Token Binding over HTTPS

Vinod Anupam

IETF 94 ● Yokohama ● November 2015

Overview

1. Recap (for newcomers)
2. Changes to tokbind-https
3. Threat model

Overview

1. Recap (for newcomers)
2. Changes to tokbind-https
3. Threat model

Recap: The Token Binding Header

Recap: The Token Binding Header

provided_token_binding: {
signature(EKM),
public_key$_{\text{example.com}}$
}

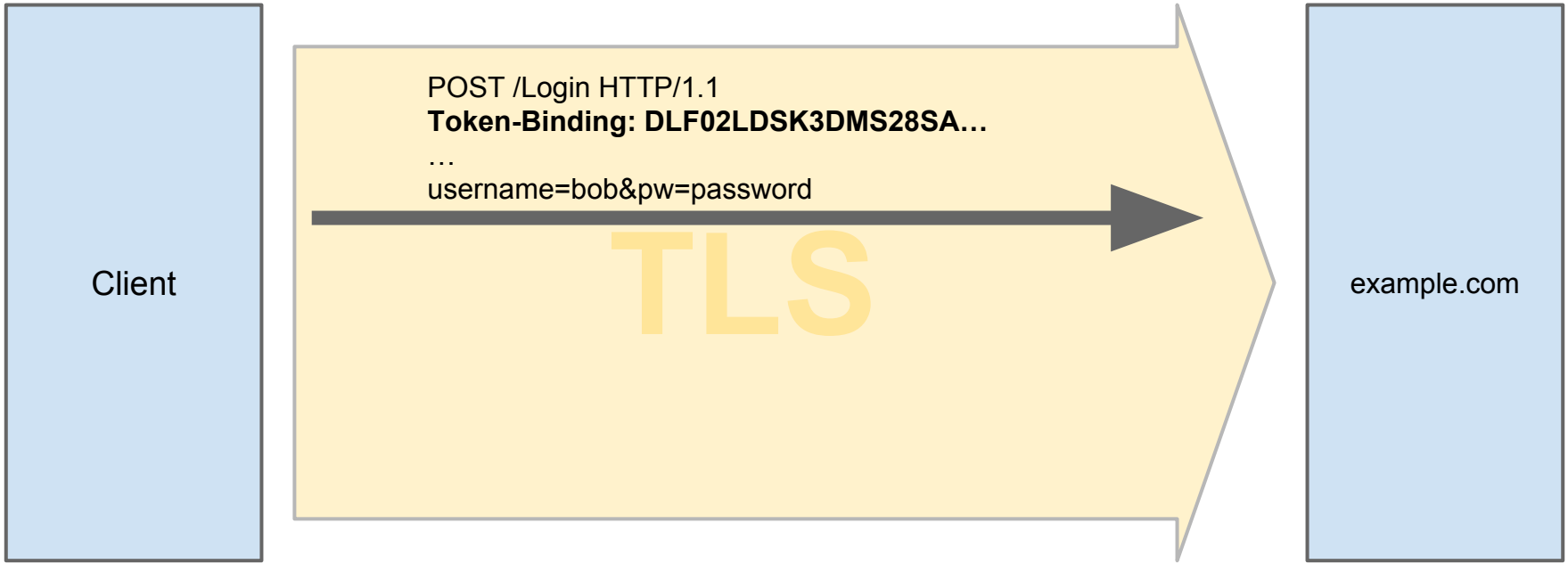
- client uses different key pairs for different servers
- client protects private keys
- client discloses public key of pair to server
- client proves to server that it controls private key

Client

example.com

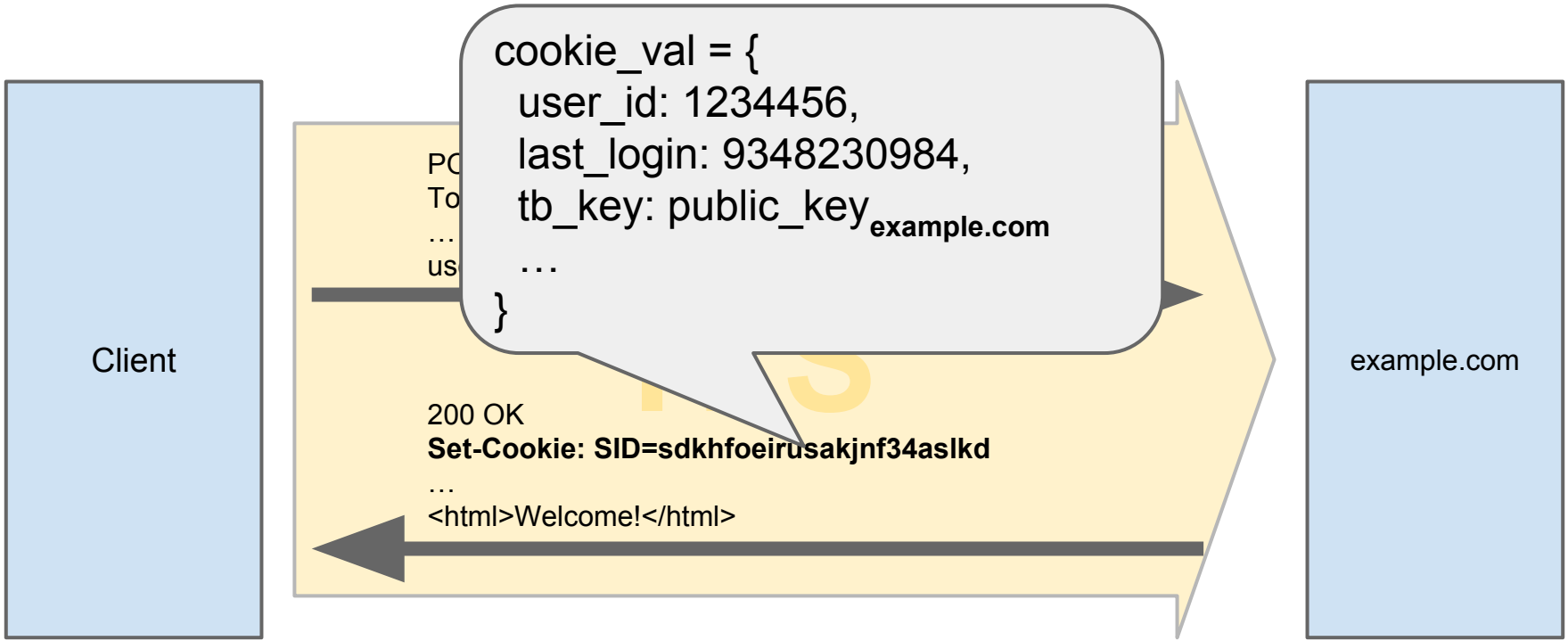
Example: Sending Header

- Client transmits Token Binding key



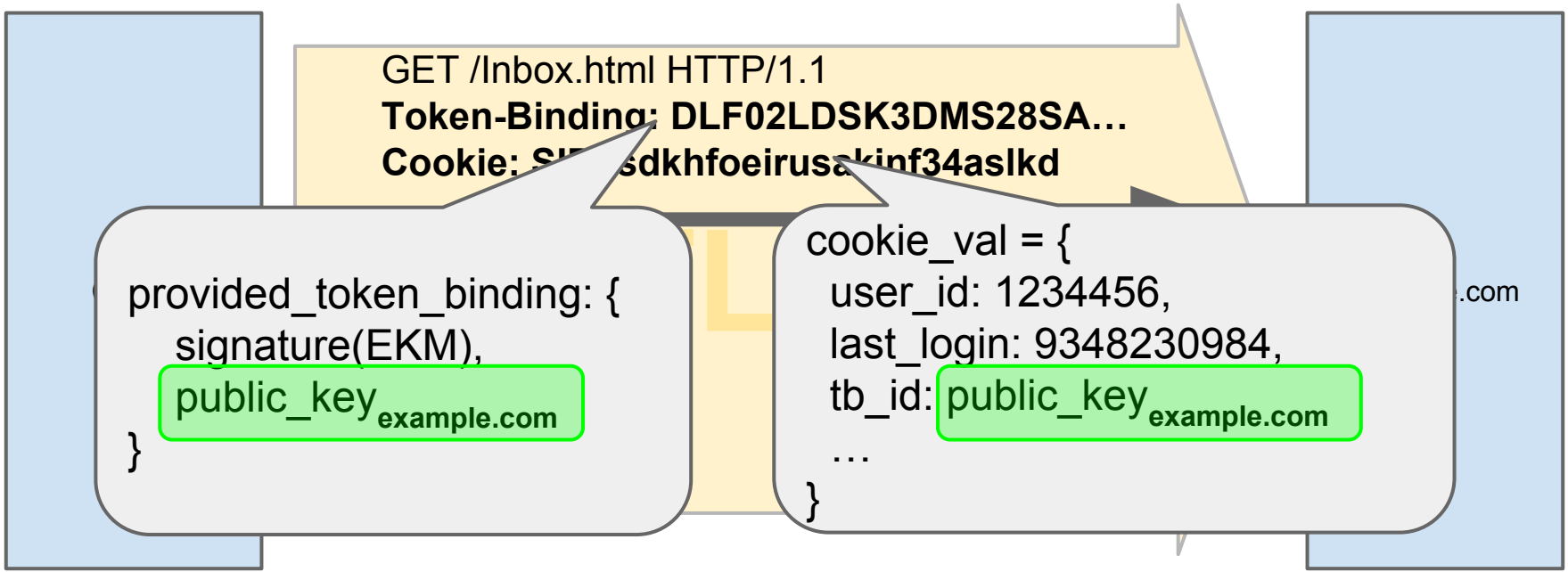
Example: Binding Cookies

- Server binds tokens to Token Binding key

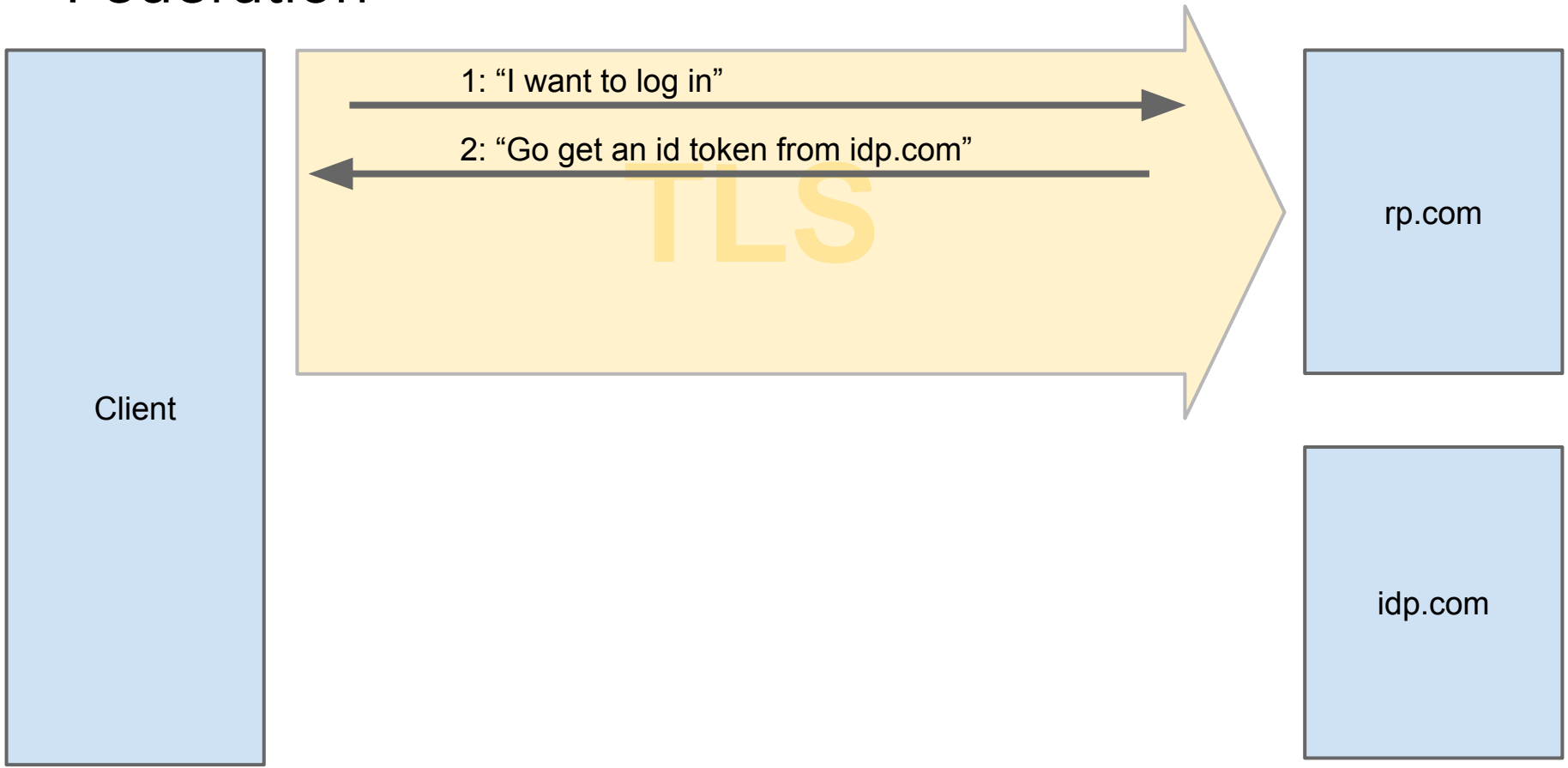


Example: Verifying Cookies

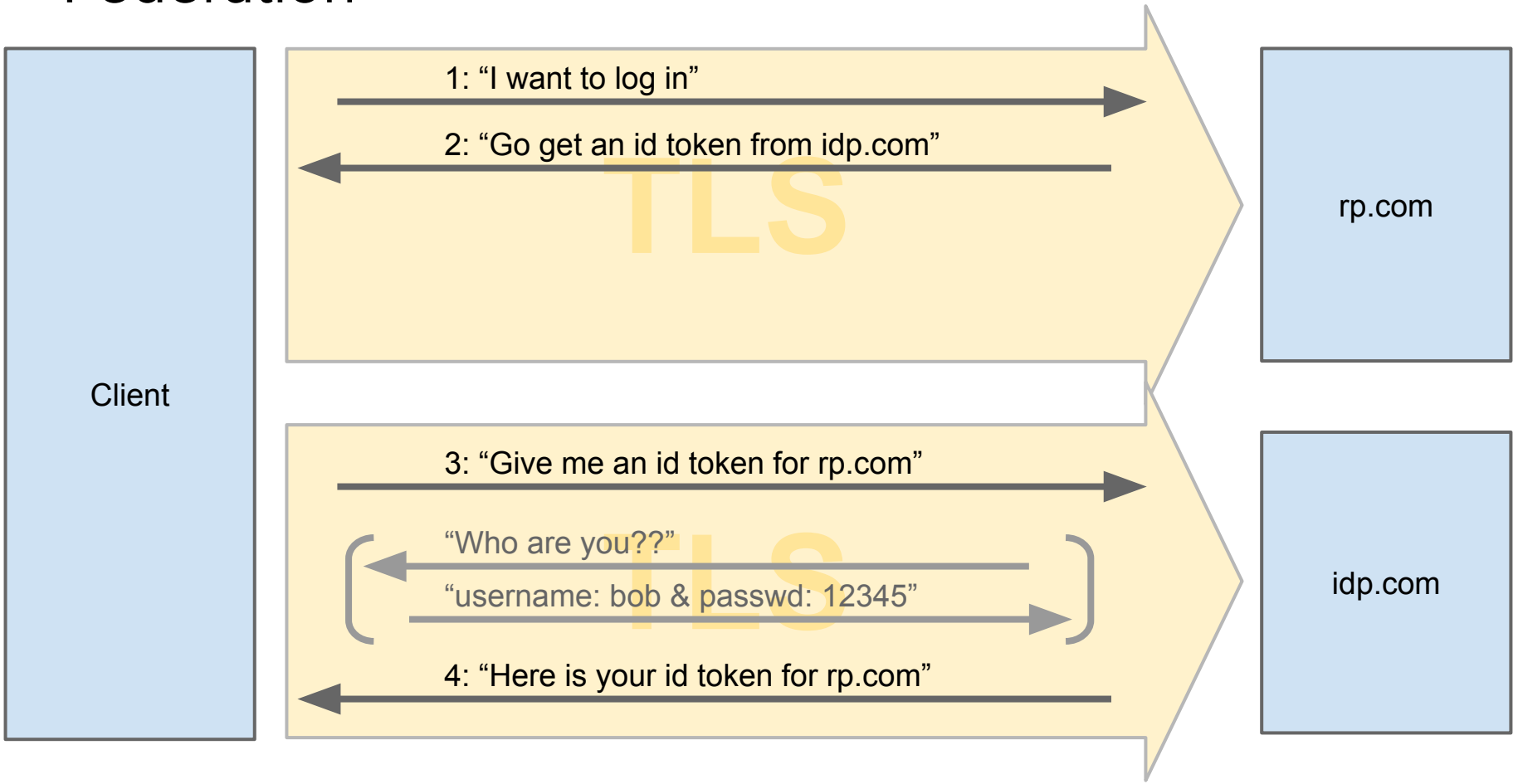
- Server confirms that cookie matches Token Binding key



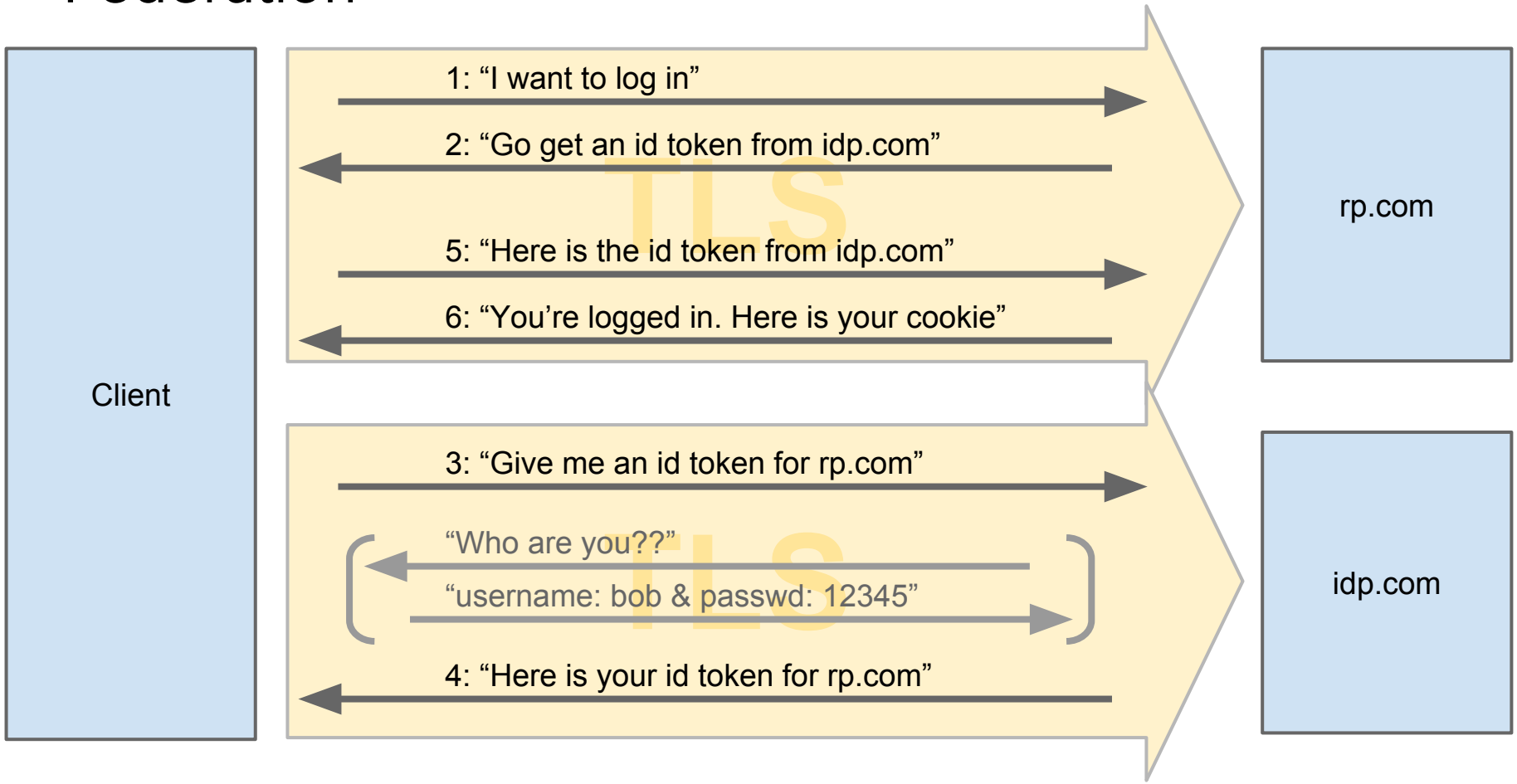
Federation


Client
1: “I want to log in”
2: “Go get an id token from idp.com”
TLS
rp.com
idp.com

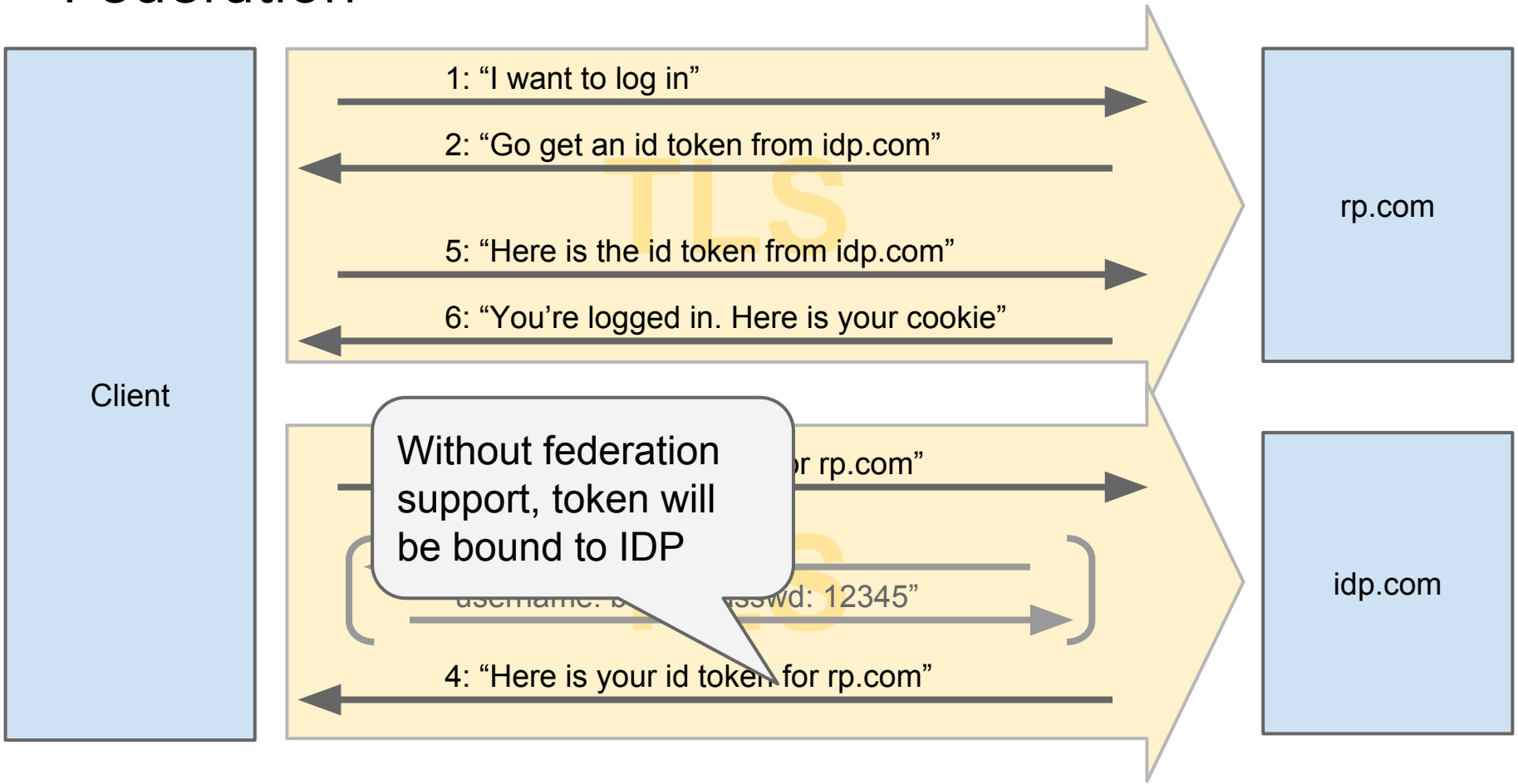
Federation


Client
1: “I want to log in”
2: “Go get an id token from idp.com”
TLS
rp.com
3: “Give me an id token for rp.com”
“Who are you??”
“username: bob & passwd: 12345”
TLS
4: “Here is your id token for rp.com”
idp.com

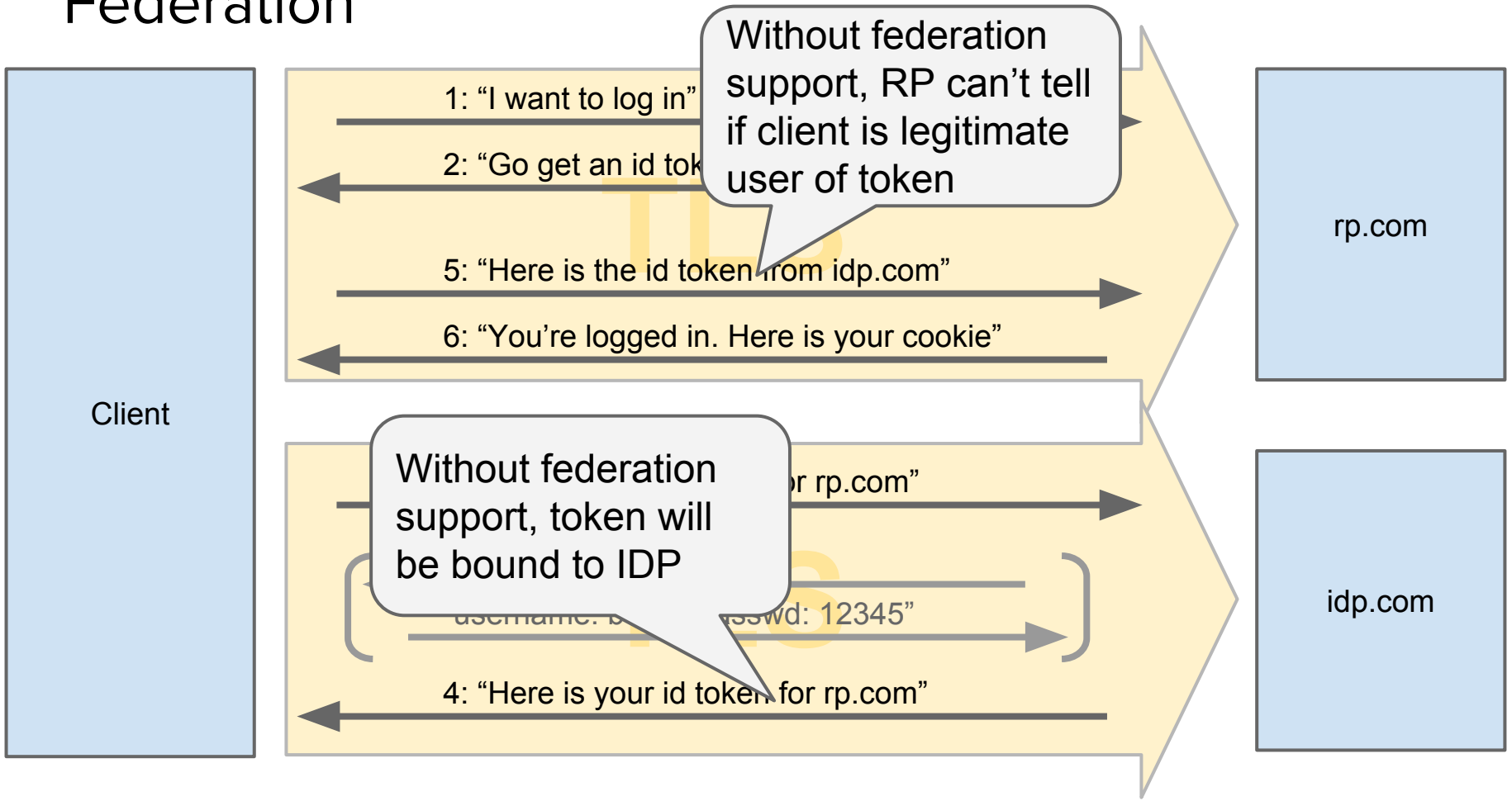
Federation


Client
1: “I want to log in”
2: “Go get an id token from idp.com”
TLS
5: “Here is the id token from idp.com”
6: “You’re logged in. Here is your cookie”
rp.com
3: “Give me an id token for rp.com”
“Who are you??”
TLS
“username: bob & passwd: 12345”
4: “Here is your id token for rp.com”
idp.com

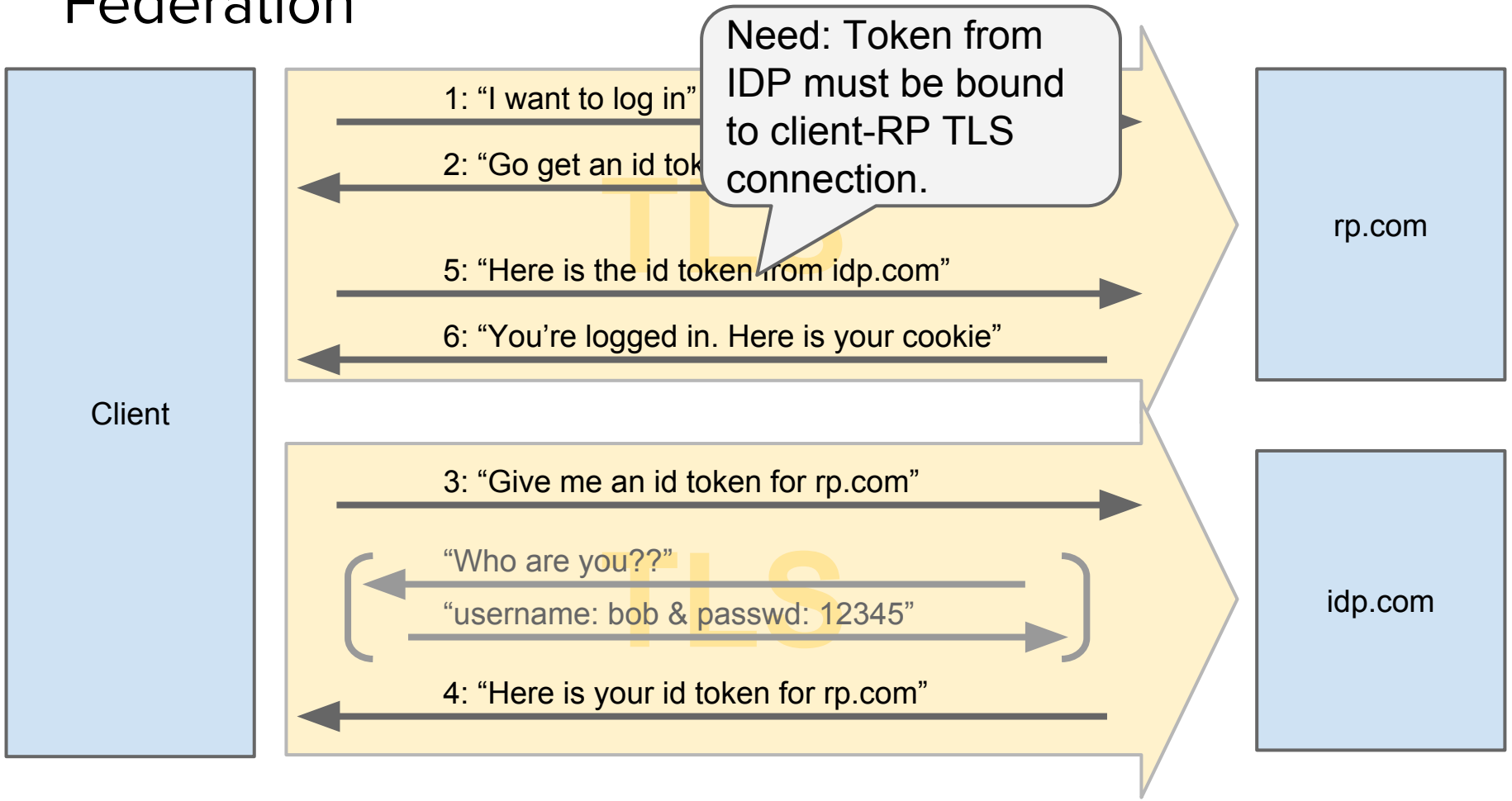
Federation


Client
1: “I want to log in”
2: “Go get an id token from idp.com”
TLS
5: “Here is the id token from idp.com”
6: “You’re logged in. Here is your cookie”
rp.com
Without federation support, token will be bound to IDP
r rp.com”
wd: 12345”
4: “Here is your id token for rp.com”
idp.com

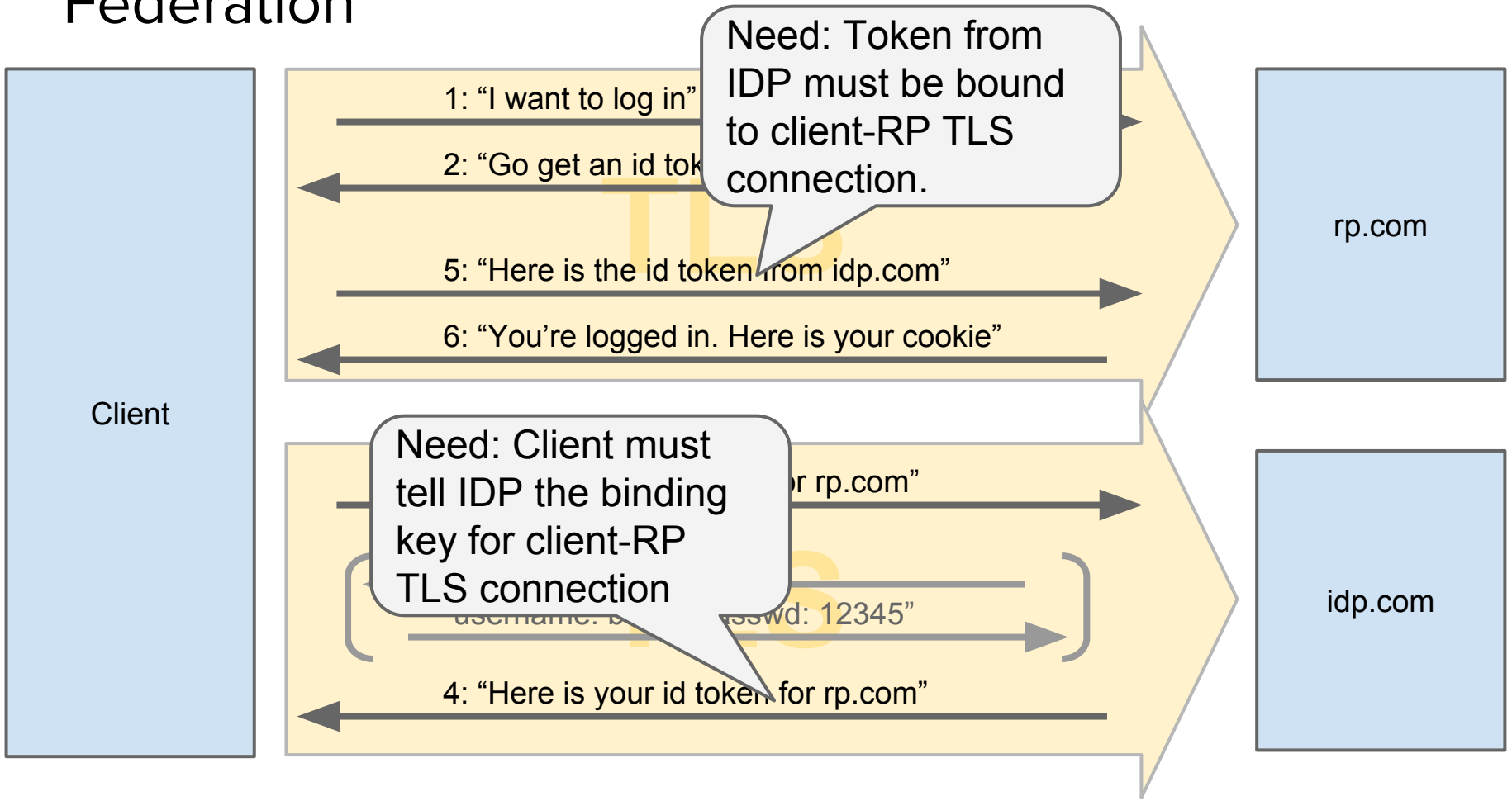
Federation


Client
1: “I want to log in”
2: “Go get an id tok
Without federation support, RP can’t tell if client is legitimate user of token
5: “Here is the id token from idp.com”
6: “You’re logged in. Here is your cookie”
rp.com
Without federation support, token will be bound to IDP
r rp.com”
swd: 12345”
4: “Here is your id token for rp.com”
idp.com

Federation


Need: Token from IDP must be bound to client-RP TLS connection.
Client
1: “I want to log in”
2: “Go get an id tok
5: “Here is the id token from idp.com”
6: “You’re logged in. Here is your cookie”
rp.com
3: “Give me an id token for rp.com”
“Who are you??”
“username: bob & passwd: 12345”
4: “Here is your id token for rp.com”
idp.com

Federation


Client
1: “I want to log in”
2: “Go get an id tok
Need: Token from IDP must be bound to client-RP TLS connection.
5: “Here is the id token from idp.com”
6: “You’re logged in. Here is your cookie”
rp.com
Need: Client must tell IDP the binding key for client-RP TLS connection
r rp.com”
swd: 12345”
4: “Here is your id token for rp.com”
idp.com

How to Trigger Referred Token Bindings?

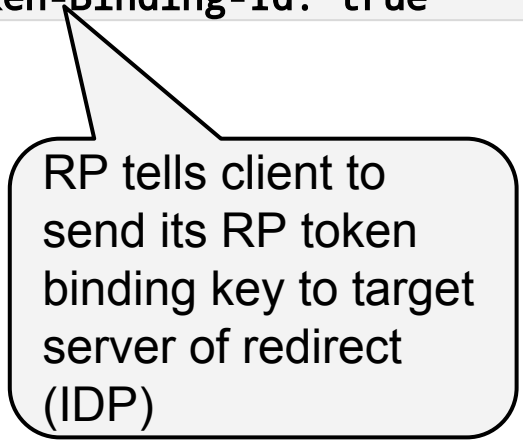
Relying Party uses HTTP Redirect

```
302 Moved Temporarily
Location: https://idp.com/rp-login
Include-Referer-Token-Binding-Id: true
```

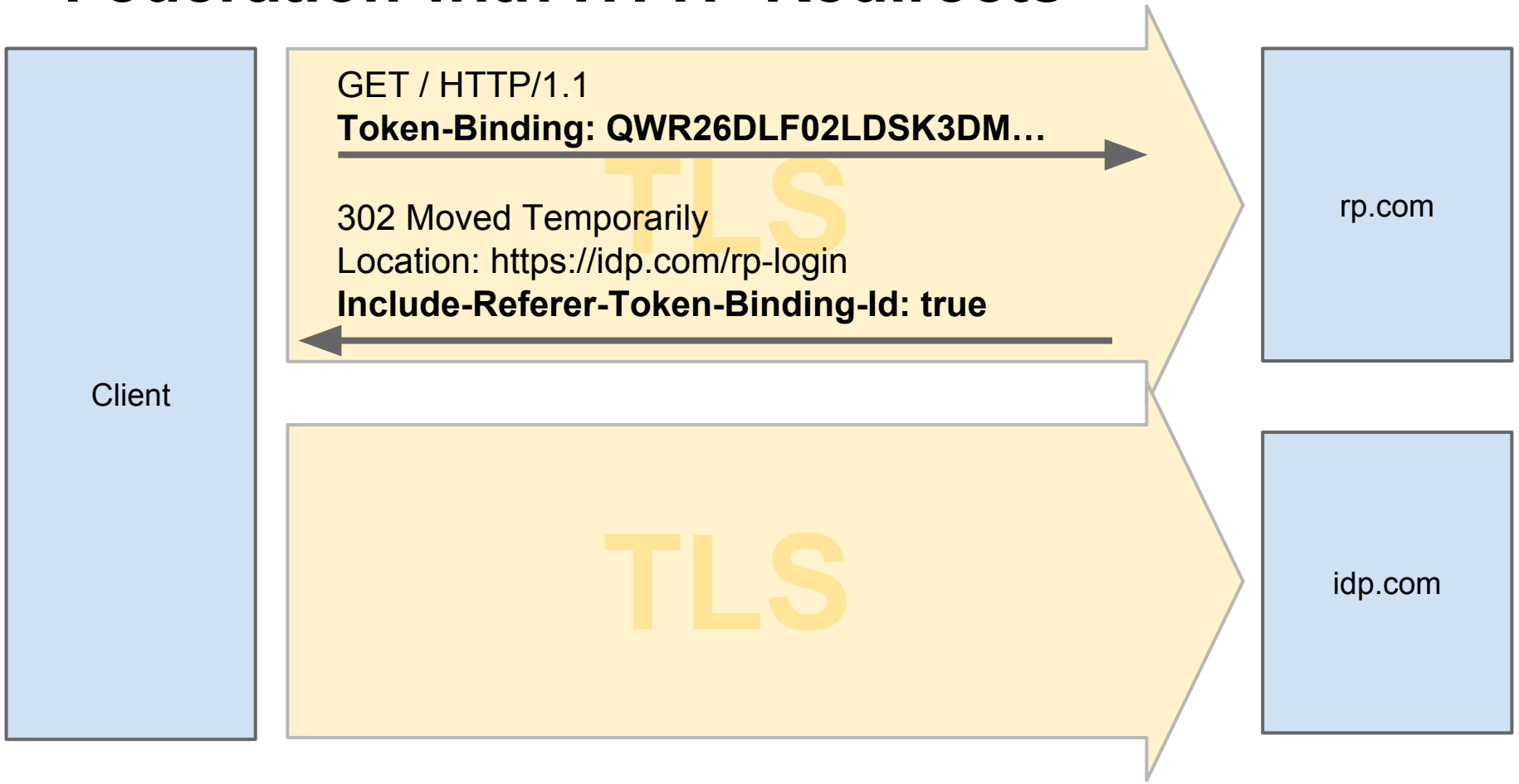
How to Trigger Referred Token Bindings?

Relying Party uses HTTP Redirect

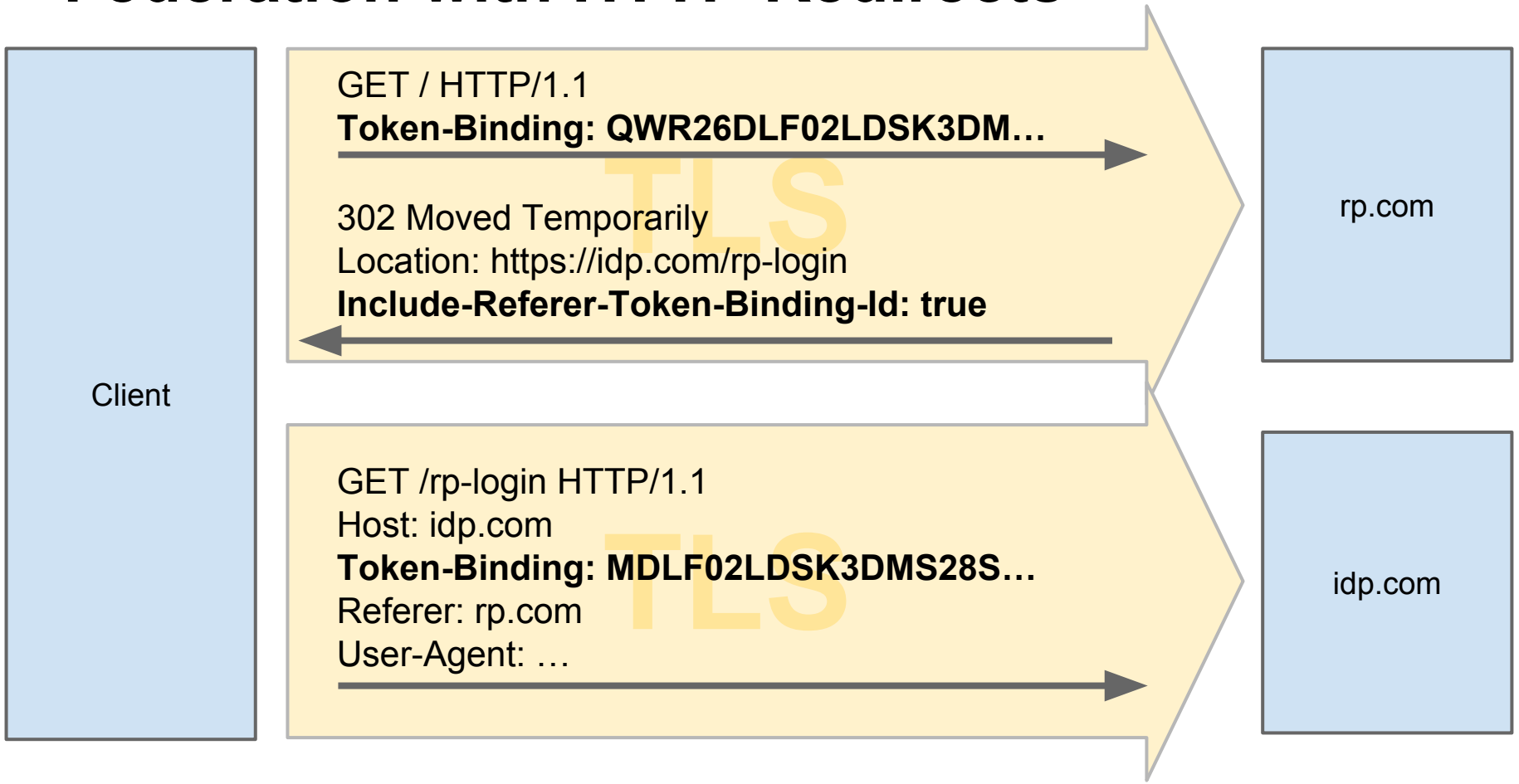
```
302 Moved Temporarily
Location: https://idp.com/rp-login
Include-Referer-Token-Binding-Id: true
```



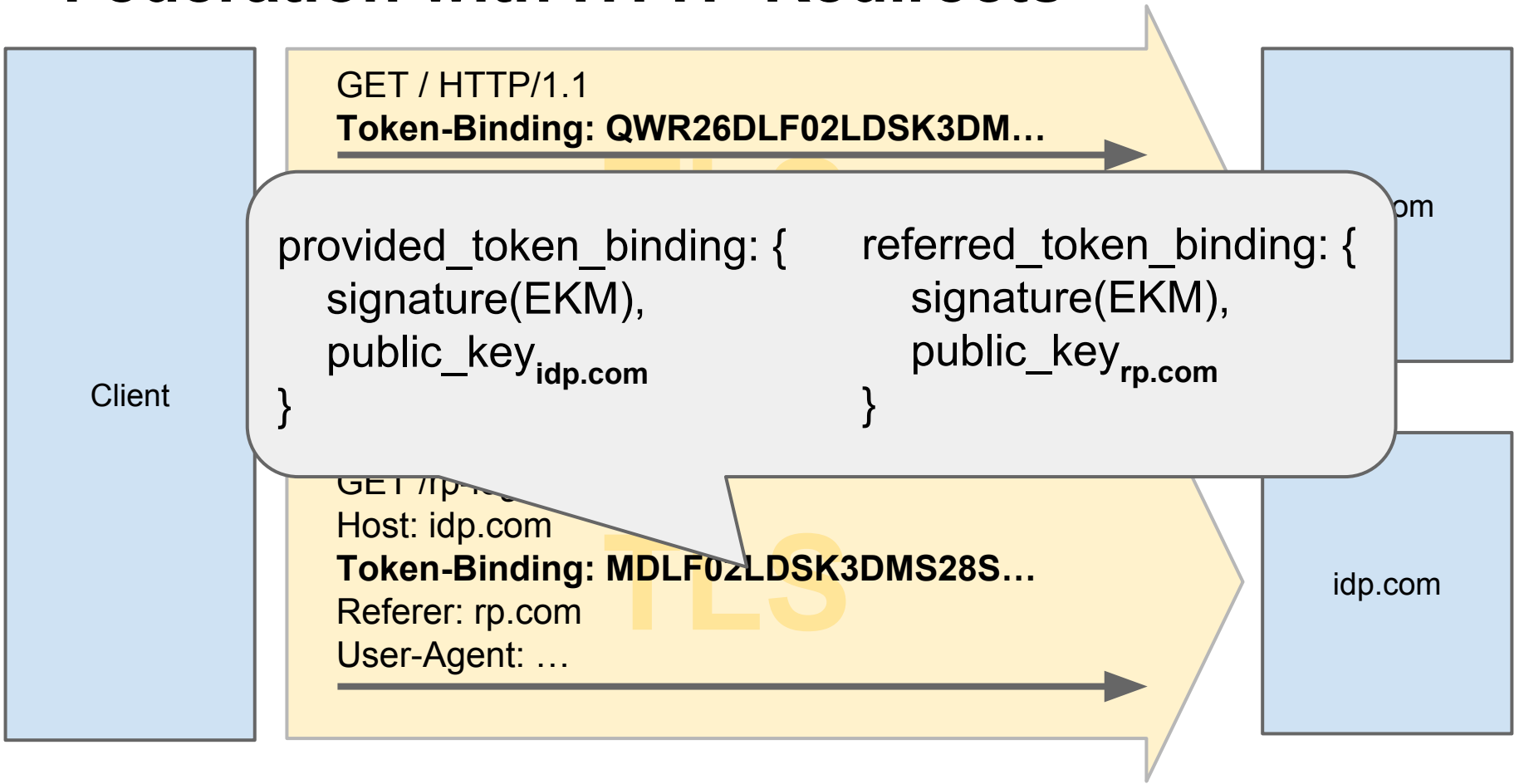
Federation with HTTP Redirects



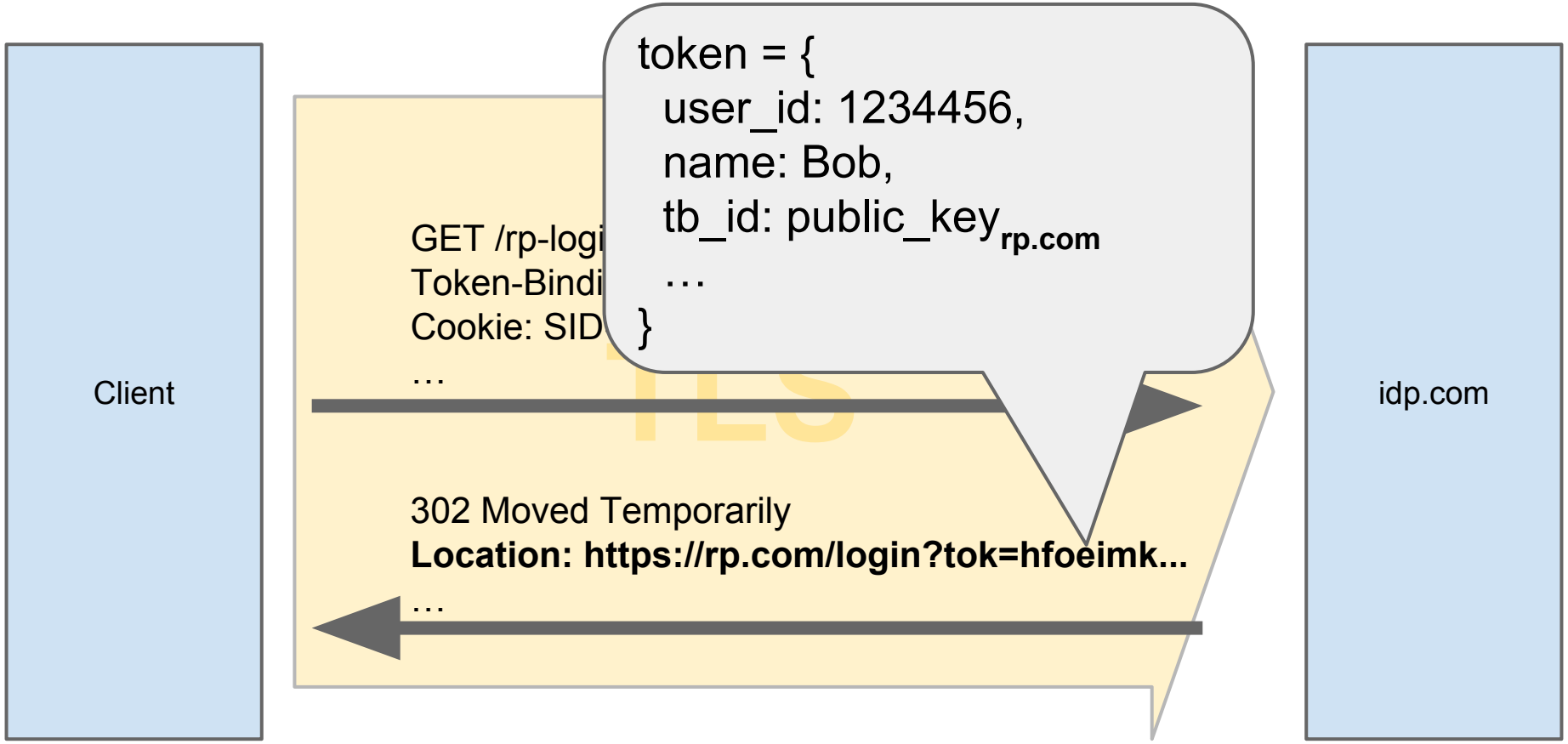
Federation with HTTP Redirects



Federation with HTTP Redirects



Federated Binding



Overview

1. Recap (for newcomers)
2. Changes to tokbind-https
3. Threat model

diff tokbind-https-01 tokbind-https-02

- Header: Sec-Token-Binding ⇒ Token-Binding
- Prove key possession by signing EKM (instead of tls_unique)
  - TLS Exported Keying Material, per RFC 5705
- Updated Security Considerations
  - Why disallow scripts from setting Token-Binding header?
  - Why prove possession of two keys for federation?

Overview

1. Recap (for newcomers)
2. Changes to tokbind-https
3. Threat model

Intent of Token Binding

server verifies public-key signature in token binding $\Longrightarrow$ **client controls corresponding private key**

Intent of Token Binding

server verifies public-key signature in token binding $\Longrightarrow$ **client controls corresponding private key**

Why?

Binding token to public key should make it possible to enforce that token can be used only by a client that can prove possession of the private key, and by nobody else.

Threats

server verifies public-key signature in token binding ⟹ **client controls corresponding private key**

1. Attacker uses victim’s private key

2. Attacker makes victim present attacker’s public key
   (== client sends attacker-generated token-binding header)

Threats

server verifies public-key signature in token binding ⟹ **client controls corresponding private key**

1. Attacker uses victim’s private key
   - countermeasure: keep private key secret
   - countermeasure: never transmit private key over network

2. Attacker makes victim present attacker’s public key
   (client sends attacker-generated token-binding header)

Threats

server verifies public-key signature in token binding ⟹ **client controls corresponding private key**

1. Attacker uses victim's private key
   - countermeasure: keep private key secret
   - countermeasure: never transmit private key over network

2. Attacker makes victim present attacker's public key (client sends attacker-generated token-binding header)
   - countermeasure: keep EKM secret

Threats

server verifies public-key signature in token binding ⟹ **client controls corresponding private key**

1. Attacker uses victim’s private key
   - countermeasure: keep private key secret
   - countermeasure: never transmit private key over network

2. Attacker makes victim present attacker’s public key (client sends attacker-generated token-binding header)
   - countermeasure: keep EKM secret
   - countermeasure: don’t let attacker set Token-Binding header

Threats

server verifies public-key signature in token binding ⟹ **client controls corresponding private key**

1. Attacker uses victim’s private key
   - countermeasure: keep private key secret
   - countermeasure: never transmit private key over network

2. Attacker makes victim present attacker’s public key (client sends attacker-generated token-binding header)
   - countermeasure: keep EKM secret
   - countermeasure: don’t let attacker set Token-Binding header
   - countermeasure: make client prove possession of every key in header

Questions