On the Security and Privacy of Modern Single Sign-On in the Web

(Not Only) Attacks on OAuth and OpenID Connect

Daniel Fett

Guido Schmitz

Daniel Fett
@dfett42

Guido Schmitz
@gtrsde

Universität Trier

Universität Stuttgart

soon!

Daniel Fett
@dfett42

Guido Schmitz
@gtrsde

Universität Trier

Universität Stuttgart

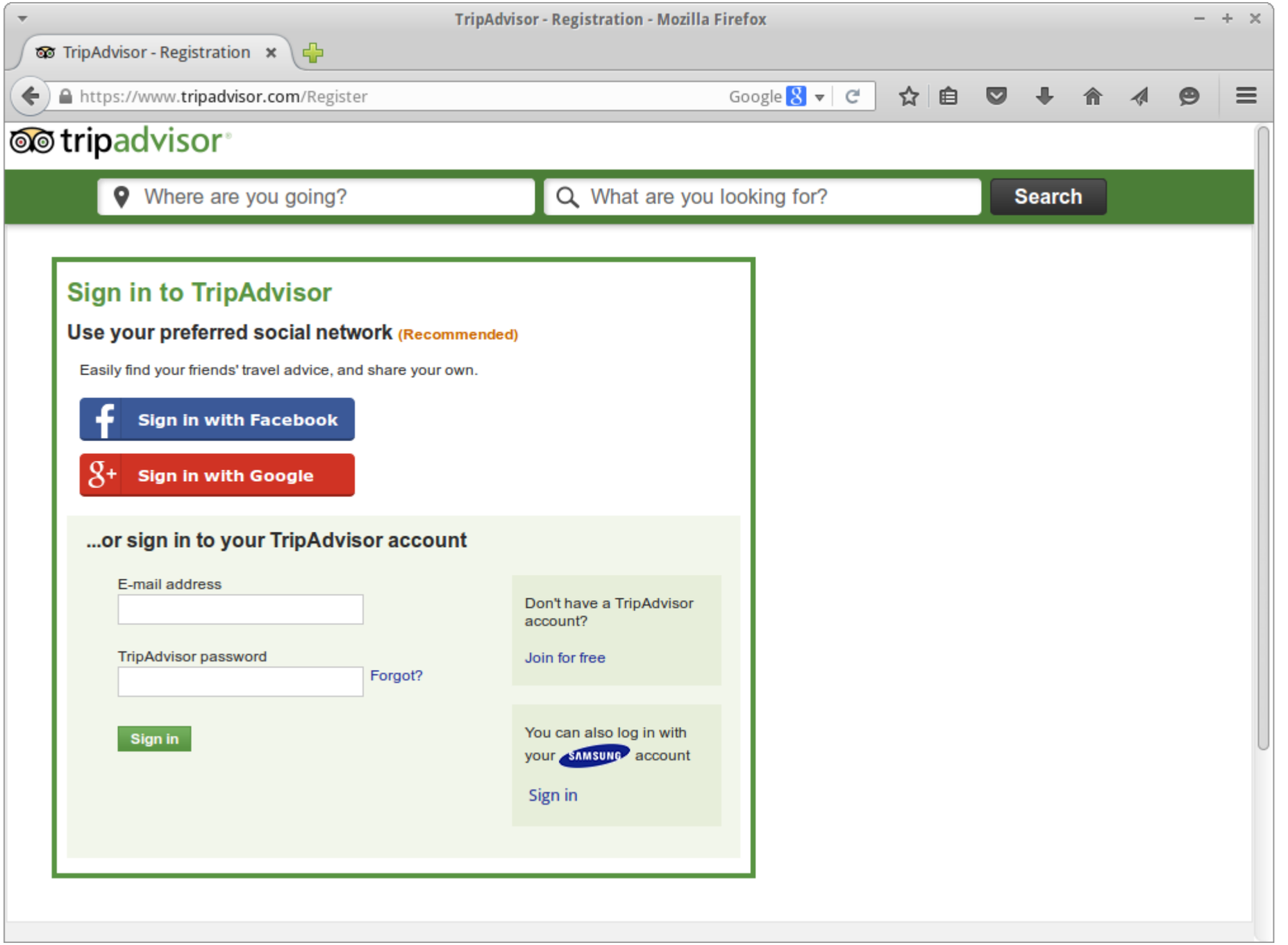
soon!

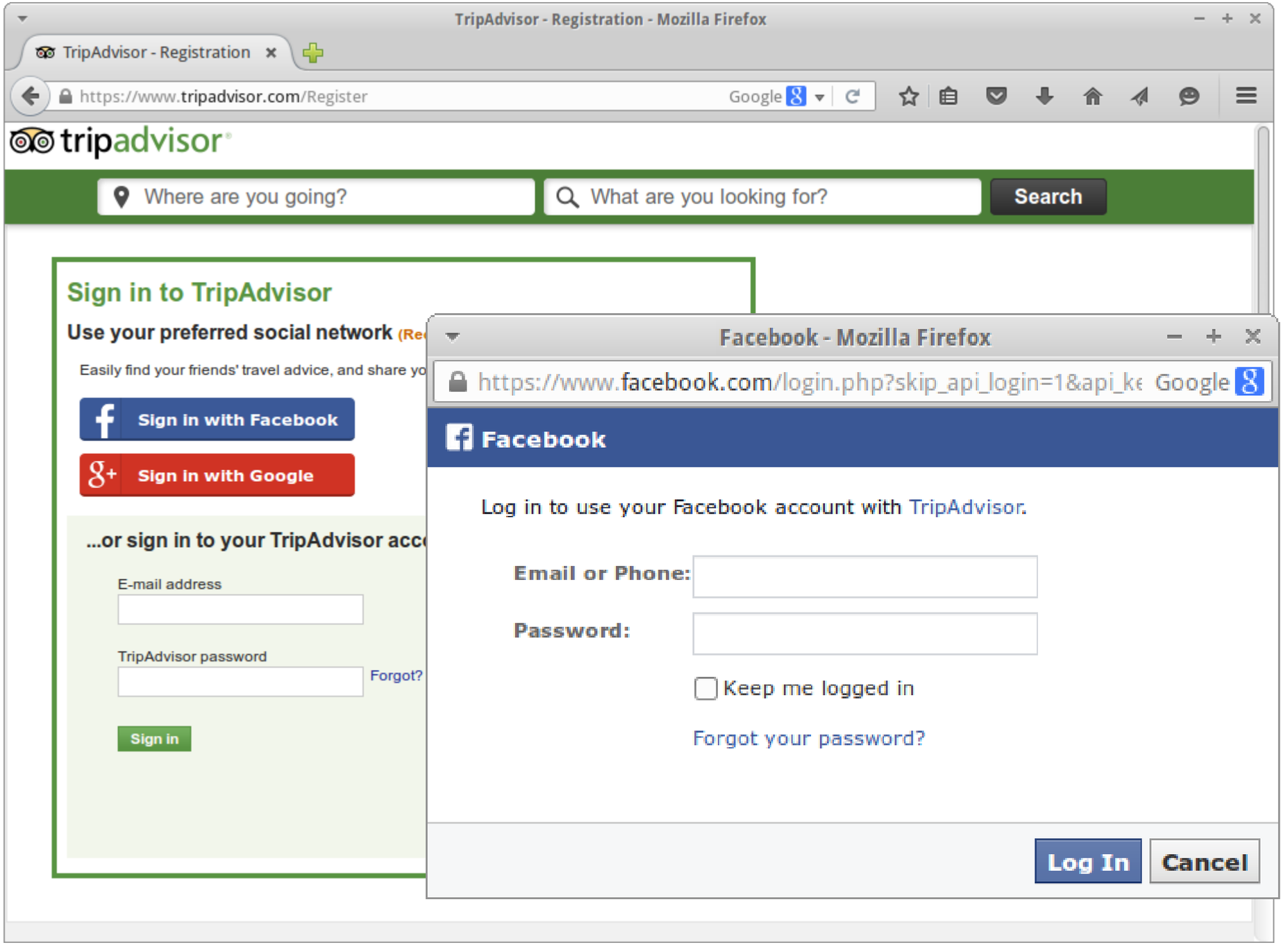
Maschinendeck.org

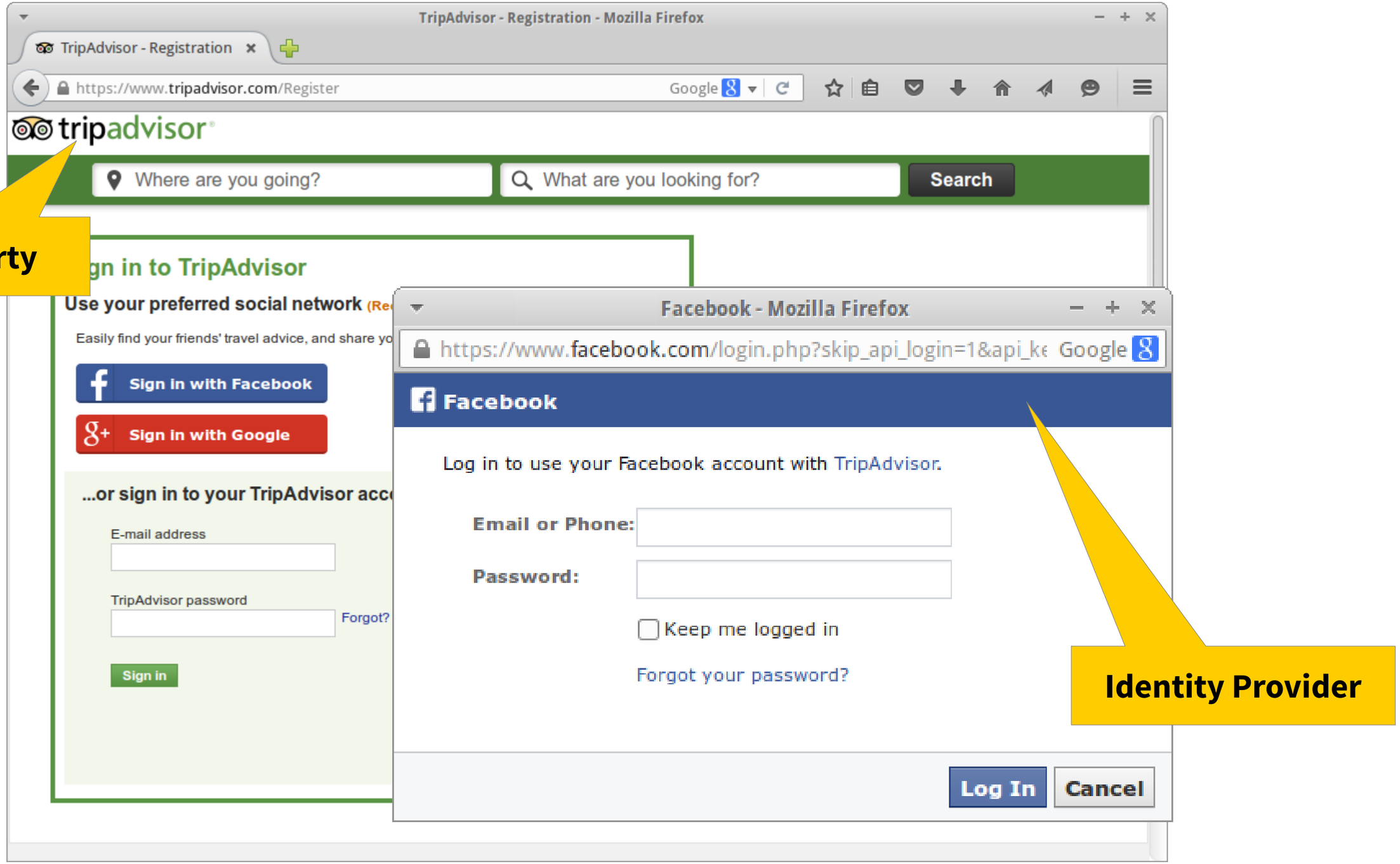
PiAndMore.de

Daniel Fett
@dfett42

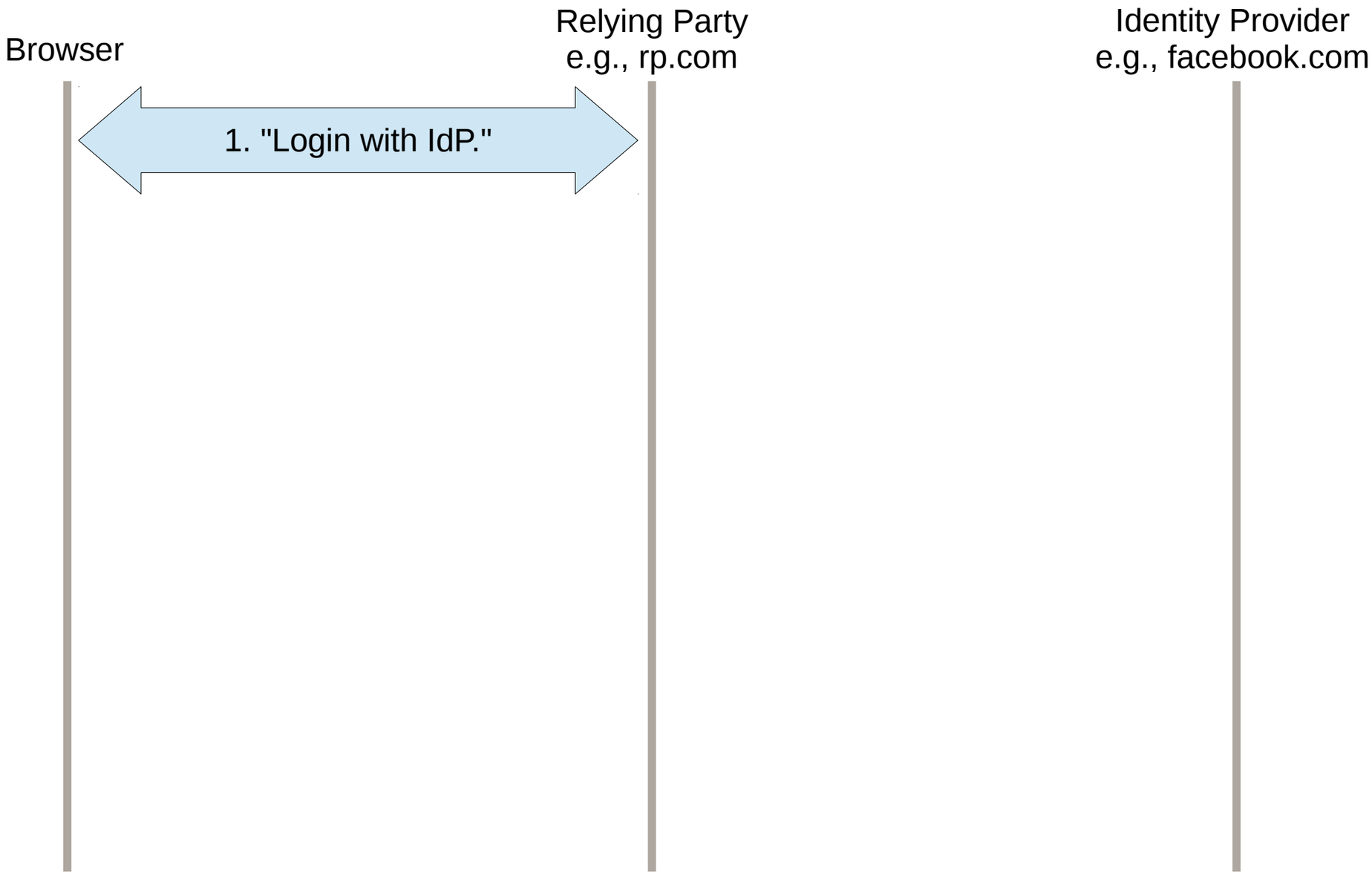
Guido Schmitz
@gtrsde


TripAdvisor - Registration - Mozilla Firefox
TripAdvisor - Registration
https://www.tripadvisor.com/Register
Google
tripadvisor
Where are you going?
What are you looking for?
Search
Sign in to TripAdvisor
Use your preferred social network (Recommended)
Easily find your friends' travel advice, and share your own.
Sign in with Facebook
Sign in with Google
...or sign in to your TripAdvisor account
E-mail address
TripAdvisor password
Forgot?
Sign in
Don't have a TripAdvisor account?
Join for free
You can also log in with your SAMSUNG account
Sign in

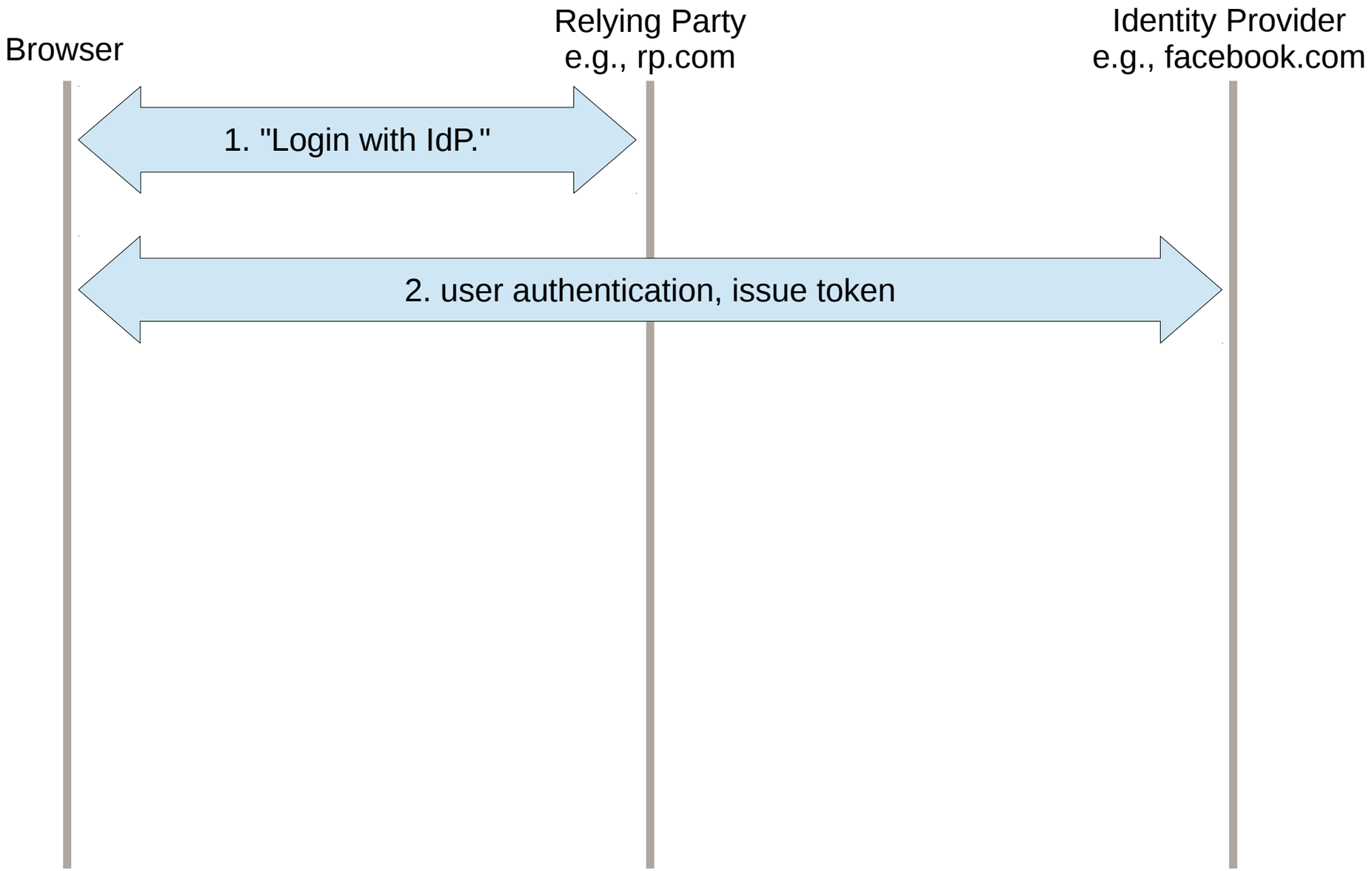

TripAdvisor - Registration - Mozilla Firefox
TripAdvisor - Registration
https://www.tripadvisor.com/Register
Google
tripadvisor
Where are you going?
What are you looking for?
Search
Sign in to TripAdvisor
Use your preferred social network
Easily find your friends' travel advice, and share yo
Sign in with Facebook
Sign in with Google
...or sign in to your TripAdvisor acc
E-mail address
TripAdvisor password
Forgot?
Sign in
Facebook - Mozilla Firefox
https://www.facebook.com/login.php?skip_api_login=1&api_ke
Google
Facebook
Log in to use your Facebook account with TripAdvisor.
Email or Phone:
Password:
Keep me logged in
Forgot your password?
Log In
Cancel


TripAdvisor - Registration - Mozilla Firefox
TripAdvisor - Registration
https://www.tripadvisor.com/Register
Google
tripadvisor
Where are you going?
What are you looking for?
Search
Relying Party
gn in to TripAdvisor
Use your preferred social network
Easily find your friends' travel advice, and share yo
Sign in with Facebook
Sign in with Google
...or sign in to your TripAdvisor acc
E-mail address
TripAdvisor password
Forgot?
Sign in
Facebook - Mozilla Firefox
https://www.facebook.com/login.php?skip_api_login=1&api_ke
Google
Facebook
Log in to use your Facebook account with TripAdvisor.
Email or Phone:
Password:
Keep me logged in
Forgot your password?
Log In
Cancel
Identity Provider

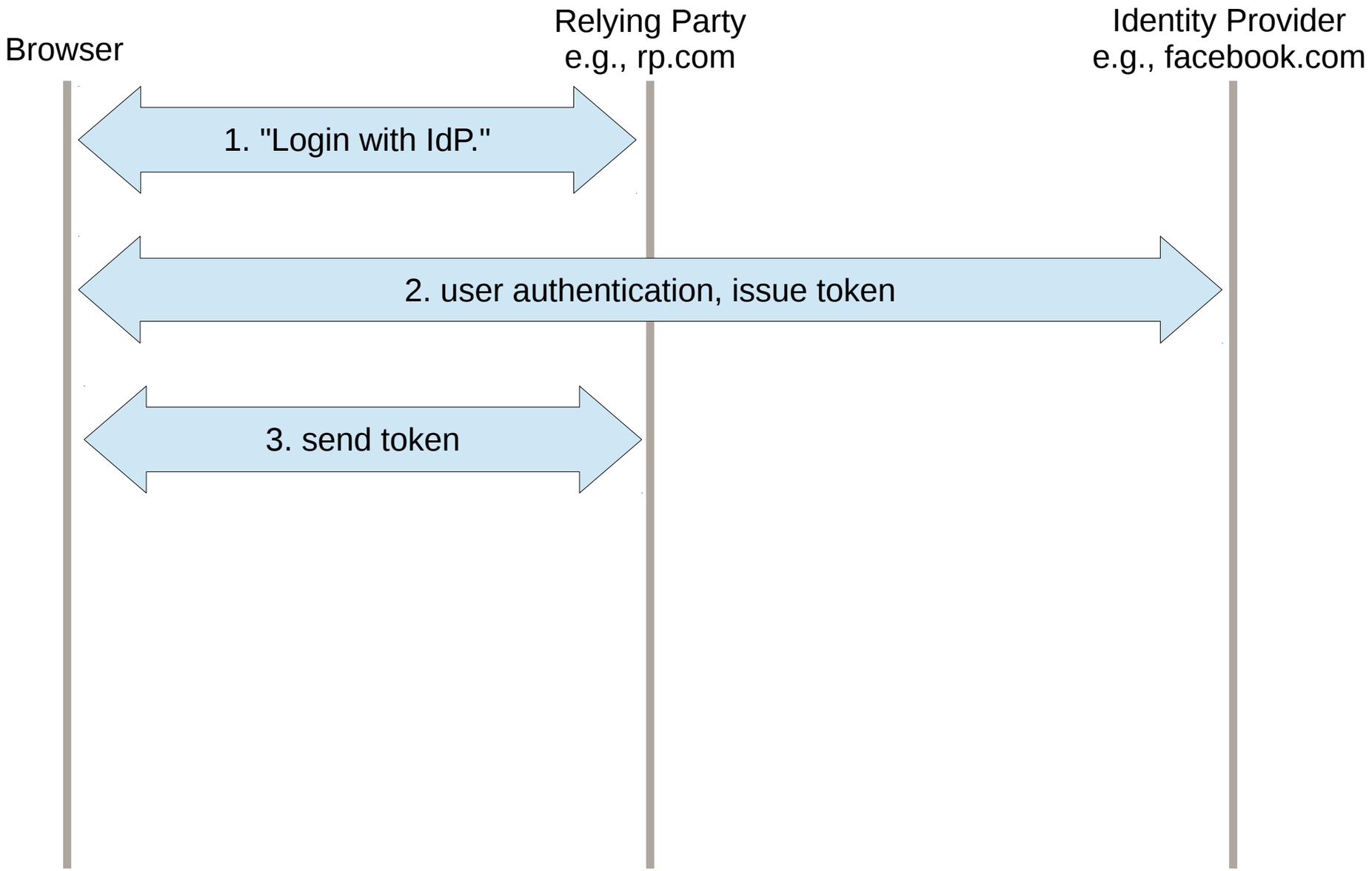
Web SSO: Basic Principle



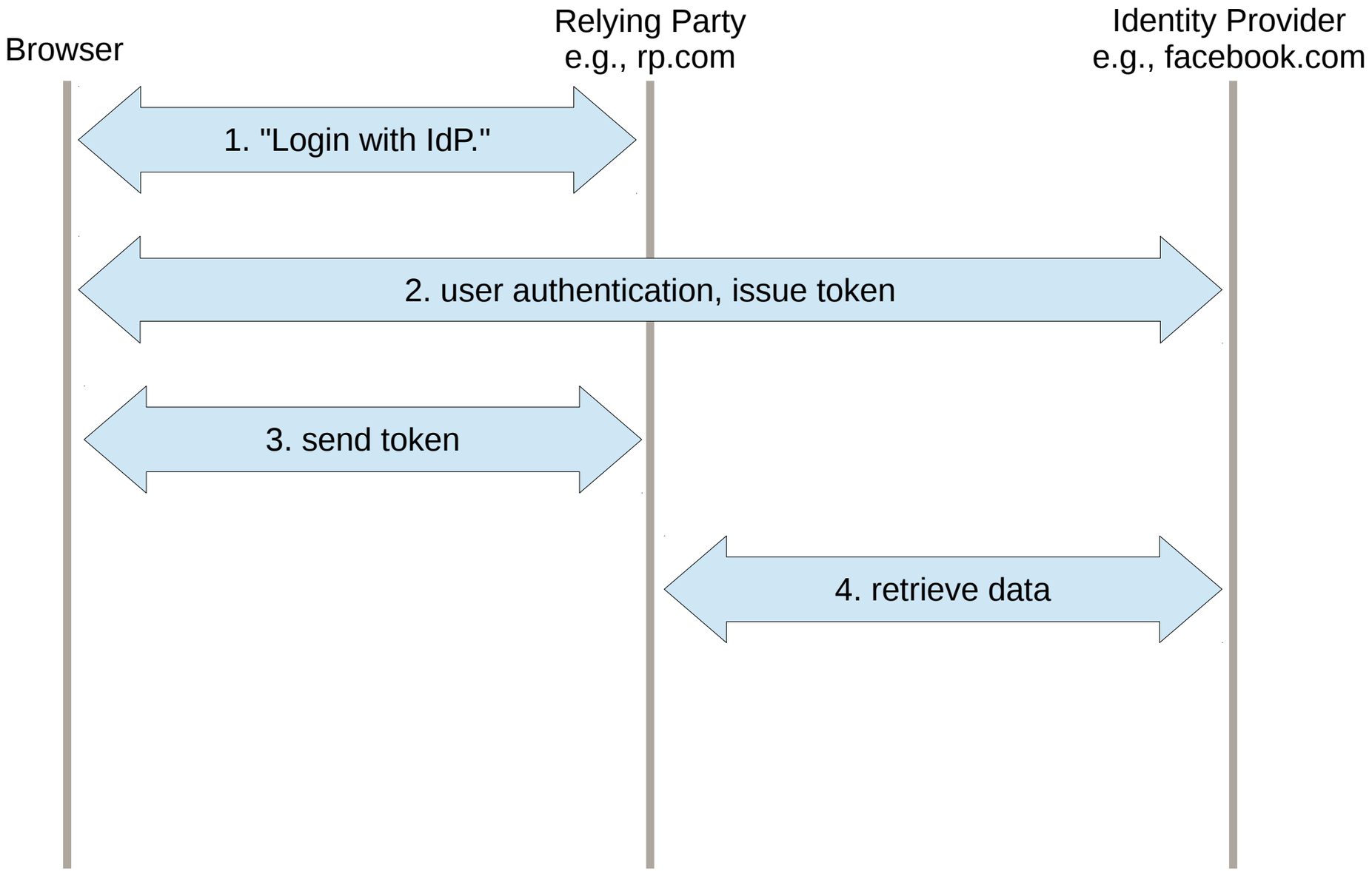
Web SSO: Basic Principle



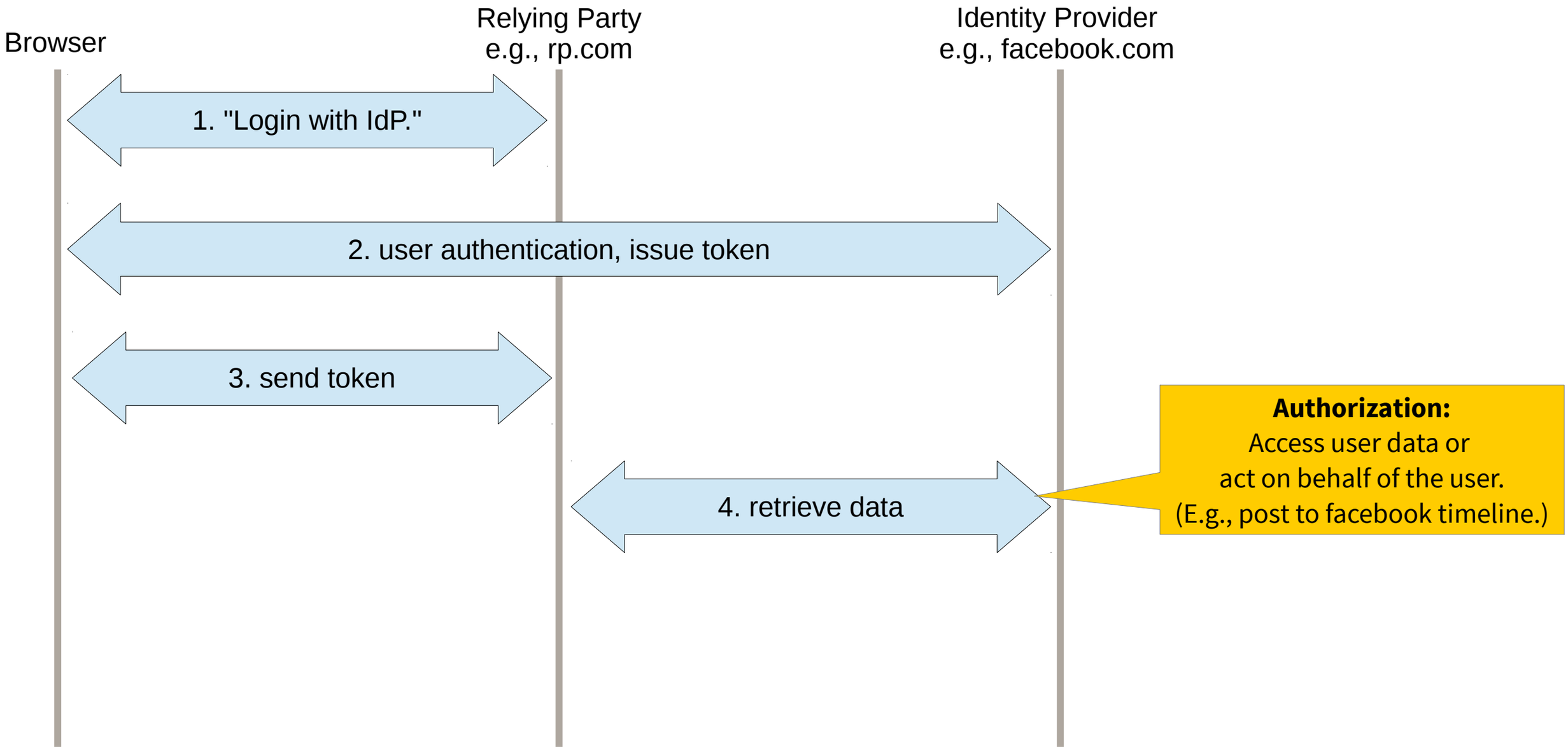
Web SSO: Basic Principle



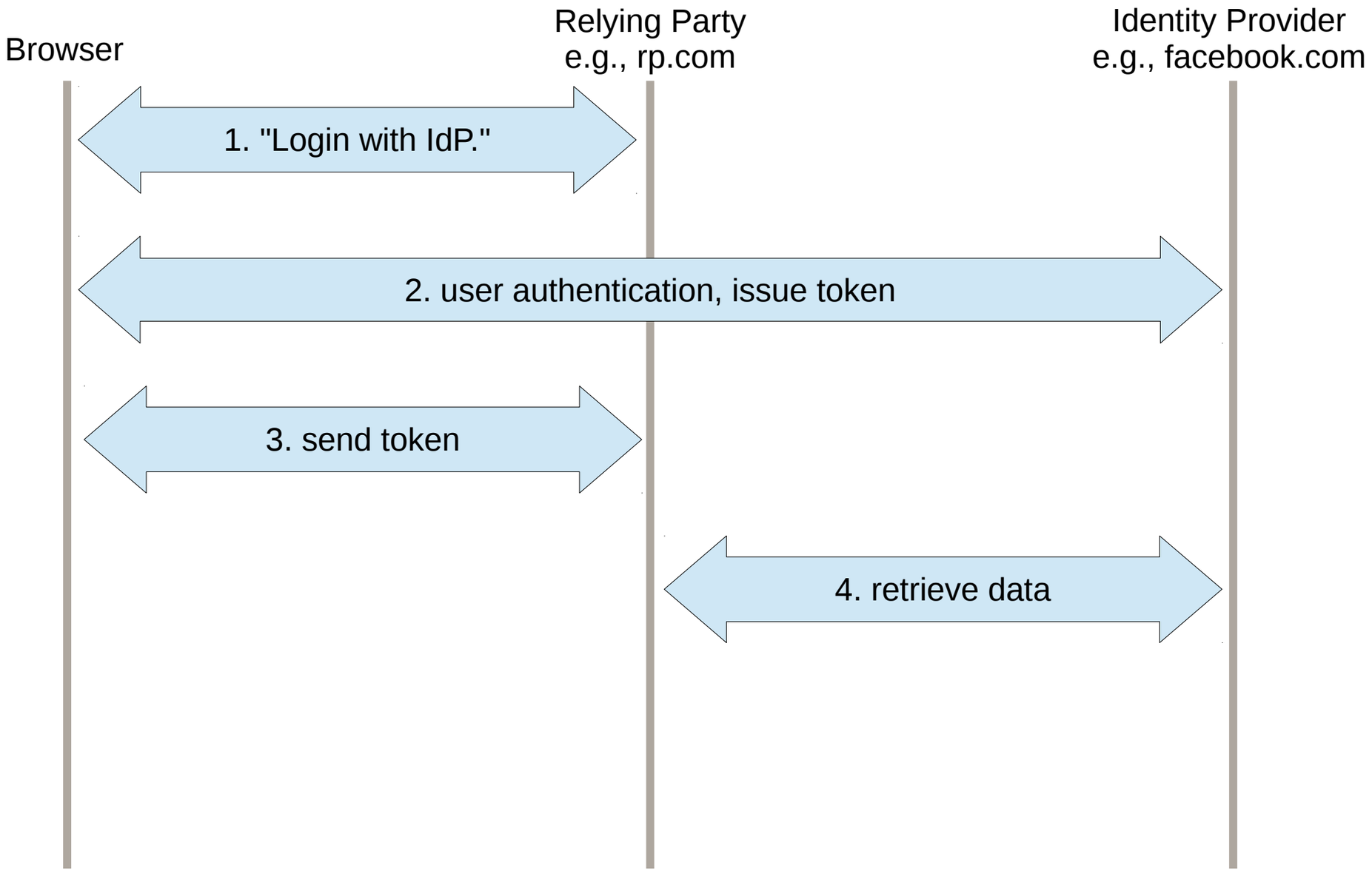
Web SSO: Basic Principle



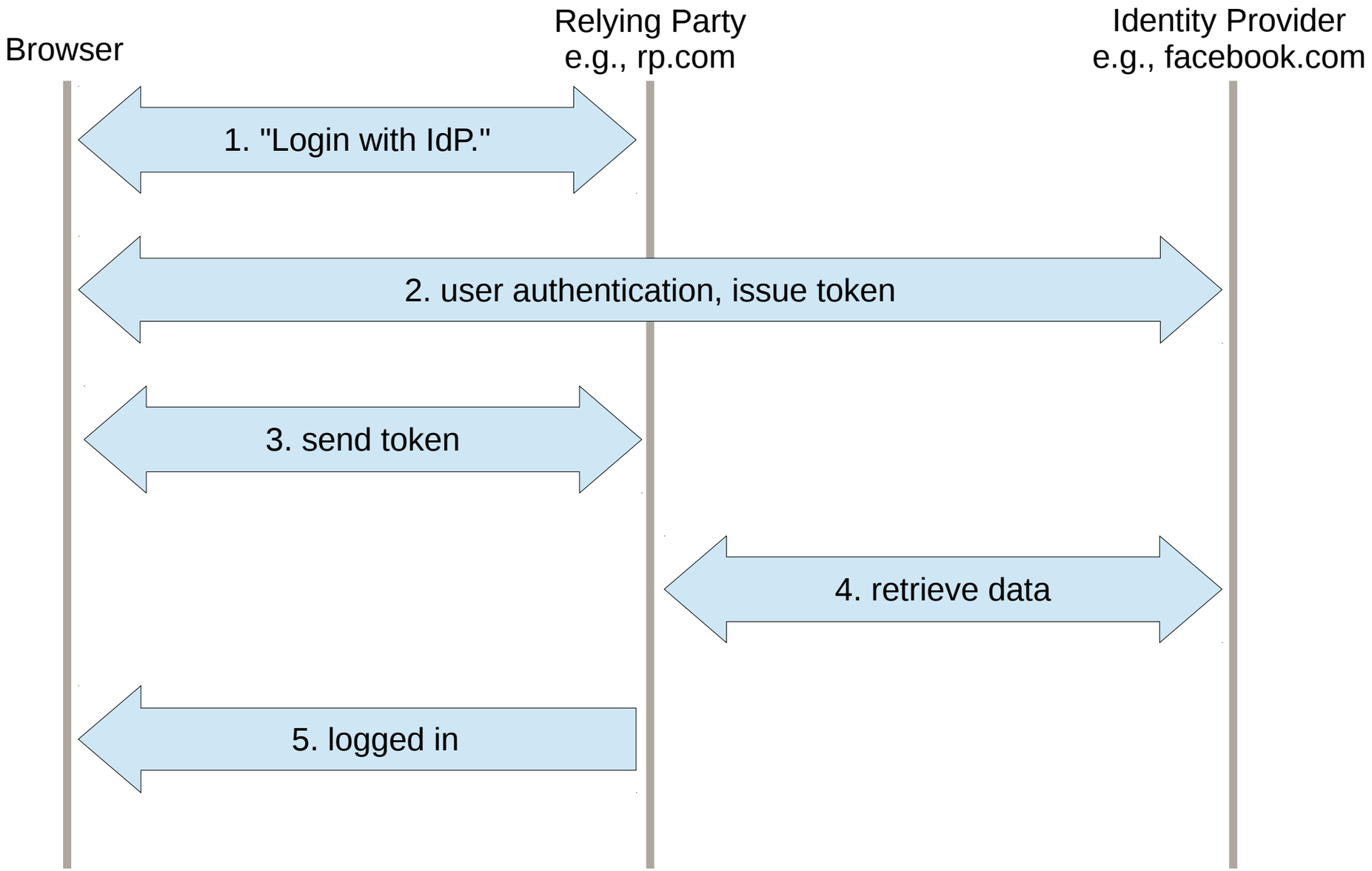
Web SSO: Basic Principle



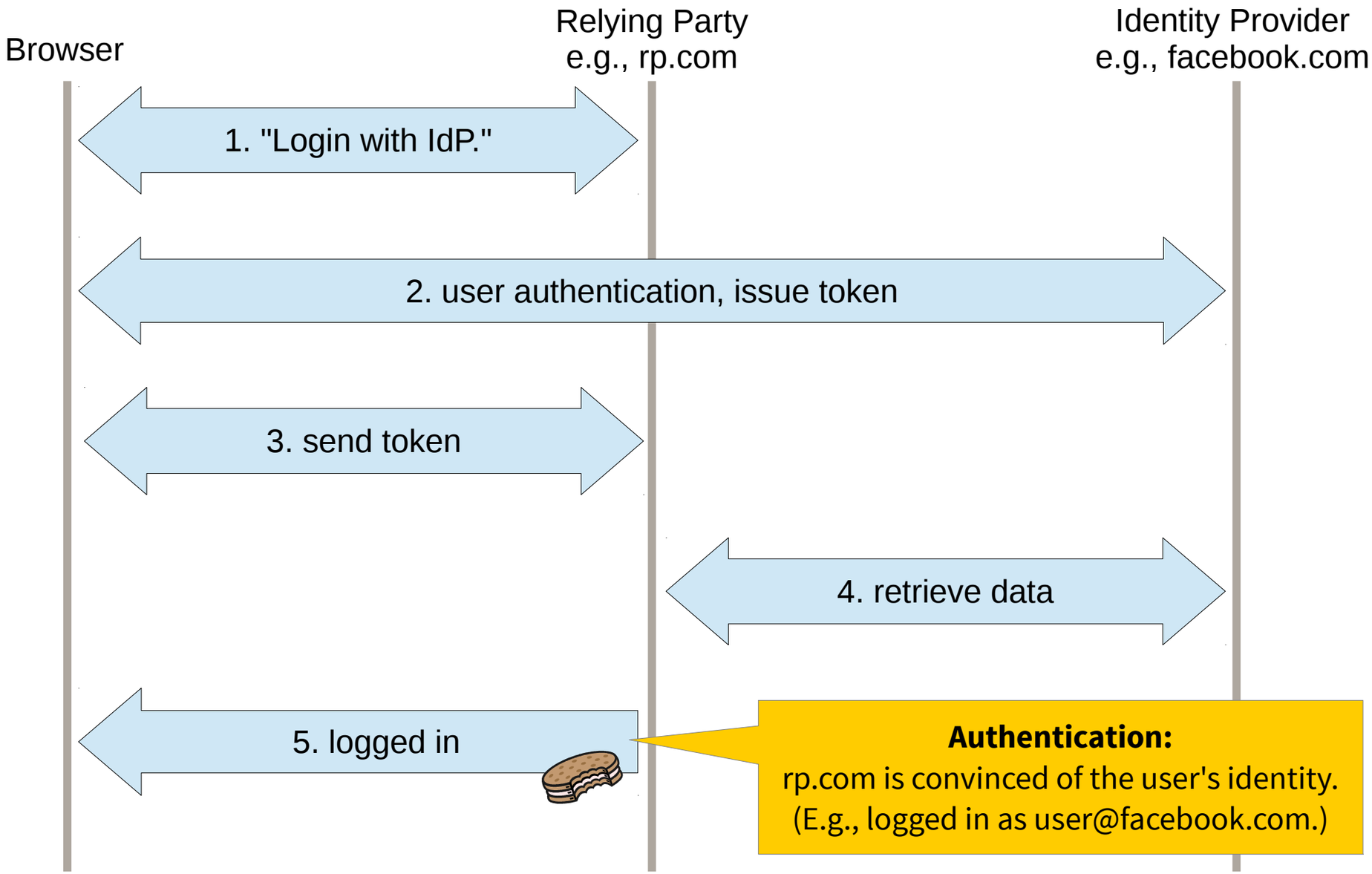
Web SSO: Basic Principle



Web SSO: Basic Principle



Web SSO: Basic Principle



Web SSO: Why should(n't) I use it?

For users

+ ease-of-use
+ harder to forget passwords

- lack of privacy
- IdP account is single point of failure

Web SSO: Why should(n't) I use it?

For users

+ ease-of-use
+ harder to forget passwords
- lack of privacy
- IdP account is single point of failure

For Relying Parties

+ less data to store
+ no password recovery, etc.
- less control over user's accounts
- IdP account is single point of failure

Web SSO: Why should(n't) I use it?

For users

+ ease-of-use
+ harder to forget passwords
- lack of privacy
- IdP account is single point of failure

For Relying Parties

+ less data to store
+ no password recovery, etc.
- less control over user's accounts
- IdP account is single point of failure

For Identity Providers

+ more user data (for advertising, etc.)
+ service for users
- more user data
- implementation overhead

Not-so-Modern SSO Systems

OAuth 1.0

- Old! (2006)
- Many flaws
- Not in widespread use any longer (exception: twitter)
- **Do not use!**

Not-so-Modern SSO Systems

OAuth 1.0

- Old! (2006)
- Many flaws
- Not in widespread use any longer (exception: twitter)
- **Do not use!**

OpenID

- Old! (2007)
- Not user-friendly
- Super-flexible (read: not developer-friendly)
- **Do not use!**

Modern Single-Sign On Systems

OAuth 2.0

- Modern (2011) authorization protocol
- **Completely incompatible to OAuth < 2.0**
- Bearer token approach, no crypto (except transport layer)
- Used everywhere (almost!)
- Not for authentication. *Repeat: Not for authentication!*

Quelle: Internet

Modern Single-Sign On Systems

OAuth 2.0

- Modern (2011) authorization protocol
- **Completely incompatible to OAuth < 2.0**
- Bearer token approach, no crypto (except transport layer)
- Used everywhere (almost!)
- Not for authentication. *Repeat: Not for authentication!*
- Nonetheless used for authorization **and authentication**
- Many flaws (mostly fixed)

Modern Single-Sign On Systems

OAuth 2.0

- Modern (2011) authorization protocol
- **Completely incompatible to OAuth < 2.0**
- Bearer token approach, no crypto (except transport layer)
- Used everywhere (almost!)
- Not for authentication. *Repeat: Not for authentication!*
- Nonetheless used for authorization **and authentication**
- Many flaws (mostly fixed)

OpenID Connect

- Very modern (2014)!
- Authentication layer **on top of OAuth**
- **Completely incompatible to OpenID**
- IdP Discovery and dynamic RP Registration

Web SSO: The Chart of Confusion

Authorization

OAuth 1.0

Web SSO: The Chart of Confusion

Authorization

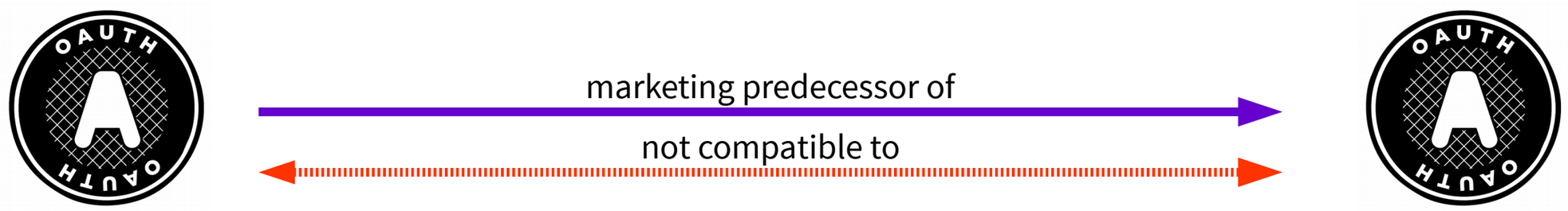
OAuth 1.0

marketing predecessor of

OAuth 2.0

Web SSO: The Chart of Confusion

Authorization



Web SSO: The Chart of Confusion

Authorization

marketing predecessor of

not compatible to

OAuth 1.0

OAuth 2.0

Authentication

foundation for

Web SSO: The Chart of Confusion

Authorization

marketing predecessor of

not compatible to

OAuth 1.0

OAuth 2.0

Authentication

foundation for

OpenID Connect

Web SSO: The Chart of Confusion

Authorization

marketing predecessor of

not compatible to

OAuth 1.0

OAuth 2.0

Authentication

foundation for

marketing predecessor of

OpenID

OpenID Connect

Web SSO: The Chart of Confusion

Authorization

marketing predecessor of

not compatible to

OAuth 1.0

OAuth 2.0

Authentication

foundation for

marketing predecessor of

not compatible to

OpenID

OpenID Connect

Web SSO: The Chart of Confusion

Authorization

marketing predecessor of

not compatible to

OAuth 1.0

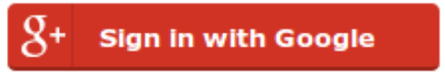
OAuth 2.0

Authentication

foundation for

marketing predecessor of

not compatible to



OpenID

OpenID Connect

Modern Single-Sign On Systems

Mozilla Persona
aka. BrowserID

Modern Single-Sign On Systems

Mozilla Persona
aka. BrowserID

- Different approach: E-Mail providers as Identity Providers (IdPs)
- **Privacy goal:** IdPs do not learn where you log in (see later)
- Developed by Mozilla
- Target: Integrate into browsers

... which never happened.

Modern Single-Sign On Systems

Mozilla Persona
aka. BrowserID

- Different approach: E-Mail providers as Identity Providers (IdPs)
- **Privacy goal:** IdPs do not learn where you log in (see later)
- Developed by Mozilla
- Target: Integrate into browsers

  ... which never happened.
- New target: Pure web implementation using HTML5 & stuff

  ... with "bridges" to OpenID and OAuth

Modern Single-Sign On Systems

Mozilla Persona
aka. BrowserID

- Different approach: E-Mail providers as Identity Providers (IdPs)
- **Privacy goal:** IdPs do not learn where you log in (see later)
- Developed by Mozilla
- Target: Integrate into browsers
  ... which never happened.
- New target: Pure web implementation using HTML5 & stuff
  ... with "bridges" to OpenID and OAuth
- **Failed, but interesting approach!**

What else?

- SAML
- Shibboleth
- WebAuth
- CAS
- …

So what is this all about?

Our Goal

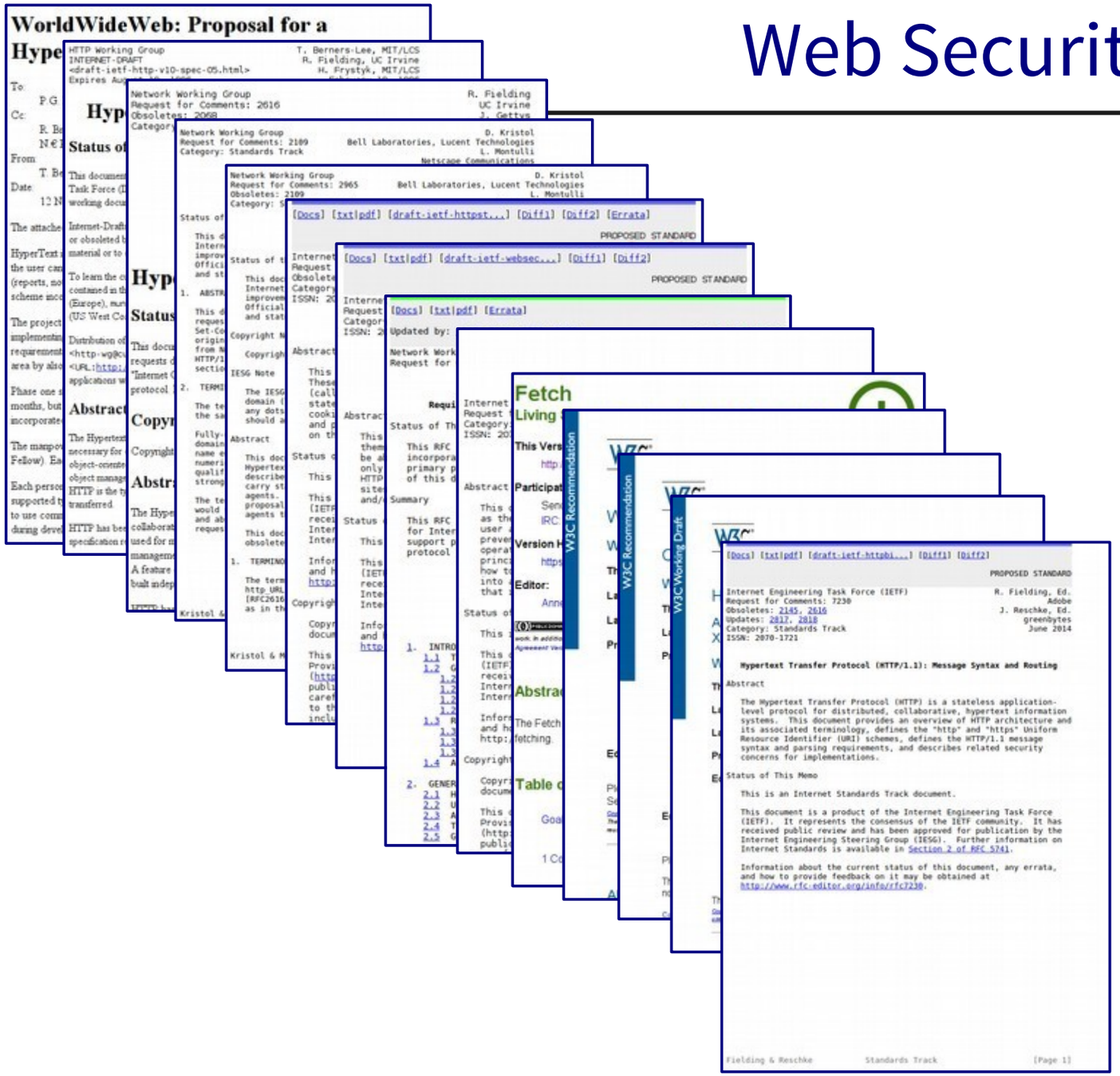
We analyze whether web mechanisms
(here: Web SSO protocols)
are secure
when implemented correctly
(i.e., following the standards and all best practices).

Our Goal

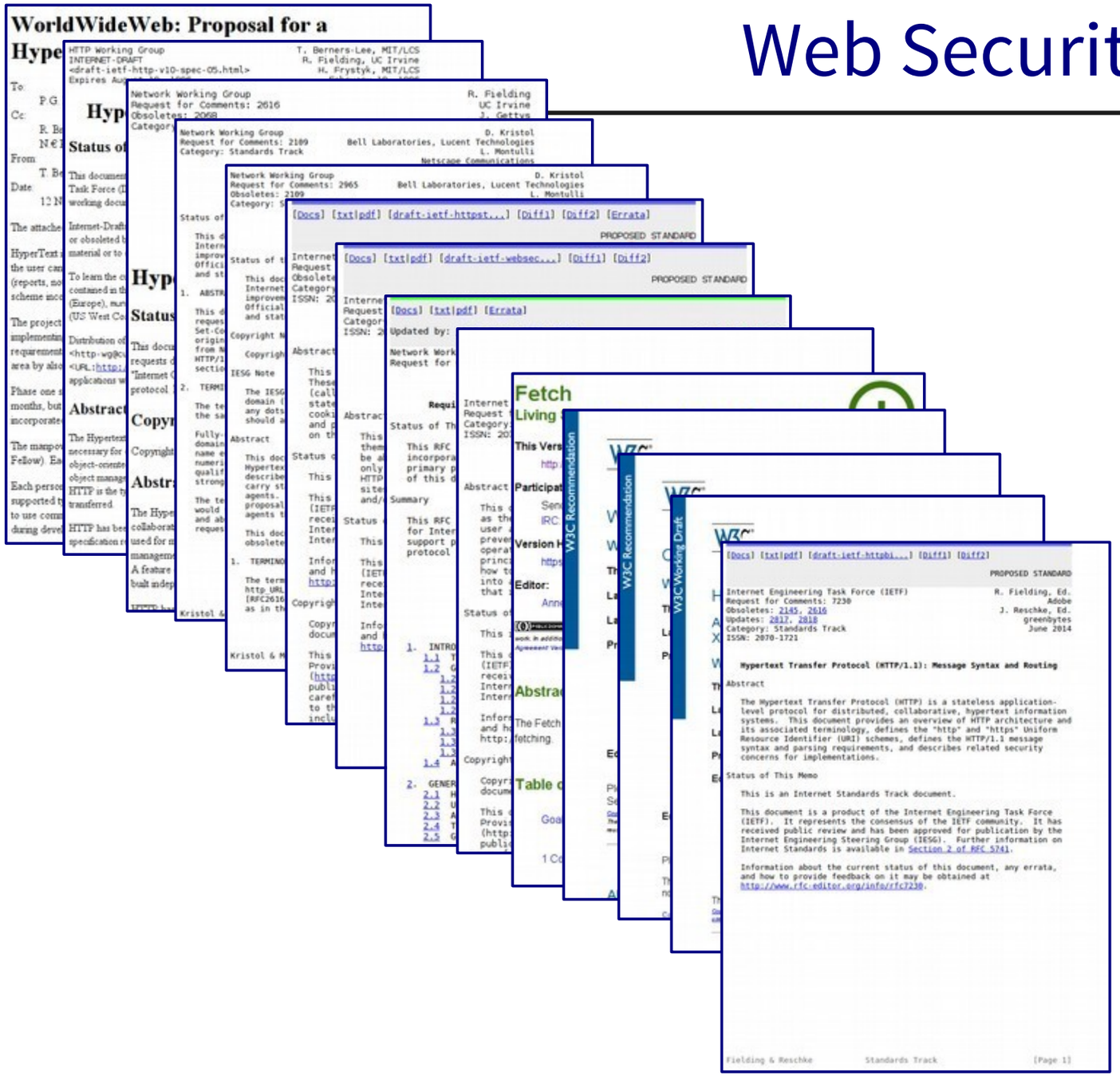
In other words:

Are the standards and protocols secure?

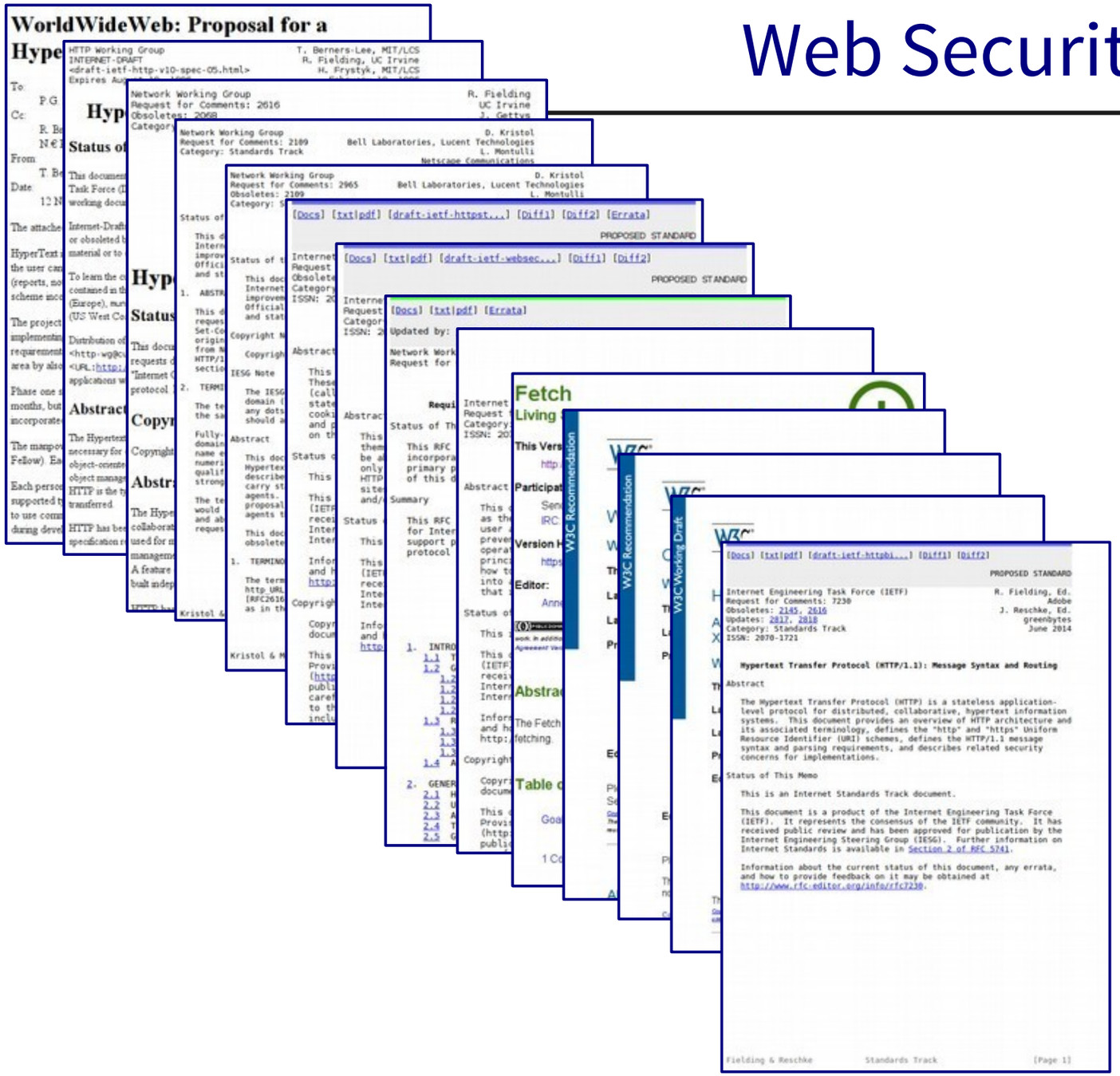
Web Security: State of the Art



Web Security: State of the Art



Web Security: State of the Art



Looks fine
to me!

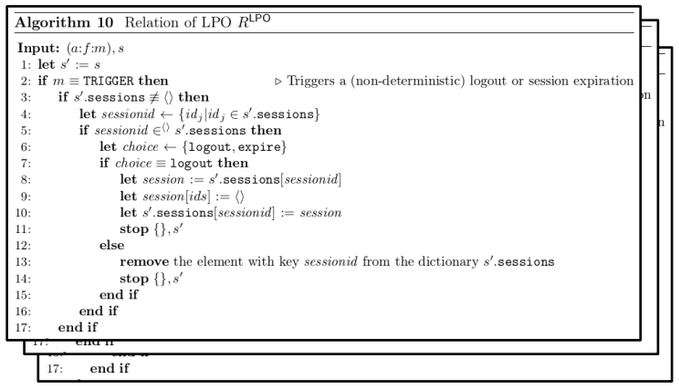
Our Approach: Formal Analysis

(part of our PhD research)

Our Approach: Formal Analysis

(part of our PhD research)

1. Create a Model of the Web Infrastructure and Web Applications



Algorithm 10 Relation of LPO R^{LPO}

Input: $(a{:}f{:}m), s$

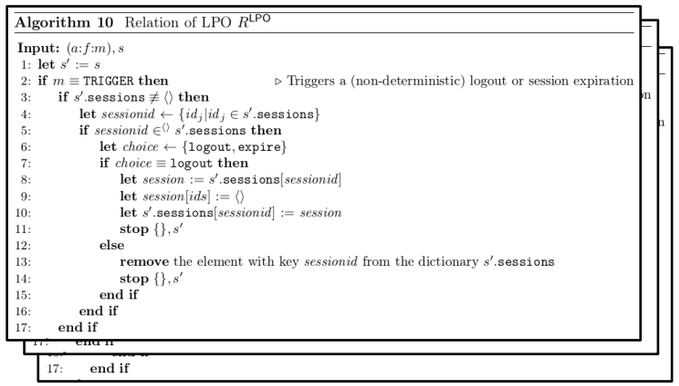
1: **let** $s' := s$
2: **if** $m \equiv$ `TRIGGER` **then** ▷ Triggers a (non-deterministic) logout or session expiration
3: **if** $s'.\mathtt{sessions} \not\equiv \langle\rangle$ **then**
4: **let** *sessionid* $\leftarrow \{id_j | id_j \in s'.\mathtt{sessions}\}$
5: **if** *sessionid* $\in^{\langle\rangle} s'.\mathtt{sessions}$ **then**
6: **let** *choice* $\leftarrow$ {`logout`, `expire`}
7: **if** *choice* $\equiv$ `logout` **then**
8: **let** *session* $:= s'.\mathtt{sessions}[sessionid]$
9: **let** *session*$[ids] := \langle\rangle$
10: **let** $s'.\mathtt{sessions}[sessionid] :=$ *session*
11: **stop** $\{\}, s'$
12: **else**
13: **remove** the element with key *sessionid* from the dictionary $s'.\mathtt{sessions}$
14: **stop** $\{\}, s'$
15: **end if**
16: **end if**
17: **end if**

(Models are incomplete by nature, but useful, see TLS 1.3)

Our Approach: Formal Analysis

(part of our PhD research)

1. Create a Model of the Web Infrastructure and Web Applications



Algorithm 10 Relation of LPO R^{LPO}

Input: $(a{:}f{:}m), s$

1: **let** $s' := s$
2: **if** $m \equiv$ `TRIGGER` **then** ▷ Triggers a (non-deterministic) logout or session expiration
3: **if** $s'.\mathtt{sessions} \not\equiv \langle\rangle$ **then**
4: **let** *sessionid* $\leftarrow \{id_j | id_j \in s'.\mathtt{sessions}\}$
5: **if** *sessionid* $\in^{\langle\rangle} s'.\mathtt{sessions}$ **then**
6: **let** *choice* $\leftarrow$ {`logout`, `expire`}
7: **if** *choice* $\equiv$ `logout` **then**
8: **let** *session* $:= s'.\mathtt{sessions}[sessionid]$
9: **let** *session*$[ids] := \langle\rangle$
10: **let** $s'.\mathtt{sessions}[sessionid] :=$ *session*
11: **stop** $\{\}, s'$
12: **else**
13: **remove** the element with key *sessionid* from the dictionary $s'.\mathtt{sessions}$
14: **stop** $\{\}, s'$
15: **end if**
16: **end if**
17: **end if**

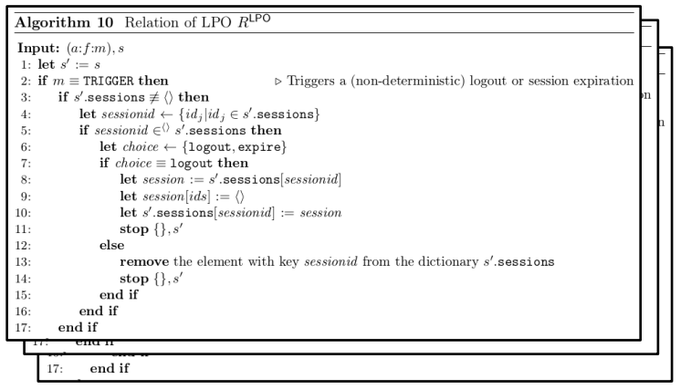
(Models are incomplete by nature, but useful, see TLS 1.3)

2. ???

Our Approach: Formal Analysis

(part of our PhD research)

1. Create a Model of the Web Infrastructure and Web Applications



Algorithm 10 Relation of LPO R^{LPO}

Input: $(a{:}f{:}m), s$

```
1: let s' := s
2: if m ≡ TRIGGER then                ▷ Triggers a (non-deterministic) logout or session expiration
3:     if s'.sessions ≢ ⟨⟩ then
4:         let sessionid ← {id_j | id_j ∈ s'.sessions}
5:         if sessionid ∈⟨⟩ s'.sessions then
6:             let choice ← {logout, expire}
7:             if choice ≡ logout then
8:                 let session := s'.sessions[sessionid]
9:                 let session[ids] := ⟨⟩
10:                let s'.sessions[sessionid] := session
11:                stop {}, s'
12:            else
13:                remove the element with key sessionid from the dictionary s'.sessions
14:                stop {}, s'
15:            end if
16:        end if
17:    end if
```

(Models are incomplete by nature, but useful, see TLS 1.3)

2. ???

3. Proofs of Security!

$$A \Rightarrow \neg B$$

None of our business ...

- Phishing
- Clickjacking
- Stupid Users (cf. phishing)
- Compromised Browsers/OSs
- Compromised Databases, etc.

What is security? What is privacy?

Authentication Properties

(A) An attacker (having full control over the network) should not be able to use a service of a relying party as an honest user.

Authentication Properties

(A) An attacker (having full control over the network) should not be able to use a service of a relying party as an honest user.

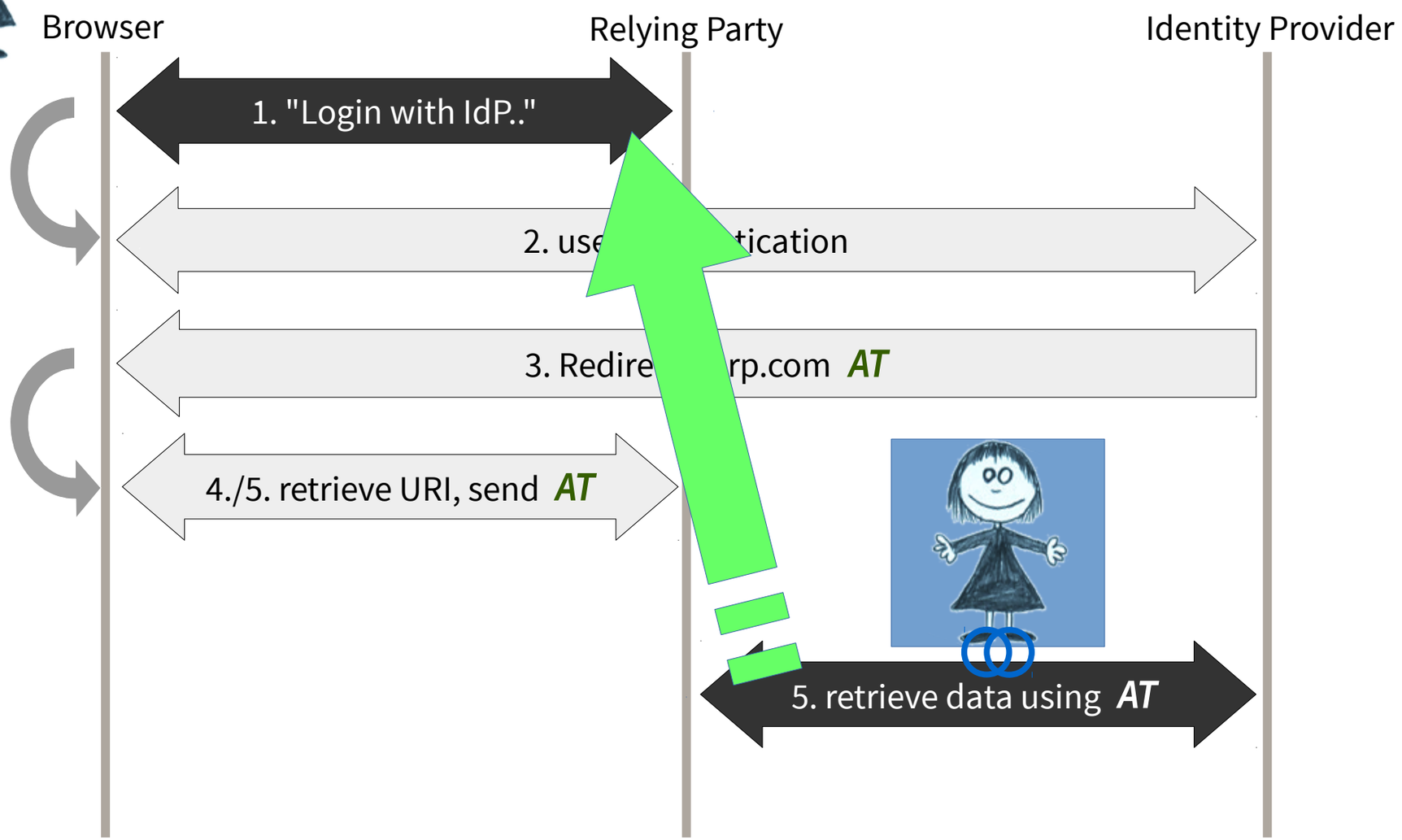
Authentication Properties

(A) An attacker (having full control over the network) should not be able to use a service of a relying party as an honest user.

(B) An attacker should not be able to authenticate an honest browser to a relying party as the attacker.

e.g., session fixation/swapping

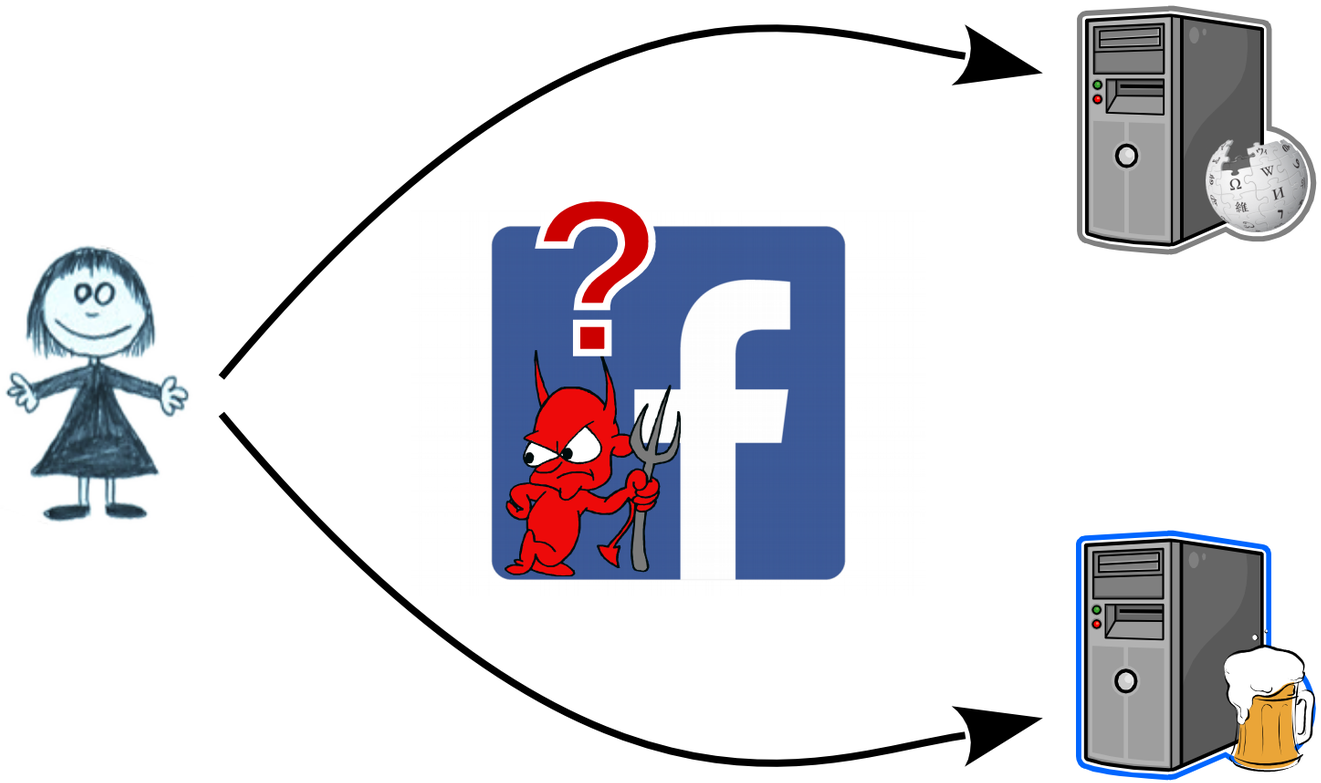
Session Integrity



Privacy

Privacy in Web SSO (IdP Privacy):

A malicious identity provider should not be able to distinguish whether Alice logs in at relying party A or B.



There are also other notions of Privacy for SSO!

Ok, let's start with OAuth (2.0!) ....

OAuth 2.0: RFC 6749 (and others)

Four modes of Interaction:

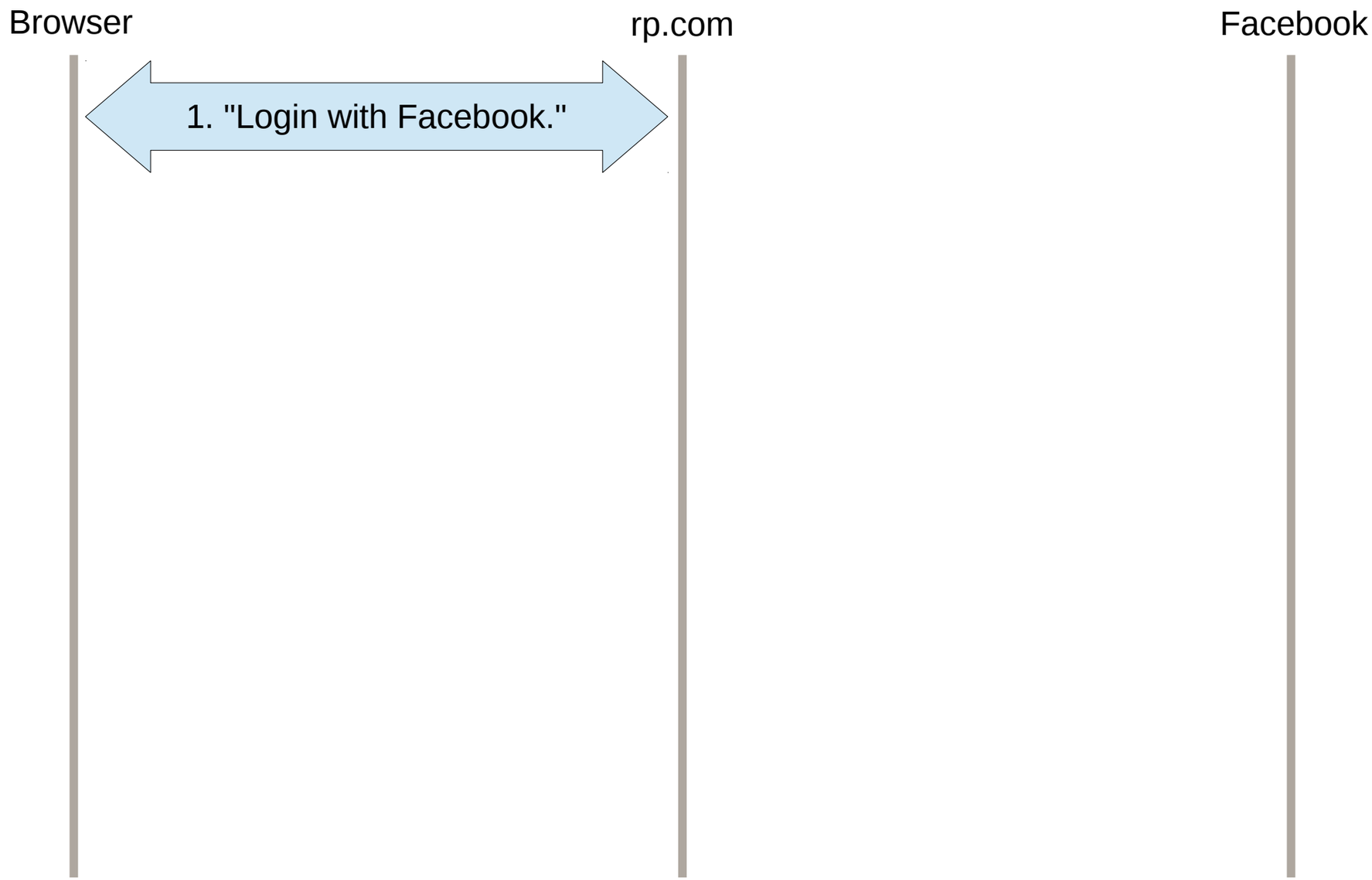
- **Implicit Mode**
- **Authorization Code Mode**

} most common

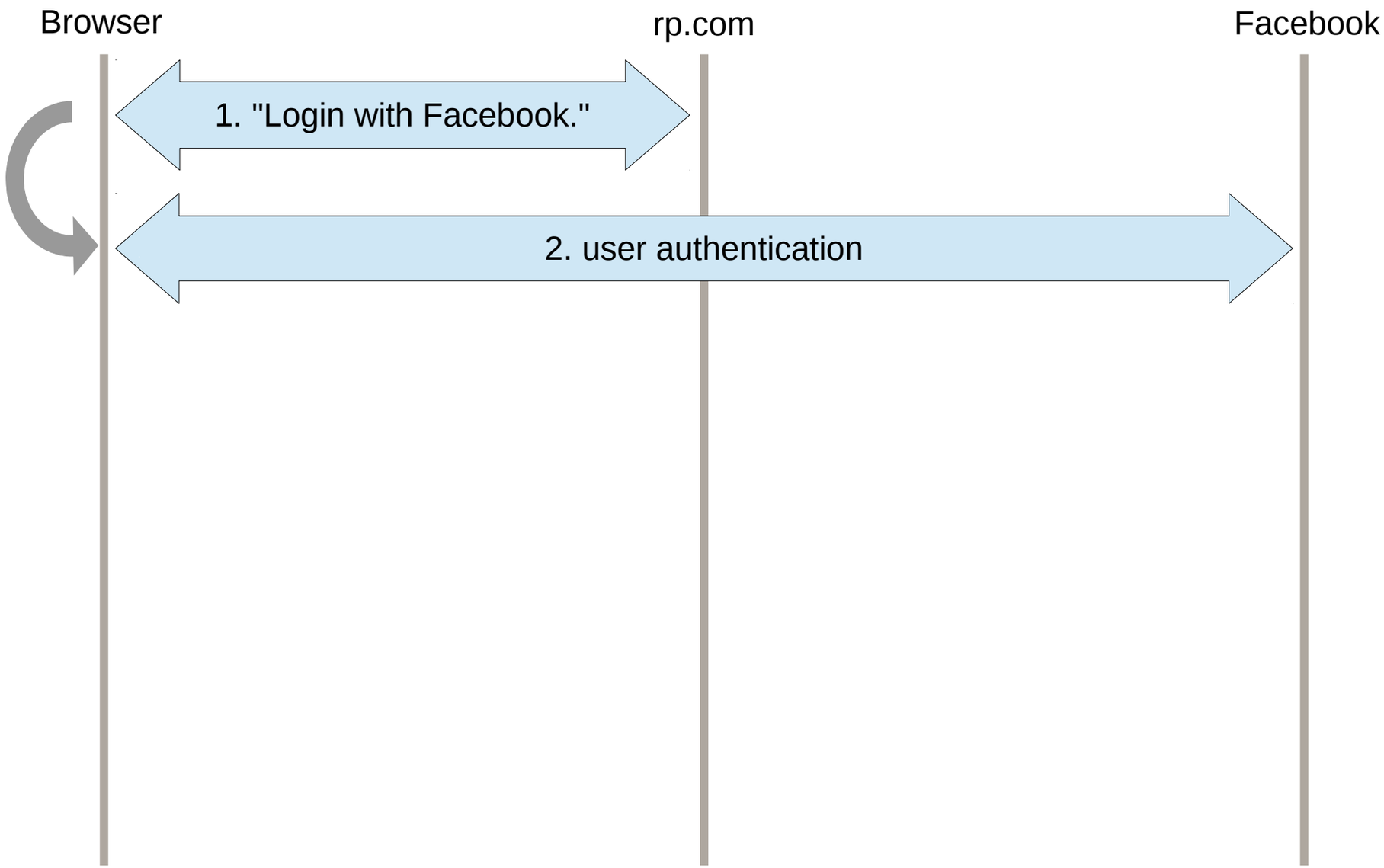
- Resource Owner Password Credentials Mode
- Client Credentials Mode

... and many other options.

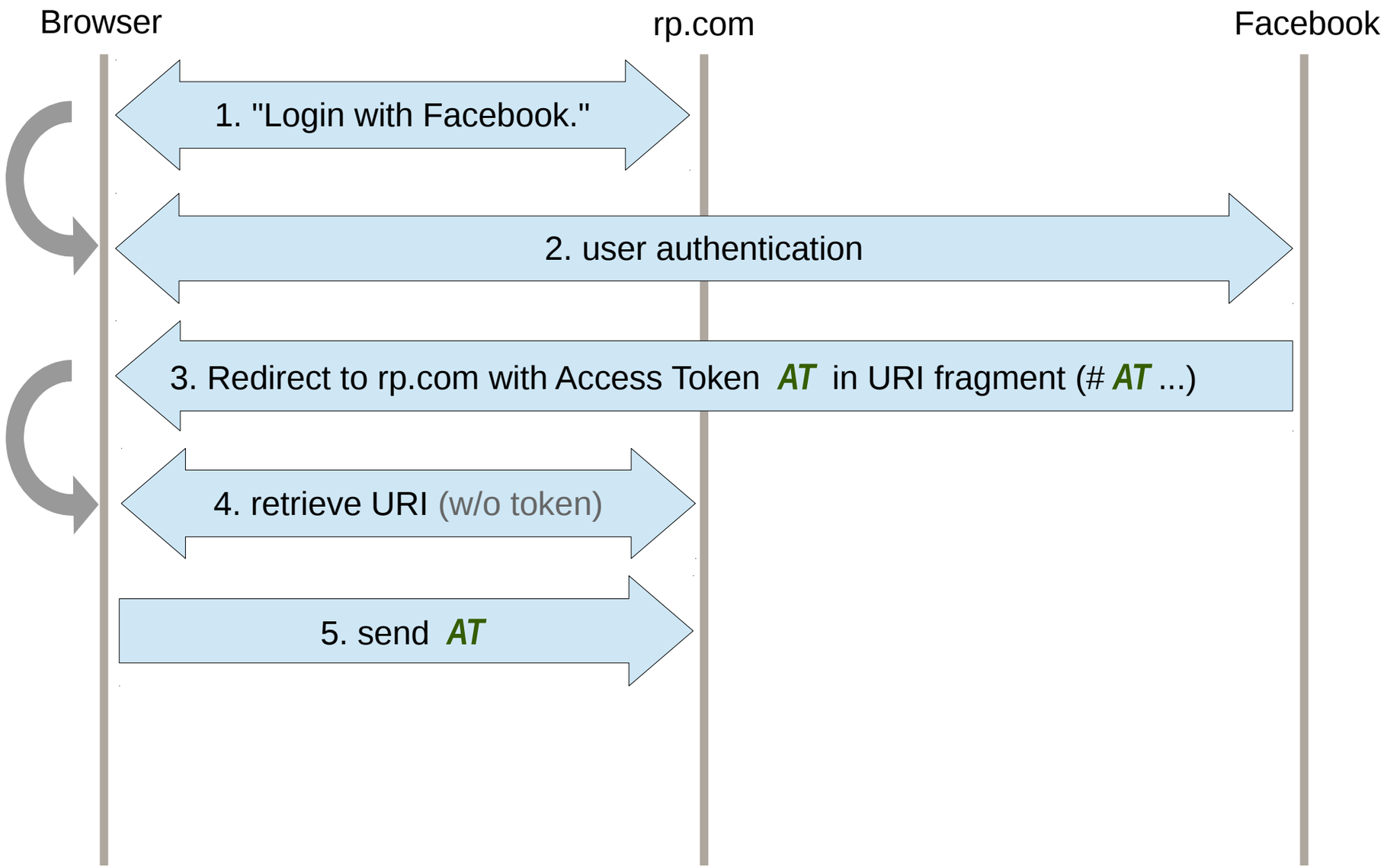
Implicit Mode



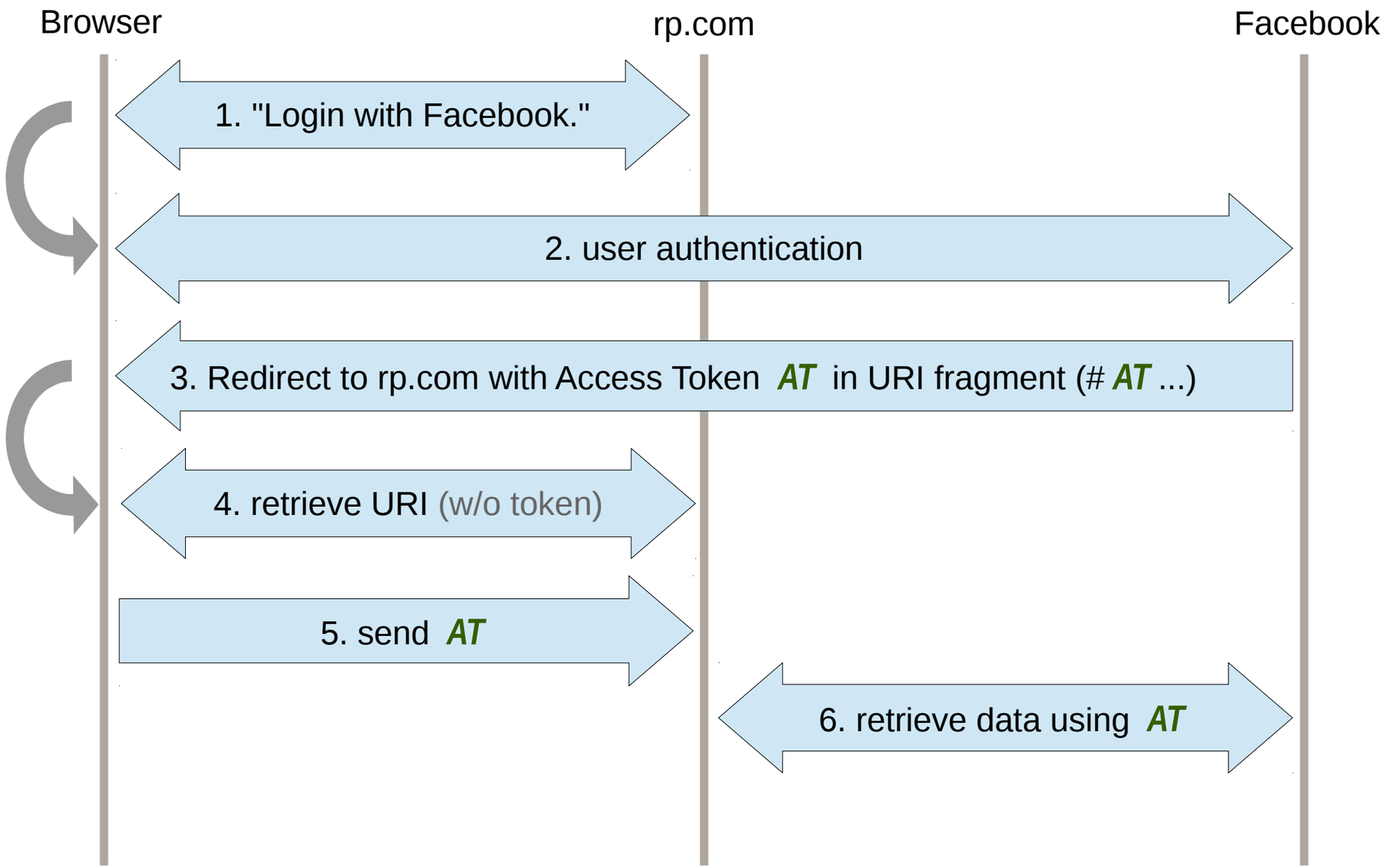
Implicit Mode



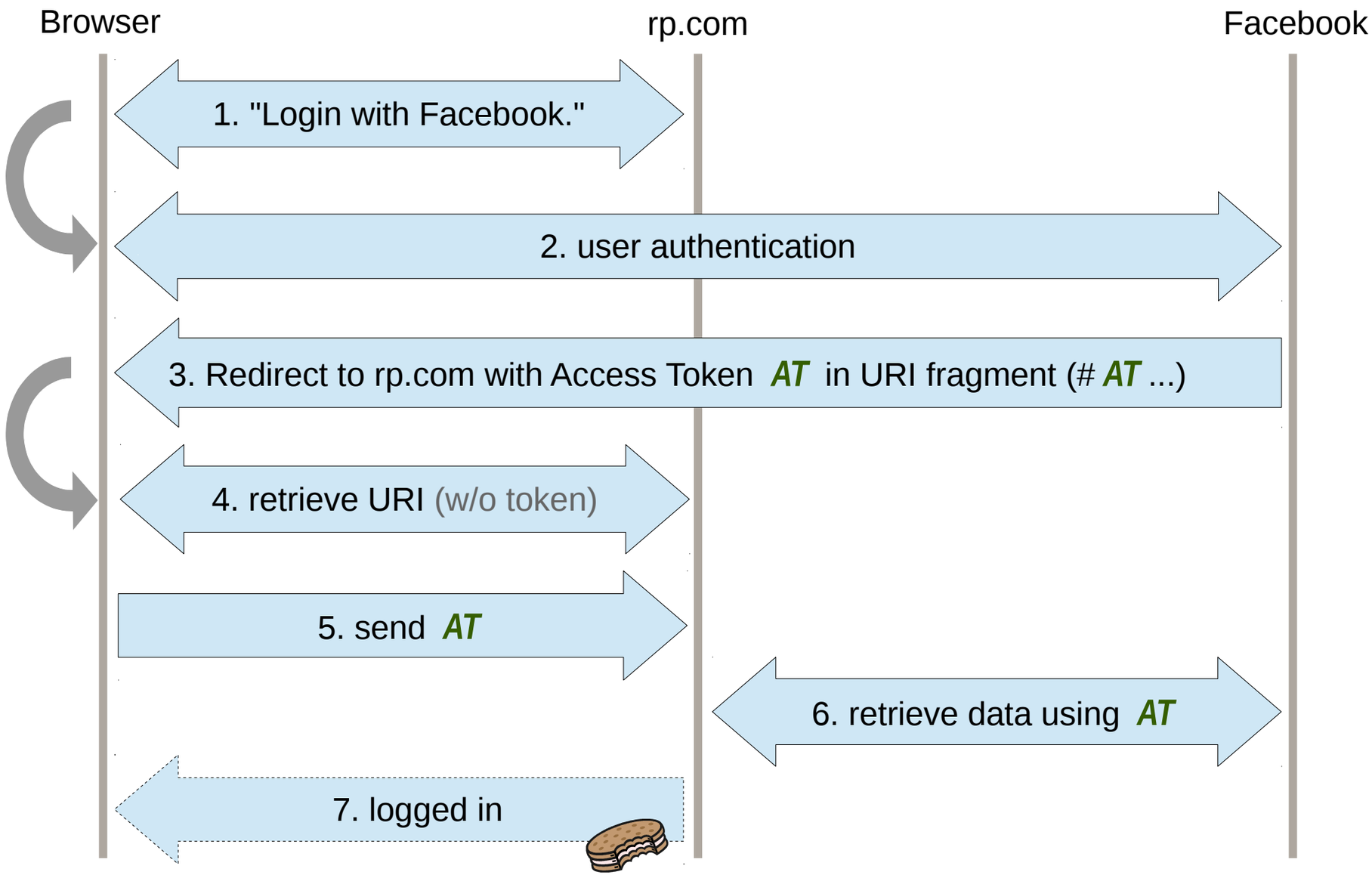
Implicit Mode



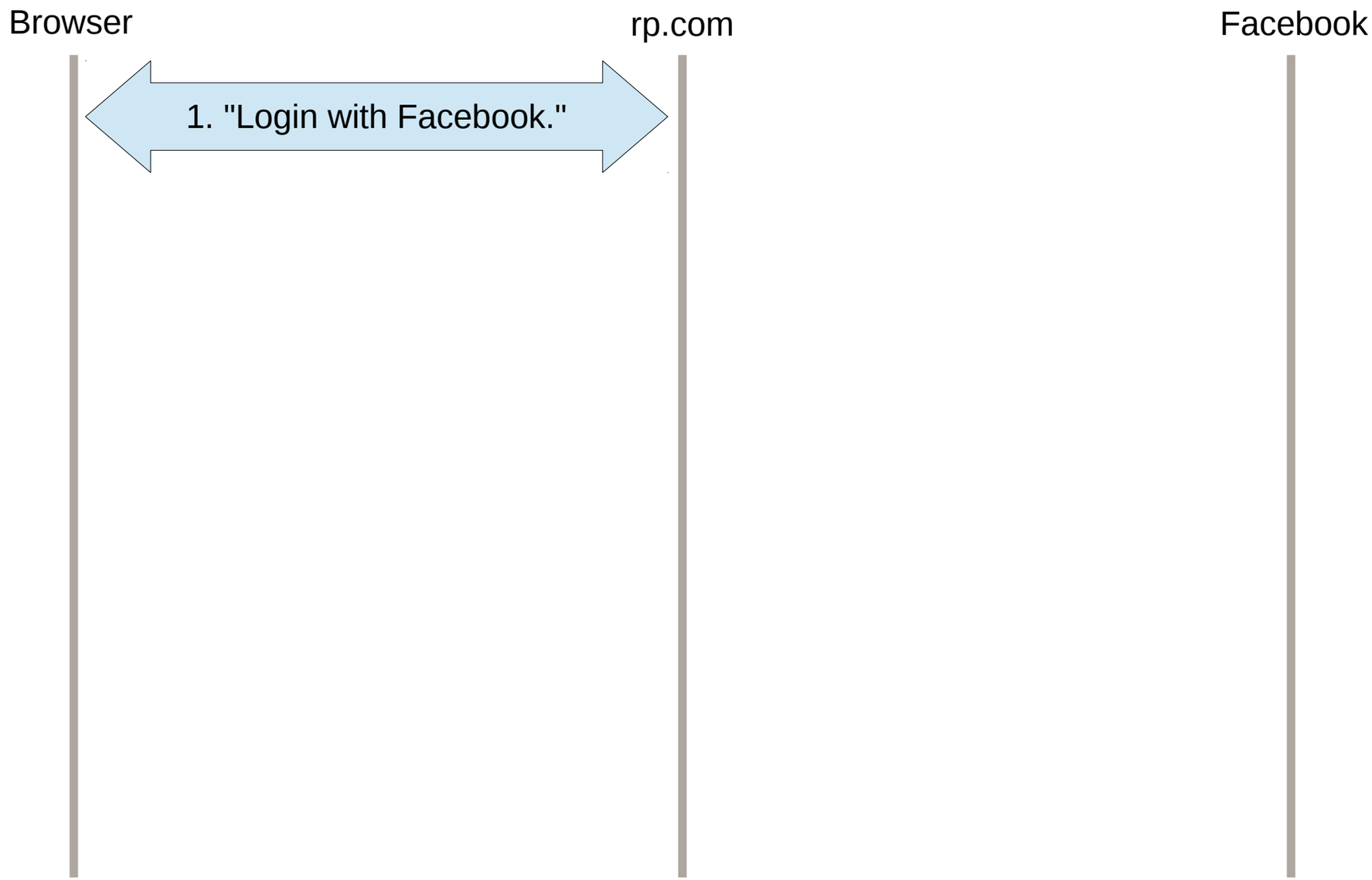
Implicit Mode



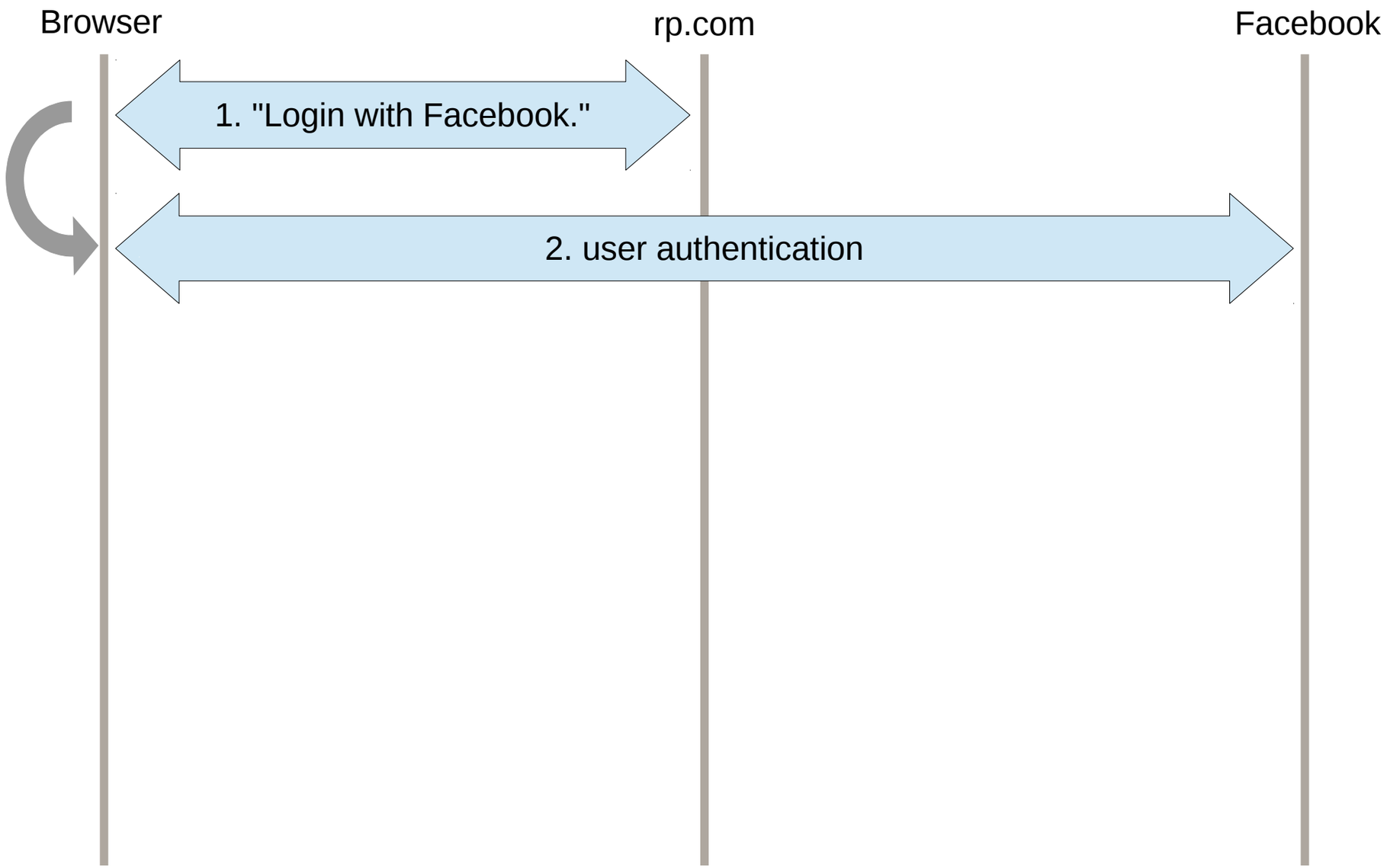
Implicit Mode



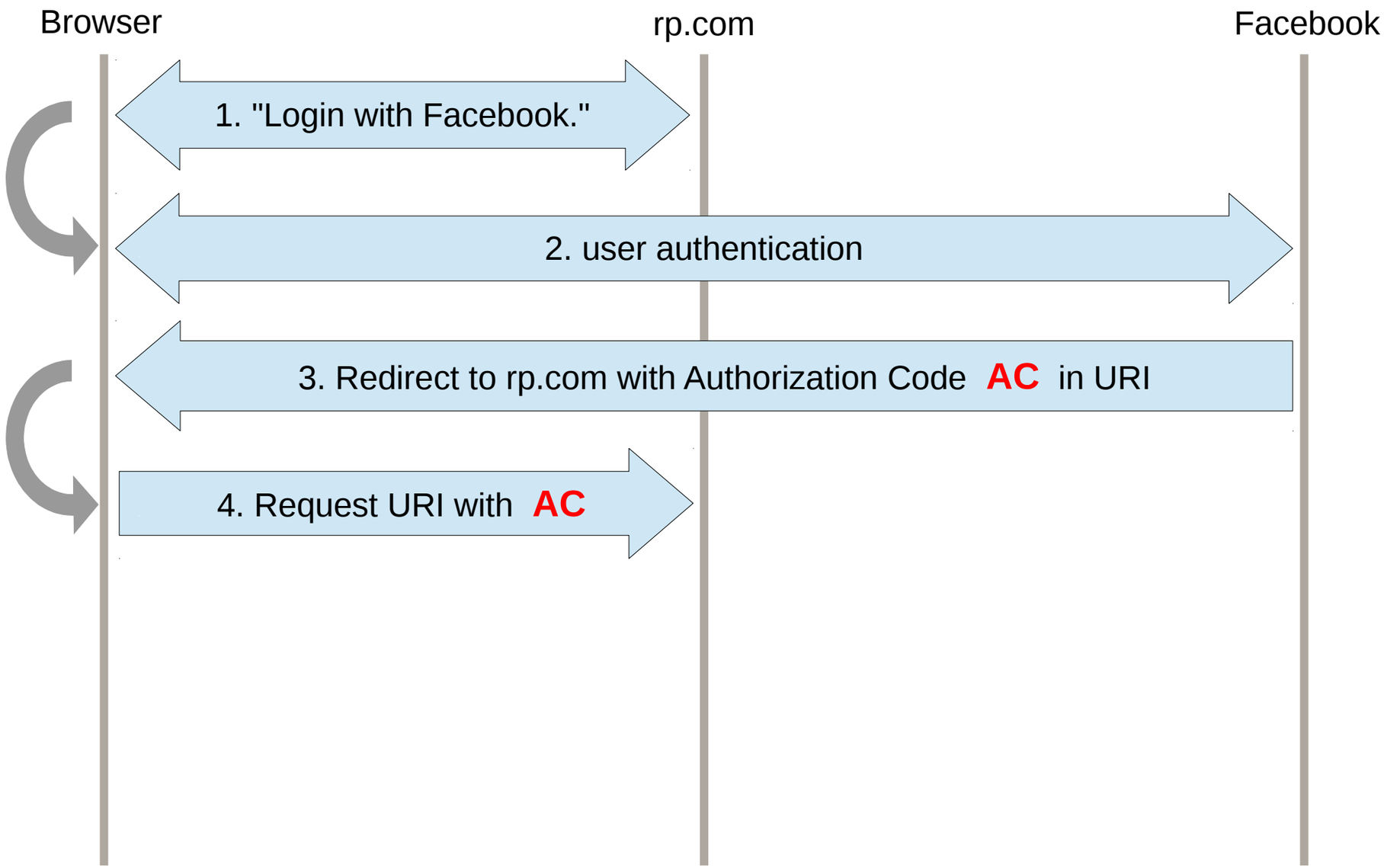
Authorization Code Mode



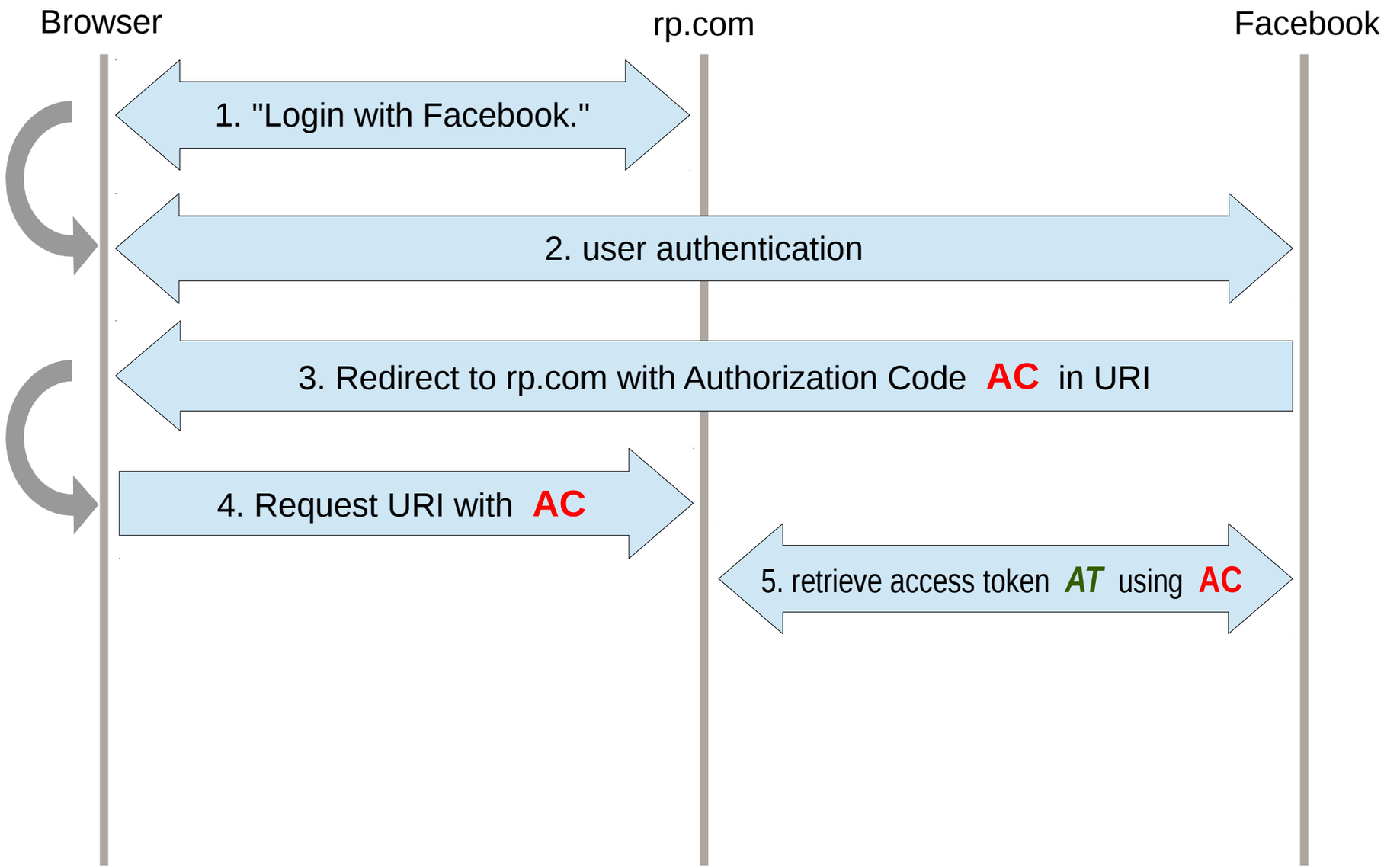
Authorization Code Mode



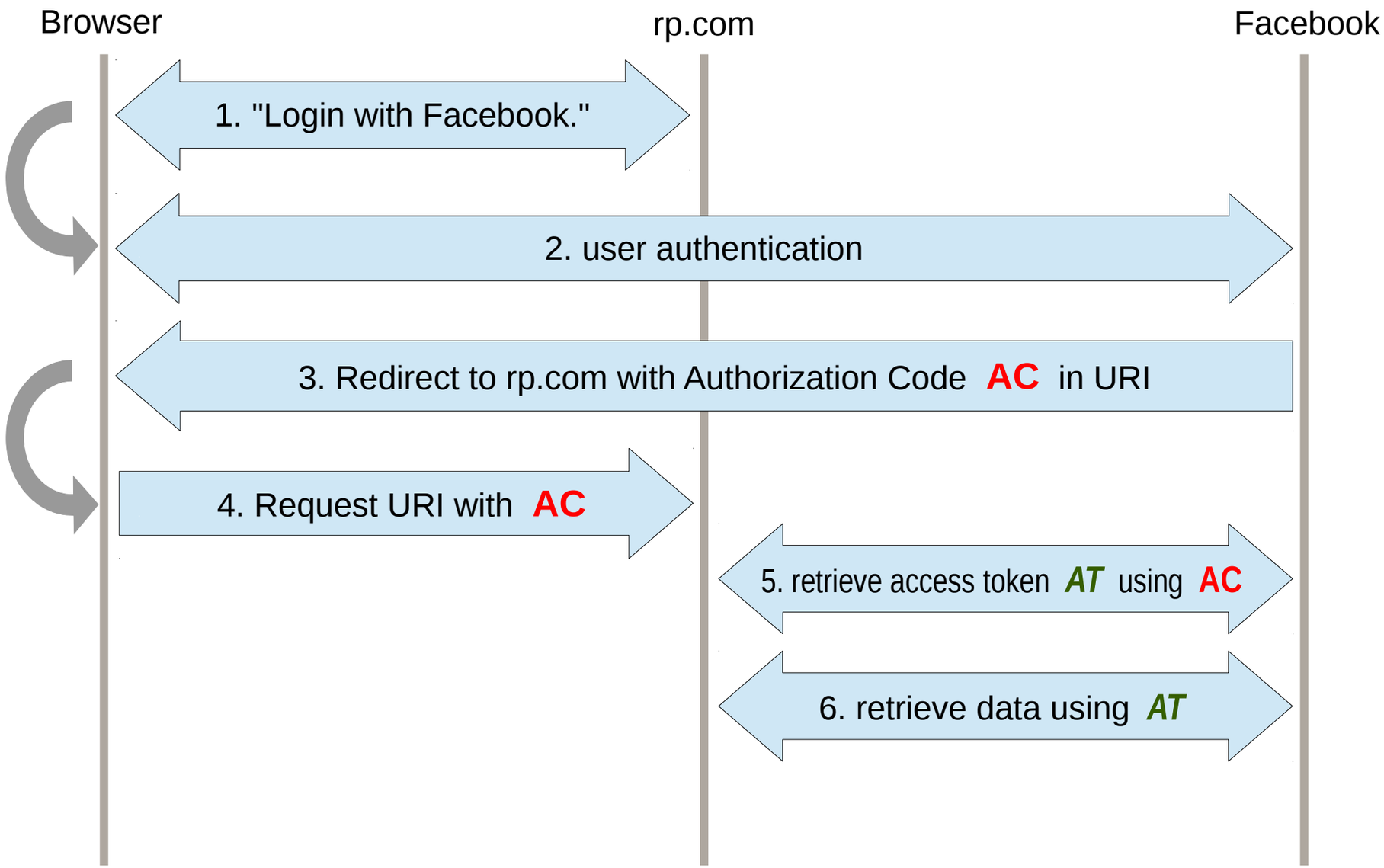
Authorization Code Mode



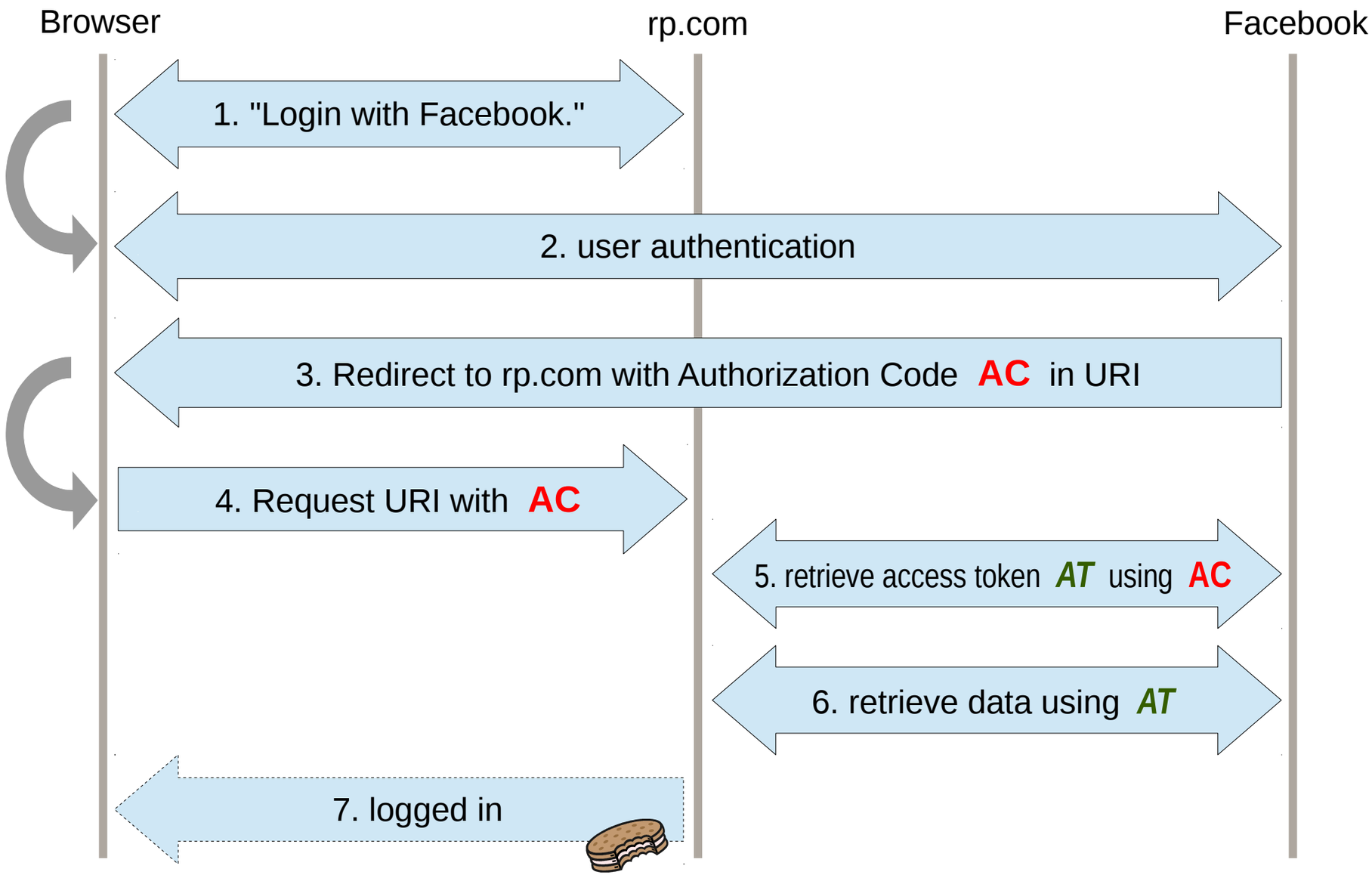
Authorization Code Mode



Authorization Code Mode



Authorization Code Mode



(Selected) Attacks on OAuth 2.0

Known Attacks...

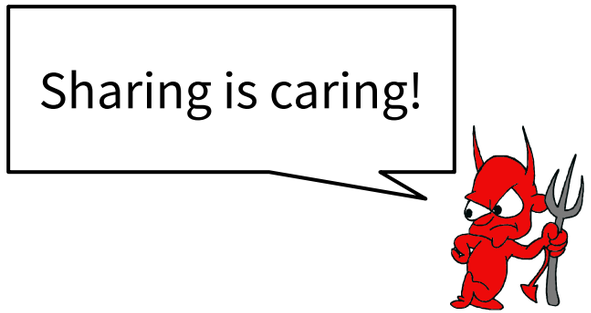
Known Attacks...

- Cut‘n‘paste attack

Known Attacks...

- Cut‘n‘paste attack

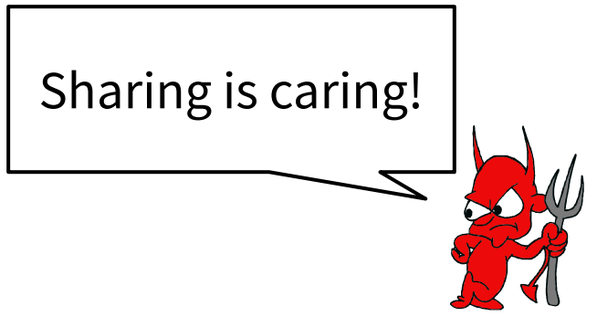
- Lack of HTTPS



Known Attacks...

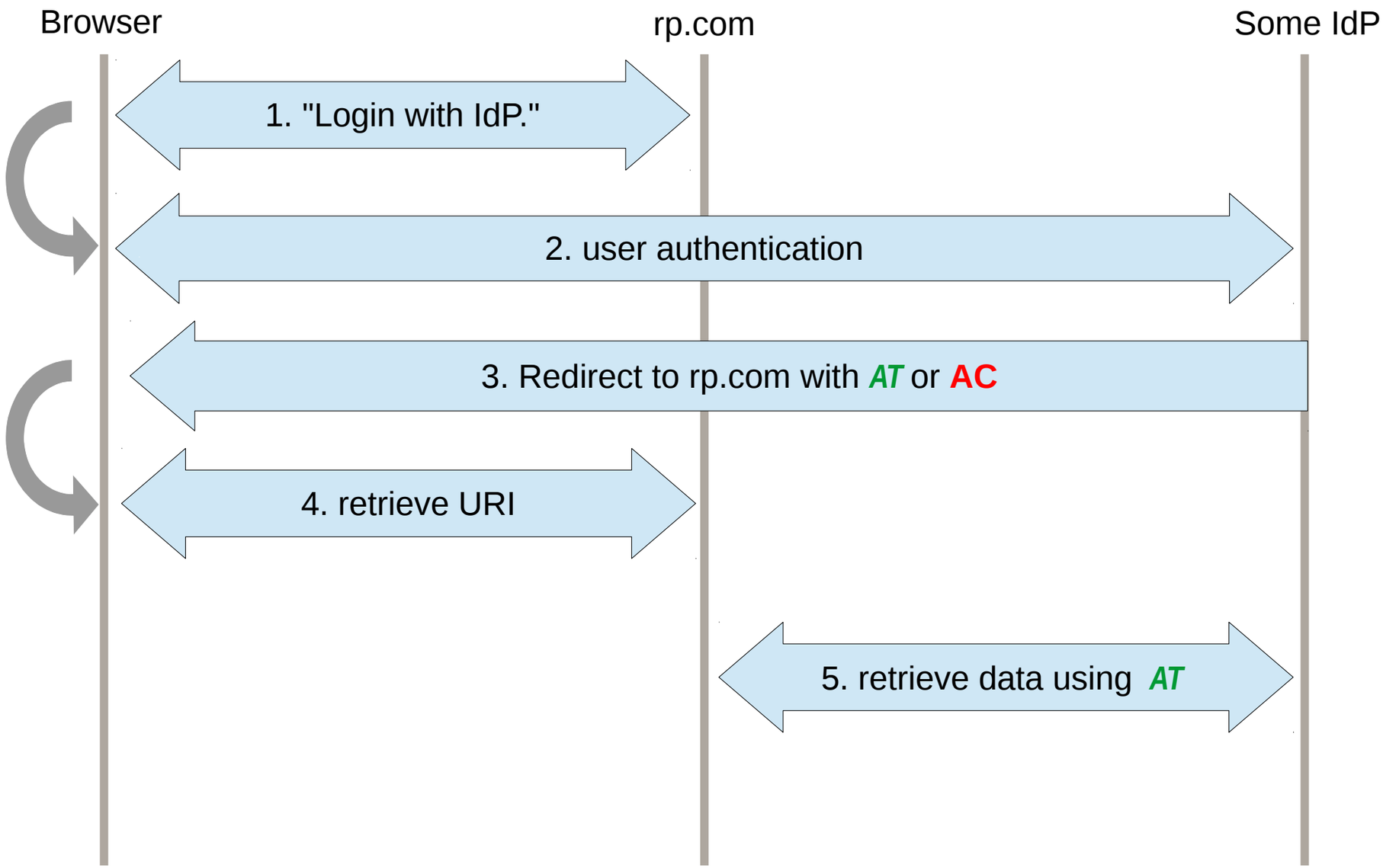
- Cut‘n‘paste attack

- Lack of HTTPS

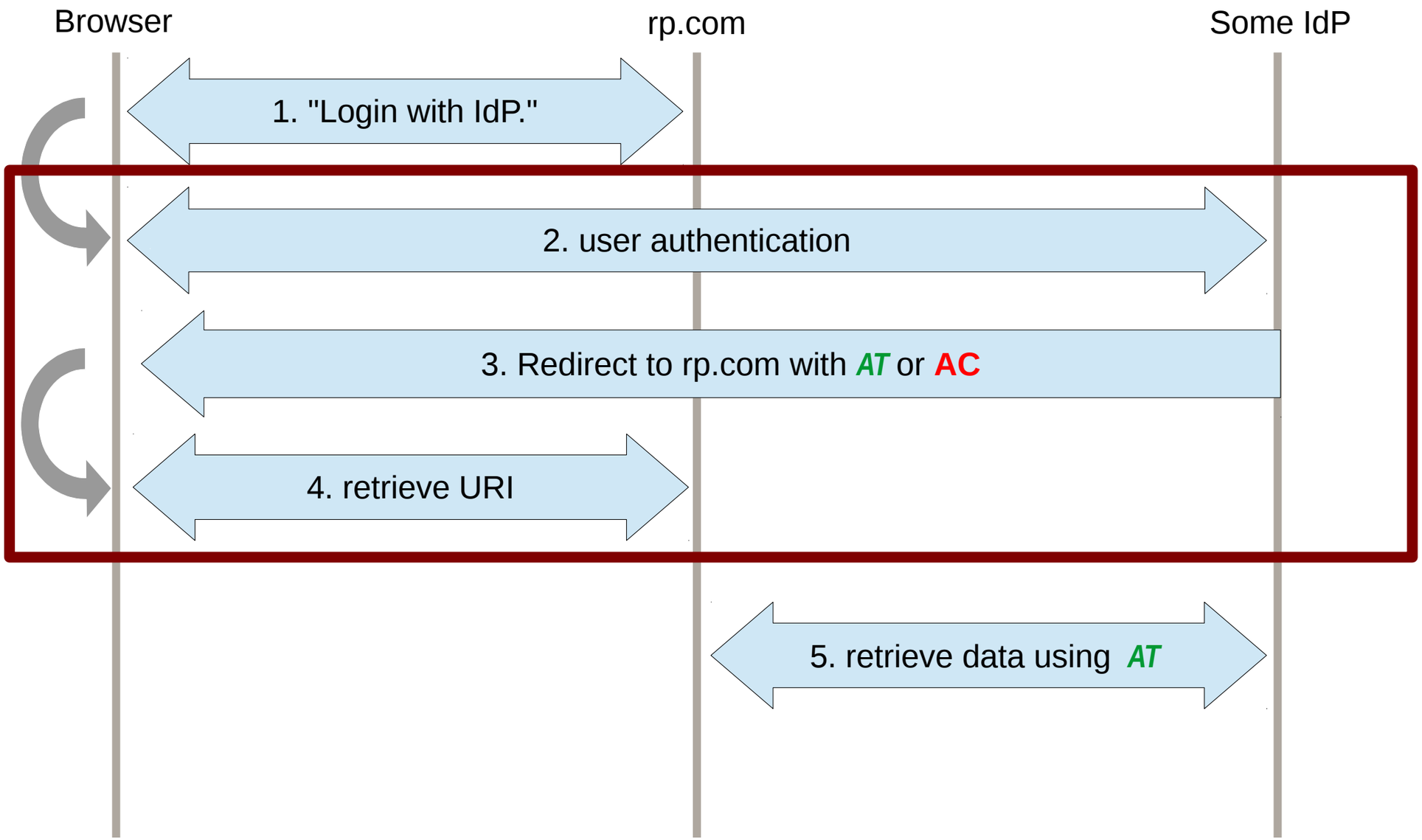


- Setting Cookies via HTTP

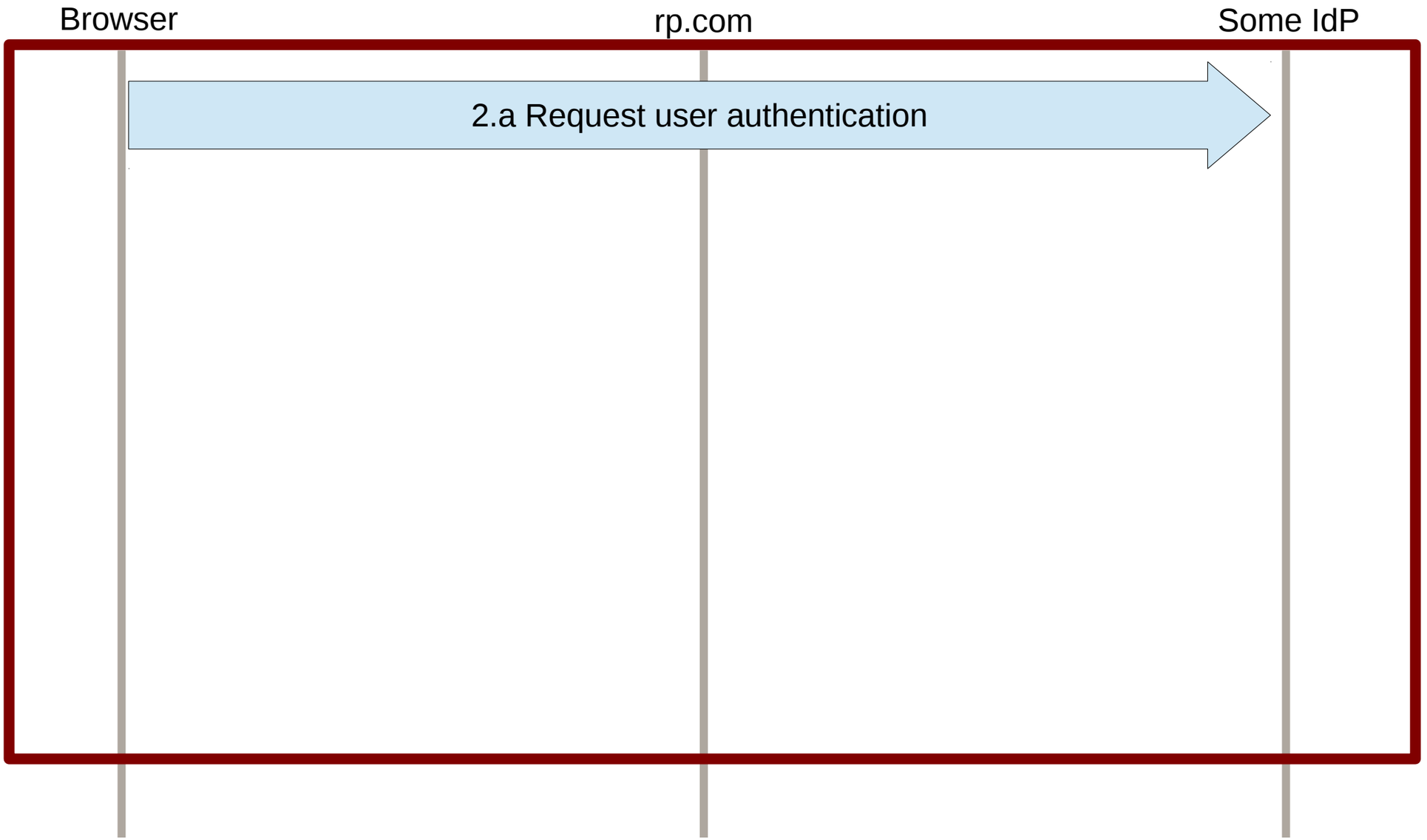
307 Redirect Attack



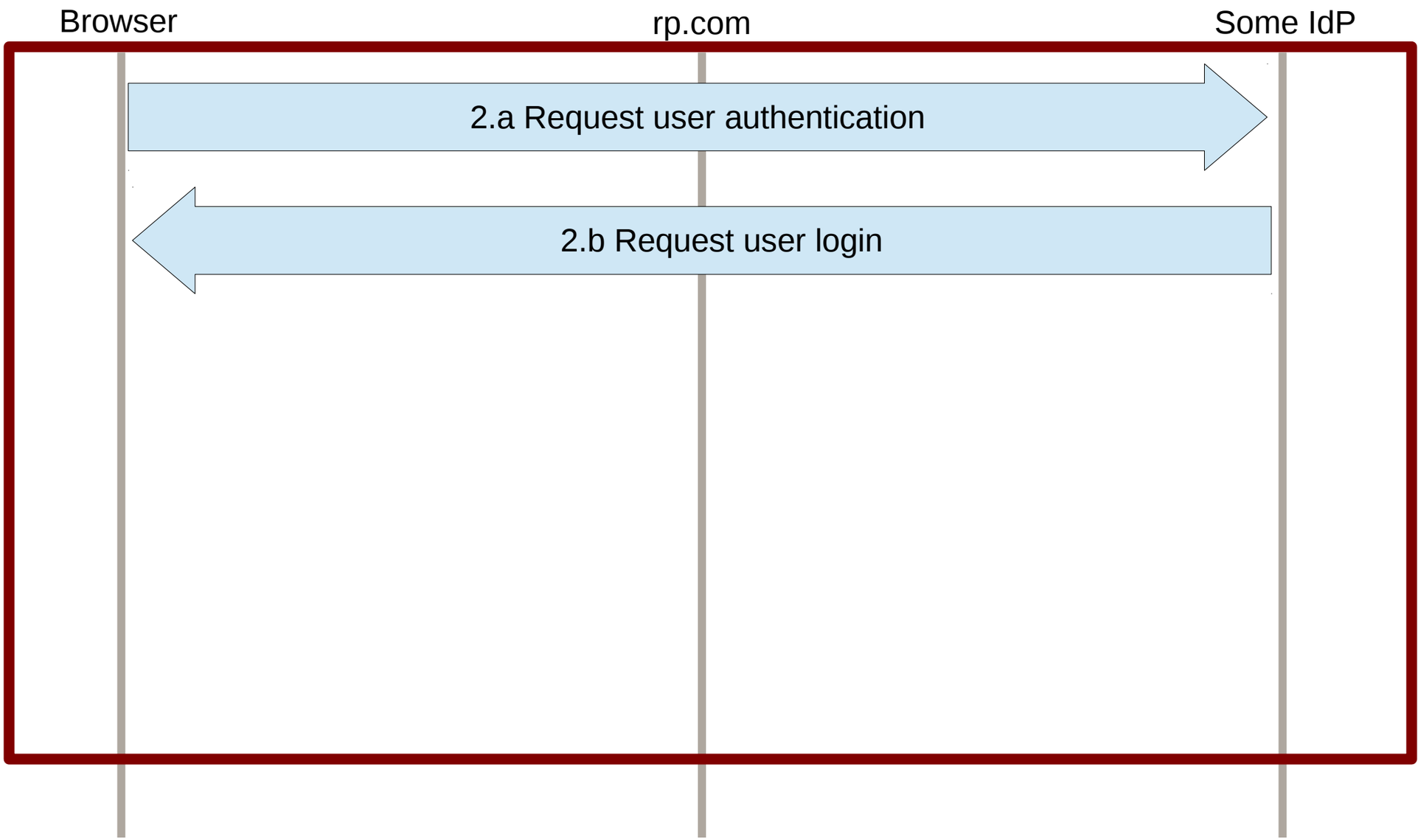
307 Redirect Attack



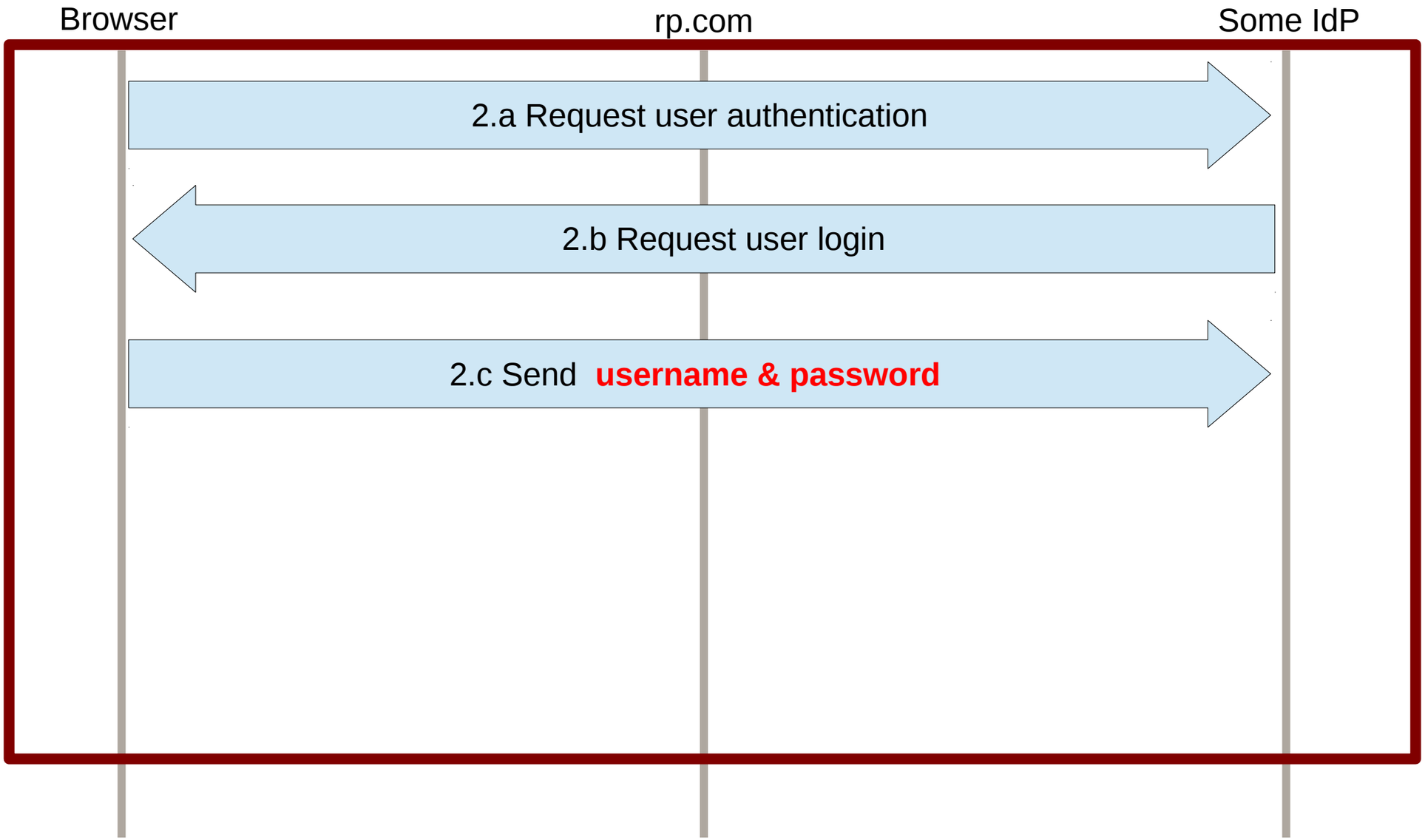
307 Redirect Attack



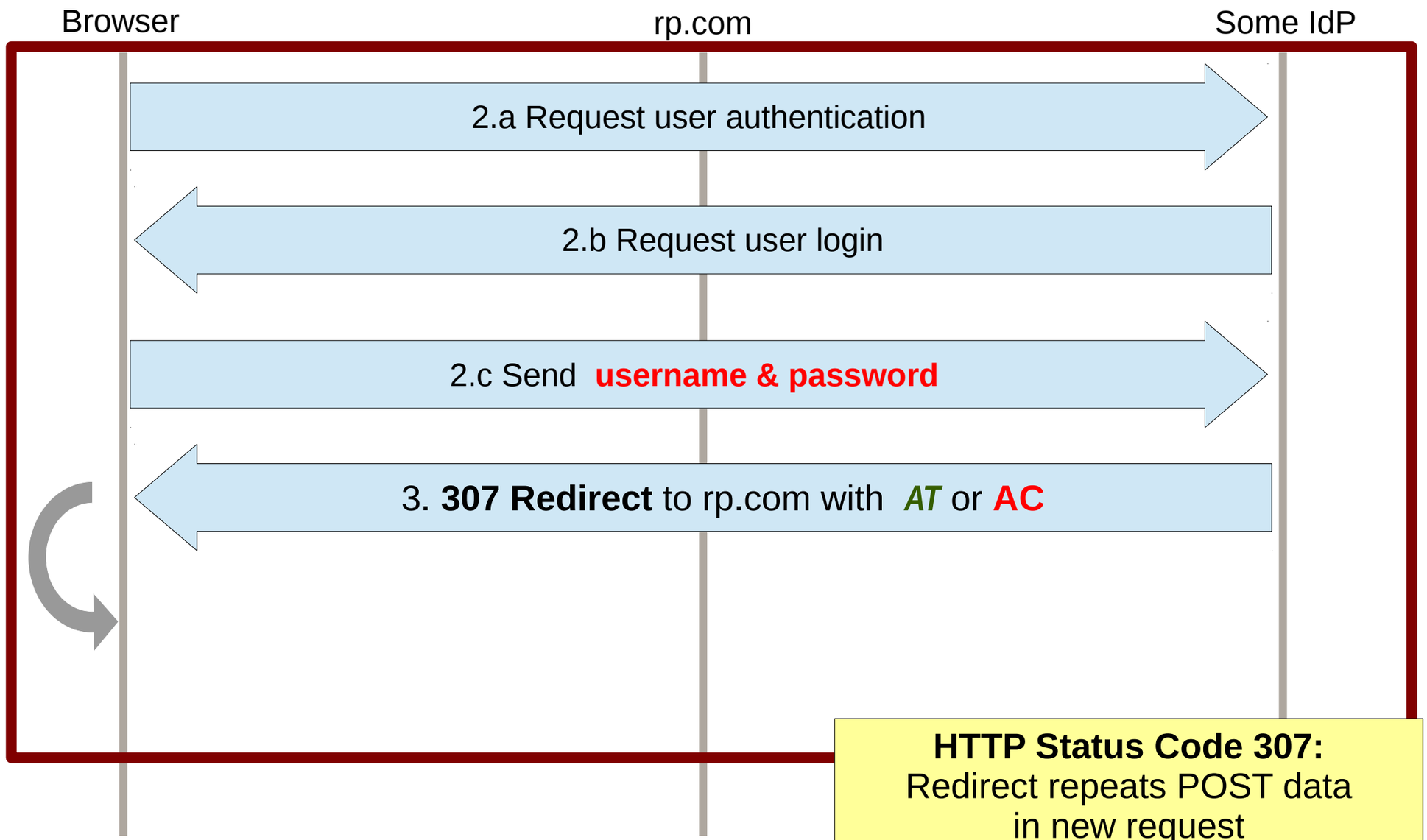
307 Redirect Attack



307 Redirect Attack

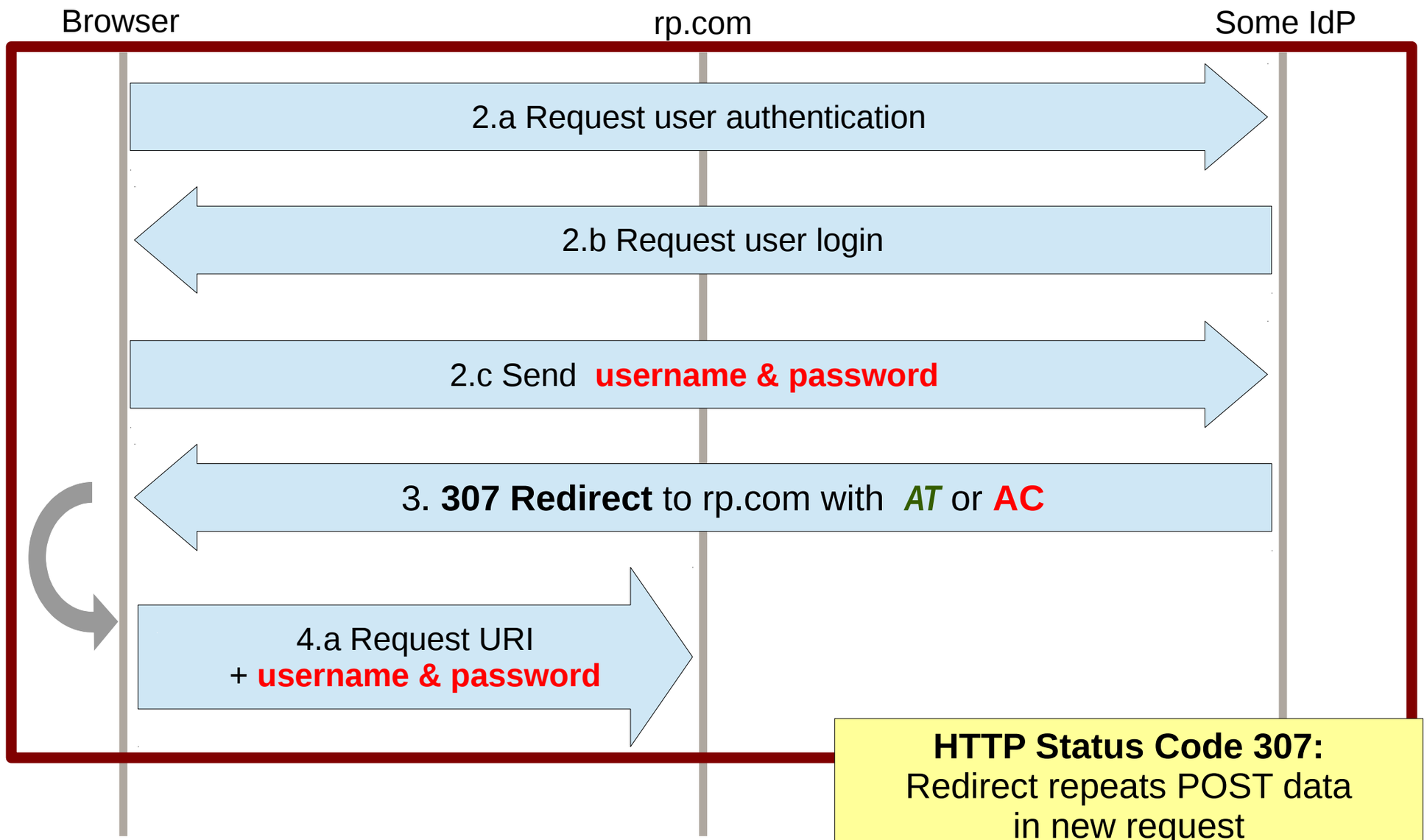


307 Redirect Attack



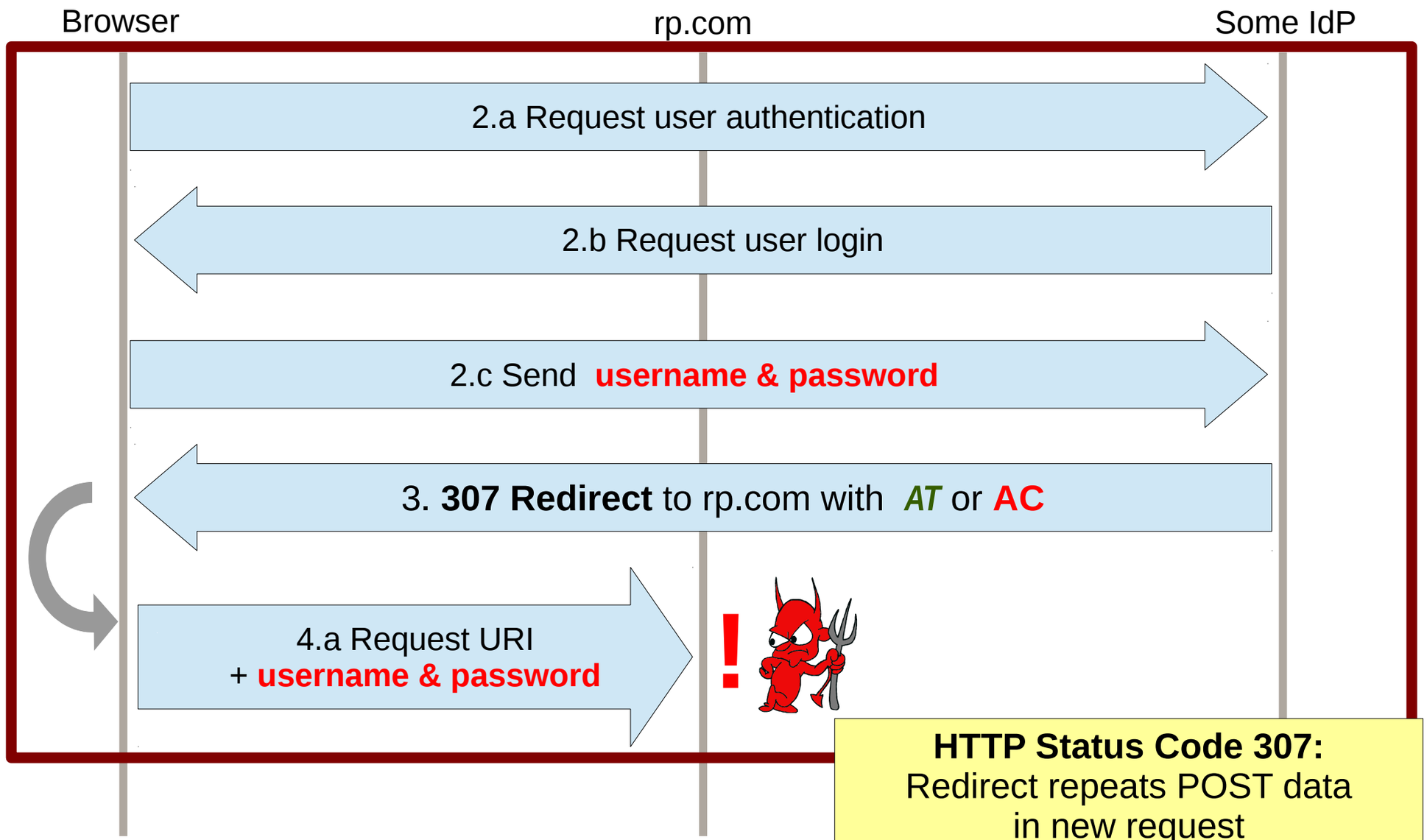
HTTP Status Code 307:
Redirect repeats POST data
in new request

307 Redirect Attack



HTTP Status Code 307:
Redirect repeats POST data
in new request

307 Redirect Attack



HTTP Status Code 307:
Redirect repeats POST data
in new request

307 Redirect Attack

OAuth standard says:

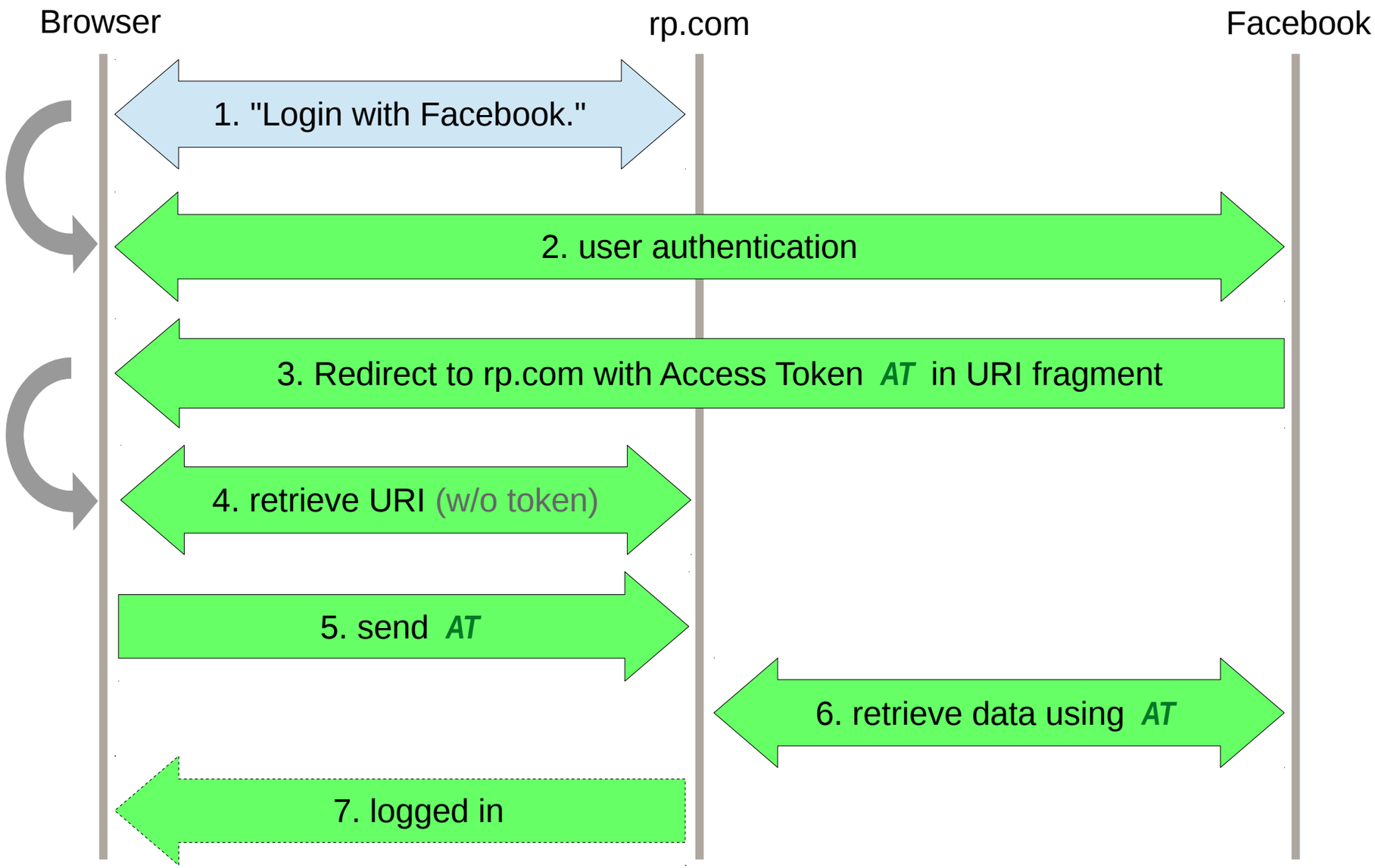
1.7. HTTP Redirections

This specification makes extensive use of HTTP redirections, in which the client or the authorization server directs the resource owner's user-agent to another destination. While the examples in this specification show the use of the HTTP 302 status code, any other method available via the user-agent to accomplish this redirection is allowed and is considered to be an implementation detail.

Mitigation:

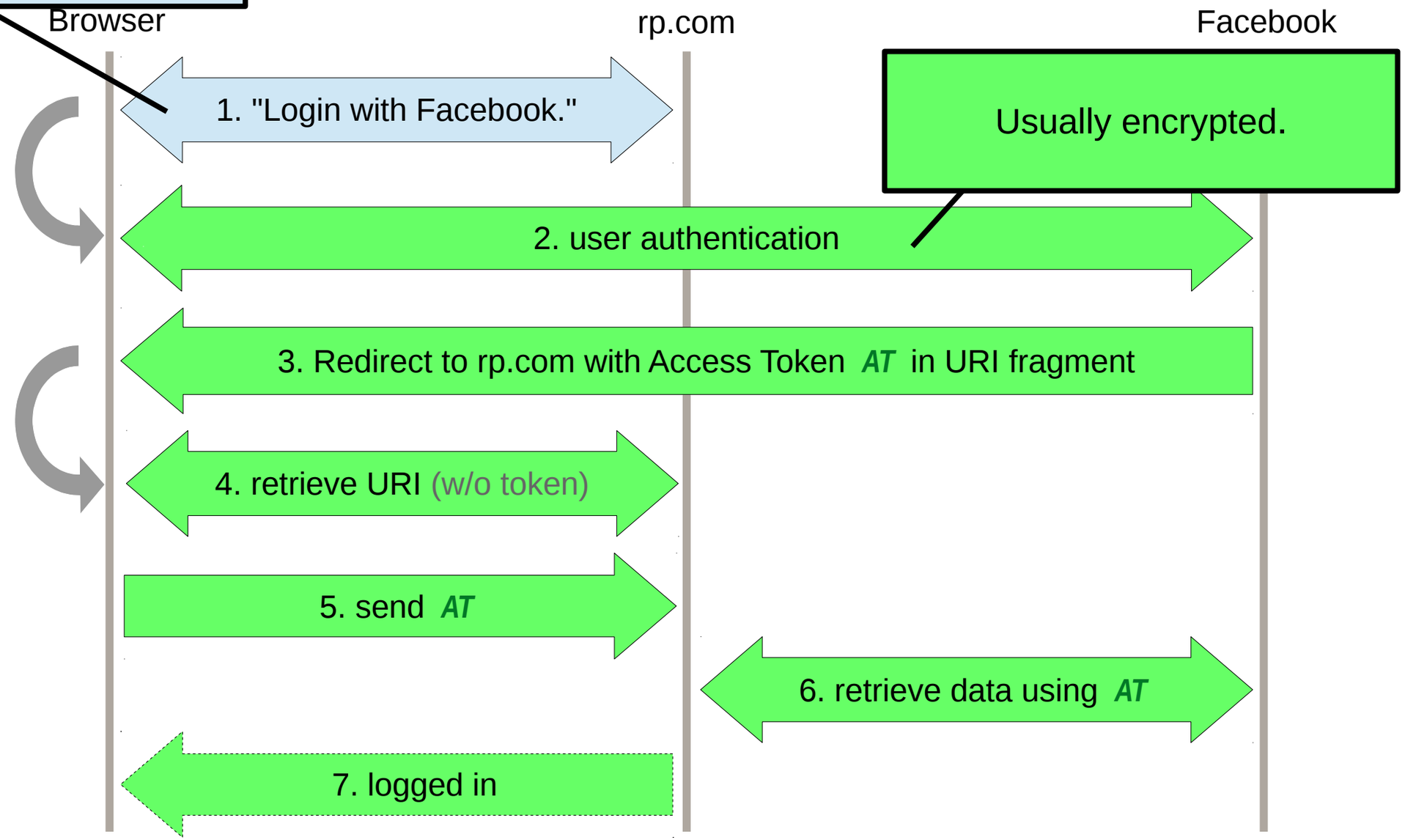
Make 303 or any other method that does not forward POST data.

IdP Mix-Up Attack in Implicit Mode

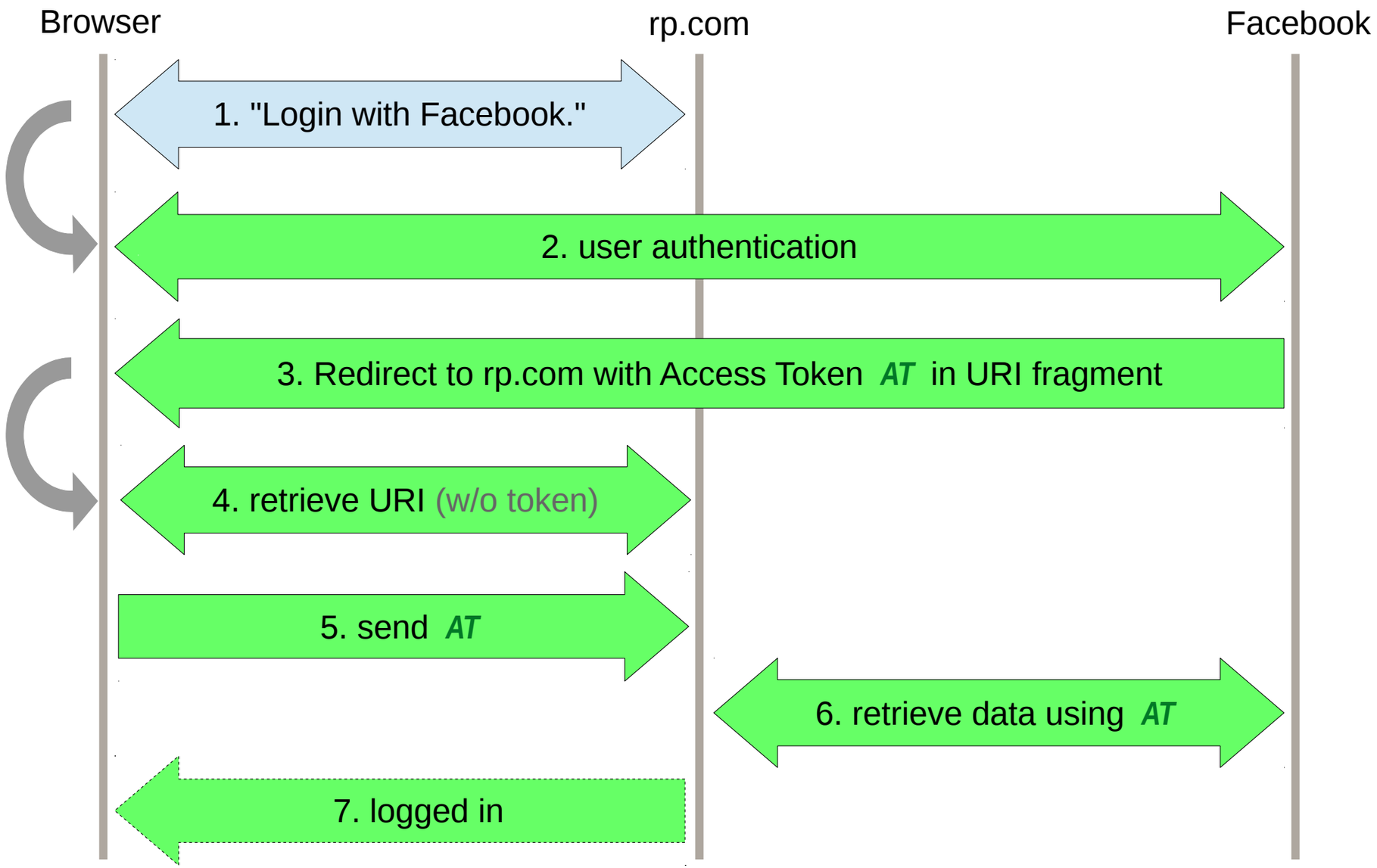


IdP Mix-Up Attack in Implicit Mode

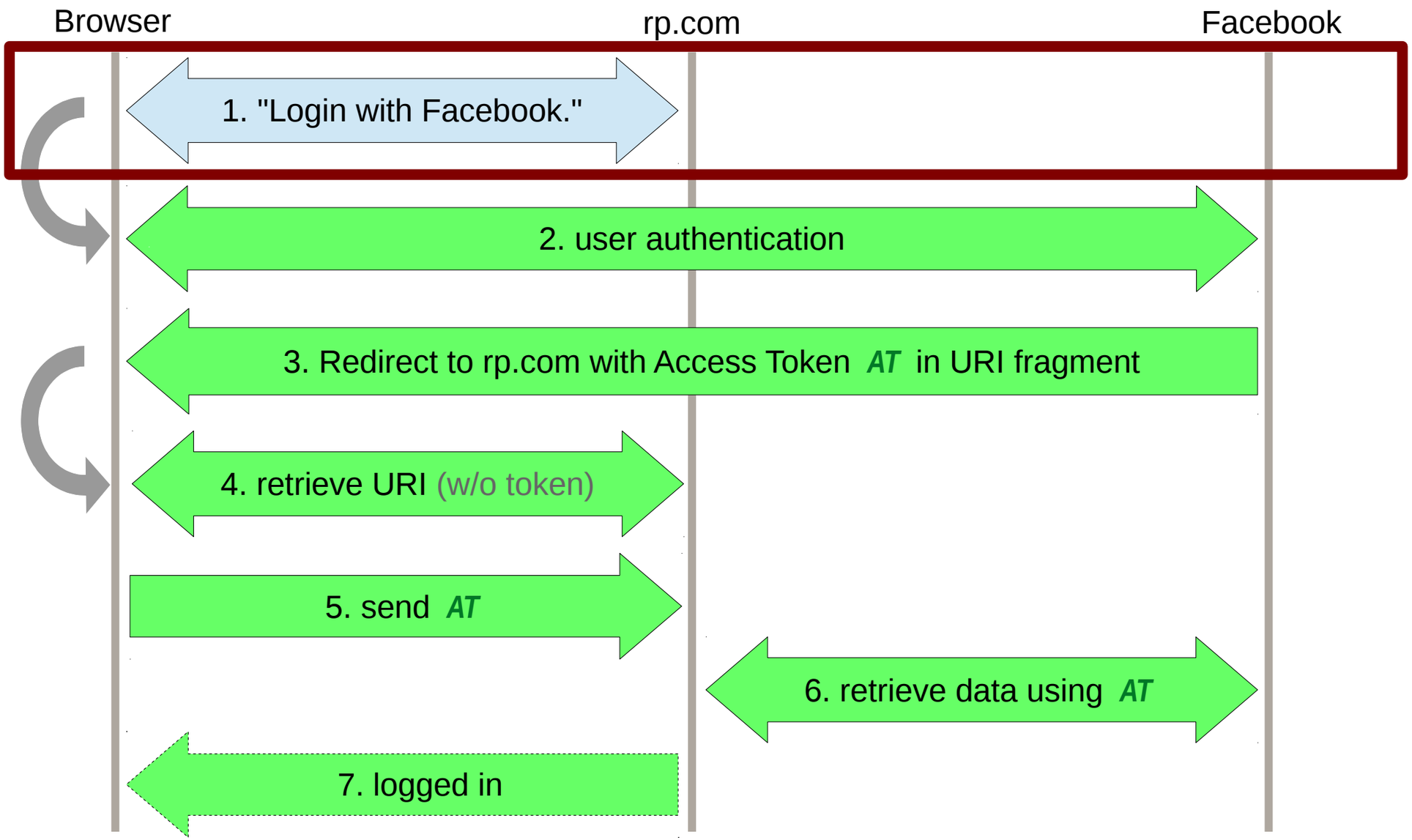
Often unencrypted or vulnerable to TLS stripping since **no confidential data** is transfered.



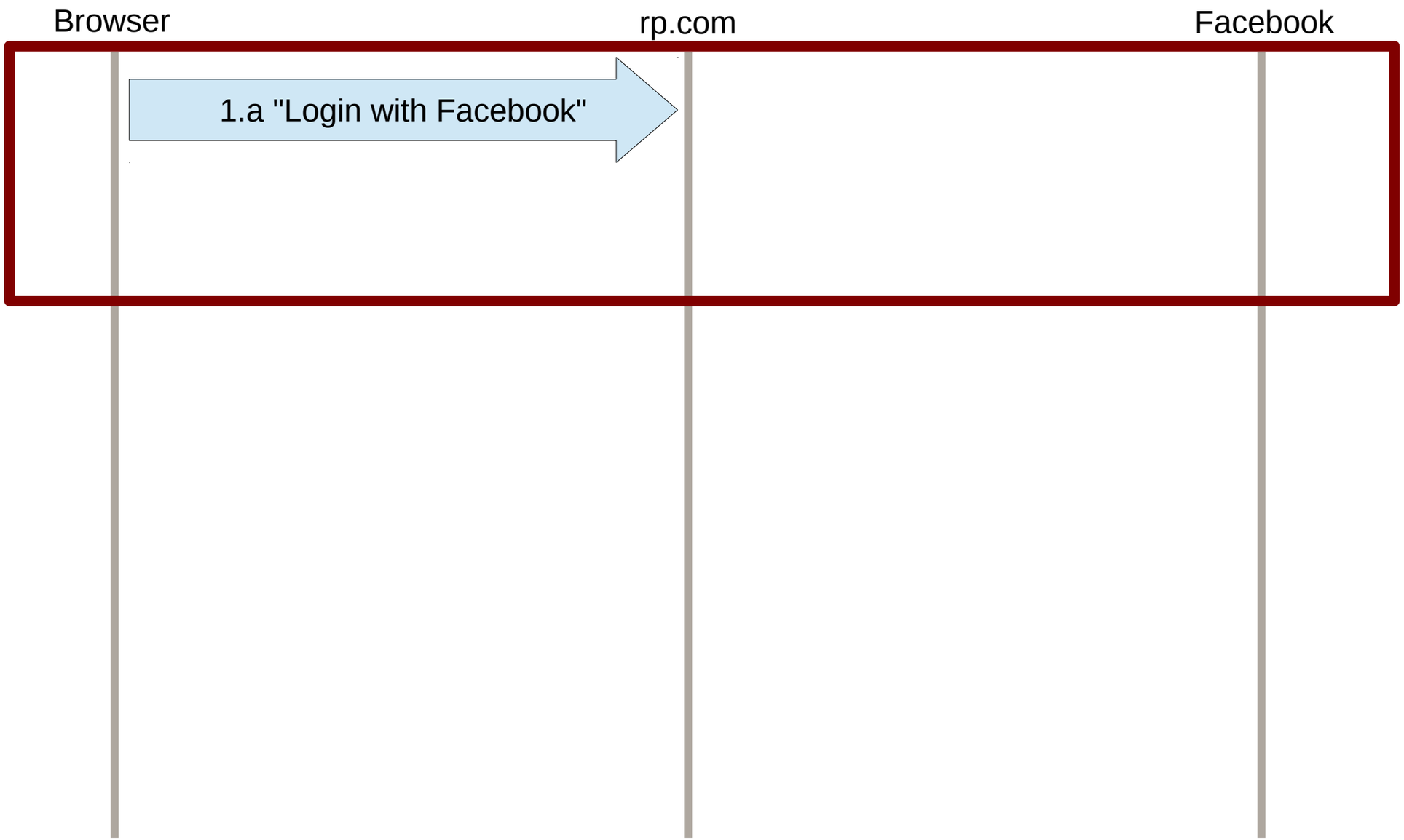
IdP Mix-Up Attack in Implicit Mode



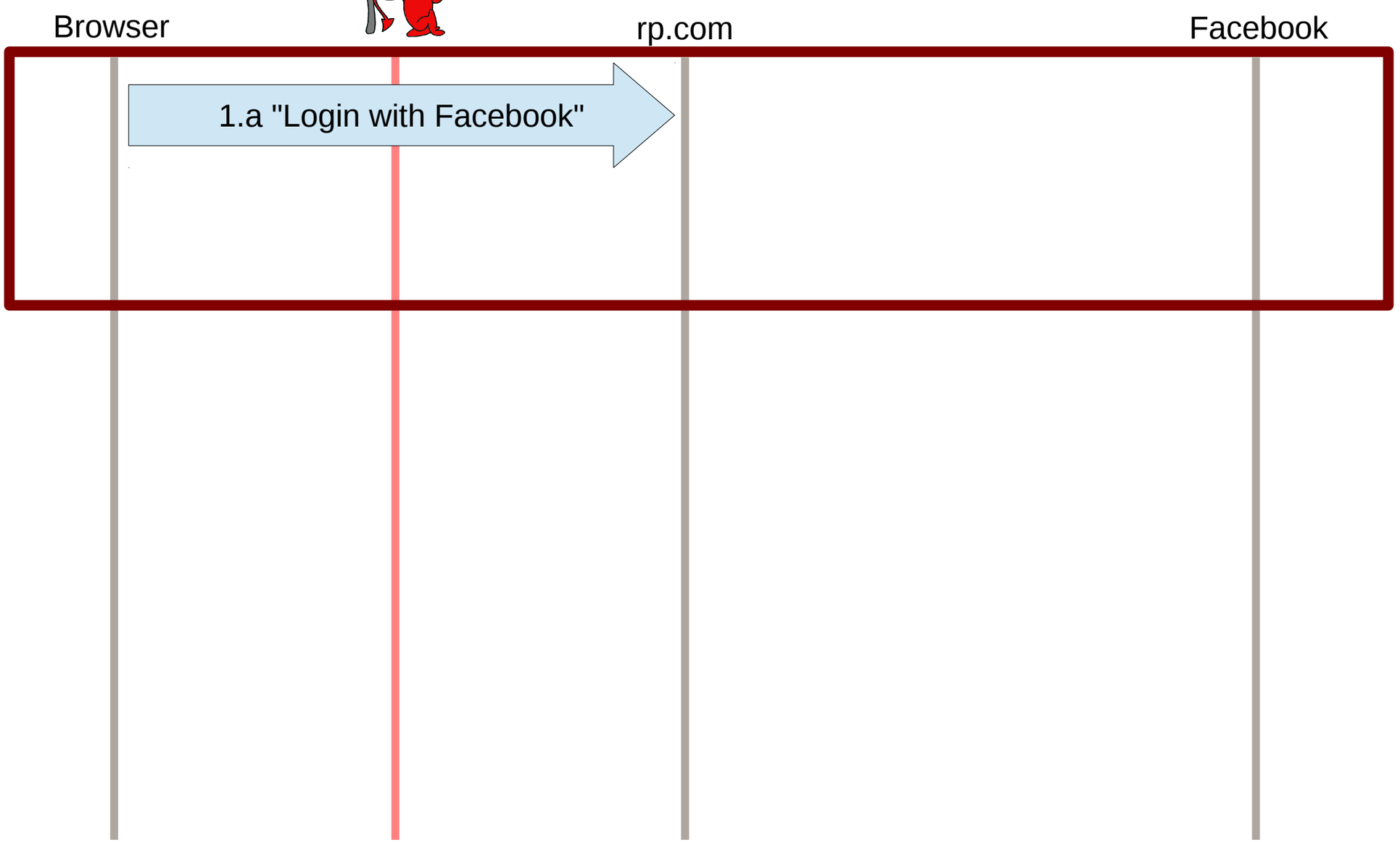
IdP Mix-Up Attack in Implicit Mode



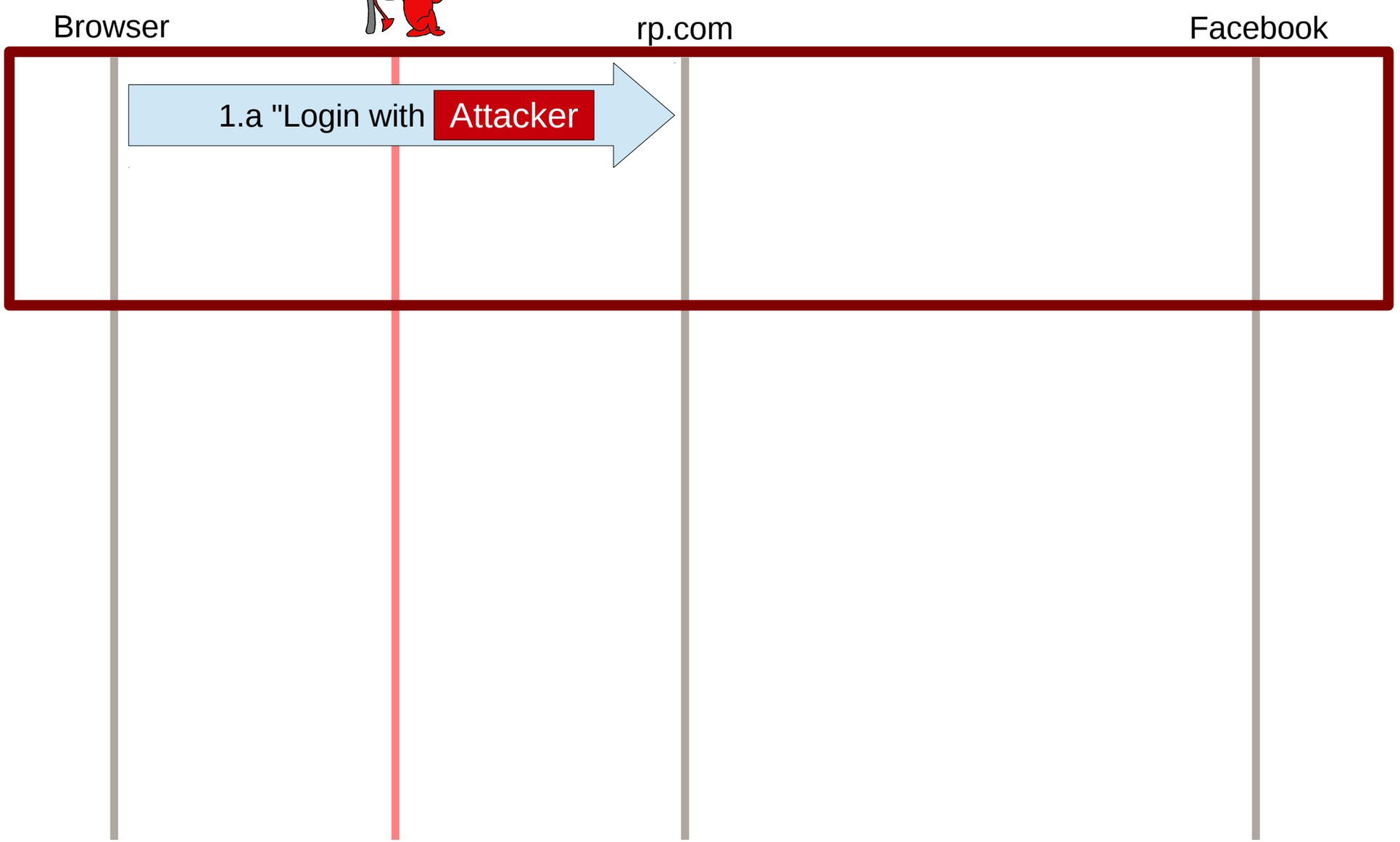
IdP Mix-Up Attack in Implicit Mode



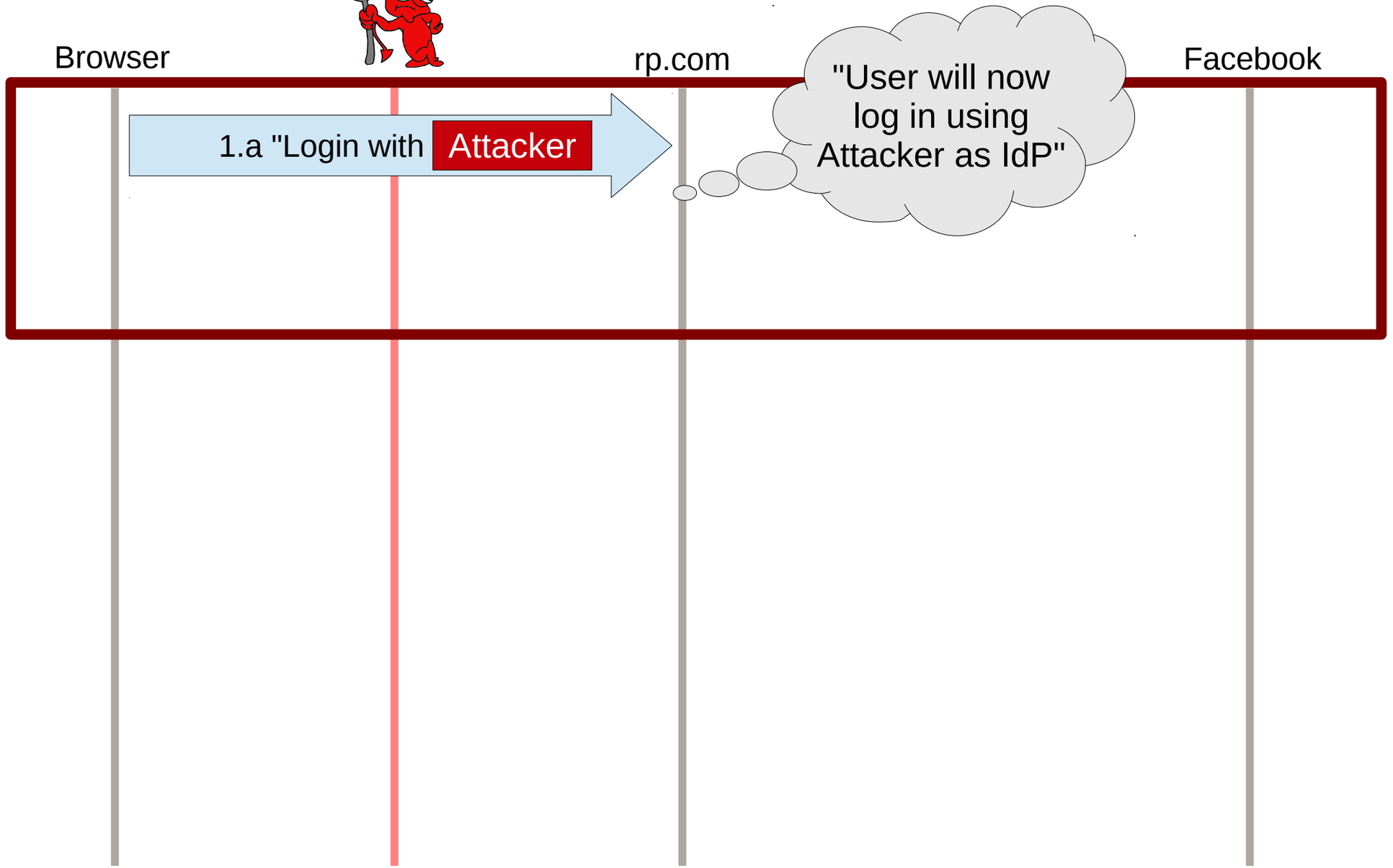
IdP Mix-Up Attack in Implicit Mode



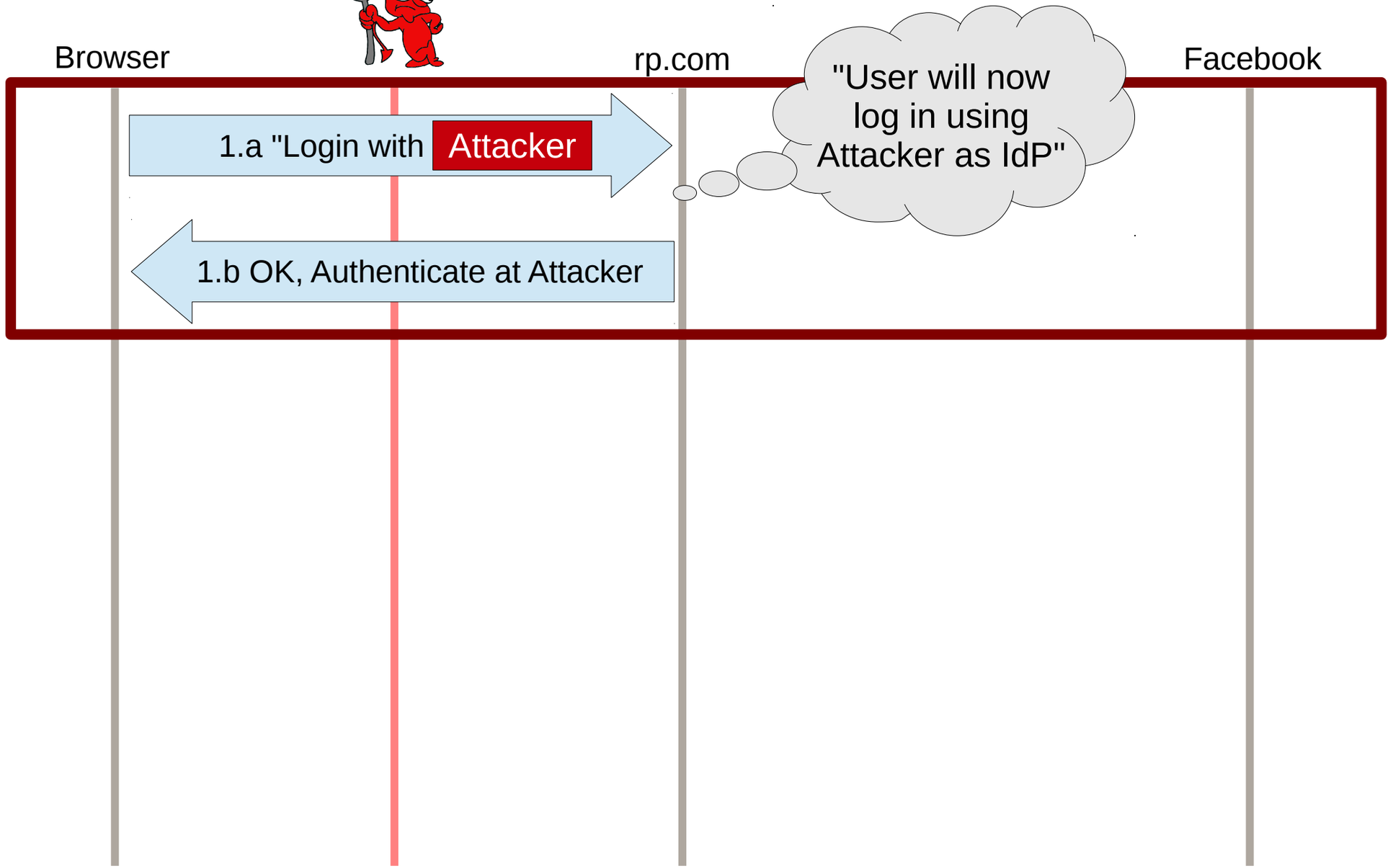
IdP Mix-Up Attack in Implicit Mode



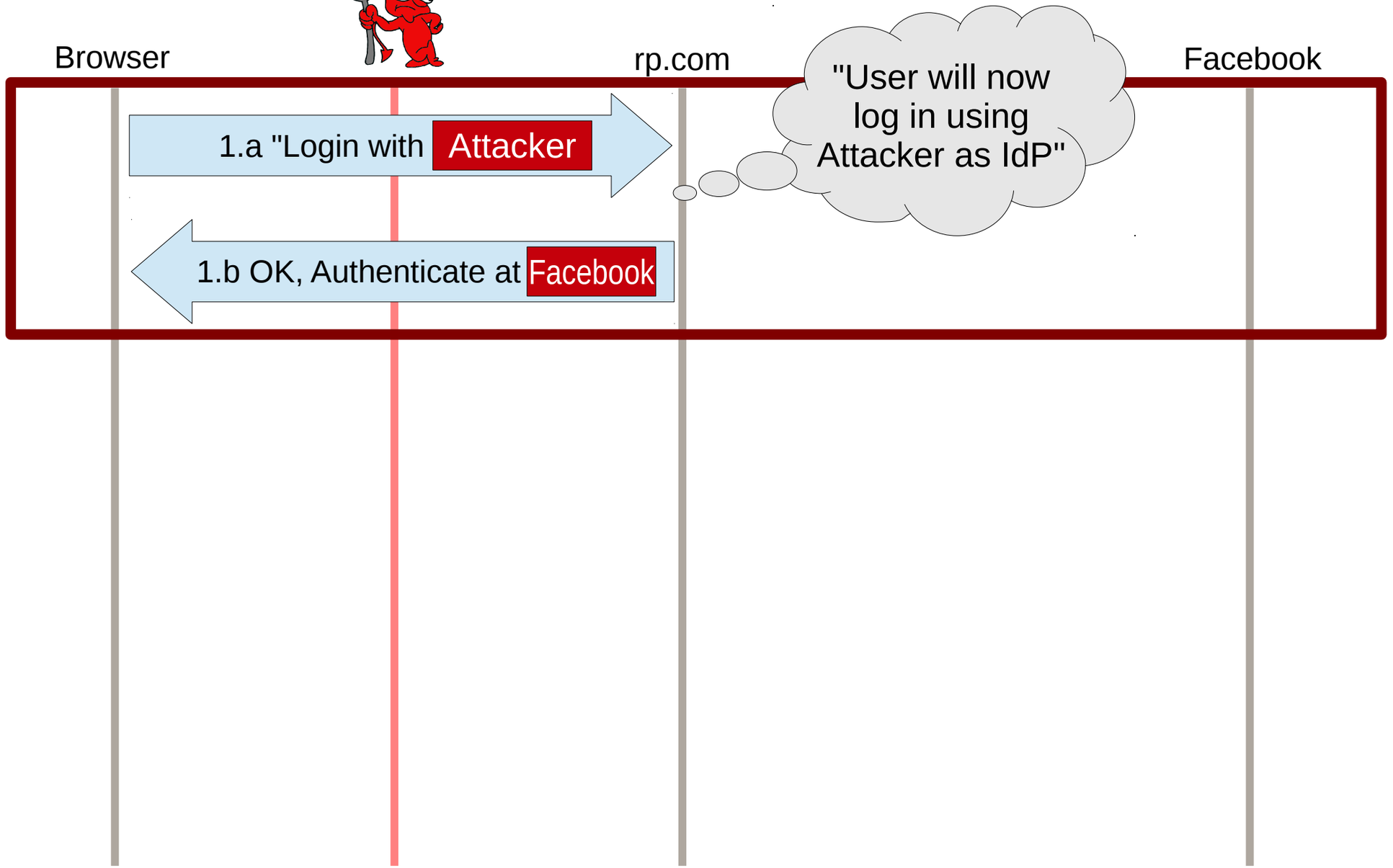
IdP Mix-Up Attack in Implicit Mode



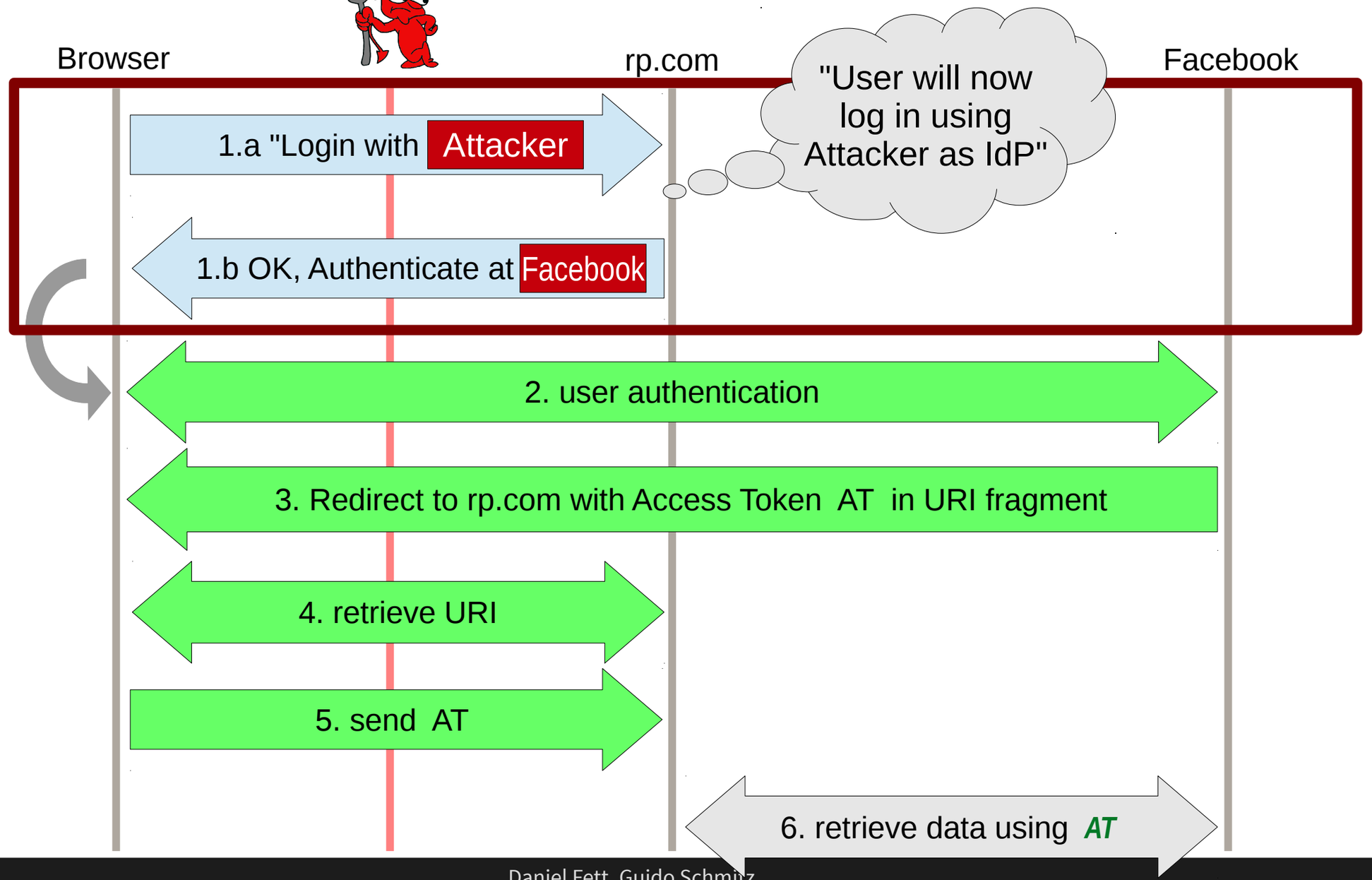
IdP Mix-Up Attack in Implicit Mode



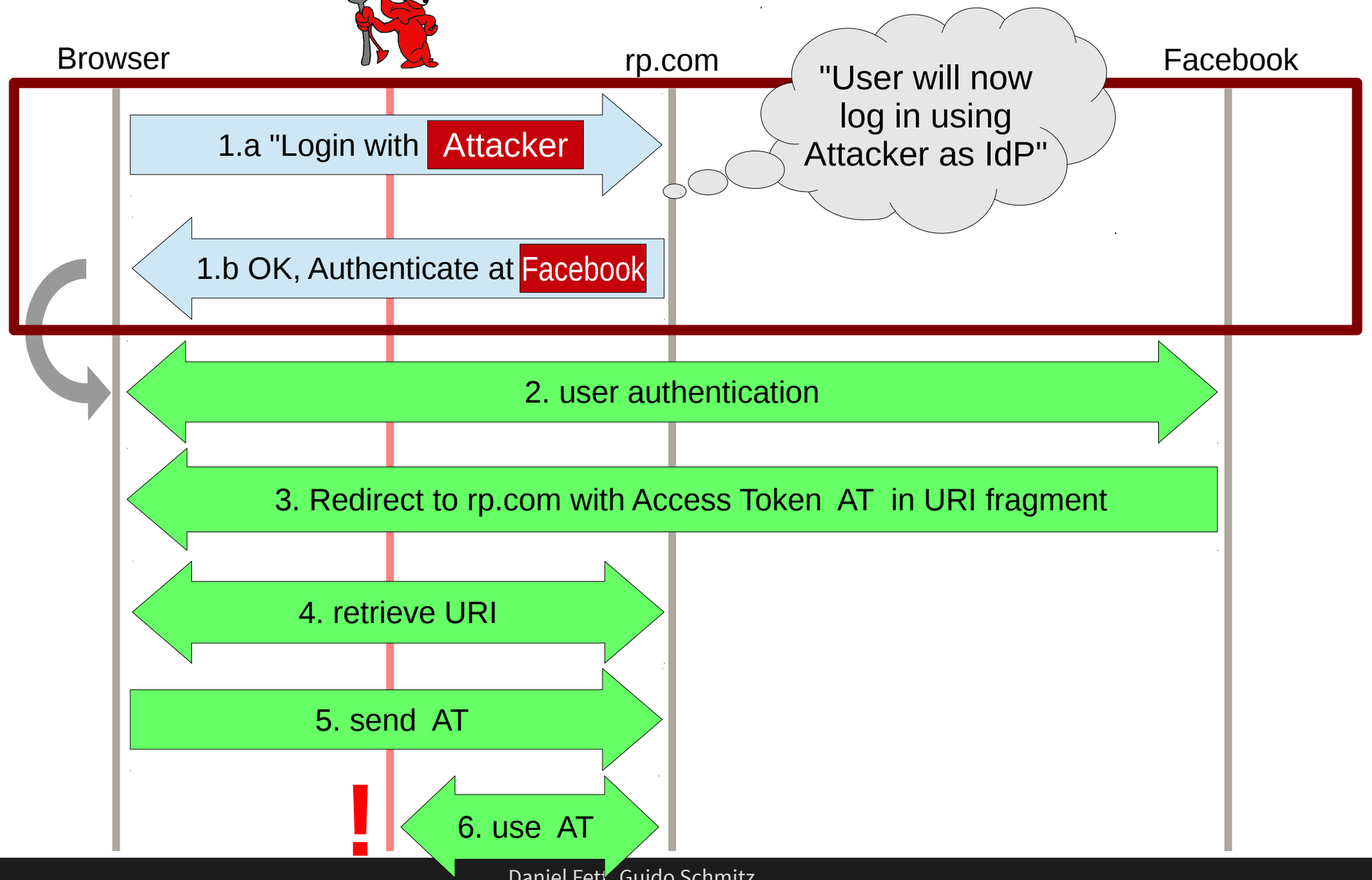
IdP Mix-Up Attack in Implicit Mode



IdP Mix-Up Attack in Implicit Mode



IdP Mix-Up Attack in Implicit Mode



IdP Mix-Up Attack

+ Variants

More details have to be taken care of:

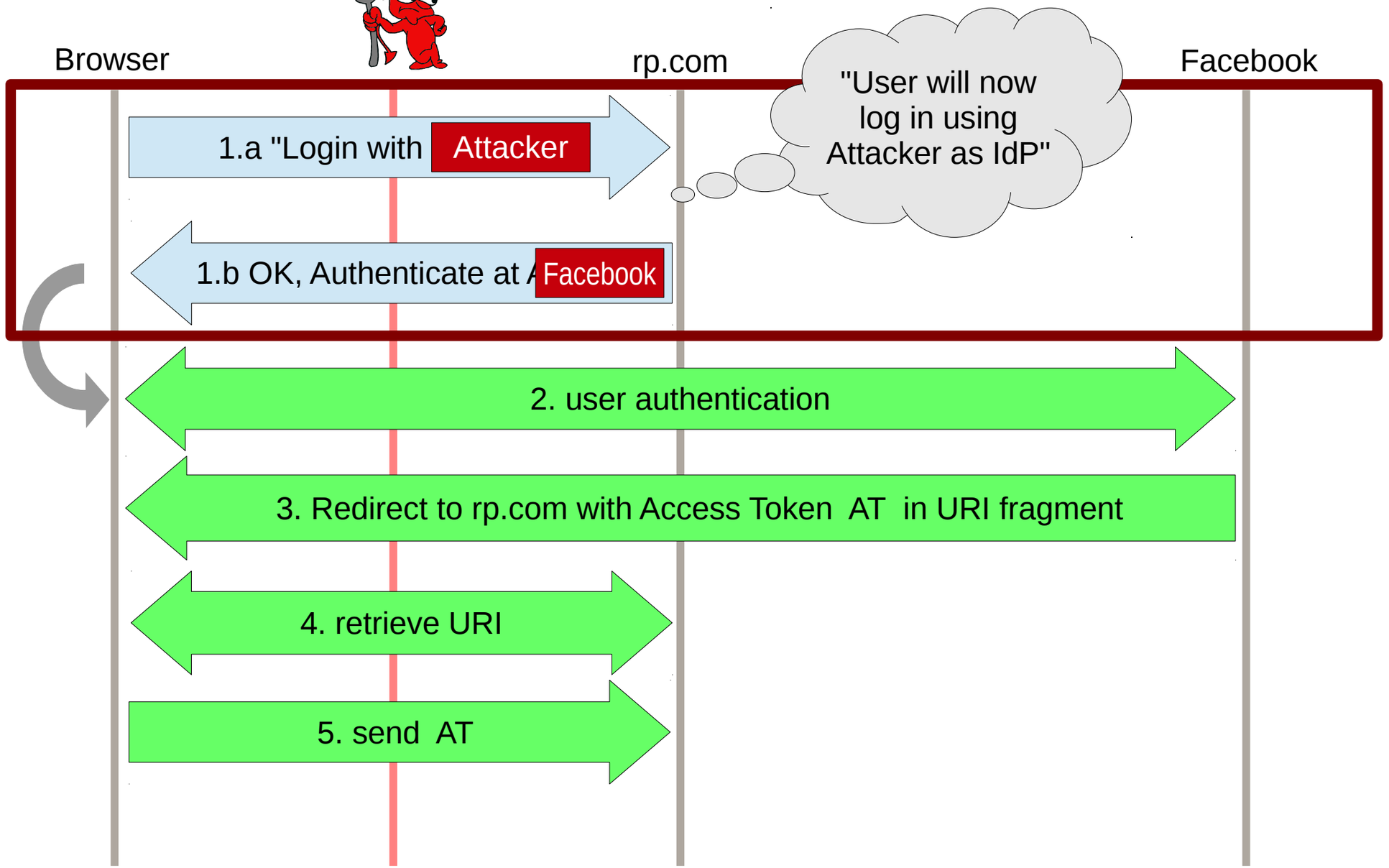
- Breaking authentication instead of authorization (some additional steps)
- Client Identifiers (i.e., RPs identify themselves to IdPs)
- Client Credentials (i.e., RPs authenticate to IdPs)
- In OpenID Connect: Message signing, "ID Token", endpoint discovery, etc.

→ **Sucessfully attacked real-world implementation.**

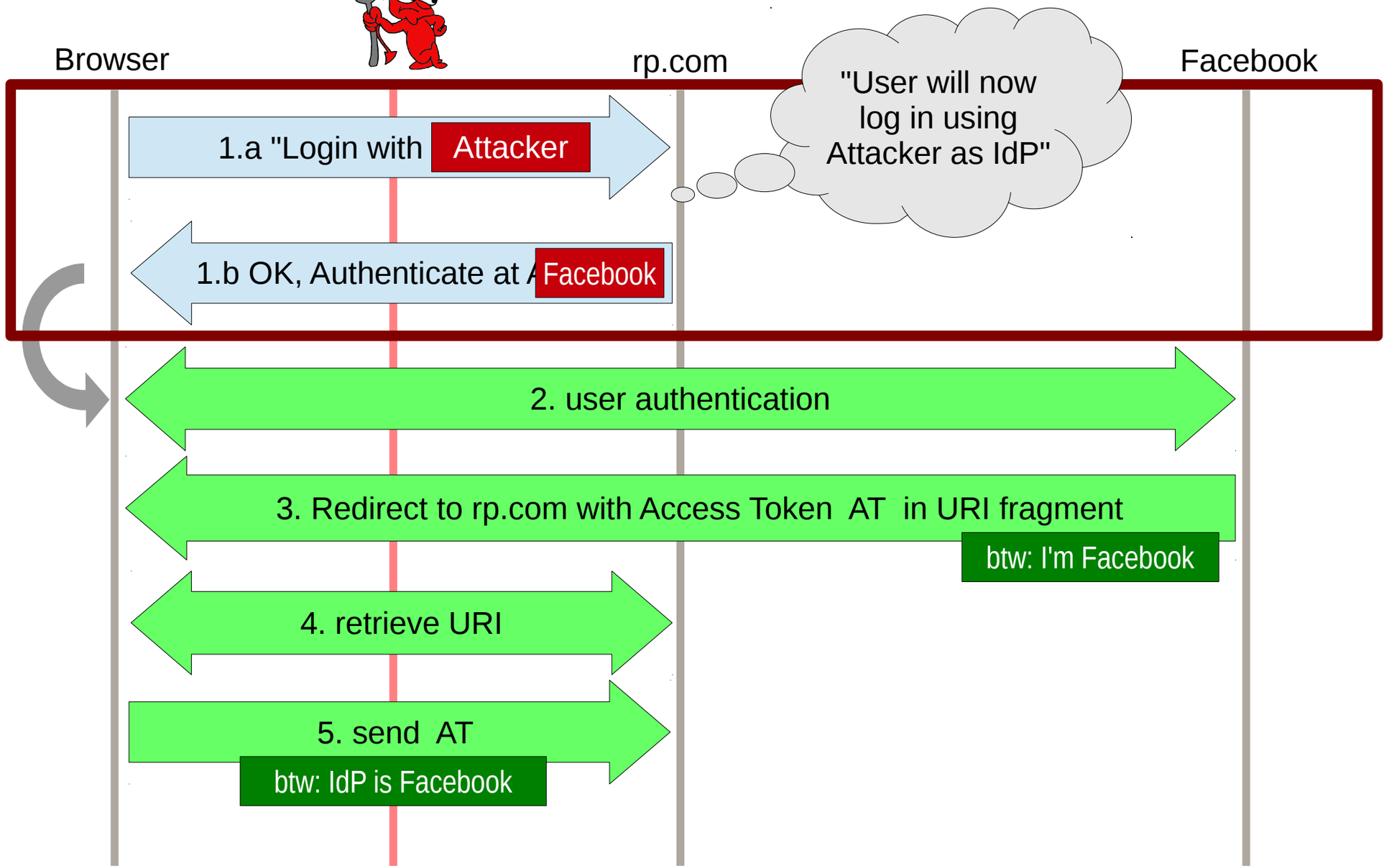
Multiple/malicious IdPs.
Related problems:

[Bansal et al. 2012]
[Mladenov et al. 2015]

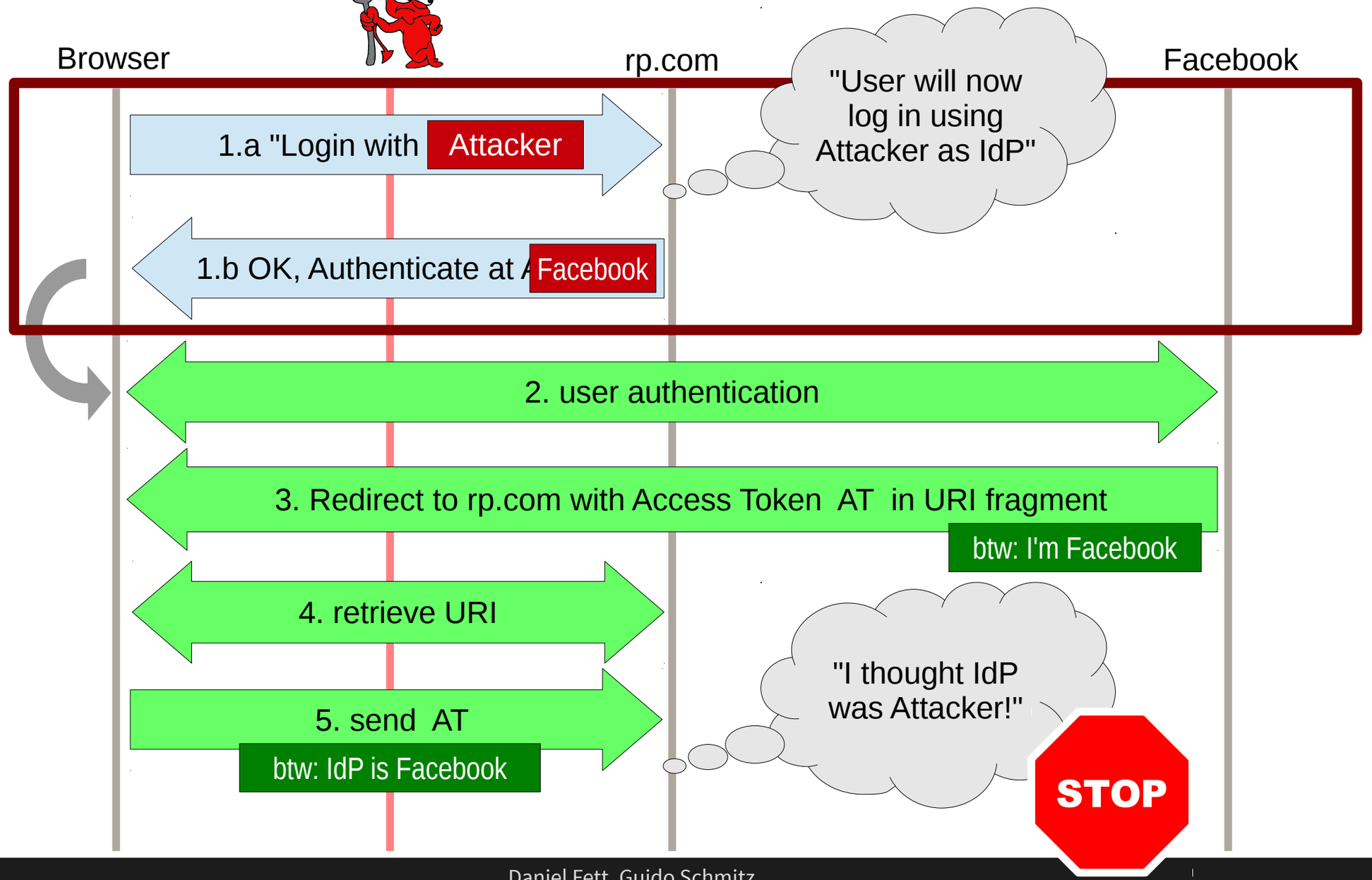
IdP Mix-Up Attack: Mitigation



IdP Mix-Up Attack: Mitigation



IdP Mix-Up Attack: Mitigation



Impact

- Emergency meeting of the IETF OAuth Working Group
- Public disclosure of the attacks
- Triggered OAuth Security Workshop
- New RFC in preparation
- Interest in formal analysis

OAuth 2.0

Security	**attacks + fix** **formally proven** 
Privacy	**NONE**

First Attempt with Privacy Feature:
BrowserID a.k.a. Mozilla Persona

BrowserID (a.k.a. Mozilla Persona)

- Web-based Single Sign-On system

- Design goals:
  - No central authority
  - Better privacy

BrowserID Overview

alice@mailprovider.com

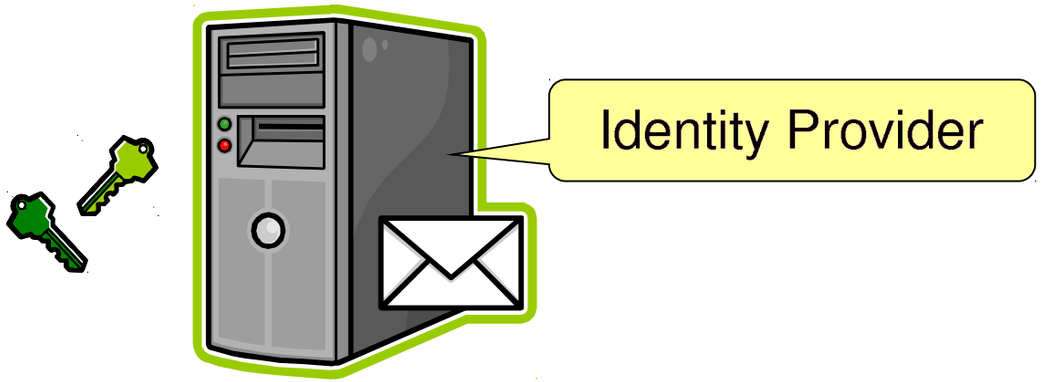
Phase 1: Provisioning

BrowserID Overview

alice@mailprovider.com

BrowserID Overview

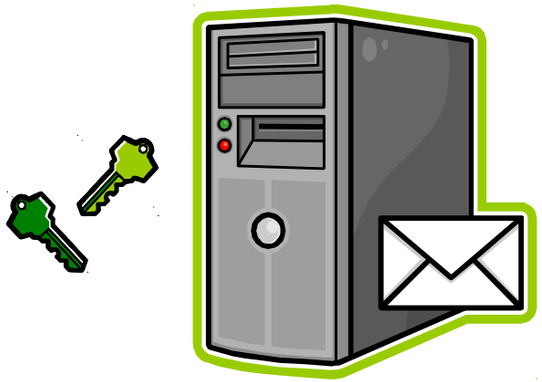
alice@mailprovider.com



mailprovider.com

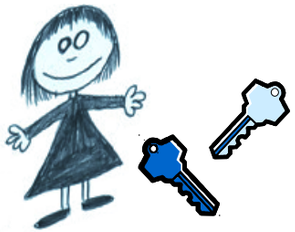
BrowserID Overview

alice@mailprovider.com

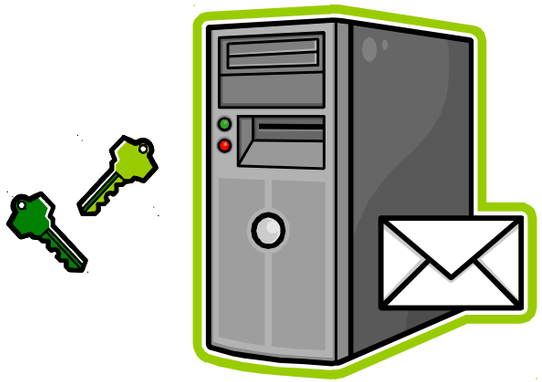


mailprovider.com

BrowserID Overview

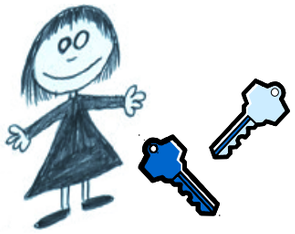


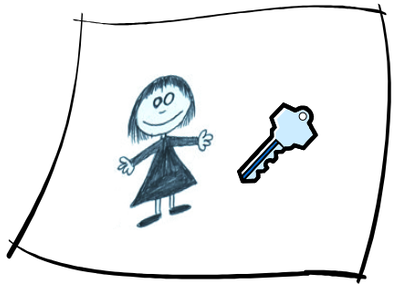
alice@mailprovider.com



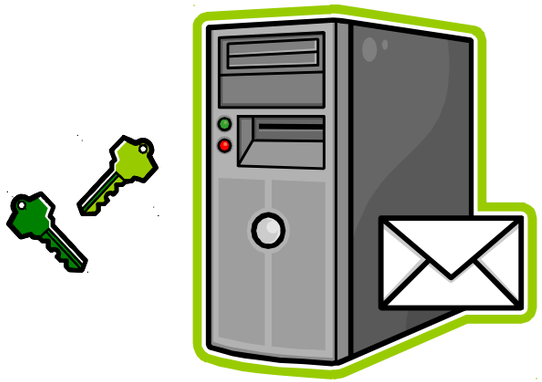
mailprovider.com

BrowserID Overview



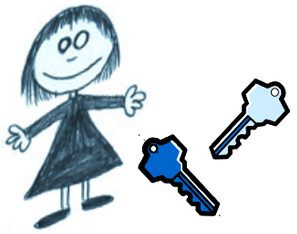


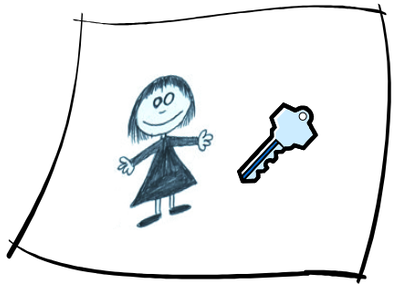
alice@mailprovider.com



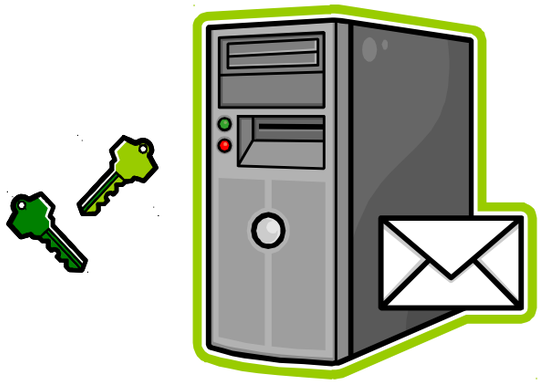
mailprovider.com

BrowserID Overview



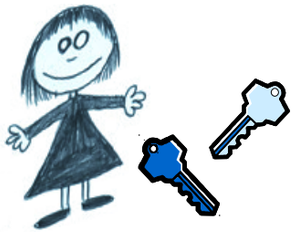


alice@mailprovider.com

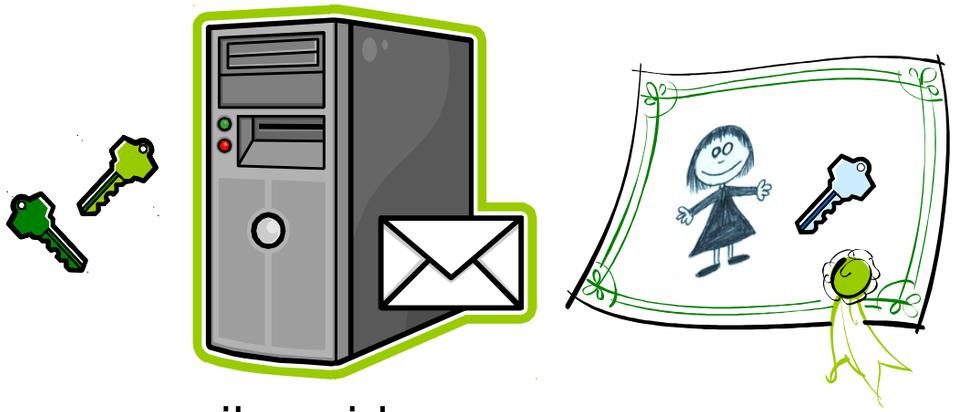


mailprovider.com

BrowserID Overview

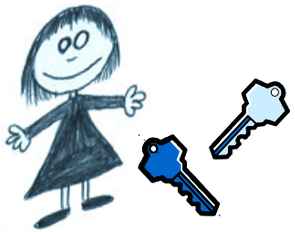


alice@mailprovider.com

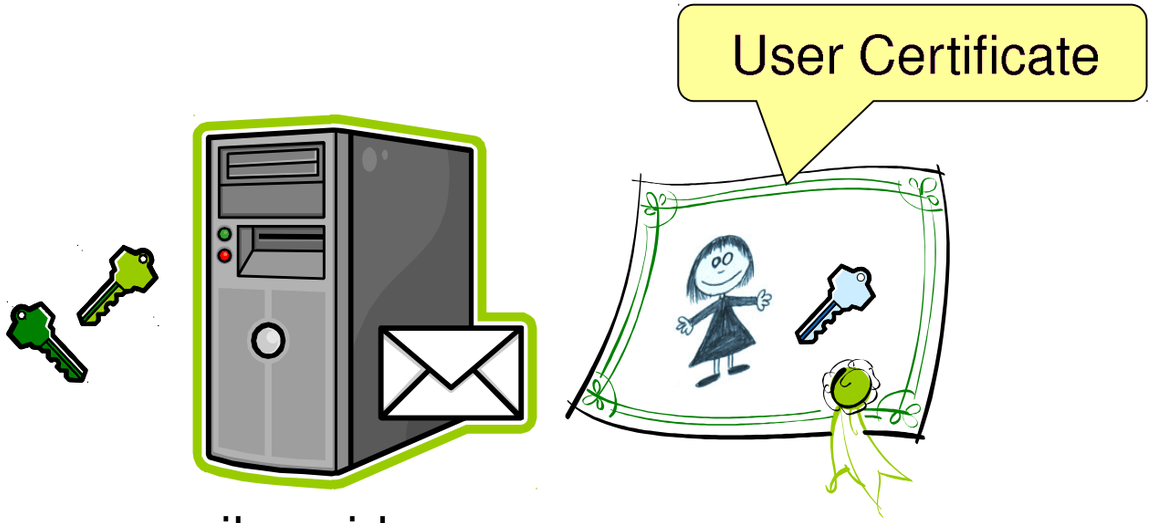


mailprovider.com

BrowserID Overview

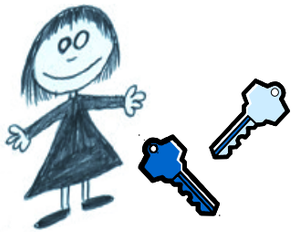


alice@mailprovider.com

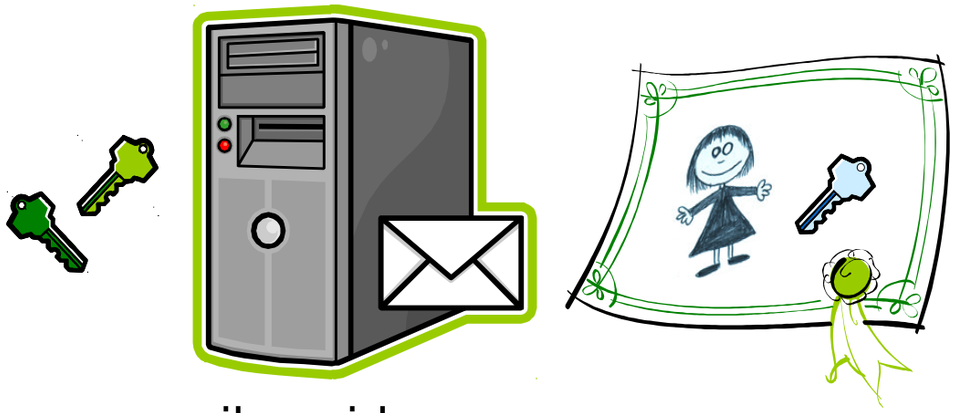


mailprovider.com

BrowserID Overview

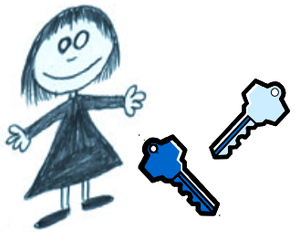


alice@mailprovider.com

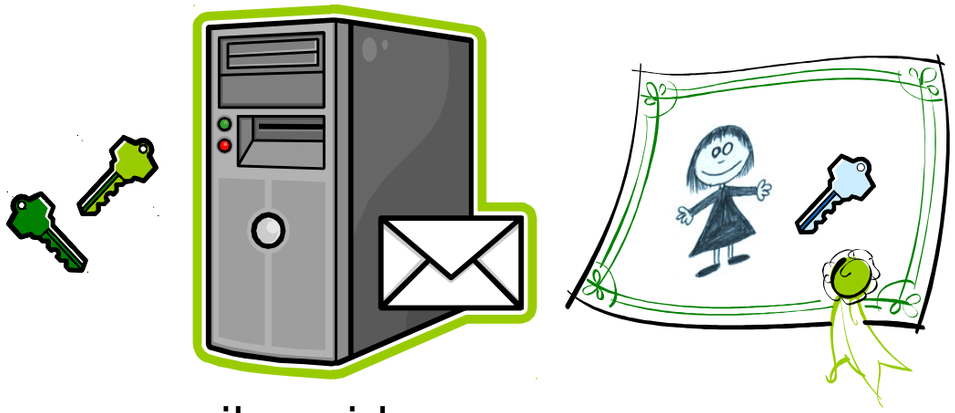


mailprovider.com

BrowserID Overview

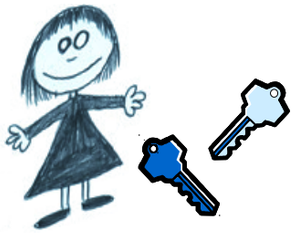


alice@mailprovider.com

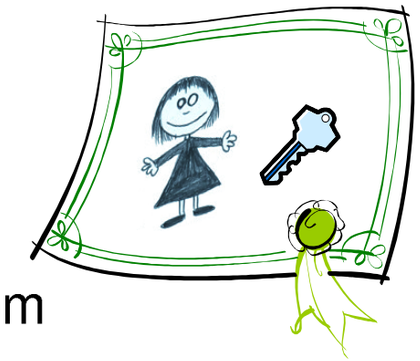


mailprovider.com

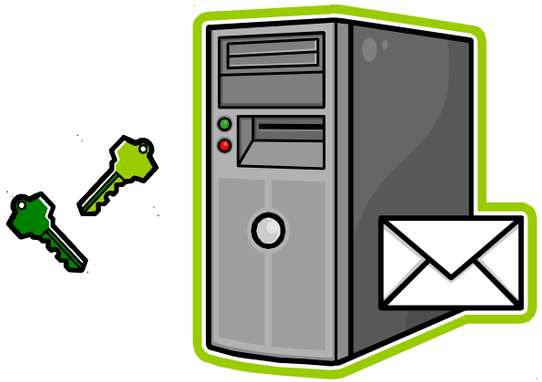
BrowserID Overview



alice@mailprovider.com

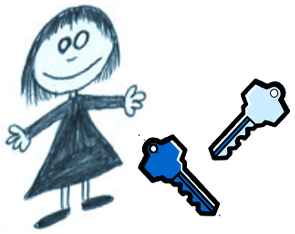


Phase 2: Authentication

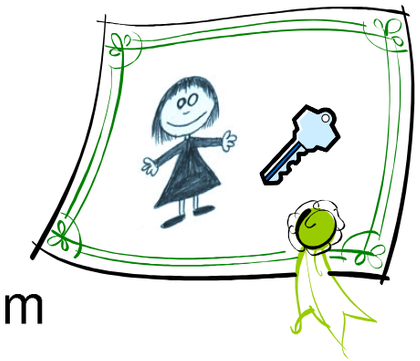


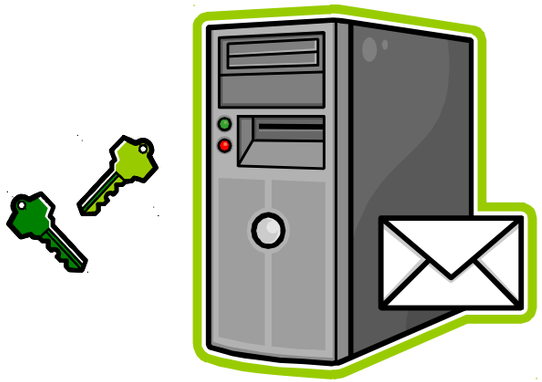
mailprovider.com

BrowserID Overview



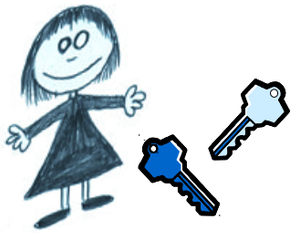
alice@mailprovider.com



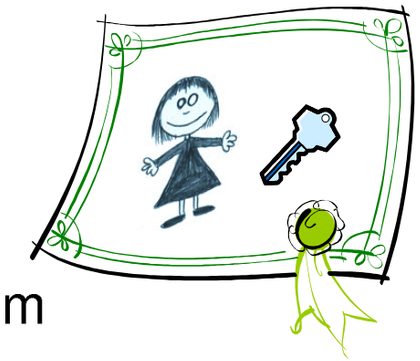


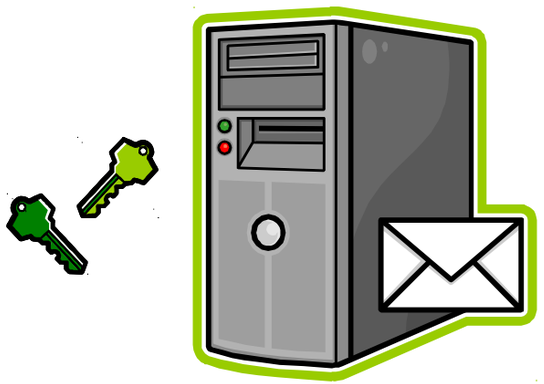
mailprovider.com

BrowserID Overview



alice@mailprovider.com



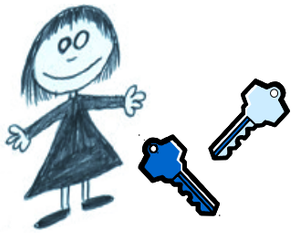


mailprovider.com

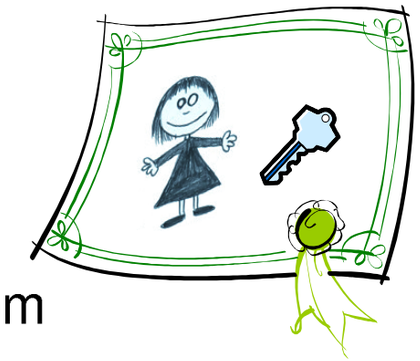
Relying Party

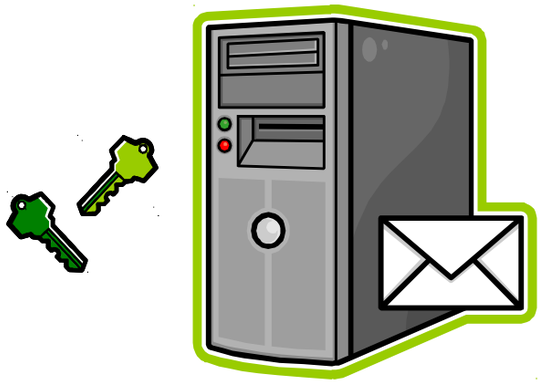
wikipedia.org

BrowserID Overview



alice@mailprovider.com

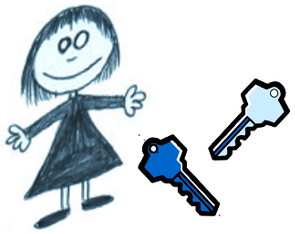




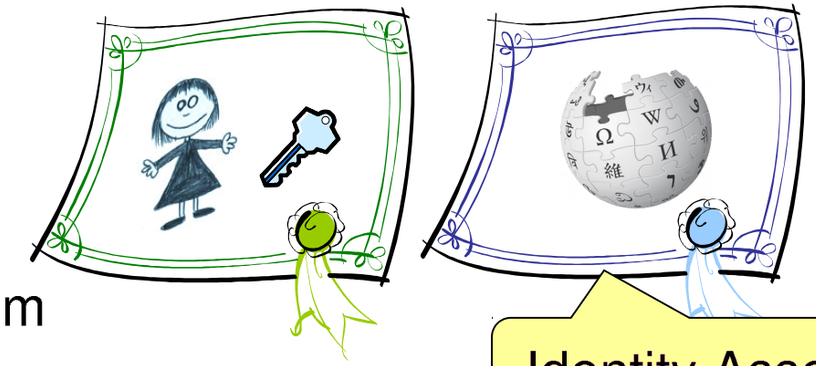
mailprovider.com

wikipedia.org

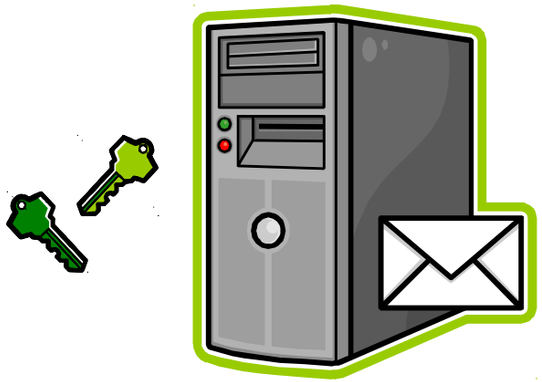
BrowserID Overview



alice@mailprovider.com



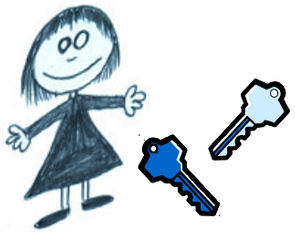
Identity Assertion



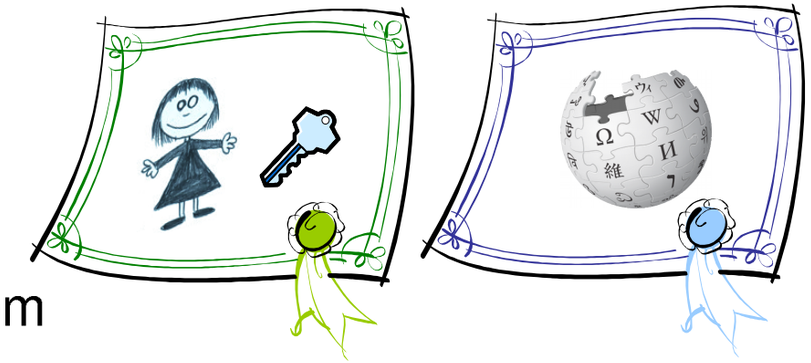
mailprovider.com

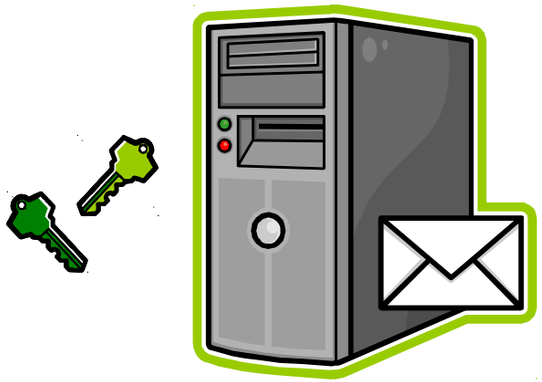
wikipedia.org

BrowserID Overview



alice@mailprovider.com

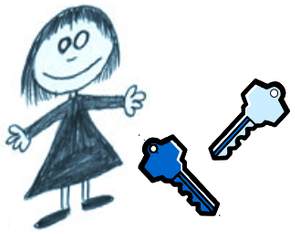




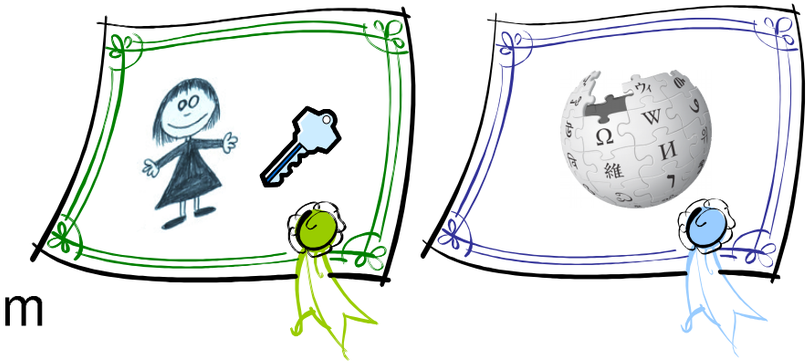
mailprovider.com

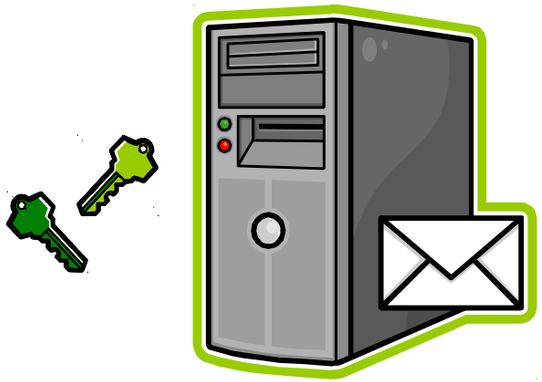
wikipedia.org

BrowserID Overview



alice@mailprovider.com

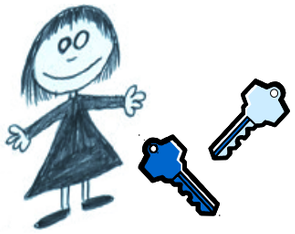




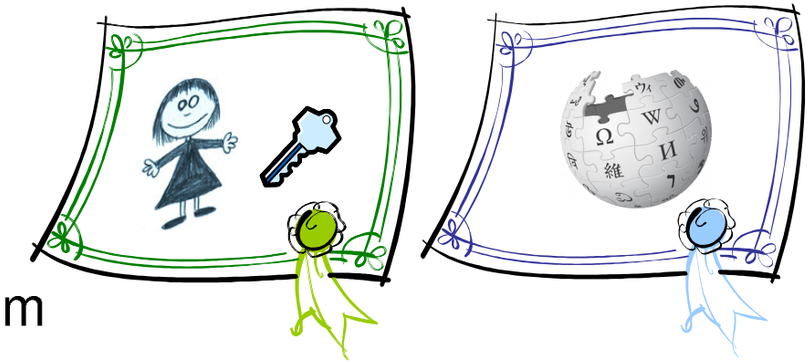
mailprovider.com

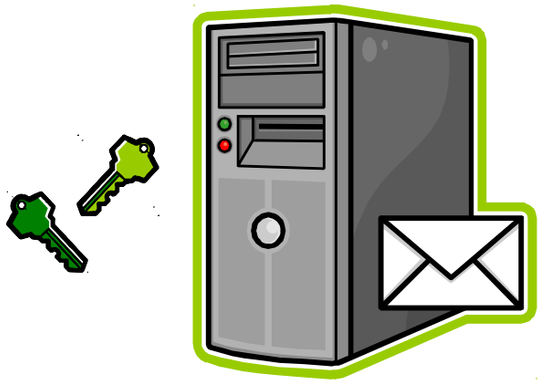
wikipedia.org

BrowserID Overview



alice@mailprovider.com

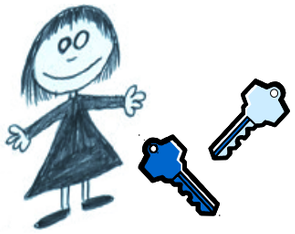




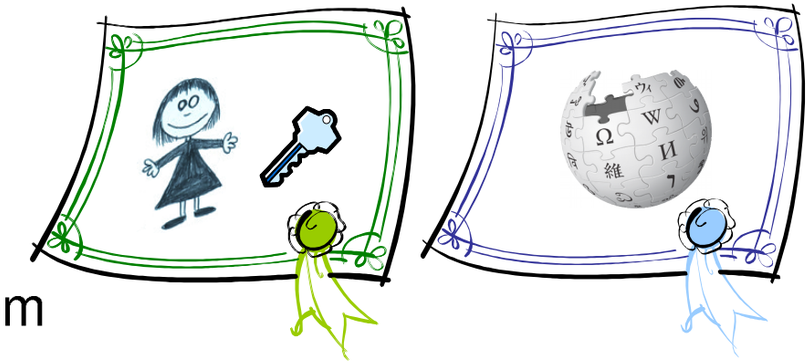
mailprovider.com

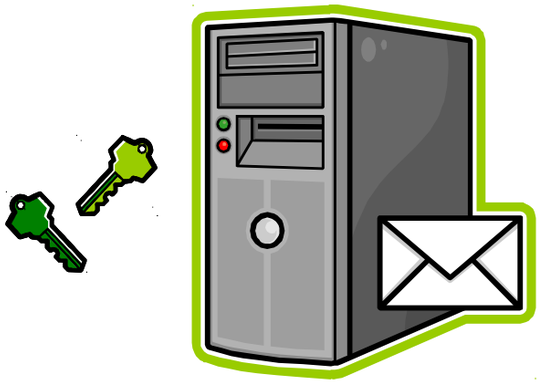
wikipedia.org

BrowserID Overview



alice@mailprovider.com

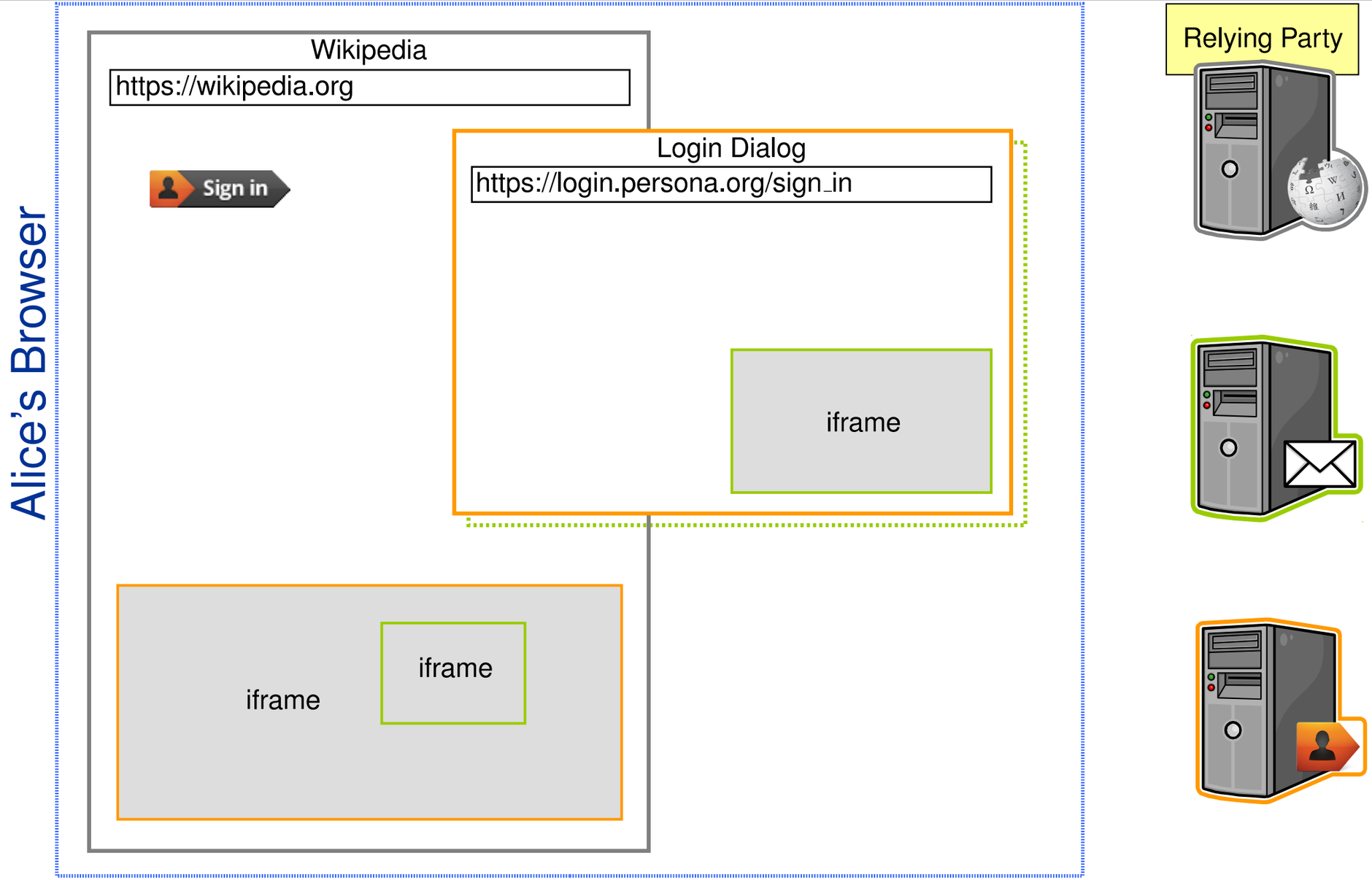




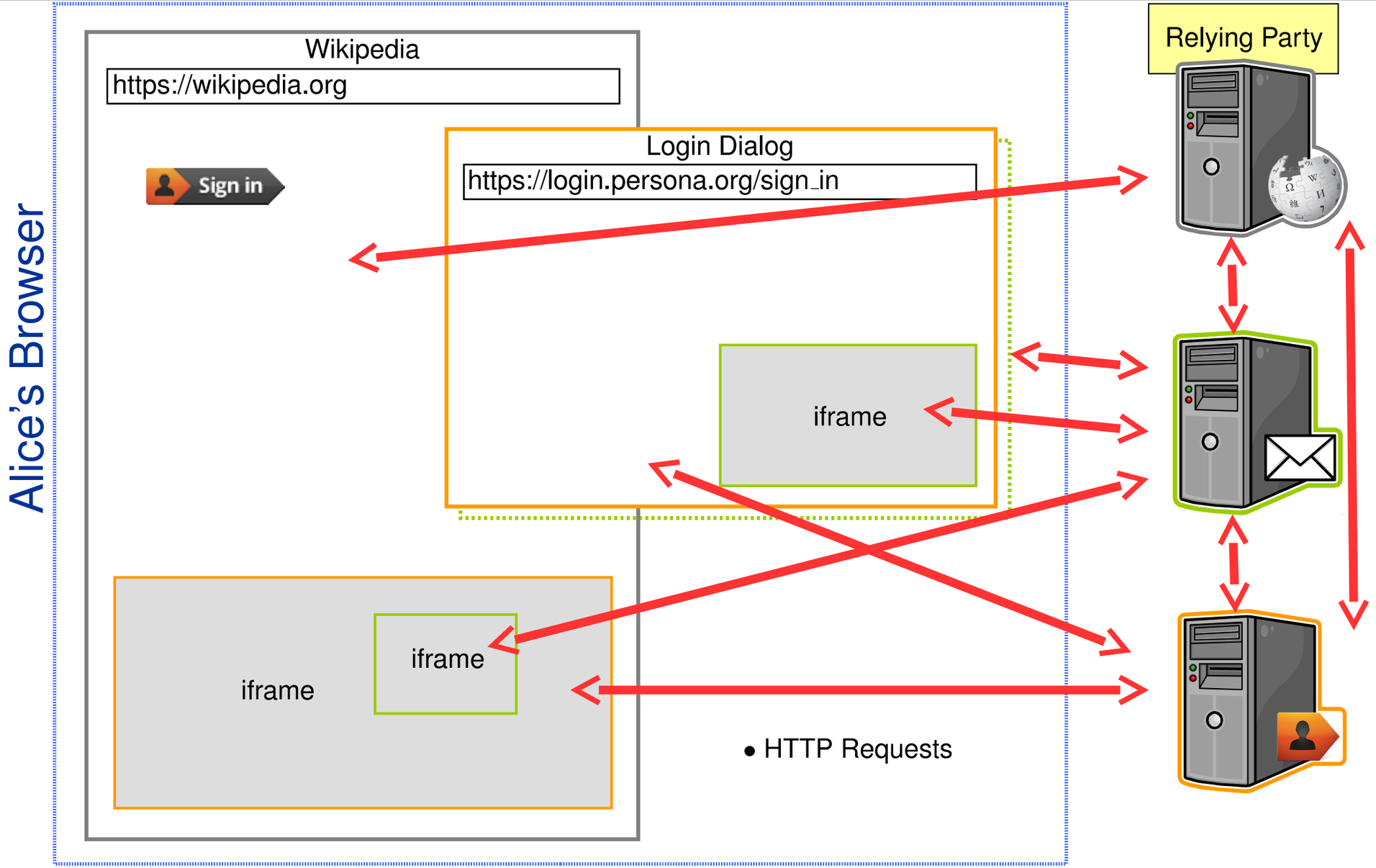
mailprovider.com

wikipedia.org

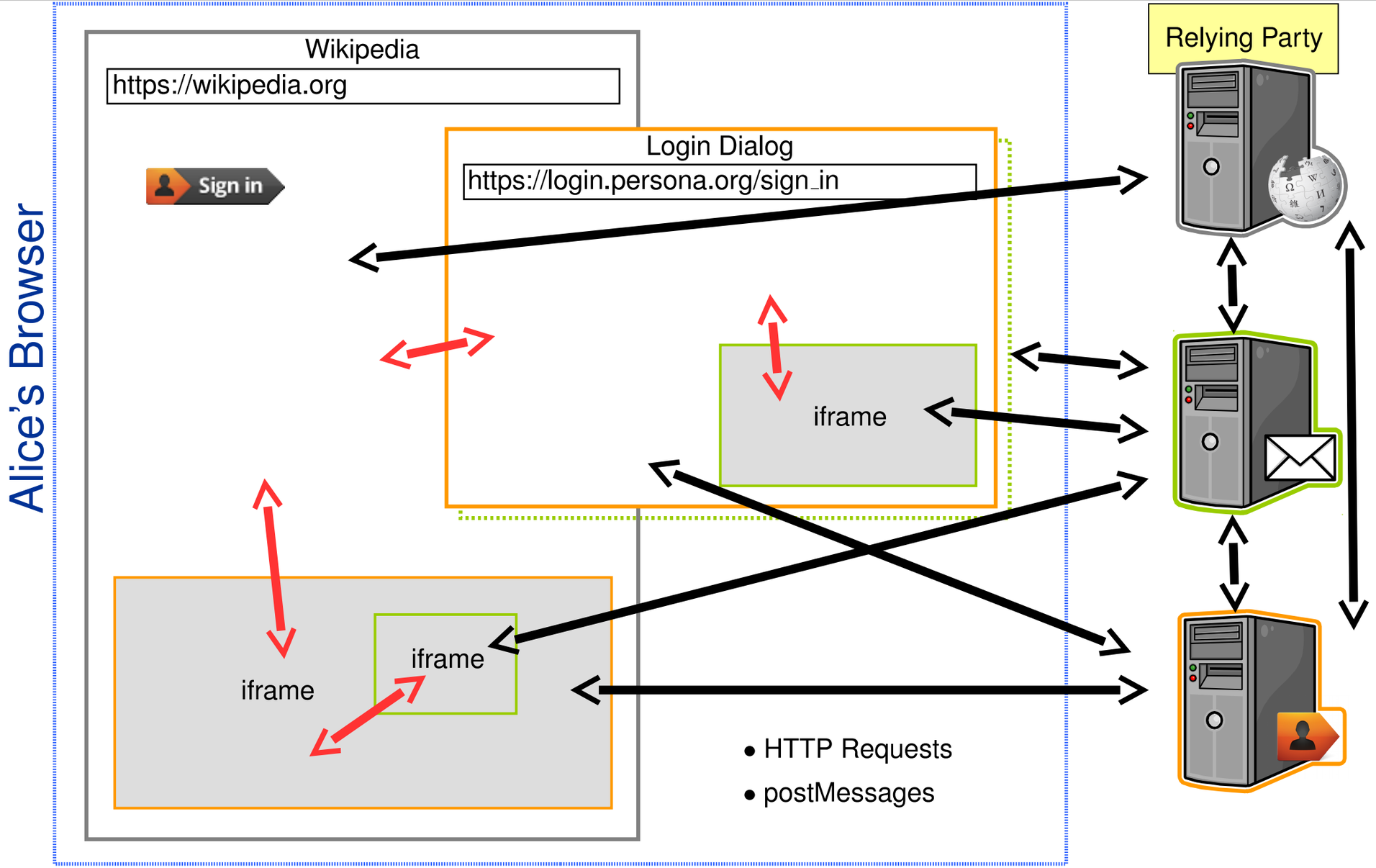
BrowserID Implementation



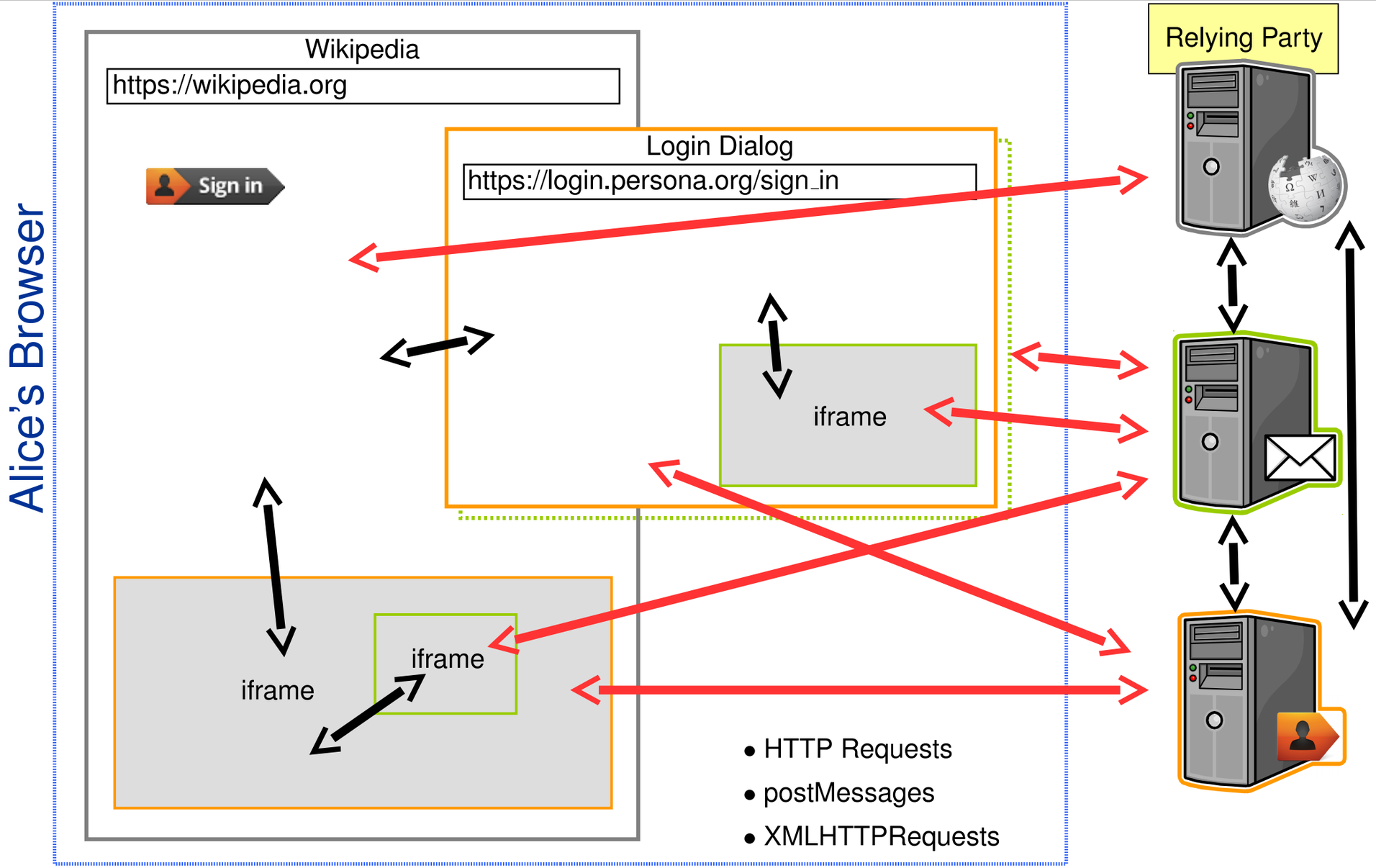
BrowserID Implementation



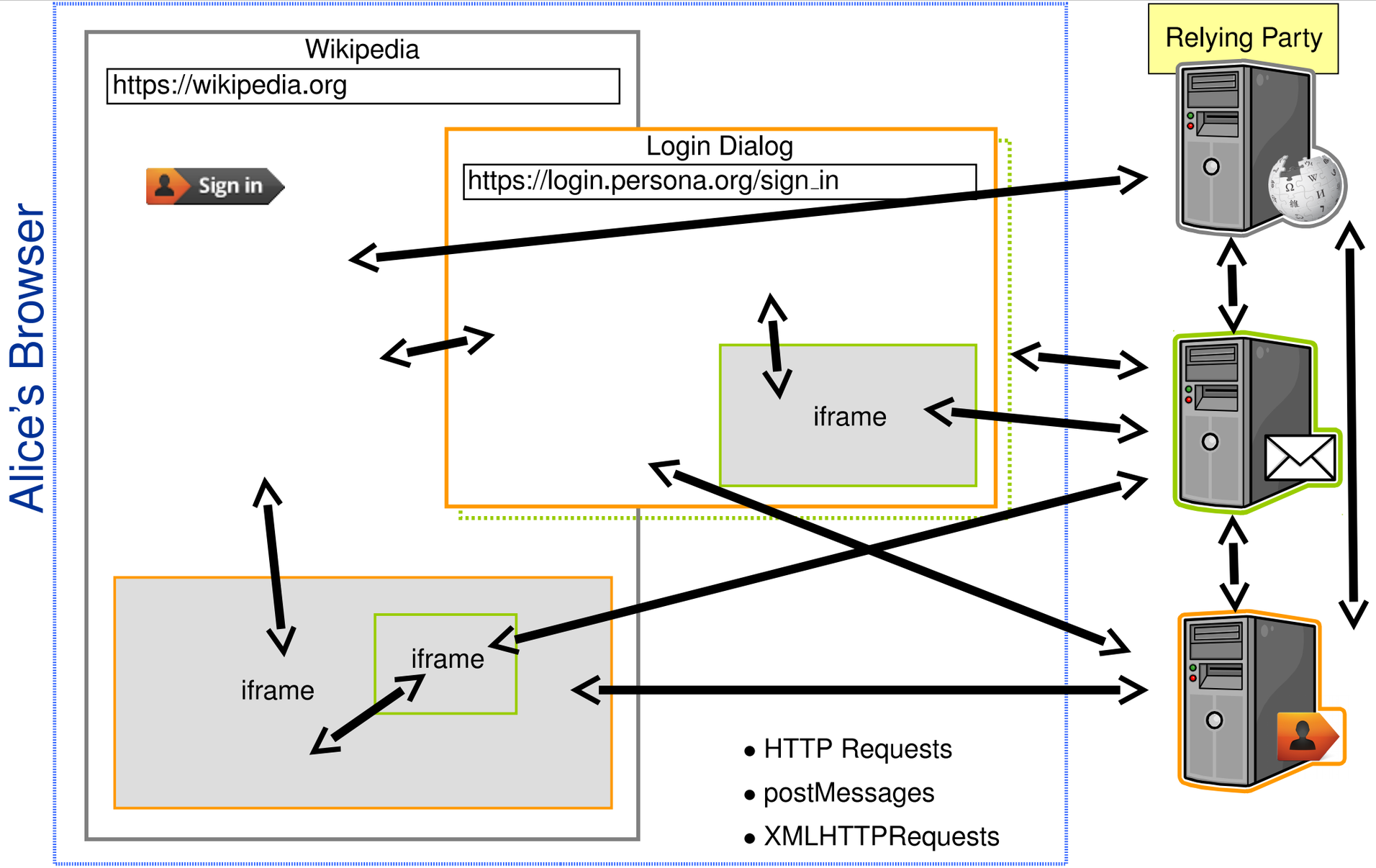
BrowserID Implementation



BrowserID Implementation

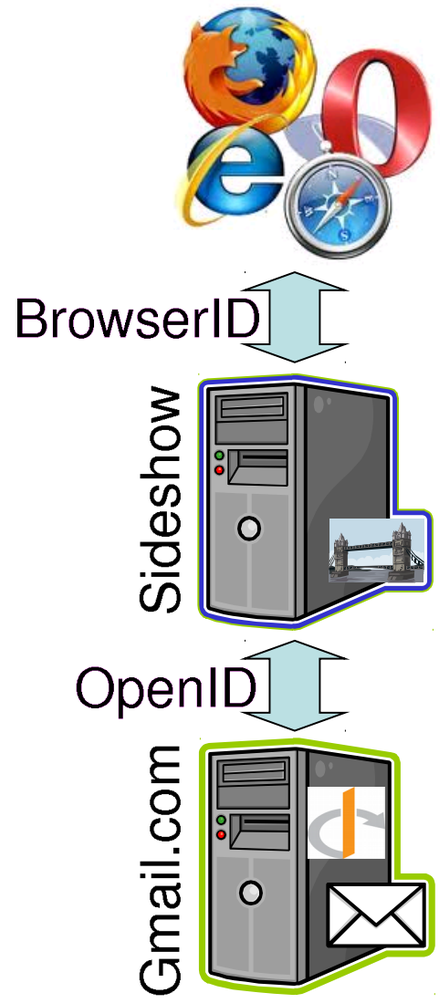


BrowserID Implementation



Identity Bridge

- Gmail.com / Yahoo.com (used to) support OpenID authentication (only)
- New party: Bridging Server
  - Sideshow (for Gmail.com)
  - BigTent (for Yahoo.com)
- User authenticates to bridging server via OpenID
- Bridging server provides BrowserID interface
- Problem: OpenID identities are not email addresses
  Solution: additional email attribute (OpenID AX)



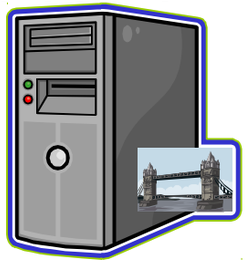
Identity Bridge

Alice's Browser

https://sideshow/...

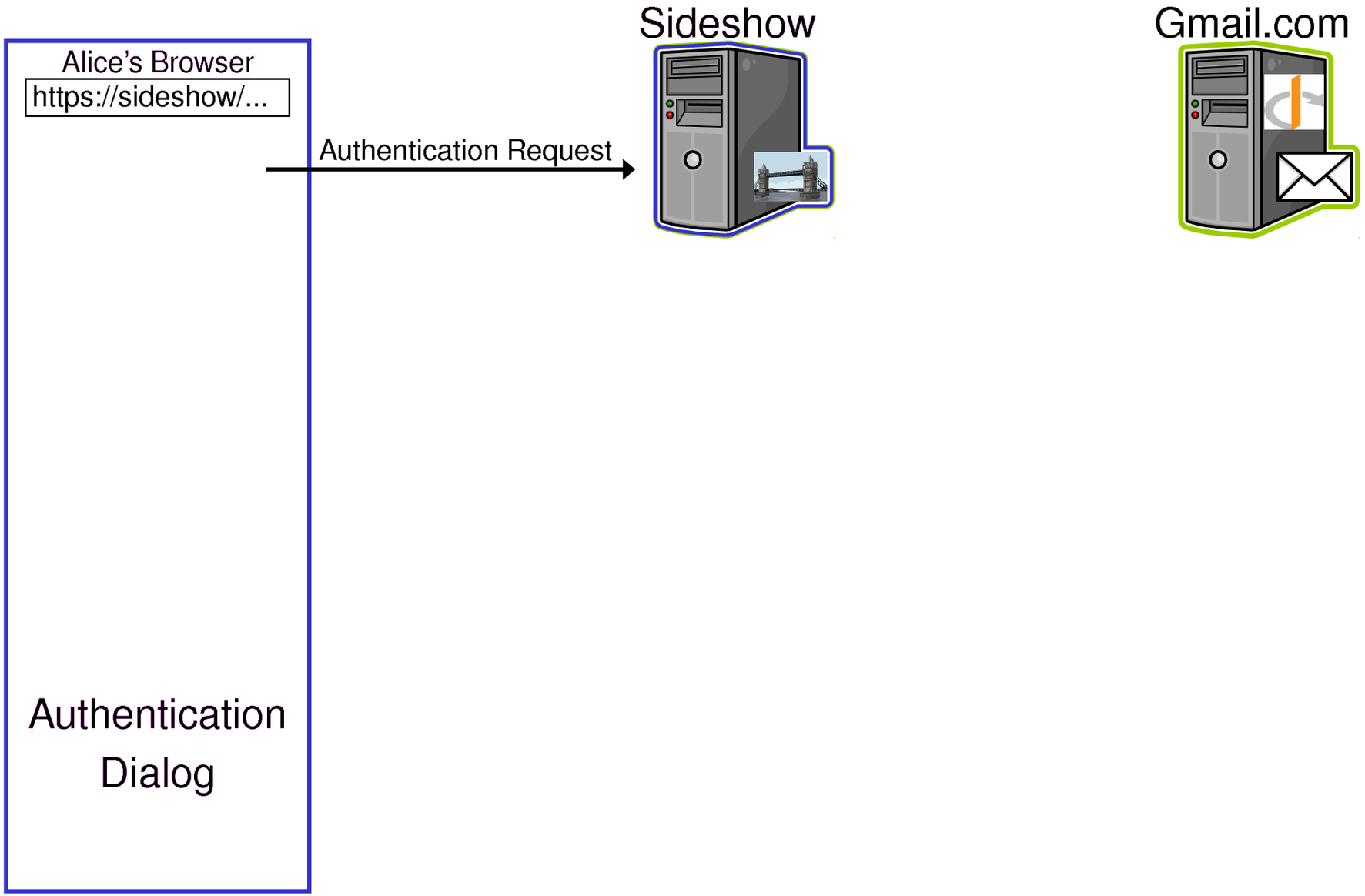
Authentication Dialog

Sideshow

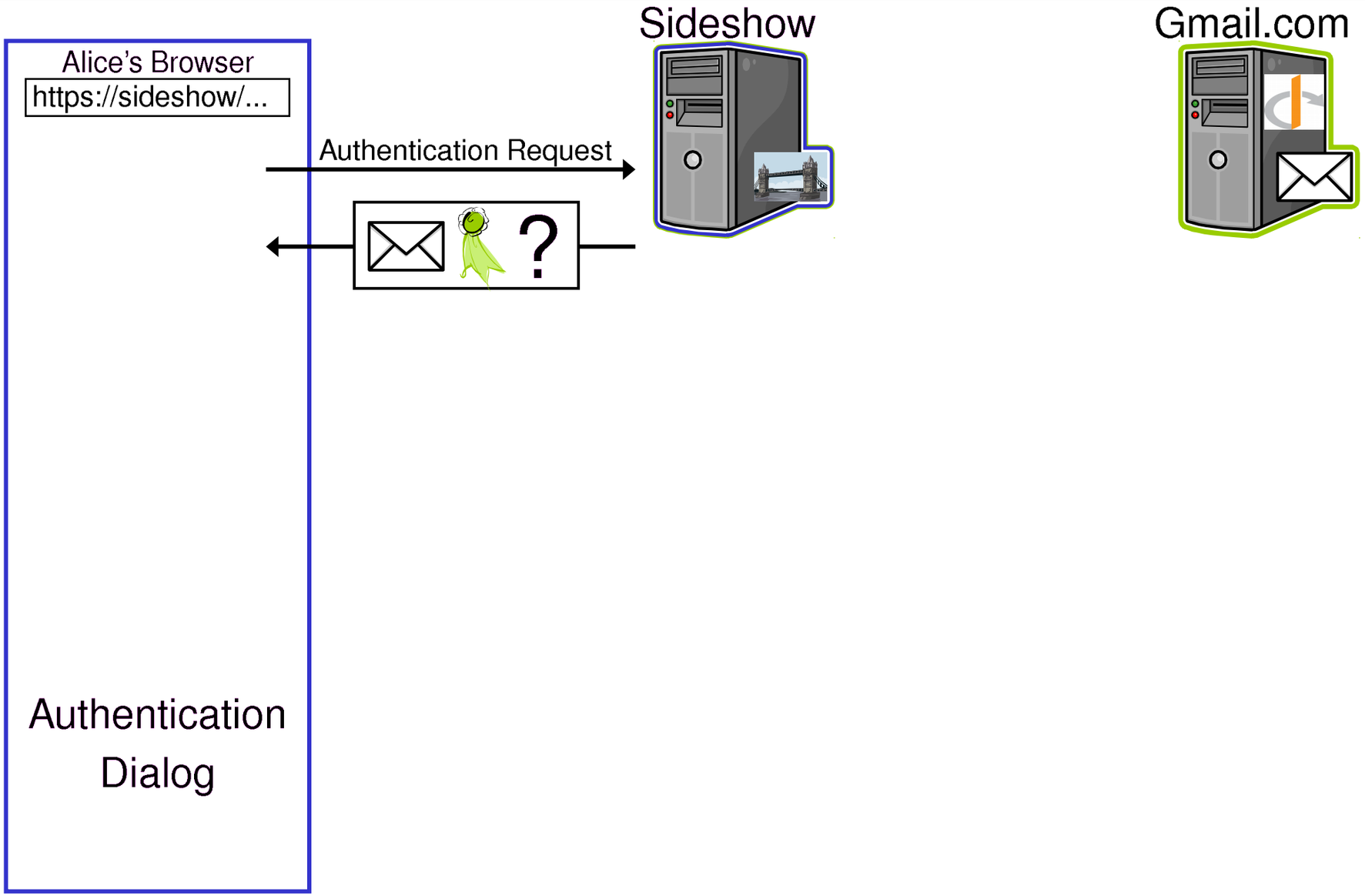


Gmail.com

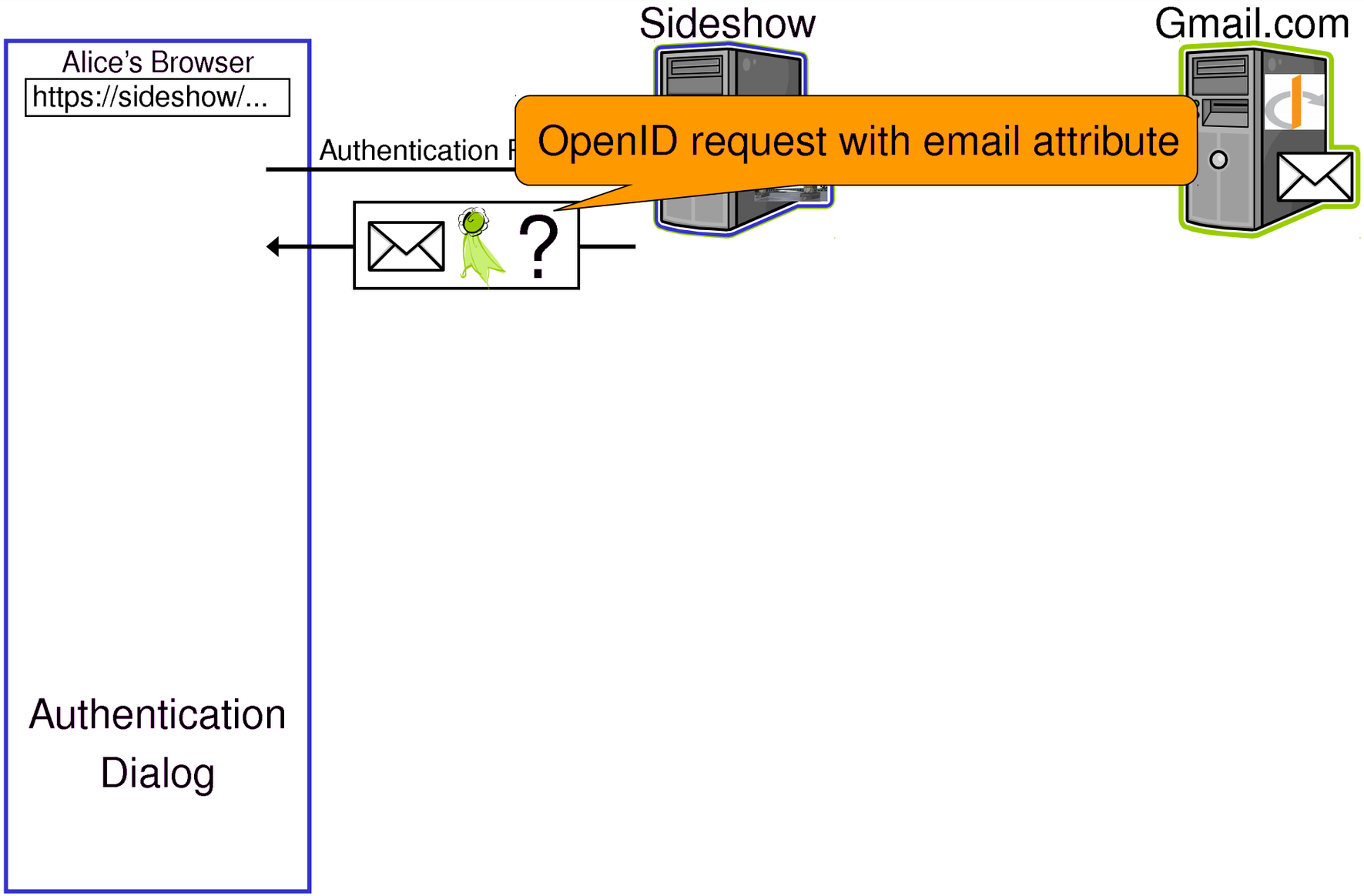
Identity Bridge



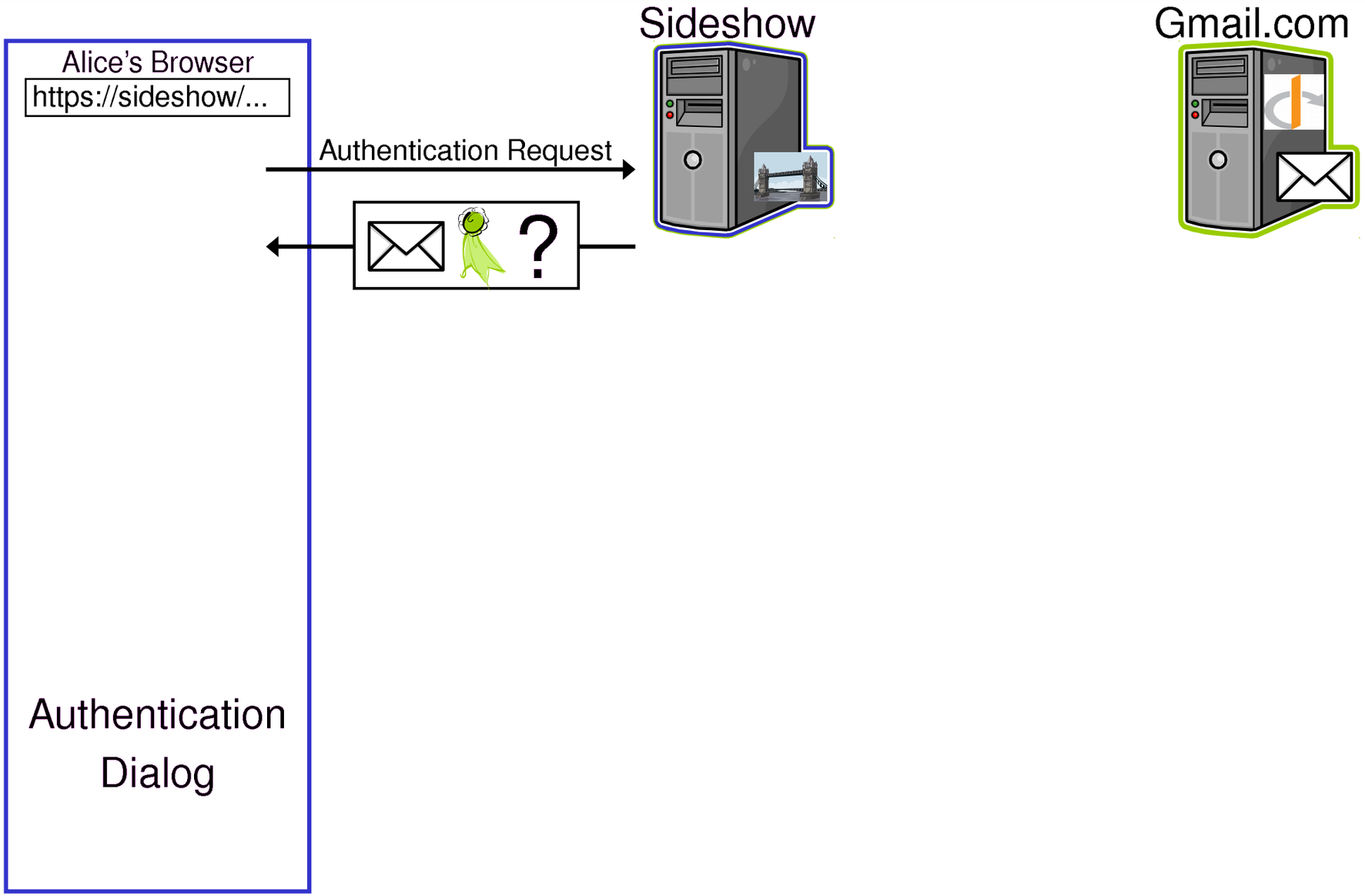
Identity Bridge



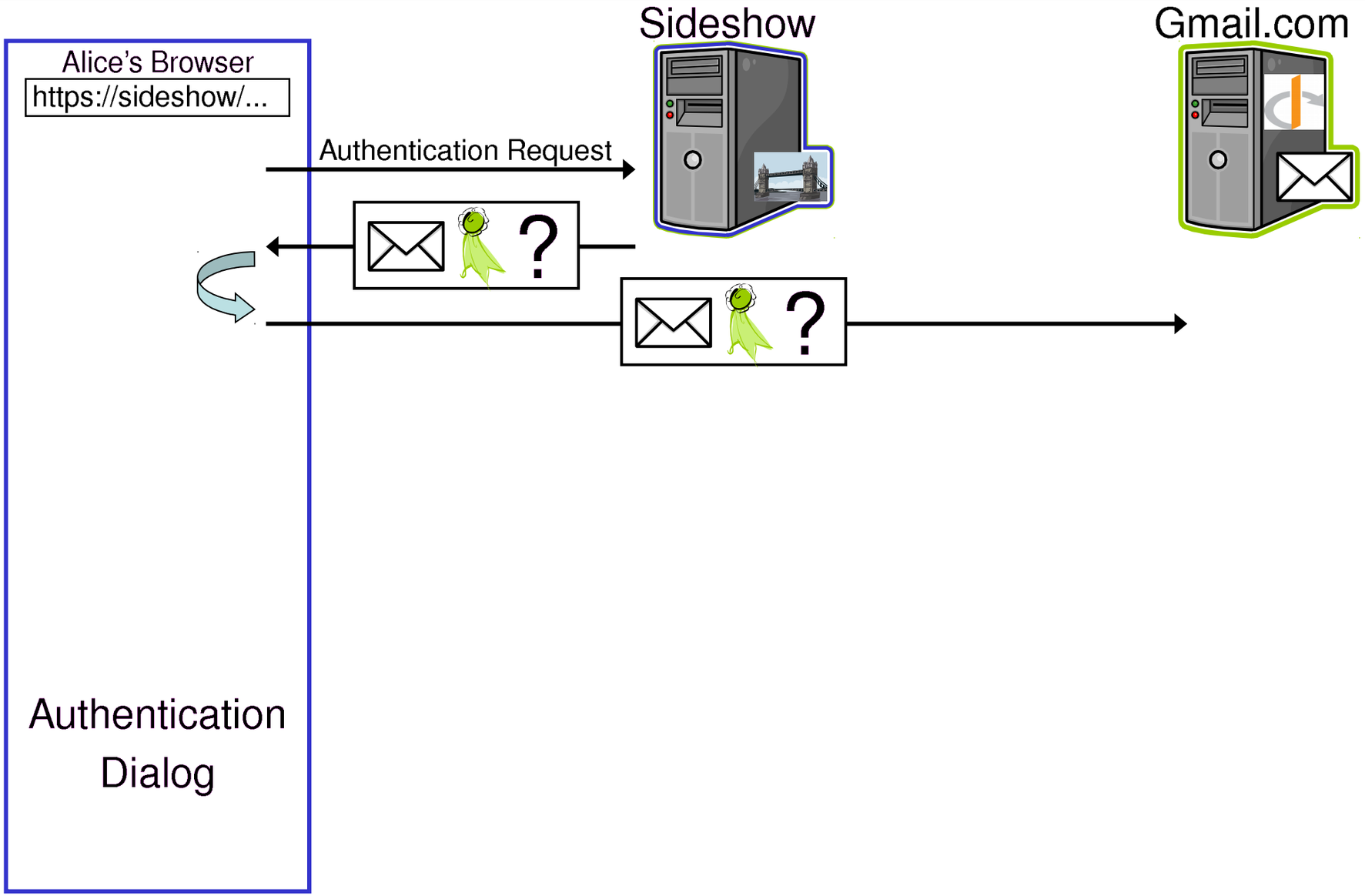
Identity Bridge



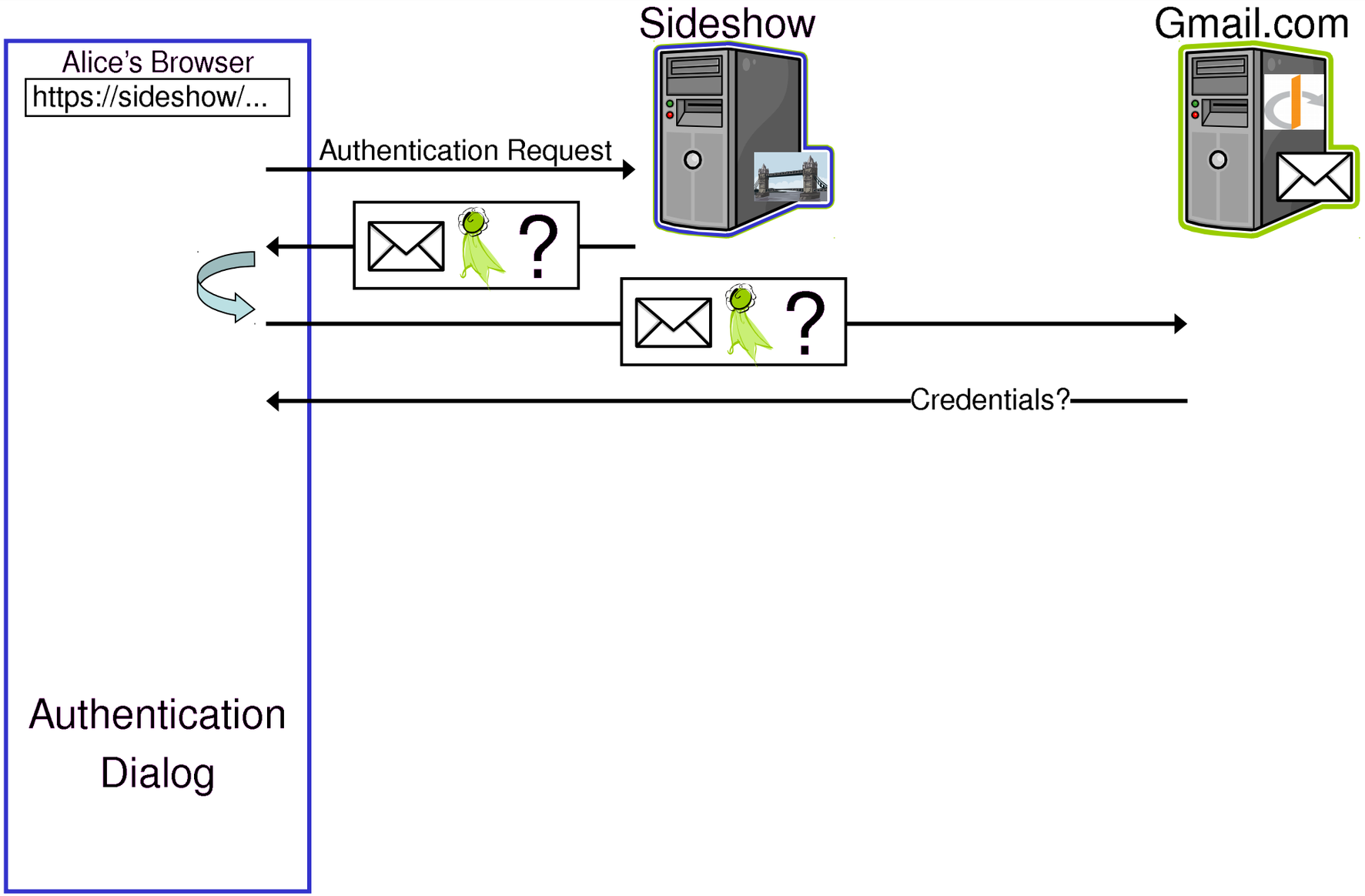
Identity Bridge



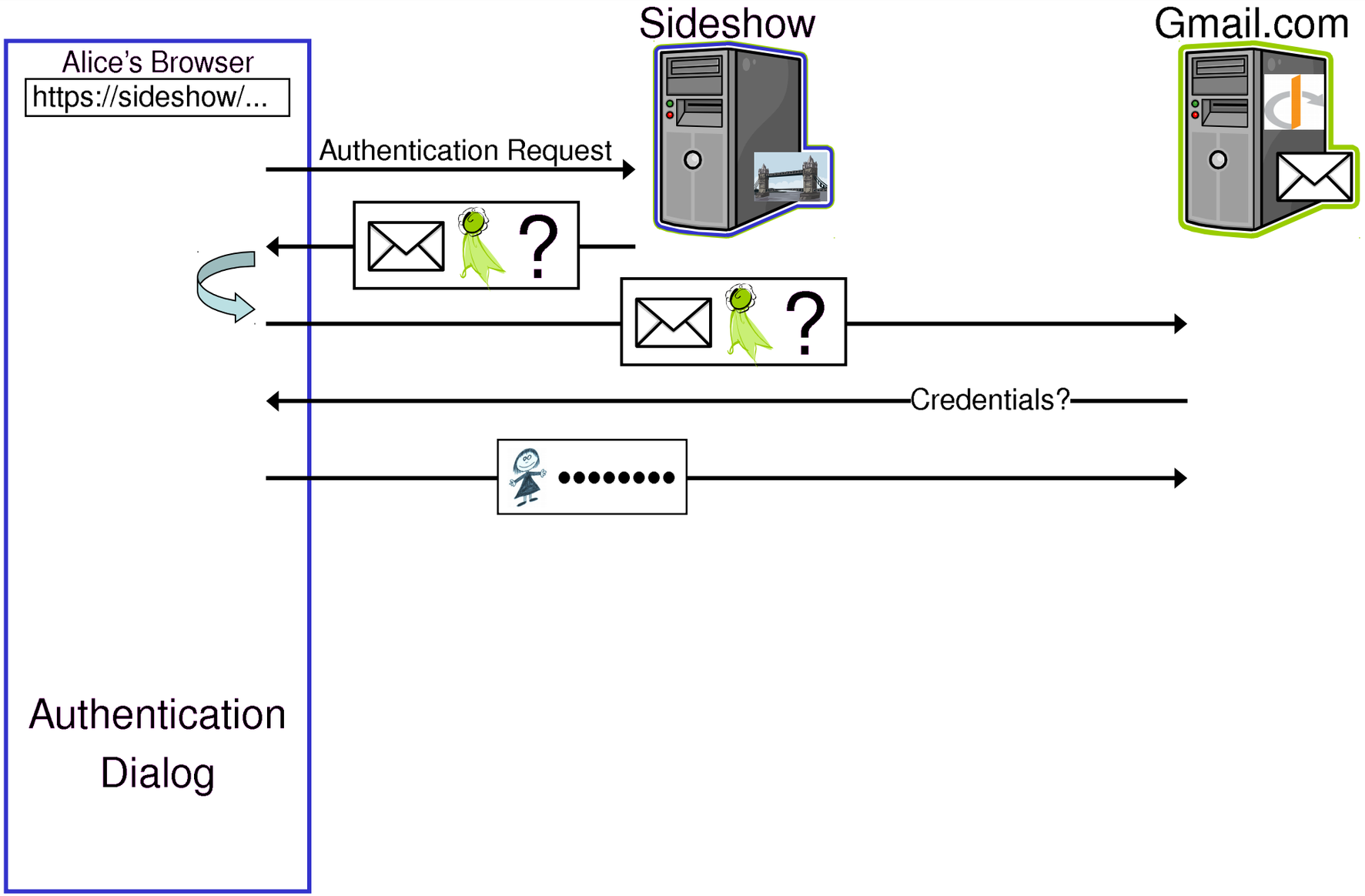
Identity Bridge



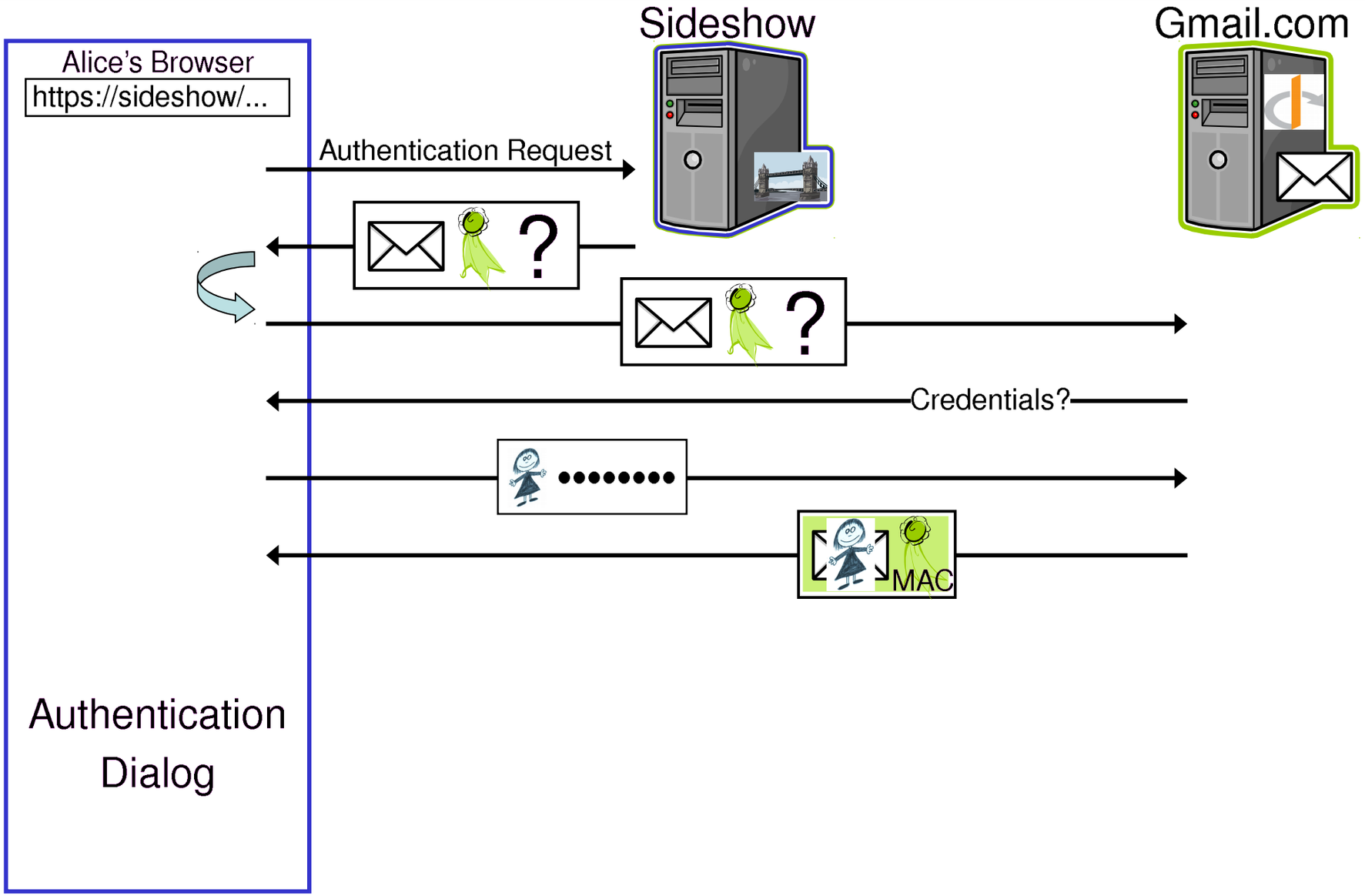
Identity Bridge



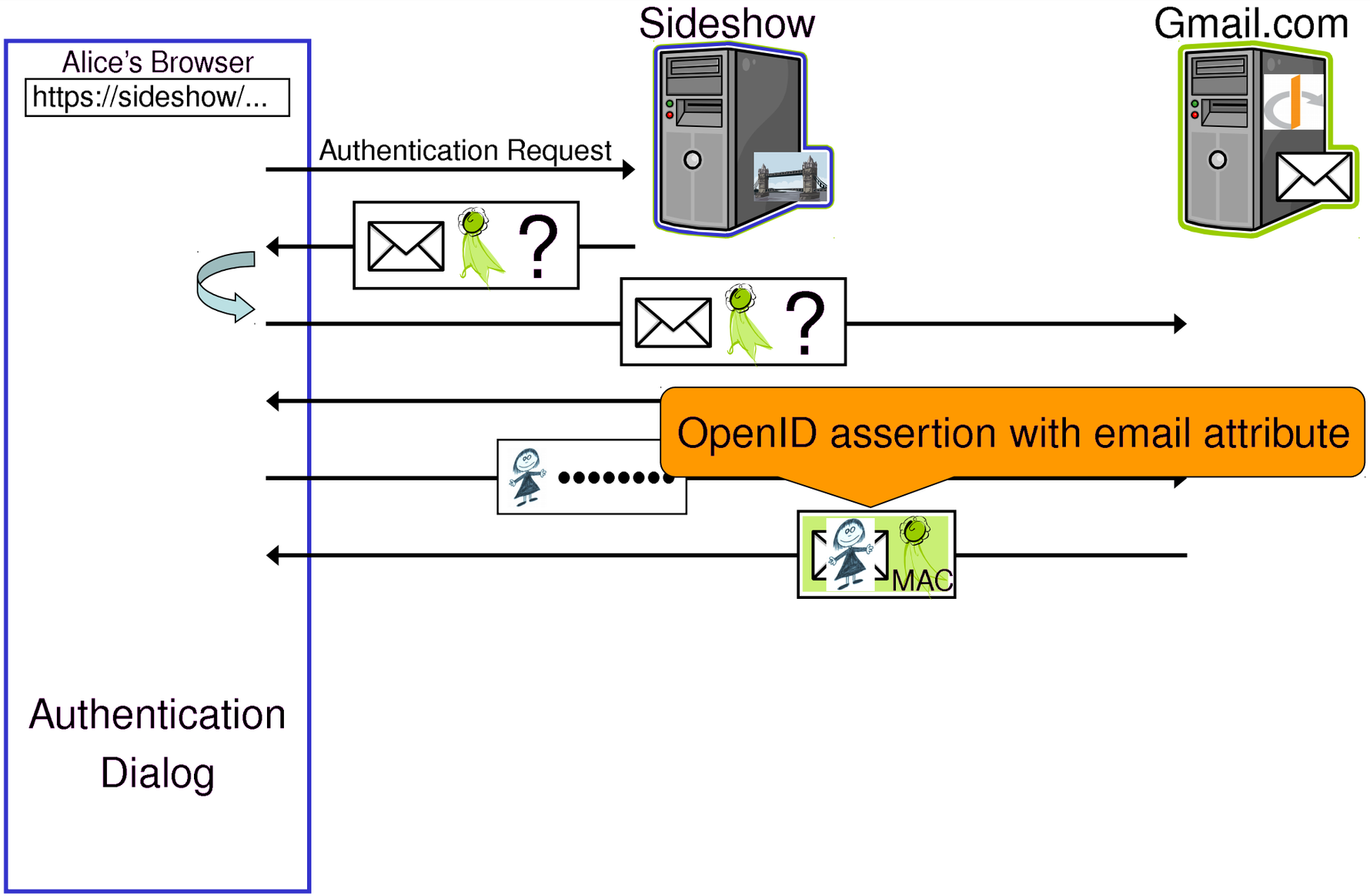
Identity Bridge



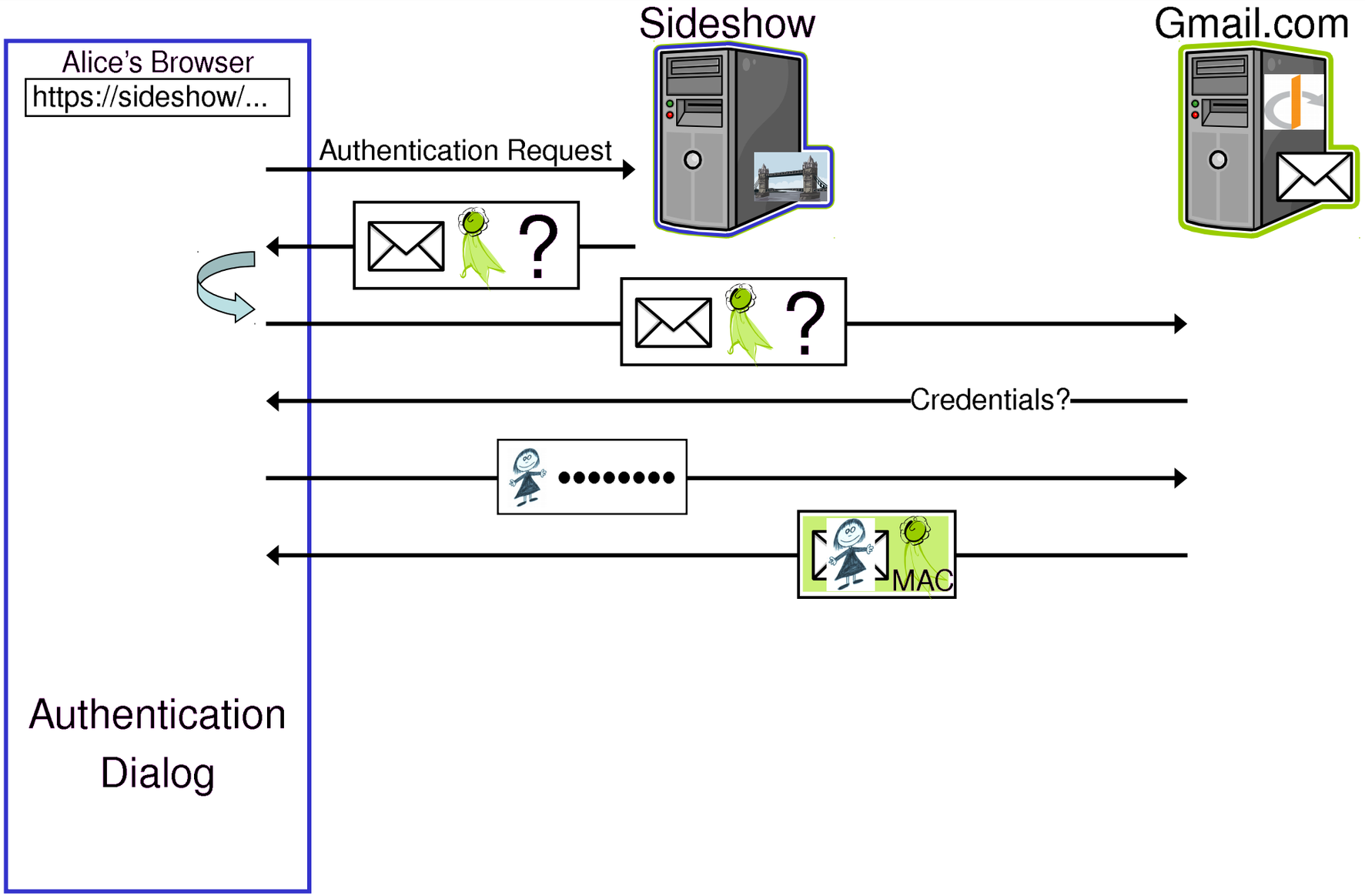
Identity Bridge



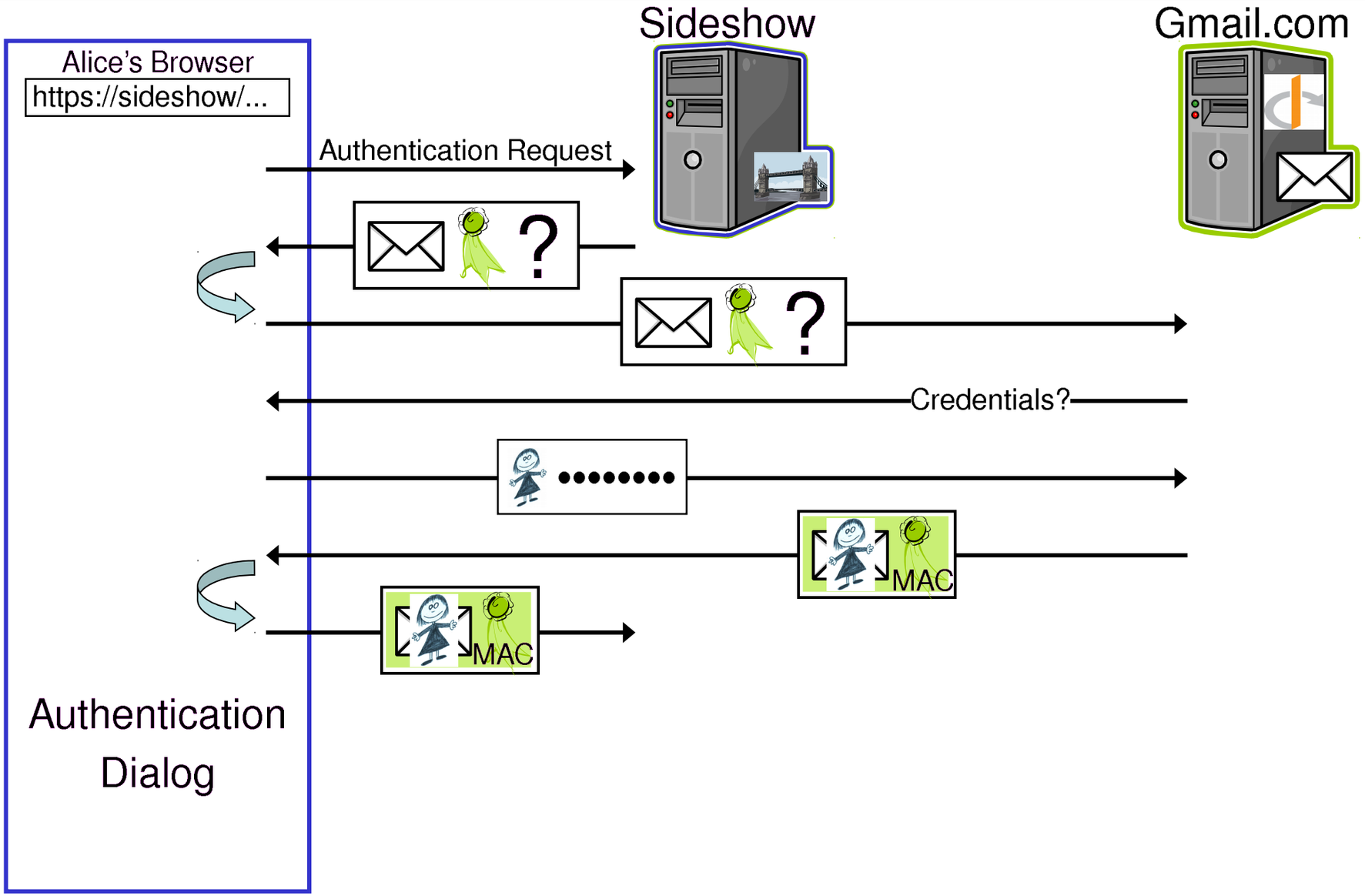
Identity Bridge



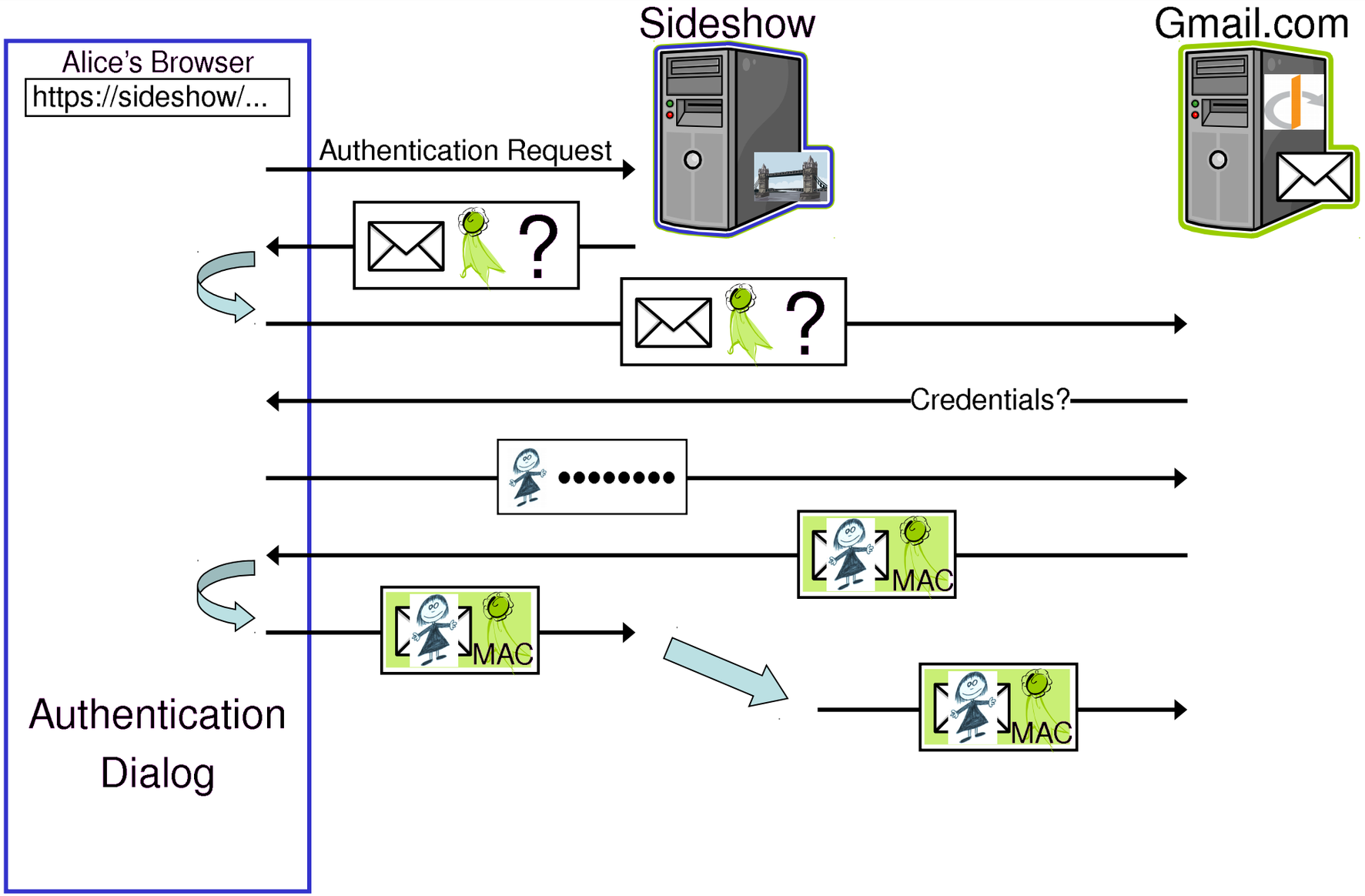
Identity Bridge



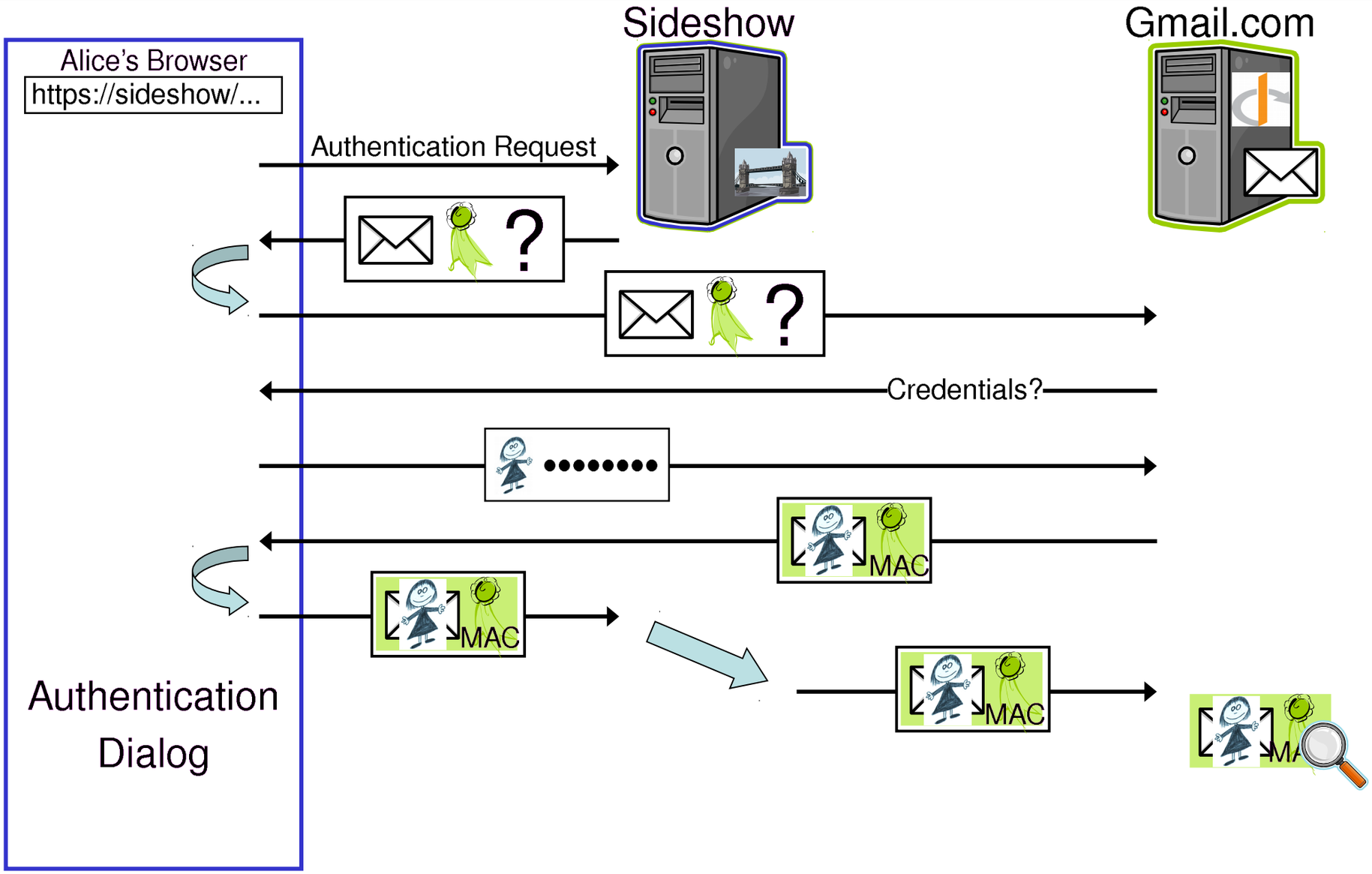
Identity Bridge



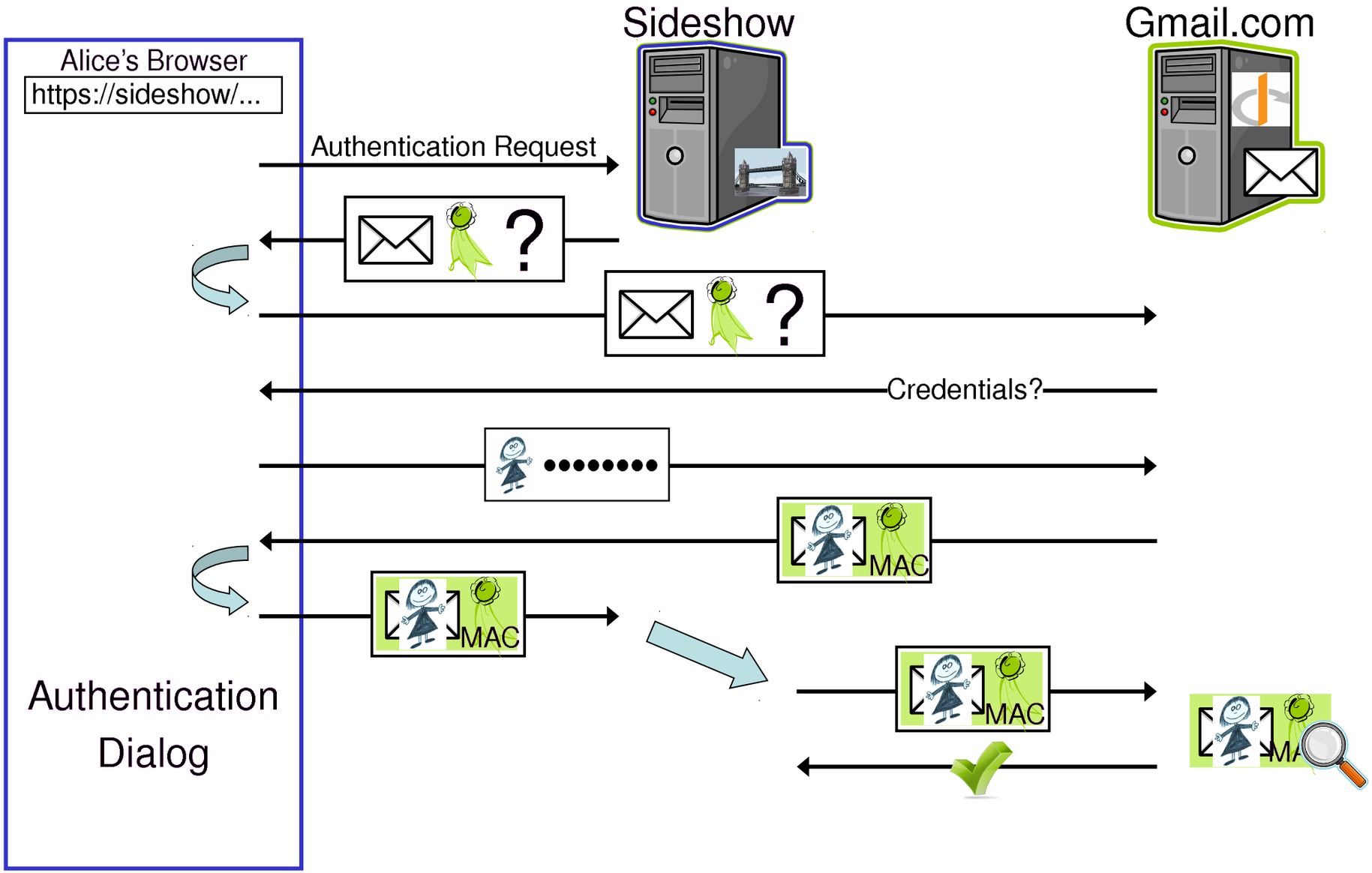
Identity Bridge



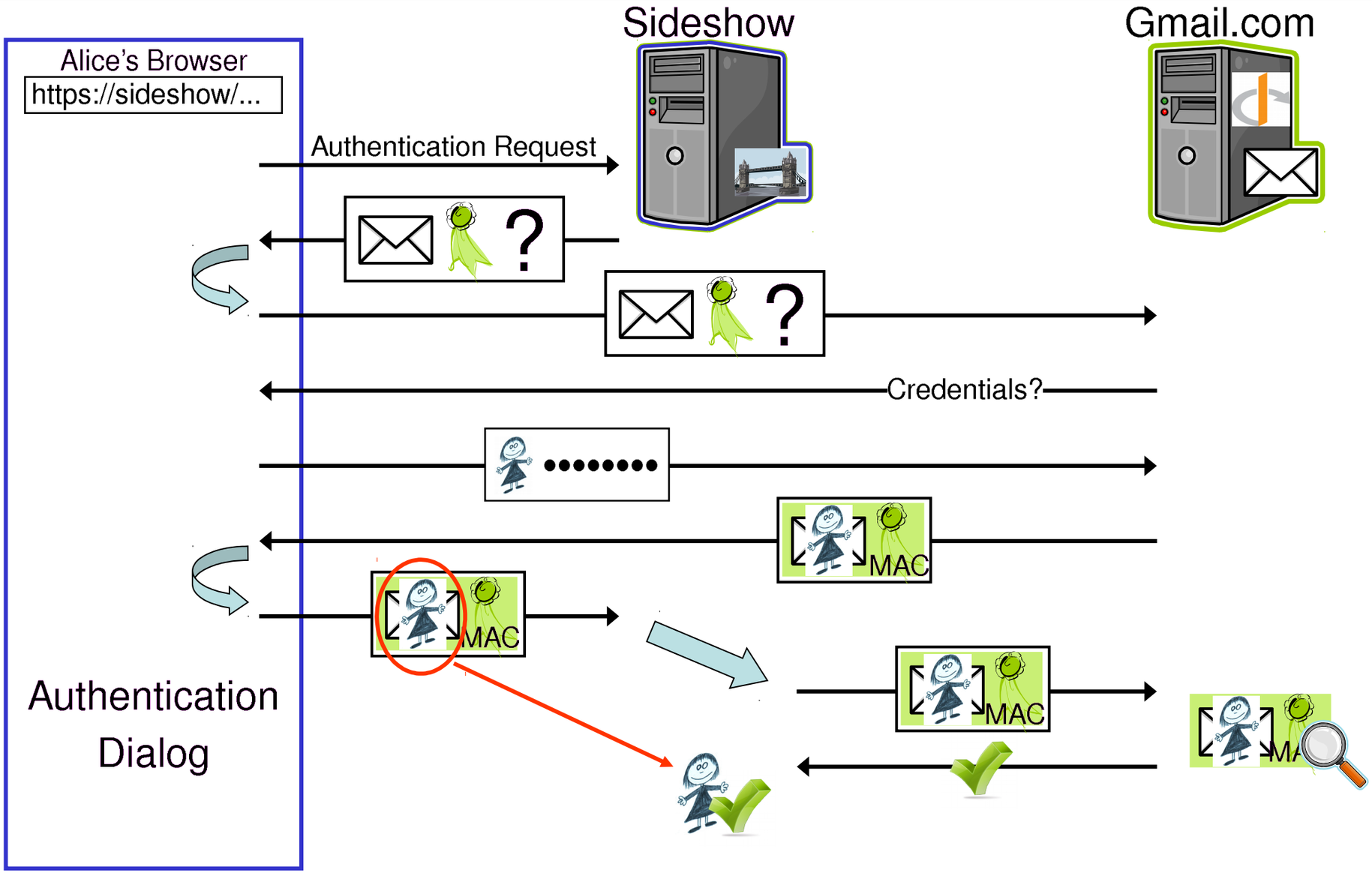
Identity Bridge



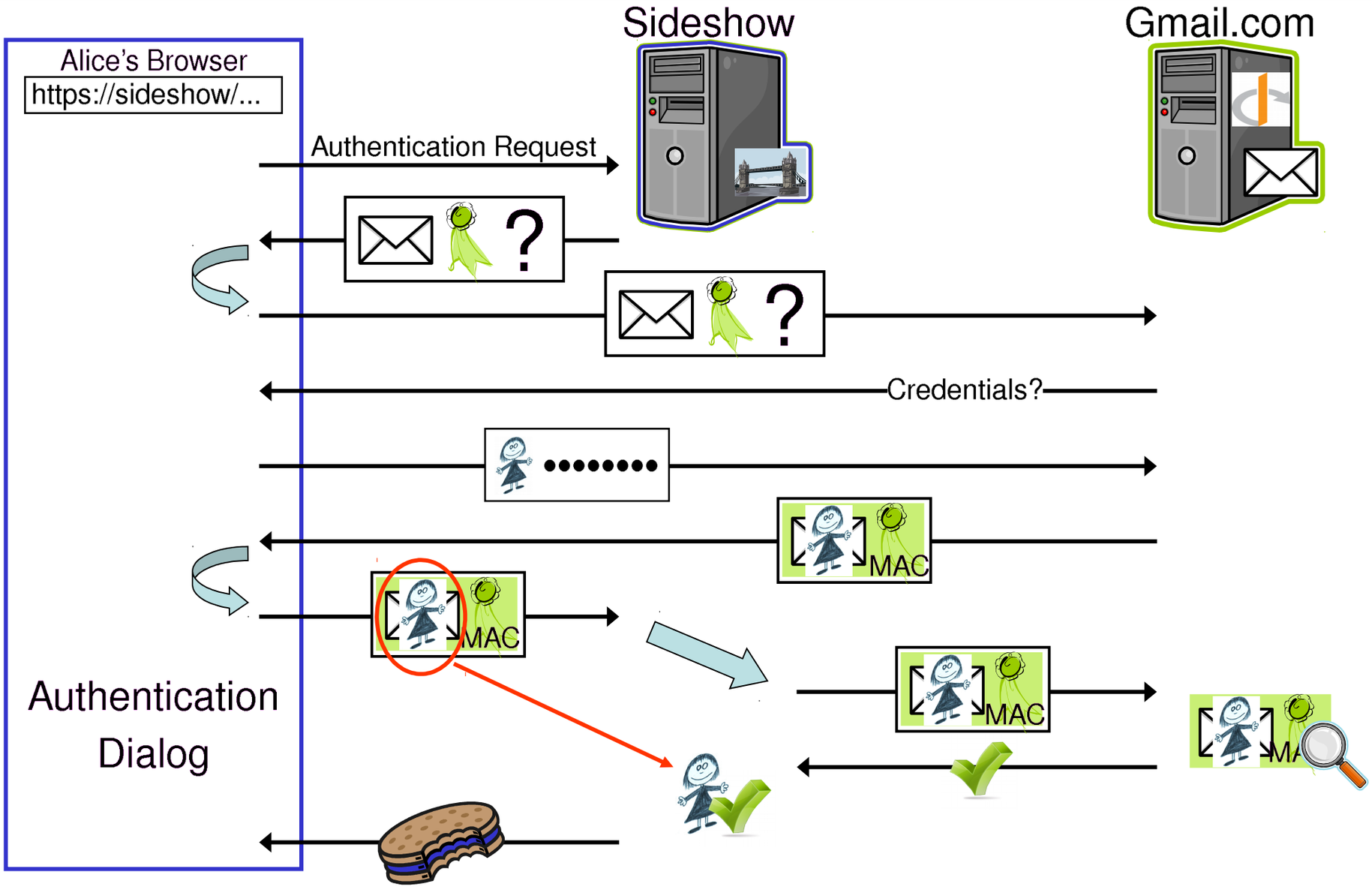
Identity Bridge



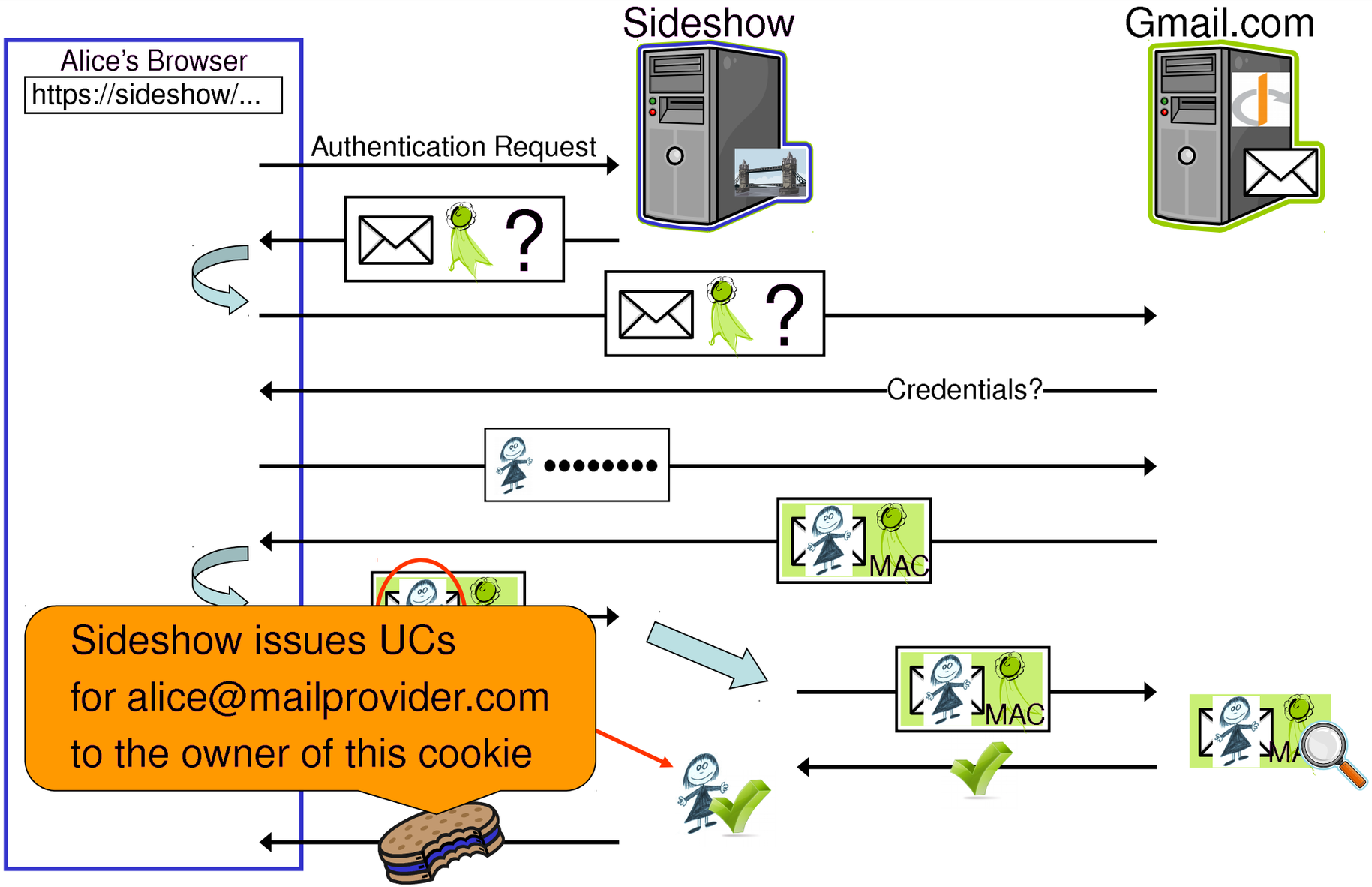
Identity Bridge



Identity Bridge



Identity Bridge



Selected Attacks on BrowserID a.k.a. Mozilla Persona

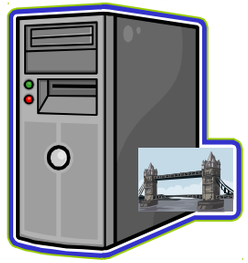
Identity Forgery (I)

Attacker's Browser

https://sideshow/...

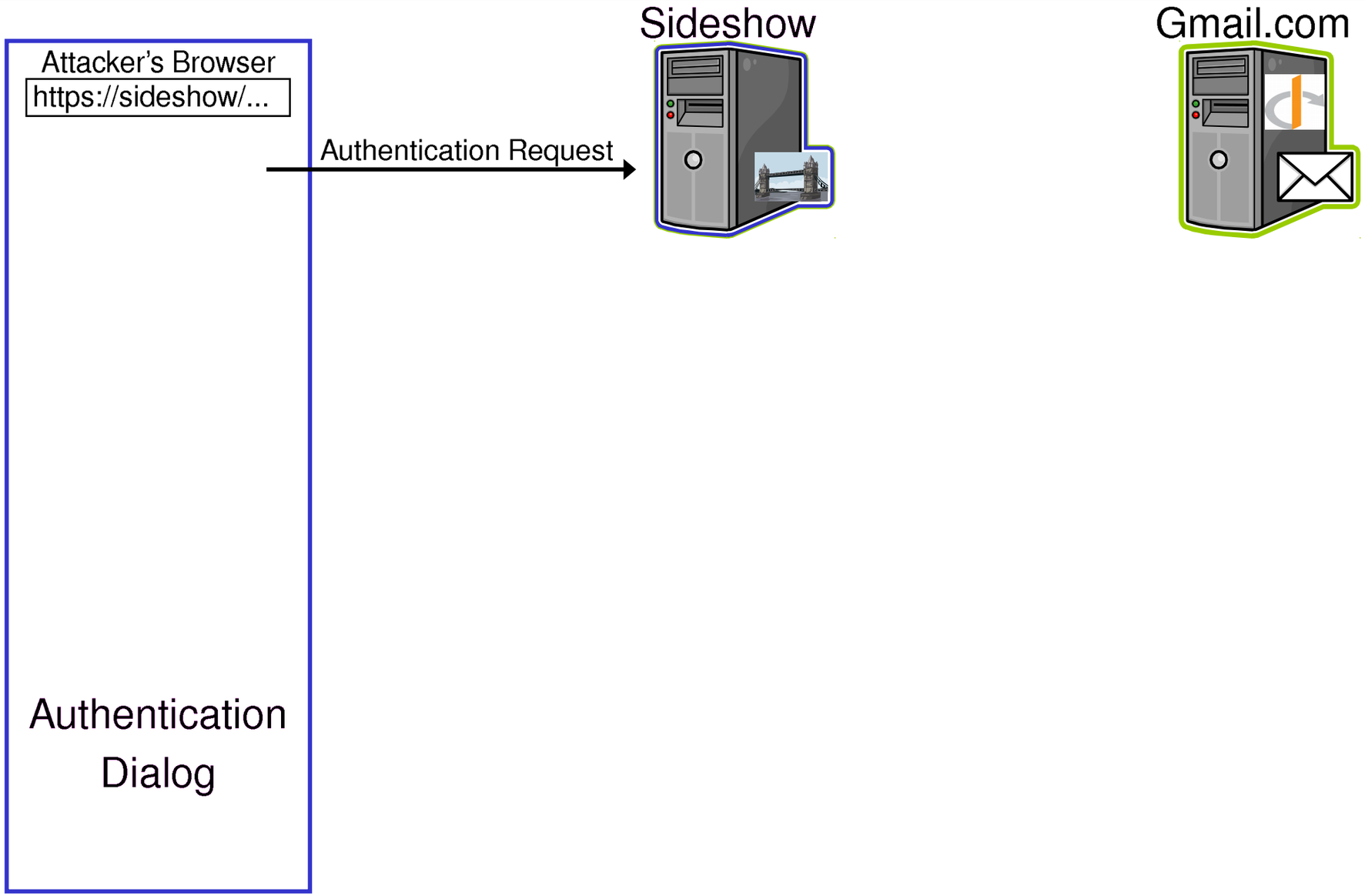
Authentication Dialog

Sideshow

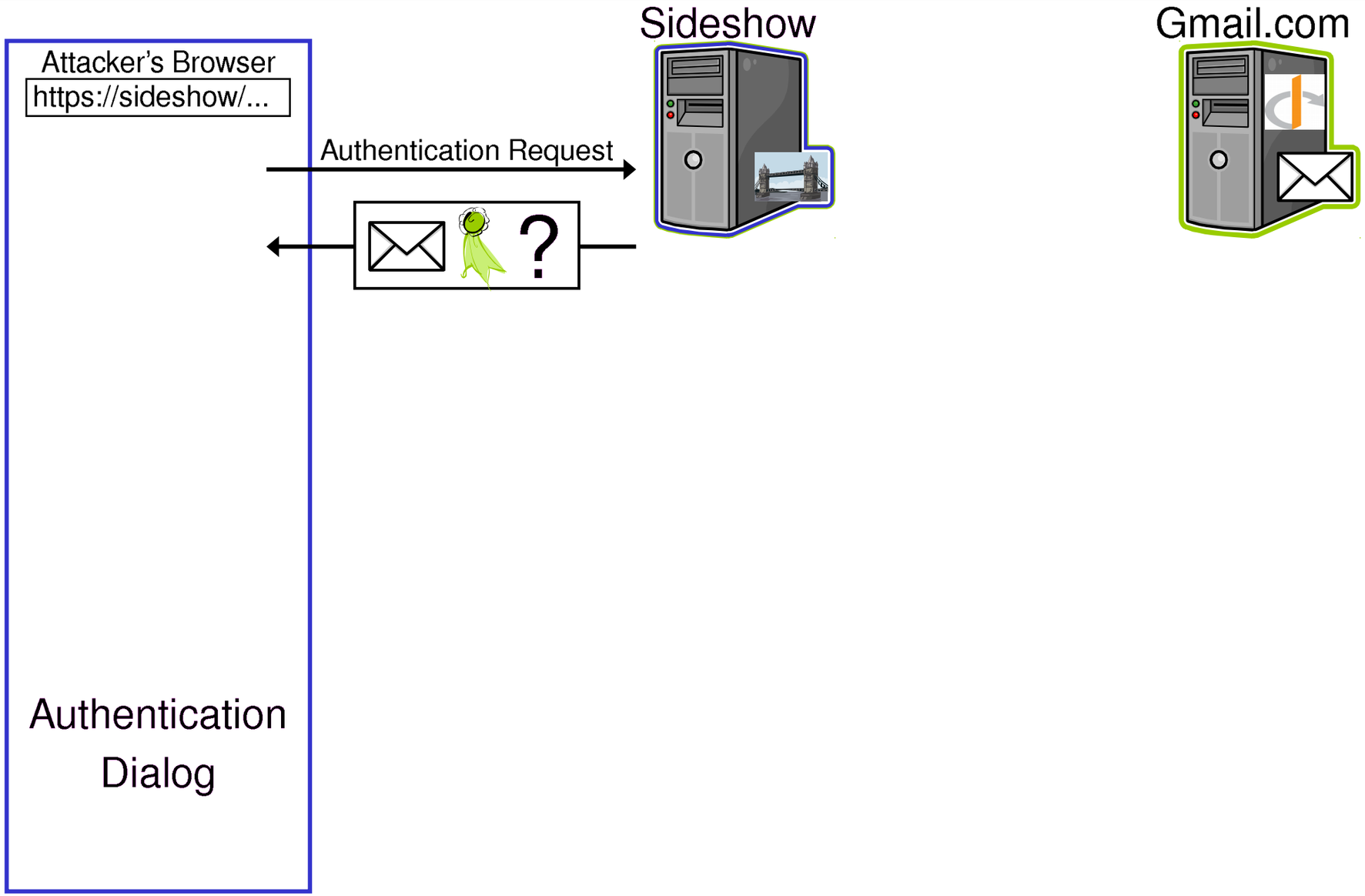


Gmail.com

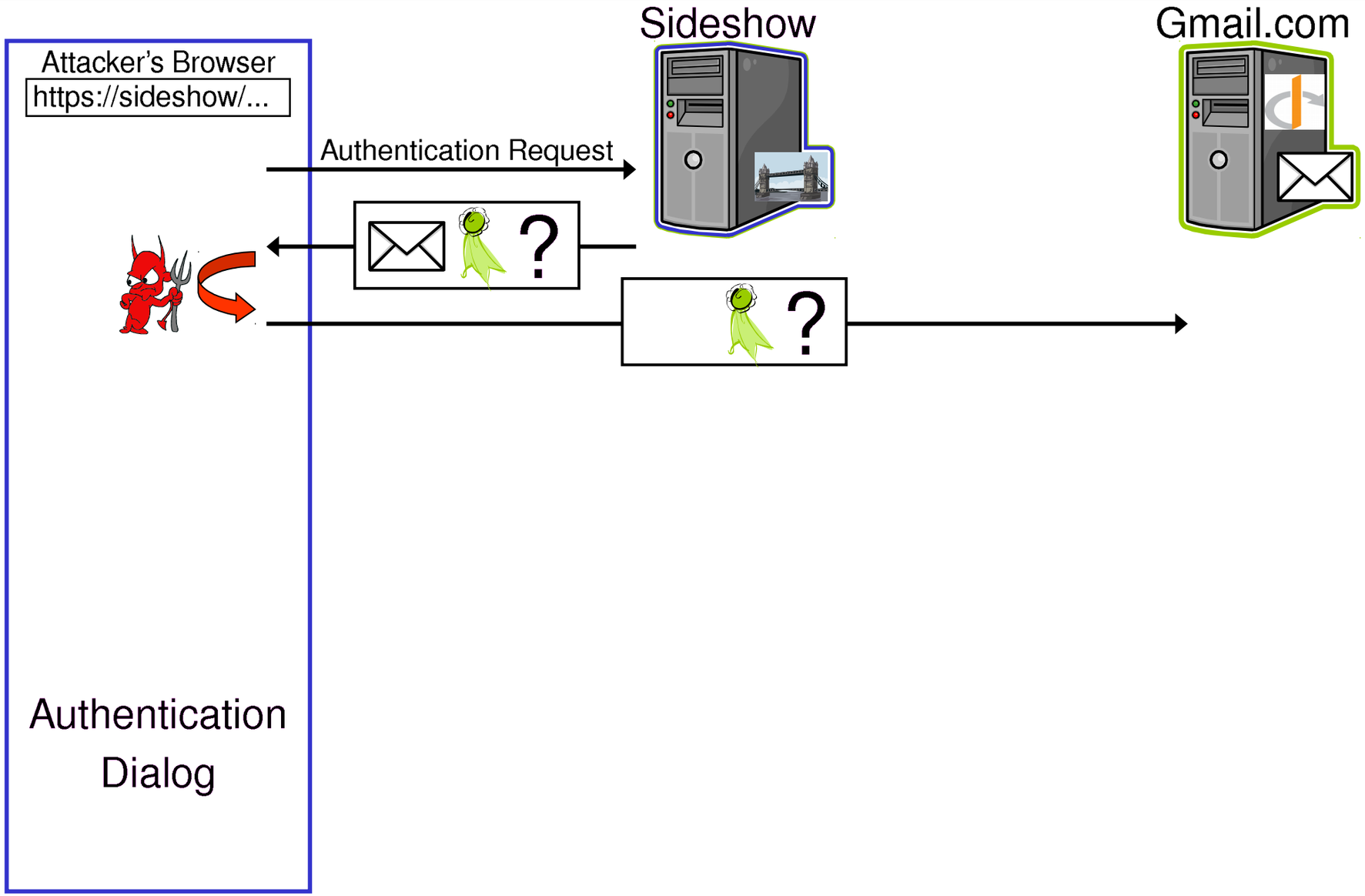
Identity Forgery (I)



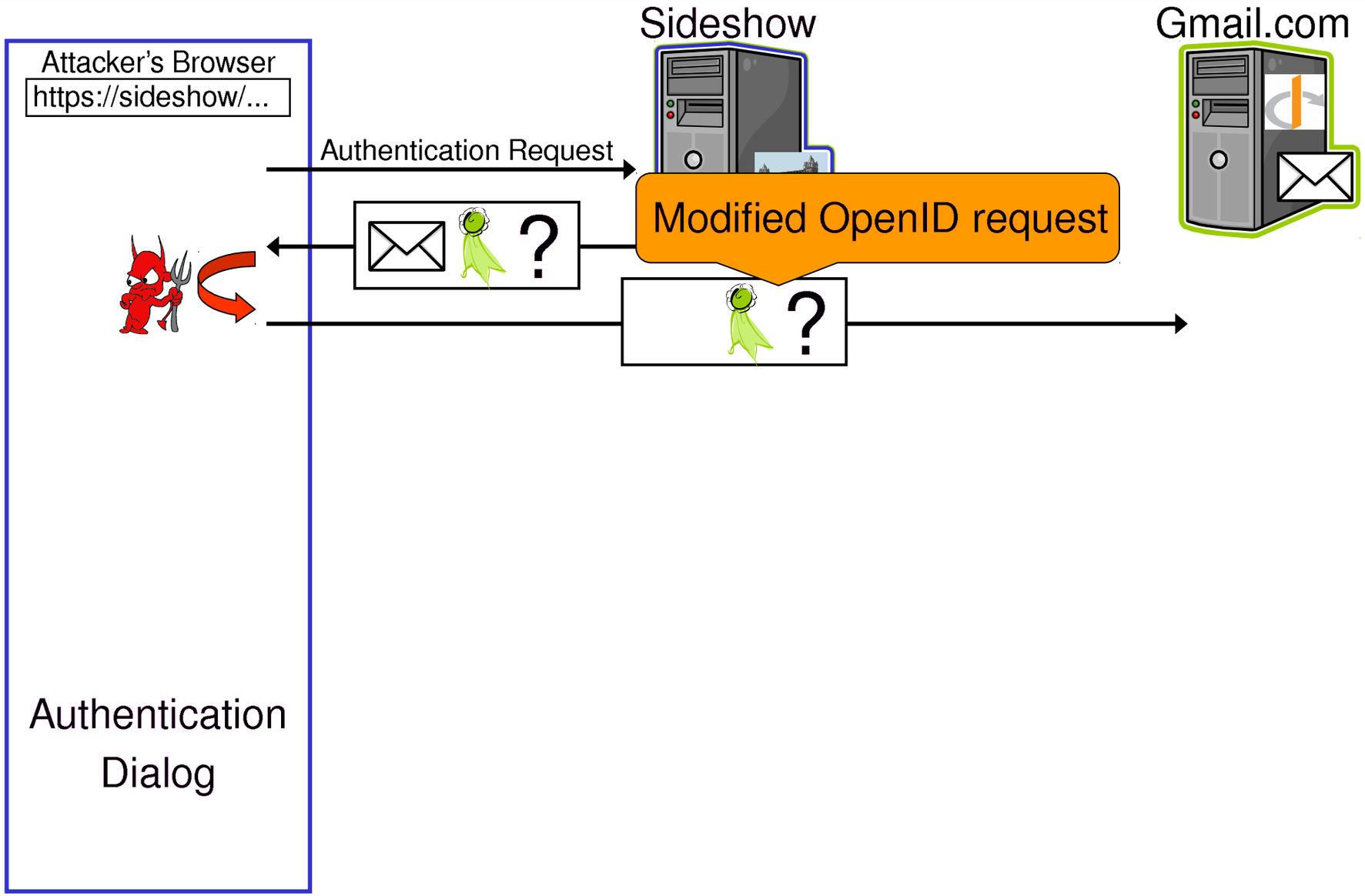
Identity Forgery (I)



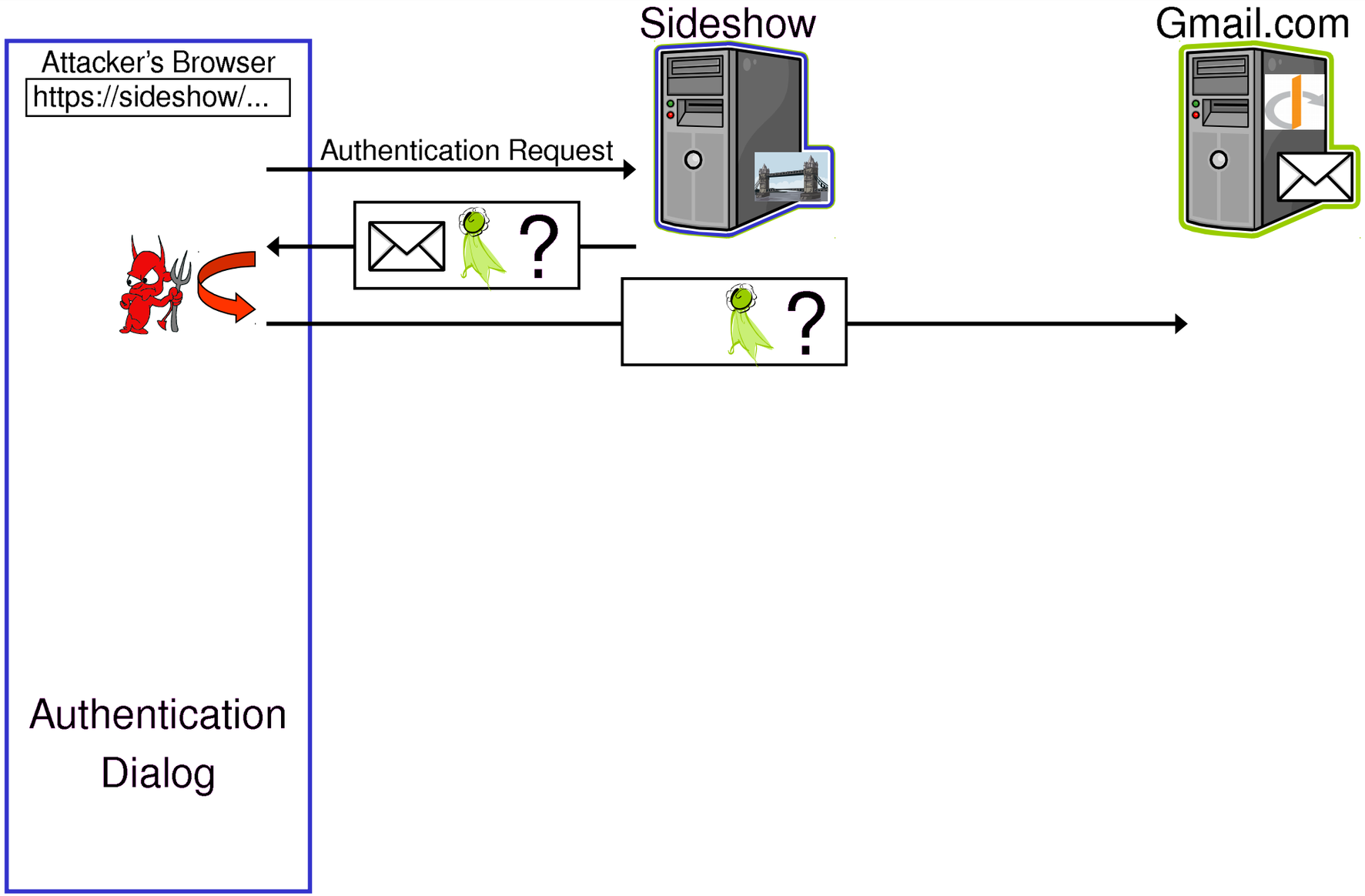
Identity Forgery (I)



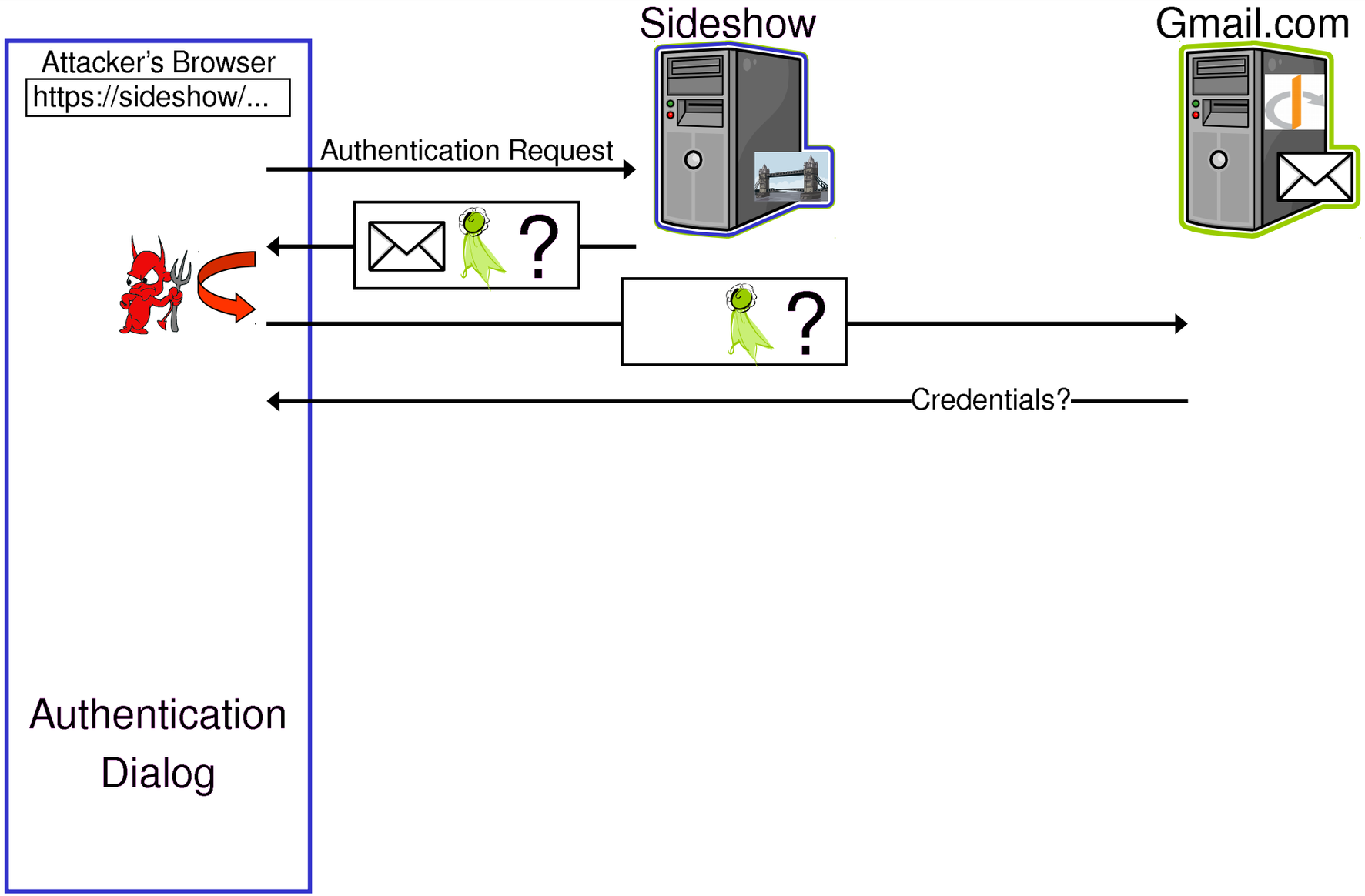
Identity Forgery (I)



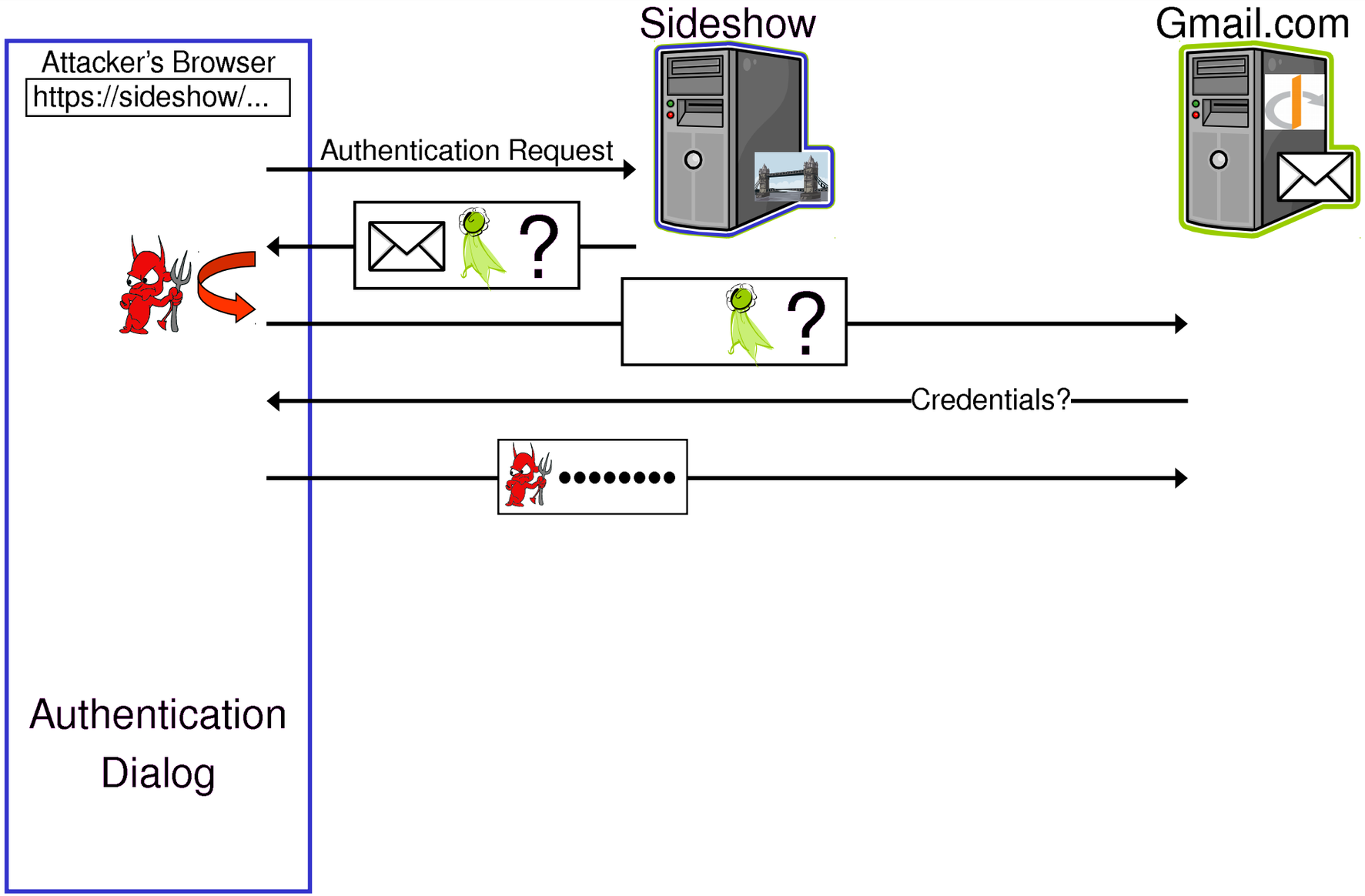
Identity Forgery (I)



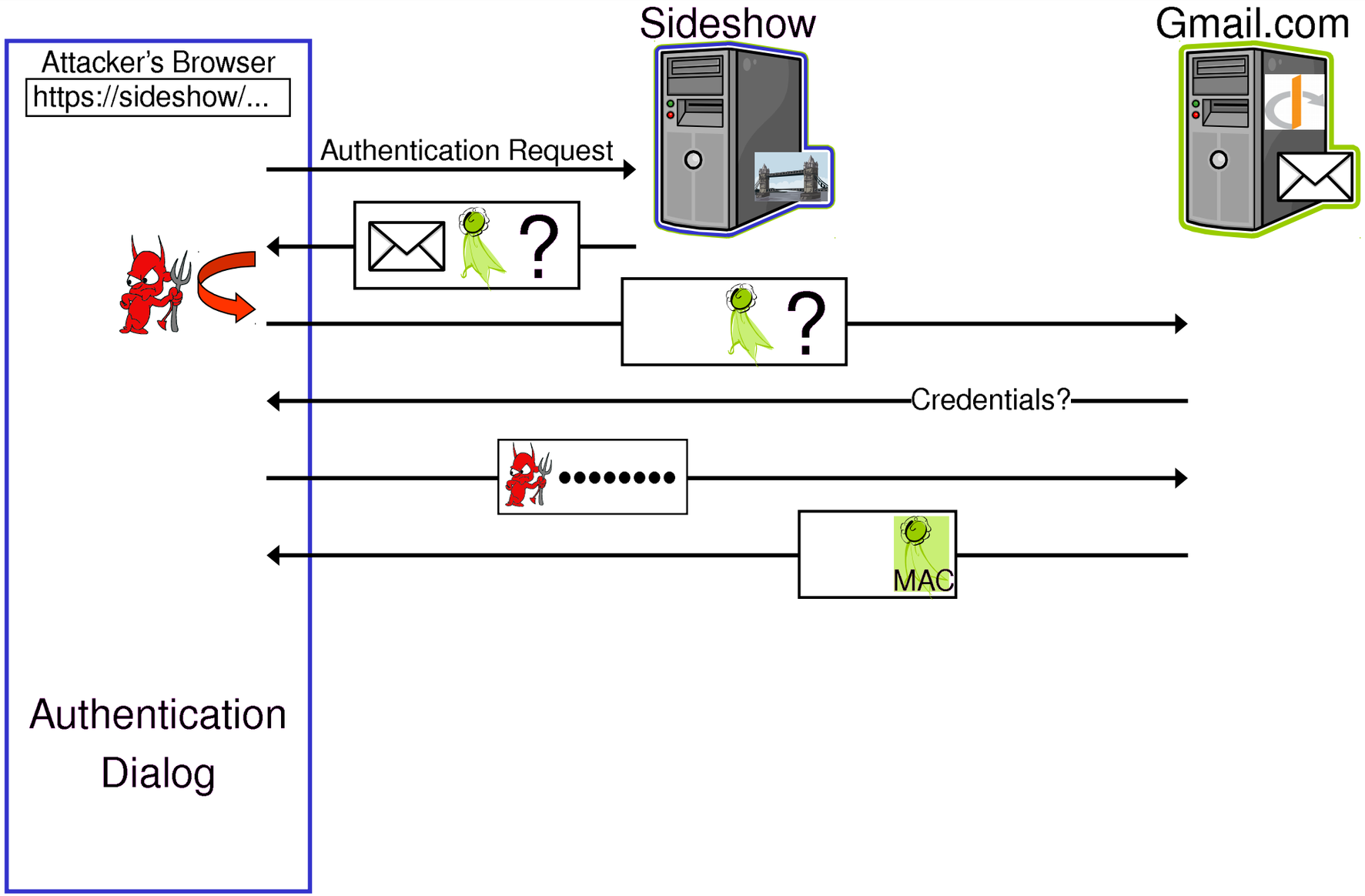
Identity Forgery (I)



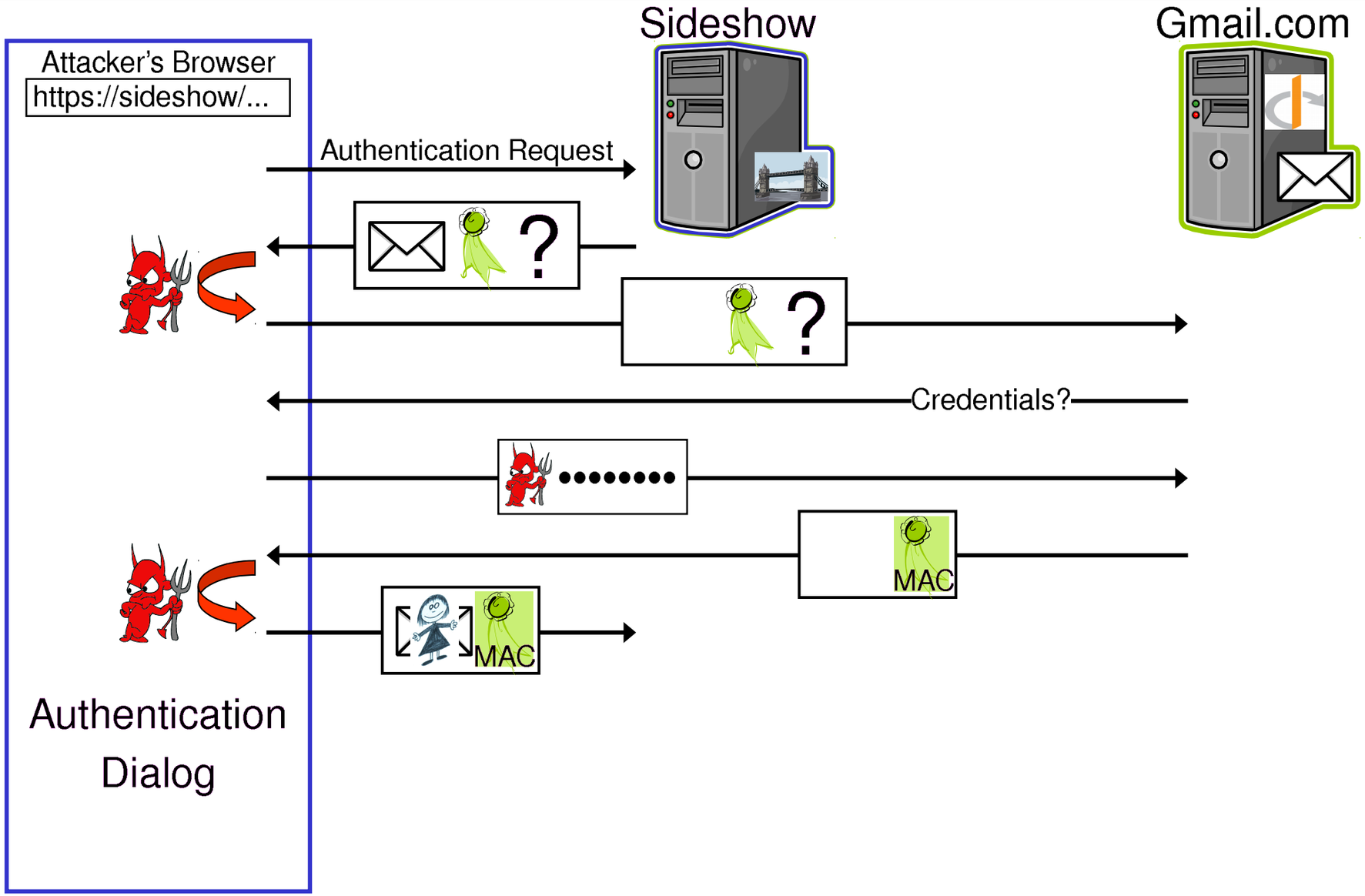
Identity Forgery (I)



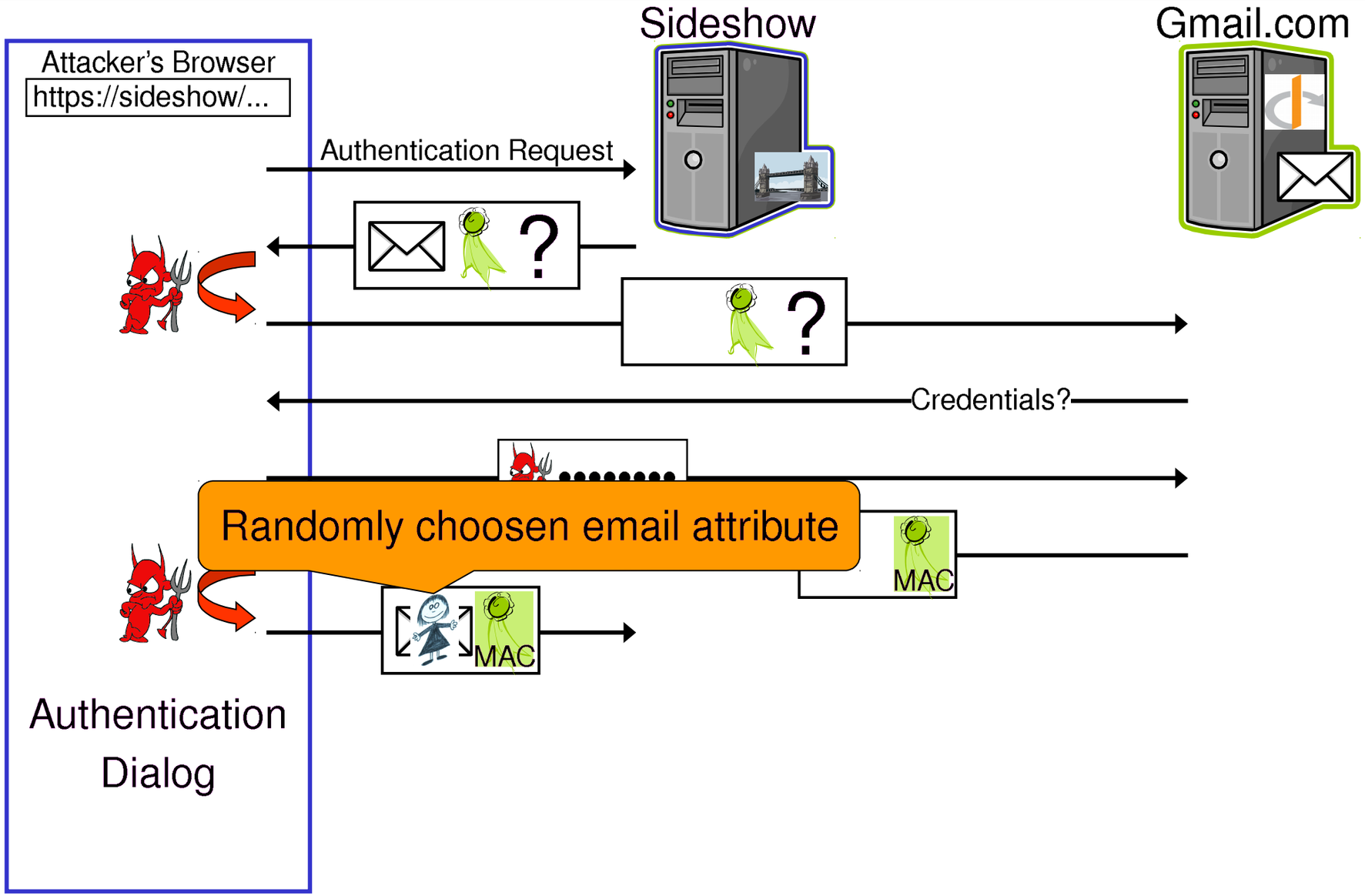
Identity Forgery (I)



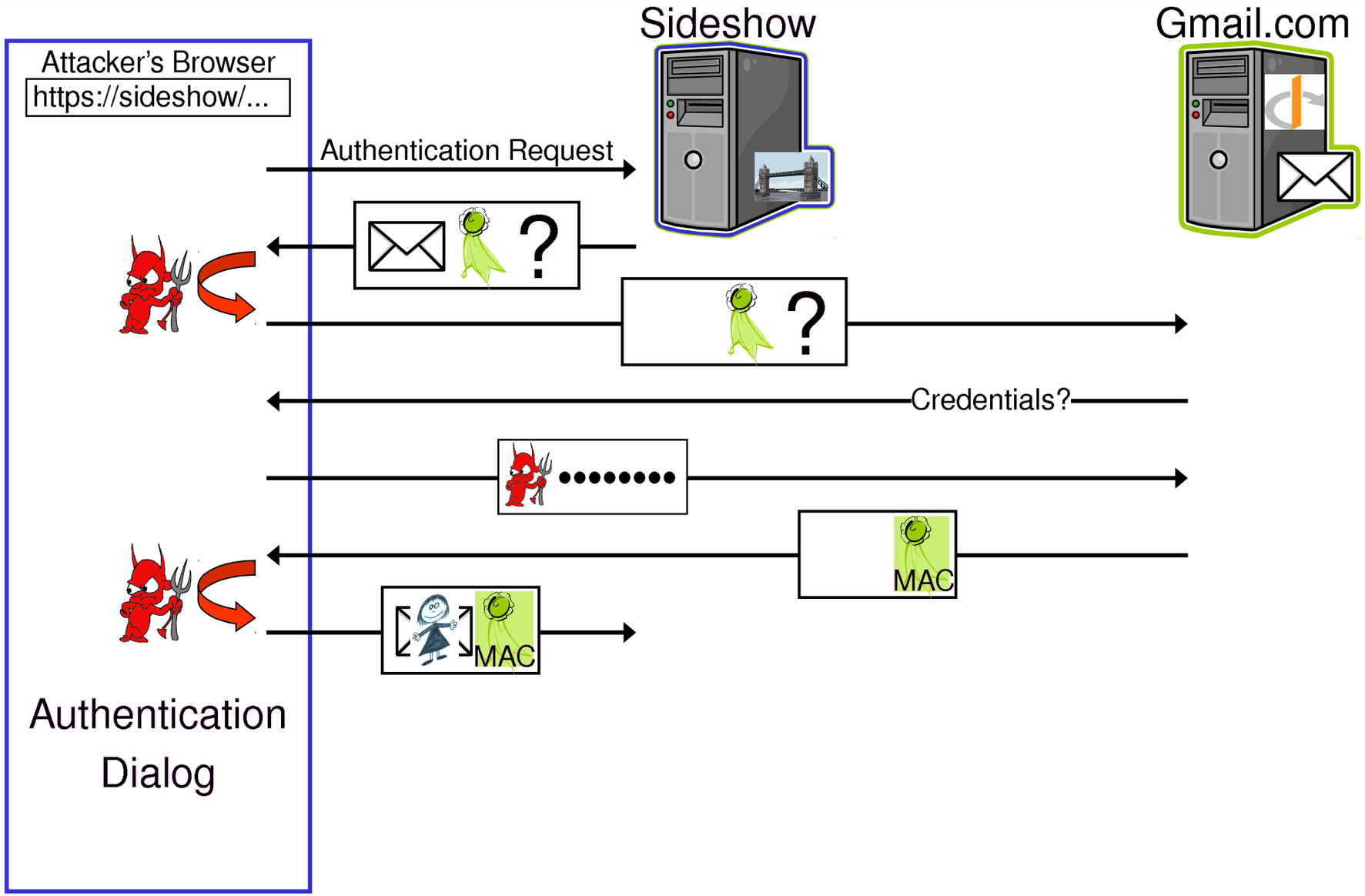
Identity Forgery (I)



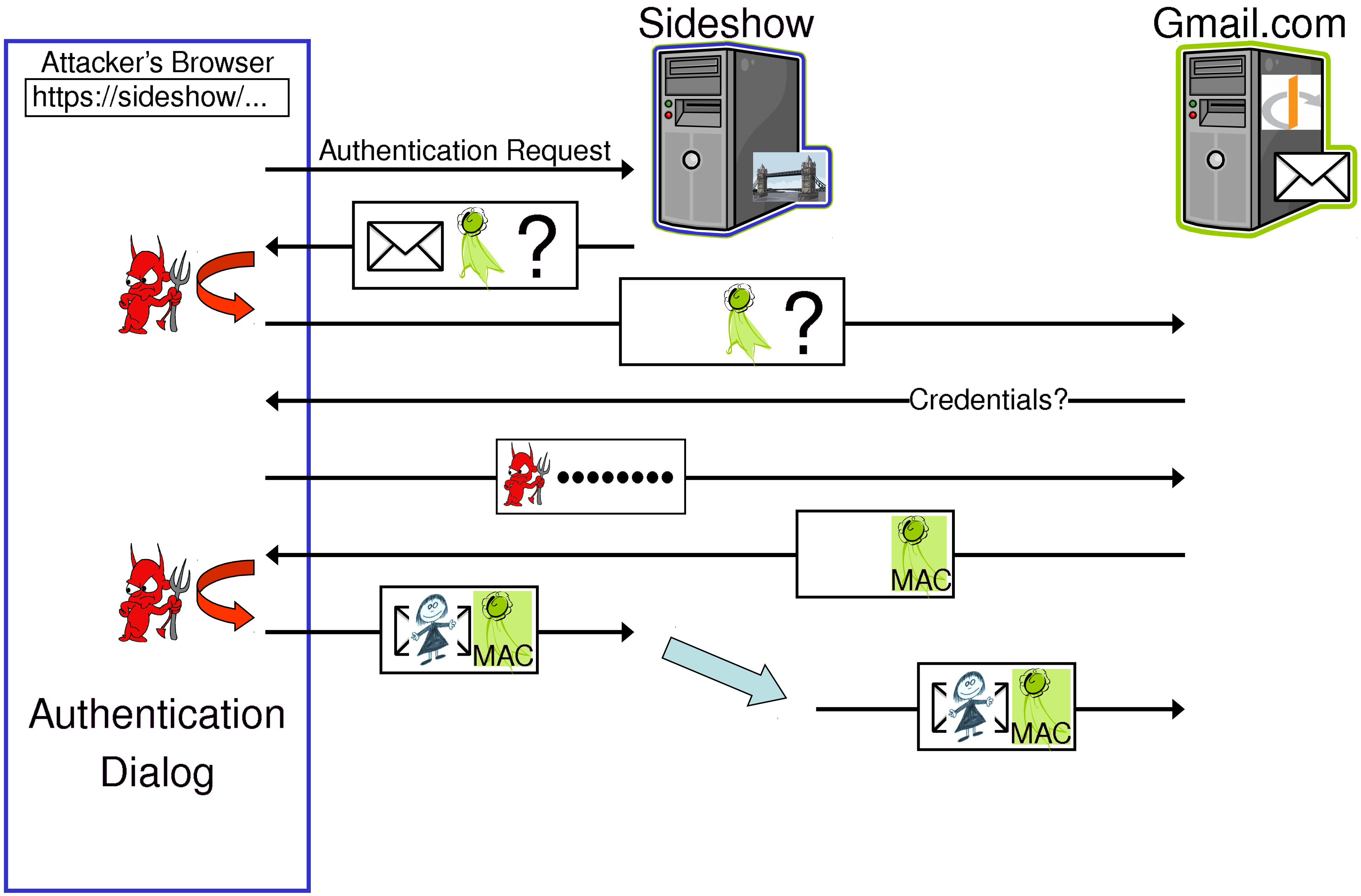
Identity Forgery (I)



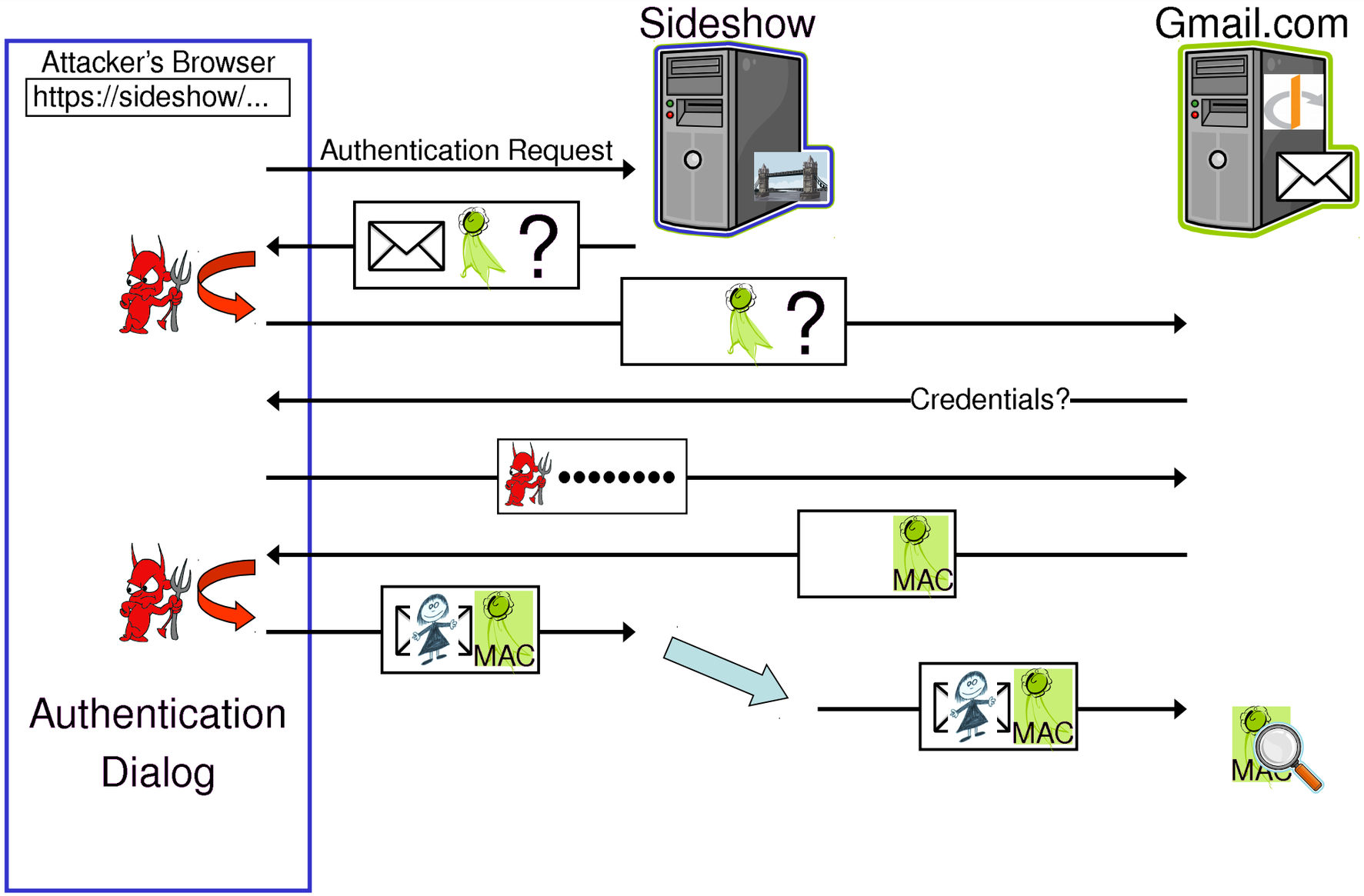
Identity Forgery (I)



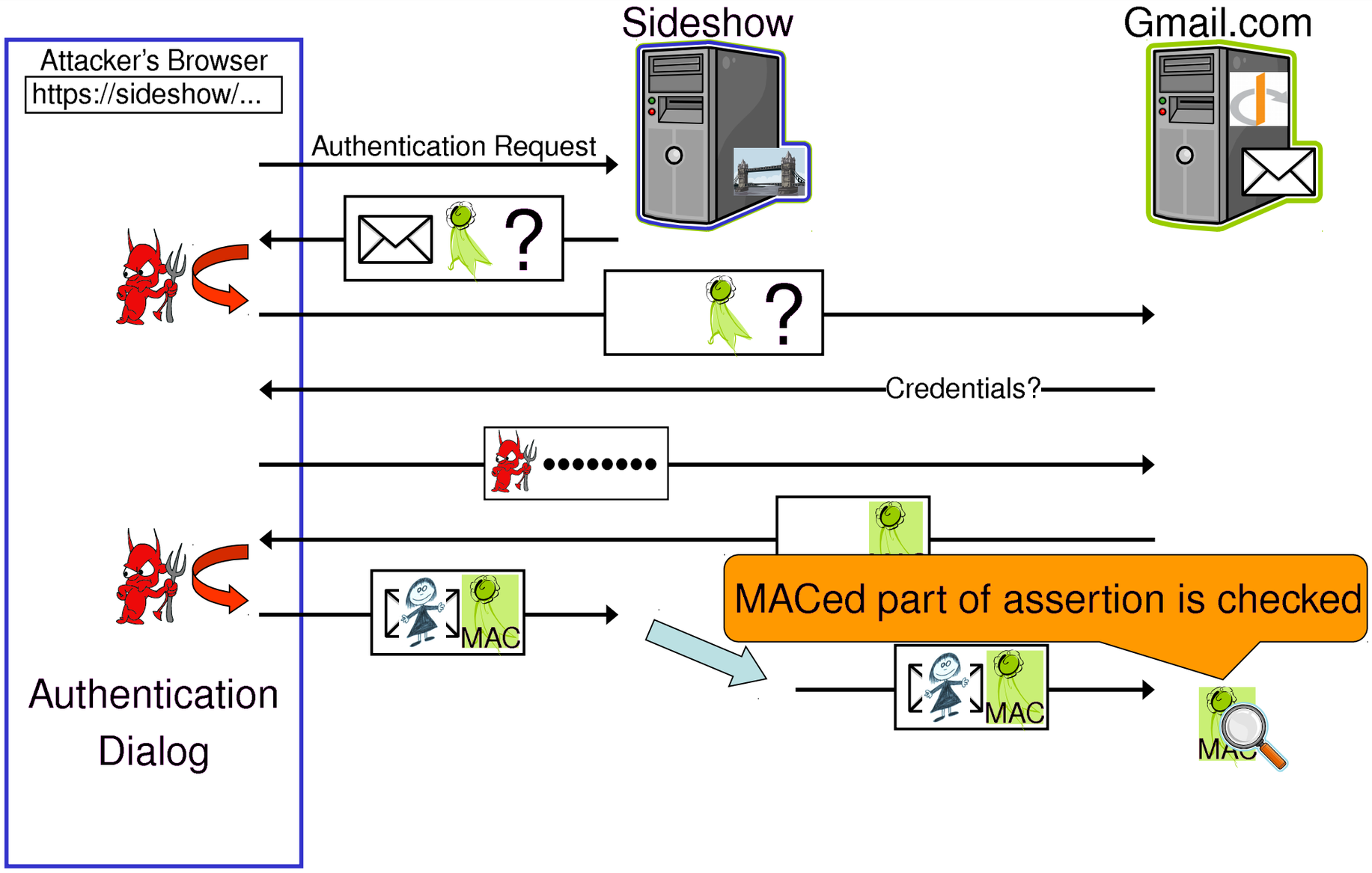
Identity Forgery (I)



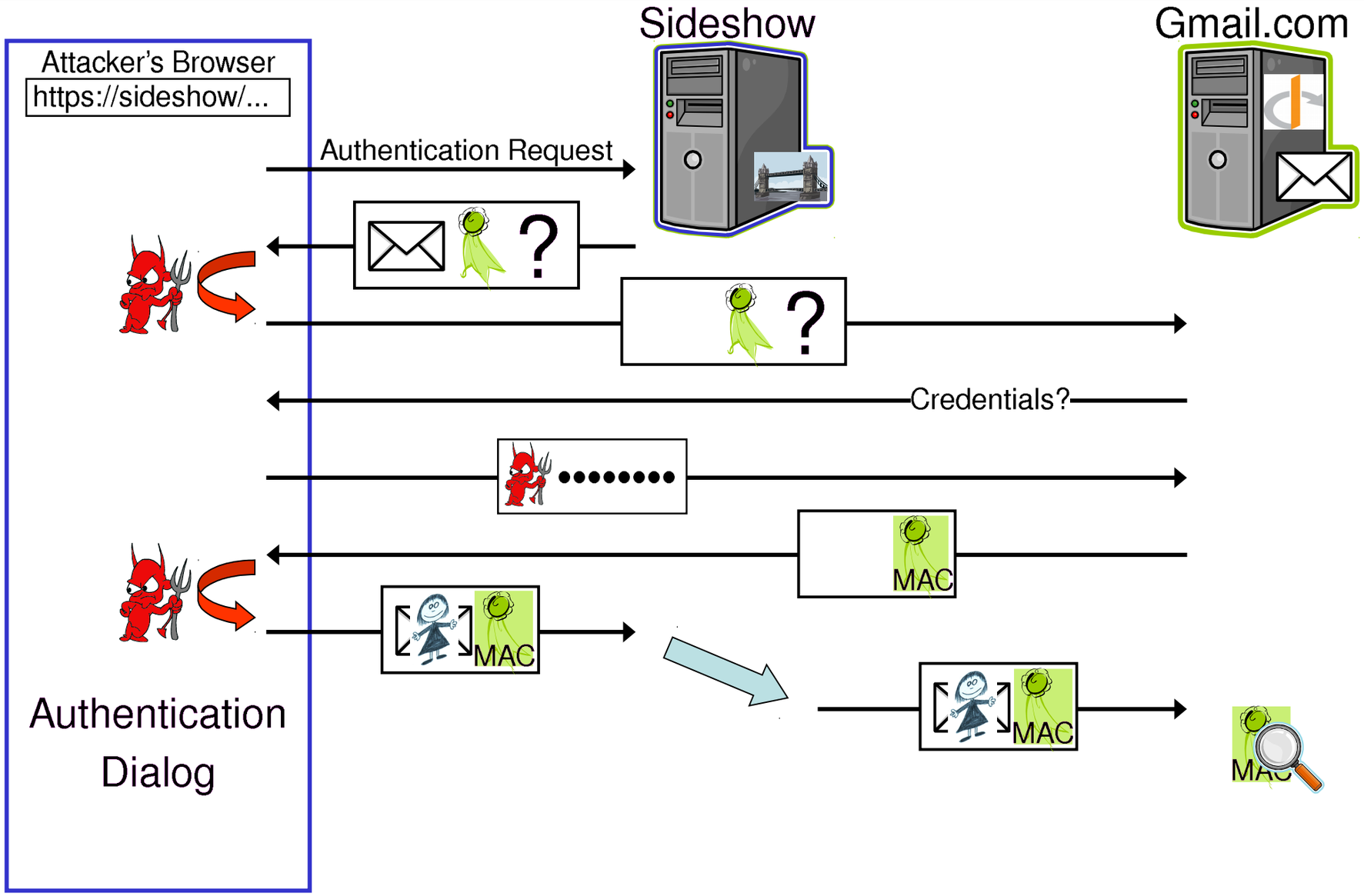
Identity Forgery (I)



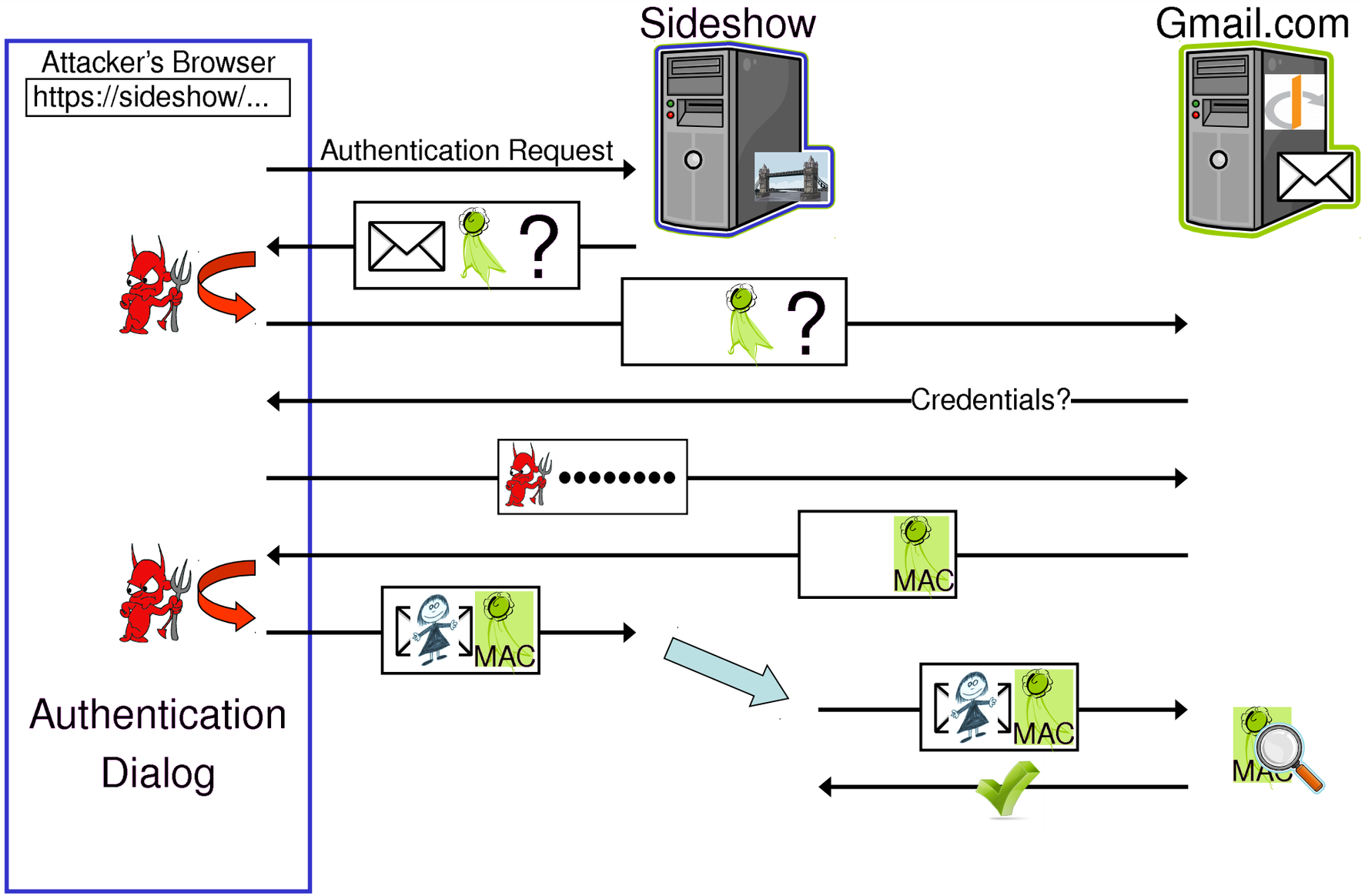
Identity Forgery (I)



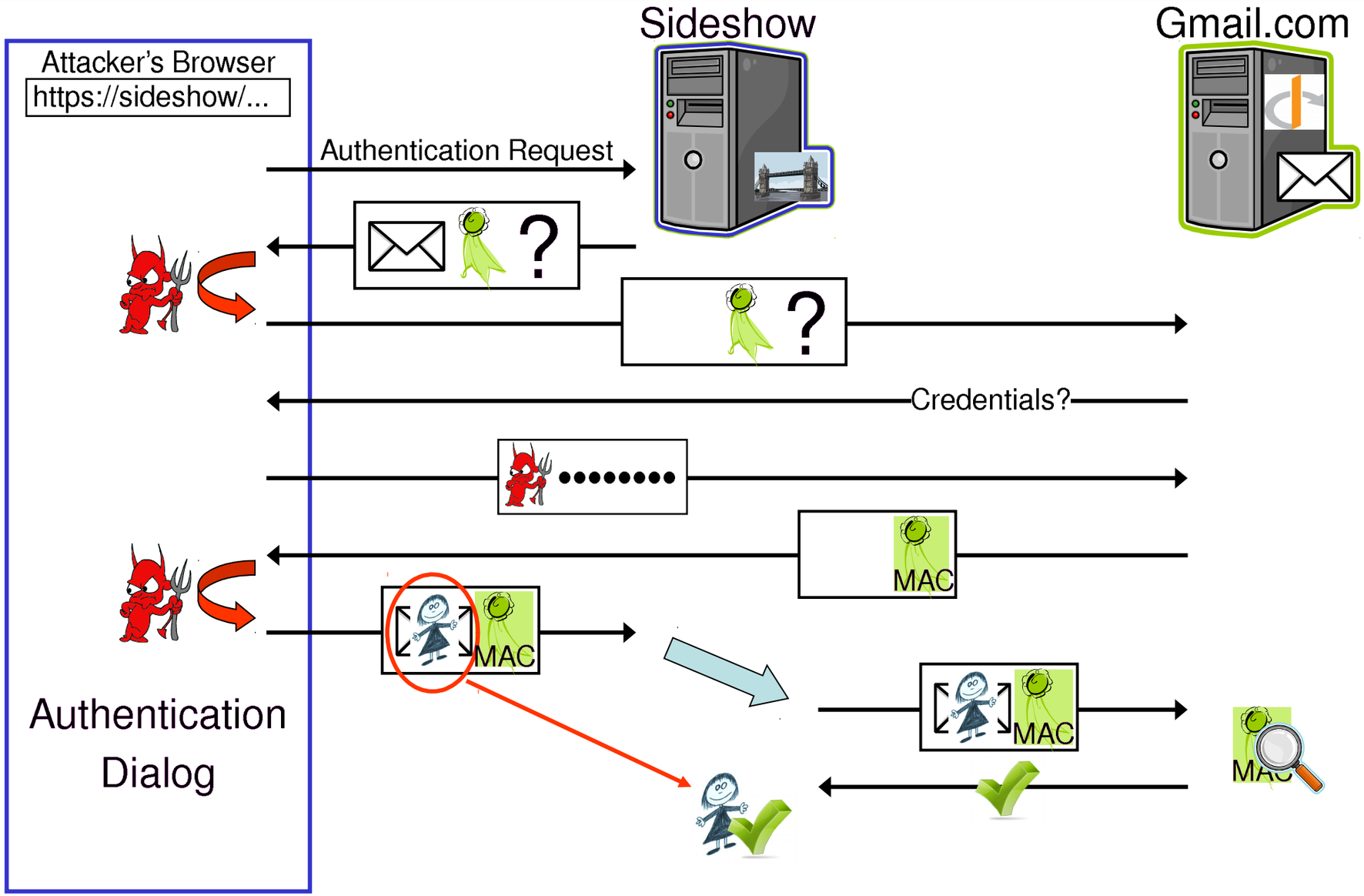
Identity Forgery (I)



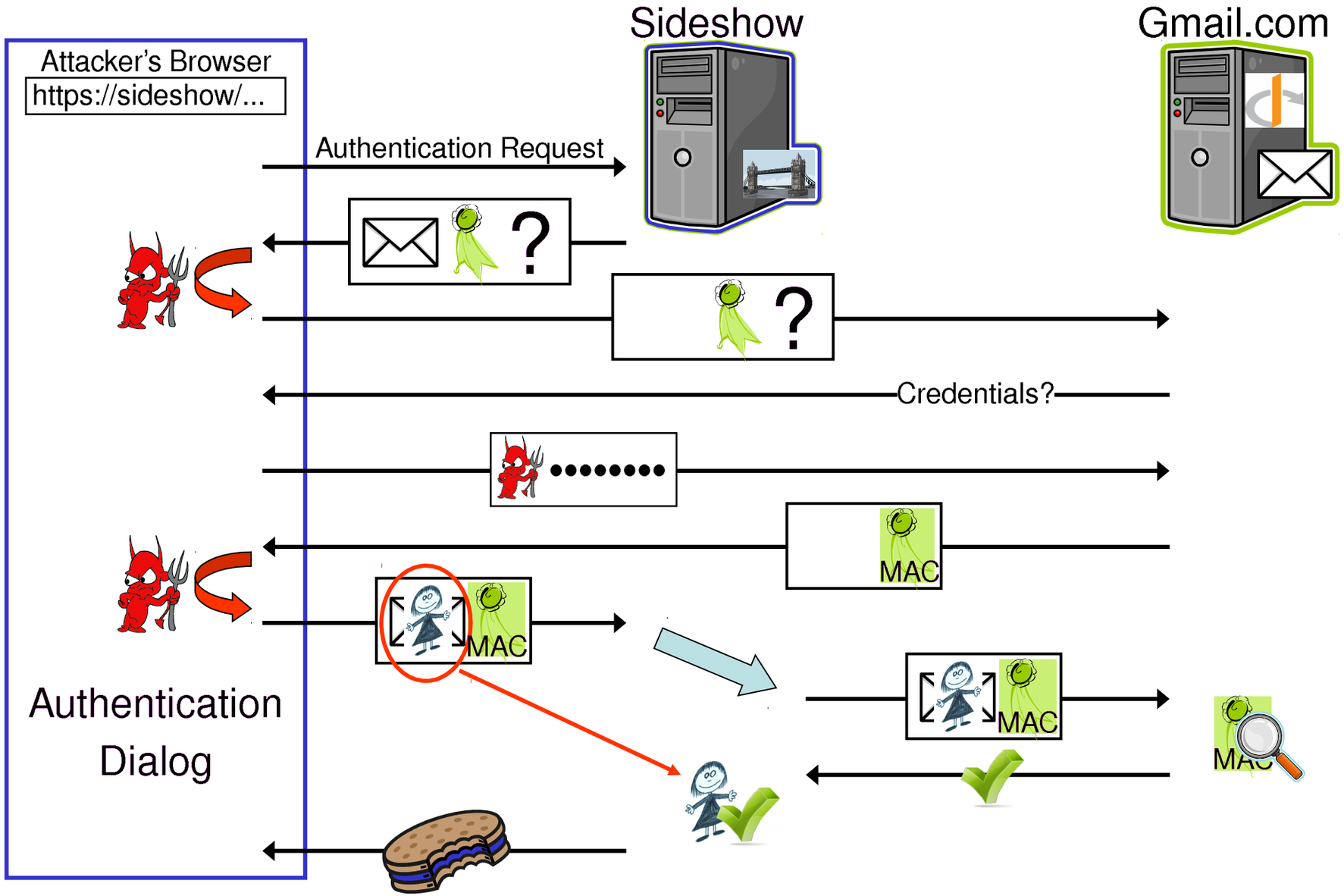
Identity Forgery (I)



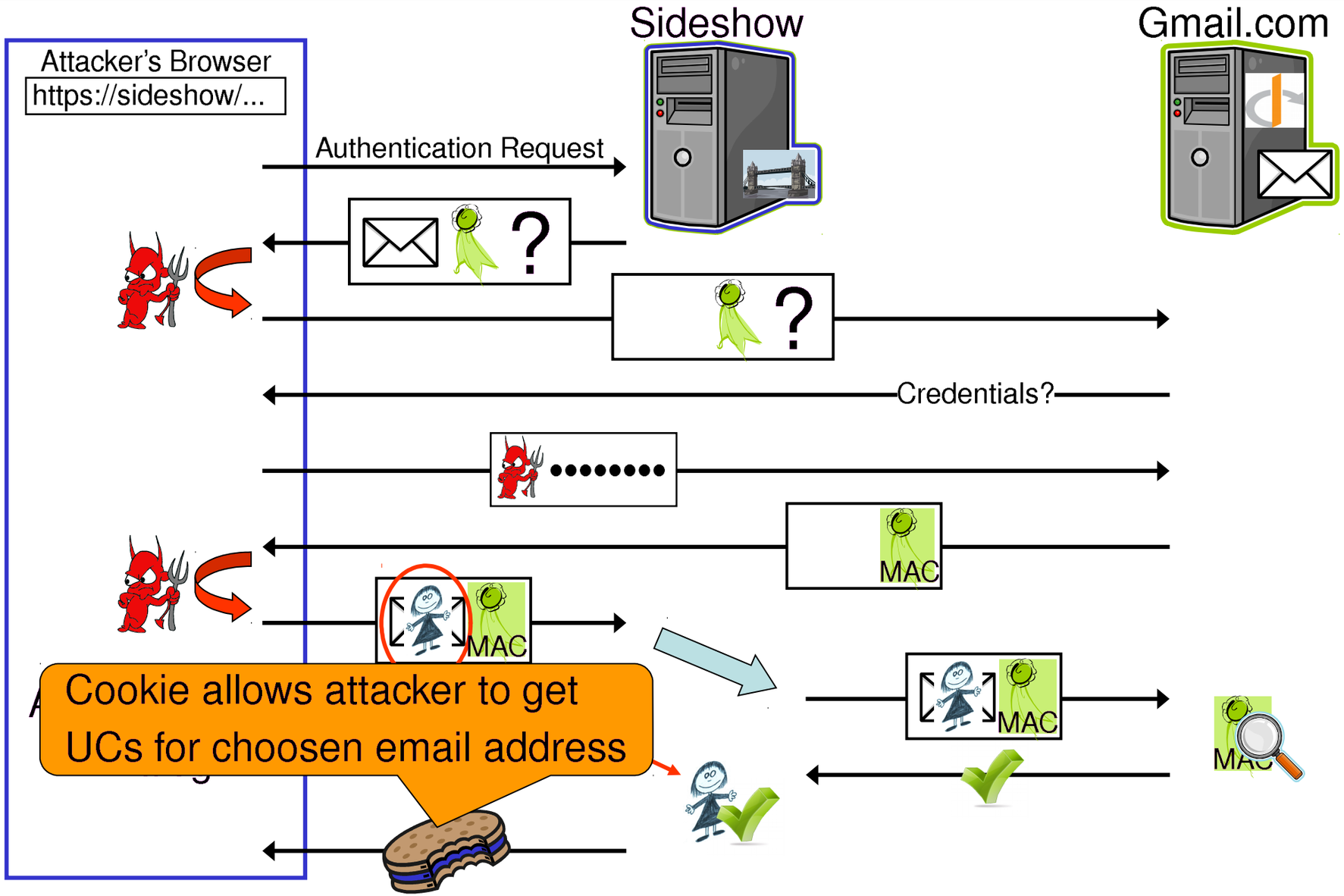
Identity Forgery (I)



Identity Forgery (I)



Identity Forgery (I)



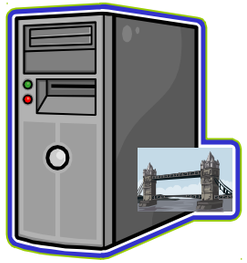
Identity Forgery (II)

Attacker's Browser

https://sideshow/...

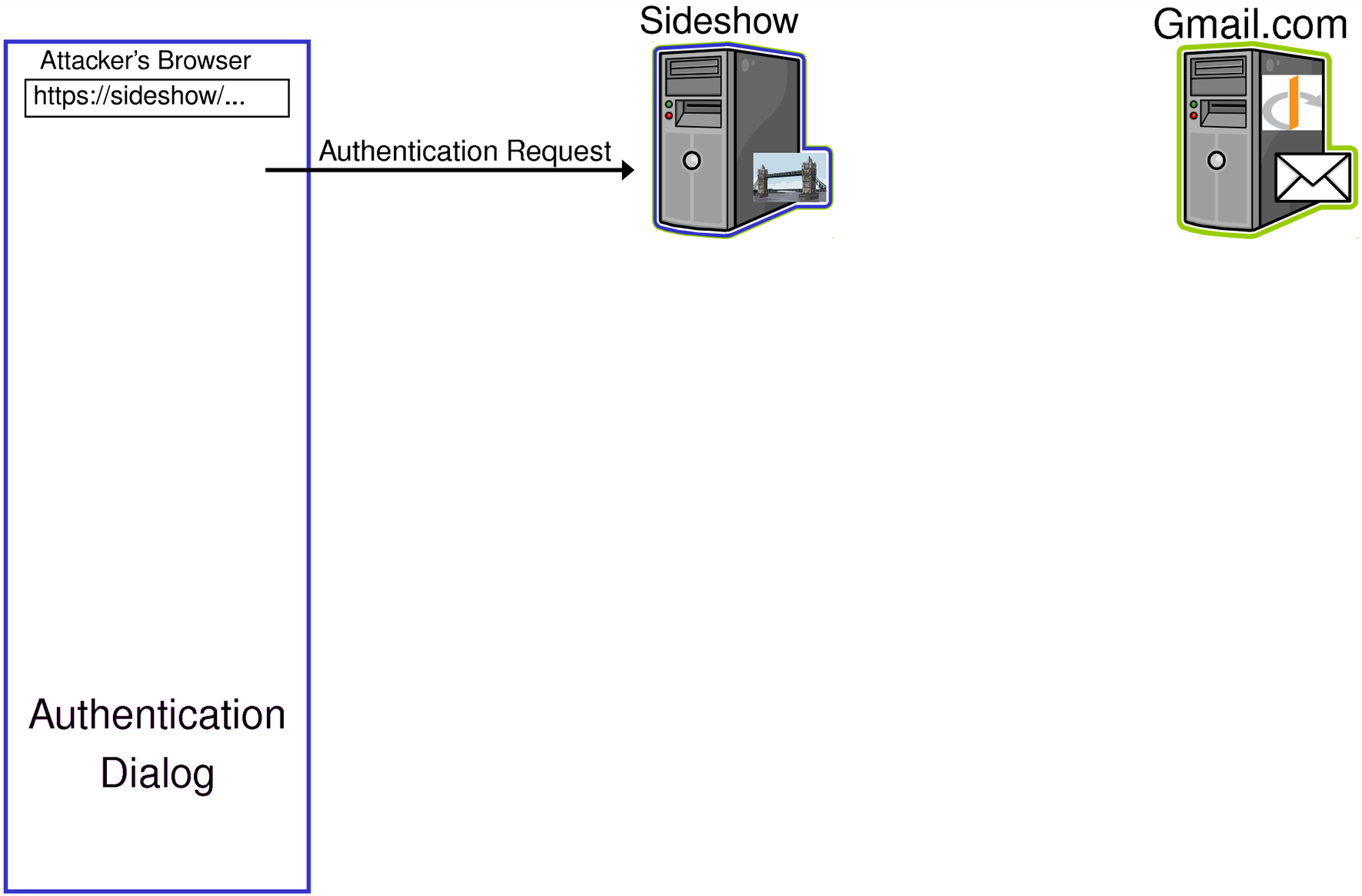
Authentication Dialog

Sideshow

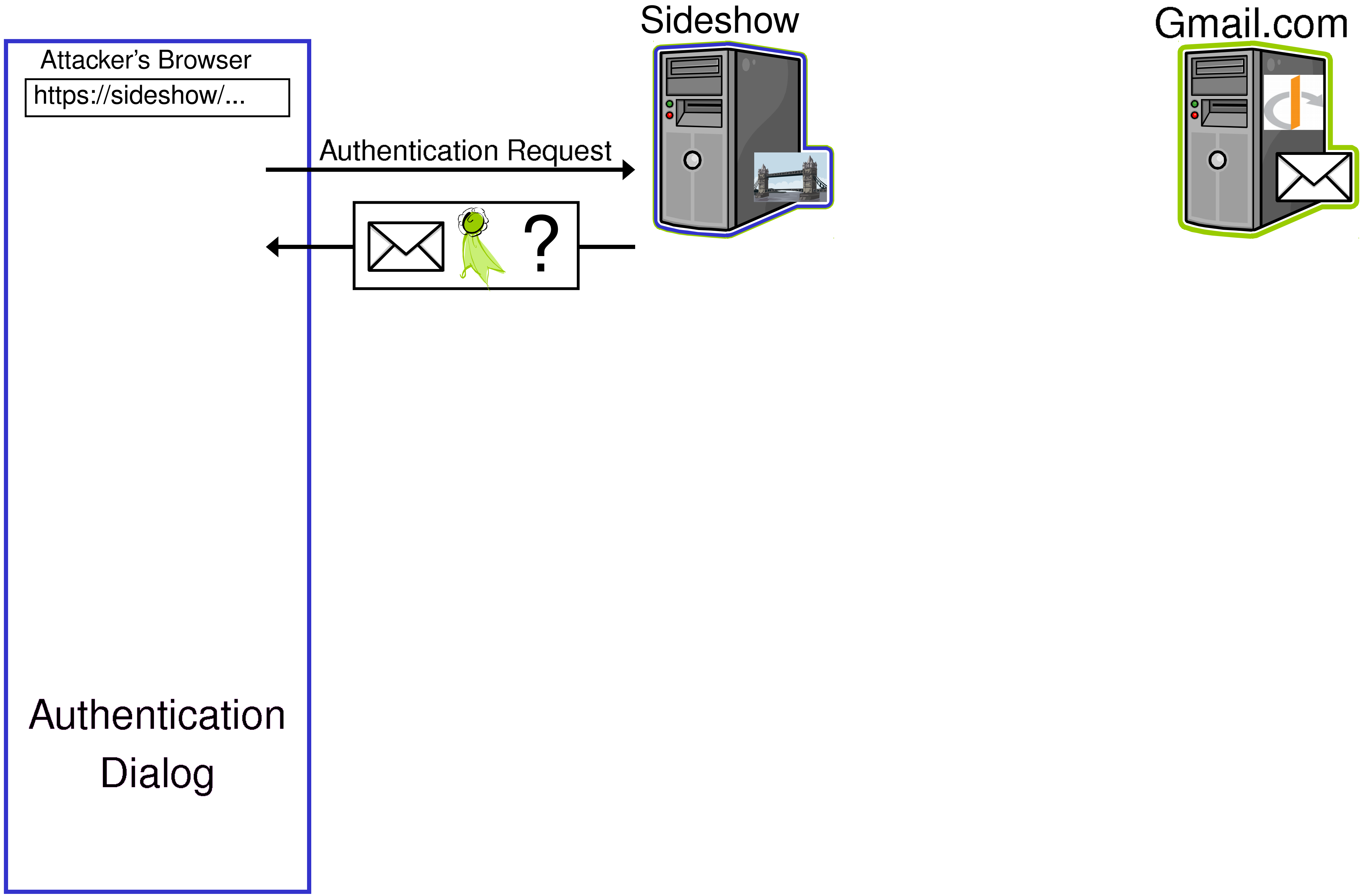


Gmail.com

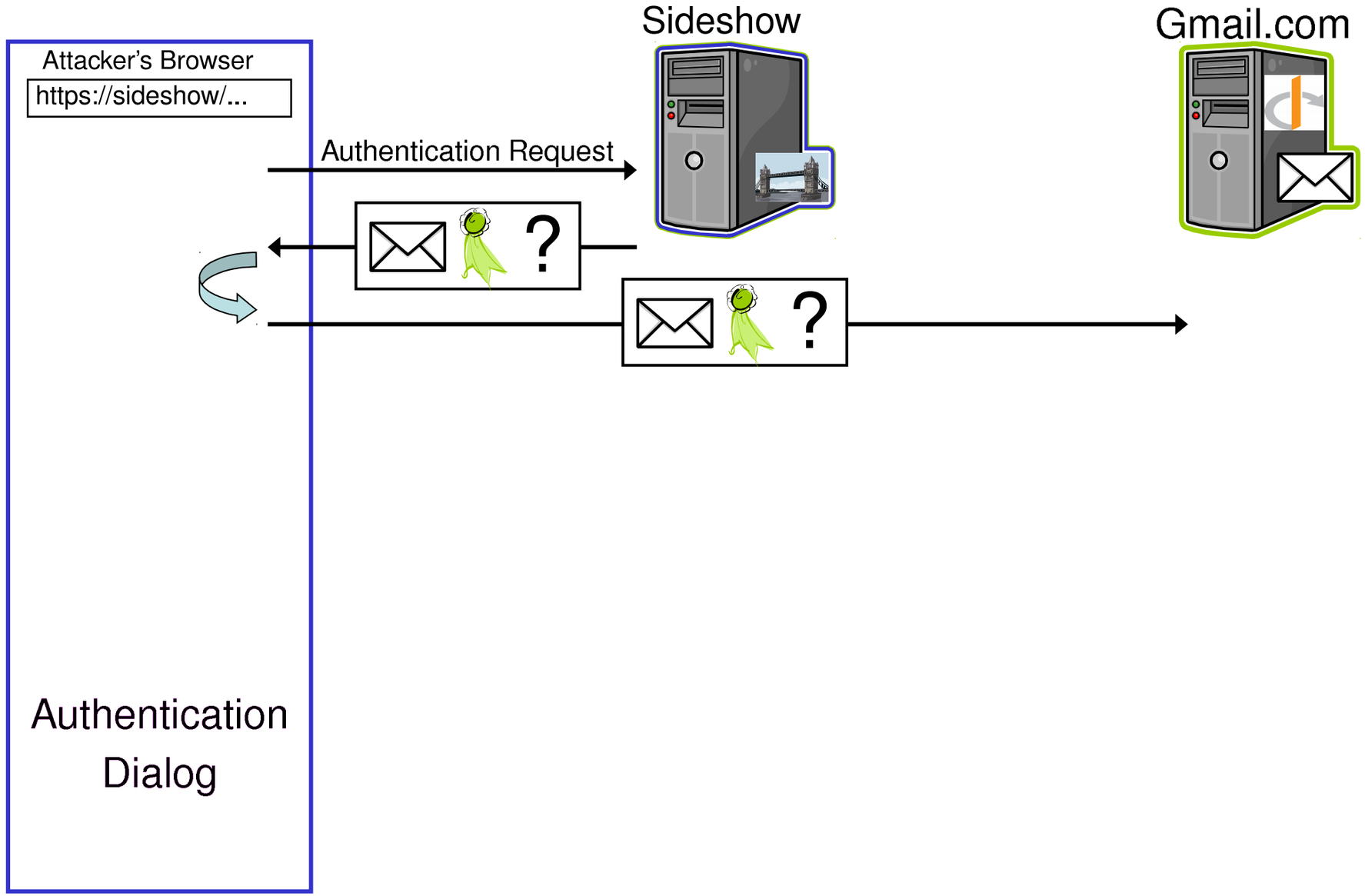
Identity Forgery (II)



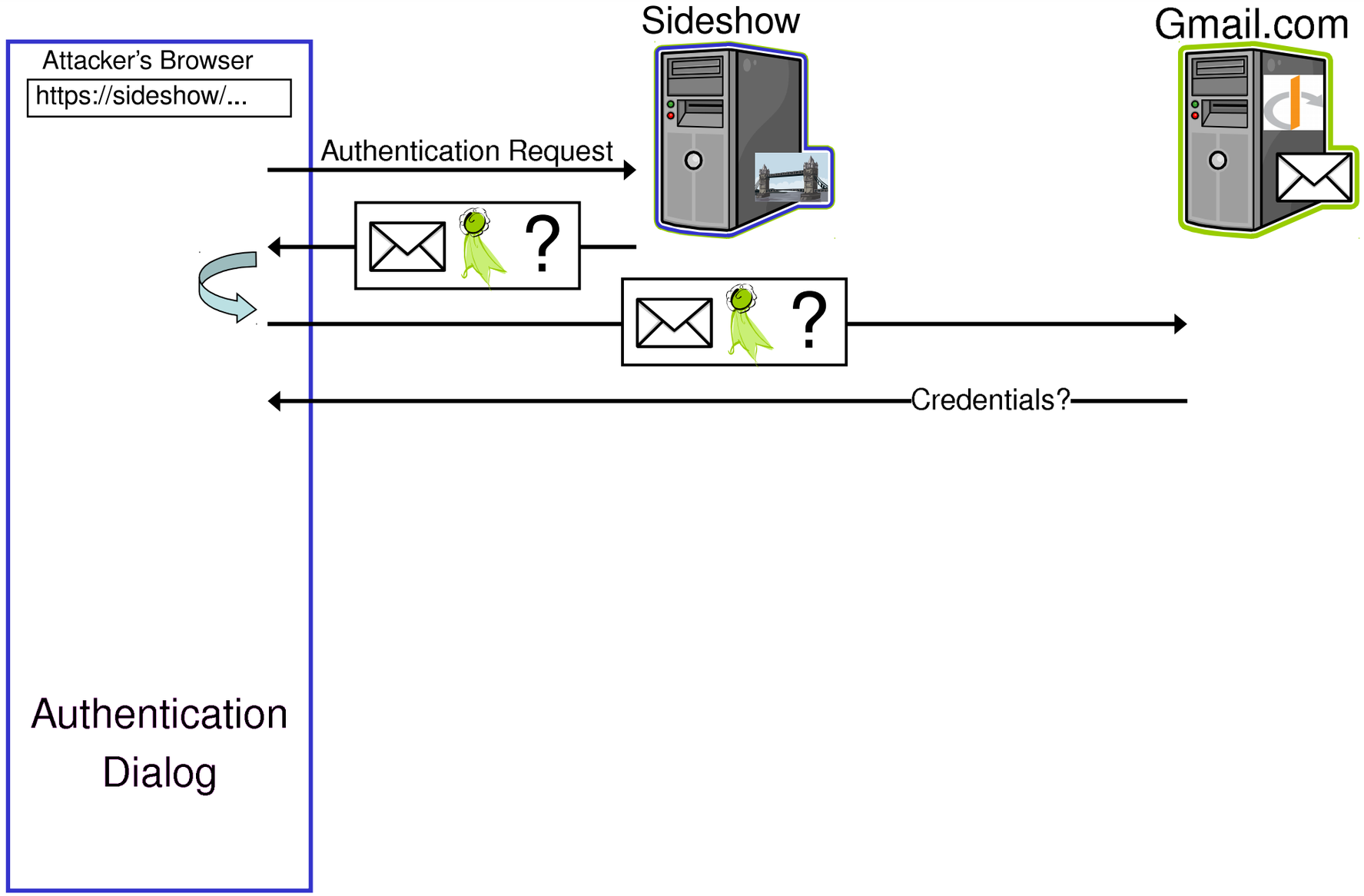
Identity Forgery (II)



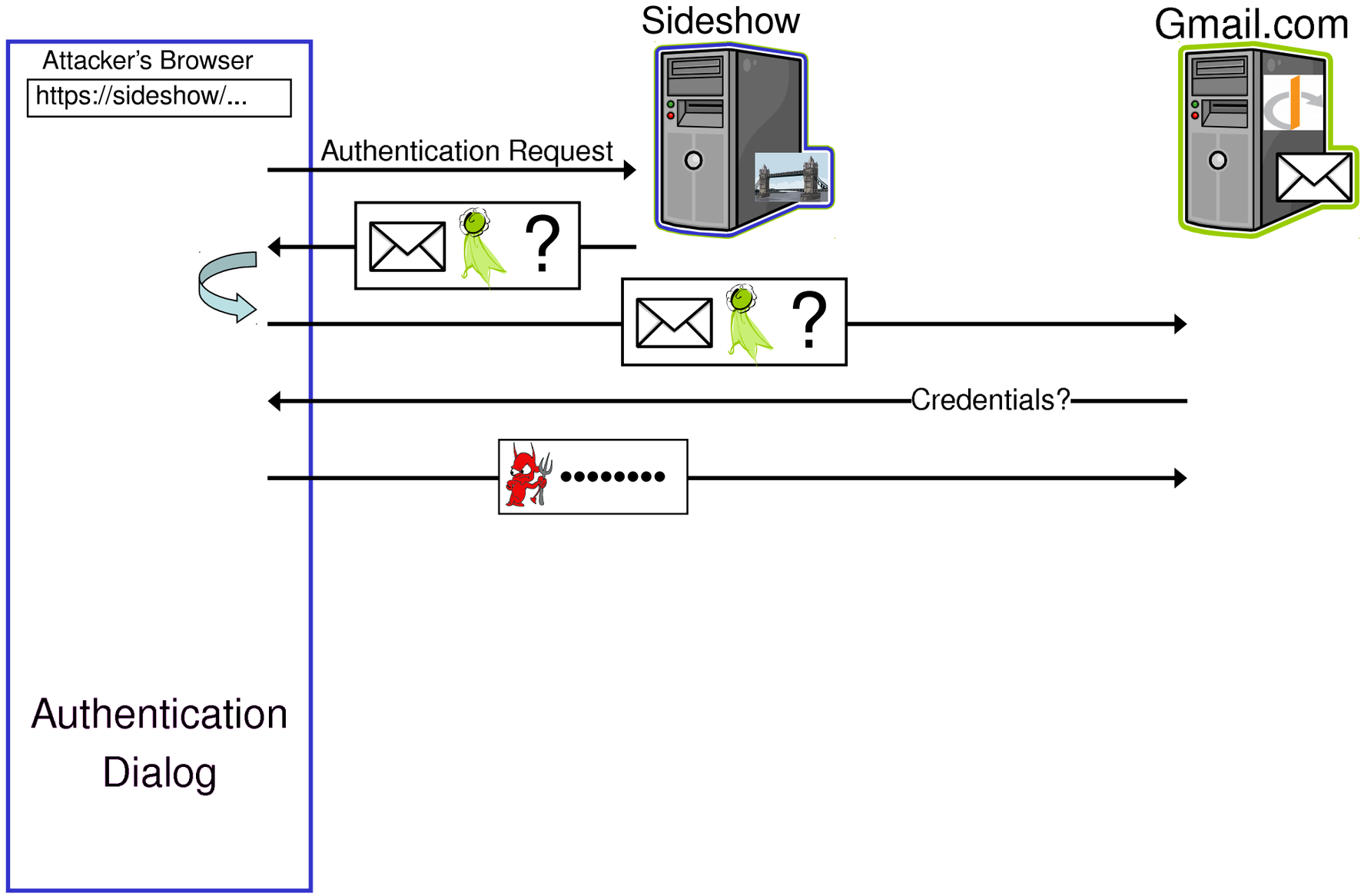
Identity Forgery (II)



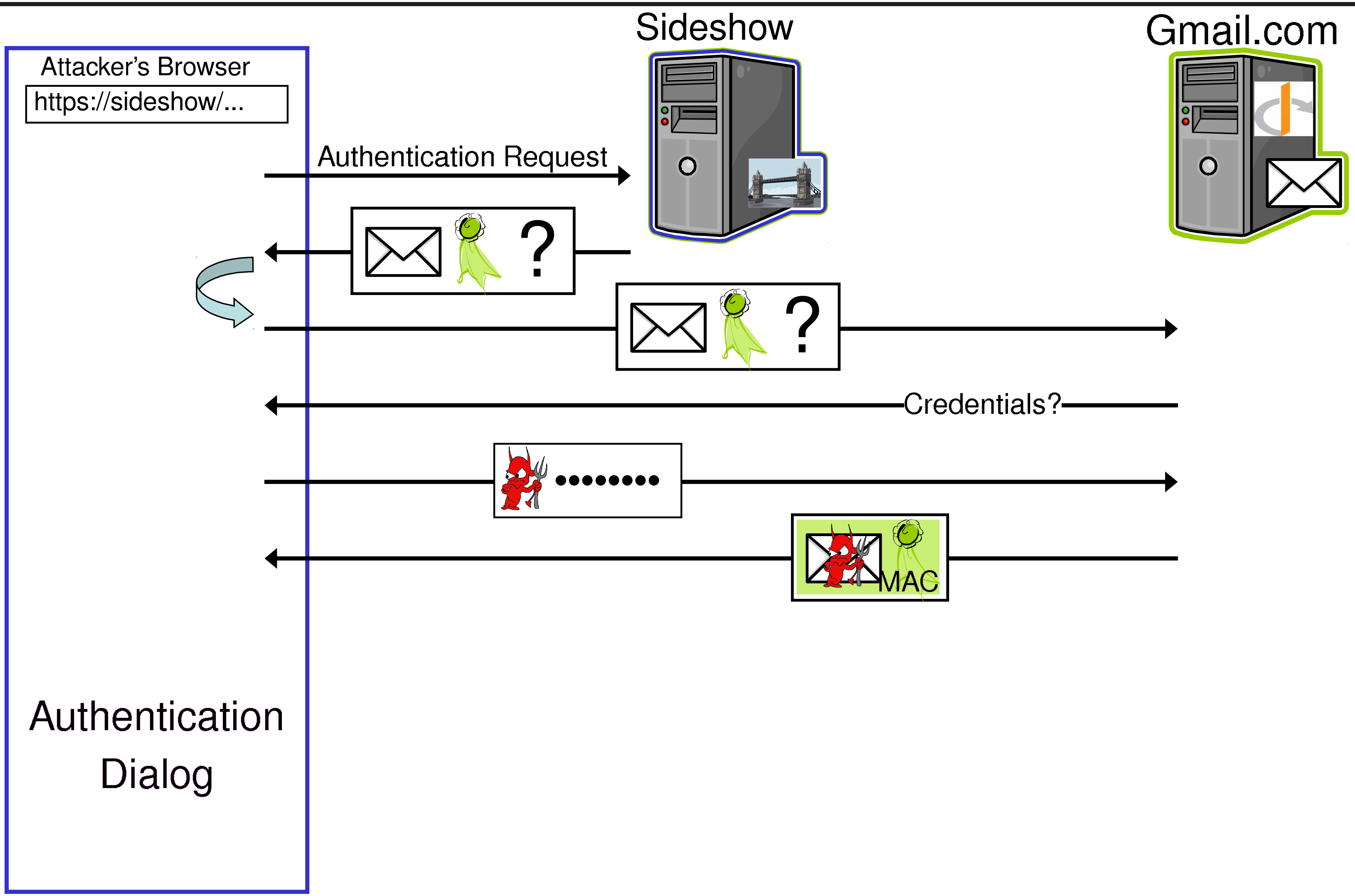
Identity Forgery (II)



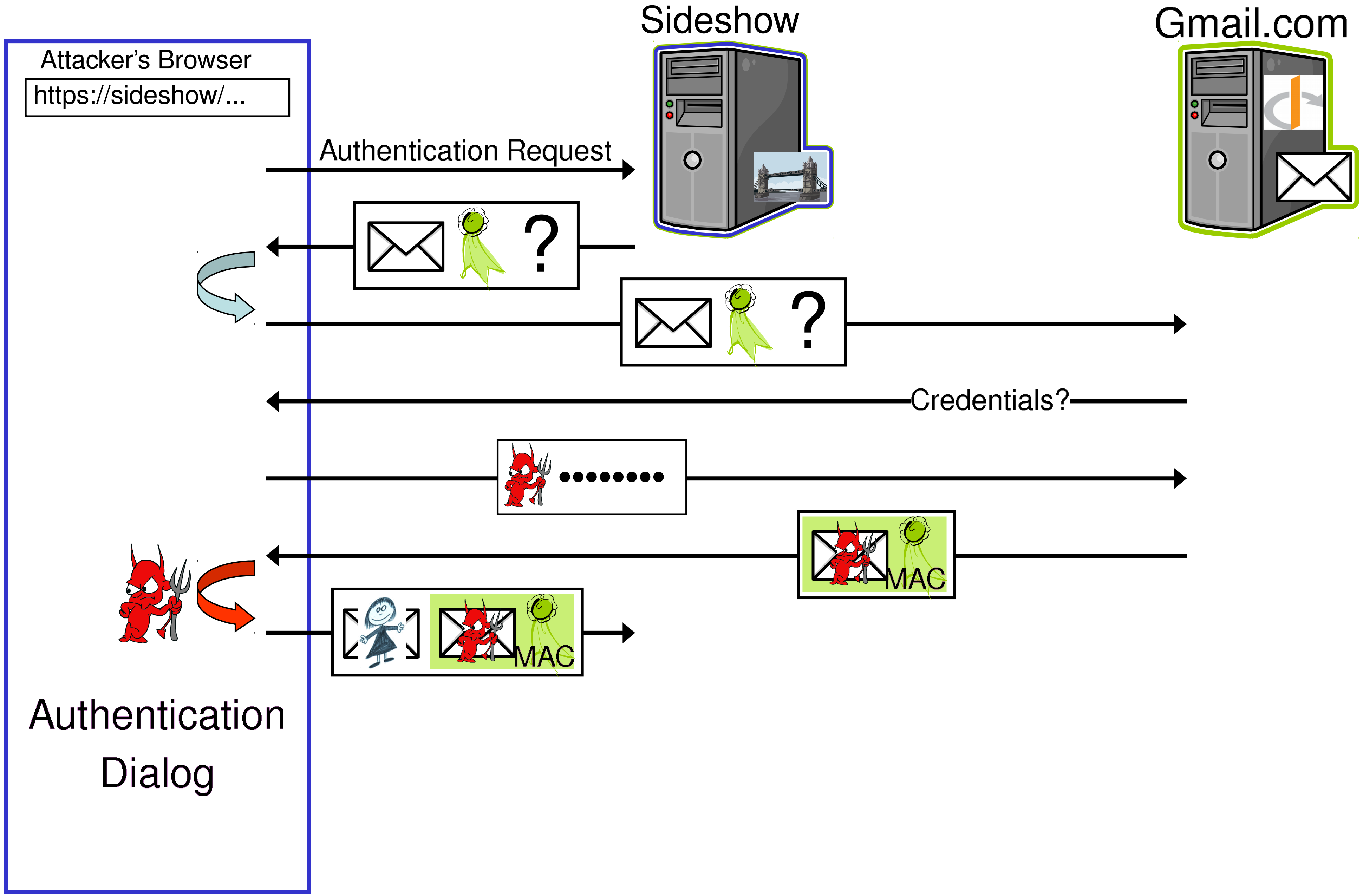
Identity Forgery (II)



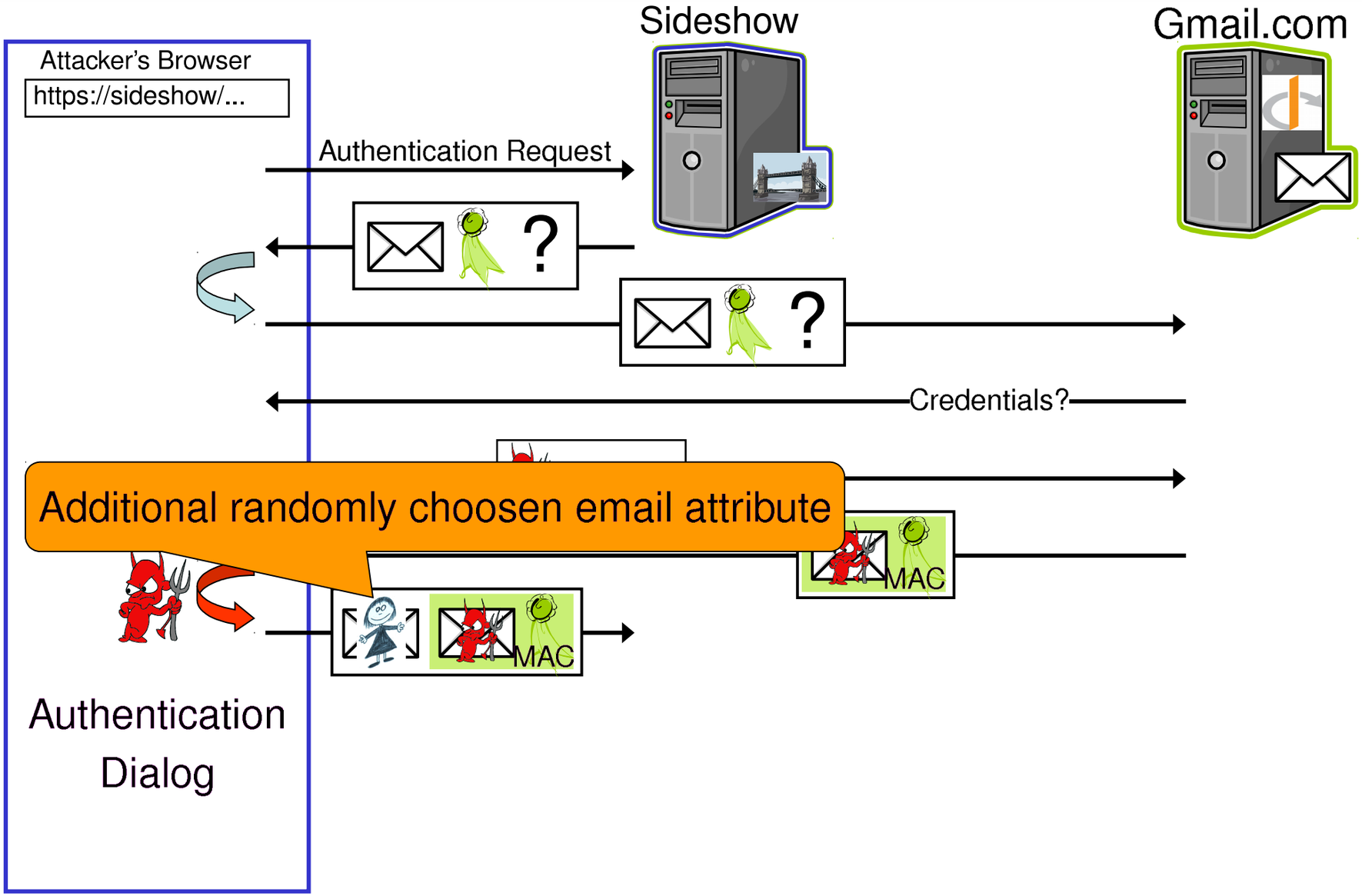
Identity Forgery (II)



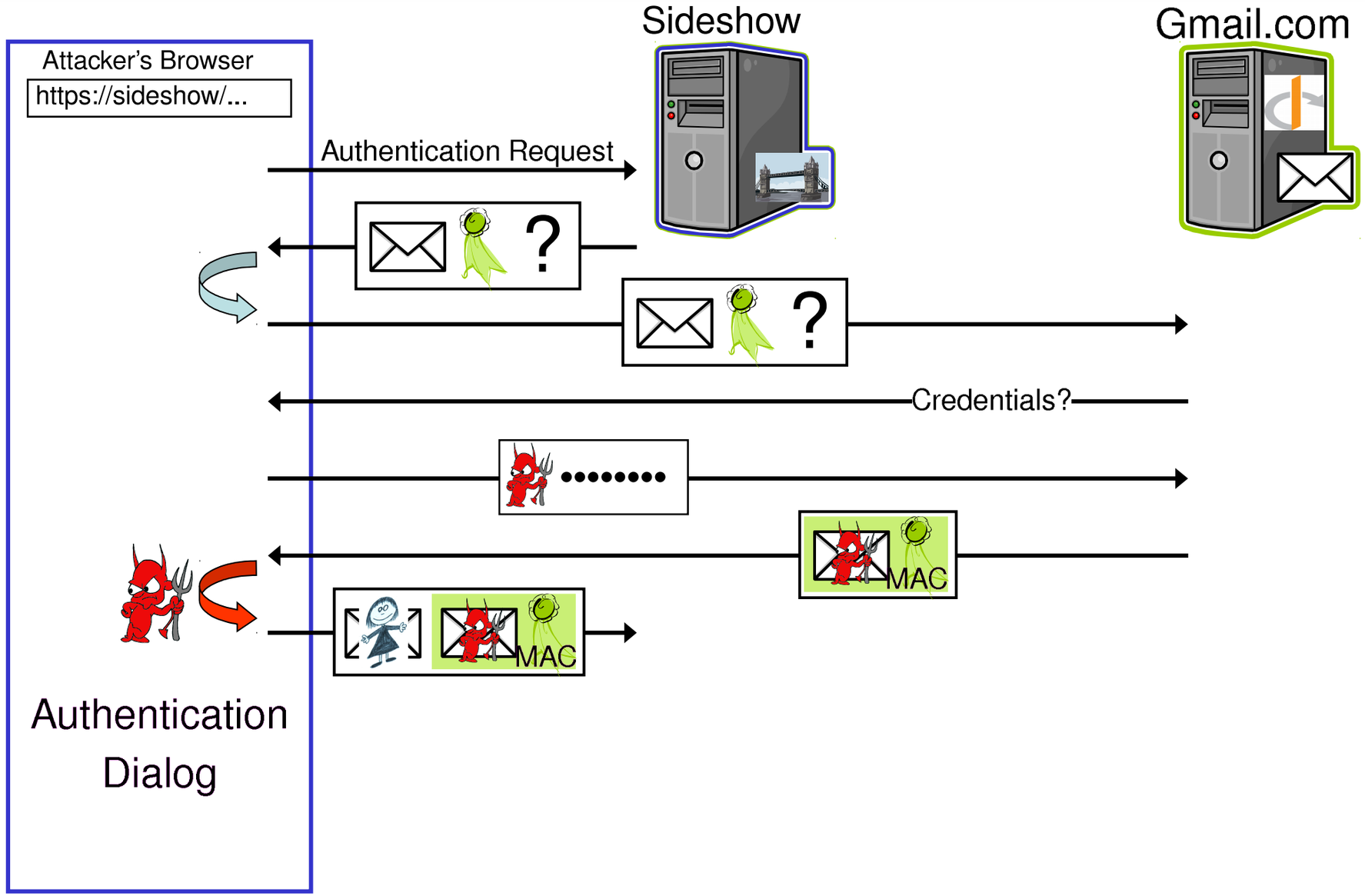
Identity Forgery (II)



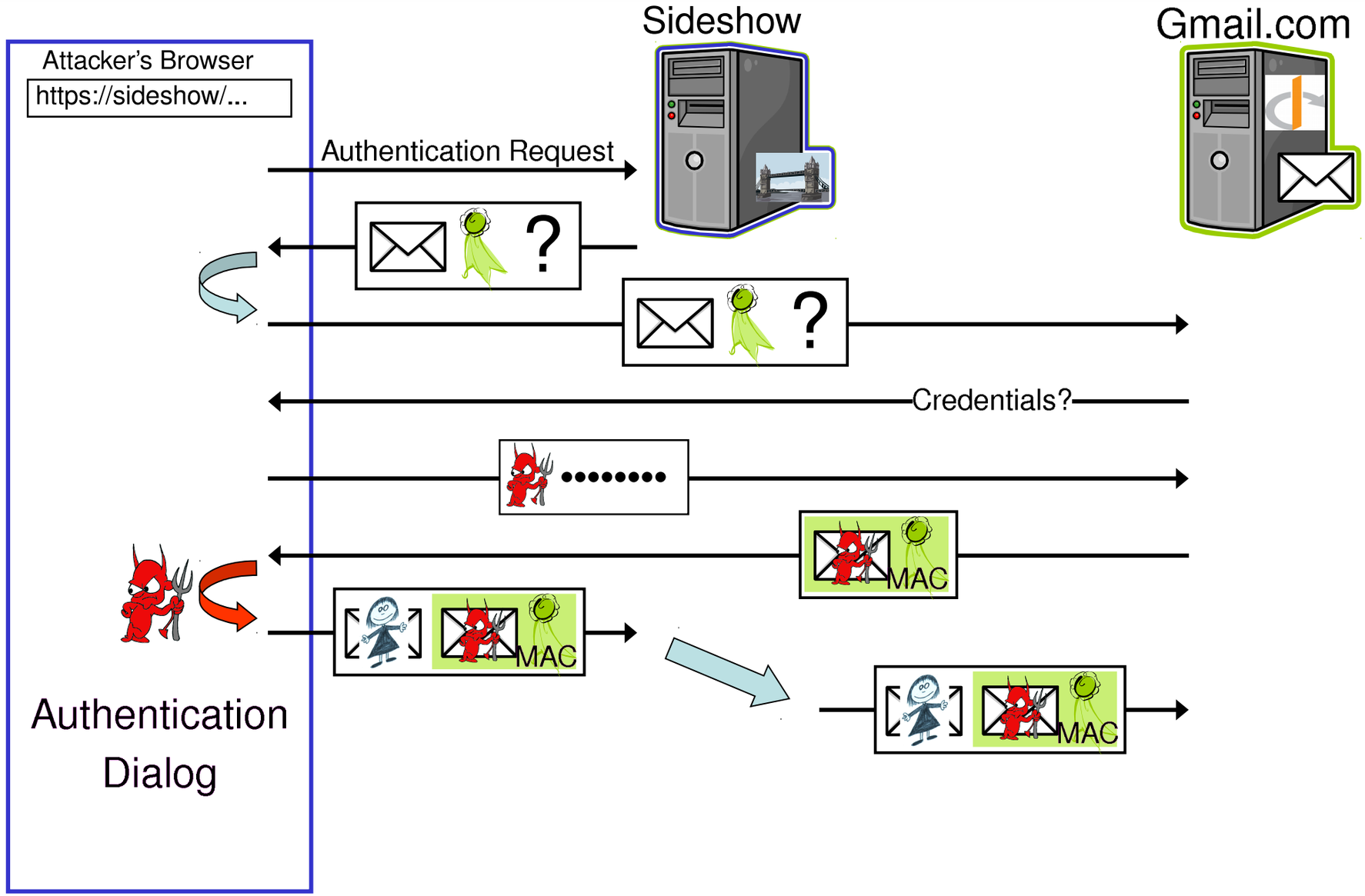
Identity Forgery (II)



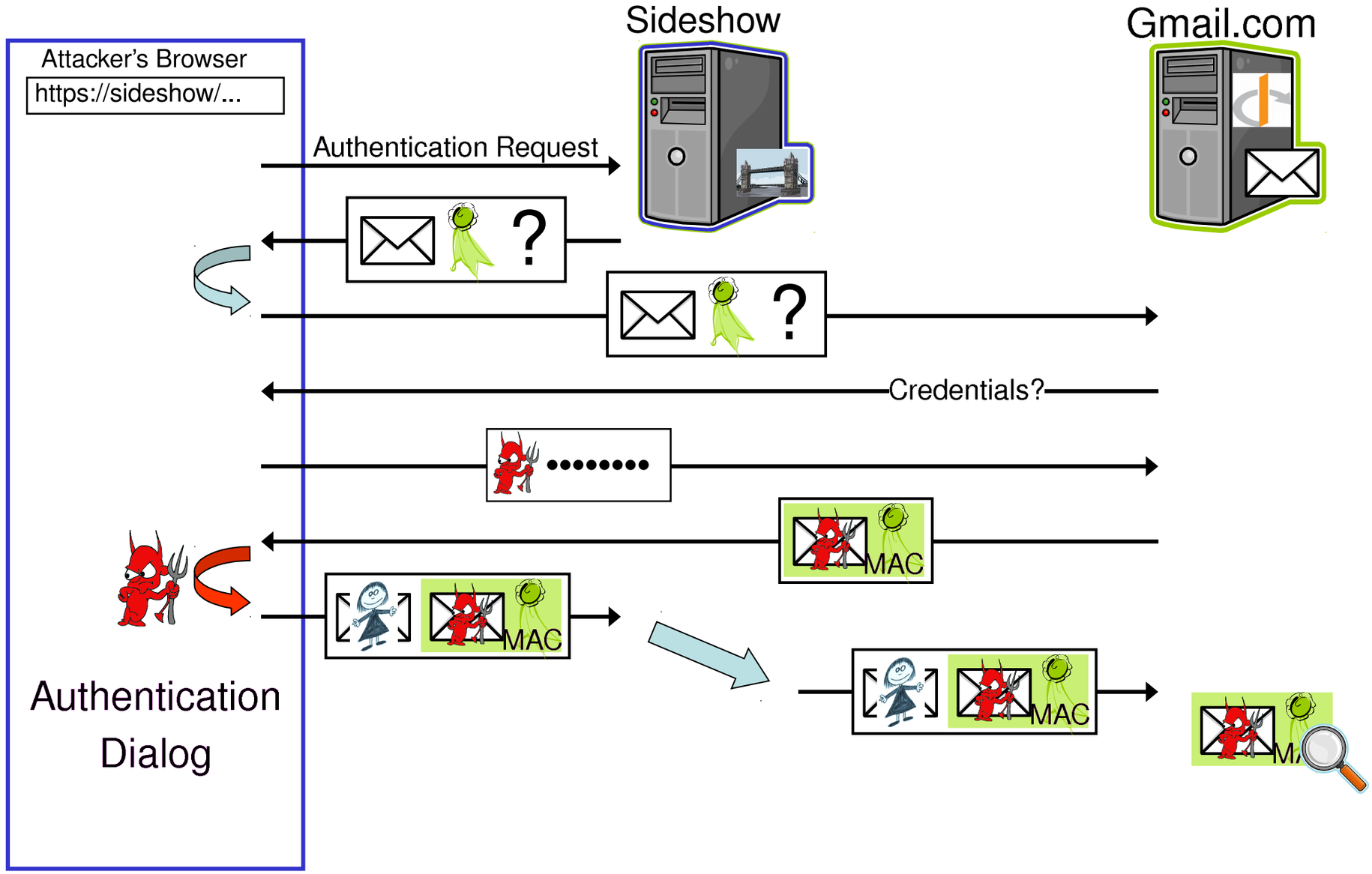
Identity Forgery (II)



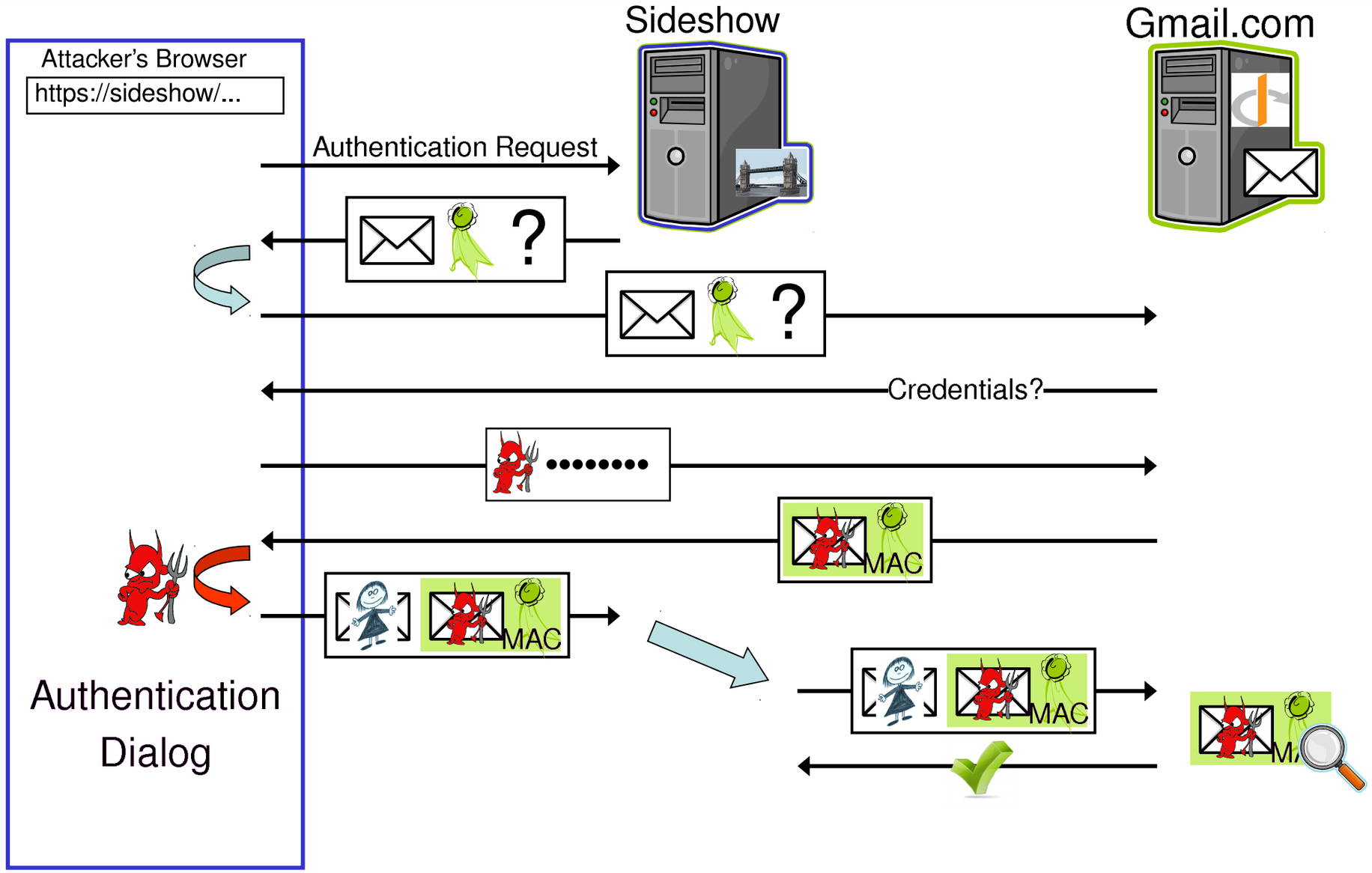
Identity Forgery (II)



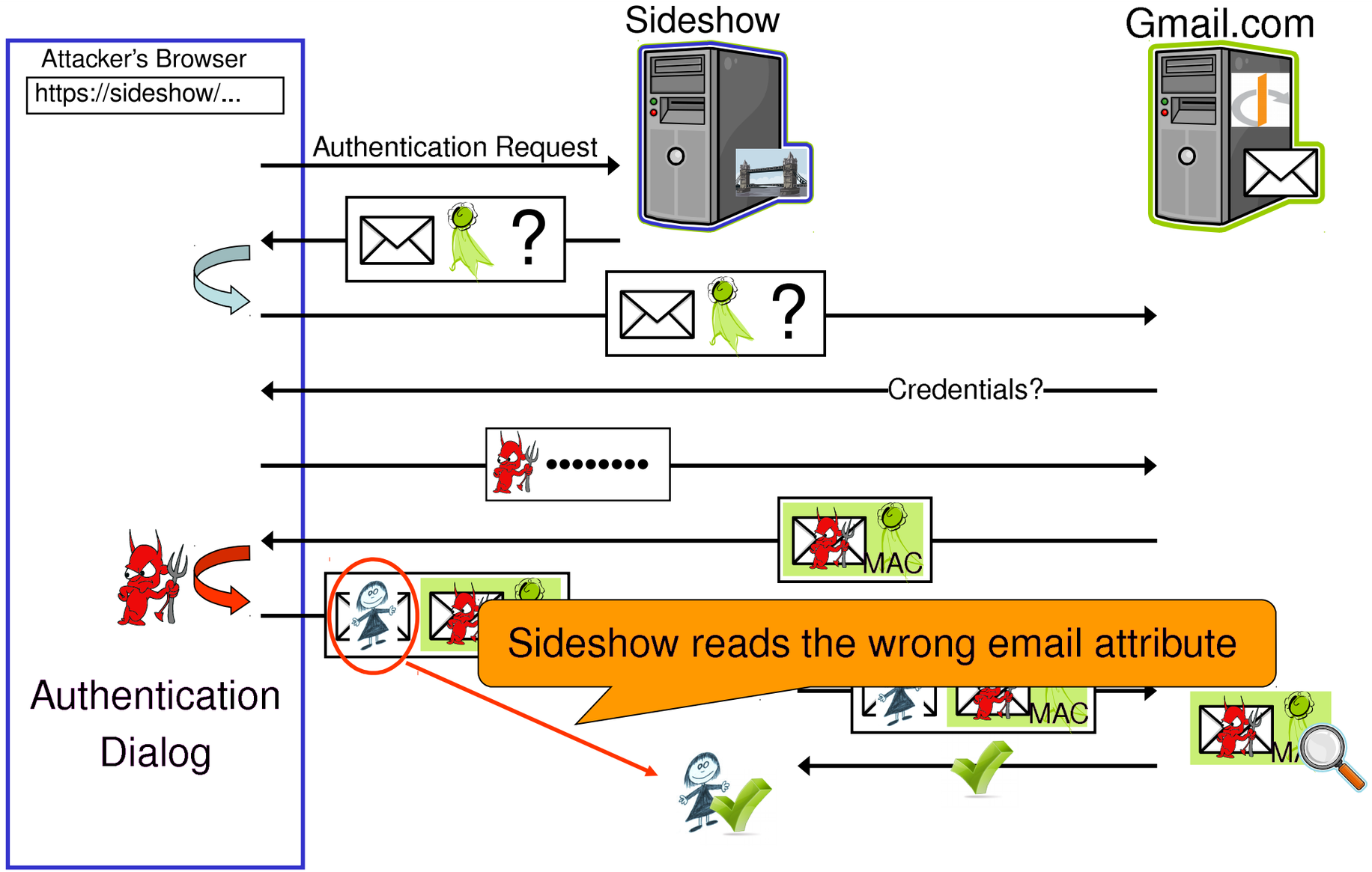
Identity Forgery (II)



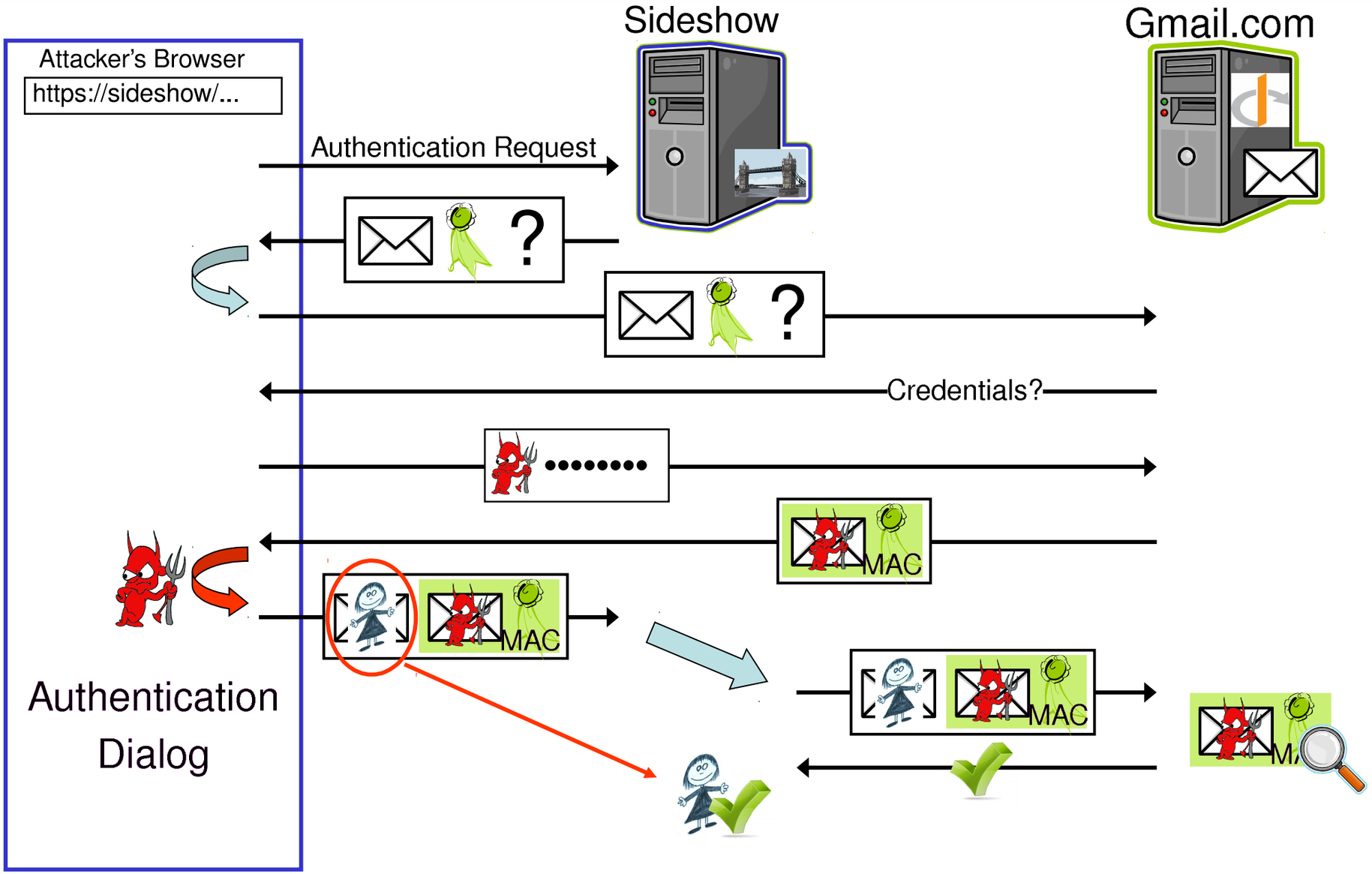
Identity Forgery (II)



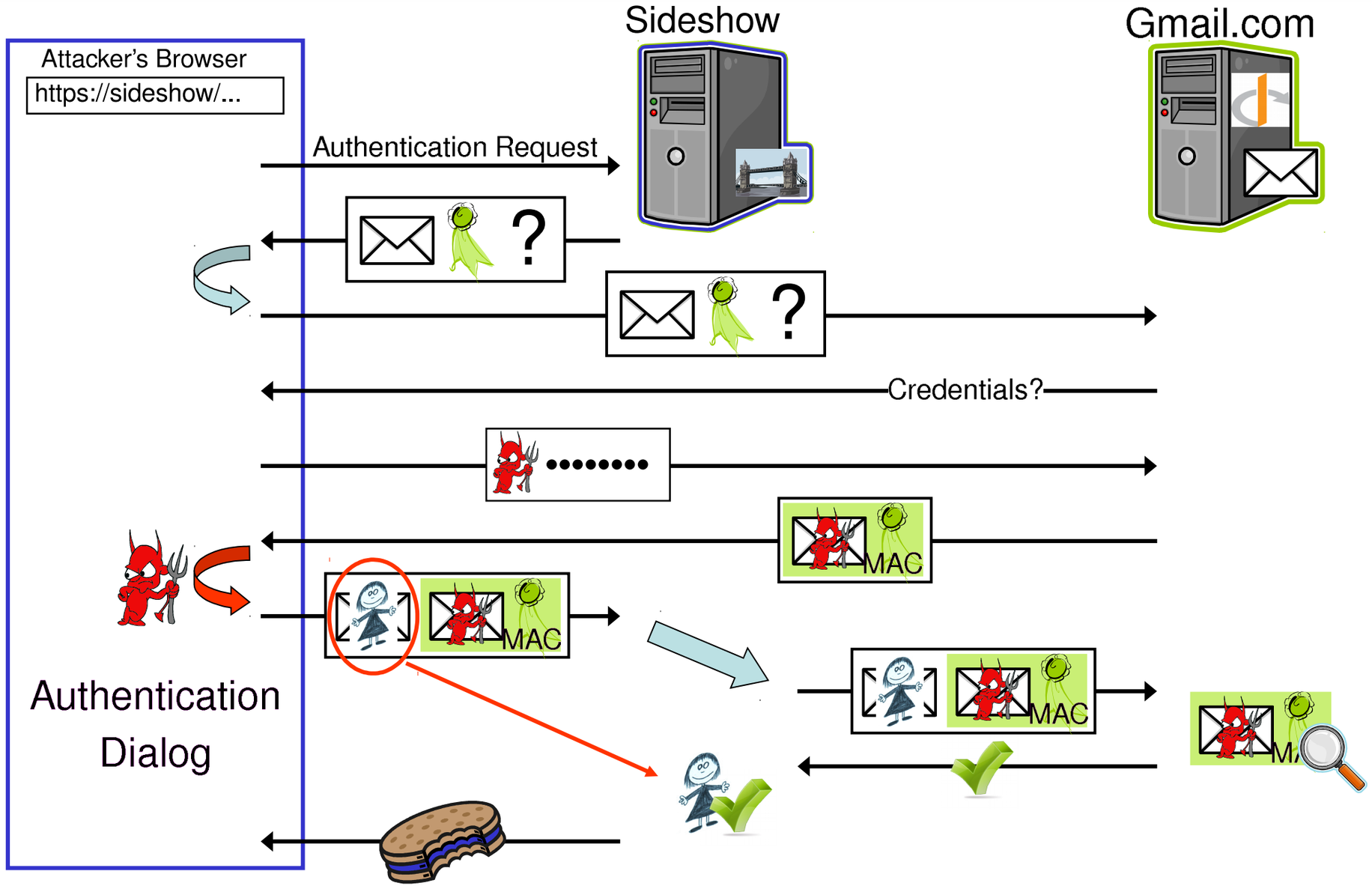
Identity Forgery (II)



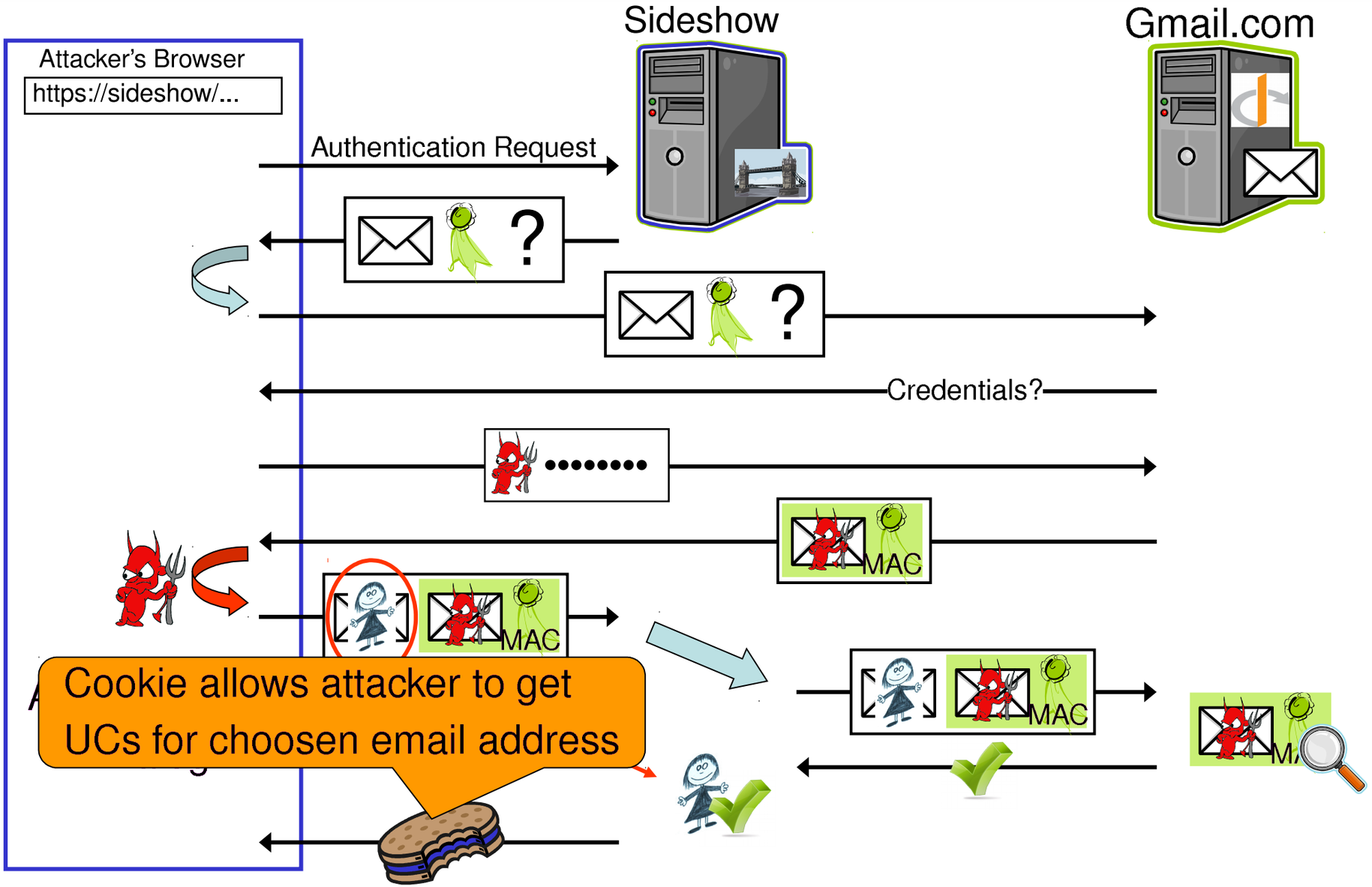
Identity Forgery (II)



Identity Forgery (II)



Identity Forgery (II)



Privacy

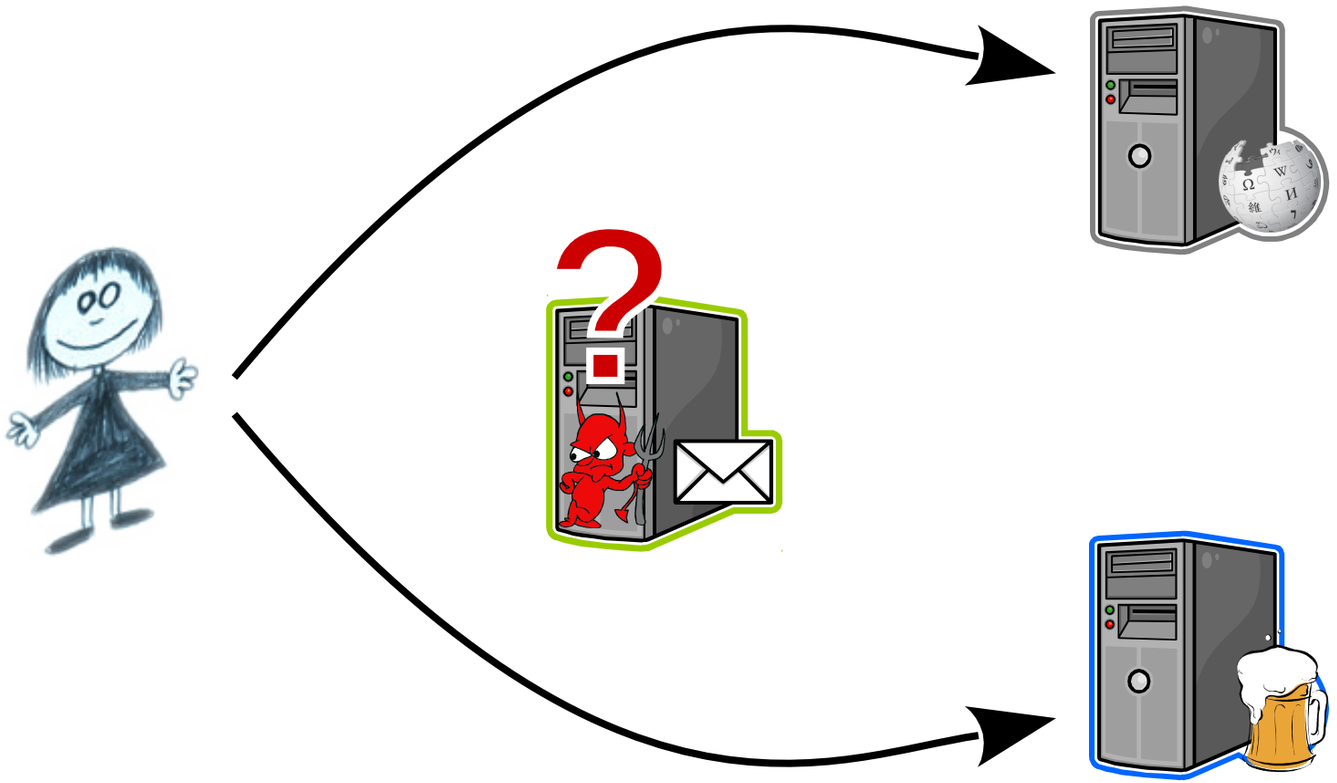
Unique claim:

"… the BrowserID protocol **never leaks** tracking information back **to the Identity Provider**." - Mozilla Persona FAQ

Privacy

Unique claim:

"… the BrowserID protocol **never leaks** tracking information back **to the Identity Provider**." - Mozilla Persona FAQ

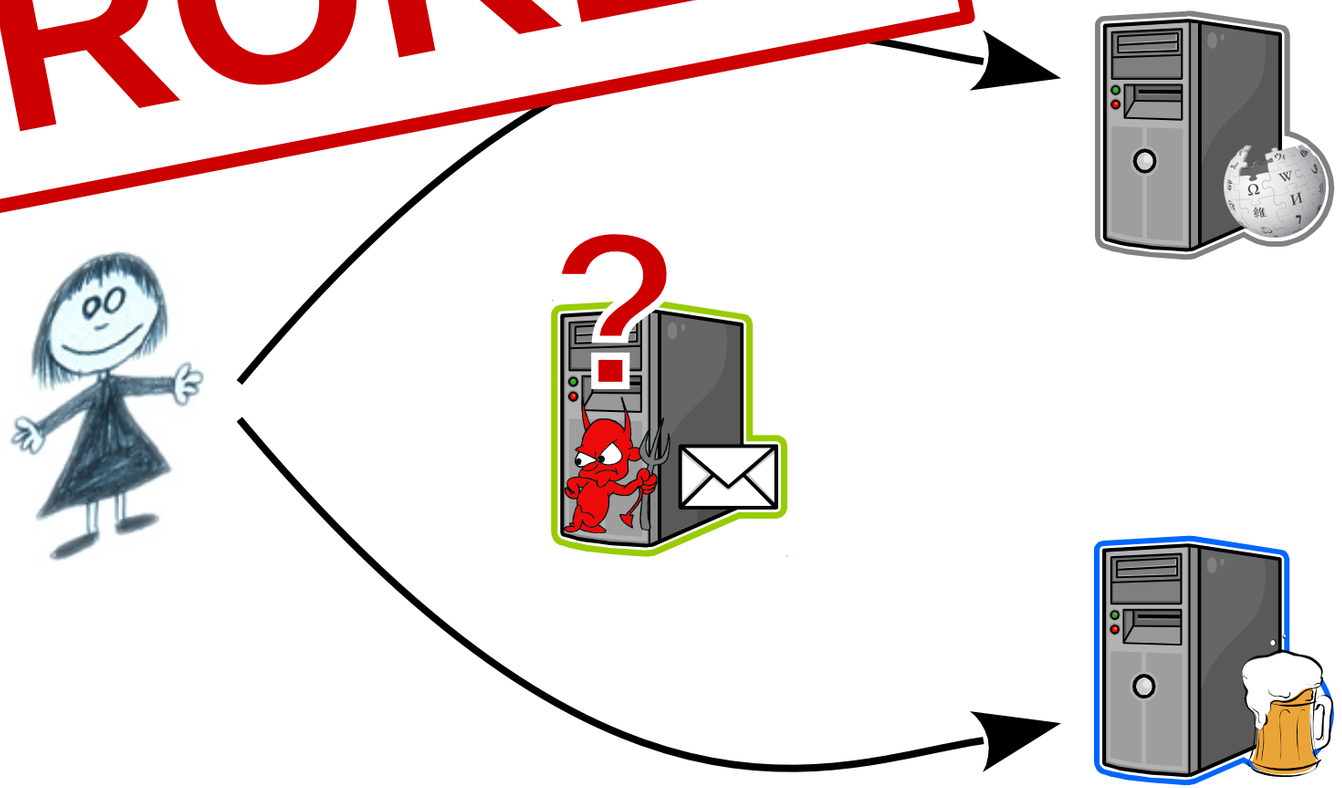


Privacy

Unique claim:

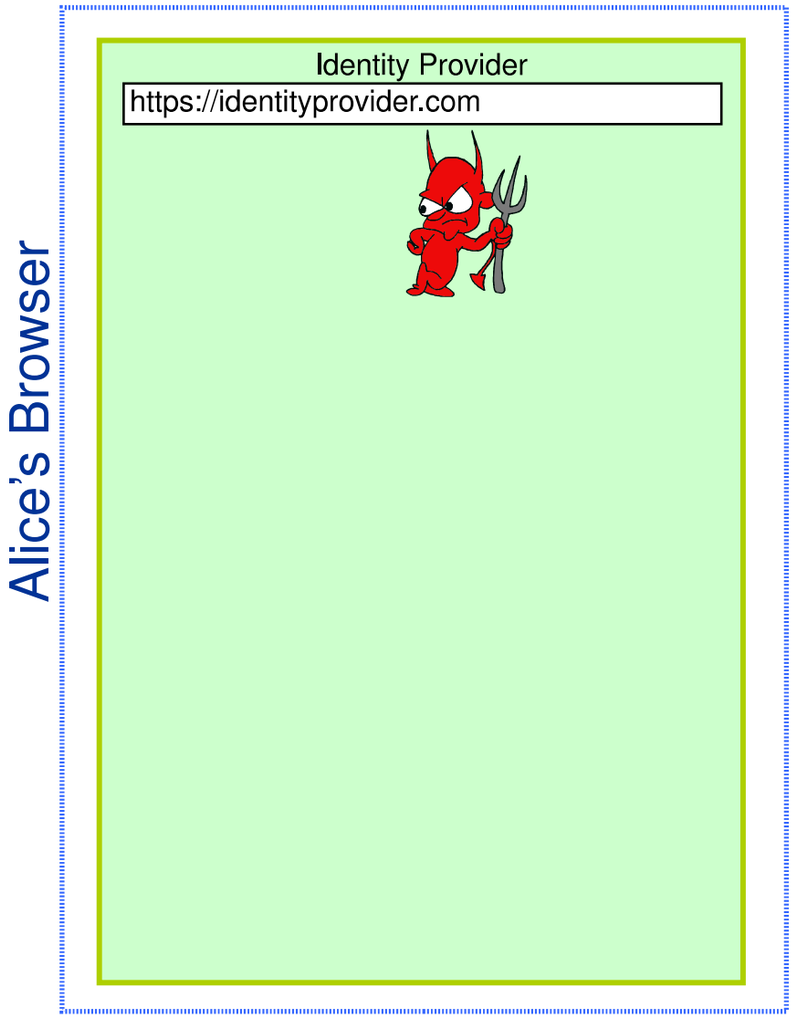
"… the BrowserID protocol **never le** back **to the Identity Provider**."

BROKEN



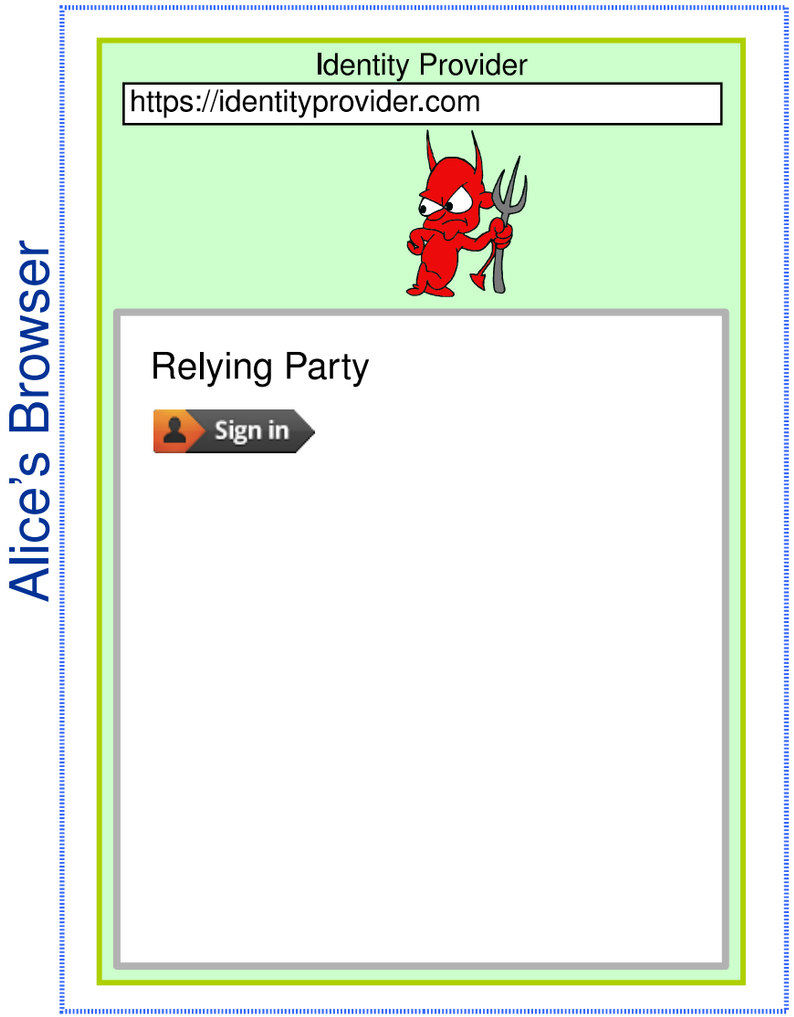
Privacy Attacks

Information is leaked by the **window structure** in the user's browser:



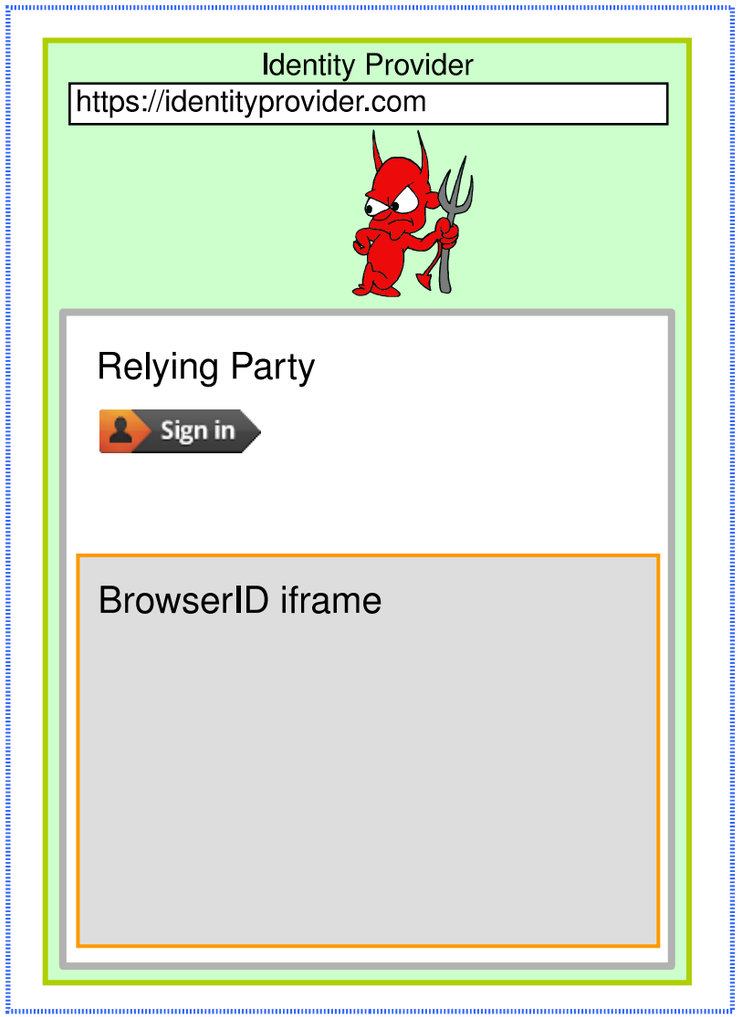
Privacy Attacks

Information is leaked by the **window structure** in the user's browser:



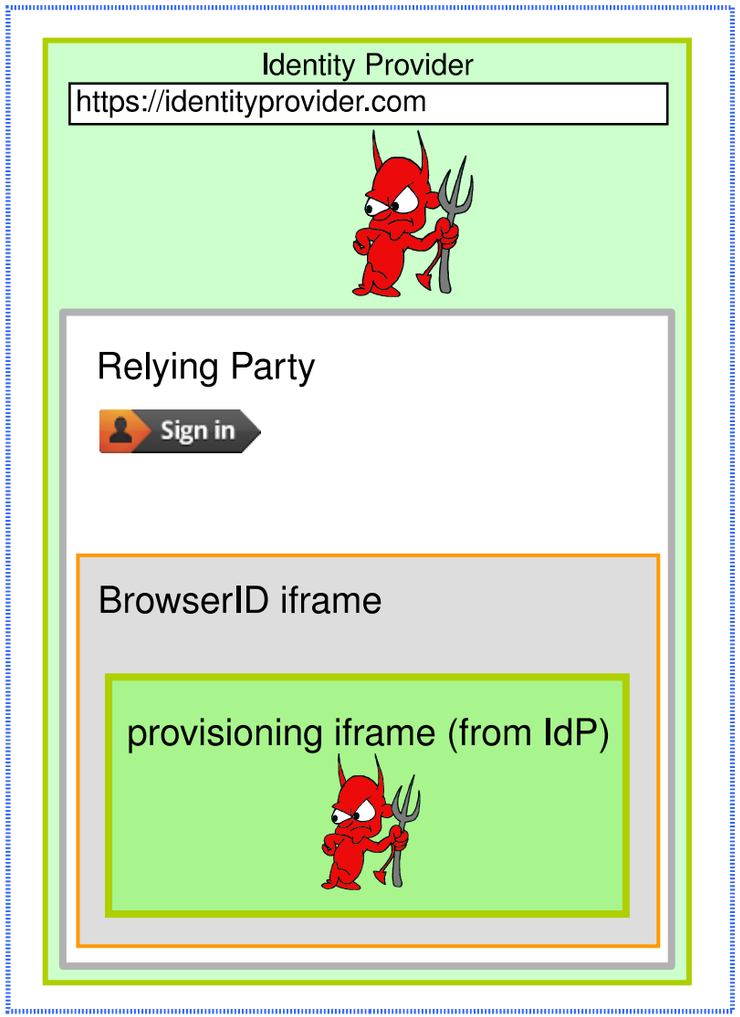
Privacy Attacks

Information is leaked by the **window structure** in the user's browser:



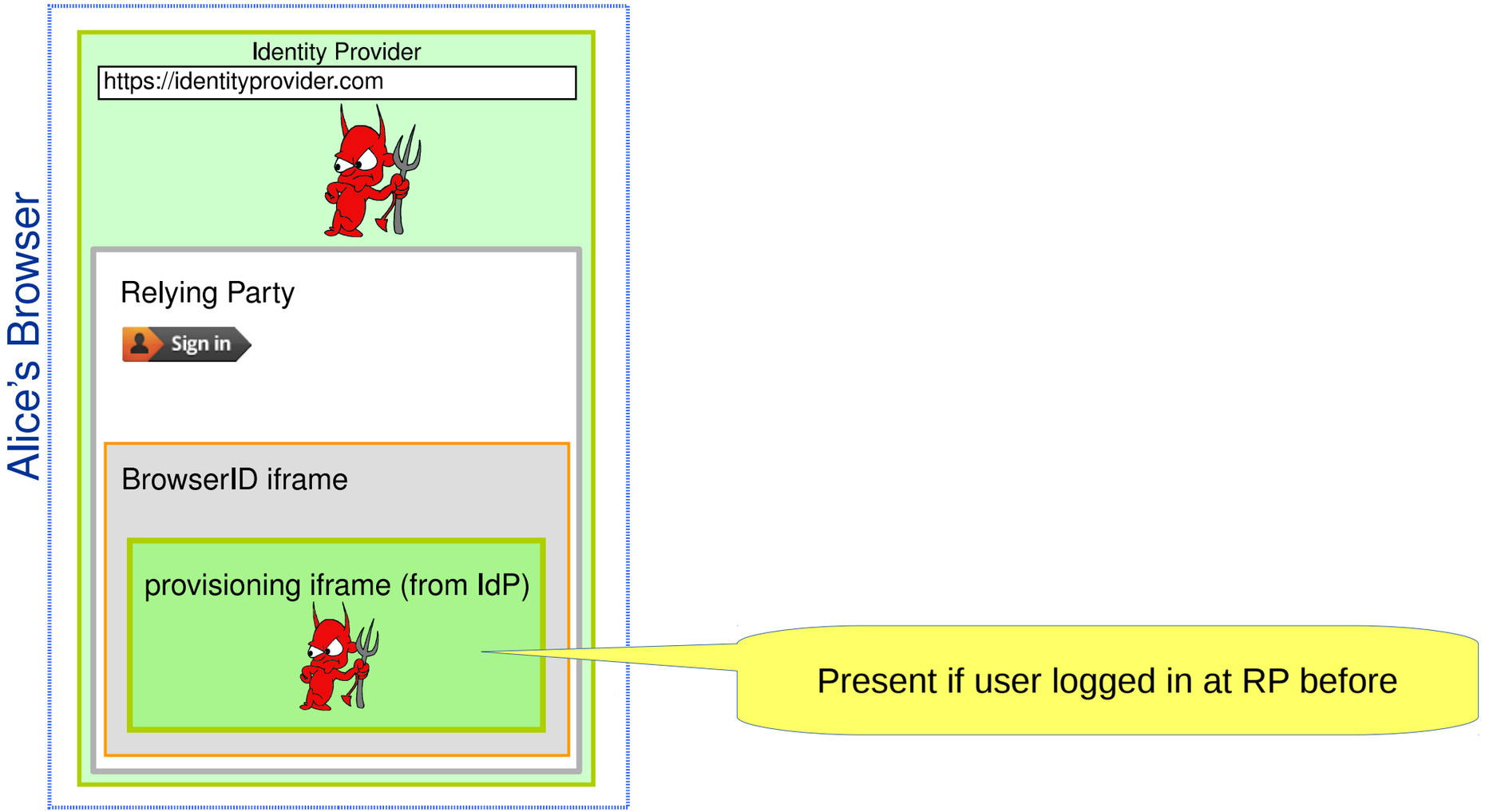
Privacy Attacks

Information is leaked by the **window structure** in the user's browser:



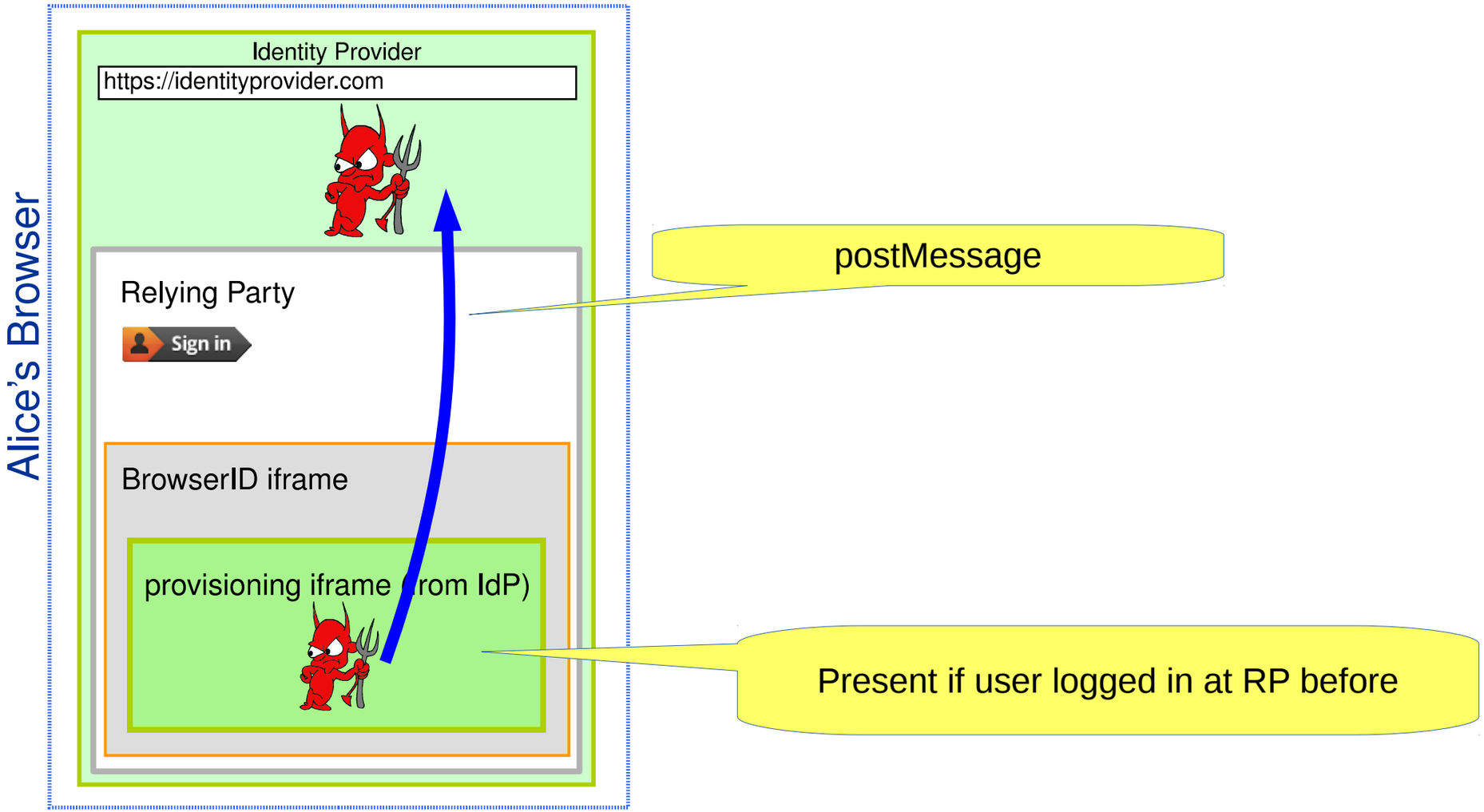
Privacy Attacks

Information is leaked by the **window structure** in the user's browser:



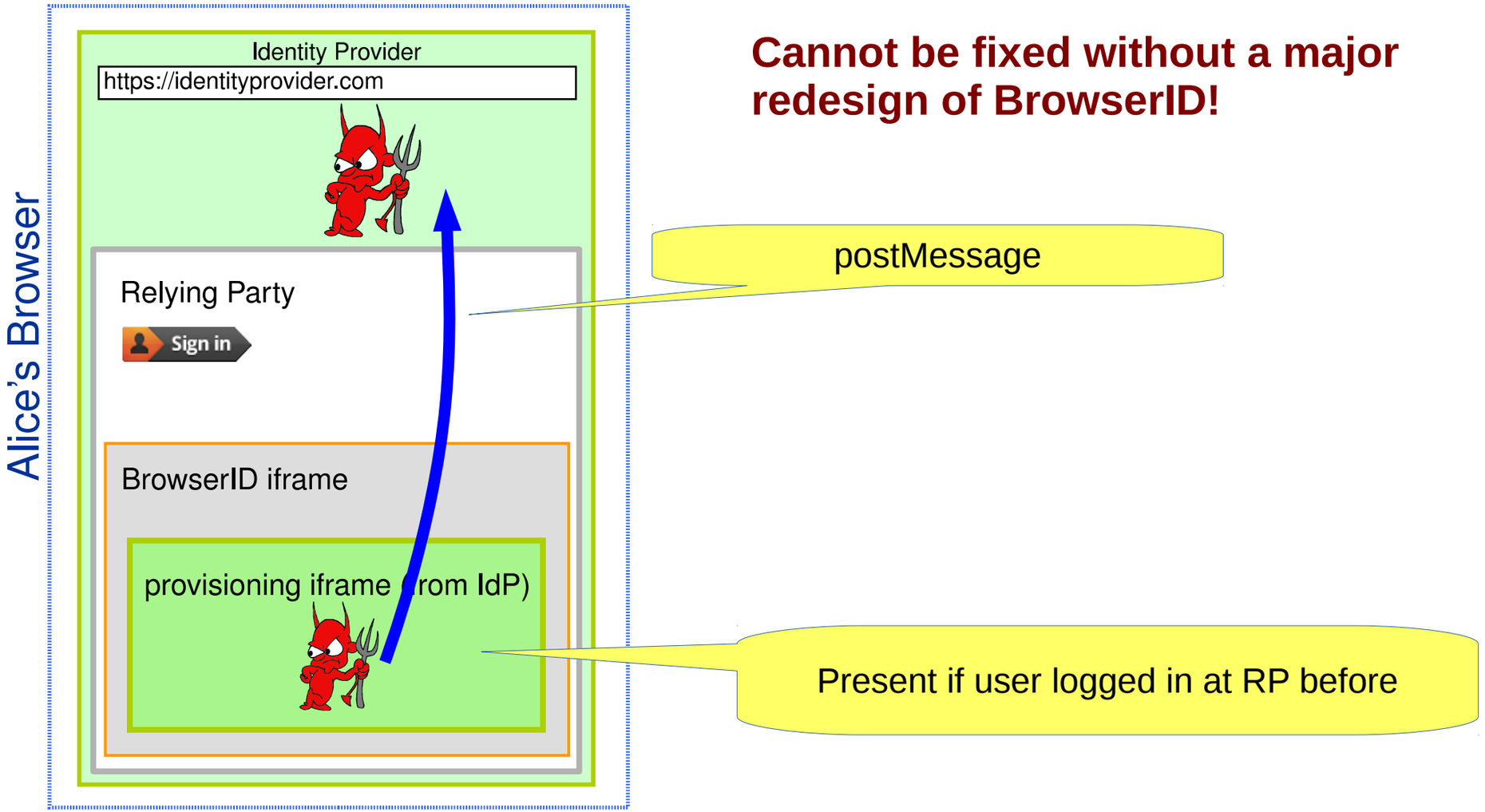
Privacy Attacks

Information is leaked by the **window structure** in the user's browser:



Privacy Attacks

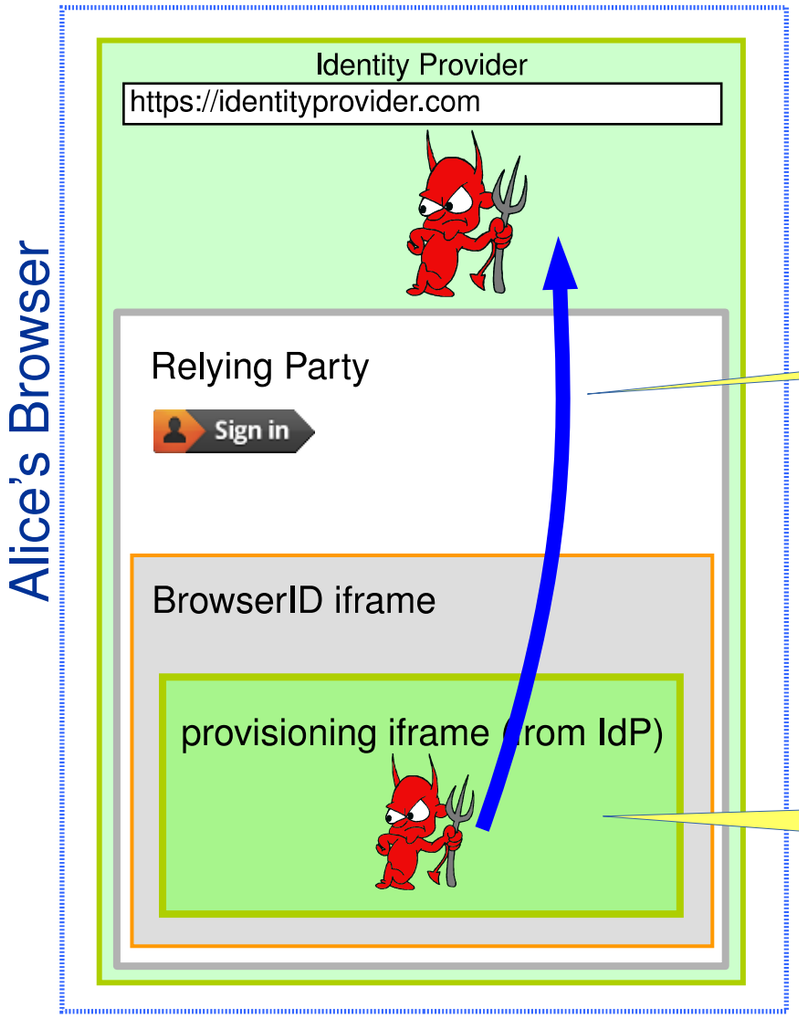
Information is leaked by the **window structure** in the user's browser:



Cannot be fixed without a major redesign of BrowserID!

Privacy Attacks

Information is leaked by the **window structure** in the user's browser:



Cannot be fixed without a major redesign of BrowserID!

postMessage

Present if user logged in at RP before

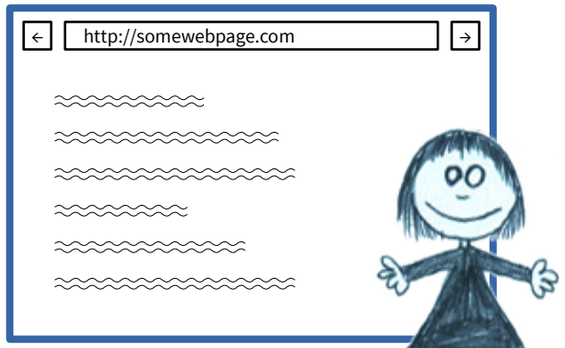
+ Variants

BrowserID

Security	**attack + fix** **formally proven** 
Privacy	

Can we build an SSO that provides security and privacy?

SPRESSO: Basic Principle

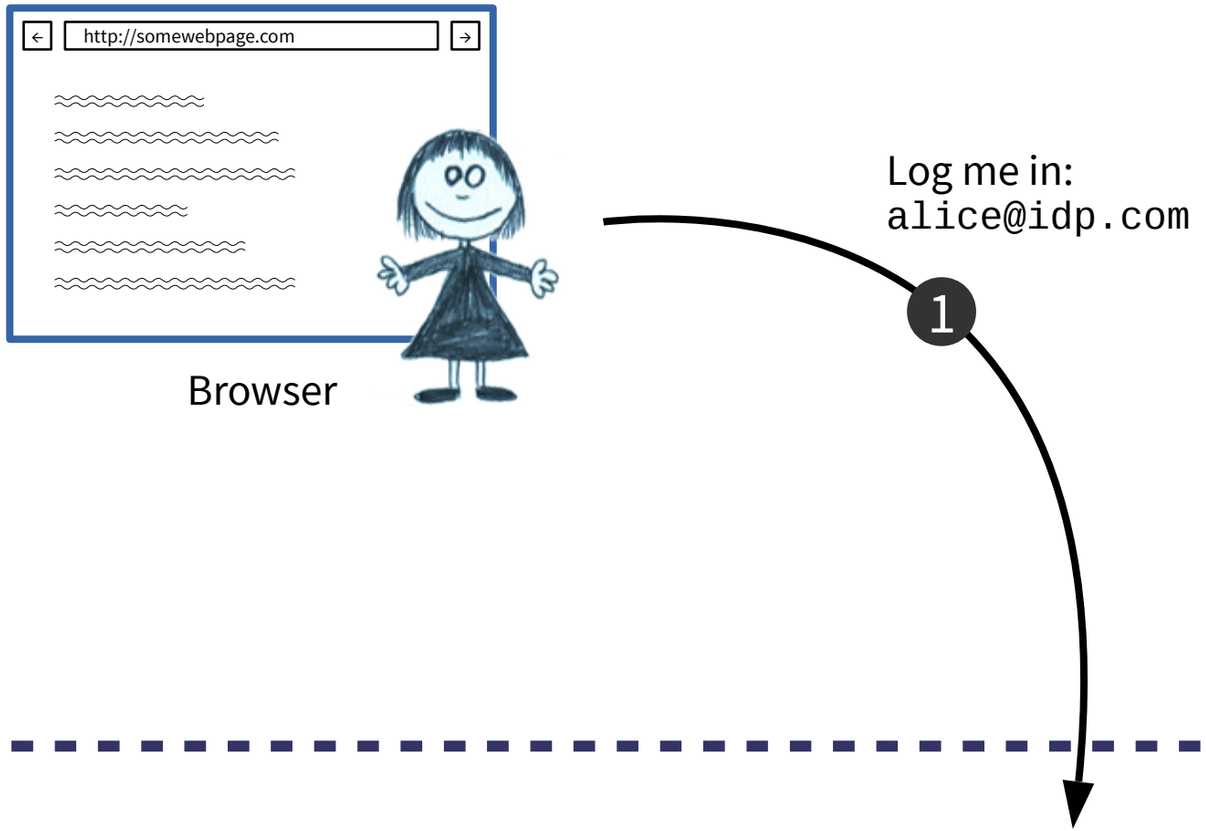


Browser

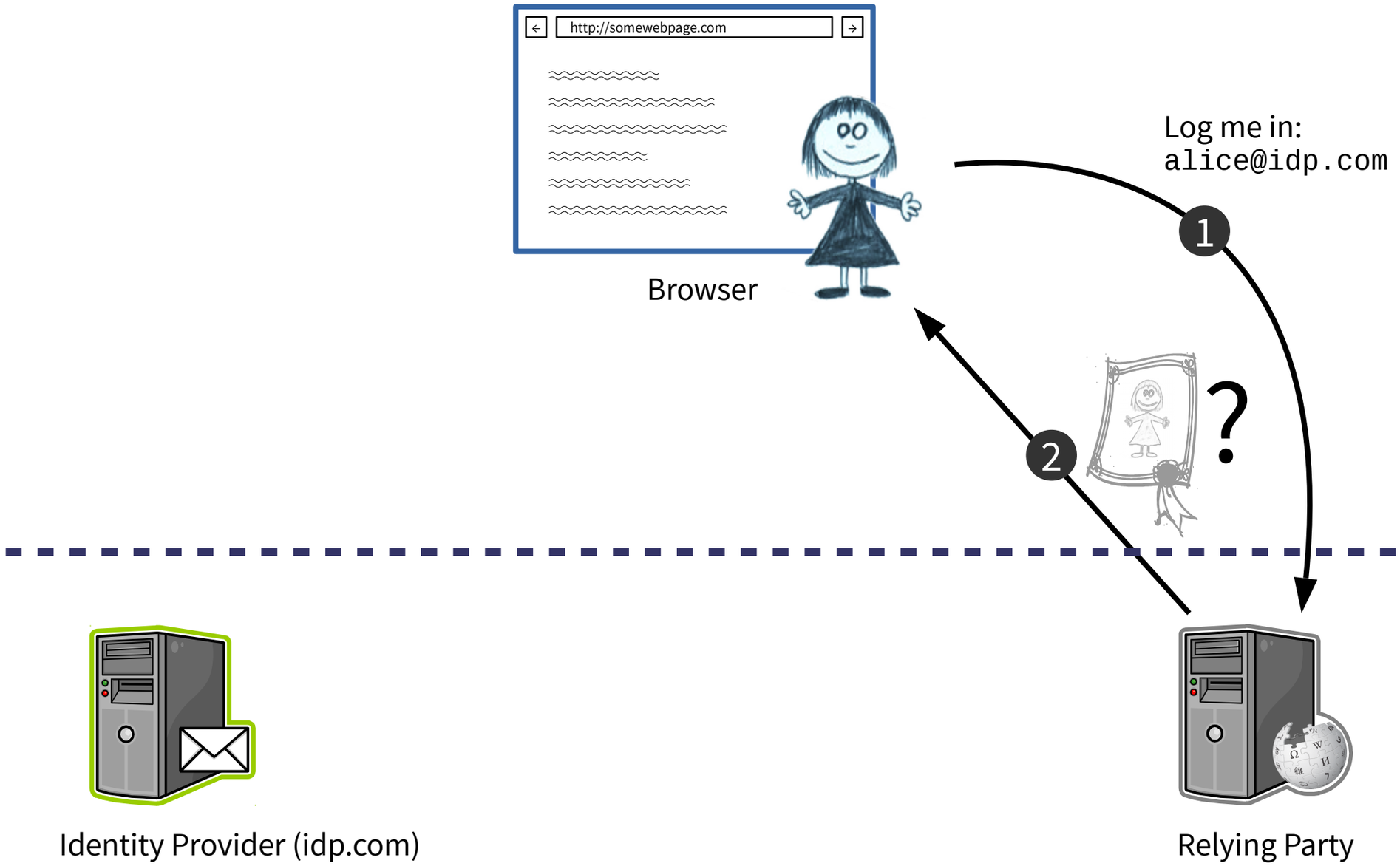
Identity Provider (idp.com)

Relying Party

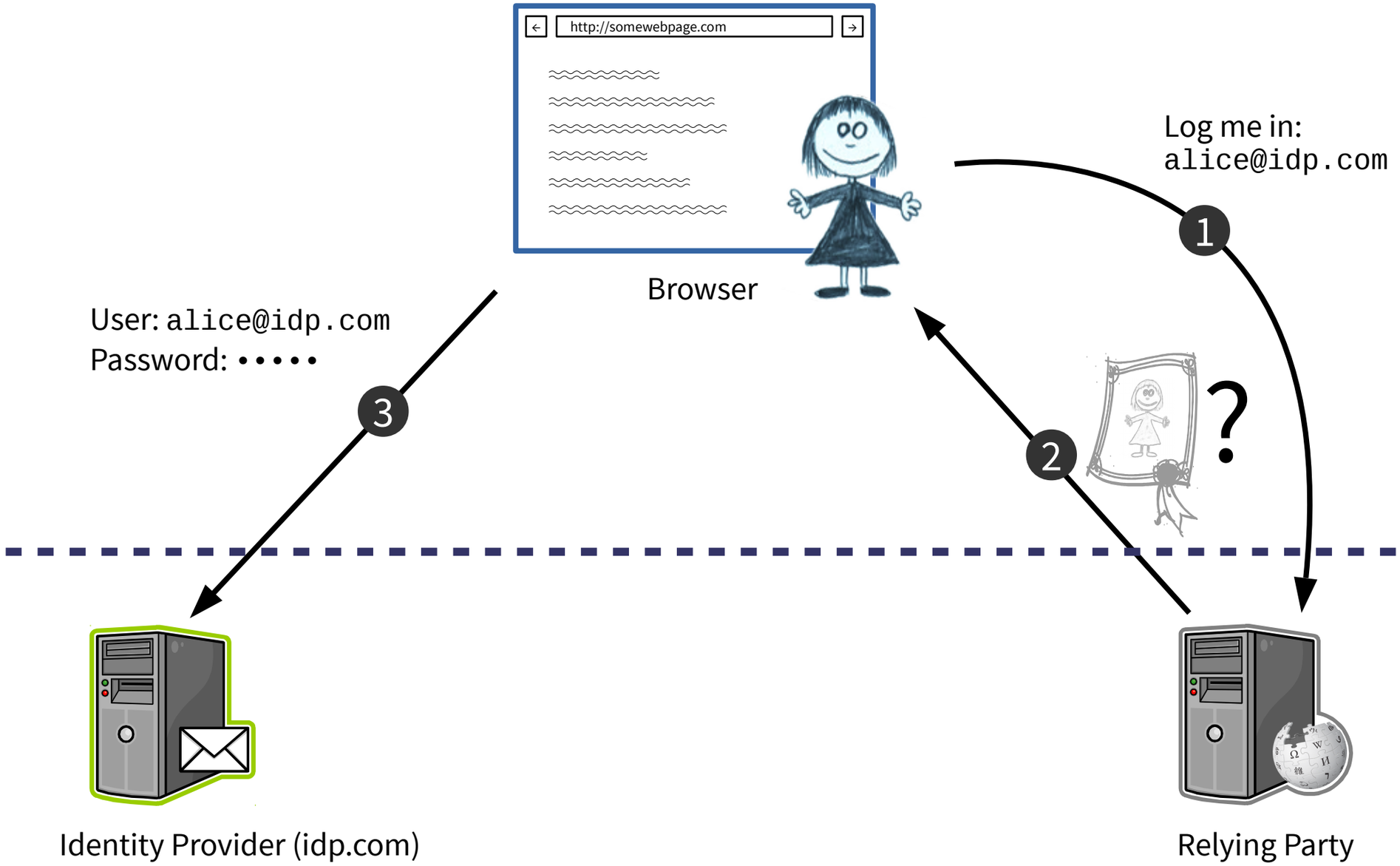
SPRESSO: Basic Principle



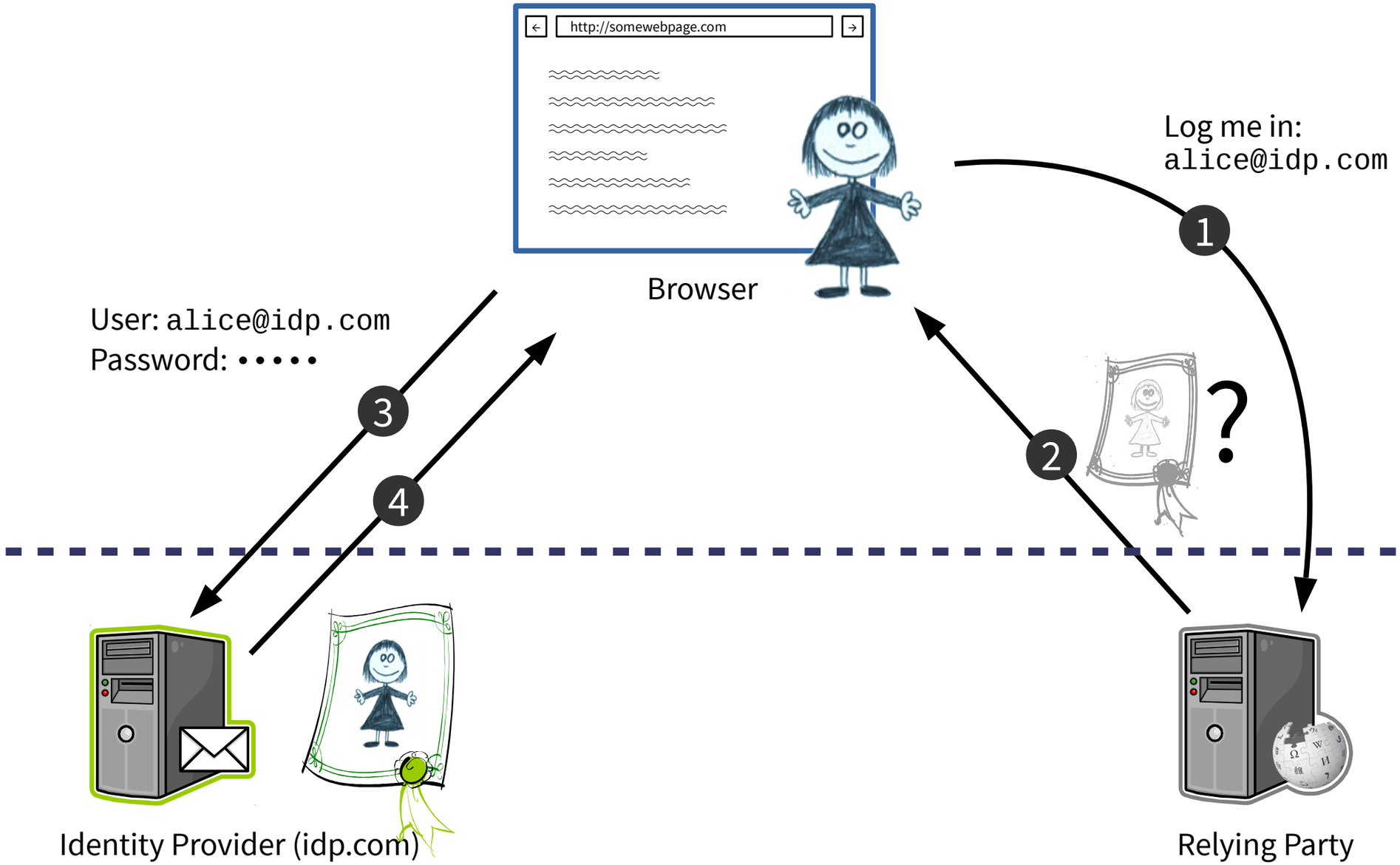
SPRESSO: Basic Principle



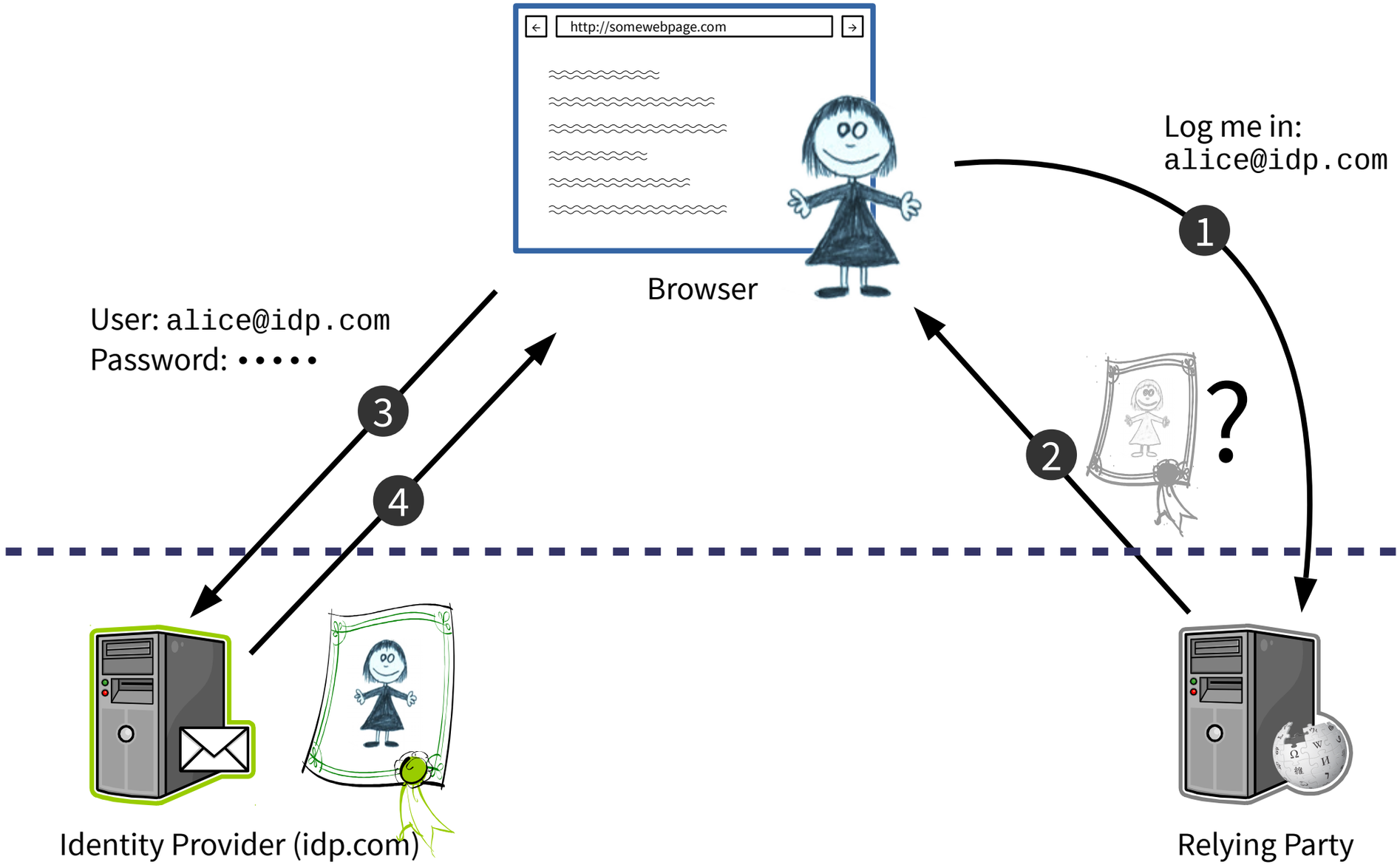
SPRESSO: Basic Principle



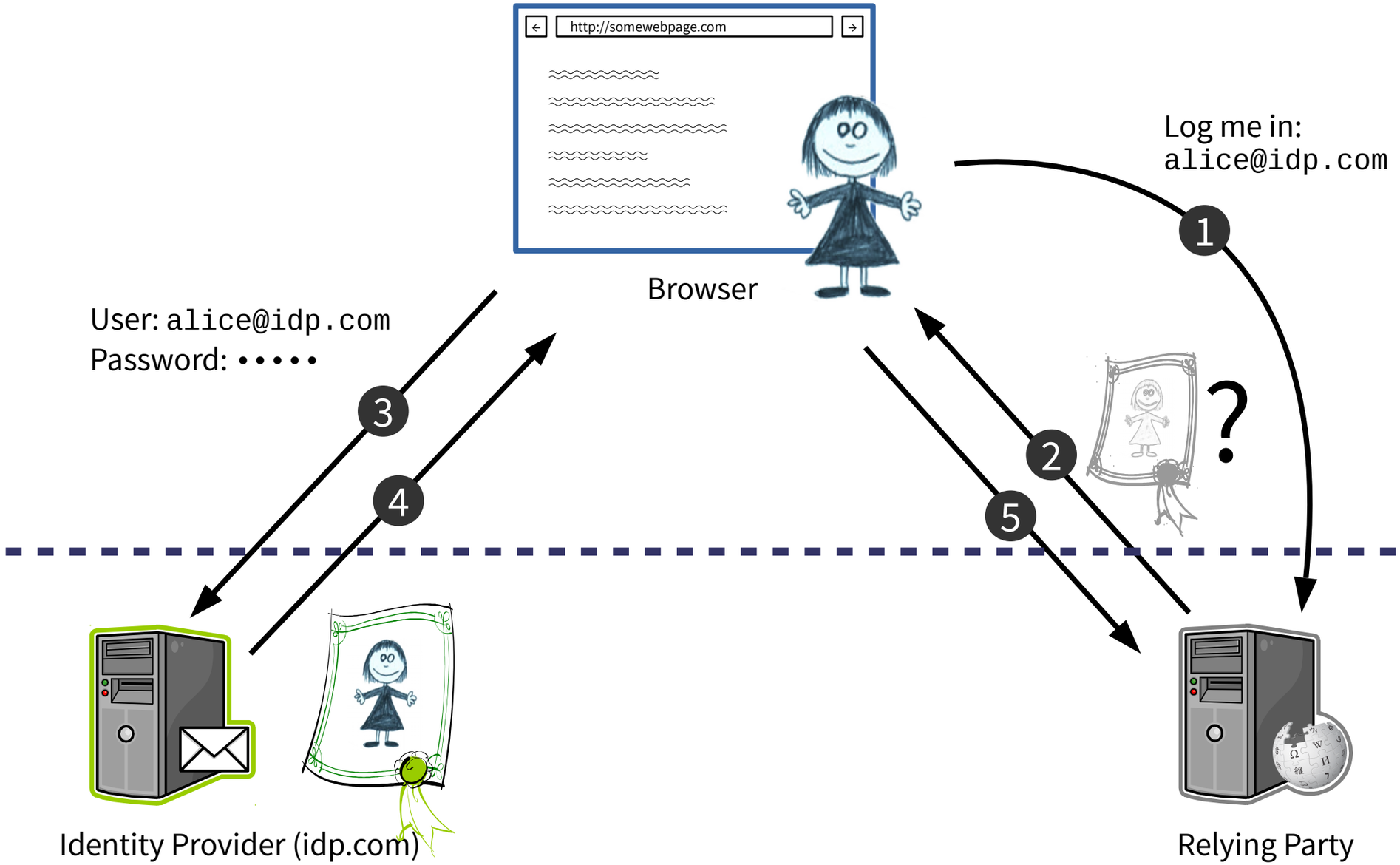
SPRESSO: Basic Principle



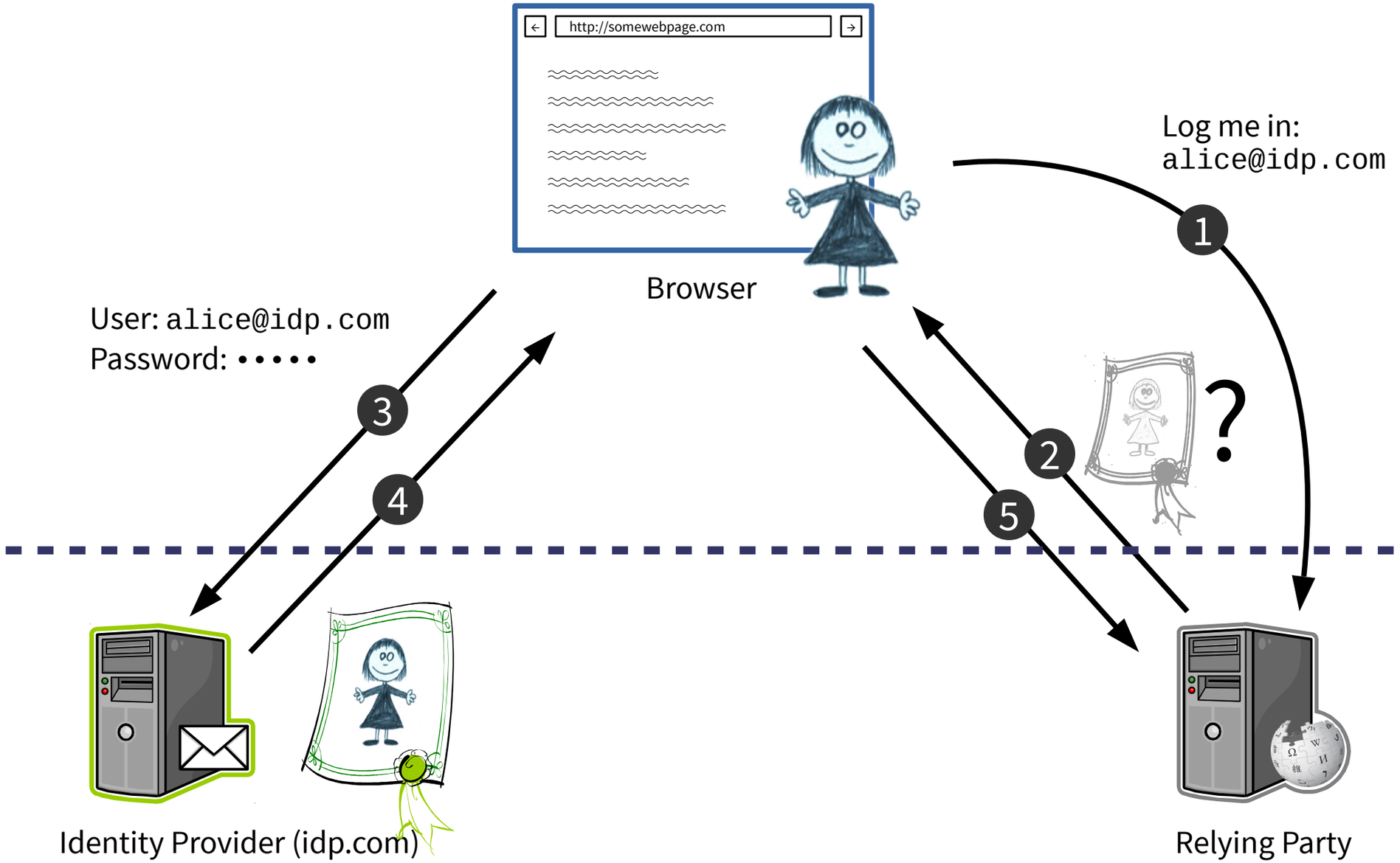
SPRESSO: Basic Principle



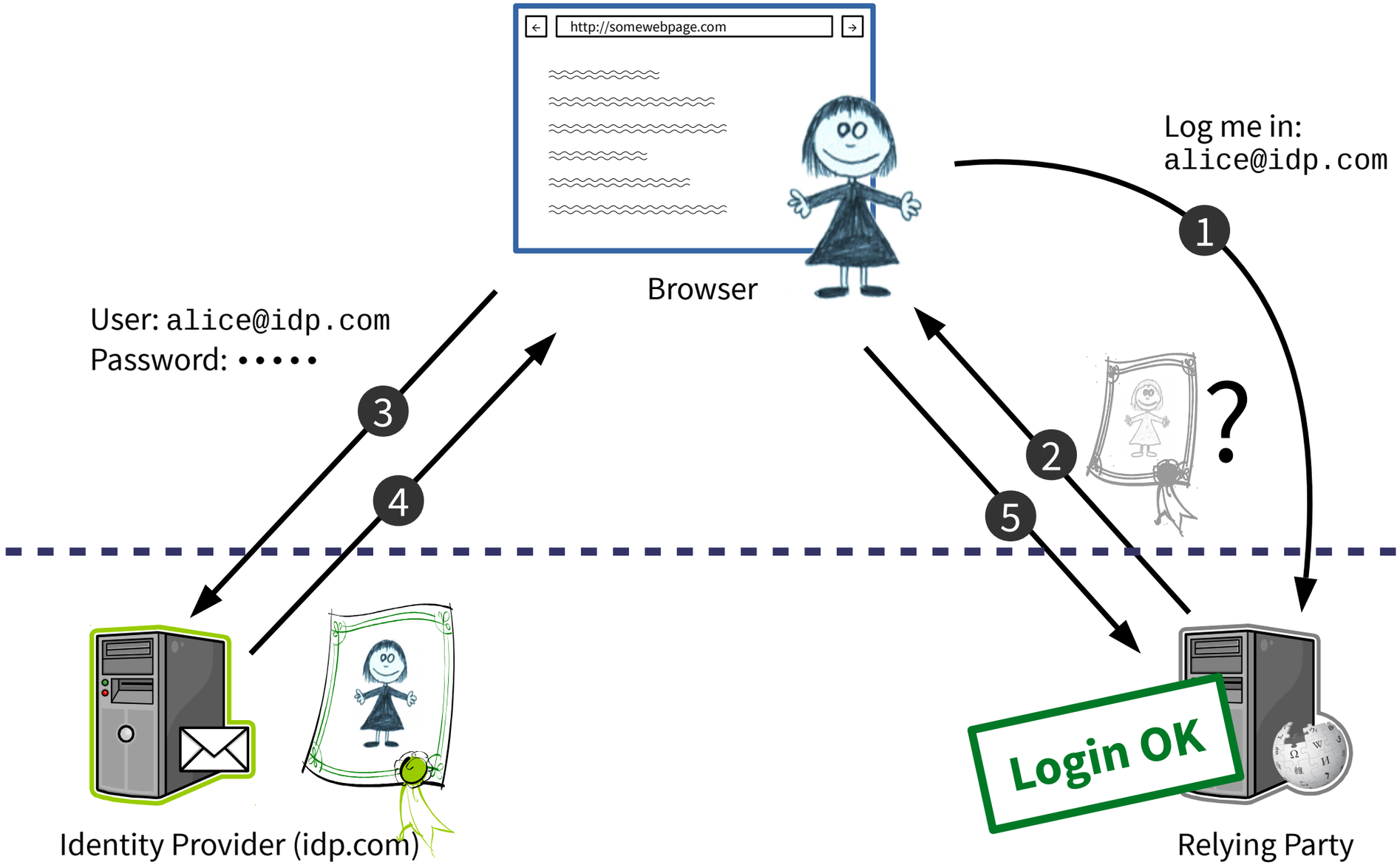
SPRESSO: Basic Principle



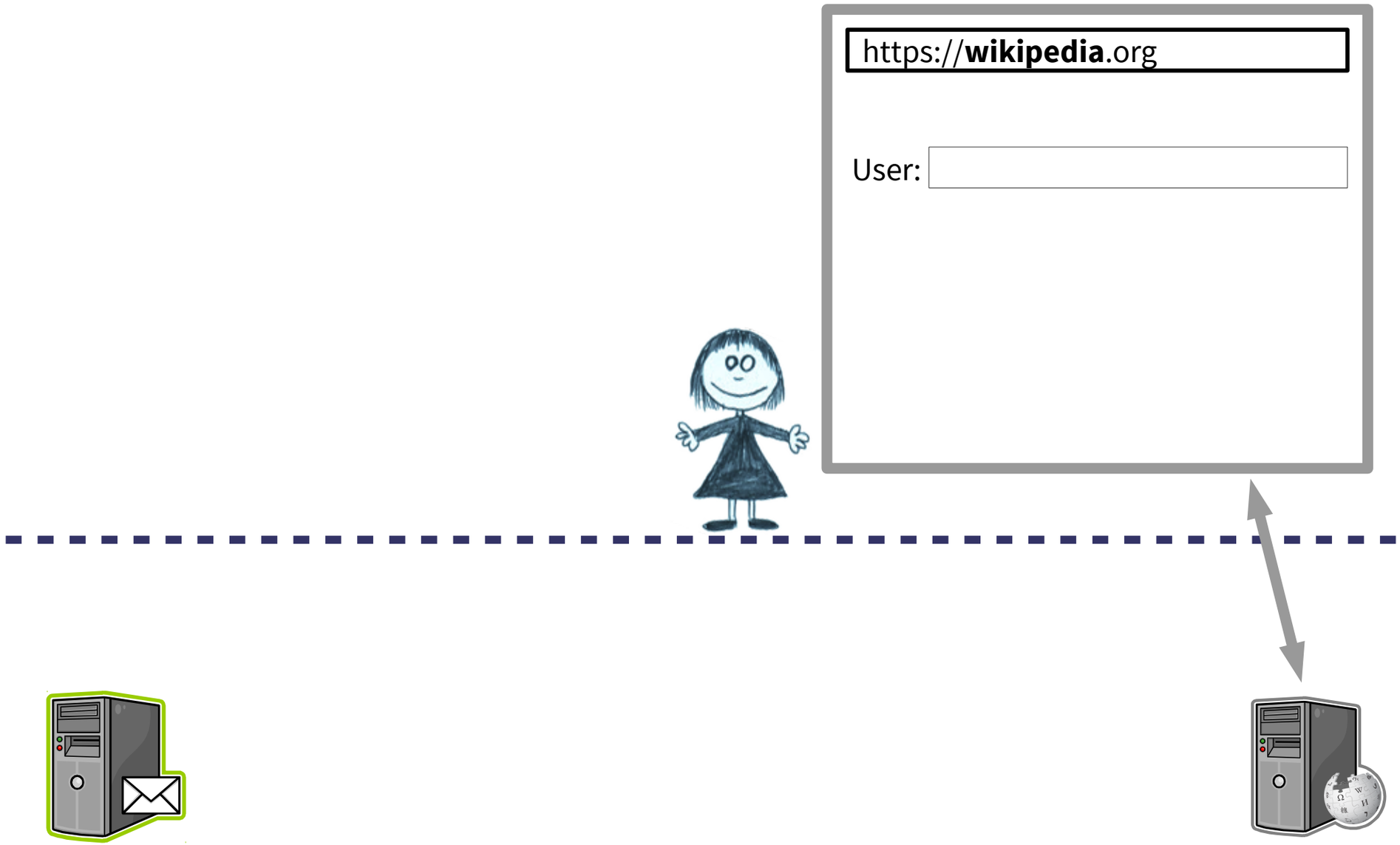
SPRESSO: Basic Principle



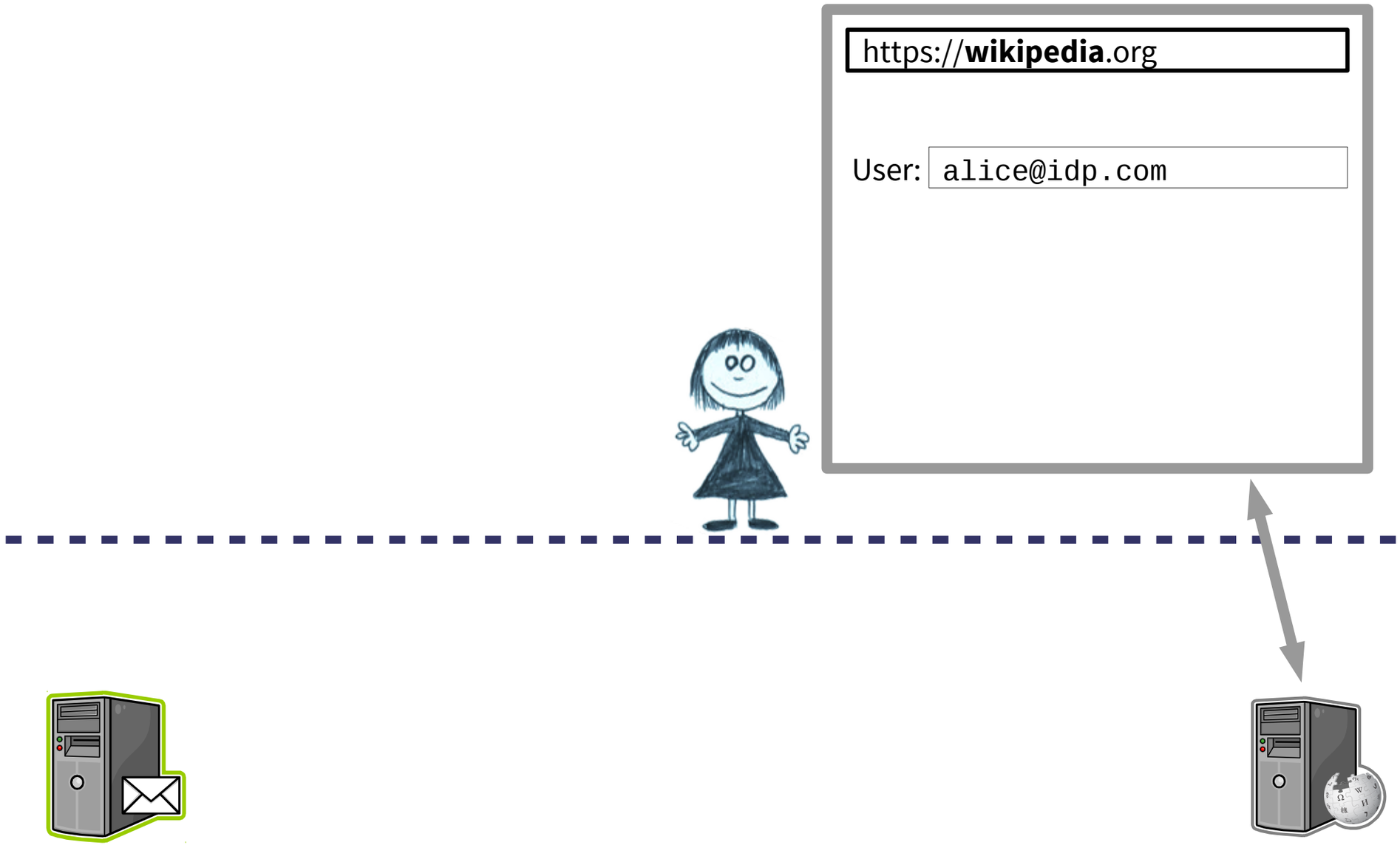
SPRESSO: Basic Principle



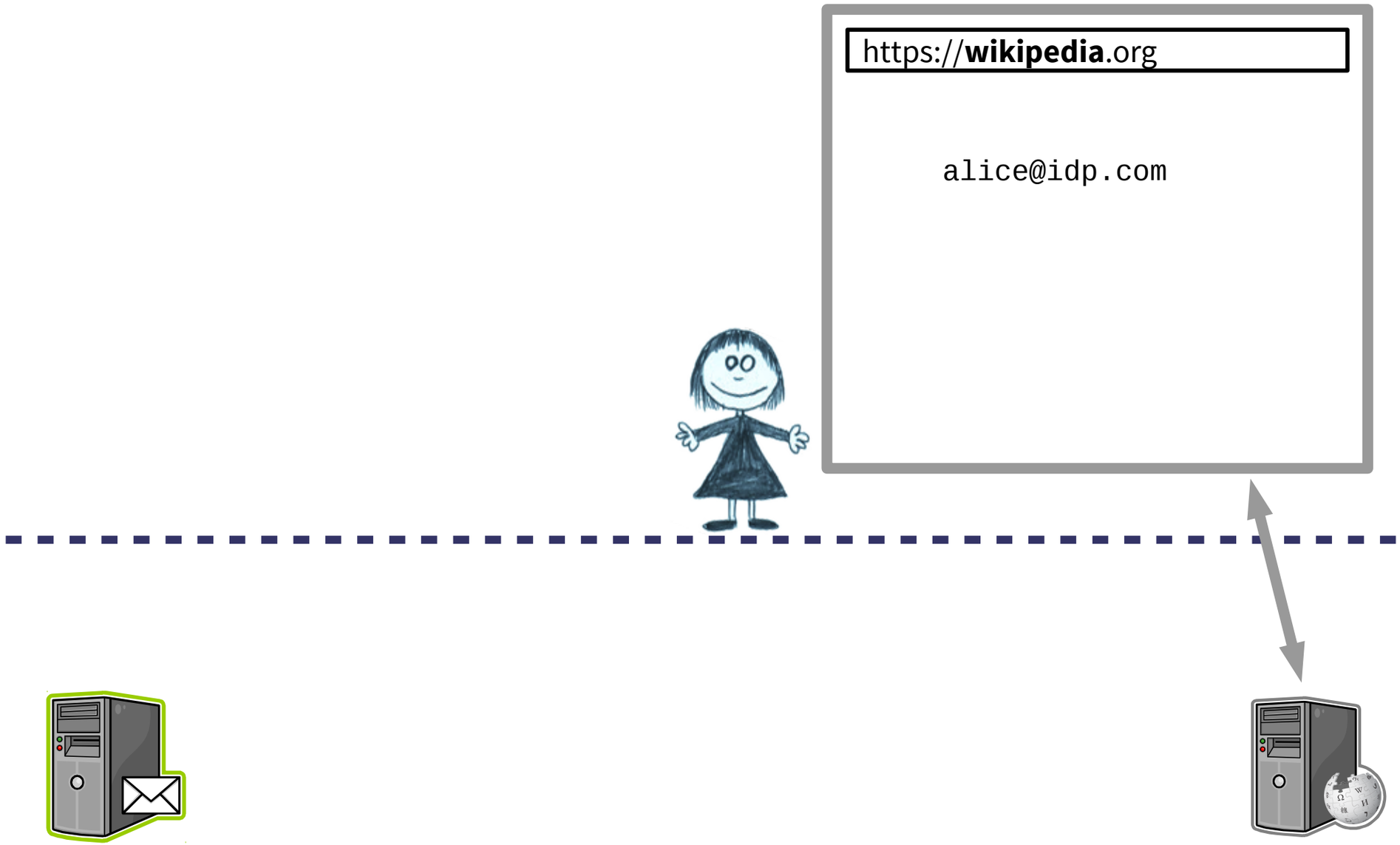
SPRESSO: Closer Look



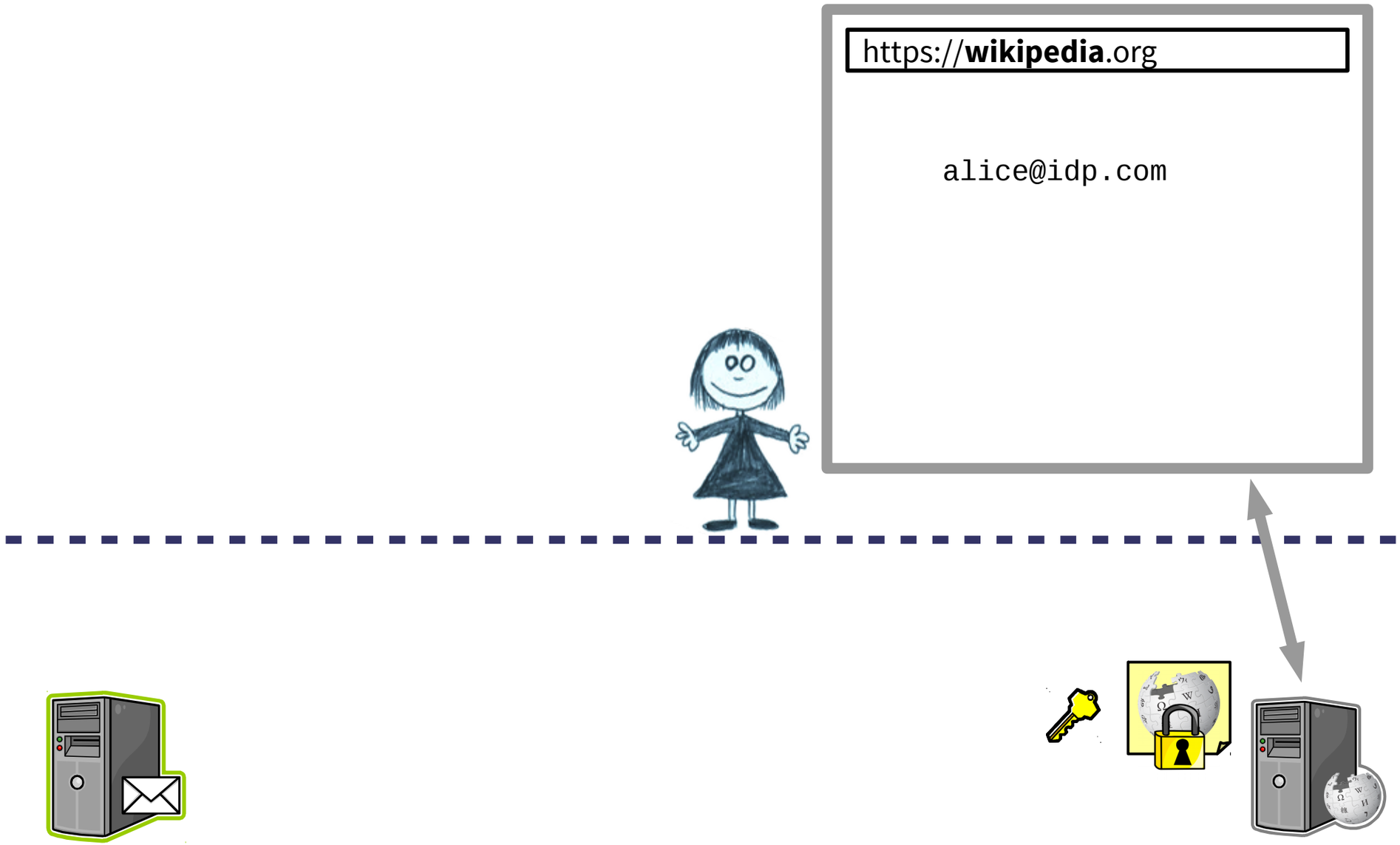
SPRESSO: Closer Look



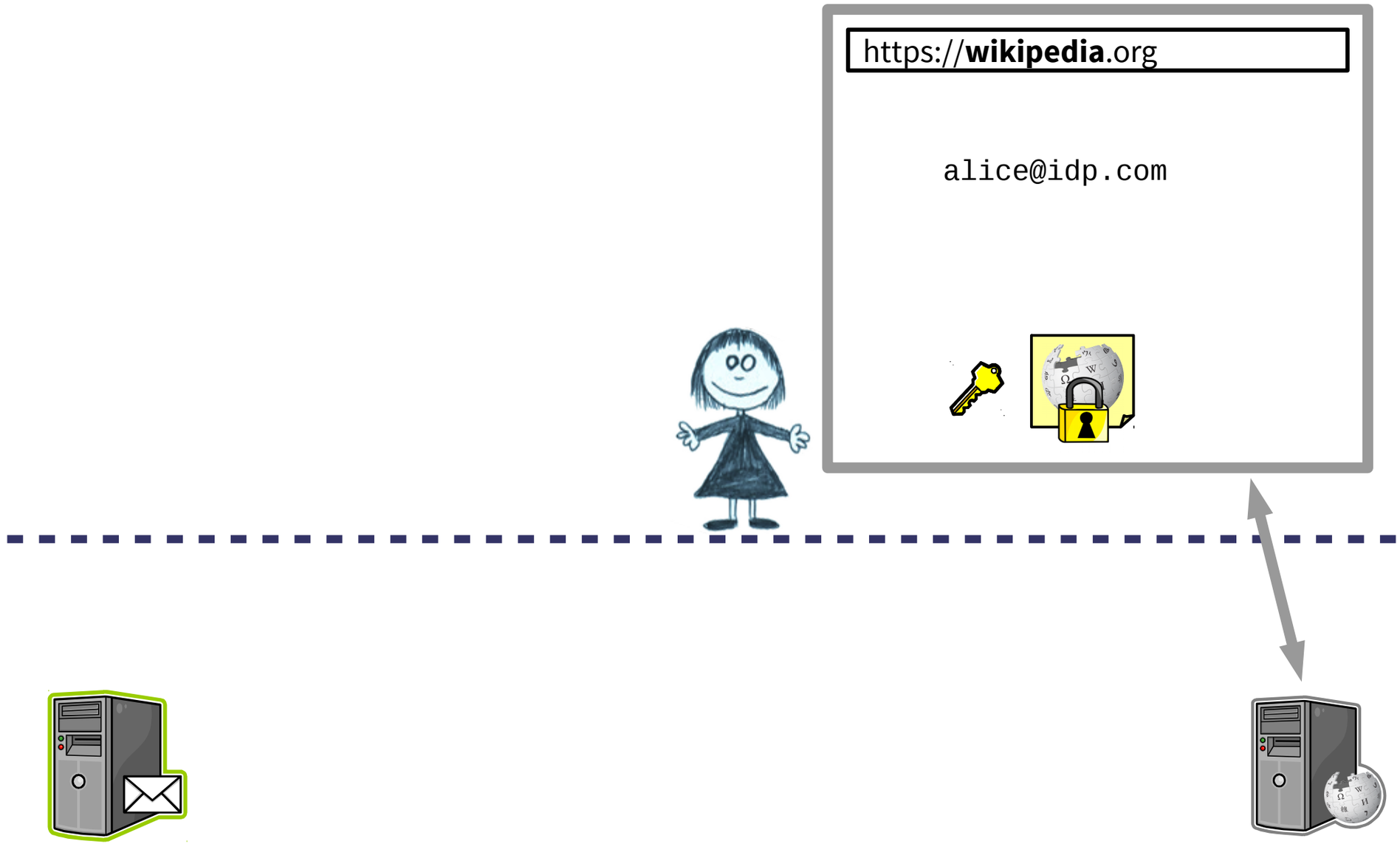
SPRESSO: Closer Look



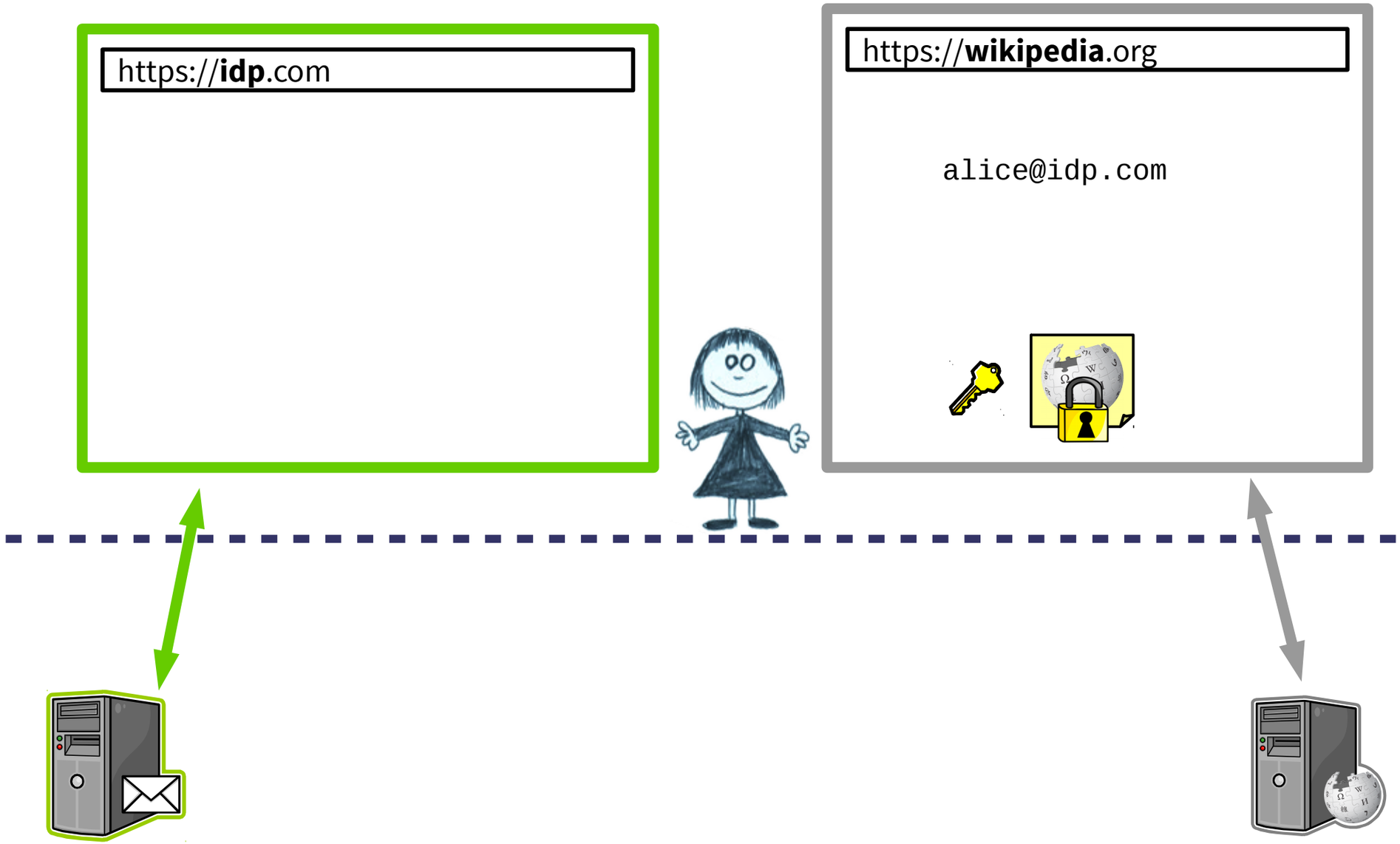
SPRESSO: Closer Look



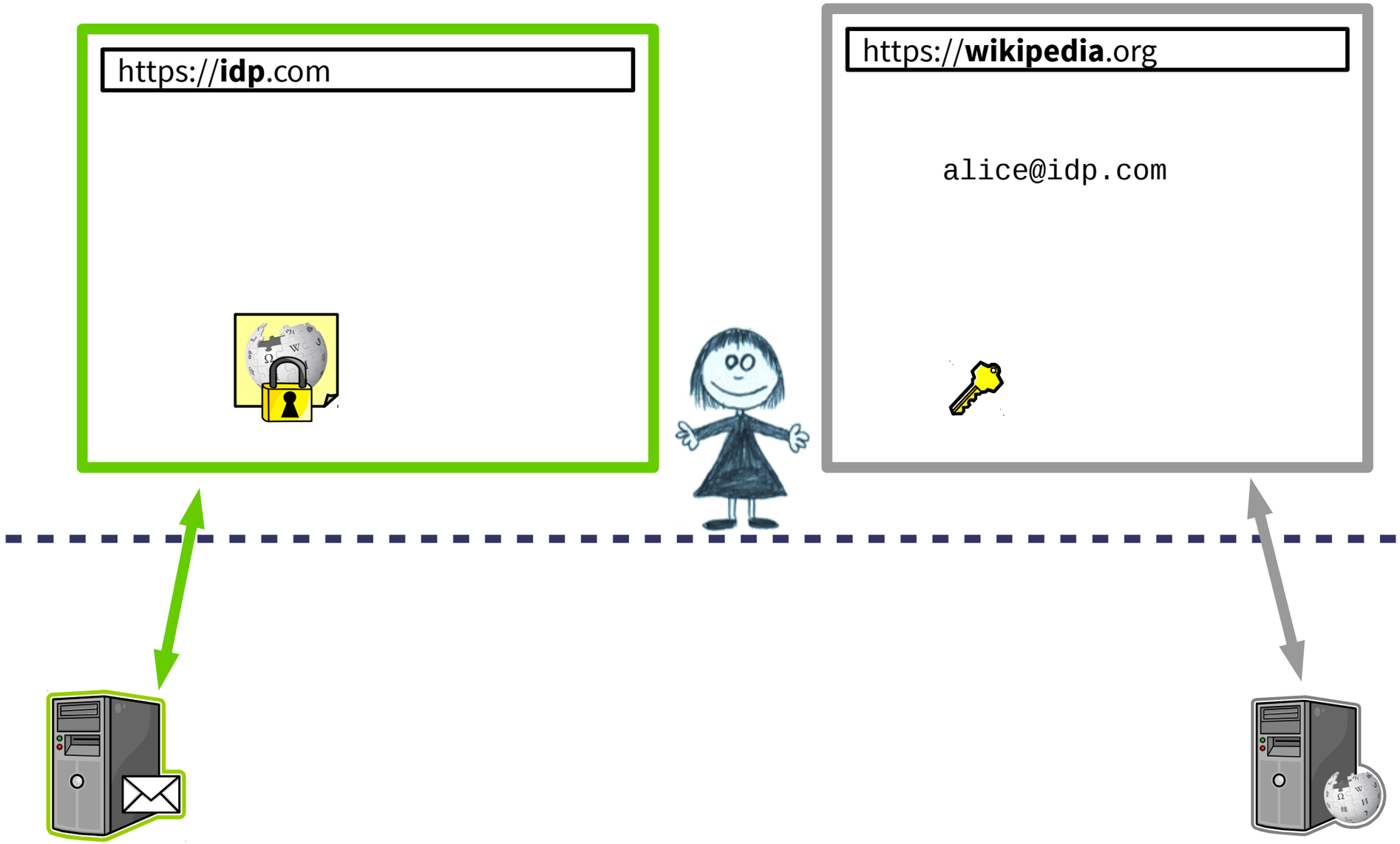
SPRESSO: Closer Look



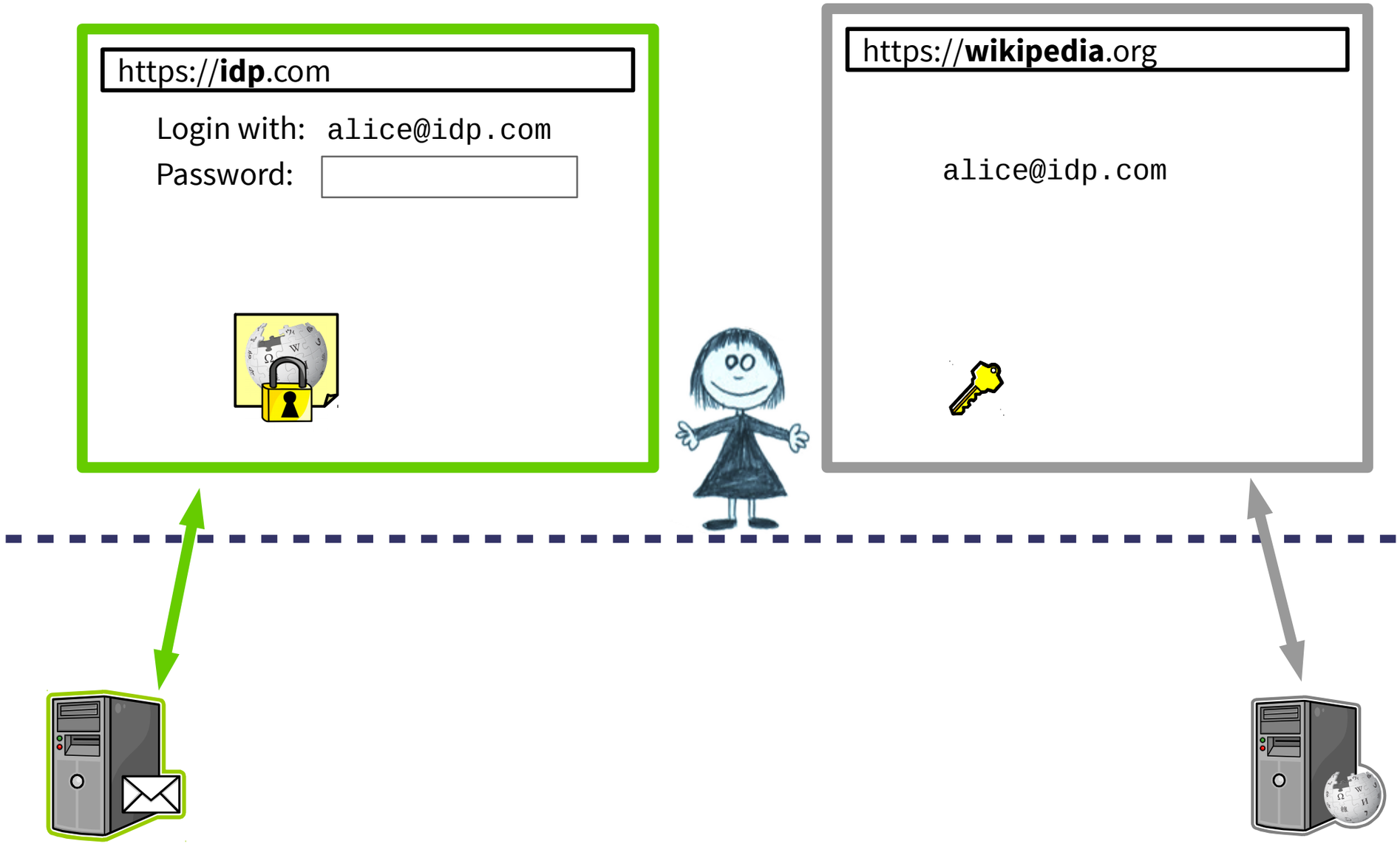
SPRESSO: Closer Look



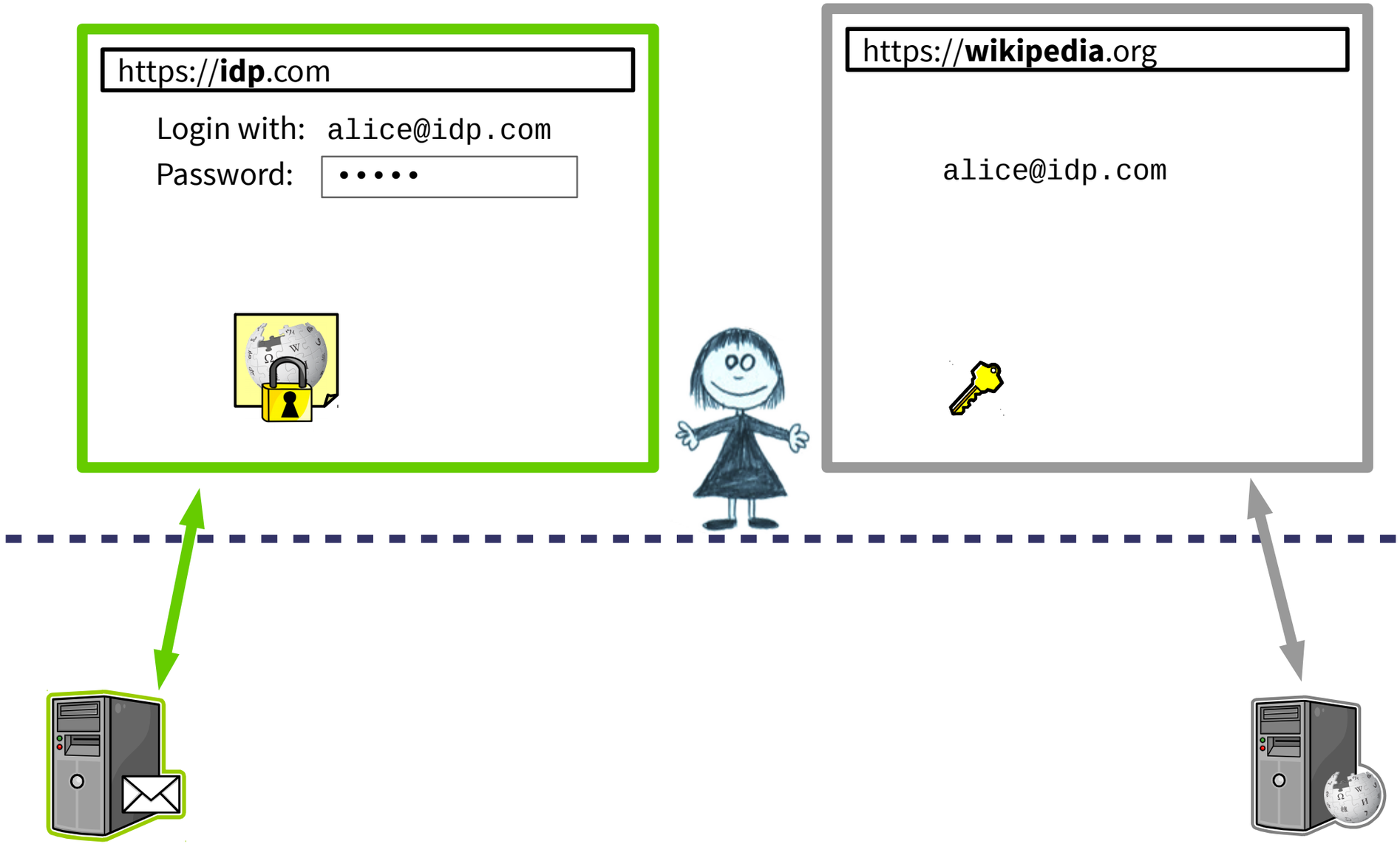
SPRESSO: Closer Look



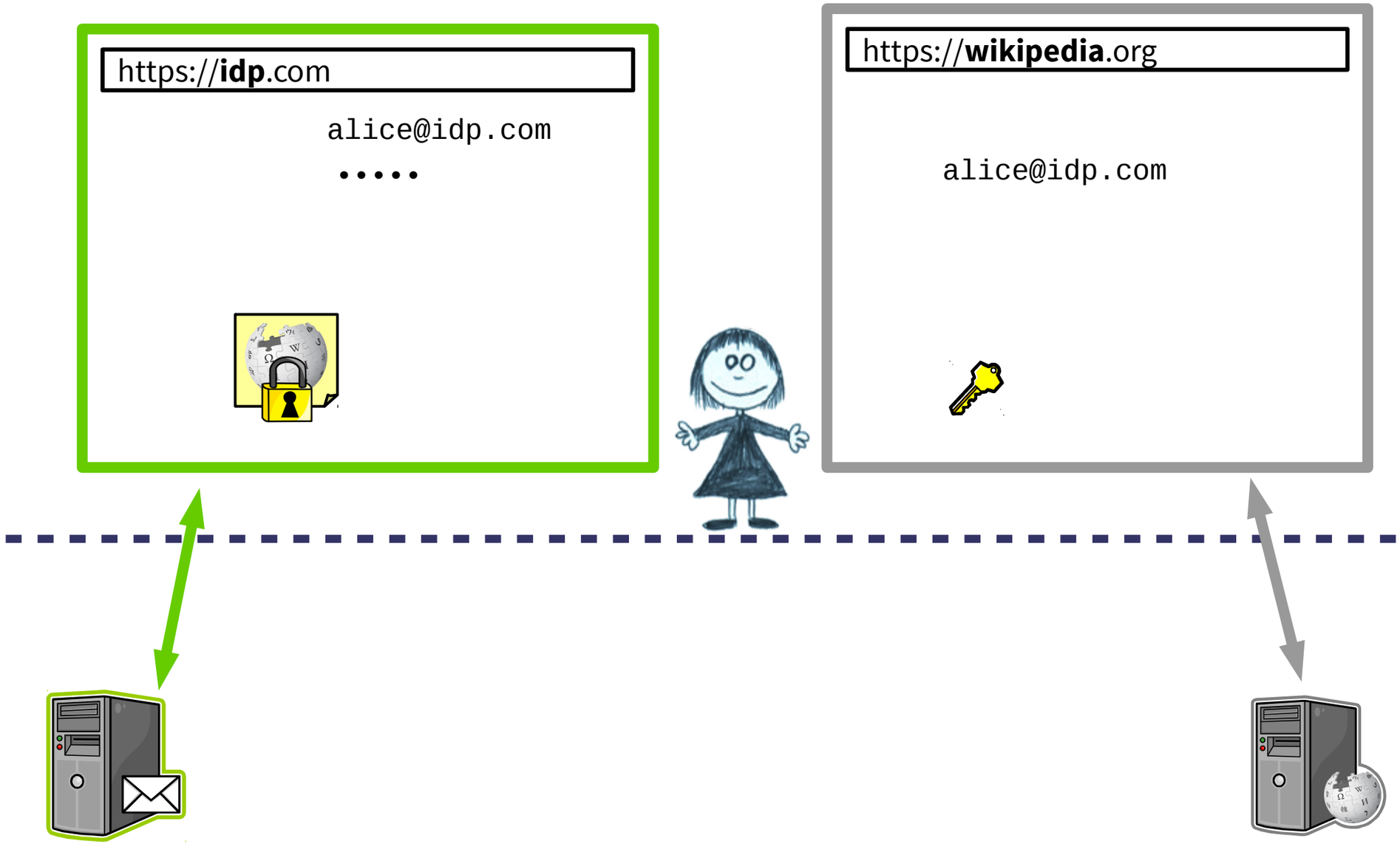
SPRESSO: Closer Look



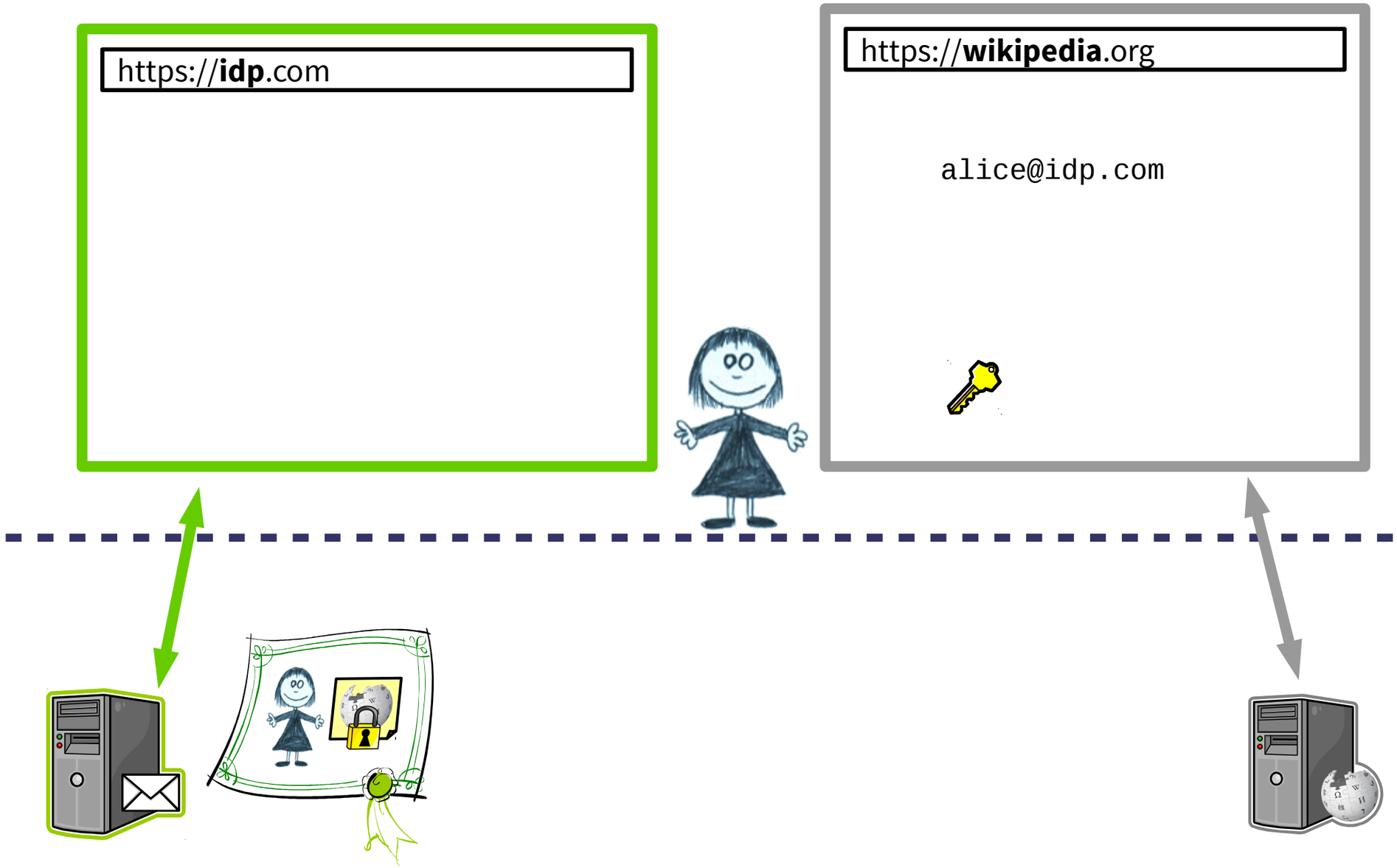
SPRESSO: Closer Look



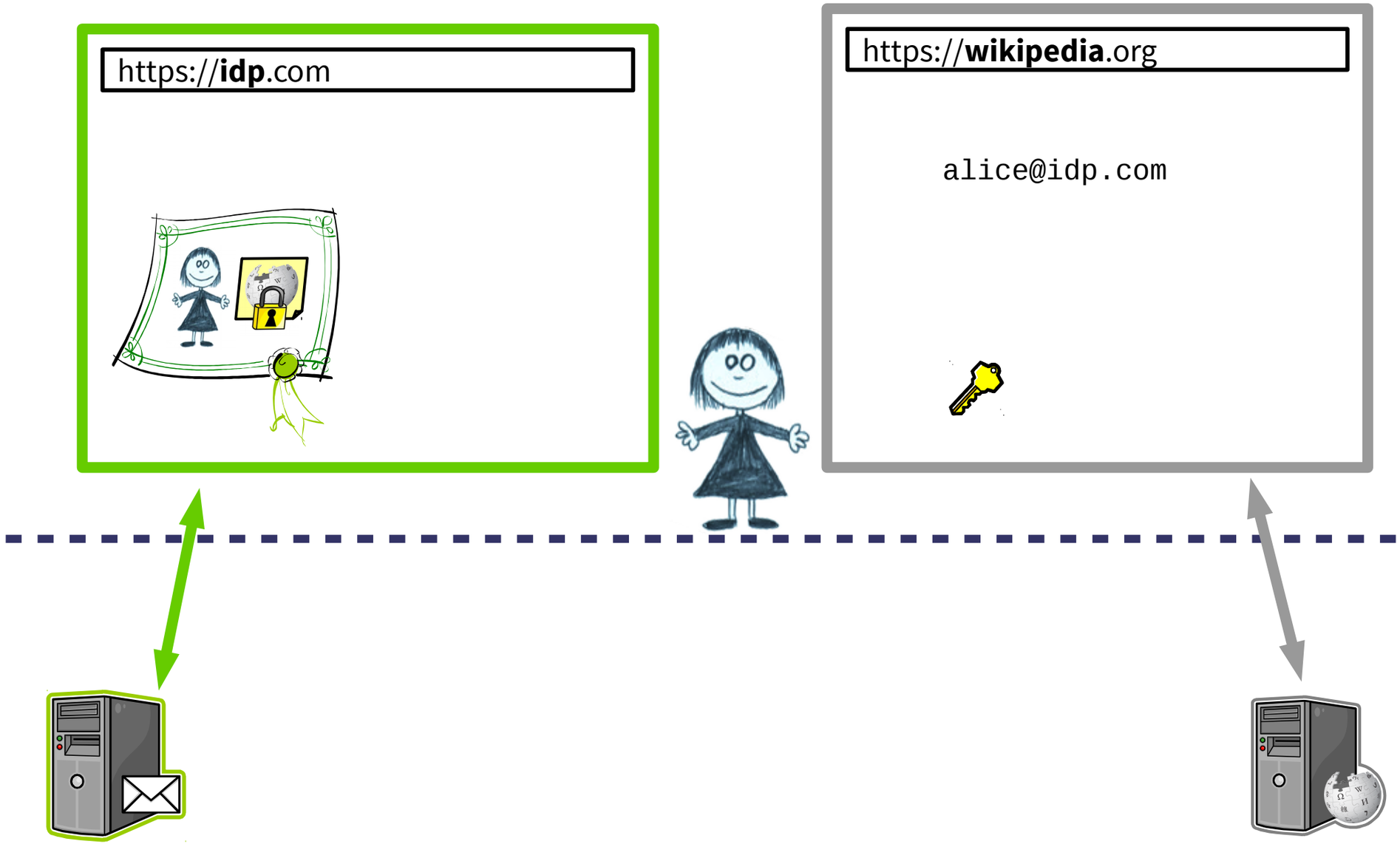
SPRESSO: Closer Look



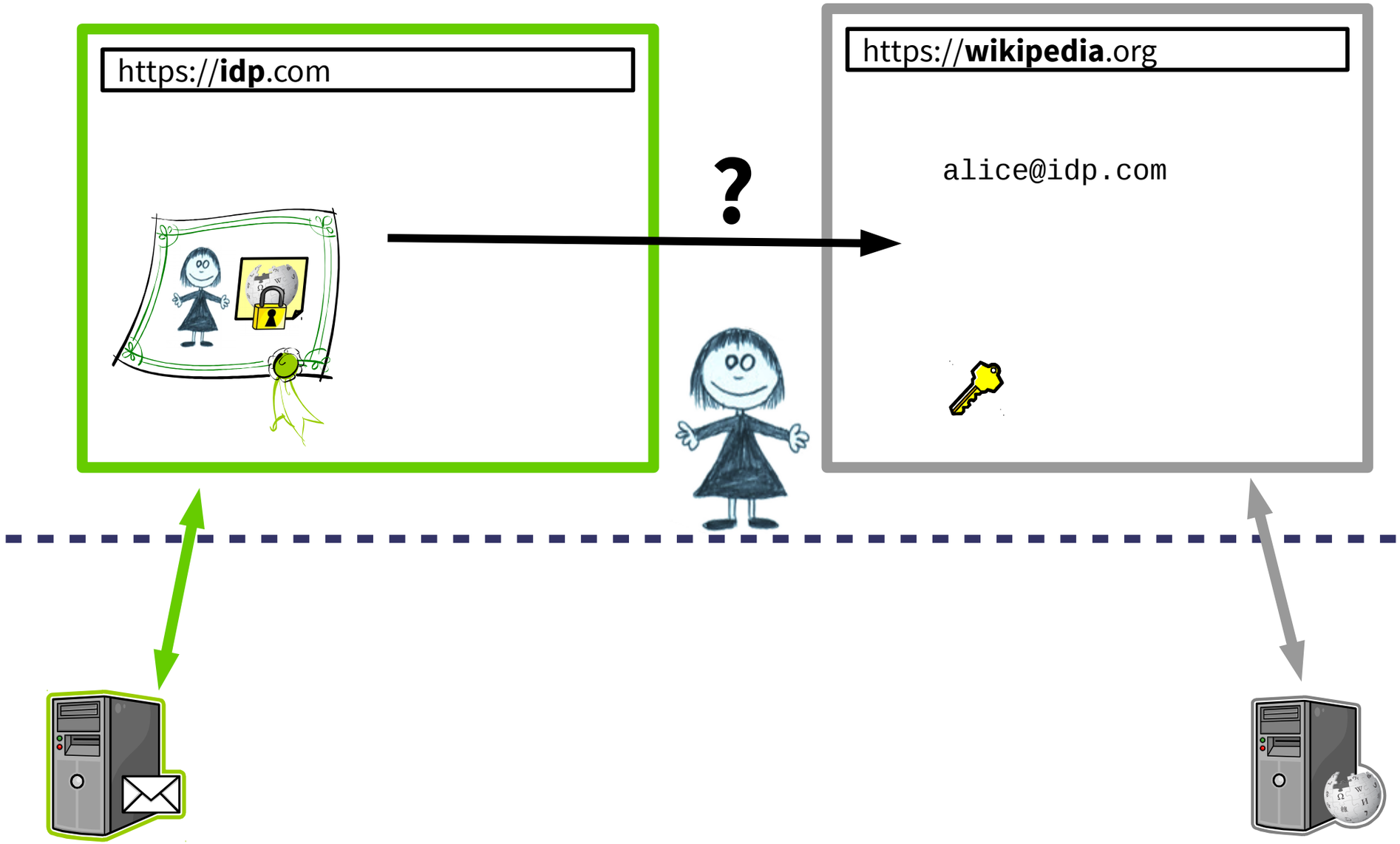
SPRESSO: Closer Look



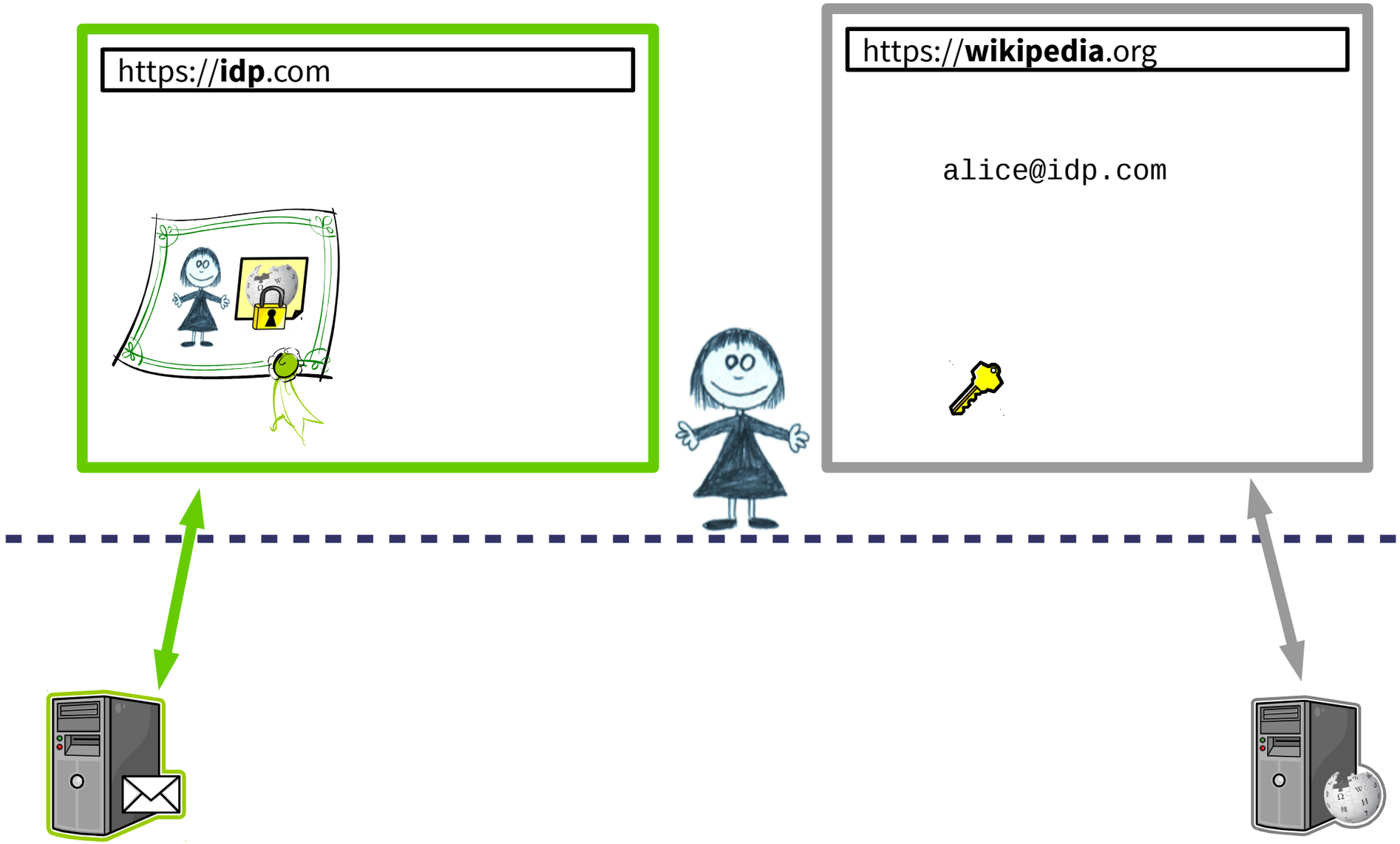
SPRESSO: Closer Look



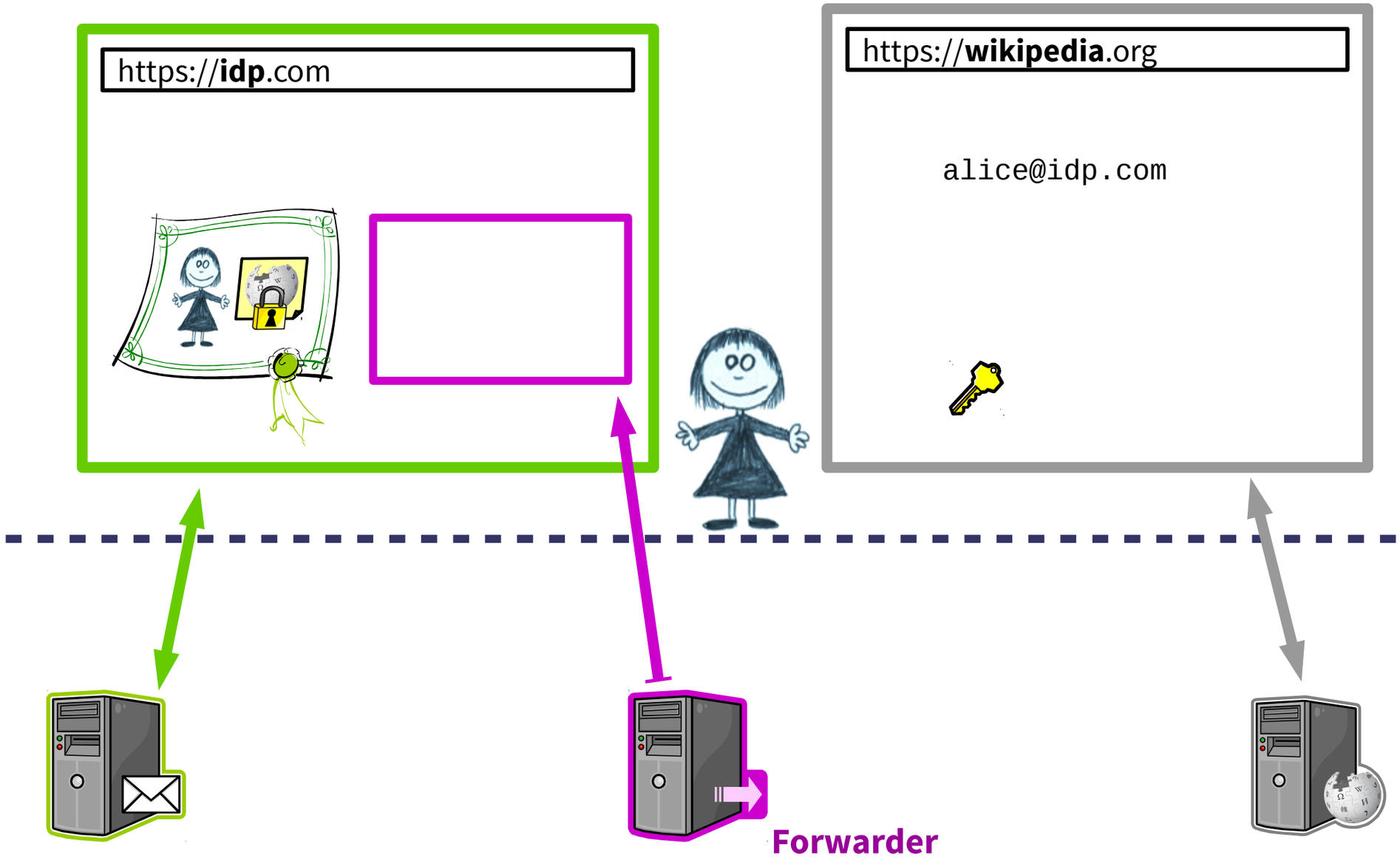
SPRESSO: Closer Look



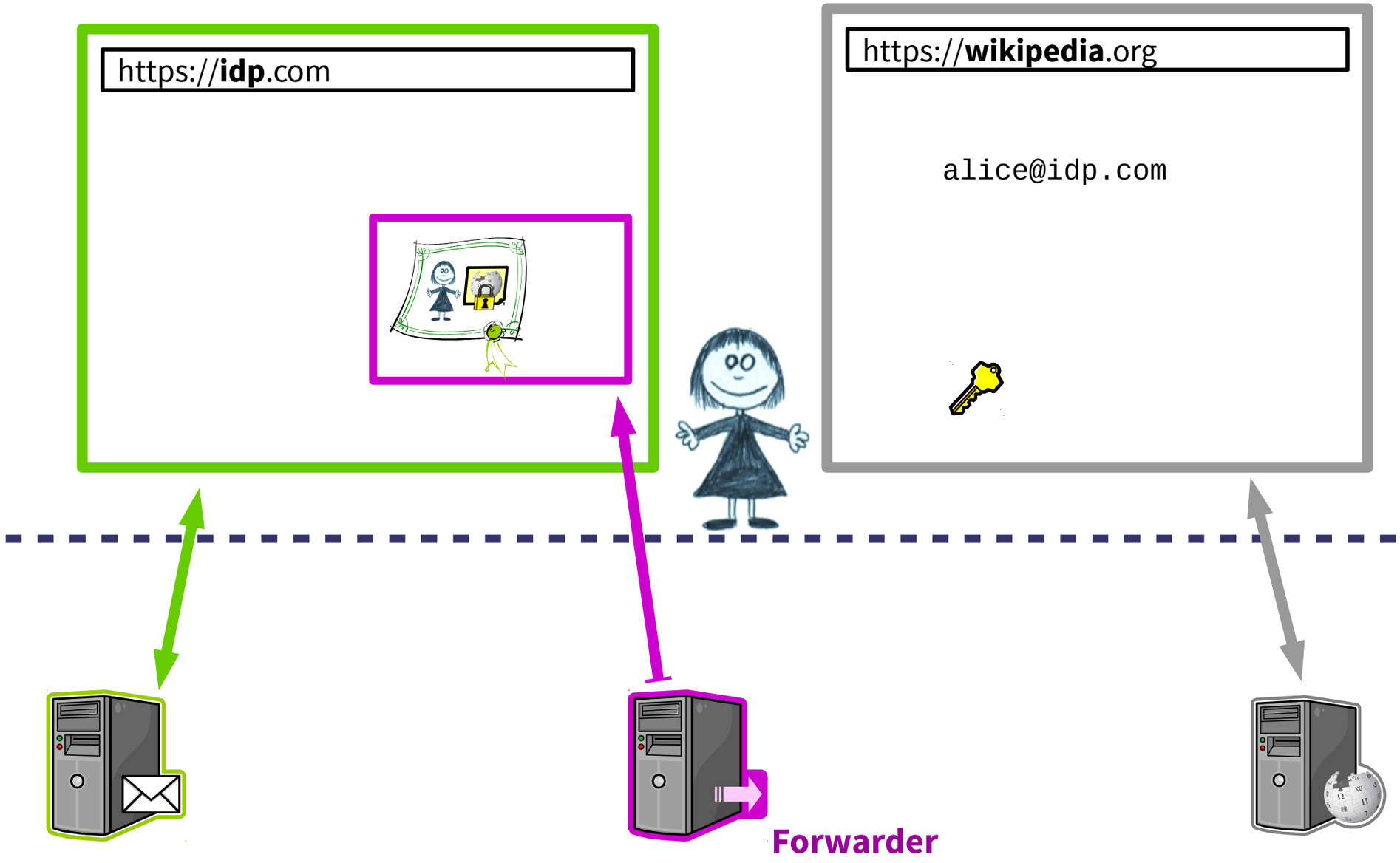
SPRESSO: Closer Look



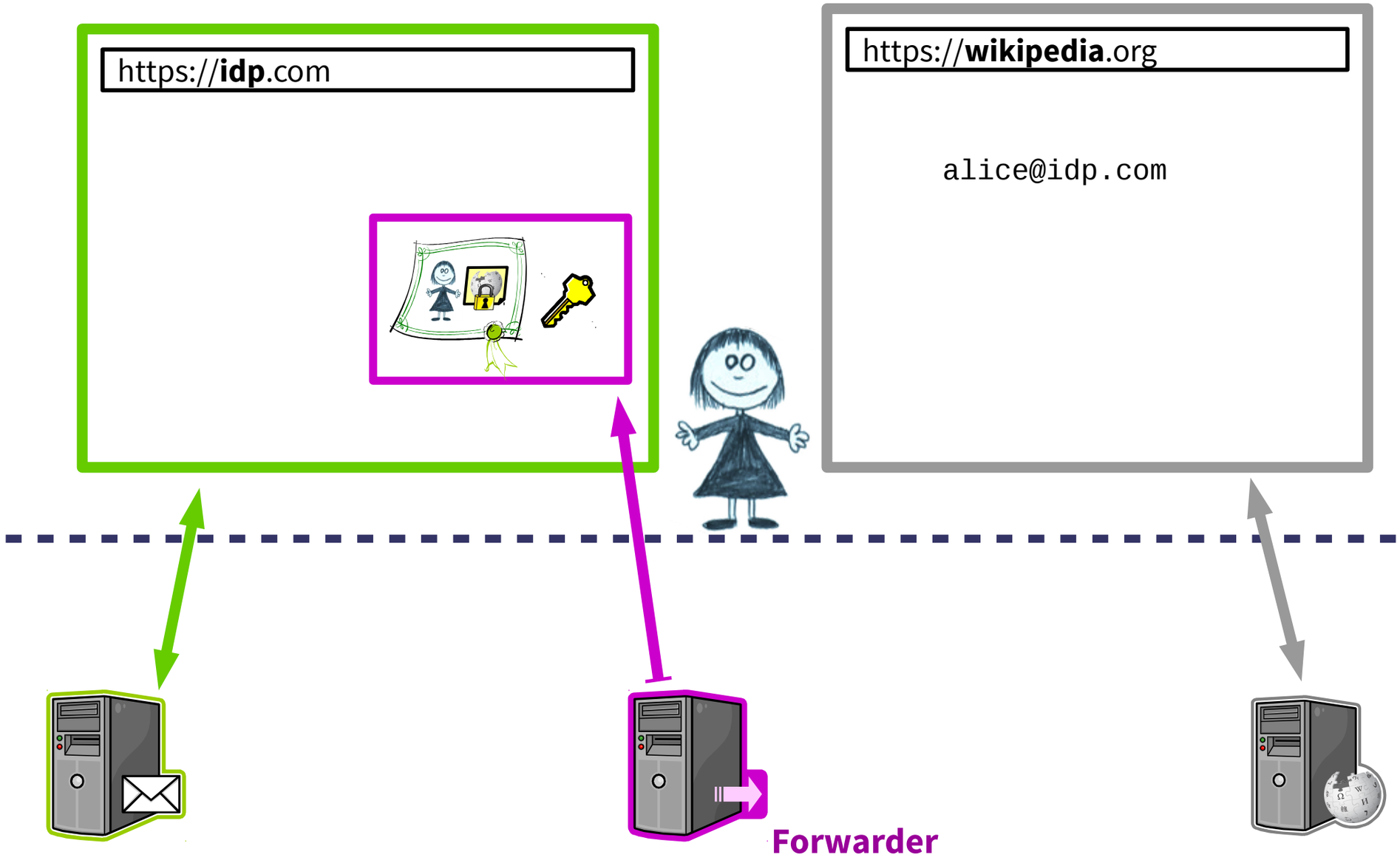
SPRESSO: Closer Look



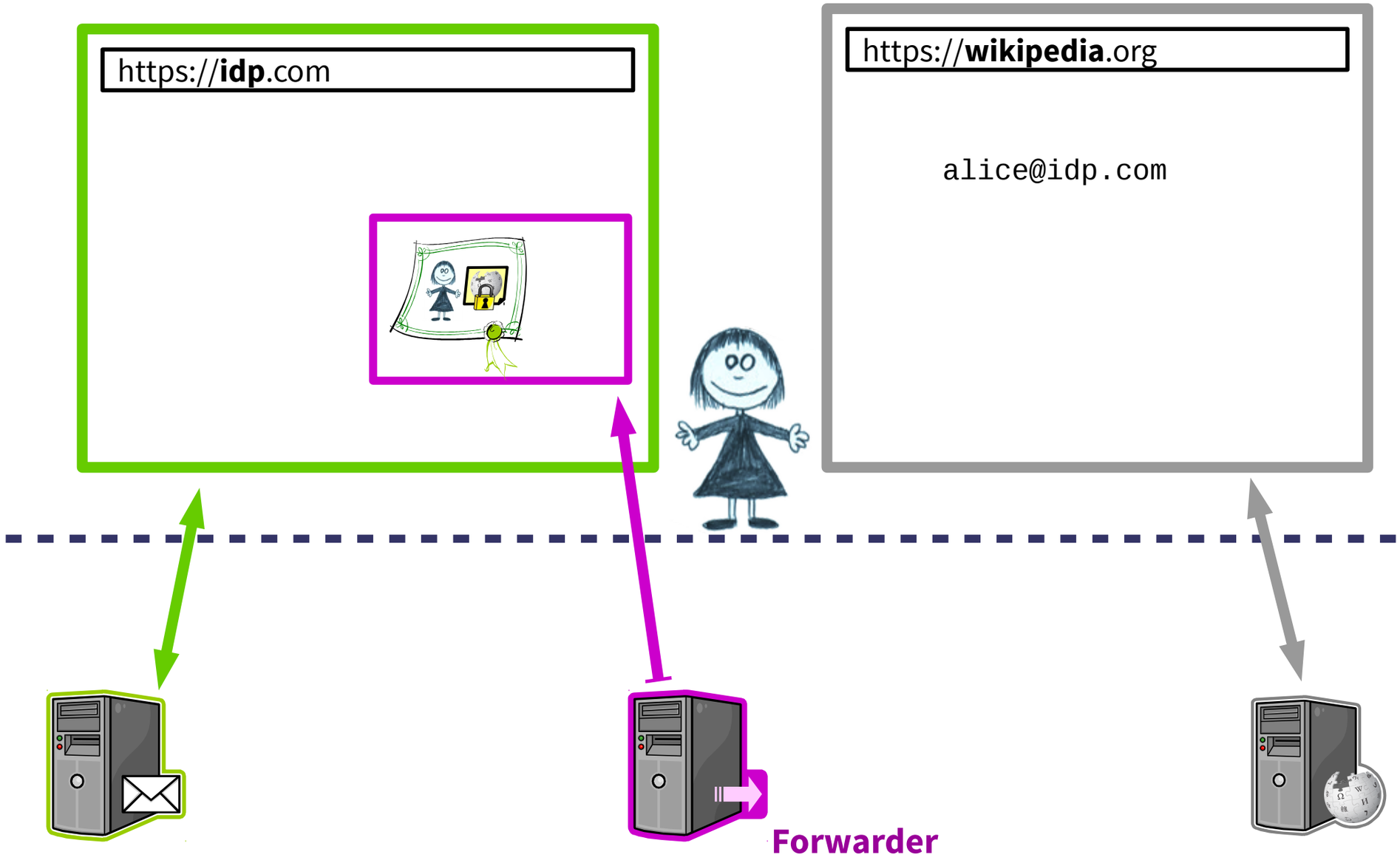
SPRESSO: Closer Look



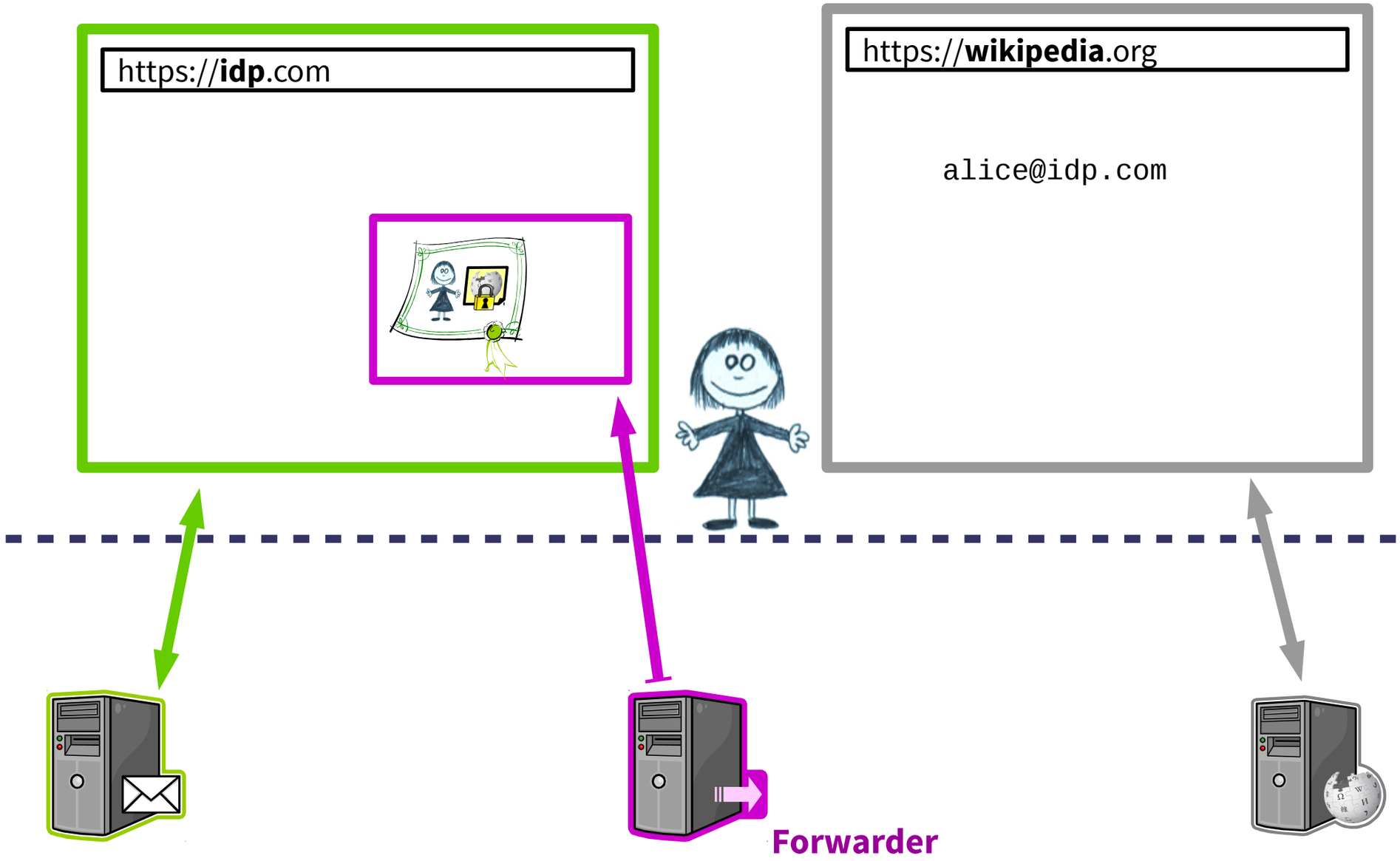
SPRESSO: Closer Look



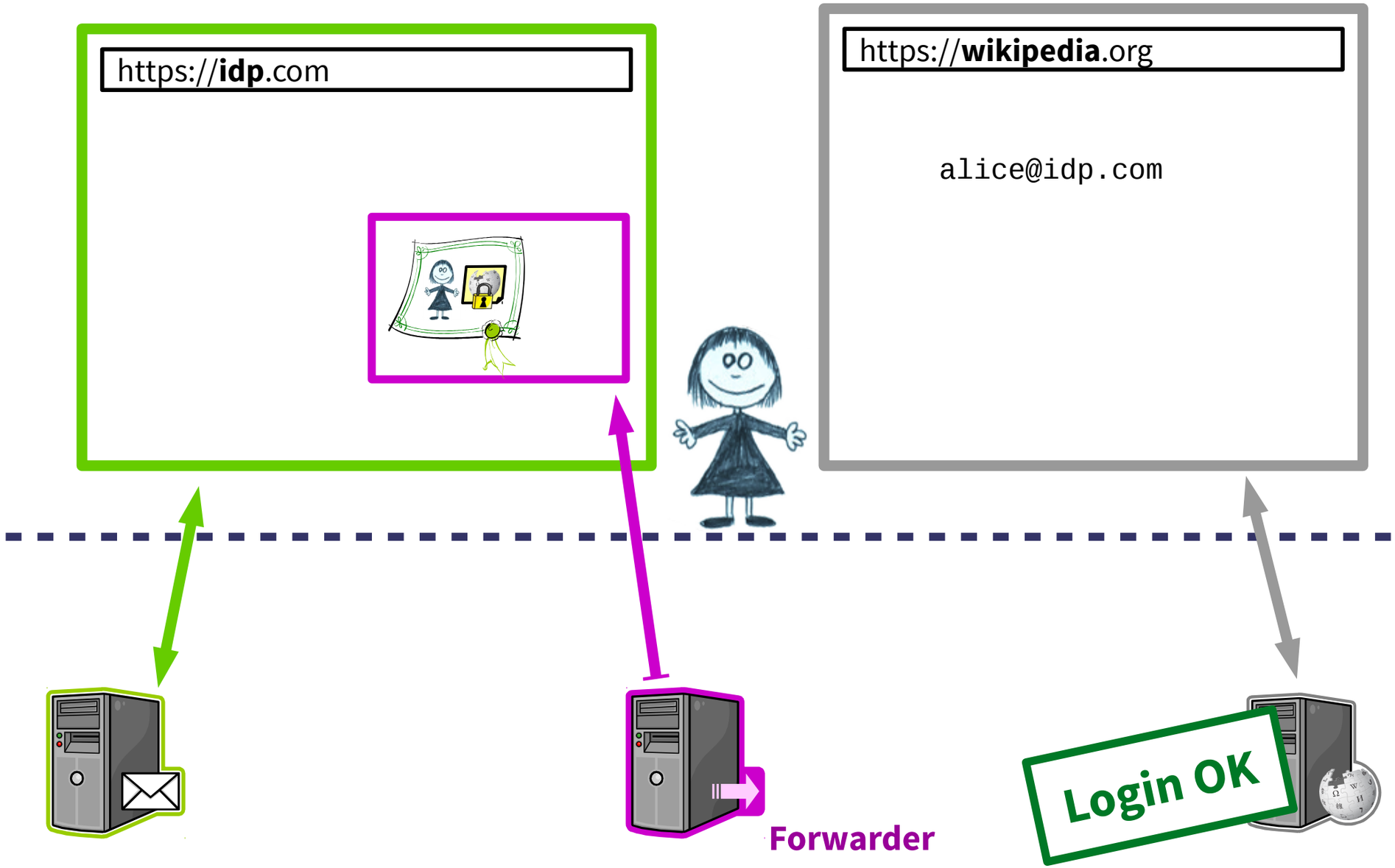
SPRESSO: Closer Look



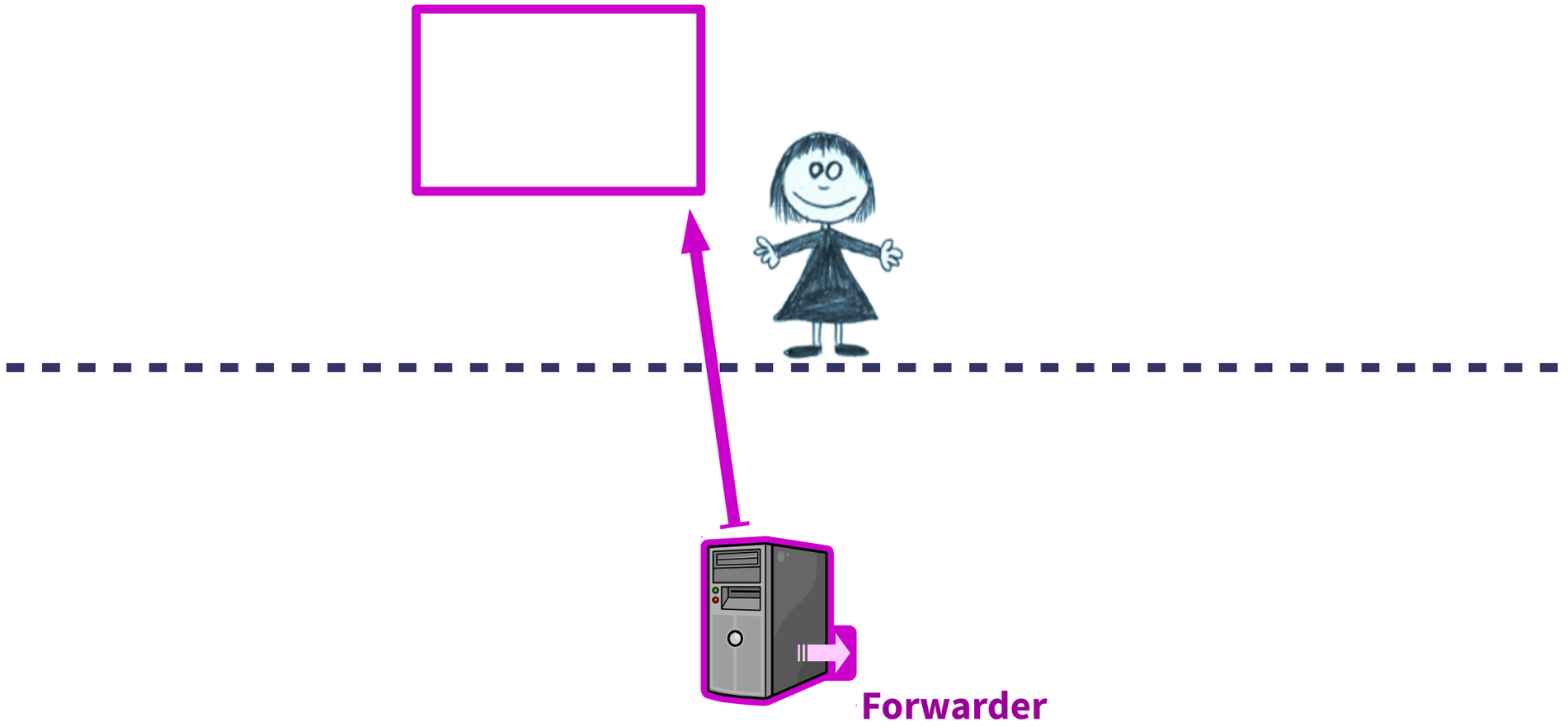
SPRESSO: Closer Look



SPRESSO: Closer Look

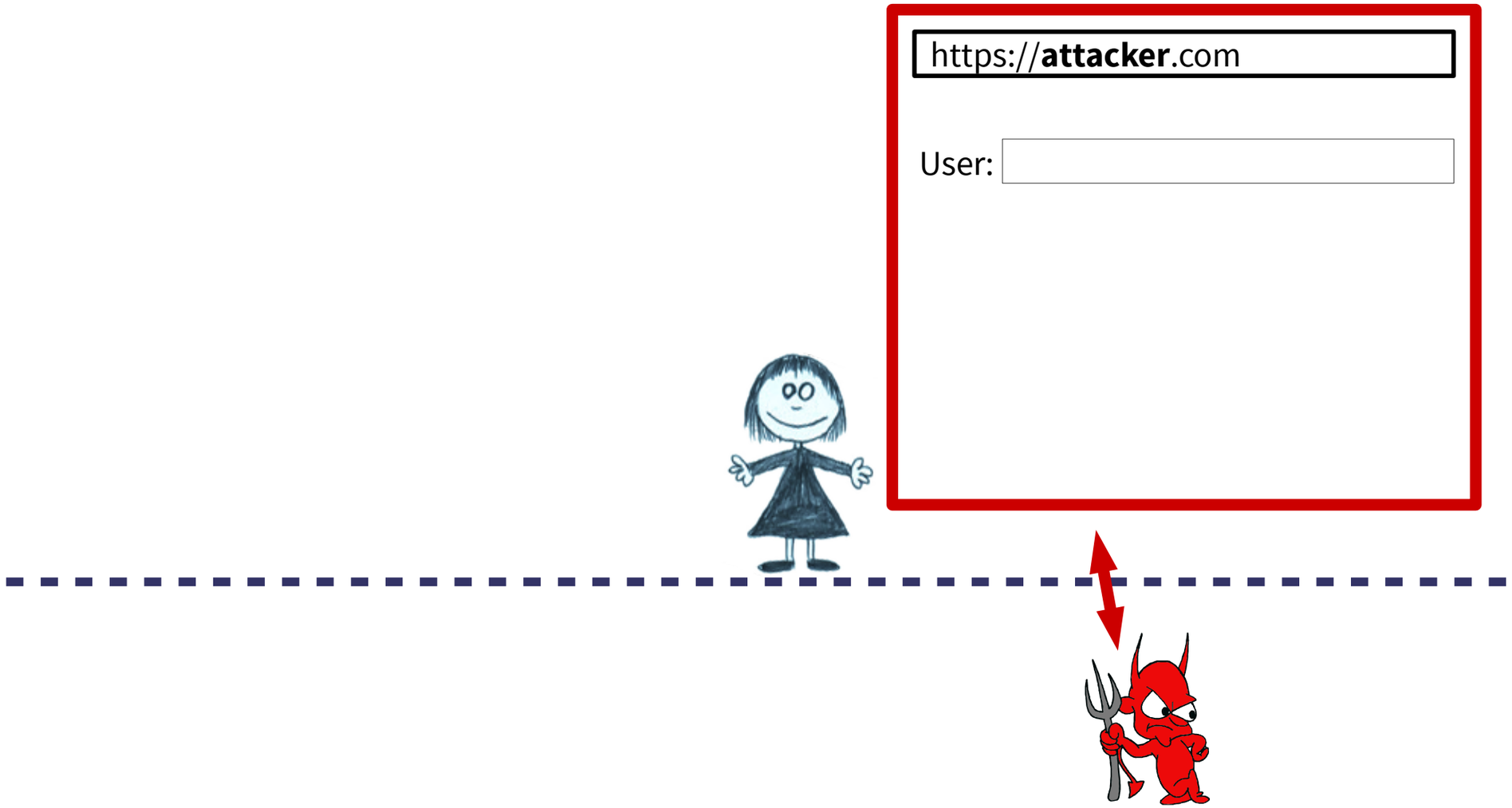


Why Forwarder?

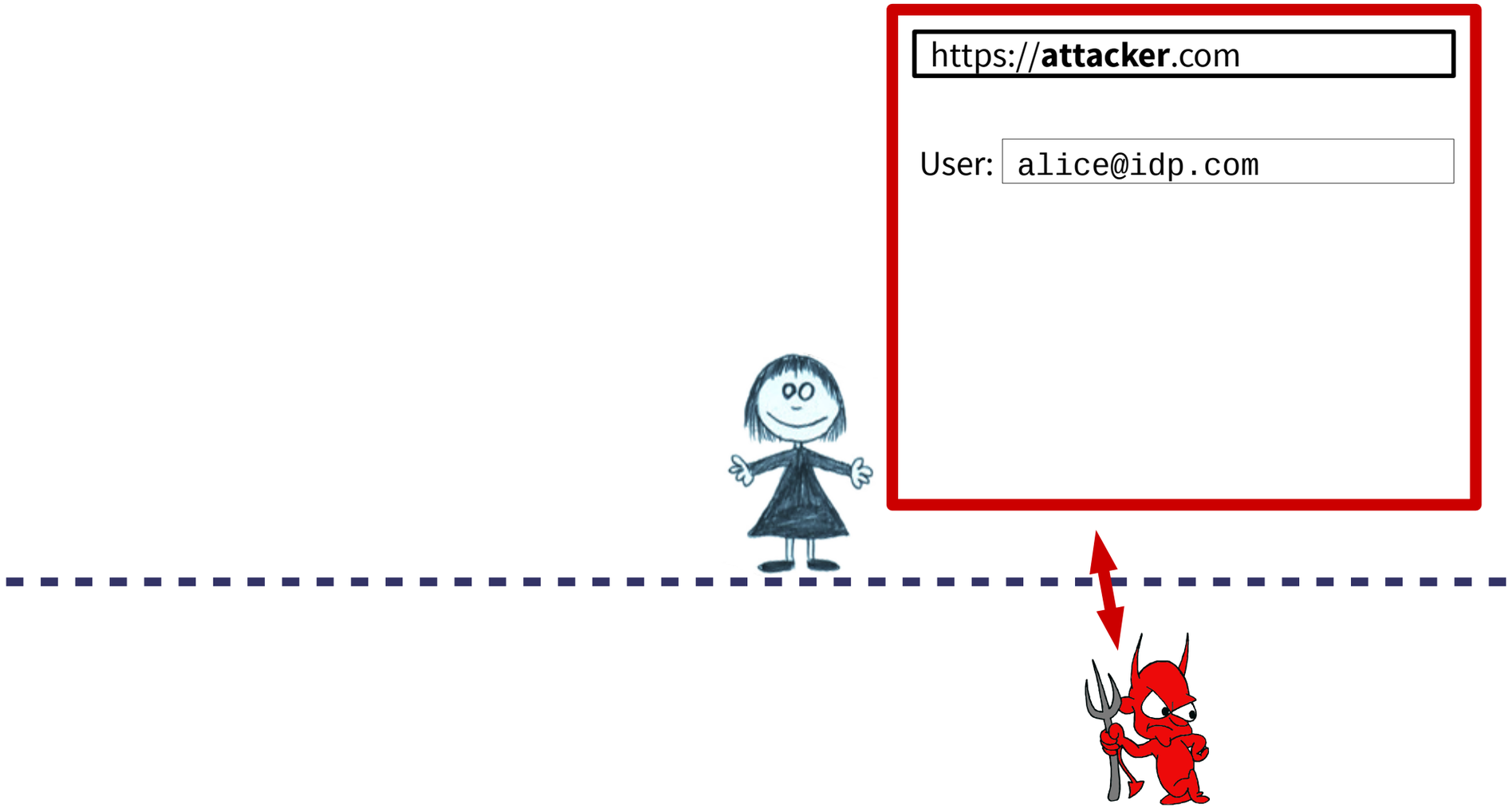


Why Forwarder?

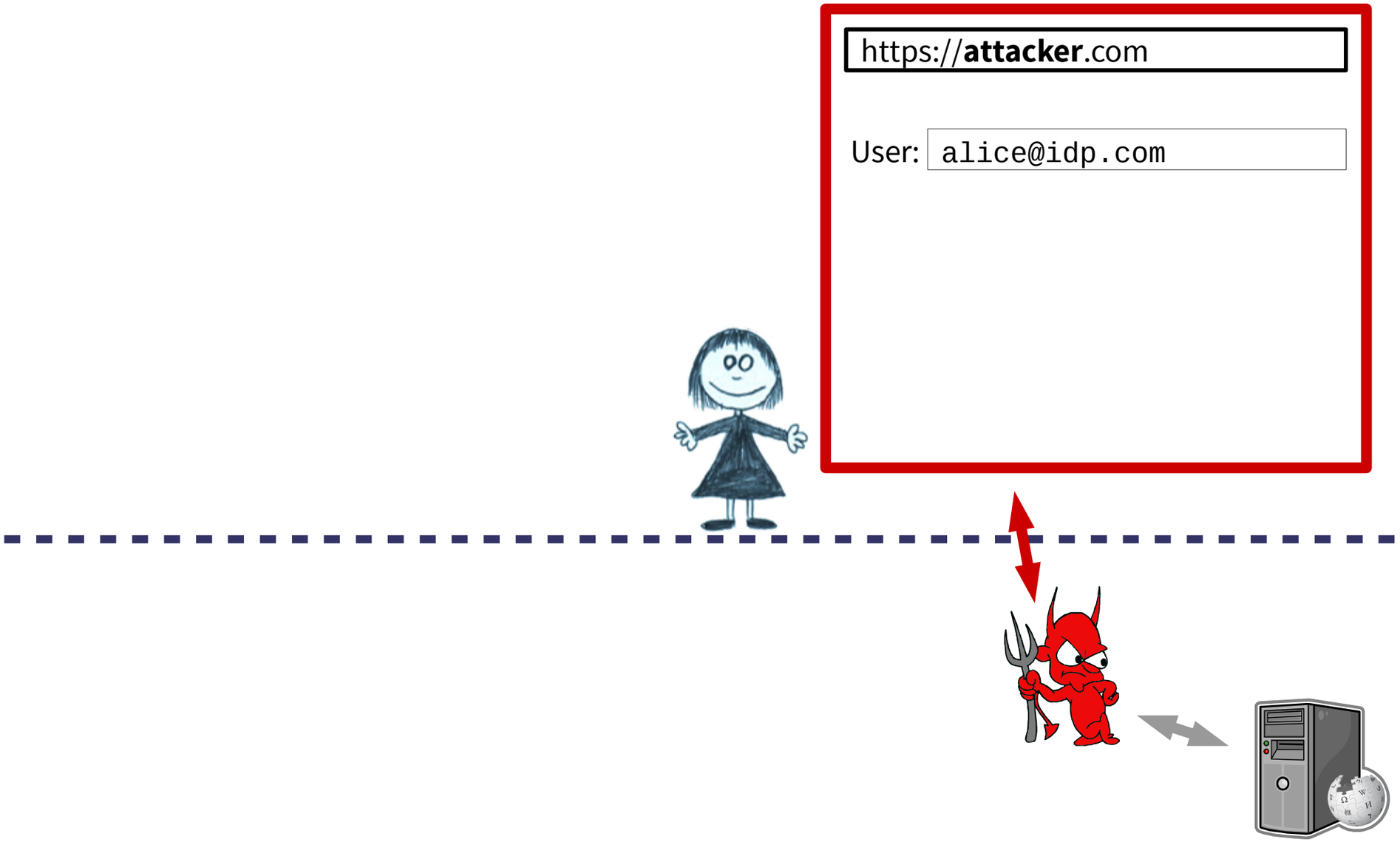
Why Forwarder?



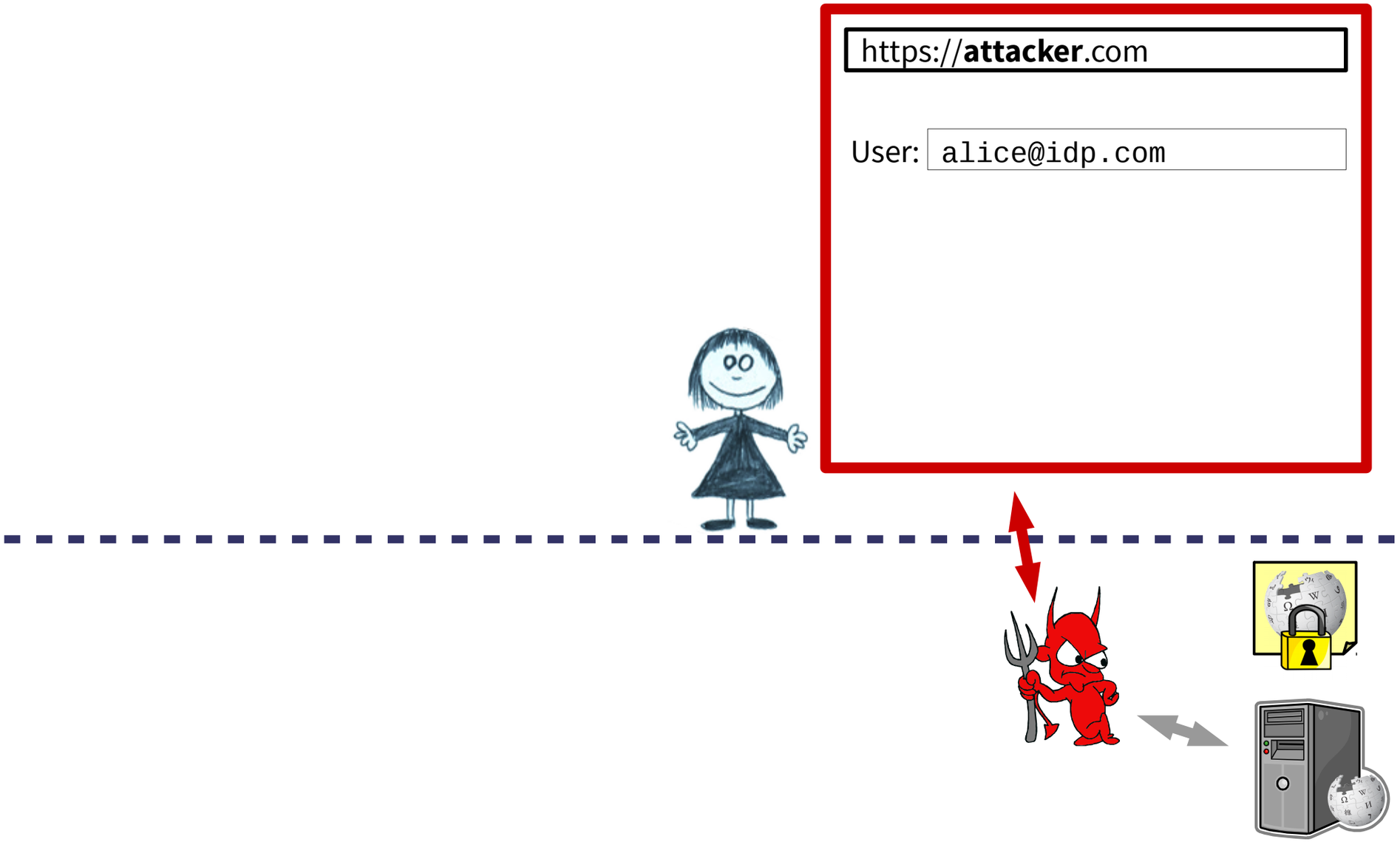
Why Forwarder?



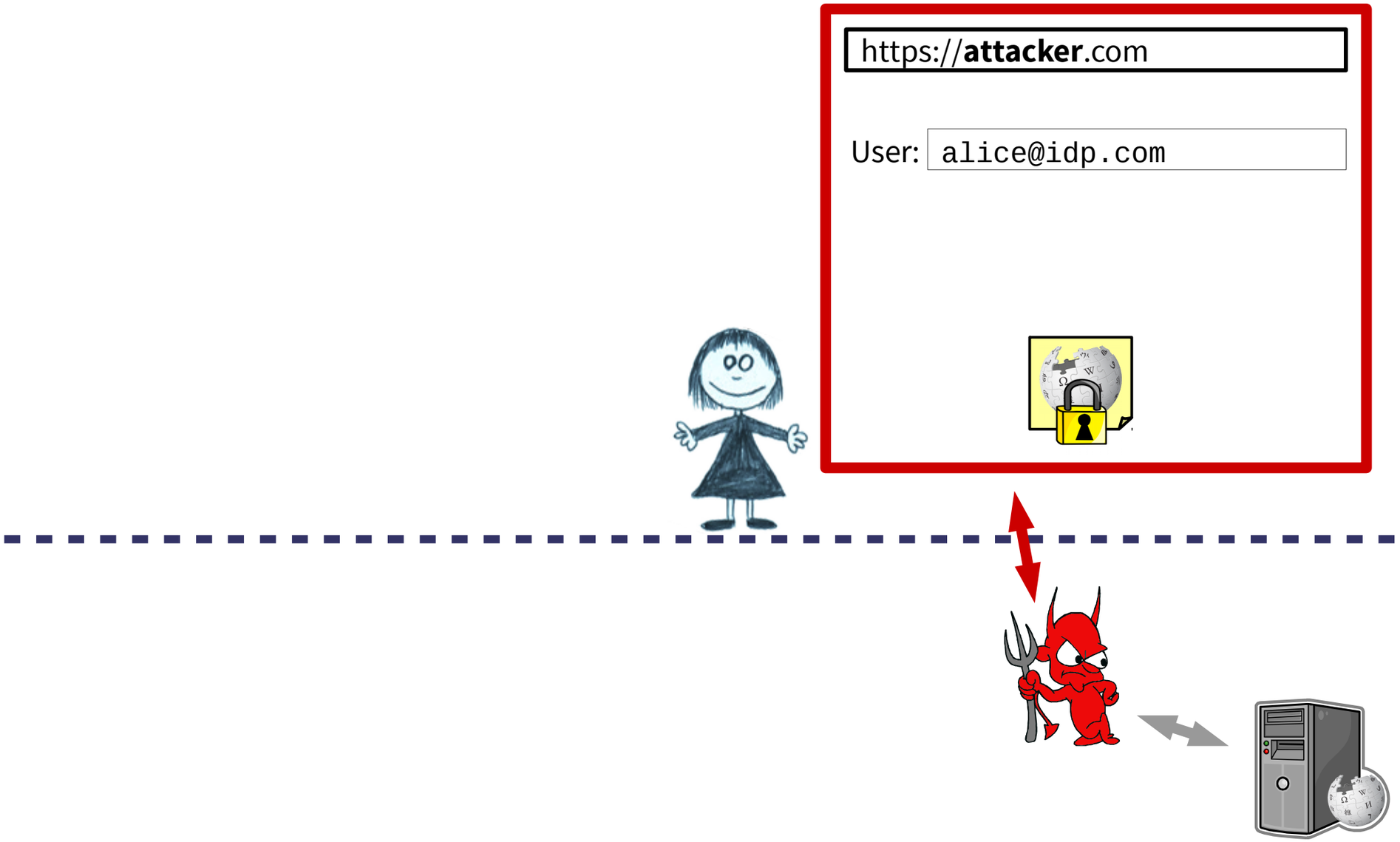
Why Forwarder?



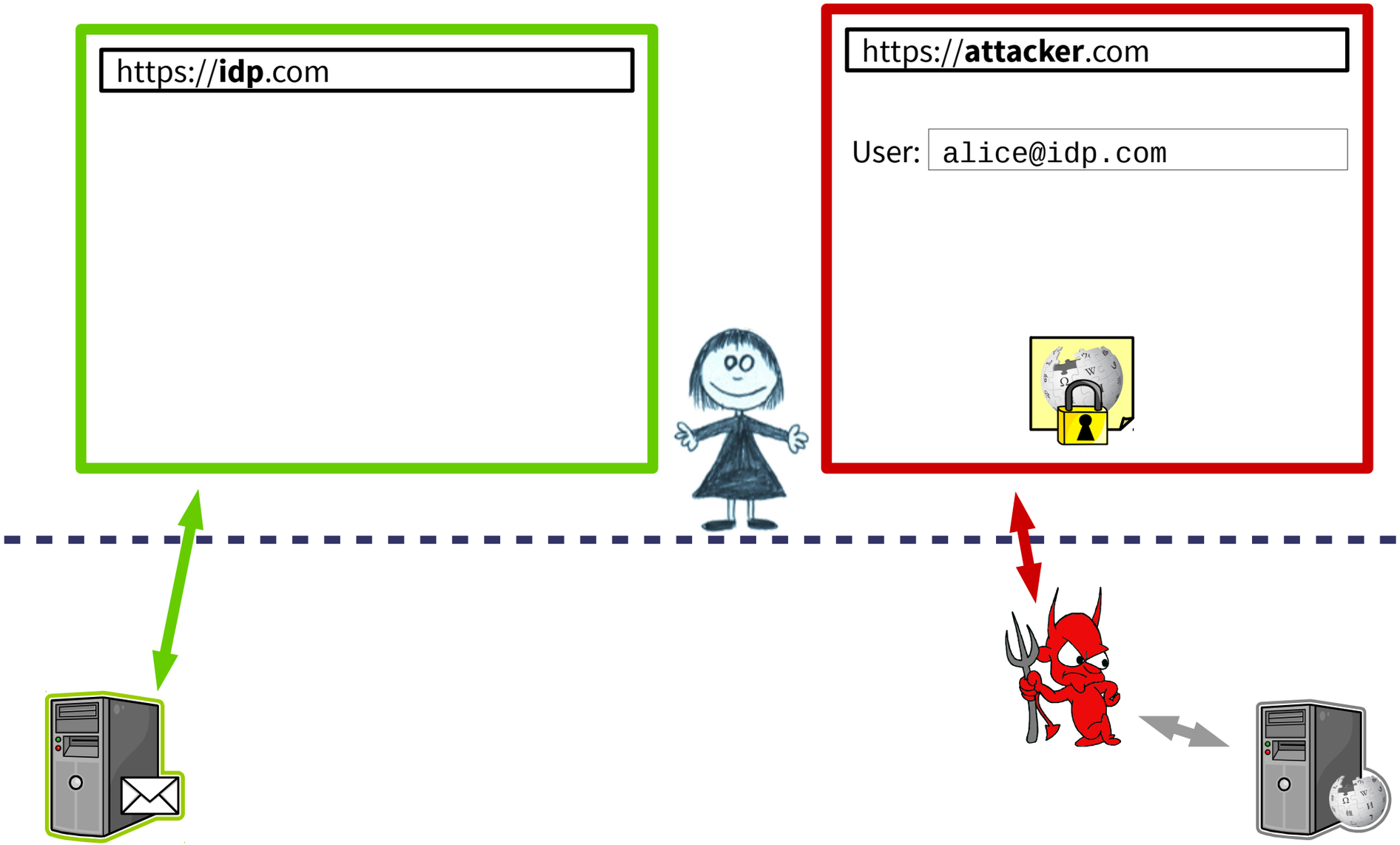
Why Forwarder?



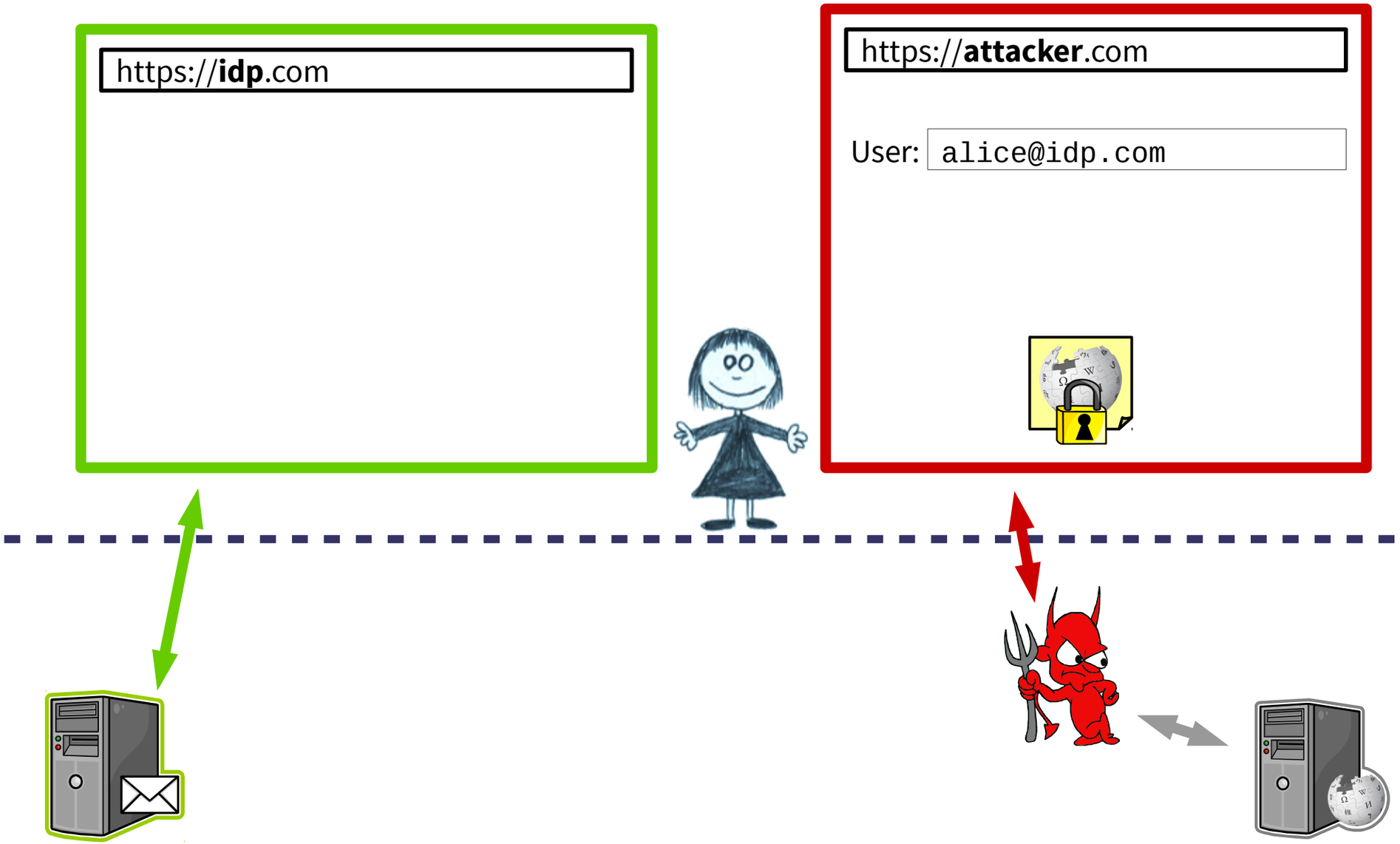
Why Forwarder?



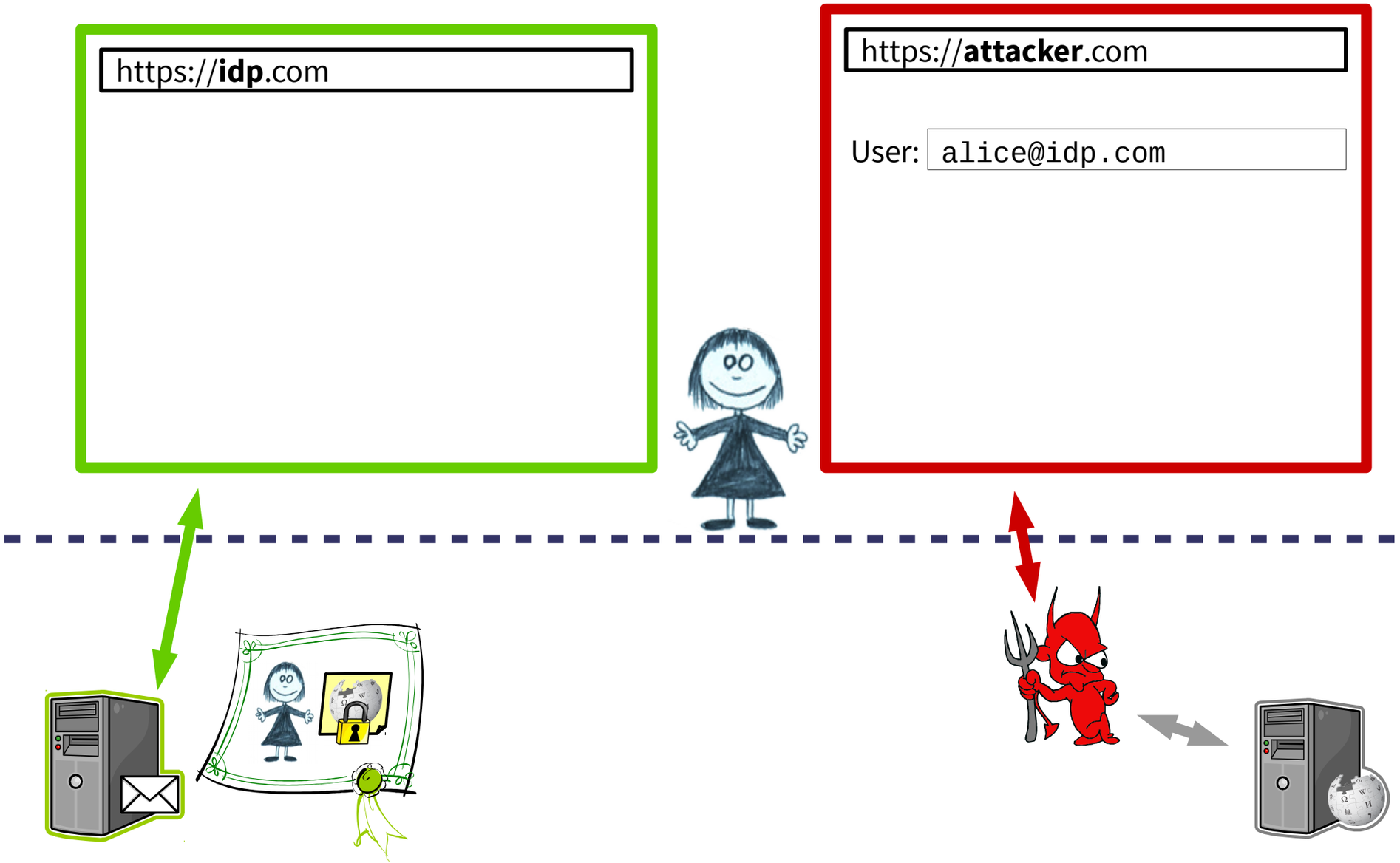
Why Forwarder?



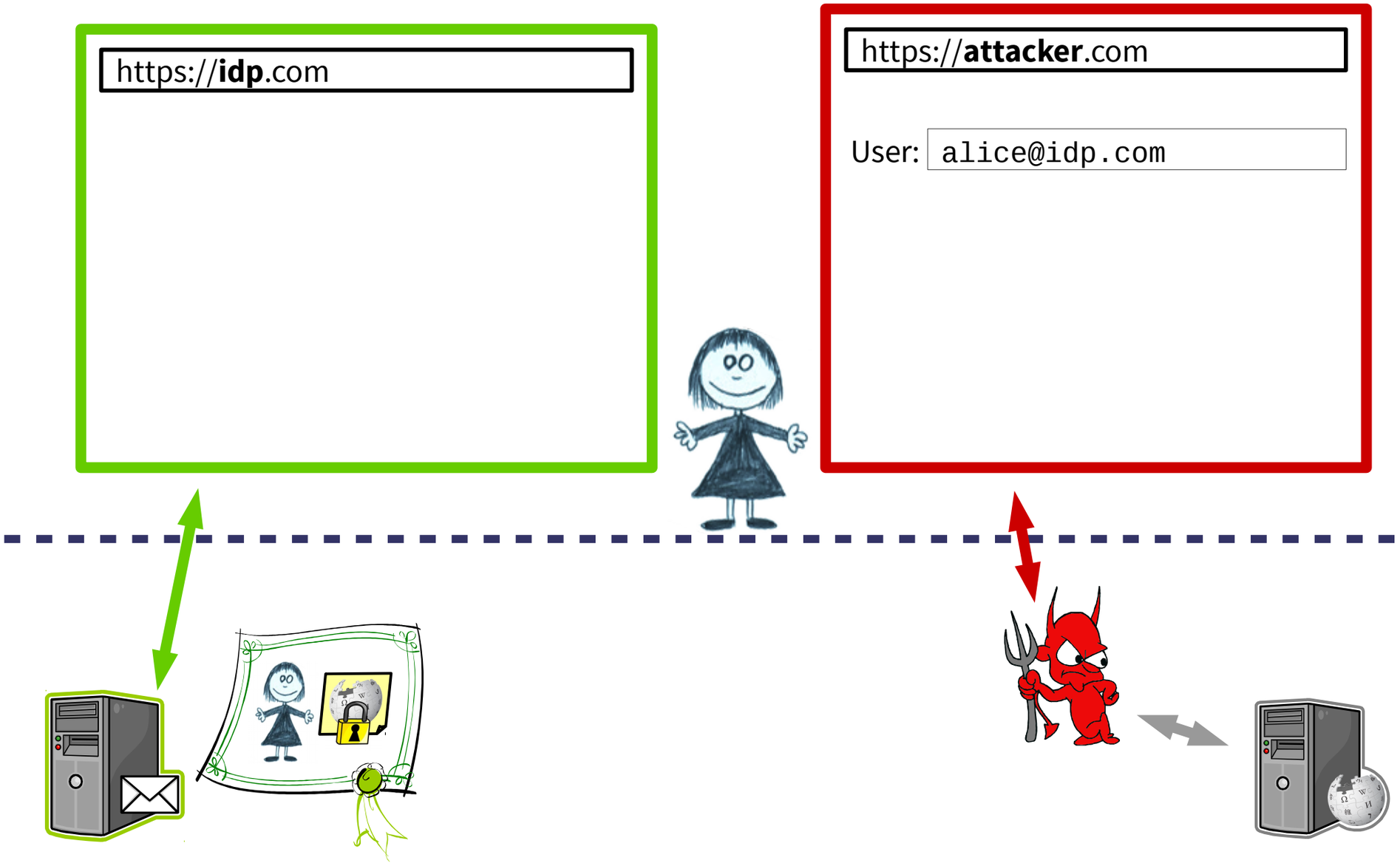
Why Forwarder?



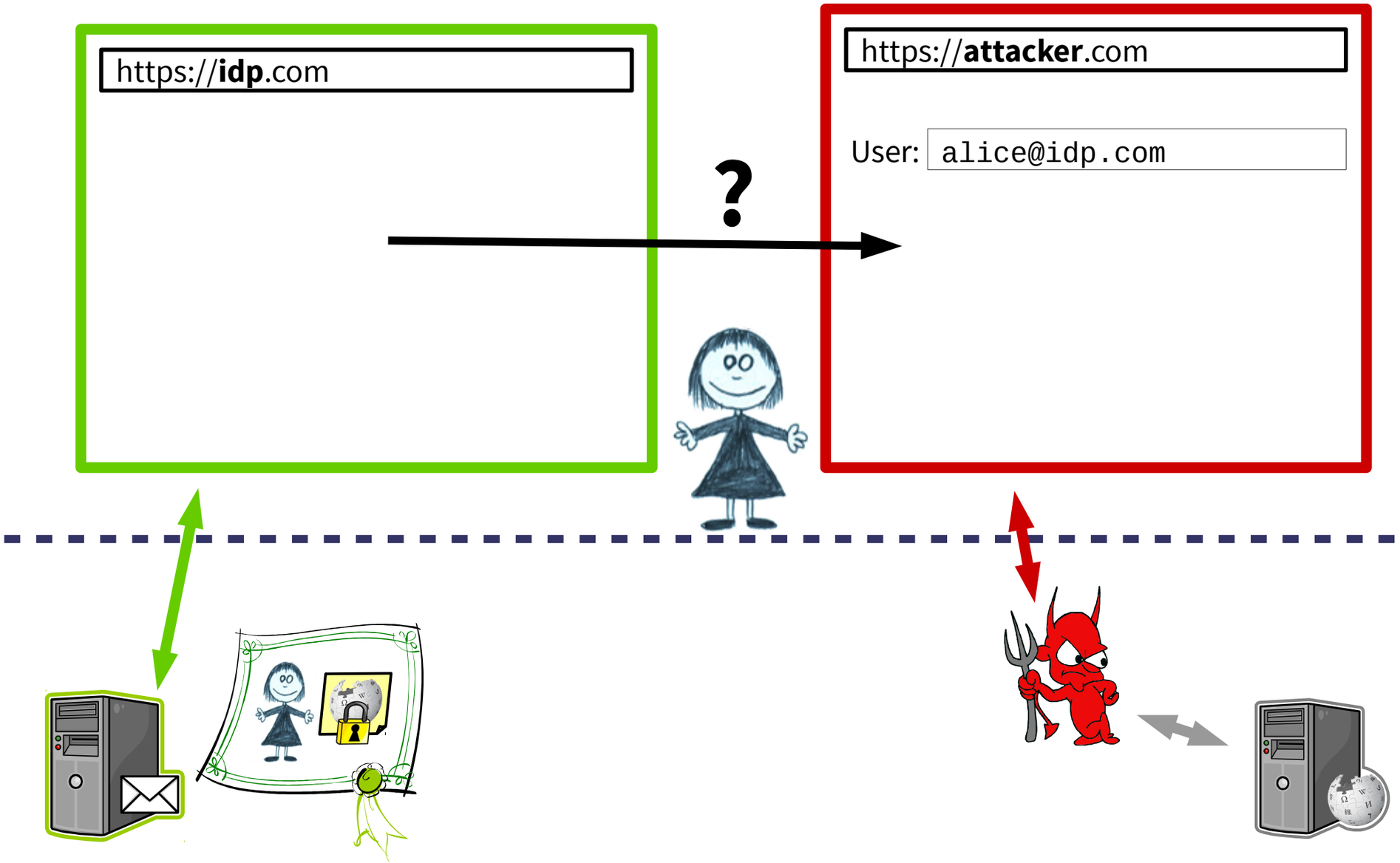
Why Forwarder?



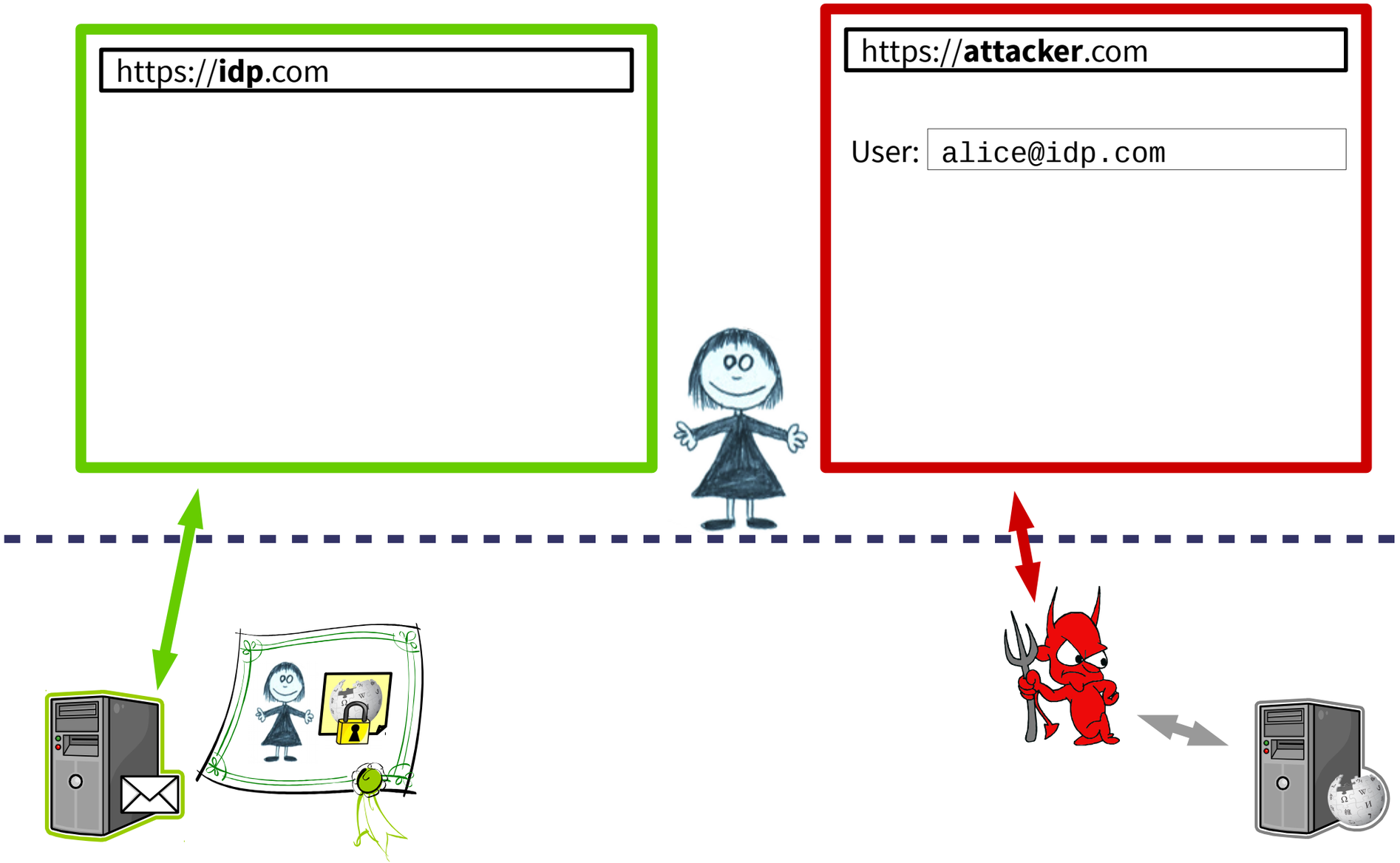
Why Forwarder?



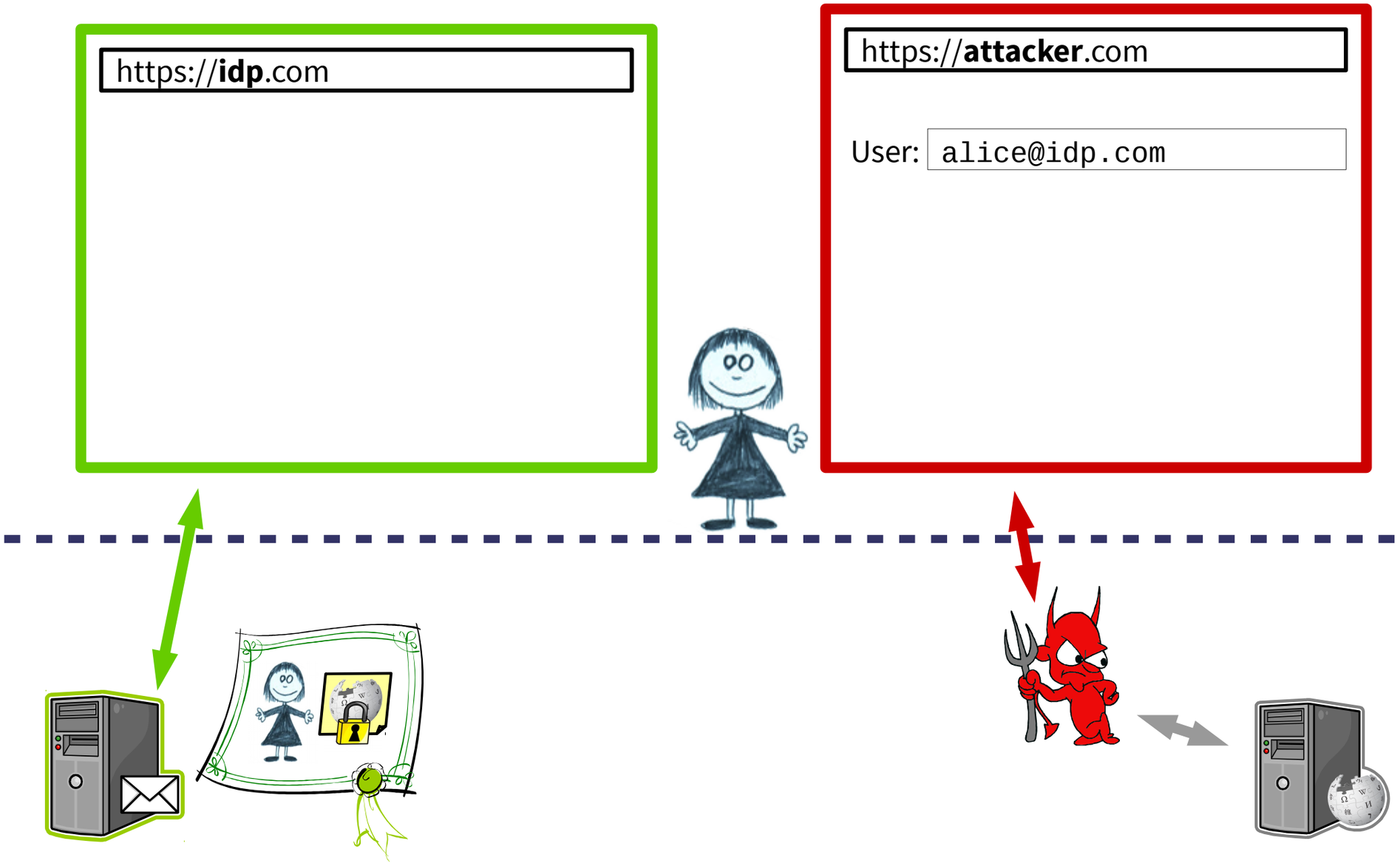
Why Forwarder?



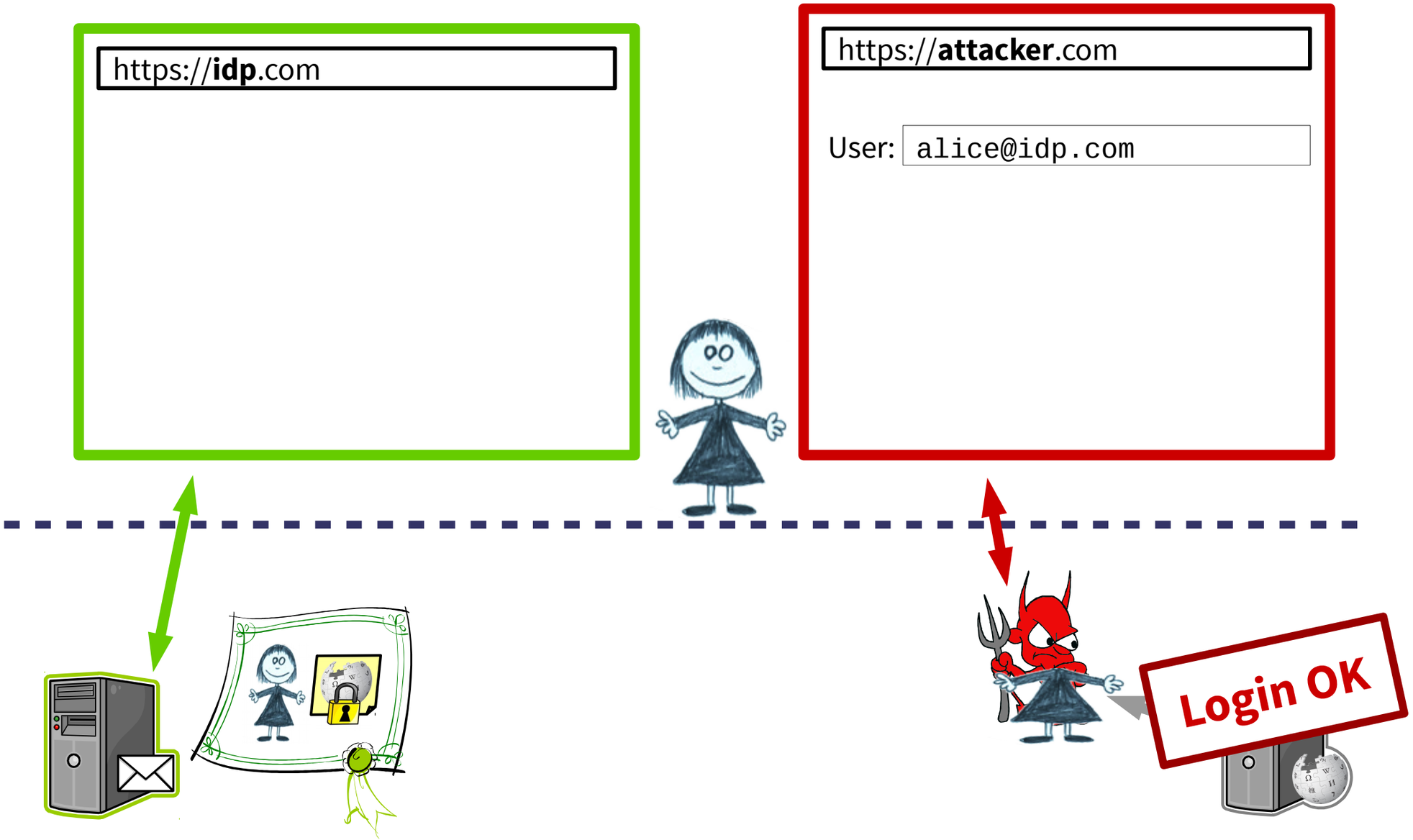
Why Forwarder?



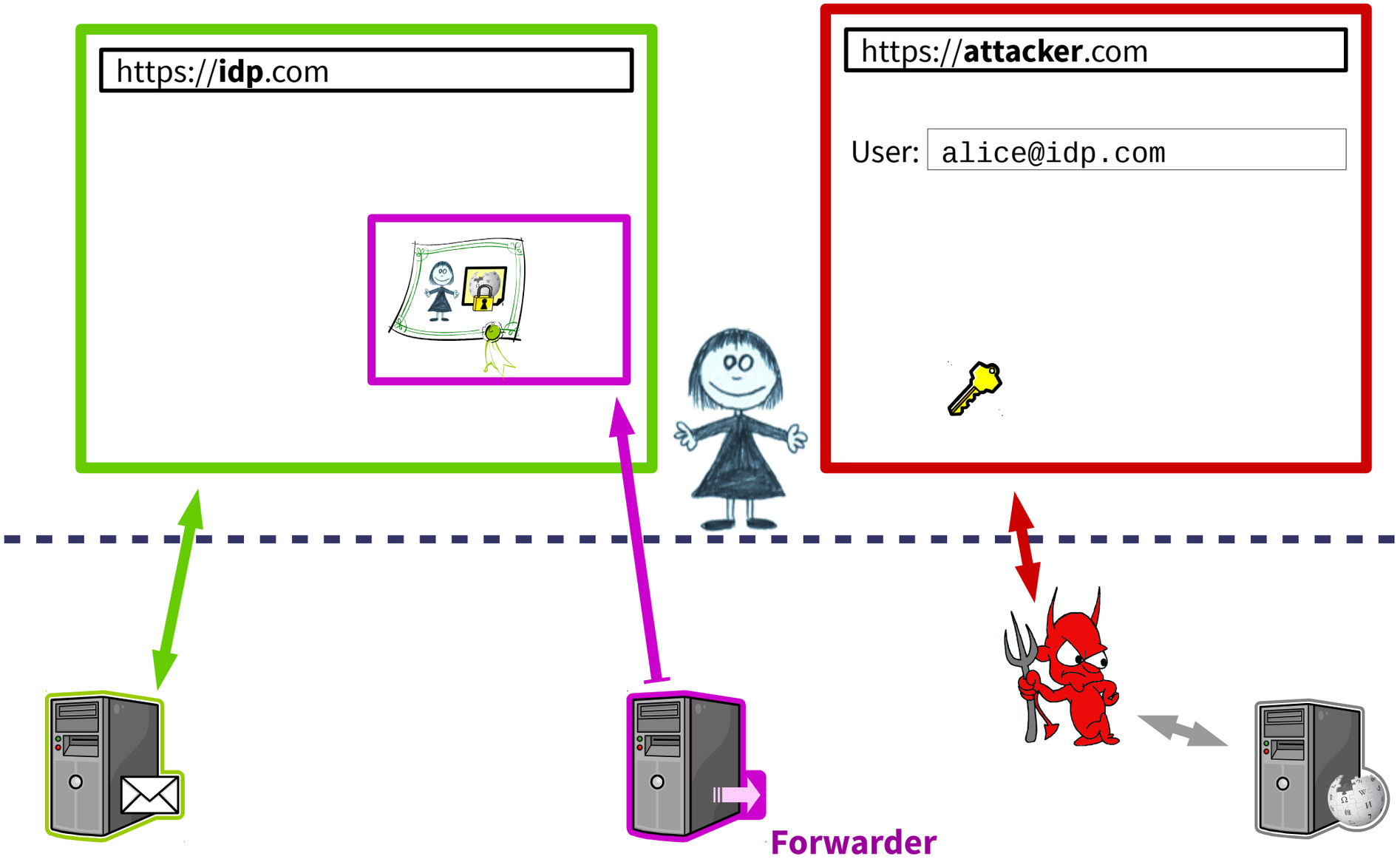
Why Forwarder?



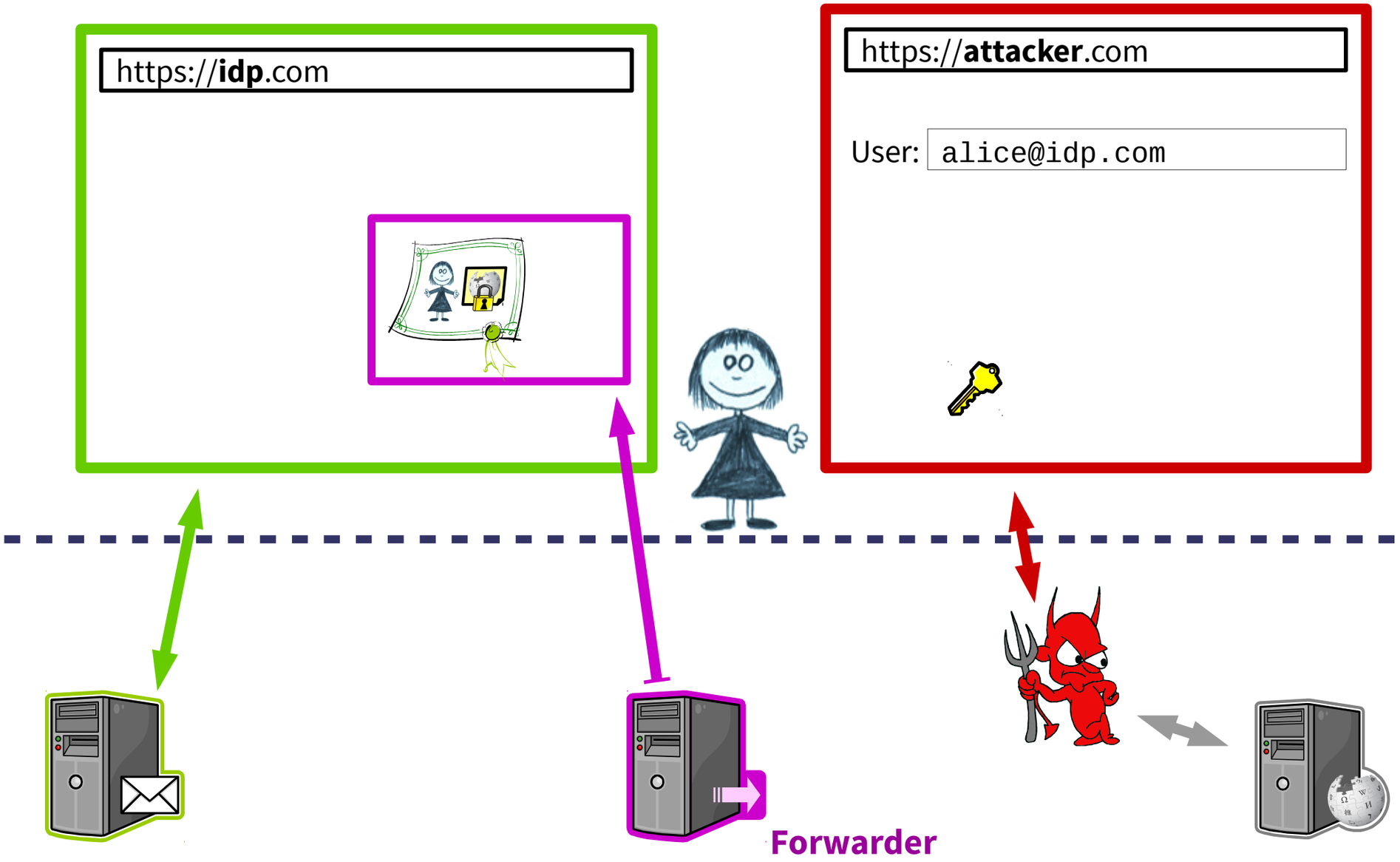
Why Forwarder?



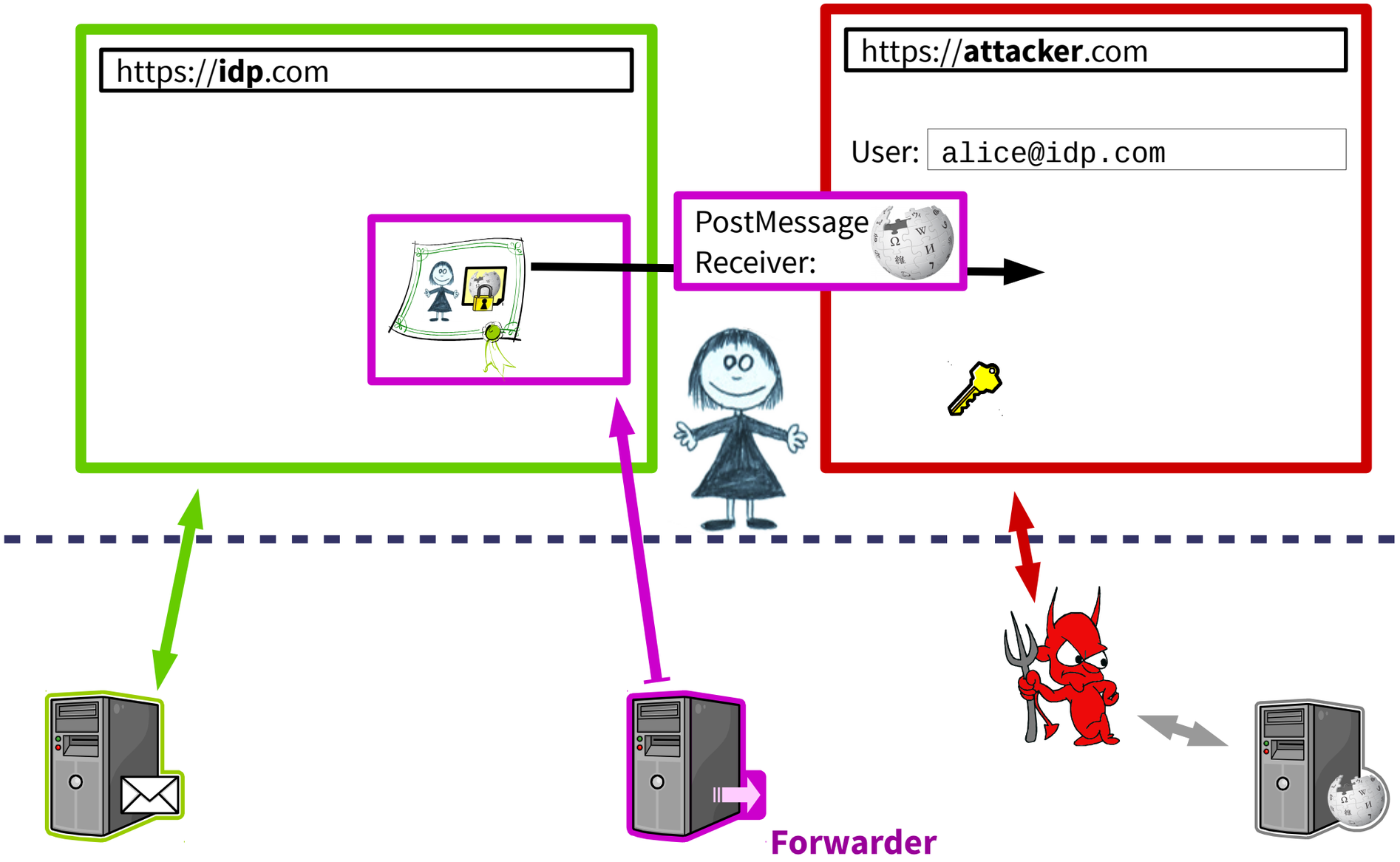
Why Forwarder?



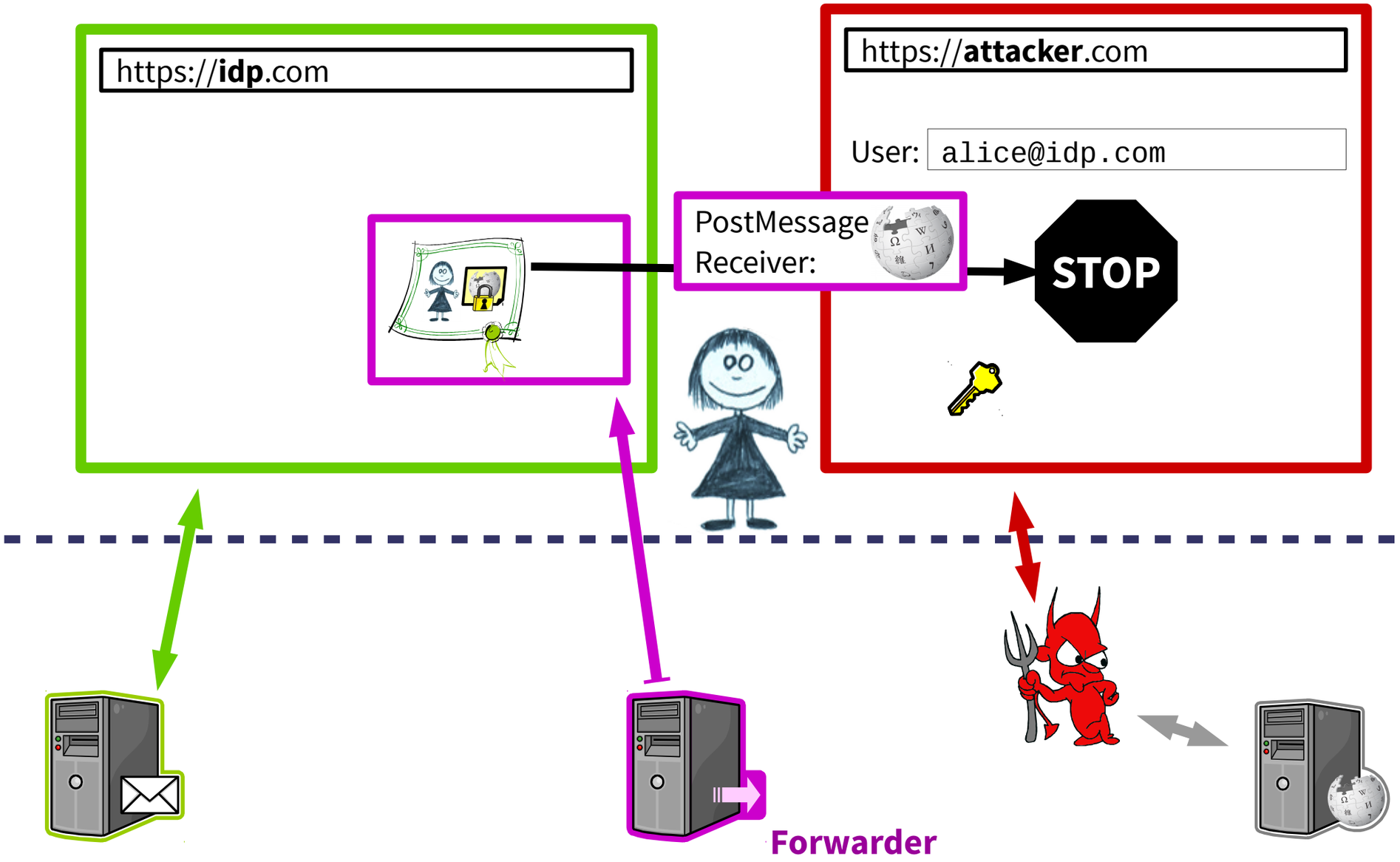
Why Forwarder?



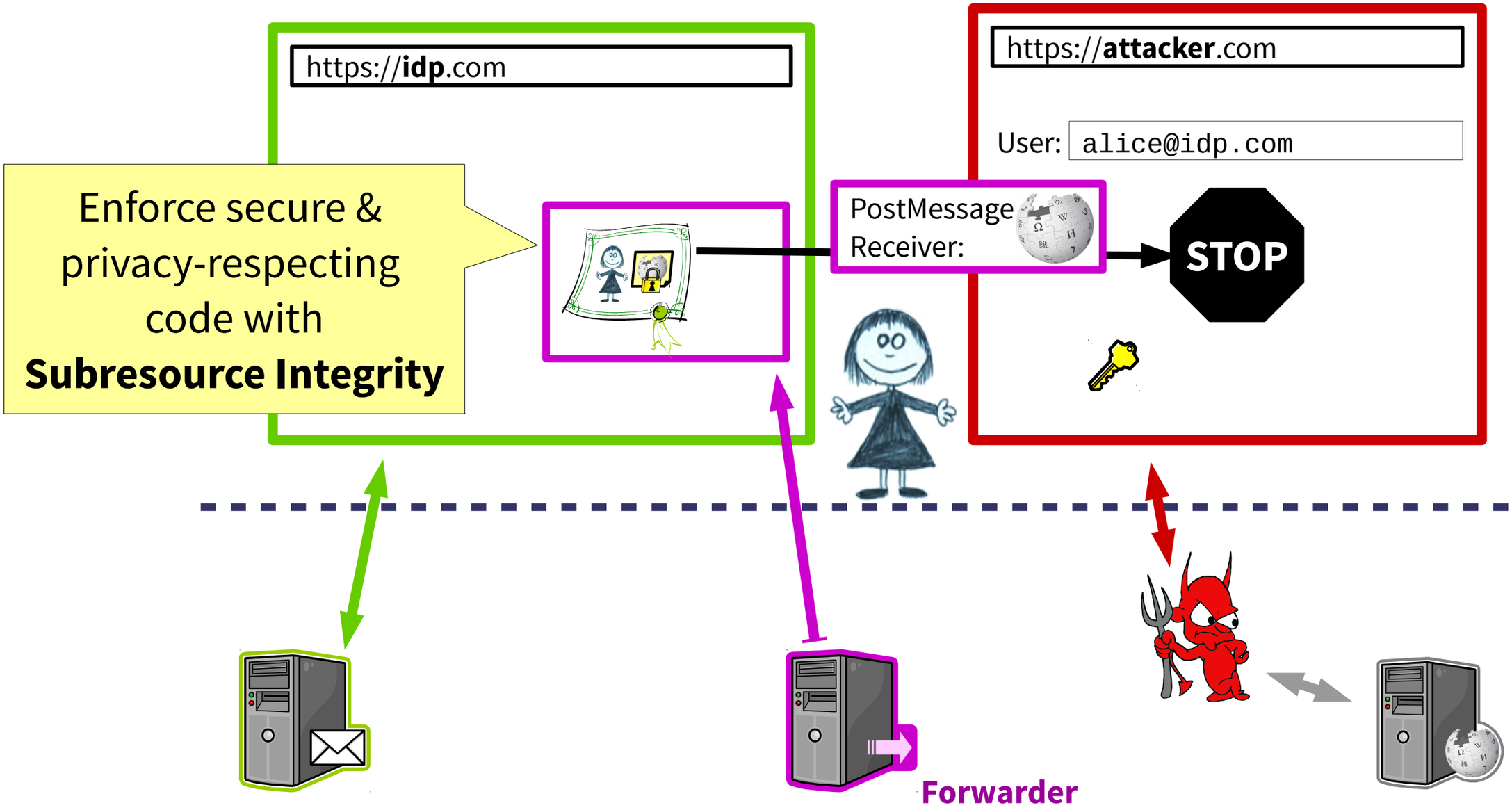
Why Forwarder?



Why Forwarder?



Why Forwarder?



SPRESSO

Secure, Privacy-Respecting Single Sign-On

Features:

- Strong privacy
- Strong authentication
- Open and decentralized
- Web standards compliant, native HTML5 

Formal proofs:

Privacy and authentication

→ https://spresso.me

What's the takeaway?

Various SSO Sytems

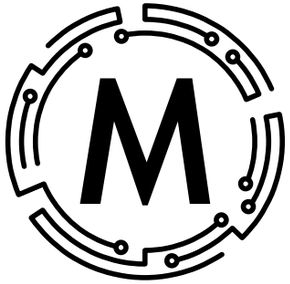
- OAuth 2.0 + OpenID Connect:
  - Formally proven
  - OK, if implemented correctly and *no privacy* works for you
- OAuth 1.0, OpenID: don‘t use
- BrowserID / Mozilla Persona: dead, broken privacy
- SPRESSO:
  - Formally proven
  - Currently just a Proof-of-Concept

Developers, Developers, Developers:

- Use libraries (e.g., pyoidc)
- never implement your own OAuth/etc.
- RFCs hard to read, information spread across several documents, not written clearly, not always up to date
  - yet still important reference

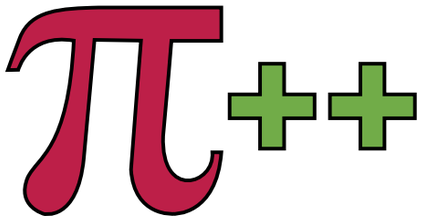
Thank You!

Find us at the **Maschinendeck** assembly in Hall 3 …



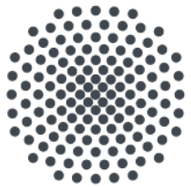
Maschinendeck.org

… or at the next **Pi and More,** January 14, in Krefeld …



PiAndMore.de

… or at the **University of Stuttgart** (starting January)



Universität Stuttgart