

Breaking FIDO

Are Exploits in There?

yubico

FIDO U2F (Universal 2nd Factor)

- **Analyzing FIDO U2F**
- **Attack and Countermeasures**
- **Implementation Considerations**
- **Resources**

User Experience

1. Enter username/pwd

Sign in


Username or email address

Password (forgot password)

Sign in


2. Insert U2F device

Two-factor authentication

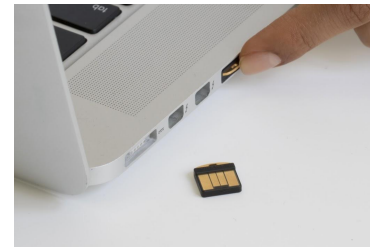


Insert your security key

Press the button on your security key device to finish signing in. If it does not have a button, just re-insert it.

 Press the button on your security key...

3. Touch U2F device



Core Features & Supporting Sites

- **Scalable**
 - Works across any number of services
 - Remote provisioning
- **Secure**
 - Protects against phishing & MitM
 - Verifies user presence
- **Simple UX**
 - Single gesture operation
- **Open Standard**
 - Native platform/browser support

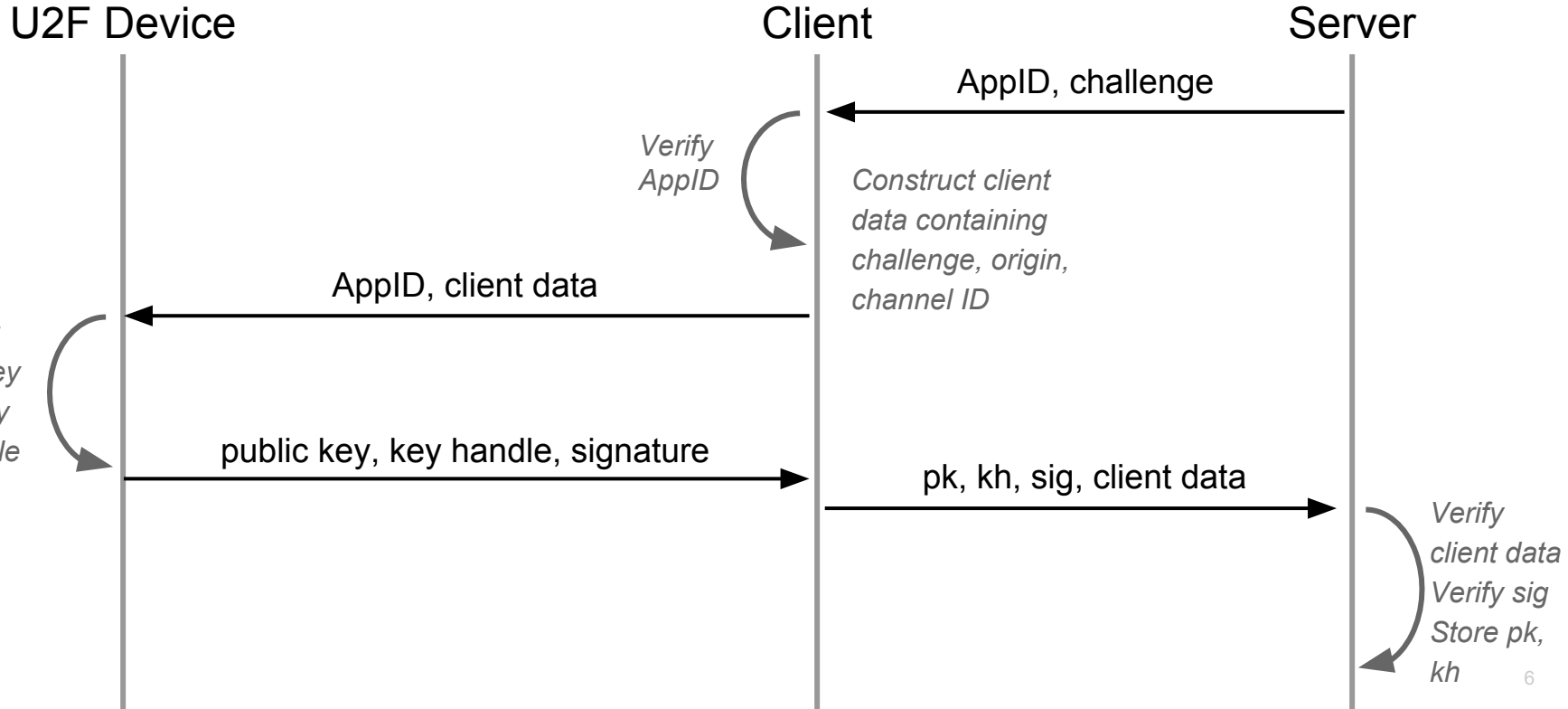


Challenge Parameter (c)

Client Data = Challenge, Origin, Channel ID

c = SHA-256 hash (Client Data)

Registration

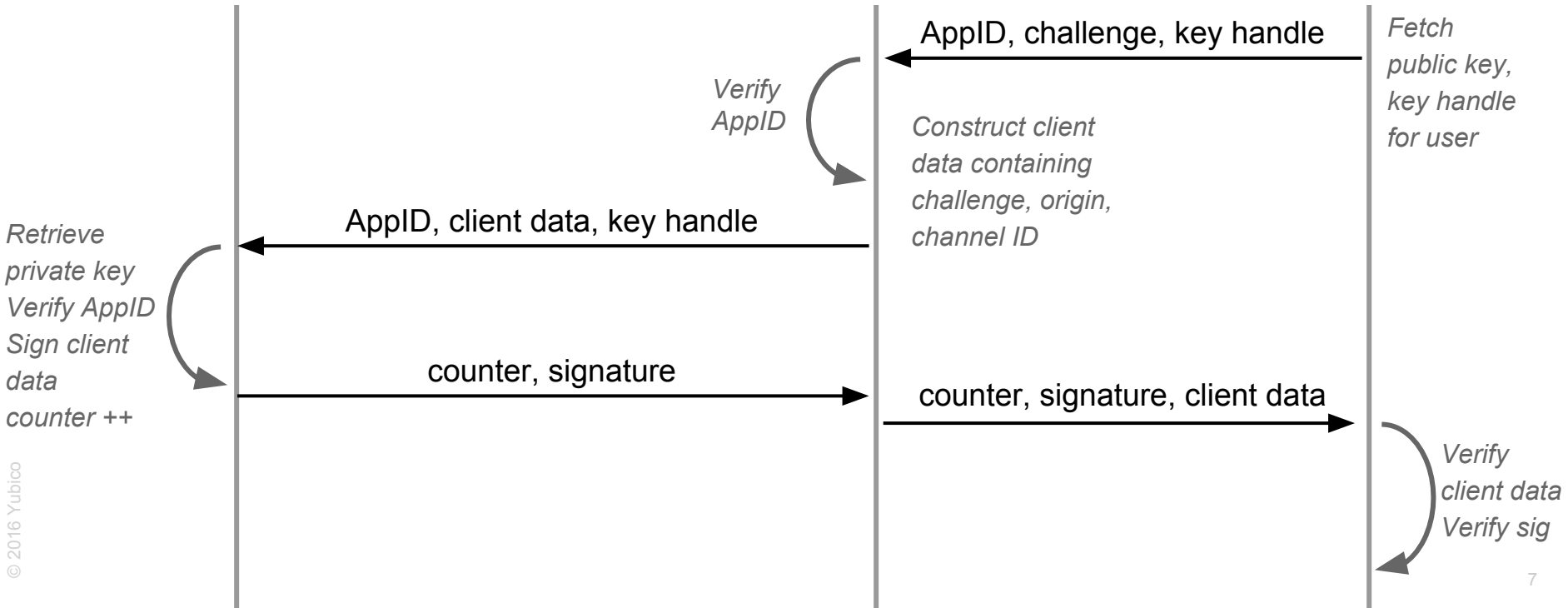


Authentication

U2F Device

Client

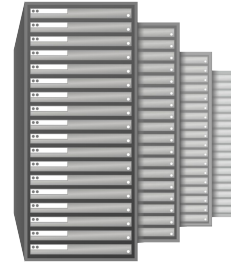
Server





Attacker

Replay sessions



acme.com

Reuse credentials



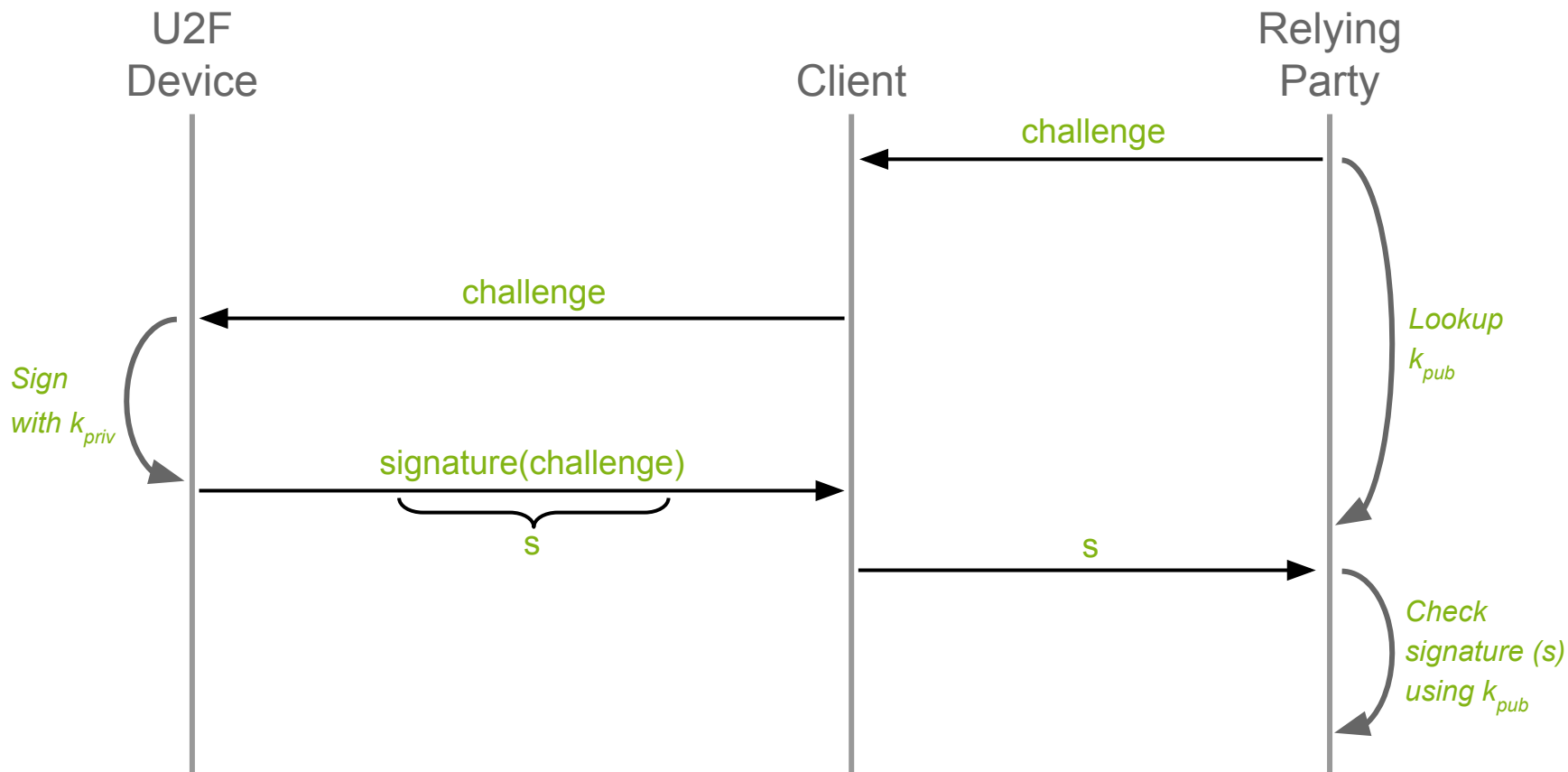
not_protected.com

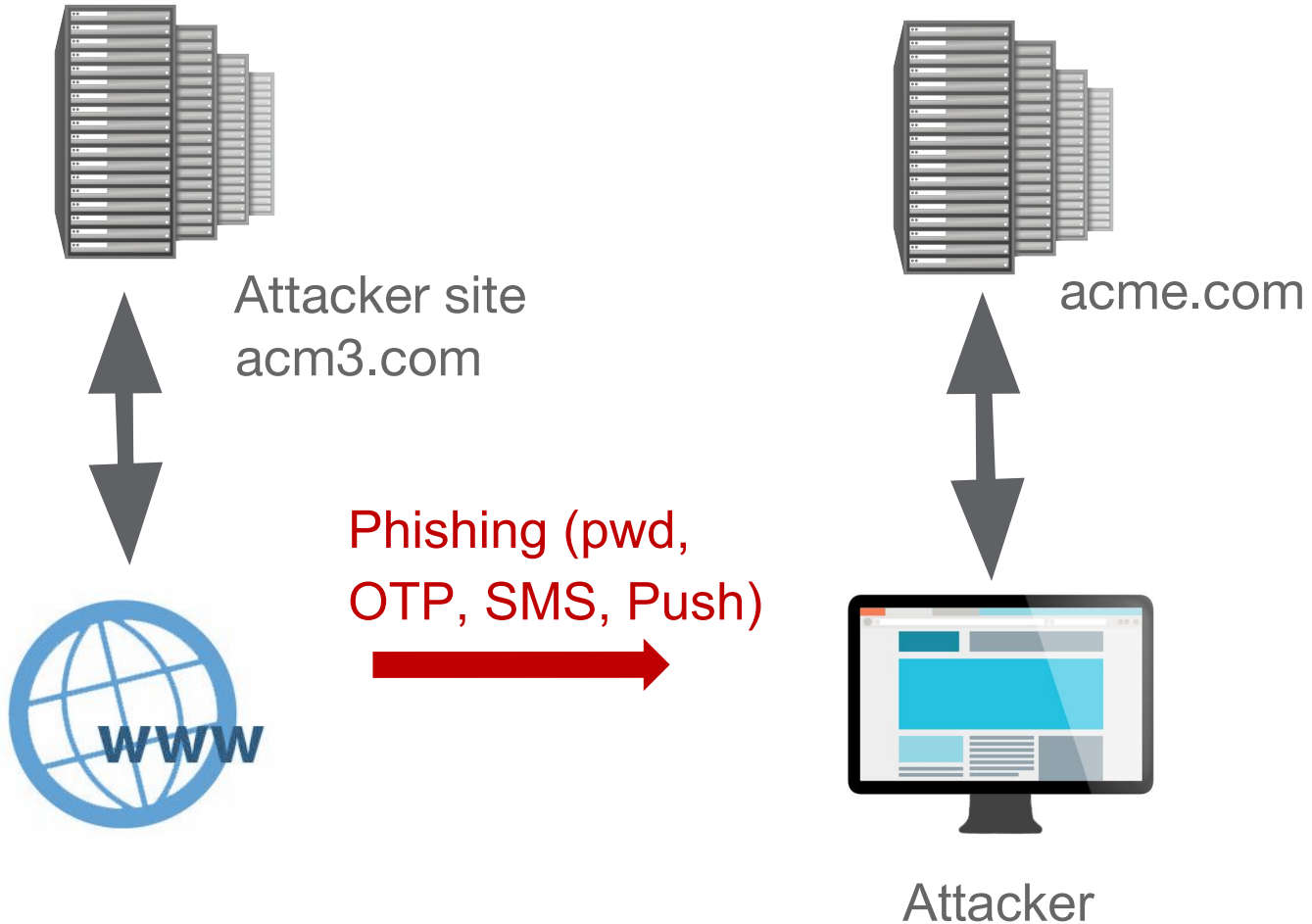
Password DB stolen

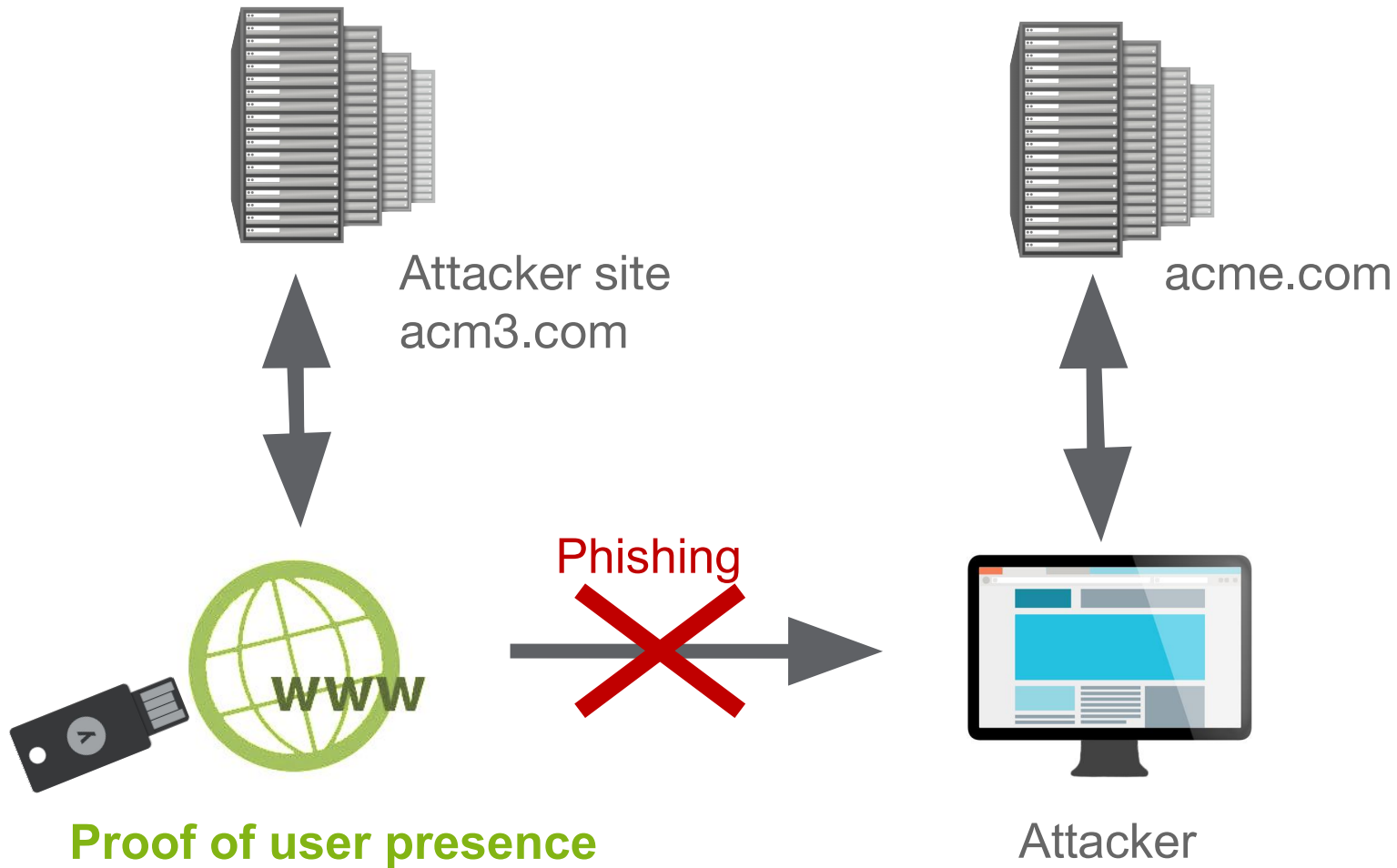


Attacker

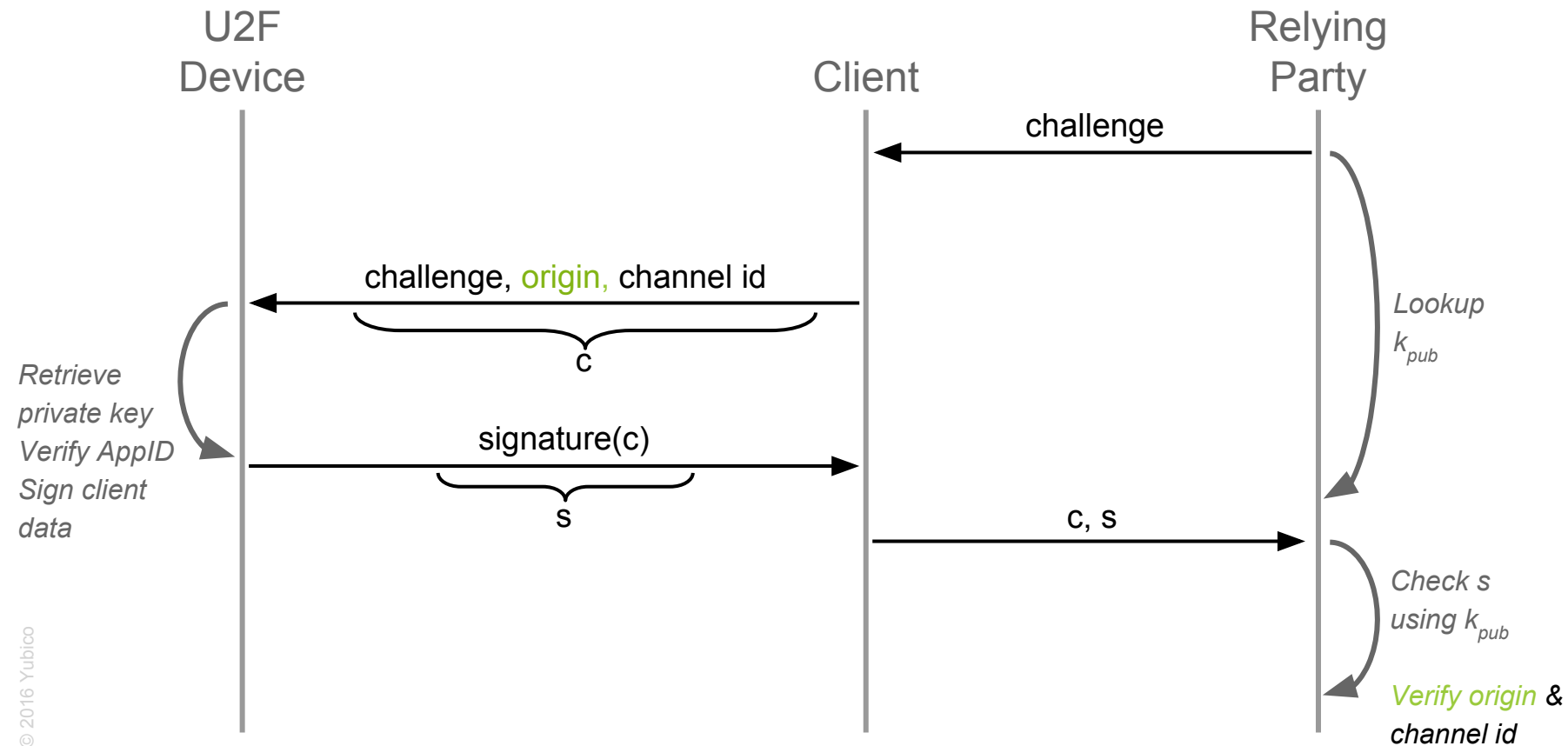
Password Reuse & Replay Attack

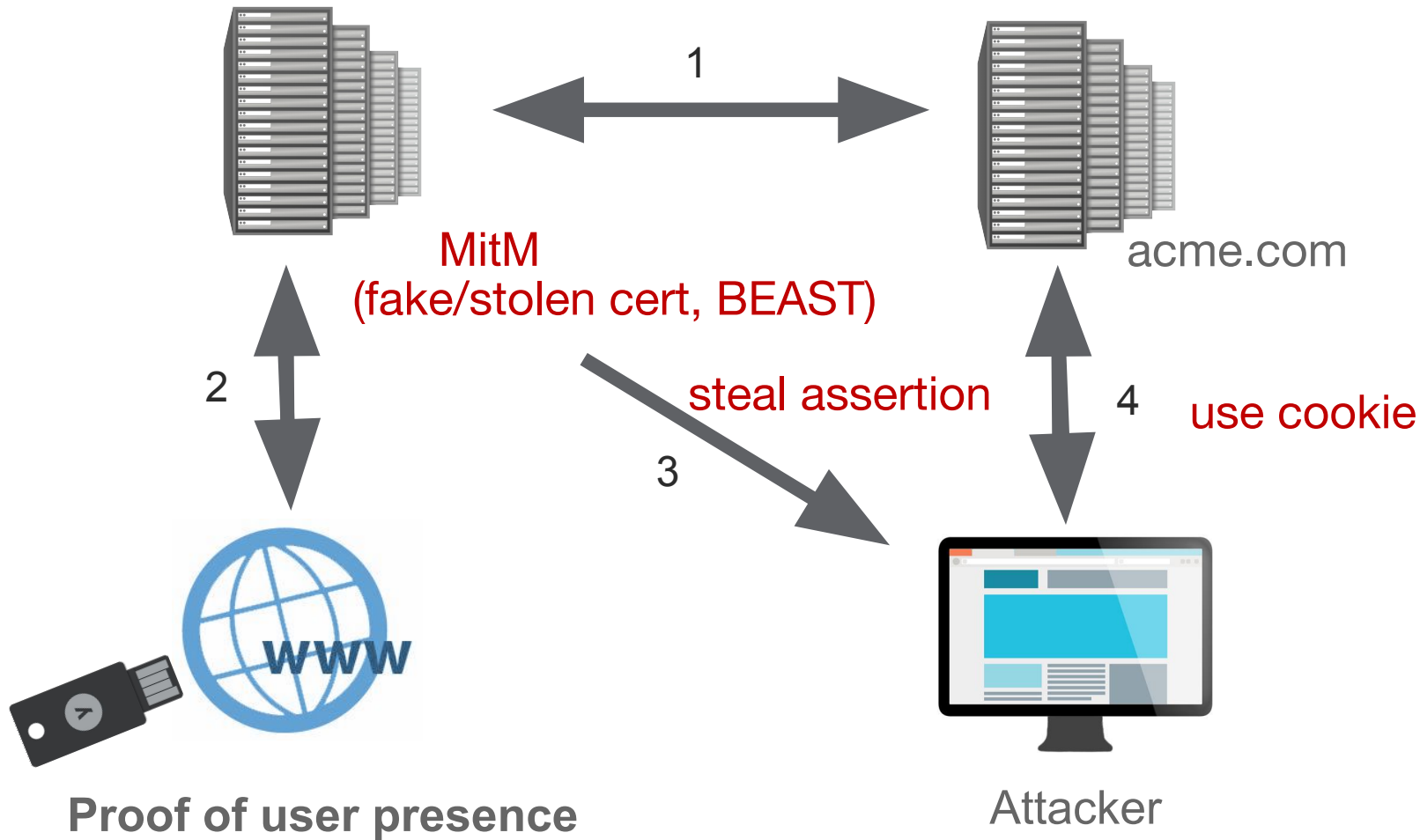




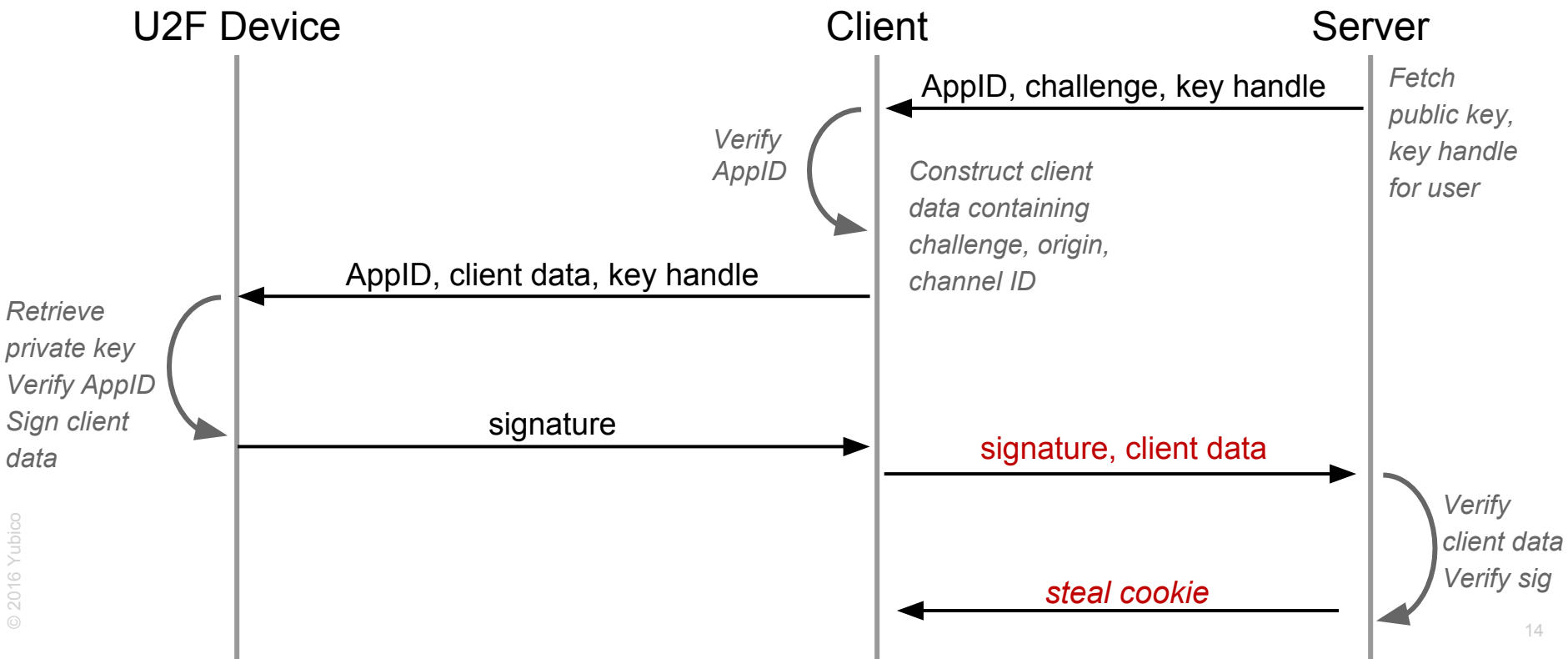


Phishing Protection

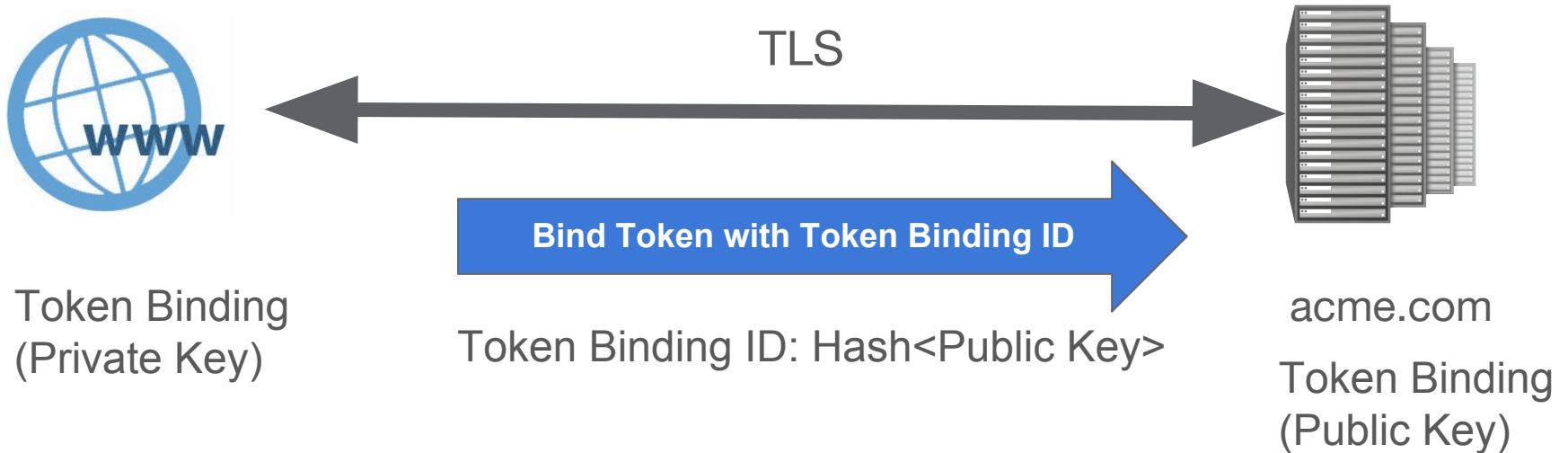


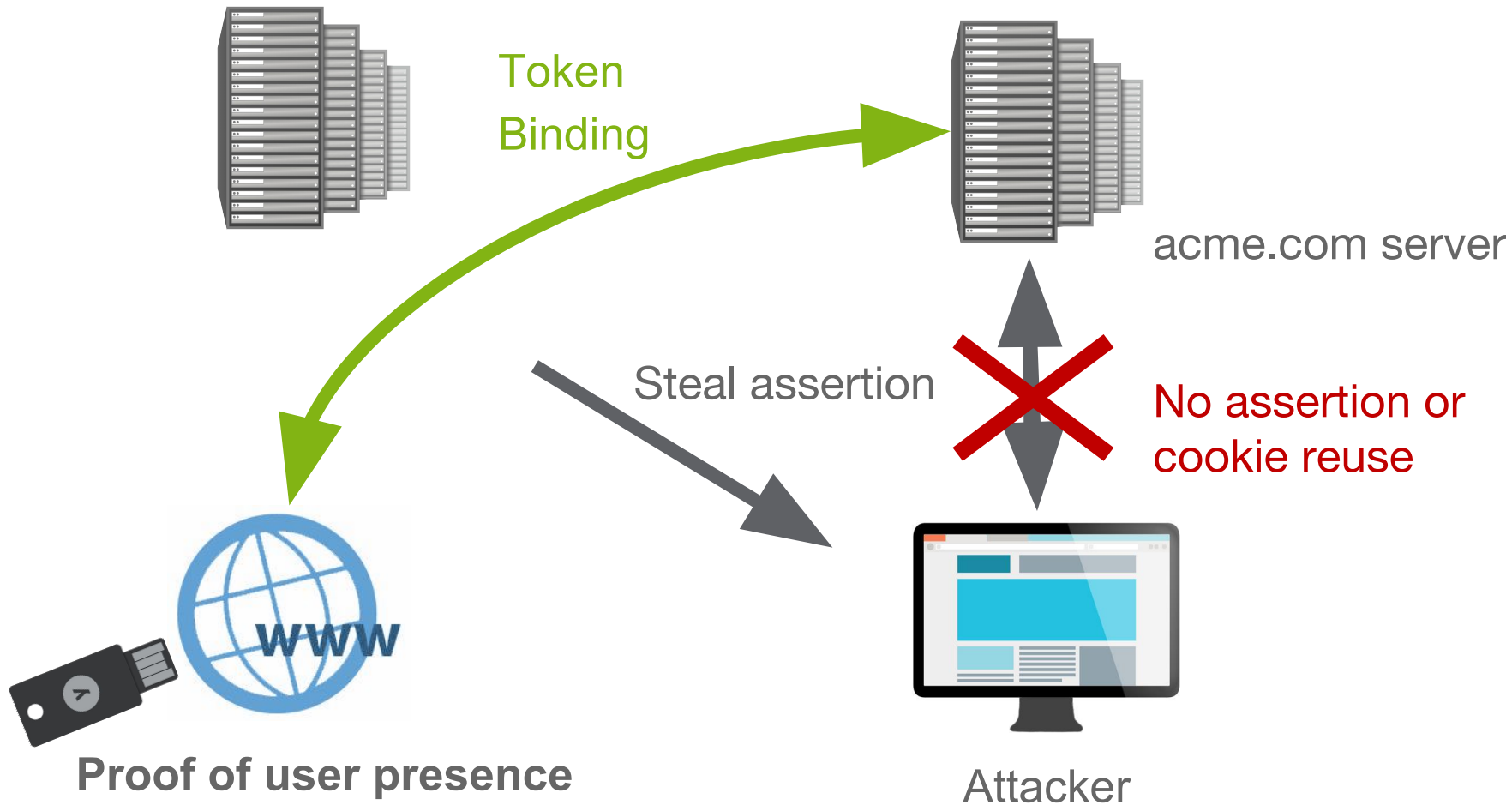


Hijack User Login Session

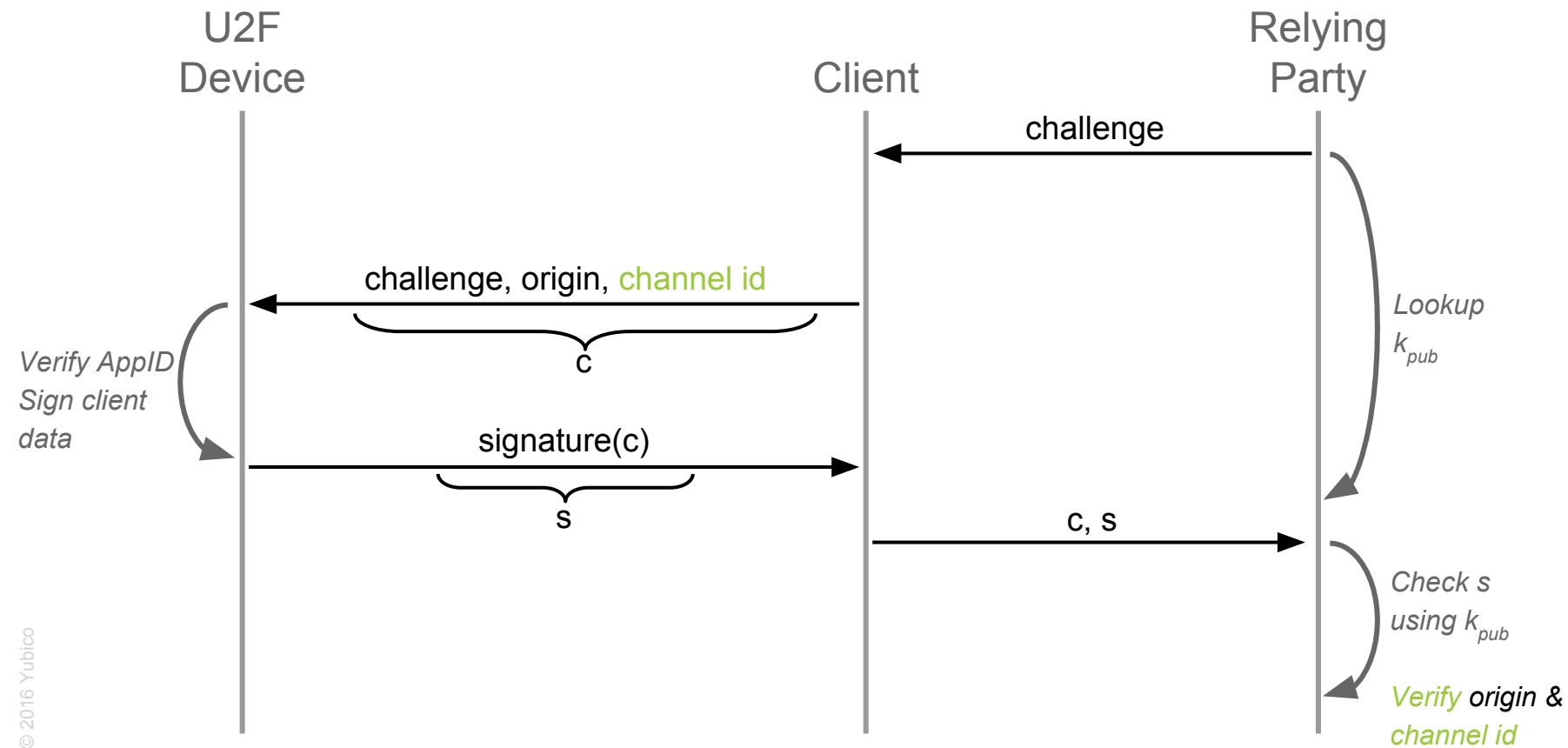


Channel ID (Token Binding)





MitM Protection





“I promise a user was here”

“The server challenge was: KSDJsdASc8-A17pW”

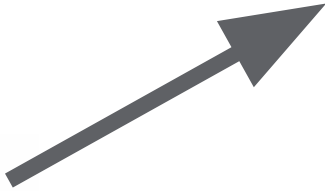
“The origin was: accounts.acme.com”

“The TLS connection state was: 3454567”

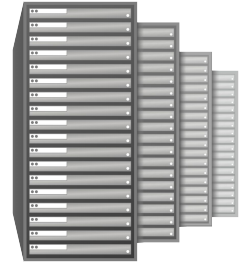
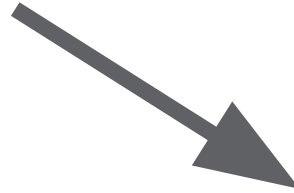
Sincerely, your Browser

- Origin mismatch for key handle => **MitM attack!**
- Incorrect origin name => **MitM attack!**
- Channel ID mismatch => **MitM attack!**

Malware



Attacker



acme.com server

Compromised Client

U2F Device

Client

Server

*Fetch
public key,
key handle
for user*

AppID, challenge, key handle

*Verify
AppID*

*Construct client
data containing
challenge, origin,
channel ID*

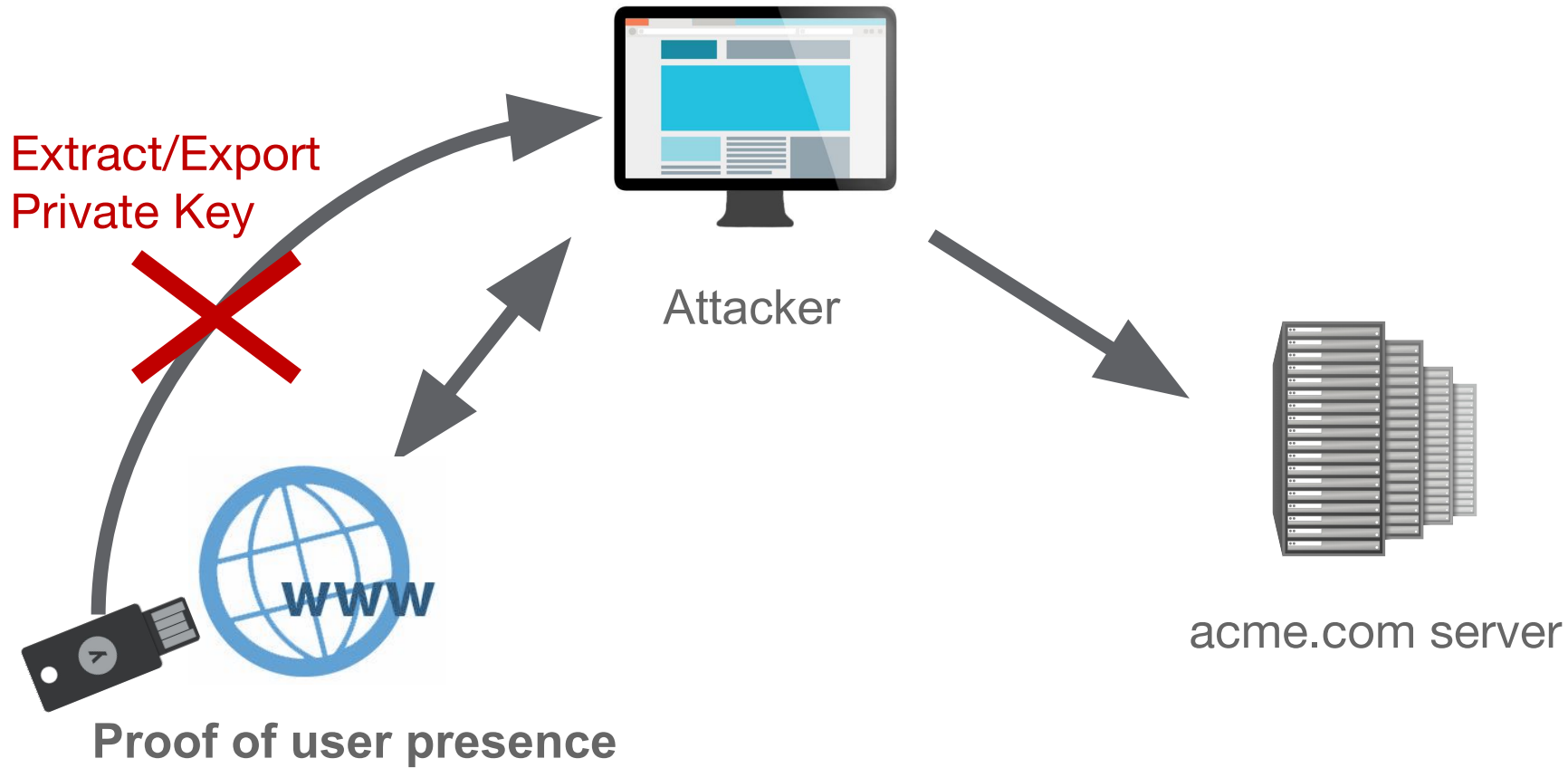
AppID, client data, key handle

*Retrieve
private key
Verify AppID
Sign client
data*

signature

signature, client data

*Verify
client data
Verify sig*



What are the Alternatives?

- Smart card => **Scaling**
- OTP/SMS => **Phishing**
- Push notification => **Phishing**

Google's research paper on FIDO U2F deployment: yubico.com/google-study

Implementation Considerations

- **Registration flow**
 - TOFU (Trust On First Use)
 - Attacker enables FIDO on account first
 - User sets up weak backup options
- **Recovery flow**
 - Lowering the level of assurance
 - Lost/stolen authenticator/device
 - Onboarding credentials to new devices

How to Get Started?

Learn

Read the specifications dev.yubi.co/U2F & github.com/dainnilsson/u2f-tutorial

Go through a MiniTwit U2F tutorial [MiniTwit training video](#)

Implement

Google reference code github.com/google/u2f-ref-code

Build your own U2F server dev.yubi.co/U2F/libraries

Use Yubico standalone U2F server dev.yubi.co/u2fval

Test

Yubico U2F demo server demo.yubico.com/u2f

Google U2F demo server u2fdemo.appspot.com

Thank you!

Questions