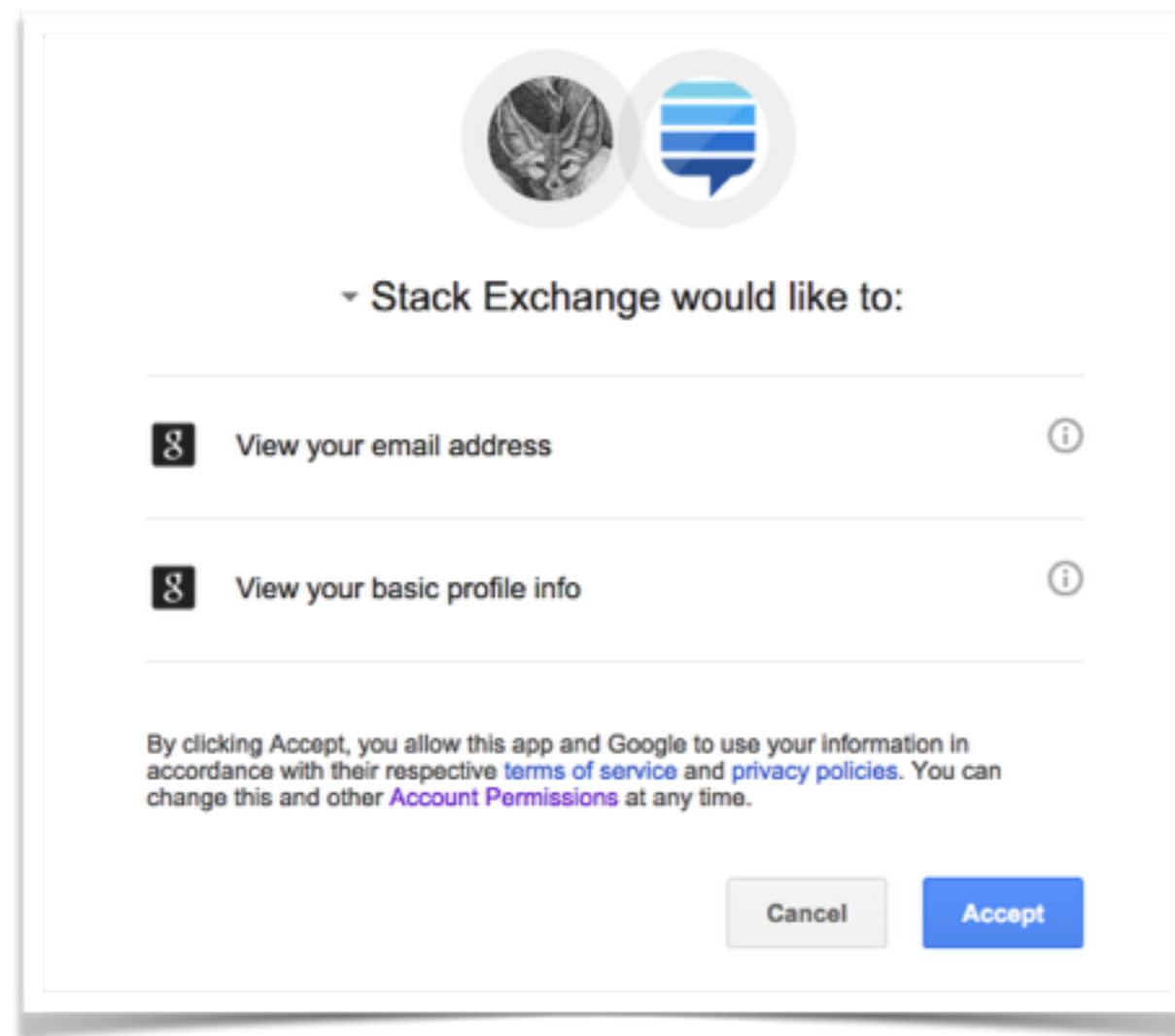


The screenshot shows the Stack Overflow login interface. At the top, the Stack Overflow logo is on the left, and navigation tabs for 'Questions', 'Tags', 'Users', 'Badges', and 'Unanswered' are on the right. Below the navigation, there are 'Log in' and 'Sign up' buttons. A heading reads 'Log in using any of the following services:'. Below this, there are three main login buttons: 'Log in using Google', 'Log in using Facebook', and 'Log in using Stack Exchange'. To the right of the 'Log in using Stack Exchange' button is a link for 'forgot password'. Below these buttons is a link for 'More login options' which leads to a row of seven small icons representing various social and identity providers. At the bottom, there is a text input field with the placeholder text 'Or, you can manually enter your OpenId' and a 'Submit' button.

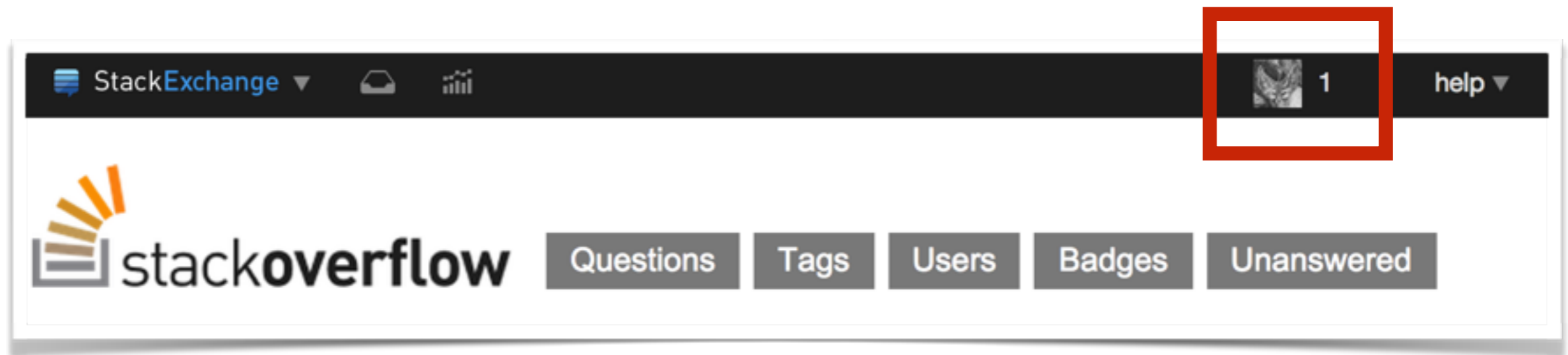
- Idea: log in to a website with your Facebook, Google, etc. account



- Idea: log in to a website with your Facebook, Google, etc. account



- Idea: log in to a website with your Facebook, Google, etc. account



Single Sign-on (SSO) (2)

OAuth 2.0 Flow

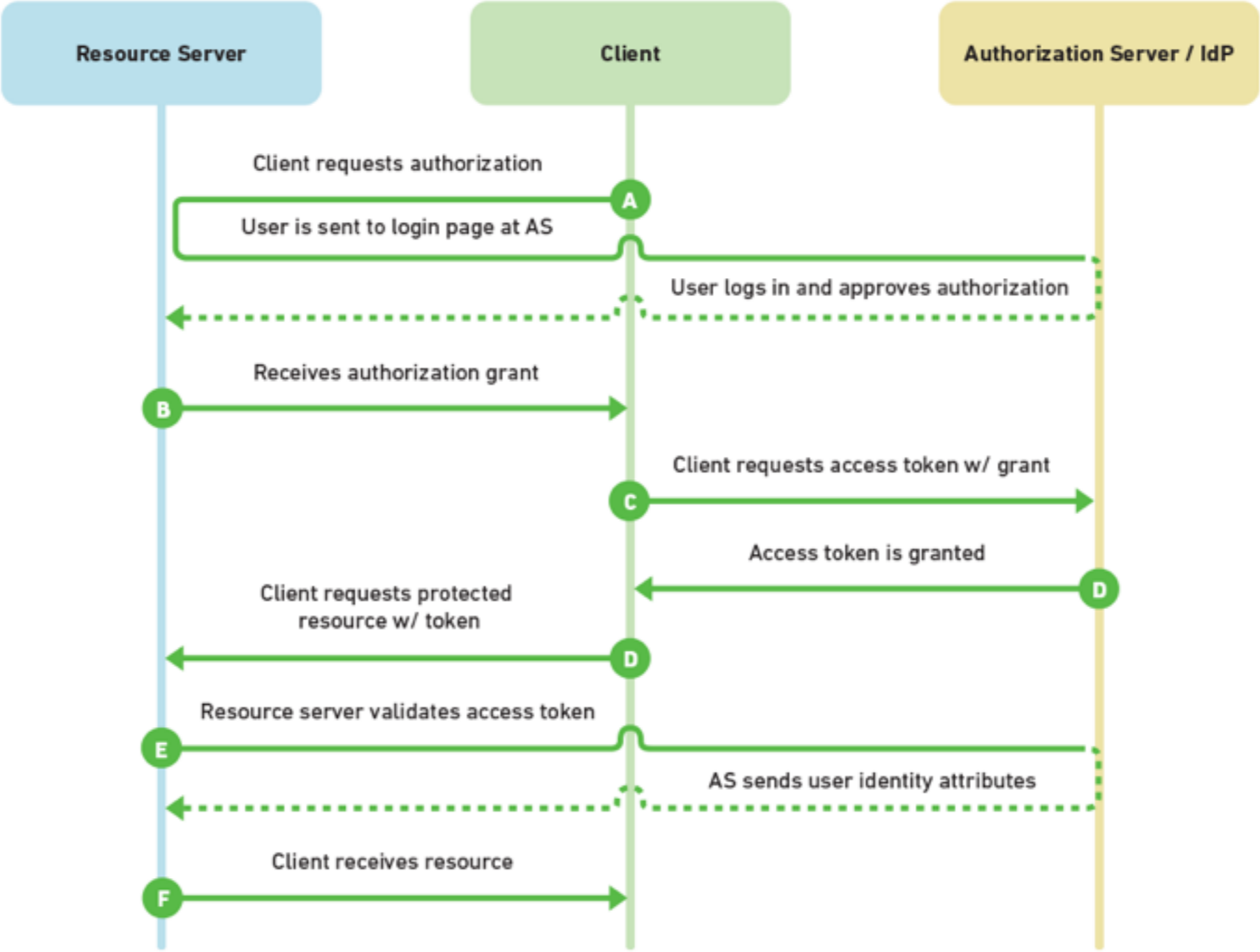


Image by Mutually Human, via <http://www.mutuallyhuman.com/blog/2013/05/09/choosing-an-sso-strategy-saml-vs-oauth2/>.

- SSO vulnerabilities mean
 - User impersonation
 - Data/privacy leaks

- SSO vulnerabilities mean
 - User impersonation
 - Data/privacy leaks
- Vulnerabilities are prolific
 - Wang et al. identified five vulnerabilities in which an attacker can impersonate a user [Oakland '12].
 - Sun et al. show that 6.5% of relying parties are vulnerable to impersonation attacks [CCS '12].

- Identity provider (IdP)
 - A centralized identification service
 - Trusted and benign
- Relying party (RP)
 - A third party using the IdP to authenticate users
 - Potentially malicious
- User
 - Wants to use the RP's service
 - Trusted and benign



Threat Model - Attacks (1)

- In-scope
 - Benign RP initiates request, malicious RP receives response

- In-scope
 - Benign RP initiates request, malicious RP receives response

```
GET https://www.idp.com/login?  
app_id=****&redirection_url=https://www.idp.com/granter?  
next_url=https://www.rp.com/login
```

```
Host: www.idp.com
```

```
Referer: https://www.rp.com/login
```

```
Cookie: ****
```

- In-scope
 - Benign RP initiates request, malicious RP receives response

```
GET https://www.idp.com/login?  
app_id=****&redirection_url=https://www.idp.com/granter?  
next_url=https://www.rp.com/login
```

```
Host: www.idp.com
```

```
Referer: https://www.rp.com/login
```

```
Cookie: ****
```

- In-scope
 - Benign RP initiates request, malicious RP receives response
 - Malicious RP initiates the attack

```
GET https://www.idp.com/login?  
app_id=****&redirection_url=https://www.idp.com/granter?  
next_url=https://www.rp.com/login
```

```
Host: www.idp.com
```

```
Referer: https://www.rp.com/login
```

```
Cookie: ****
```


- In-scope
 - Benign RP initiates request, malicious RP receives response
 - Malicious RP initiates the attack

⇒ Information leakage or user impersonation!

- Out-of-scope
 - Social engineering
 - Compromised or vulnerable RP
 - Malicious user (browser)
 - Implementation issues
 - Privacy leaks

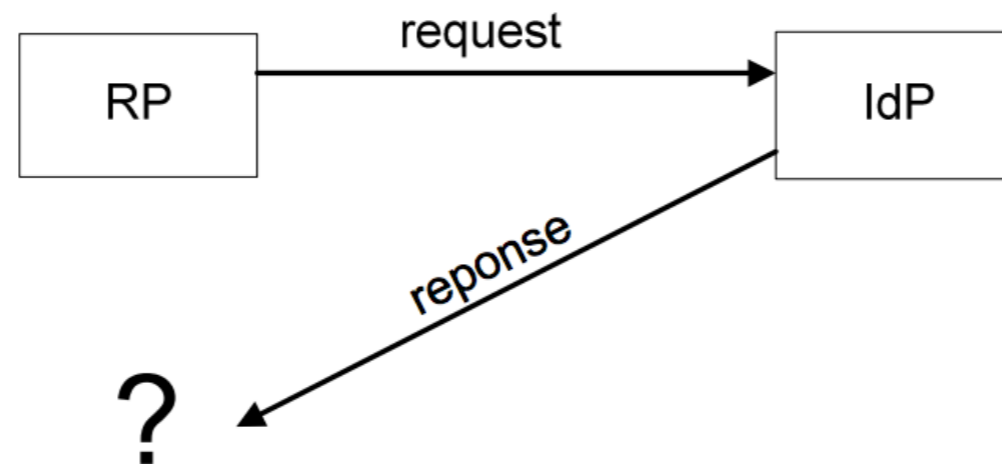
- Existing identities
 - IdP, usually web origin (<scheme, host, port>)
 - RP, unique identifier, depending on protocol, app_id or AppName
 - User, unique identifier like username or email address

- Existing identities
 - IdP, usually web origin (<scheme, host, port>)
 - RP, unique identifier, depending on protocol, app_id or AppName
 - User, unique identifier like username or email address

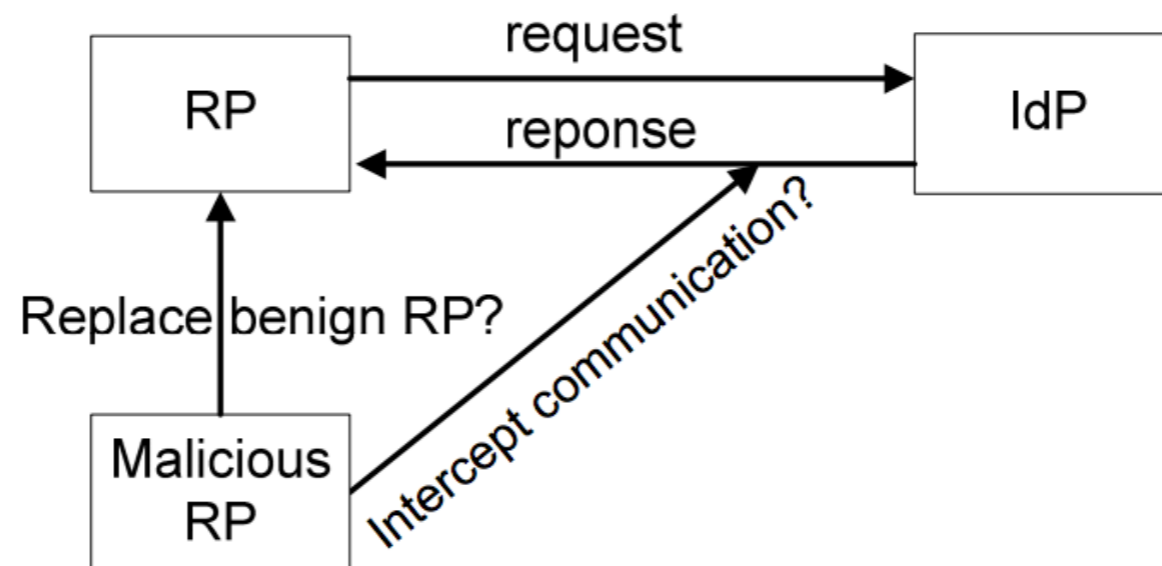
Main issue: RP identifier can be forged.

- Communication between RP and IdP

- Communication between RP and IdP
 - HTTP(s) redirection to 3rd party server (1-way channel)

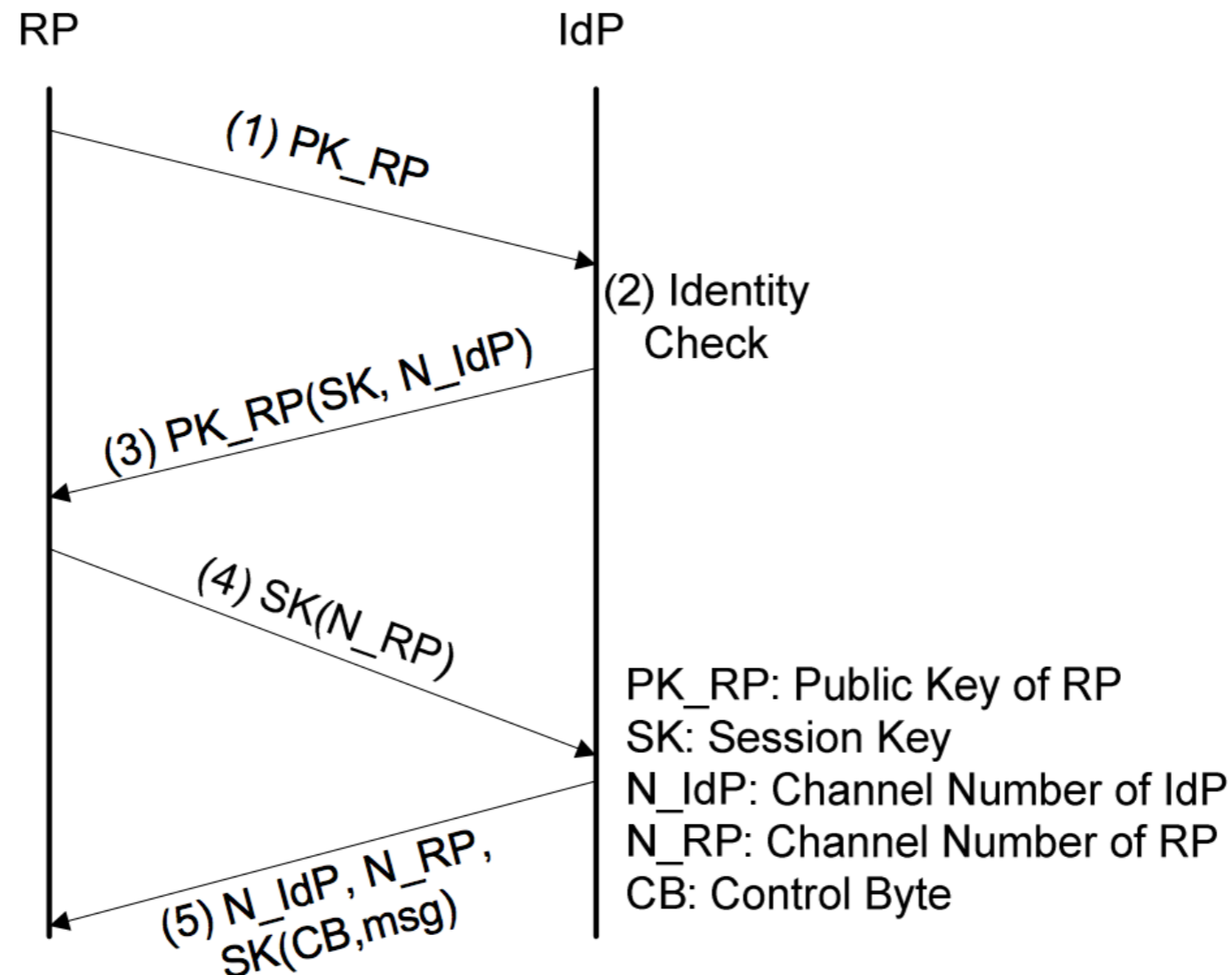


- Communication between RP and IdP
 - HTTP(s) redirection to 3rd party server (1-way channel)
 - In-browser communication channel (no authentication)

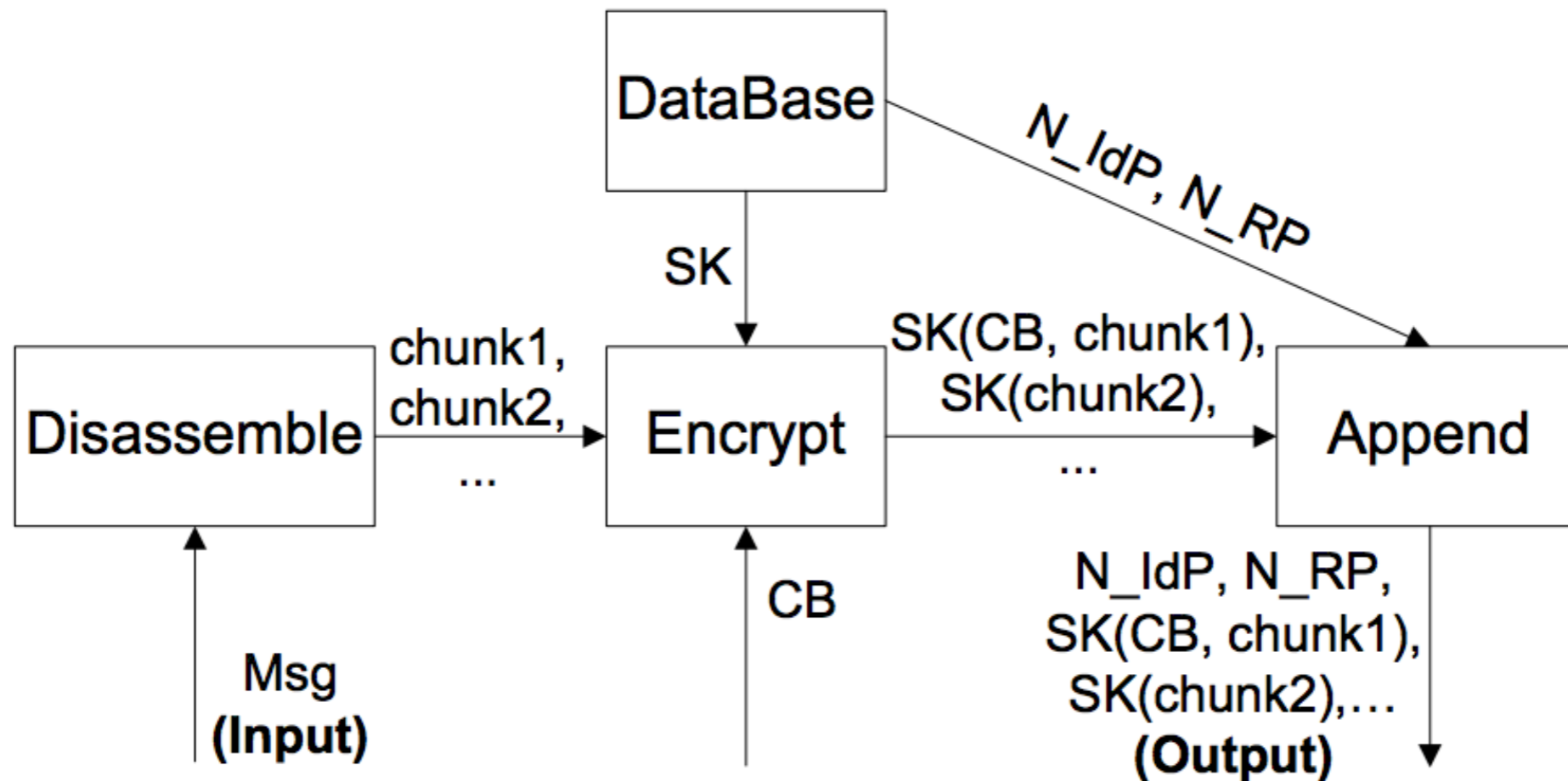


- Clean-slate design, replaces existing protocols
 - Identity
 - Web origin for RP and IdP: <scheme, host, port>
 - Communication channel
 - Dedicated
 - Bi-directional
 - Authenticated
 - Secure

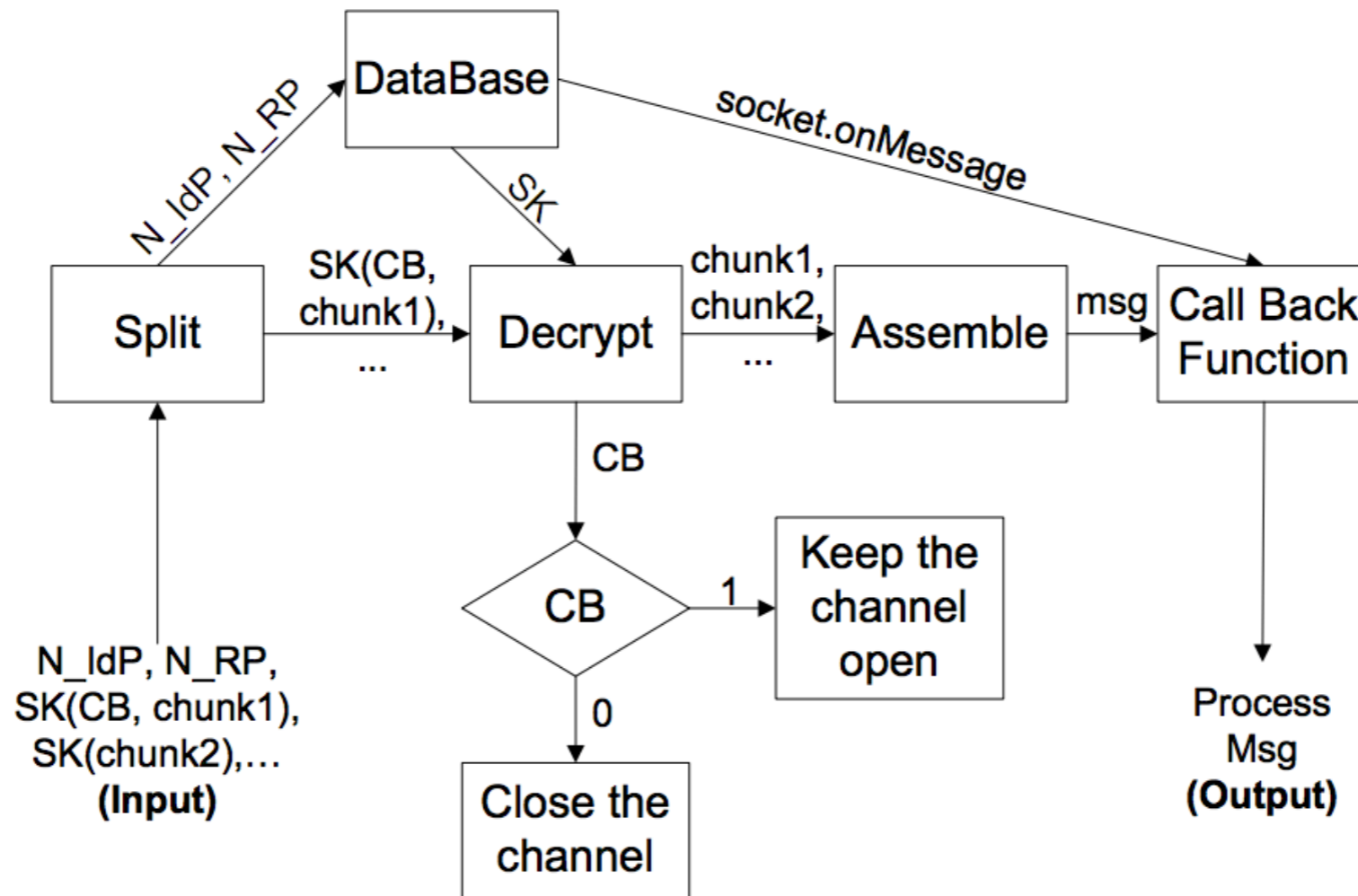
- Establishing the channel: handshake



- Establishing the channel: handshake
- Sending messages



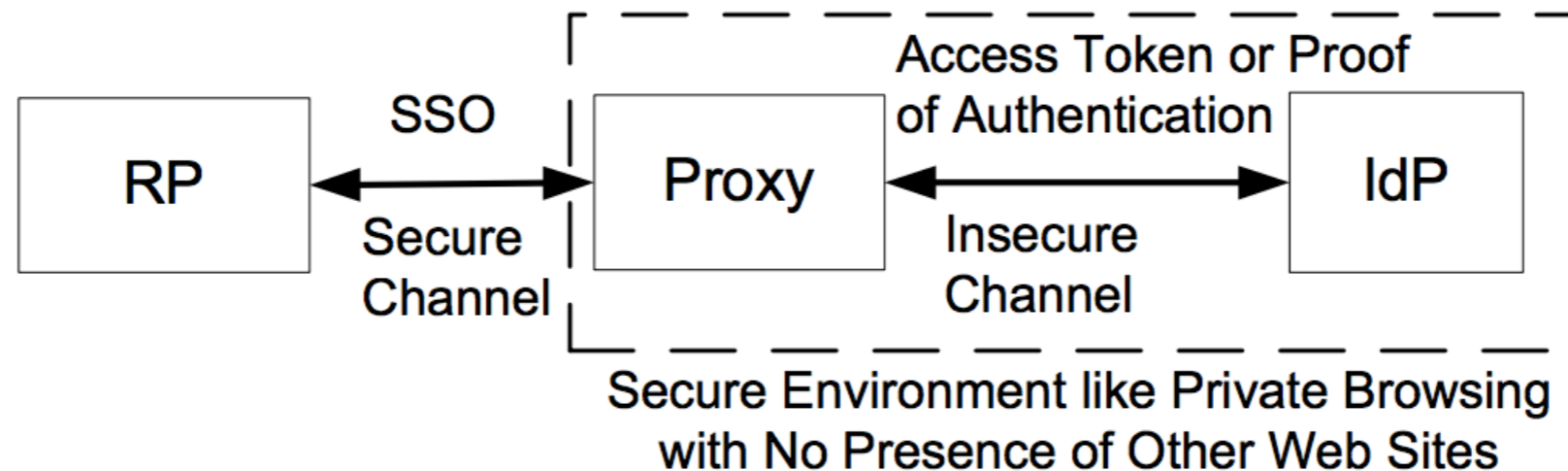
- Establishing the channel: handshake
- Sending messages
- Receiving messages



- Establishing the channel: handshake
- Sending messages
- Receiving messages
- Terminating the connection: releasing resources

- Allows smooth transition to more secure protocol
 - Does not require you to replace existing protocol
- Proxy communicates with legacy IdP
- RPs communicate with proxy

- Allows smooth transition to more secure protocol
 - Does not require you to replace existing protocol
- Proxy communicates with legacy IdP
- RPs communicate with proxy



- Prototype implementation
 - Clean-slate / IdP deployment
 - Two protocols: OpenID-like and OAuth-like
 - 252 LOC JavaScript, 264 LOC HTML, 243 LOC PHP
 - External libraries: JavaScript Cryptography Toolkit + Stanford JavaScript Crypto Library
 - Proxy / RP deployment
 - Based on a Facebook application

- Formally verified design with ProVerif
 - Channel verification
 - Attacker: passive (sniffing), active (sending messages)
 - Result: an attacker cannot obtain the plain text message
 - Protocol verification
 - Attacker: network (passive) and web attackers (active)
 - Result: an attacker cannot obtain any useful information
 - Proxy verification
 - Attacker: passive (sniffing), active (sending messages)
 - Result: an attacker can obtain and modify the messages sent over the insecure communication channel between proxy and legacy IdP

- Our protocol prevents all impersonation attacks identified by Wang et al. [Oakland '12]:
 - Facebook and New York Times
 - Facebook and Zoho
 - Facebook Legacy Canvas Auth
 - JanRain wrapping GoogleID
 - JanRain wrapping Facebook

Channel operation

Operation	Delay [ms]
Establishing the channel	164±12
Sending a message	32±2
Destroying a channel	70±3

Channel operation

Operation	Delay [ms]
Establishing the channel	164±12
Sending a message	32±2
Destroying a channel	70±3

Establishing the channel

Operation	Delay [ms]
Message #1: PK_RP	92±9
Message #2: PK_RP(SK, N_IdP)	29±2
Message #3: SK(N_RP)	43±3

Detailed breakdown of the protocol

Operation	Delay [ms]
(1) Creating the channel between RP and IdP	164±11
(2) Creating the IdP inline frame	57±3
(3) Sending the first message from RP to IdP	32±2
(4) Creating the IdP inline frame for authentication	57±3
(5) Creating the second channel inside the IdP	165±11
(6) Authenticating the user	56±4
(7) Requesting the user's permissions	57±3
(8) Sending the token inside the IdP's inline frame	32±2
(9) Sending the token to the RP	33±2
Total	653±21

(2), (4), (6), and (7) are dominated by network latency, which is 50ms here.

- Pointed out root cause why RPI attacks exist: non-dedicated, insecure, one-way channel between RP and IdP

- Pointed out root cause why RPI attacks exist: non-dedicated, insecure, one-way channel between RP and IdP
- Proposed a dedicated bi-directional secure channel to remedy existing short-comings

- Pointed out root cause why RPI attacks exist: non-dedicated, insecure, one-way channel between RP and IdP
- Proposed a dedicated bi-directional secure channel to remedy existing short-comings
- Designed SSO protocol on top of channel design

- Pointed out root cause why RPI attacks exist: non-dedicated, insecure, one-way channel between RP and IdP
- Proposed a dedicated bi-directional secure channel to remedy existing short-comings
- Designed SSO protocol on top of channel design
- Presented a proxy design for easy adoptability

- Pointed out root cause why RPI attacks exist: non-dedicated, insecure, one-way channel between RP and IdP
- Proposed a dedicated bi-directional secure channel to remedy existing short-comings
- Designed SSO protocol on top of channel design
- Presented a proxy design for easy adoptability
- Formally verified security of the SSO protocol

- Pointed out root cause why RPI attacks exist: non-dedicated, insecure, one-way channel between RP and IdP
- Proposed a dedicated bi-directional secure channel to remedy existing short-comings
- Designed SSO protocol on top of channel design
- Presented a proxy design for easy adoptability
- Formally verified security of the SSO protocol
- Evaluated protocol performance / overhead

Thank you for your attention!



THE COMPUTER SECURITY GROUP AT UC SANTA BARBARA

kevin@borgolte.me
<http://kevin.borgolte.me>
twitter: @caovc

Thank you for your attention!

Questions?



THE COMPUTER SECURITY GROUP AT UC SANTA BARBARA

kevin@borgolte.me
<http://kevin.borgolte.me>
twitter: @caovc

	Deployment	Protection Crowd	Preventing Impersonation Attacks	Proactive Deployment
InteGuard	IdP, Gateway	IdP Users, physical machines	✓	✗
AuthScan	IdP	IdP Users	✓	✗
Explicating SDKs	IdP	IdP Users	✓	✗
Defensive JavaScript	IdP, RP	IdP Users, RP Users	✗	✓
WebSSO (our work)	IdP, RP	IdP Users, RP Users	✓	✓