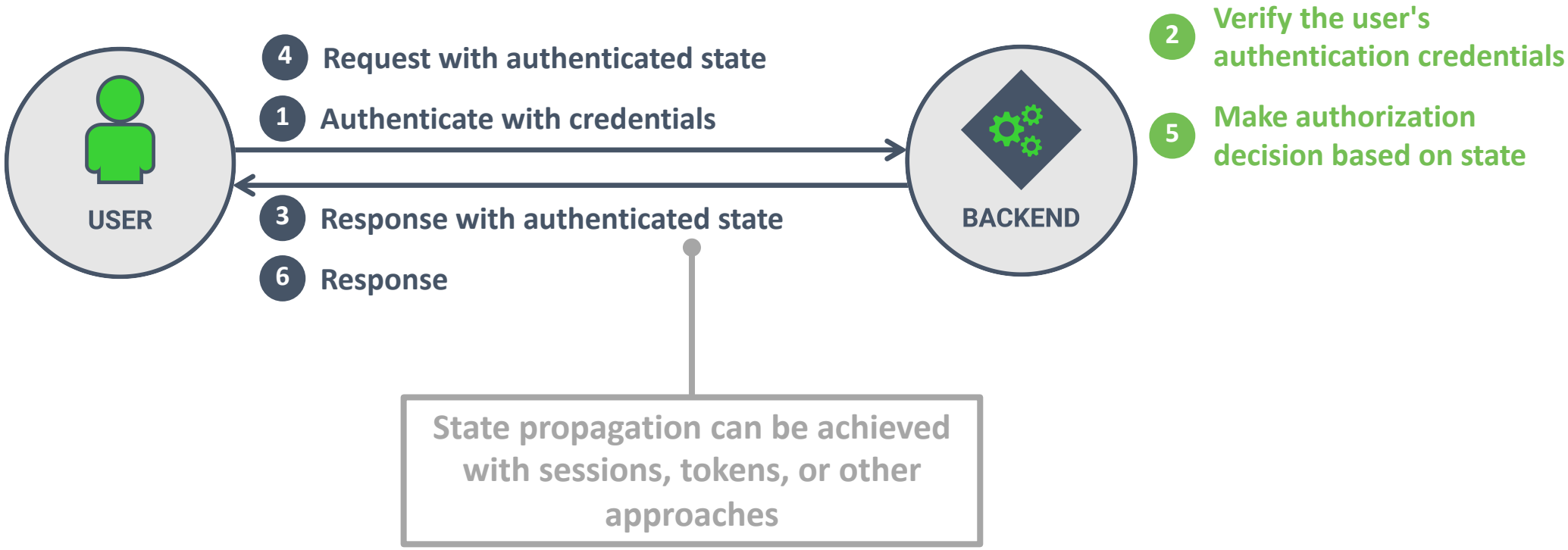
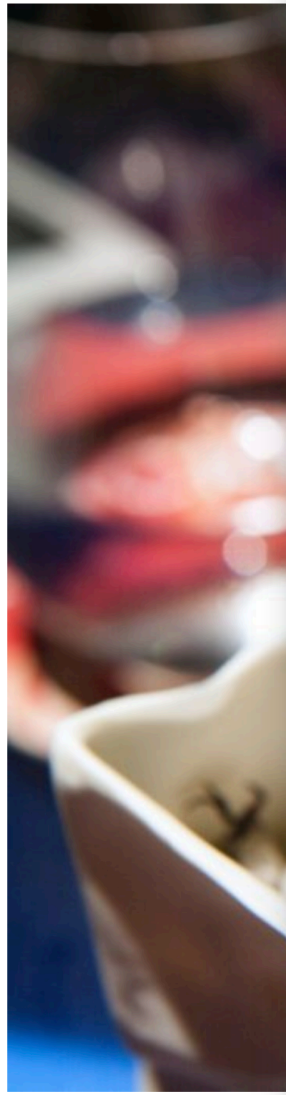
Serving the right recipe for user authentication

Dr. Philippe De Ryck

https://Pragmatic Web Security.com


USER
BACKEND
4 Request with authenticated state
1 Authenticate with credentials
3 Response with authenticated state
6 Response
2 Verify the user's authentication credentials
5 Make authorization decision based on state
State propagation can be achieved with sessions, tokens, or other approaches


USER
BACKEND
1 Authenticate with credentials
2 Verify the user's authentication credentials
3 Response with authenticated state
4 Request with authenticated state
5 Make authorization decision based on state
6 Response



Tickets (Bar
Hype

r/shittyfoodporn · Crossposted by u/supergamer1313 1 hour ago

$1 NYC Pizza Slice

r/FoodPorn · Posted by u/BravoVogue 7 hours ago 3

$1 NYC Pizza Slice

6.7k points · 325 comments

3 Comments Give Award Share Save Hide Report 83% Upvoted

Wong
ering Eater

th The

I am *Dr. Philippe De Ryck*

Pragmatic Web Security
Security for developers

Founder of Pragmatic Web Security

Google Developers Experts

Google Developer Expert

Auth0 Ambassador / Expert

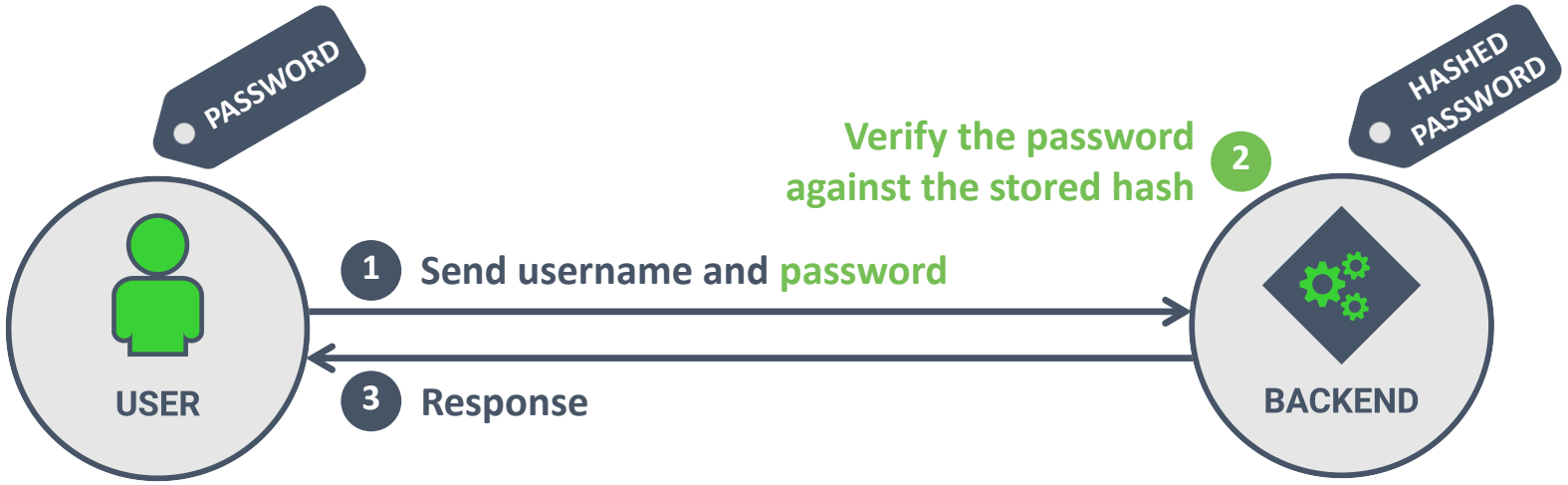
Secure Application Development

SecAppDev organizer

I help developers with security

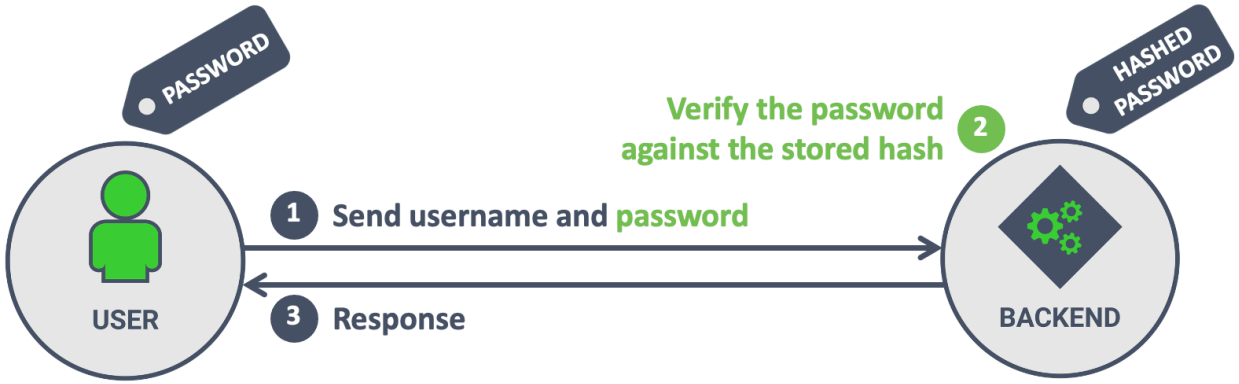
- **Academic-level security training**
- **Hands-on in-depth online courses**
- **Security advisory services**

https://pragmaticwebsecurity.com


PASSWORD
HASHED PASSWORD
Verify the password against the stored hash
2
1
Send username and password
3
Response
USER
BACKEND

PASSWORD-BASED AUTHENTICATION

- *The password is a shared secret between the user and the backend*
- *Easy to understand and use*
- *Difficult to handle passwords securely*



BENEFITS

Lightweight mechanism with minimal overhead

Conceptually easy to understand

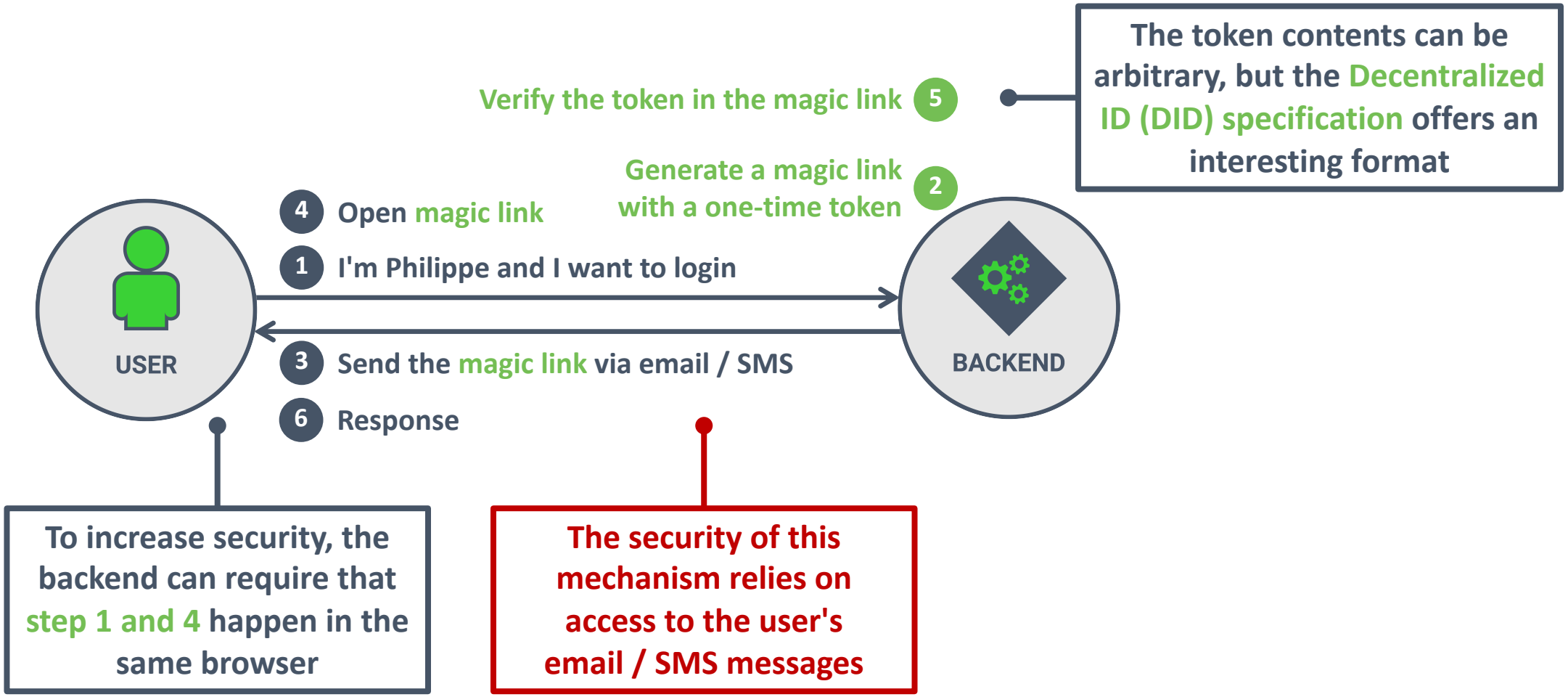
Easily portable across devices

DRAWBACKS

Passwords are re-used across applications

Passwords are vulnerable to phishing

Passwords can be stolen or brute forced


The token contents can be arbitrary, but the Decentralized ID (DID) specification offers an interesting format
Verify the token in the magic link 5
Generate a magic link with a one-time token 2
4 Open magic link
1 I'm Philippe and I want to login
USER
BACKEND
3 Send the magic link via email / SMS
6 Response
To increase security, the backend can require that step 1 and 4 happen in the same browser
The security of this mechanism relies on access to the user's email / SMS messages

AUTHENTICATING WITH MAGIC LINKS

- *No passwords or shared secrets*
- *Easy to understand and use*
- *Moves the responsibility of authentication to someone else*

BENEFITS

Conceptually easy to understand

Portable and less susceptible to phishing

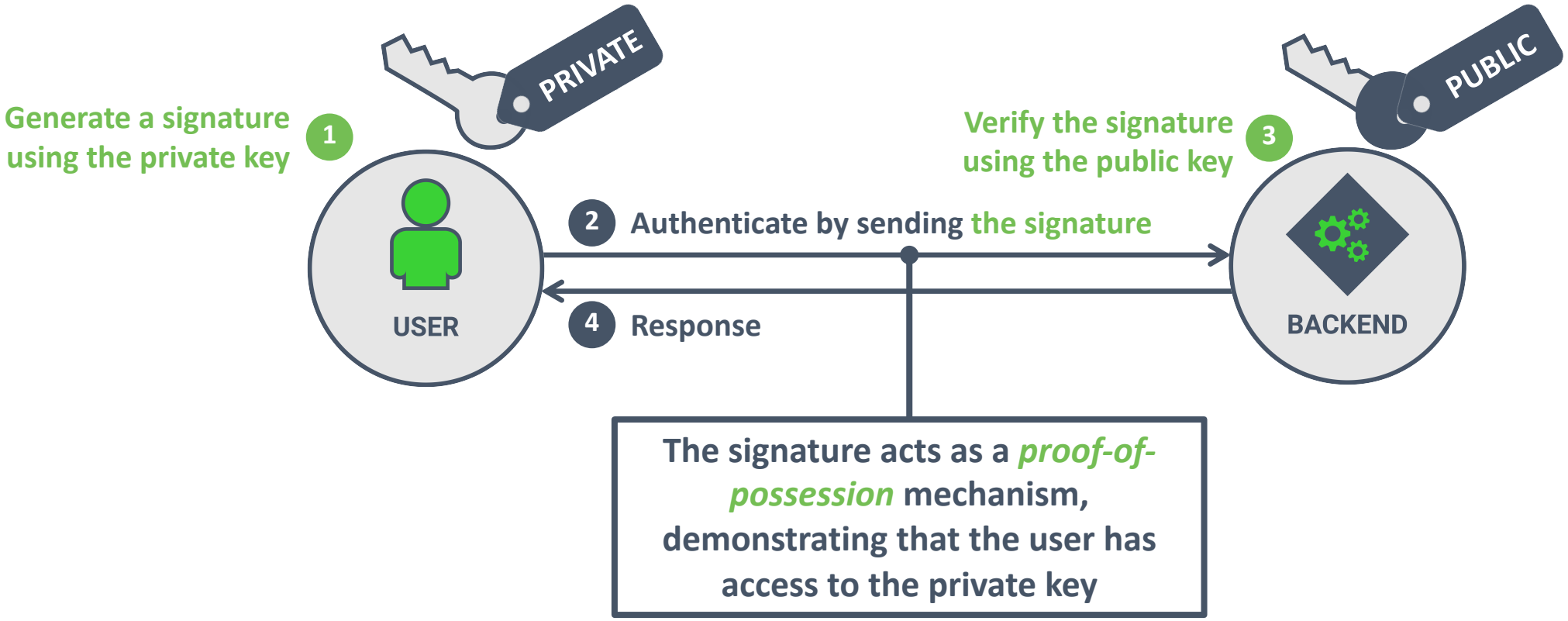
Not worse than traditional *"reset password"* features

DRAWBACKS

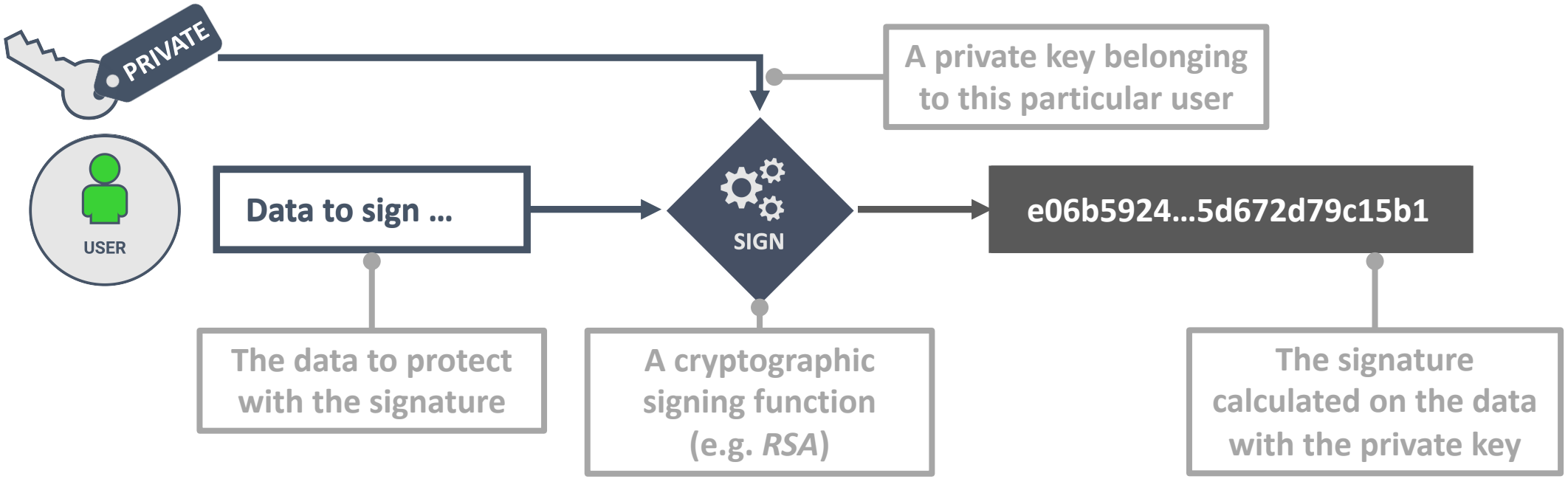
Implicit authentication through email / SMS access

Email and SMS have their own security issues

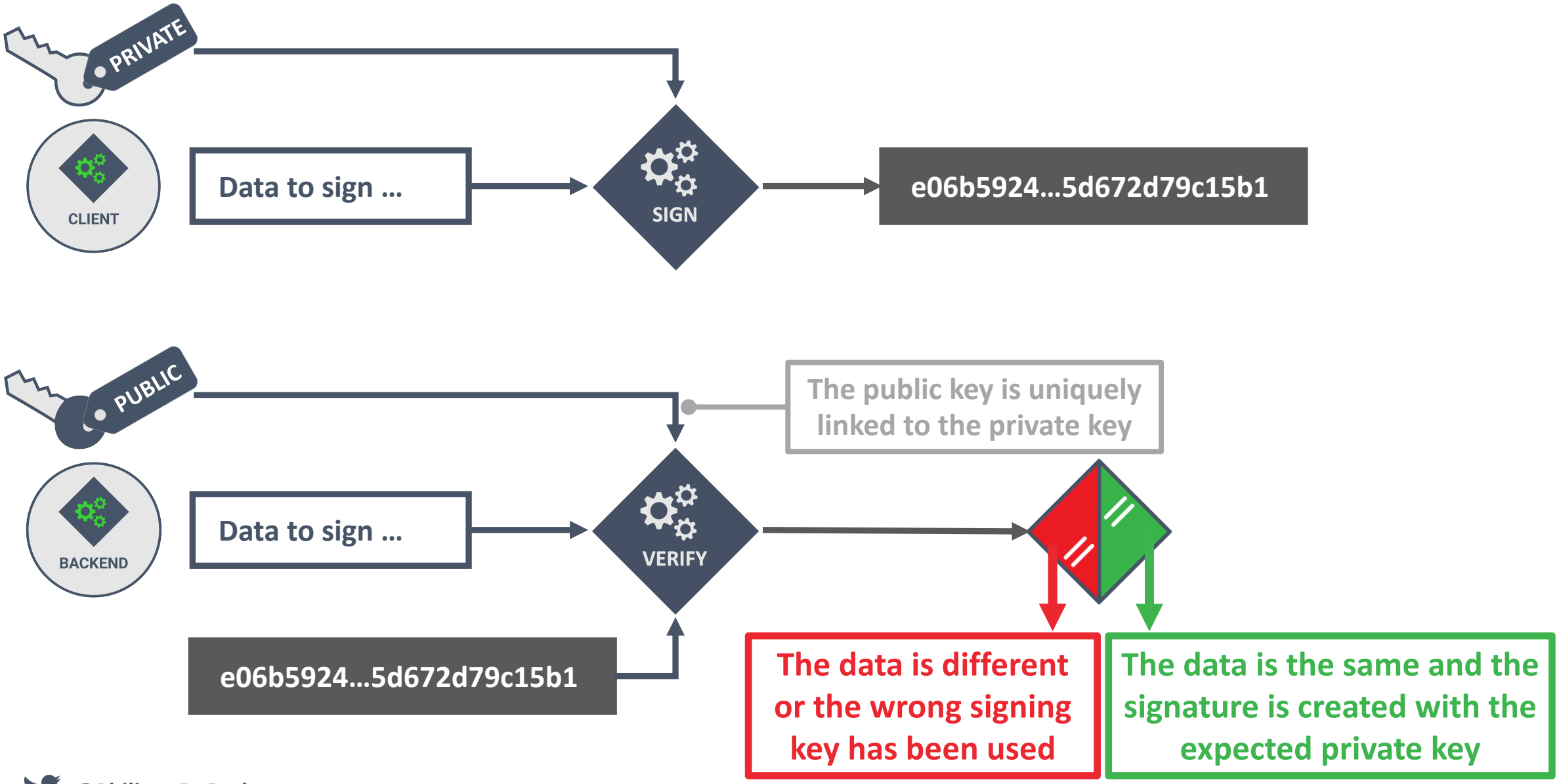
Reliance on third-party for a crucial security feature

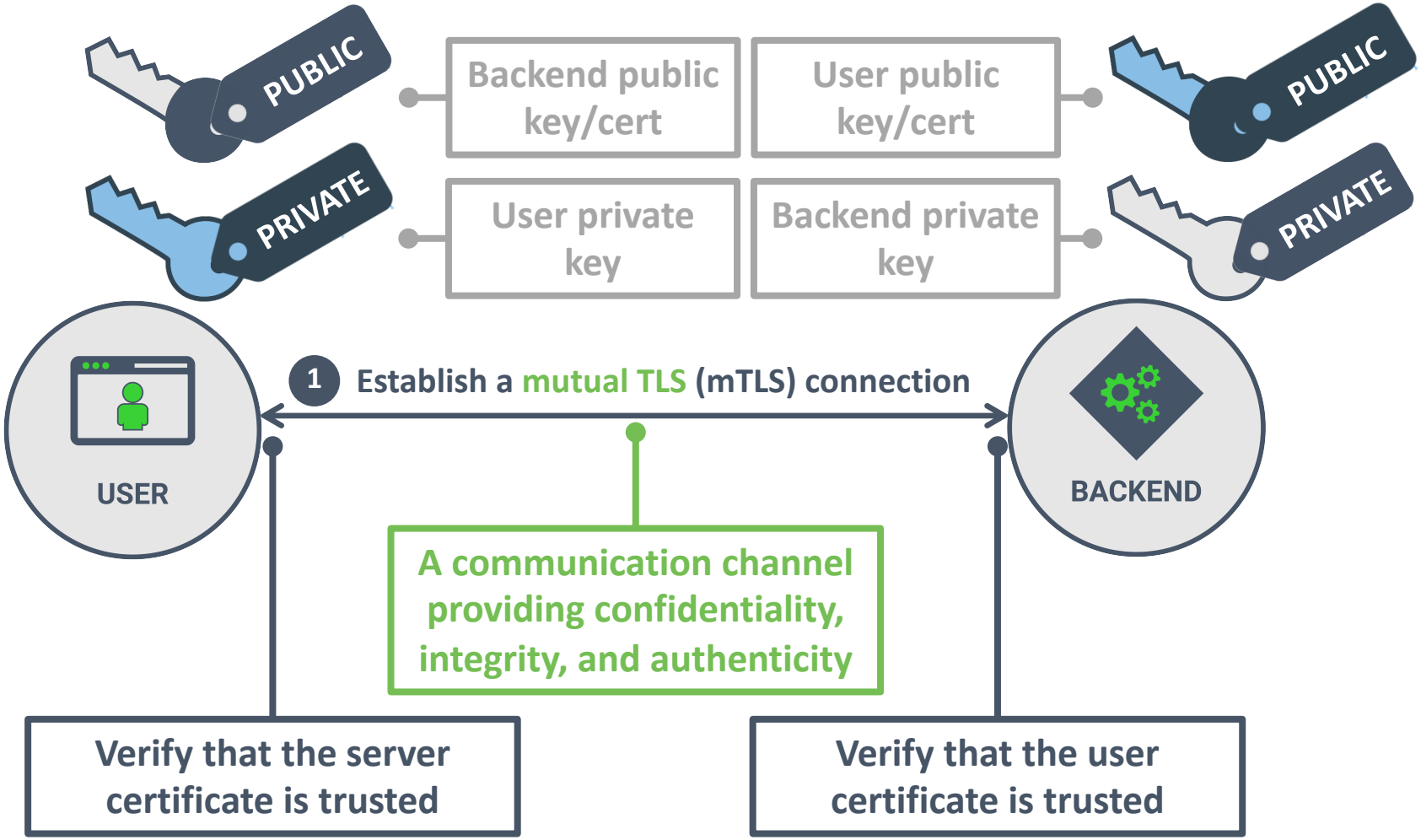

PRIVATE
PUBLIC
Generate a signature using the private key
1
Verify the signature using the public key
3
2 Authenticate by sending the signature
4 Response
USER
BACKEND
The signature acts as a *proof-of-possession* mechanism, demonstrating that the user has access to the private key

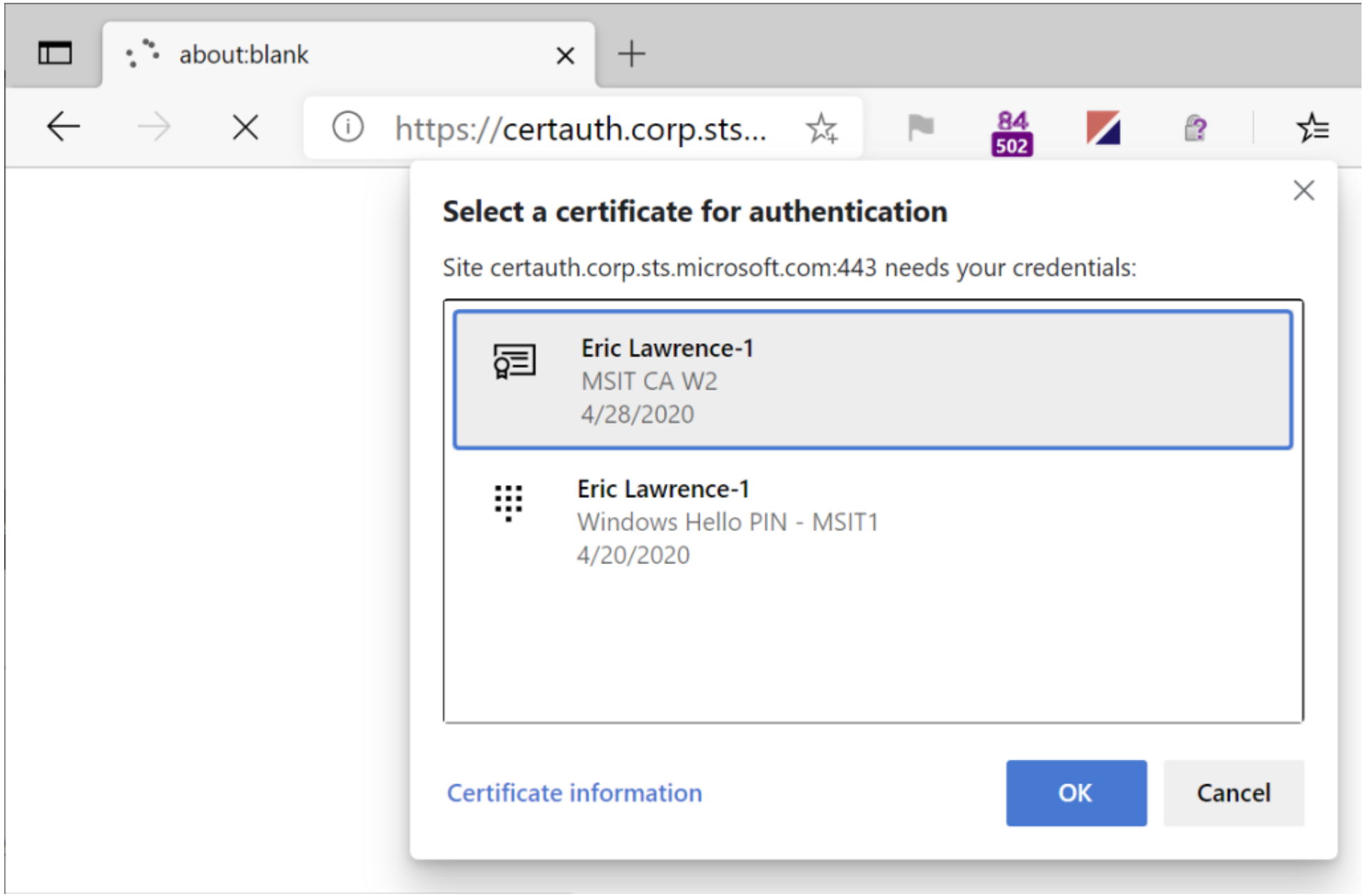
INTERMEZZO: DIGITAL SIGNATURES



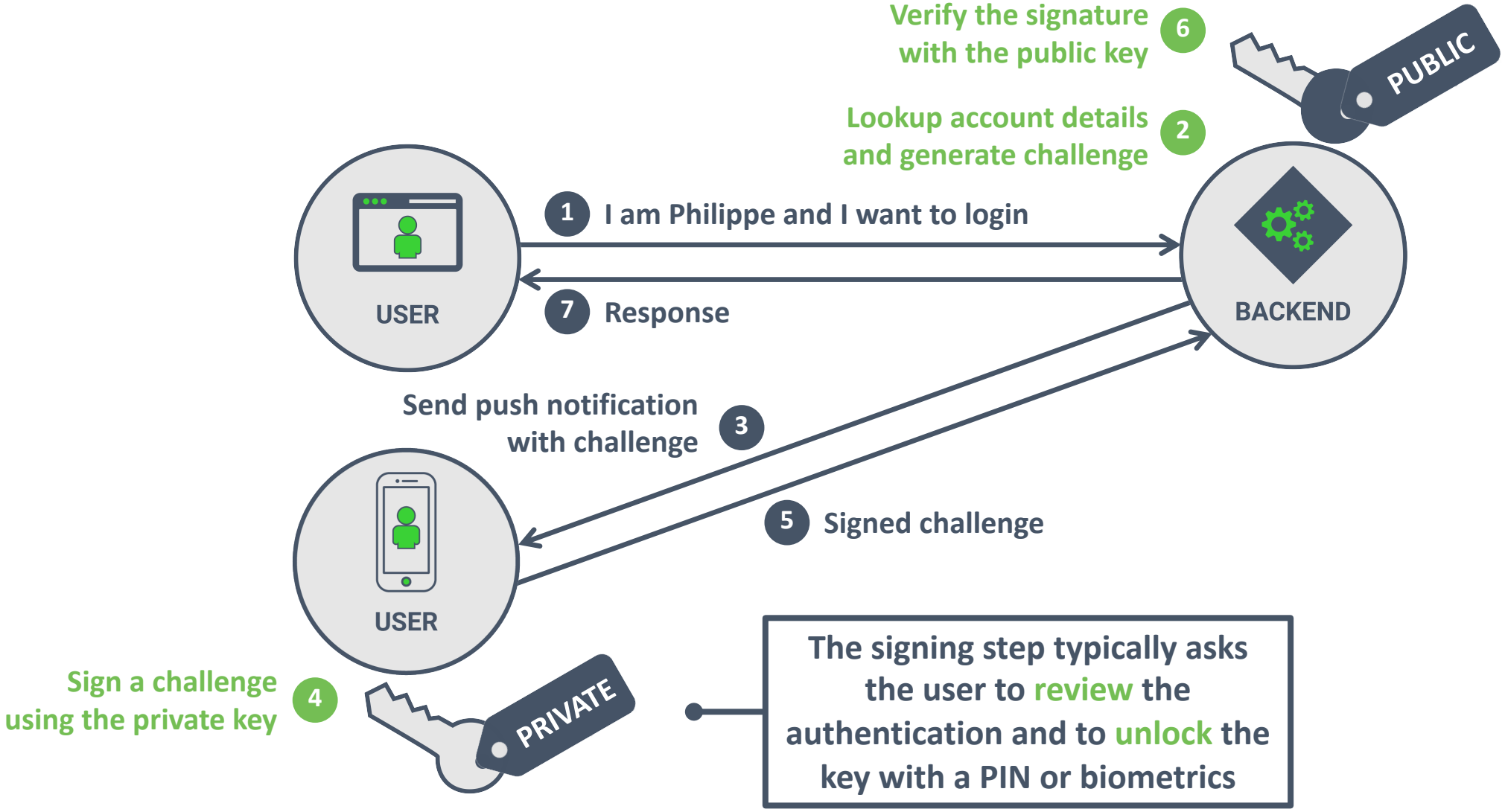
INTERMEZZO: DIGITAL SIGNATURES




PUBLIC
PRIVATE
Backend public key/cert
User public key/cert
User private key
Backend private key
PUBLIC
PRIVATE
USER
1 Establish a mutual TLS (mTLS) connection
BACKEND
A communication channel providing confidentiality, integrity, and authenticity
Verify that the server certificate is trusted
Verify that the user certificate is trusted

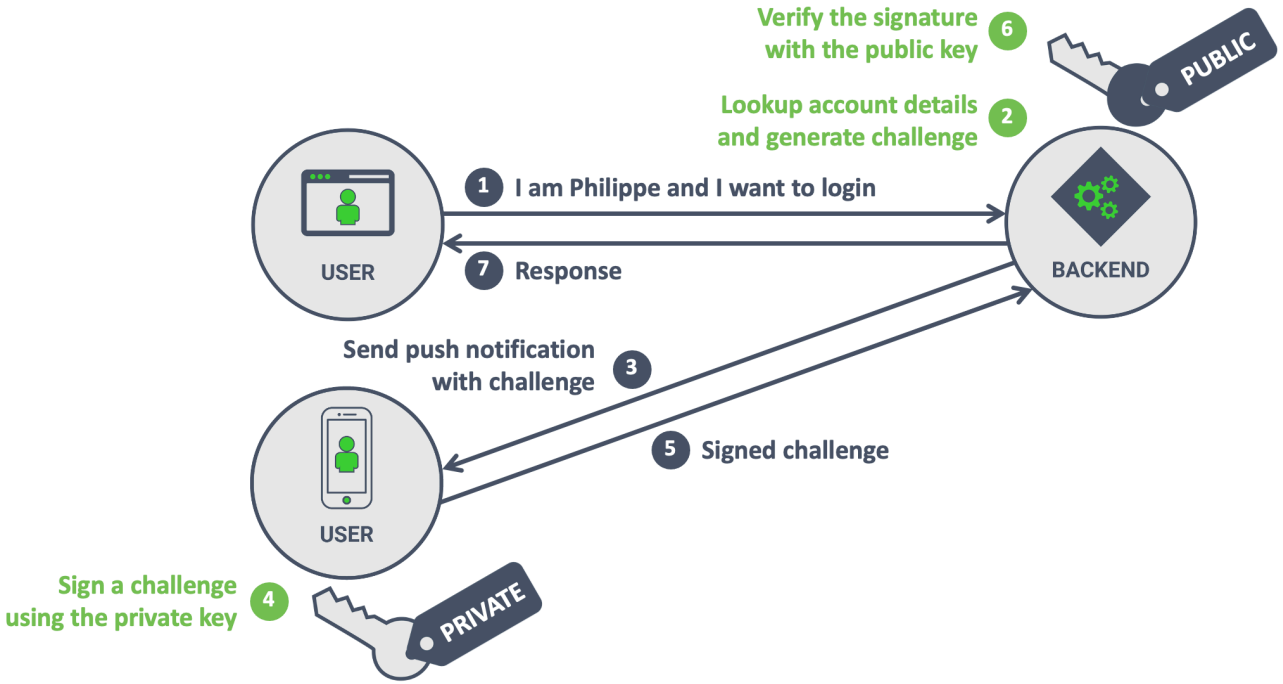


https://textslashplain.com/2020/05/04/client-certificate-authentication/


Verify the signature with the public key
6
PUBLIC
Lookup account details and generate challenge
2
1
I am Philippe and I want to login
USER
BACKEND
7
Response
Send push notification with challenge
3
5
Signed challenge
USER
Sign a challenge using the private key
4
PRIVATE
The signing step typically asks the user to review the authentication and to unlock the key with a PIN or biometrics

KEY-BASED AUTHENTICATION

- *No shared secrets, but an asymmetric key pair with a public and private part*
- *Authentication comes down to proving possession of the private key*



BENEFITS

Only the public key needs to be shared (no secrets)

Many OSes offer secure storage for crypto keys

Works really well for native mobile applications

DRAWBACKS

Requires a client-side mechanism to handle keys

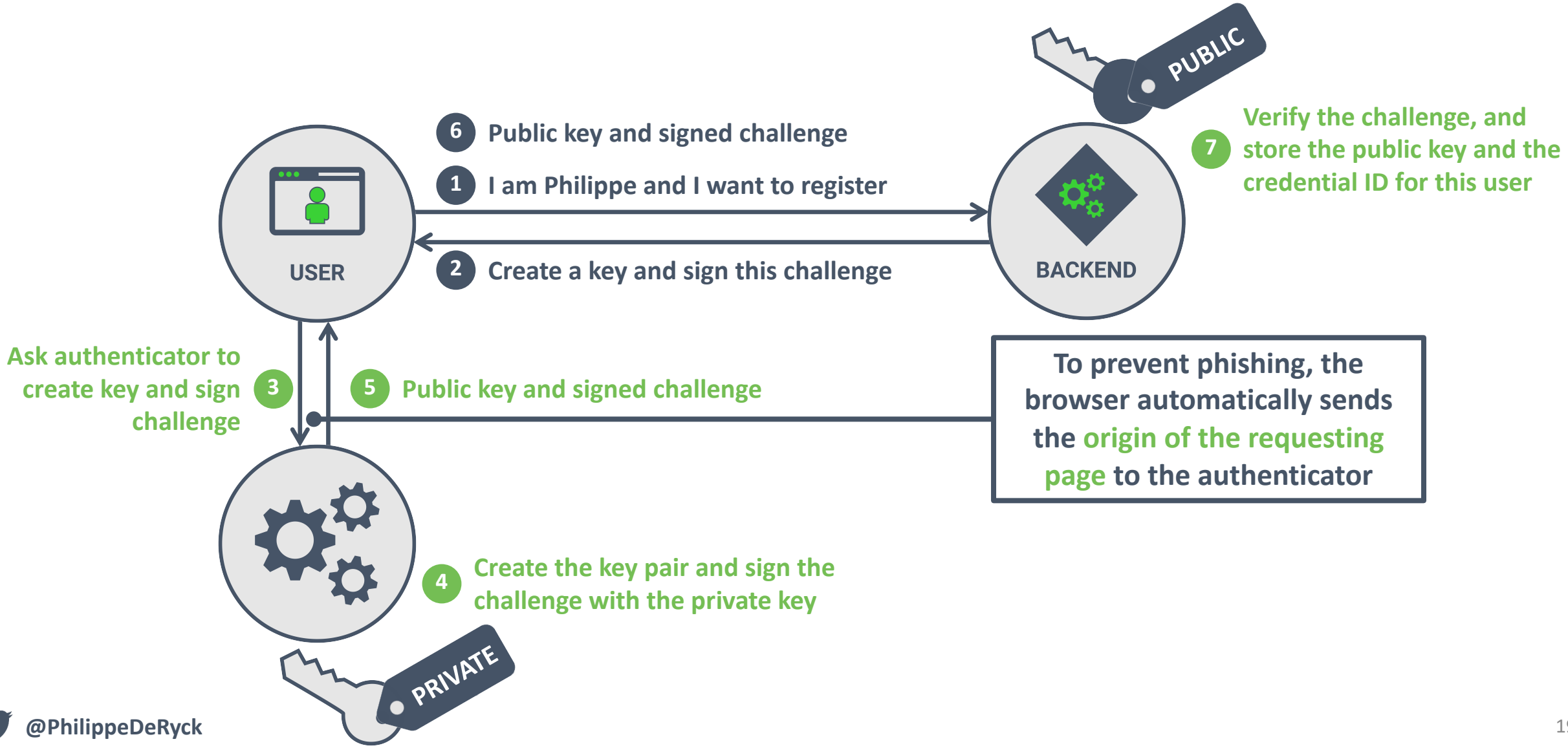
Implementing a custom key-based scheme is challenging

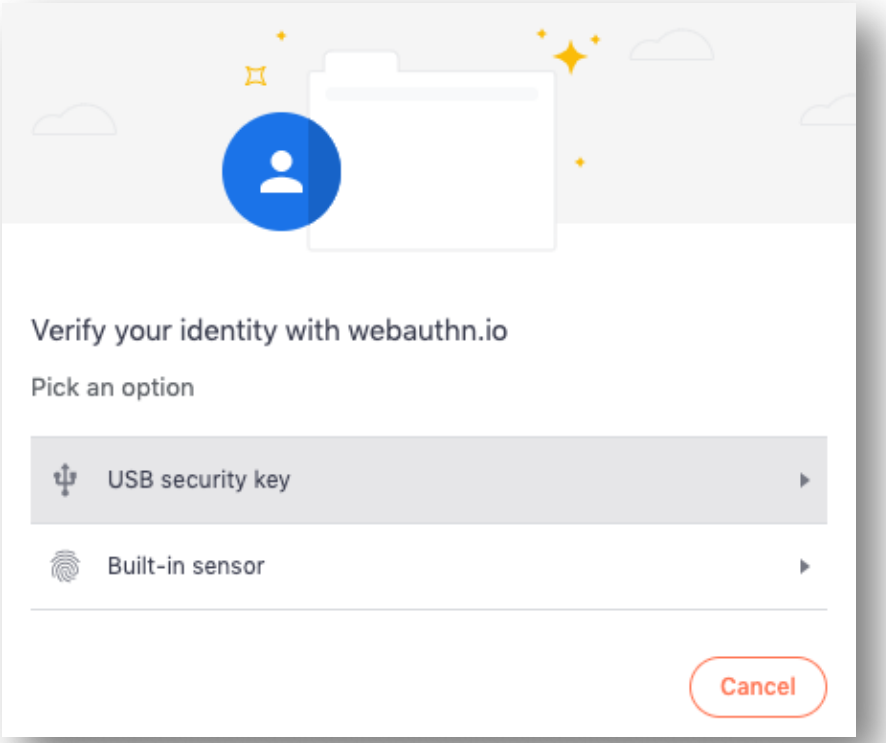
Not a scalable approach for web applications

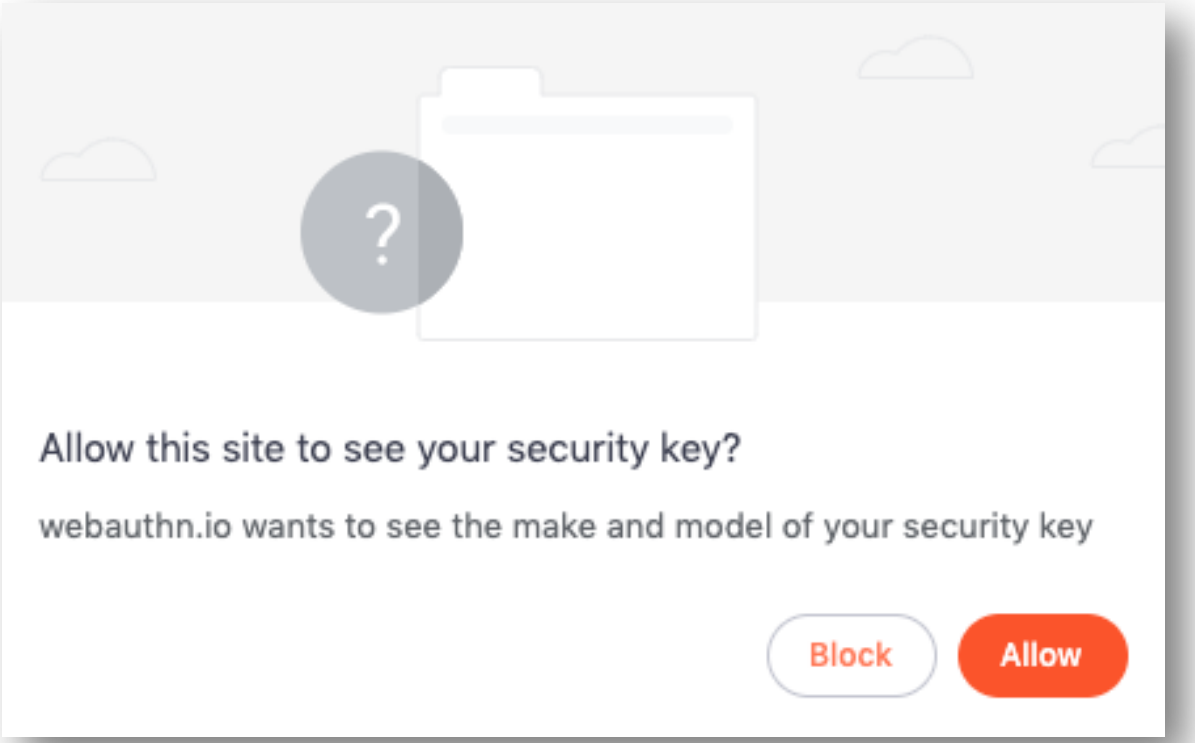
Browser support for authenticators through WebAuthn

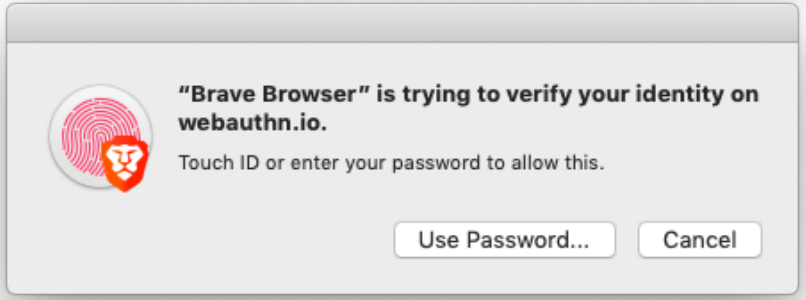
Windows 10

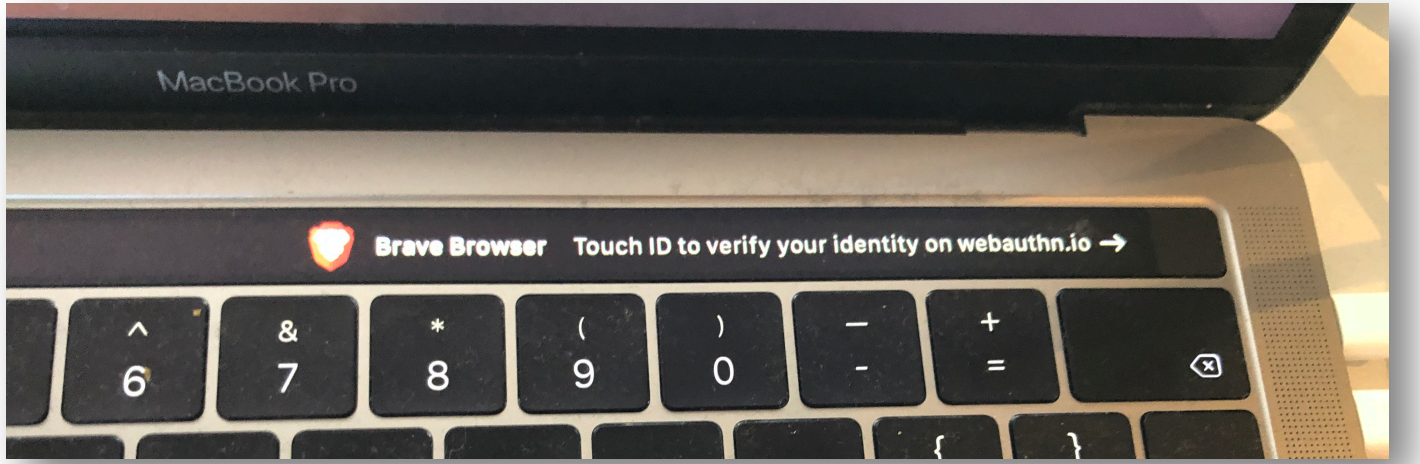
During registation, a new key pair is generated on the authenticator




Verify your identity with webauthn.io
Pick an option
USB security key
Built-in sensor
Cancel

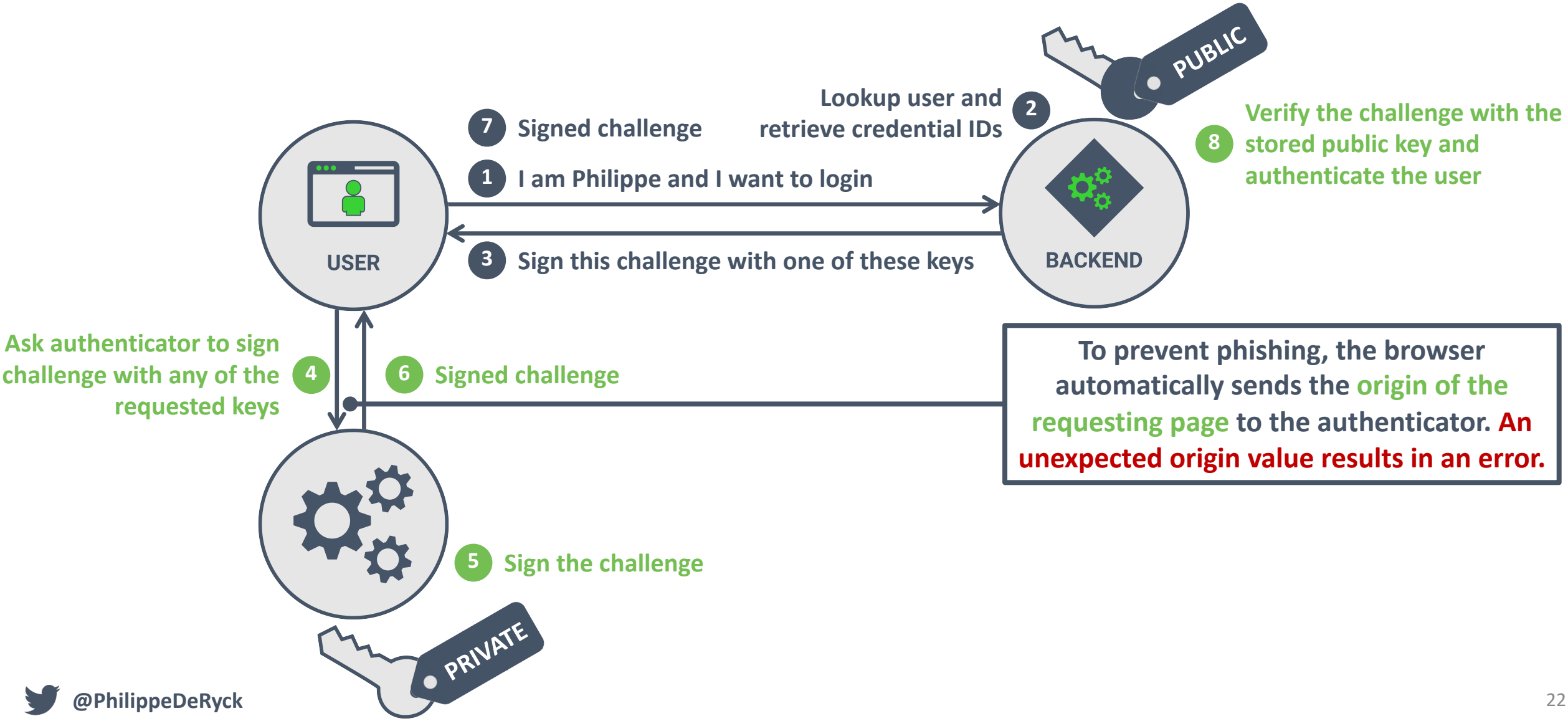

Allow this site to see your security key?
webauthn.io wants to see the make and model of your security key
Block
Allow


"Brave Browser" is trying to verify your identity on webauthn.io.
Touch ID or enter your password to allow this.
Use Password...
Cancel

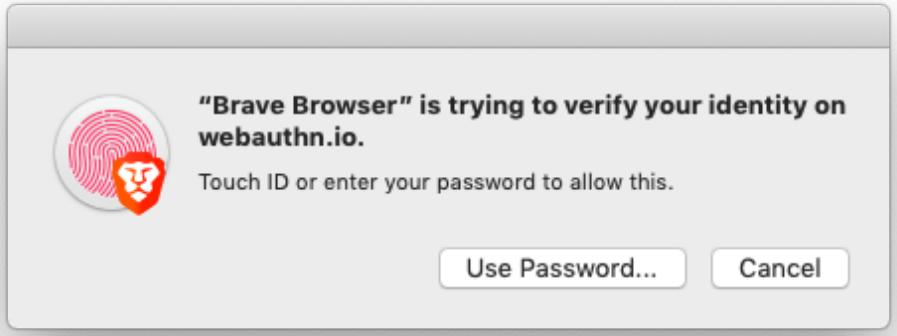

MacBook Pro
Brave Browser
Touch ID to verify your identity on webauthn.io

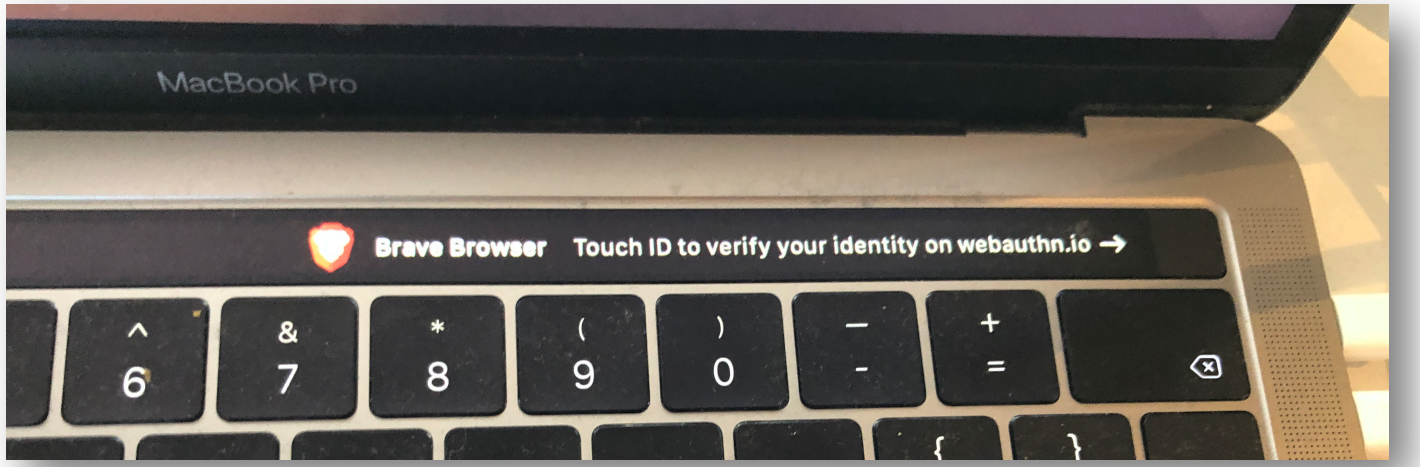
```
1  navigator.credentials.create({ publicKey: {
2    // The relying party (server)
3    rp: { name: "Restograde",
4          id: "restograde.com"
5    },
6
7    // User-specific properties, including a unique ID
8    user: { id: Uint8Array.from("ABCD1234", c => c.charCodeAt(0)),
9            name: "philippe@pragmaticwebsecurity.com",
10           displayName: "Philippe De Ryck"
11   },
12
13   //-7 refers to Elliptic Curve pubkeys with SHA-256 signatures
14   pubKeyCredParams: [{ type: "public-key", alg: -7 }],
15
16   authenticatorSelection: { authenticatorAttachment: "cross-platform" },
17   attestation: "direct",
18   timeout: 60000,
19   challenge: ... // A cryptographically random challenge from the server
20 }}).then(…)
```

During authentication, a challenge is signed by the key from the authenticator



To prevent phishing, the browser automatically sends the origin of the requesting page to the authenticator. An unexpected origin value results in an error.


"Brave Browser" is trying to verify your identity on webauthn.io.
Touch ID or enter your password to allow this.
Use Password...
Cancel


MacBook Pro
Brave Browser
Touch ID to verify your identity on webauthn.io →

The code to authenticate a user with a credential from the browser

```
1  const assertion = await navigator.credentials.get({
2    publicKey: {
3      challenge: Uint8Array.from(
4      randomStringFromServer, c => c.charCodeAt(0)),
5      allowCredentials: [{
6        id: Uint8Array.from(credentialId, c => c.charCodeAt(0)),
7        type: 'public-key',
8        transports: ['usb', 'ble', 'nfc'],
9      }],
10     timeout: 60000,
11   }
12 });
```

What about account recovery?

Without a remaining authenticator, users can be locked out

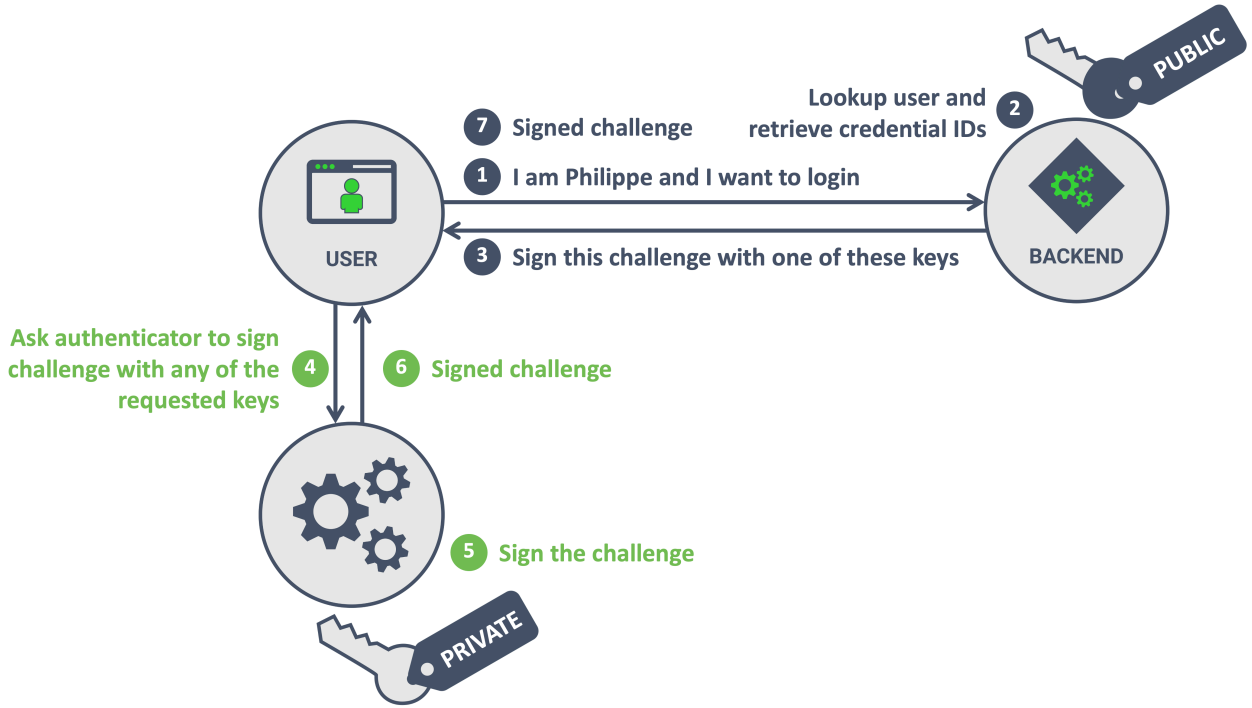
- Automatic recovery is possible, but likely represents a weakness (e.g., email access, phone number)
- Recovery from a trusted machine or existing installation (e.g., mobile app) has better security properties
- Enterprise applications can typically rely on technical support to handle account recovery

WebAuthn is a good candidate to replace passwords

- Highly recommended to offer technical users the option to use WebAuthn
- To avoid recovery issues, consider combining WebAuthn with passwords + MFA
- When sufficient authenticators are registered, allow the user to disable traditional authentication

Relying on WebAuthn

- *Offers strong cryptographic authentication*
- *Widely supported in modern browsers*
- *Supports hardware authenticators*
- *Flexible API and streamlined UX*



BENEFITS

Widely supported strong key-based authentication

Enables hardware keys as primary authenticator

Built-in privacy and phishing protection

DRAWBACKS

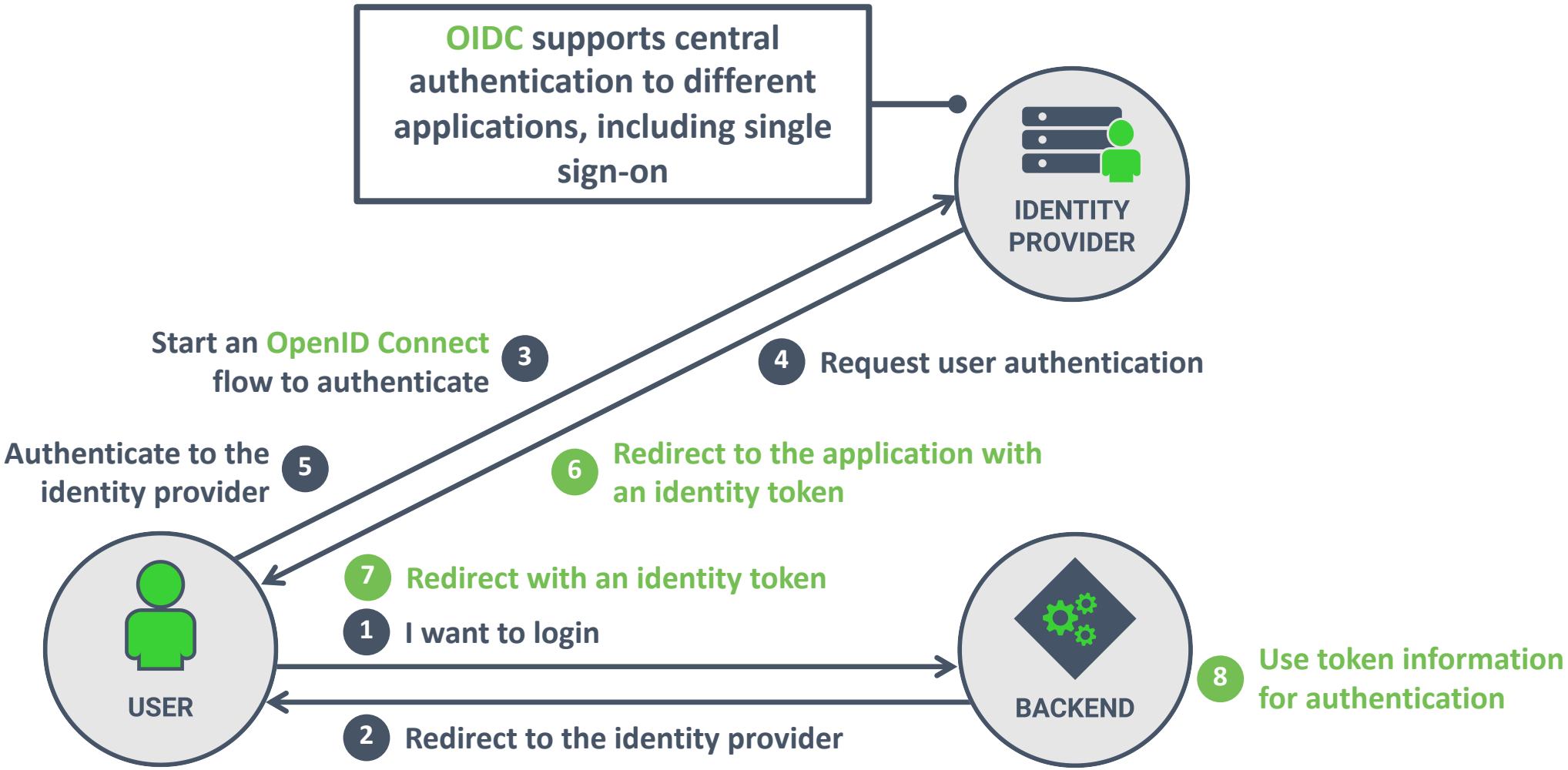
New mechanism and workflow for most users

Only a full password replacement in controlled environments

Server-side handling of signatures is very sensitive

RESTOGRADE

Login with Restograde


OIDC supports central authentication to different applications, including single sign-on
IDENTITY PROVIDER
Start an OpenID Connect flow to authenticate
3
4
Request user authentication
Authenticate to the identity provider
5
6
Redirect to the application with an identity token
7
Redirect with an identity token
1
I want to login
USER
BACKEND
8
Use token information for authentication
2
Redirect to the identity provider

The encoded identity token


```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ik5U
VkJPVFUzTXpCQk9FVXdOemhCUTBWR01rUTBRVVU1UVRZeFFV
VXlPVU5FUVVVeE5qRXlNdyJ9.eyJuaWNrbmFtZSI6InBoaWx
pcHBlIiwibmFtZSI6InBoaWxpcHBlQHByYWdtYXRpY3dlYnN
lY3VyaXR5LmNvbSIsInBpY3R1cmUiOiJodHRwczovL3MuZ3J
hdmF0YXIuY29tL2F2YXRhci9mNDBkNjRhNGIxNjc4OTUwODA
2MmU2NjRiZTZhZTU3NT9zPTQ4MCZyPXBnJmQ9aHR0cHMlM0E
lMkYlMkZjZG4uYXV0aDAuY29tJTJGYXZhdGFycyUyRnBoLnB
uZyIsInVwZGF0ZWRfYXQiOiIyMDIwLTA2LTA5VDA0OjE4OjA
0LjkwM1oiLCJlbWFpbCI6InBoaWxpcHBlQHByYWdtYXRpY3d
lYnNlY3VyaXR5LmNvbSIsImVtYWlsX3ZlcmlmaWVkIjp0cnV
lLCJpc3MiOiJodHRwczovL3N0cy5yZXN0b2dyYWRlLmNvbS8
iLCJzdWIiOiJhdXRoMHw1ZWI5MTZjMjU4YmRiNTBiZjIwMzY
2YzYiLCJhdWQiOiJGTjk4M0NFWWd4NG1kVWczTktOS0hqd2Z
OQUw1RmI0MiIsImlhdCI6MTU5MTY3NjI5MCwiZXhwIjoxNTk
xNzEyMjkwfQ.m60Br25jY8MOwIpCAjv3tRYF7IMR11ydzaP1
m6qJwsX74Sr5WUh49IK3iwaK72U6r2KXAp3_Oys9aabdoSc6
EkiYo7sho2W_fbLrUz8ocHFcTdHemuM0zoDQ6lVgobDNiwtl
eht8iNnIf9ghlRa-
TBtuL0TIRxkSHsCuJHKlWEG7zVHwll1q34XcLtkq4mnjWKlM
P5dNZoqIB_0Gek-EG05nUuoYwK7IqaZIGFLgc4EaK0fel-
MIqqDAwiD3etAkILSu7Phejk6zHwuEQlt3YzlbP5ZHNPK5hn
Sph80BPL7VMdDUWhjMdl1eW21cRq5CQNIKAJDbVLDdWqemO9
Kp_A
```


The decoded JWT payload


```
1  {
2    "nickname": "philippe",
3    "name": "philippe@pragmaticwebsecurity.com",
4    "picture": "https://s.gravatar.com/....png",
5    "updated_at": "2020-06-09T04:18:04.903Z",
6    "email": "philippe@pragmaticwebsecurity.com",
7    "email_verified": true,
8    "iss": "https://sts.restograde.com/",
9    "sub": "auth0|5eb916c258bdb50bf20366c6",
10   "aud": "FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42",
11   "iat": 1591676290,
12   "exp": 1591712290
13 }
```

The decoded JWT payload

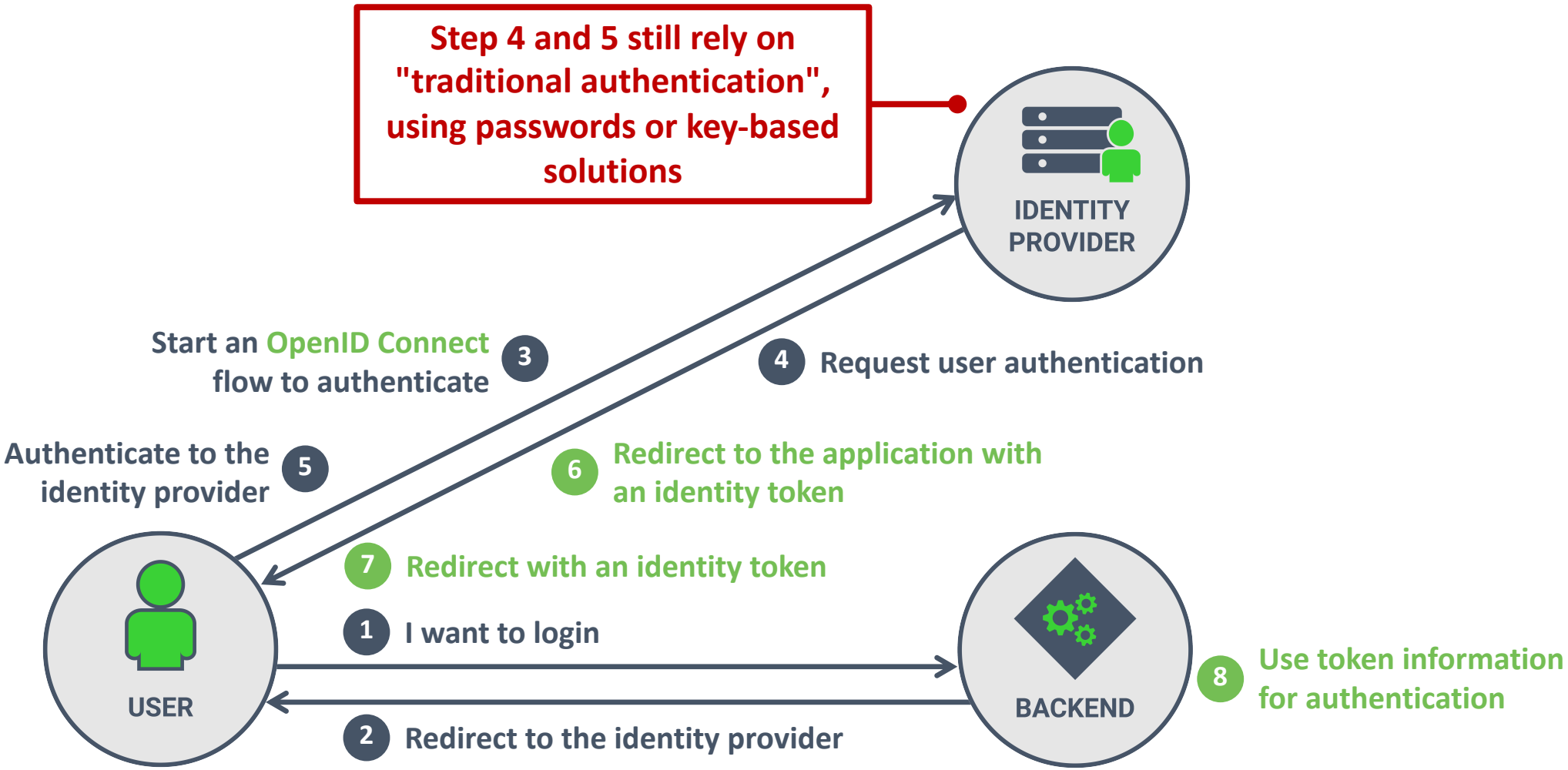
User account information from Restograde

The issuer of the identity token (STS)

The user's unique ID within the STS

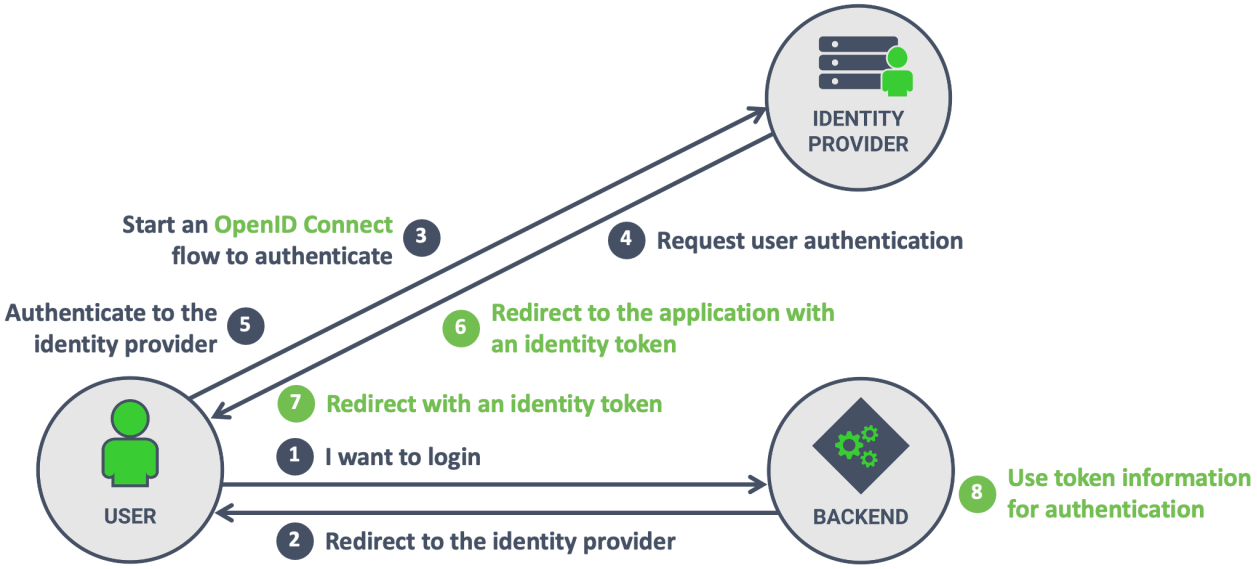
The audience of the identity token (client)

```
1  {
2    "nickname": "philippe",
3    "name": "philippe@pragmaticwebsecurity.com",
4    "picture": "https://s.gravatar.com/....png",
5    "updated_at": "2020-06-09T04:18:04.903Z",
6    "email": "philippe@pragmaticwebsecurity.com",
7    "email_verified": true,
8    "iss": "https://sts.restograde.com/",
9    "sub": "auth0|5eb916c258bdb50bf20366c6",
10   "aud": "FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42",
11   "iat": 1591676290,
12   "exp": 1591712290
13 }
```


Step 4 and 5 still rely on "traditional authentication", using passwords or key-based solutions
IDENTITY PROVIDER
Start an OpenID Connect flow to authenticate
3
4
Request user authentication
Authenticate to the identity provider
5
6
Redirect to the application with an identity token
7
Redirect with an identity token
1
I want to login
USER
BACKEND
8
Use token information for authentication
2
Redirect to the identity provider

Using OpenID Connect

- *Offloads authentication to a central service*
- *Reduces the application's responsibilities*
- *Enables tighter security*
- *Supports Single Sign-On (SSO)*



BENEFITS

Supports multiple clients with a single user account

Centralized security controls give more control

Works well with an (enterprise) application portfolio

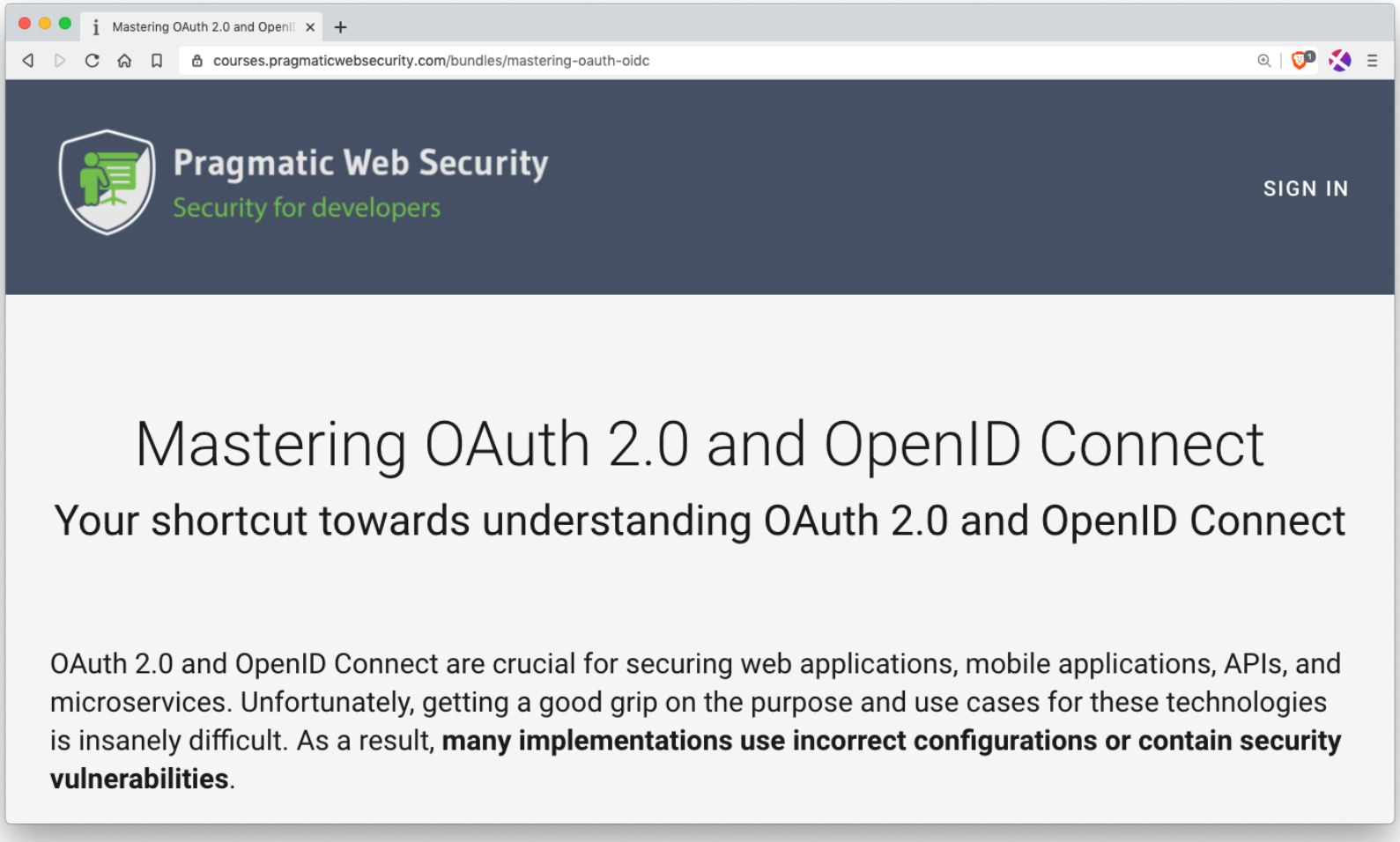
DRAWBACKS

No replacement for user authentication

Requires an identity provider (self-hosted/as a service)

OIDC is complex to understand and tricky to get right

This online course condenses dozens of confusing specs into a crystal-clear academic-level learning experience



25% discount

Use coupon code

LOCOMOCOSEC

Offer expires November 11th, 2020

Password-based authentication	Well-known but vulnerable mechanism, which should be **avoided when possible**
Authenticating with magic links	Simple authentication without user secrets for **non-sensitive applications**
Key-based authentication	Strong authentication mechanism, but **can be difficult to manage for web applications**
Relying on WebAuthn	Strong authentication mechanism, **highly recommended for web applications**
Using OpenID Connect	**Recommended to offload authentication to a dedicated service with tight security controls**

OWASP ASVS DEFINES DETAILED AUTHENTICATION REQUIREMENTS

V2: Authentication Verification Requirements

Control Objective

Authentication is the act of establishing, or confirming, someone (or something) as authentic and that claims made by a person or about a device are correct, resistant to impersonation, and prevent recovery or interception of passwords.

When the ASVS was first released, username + password was the most common form of authentication outside of high security systems. Multi-factor Authentication (MFA) was commonly accepted in security circles but rarely required elsewhere. As the number of password breaches increased, the idea that usernames are somehow confidential and passwords unknown, rendered many security controls untenable. For example, NIST 800-63 considers usernames and Knowledge Based Authentication (KBA) as public information, SMS and email notifications as "restricted" authenticator types , and passwords as pre-breached. This reality renders knowledge based authenticators, SMS and email recovery, password history, complexity, and rotation controls useless. These controls always have been less than helpful, often forcing users to come up with weak passwords every few months, but with the release of over 5 billion username and password breaches, it's time to move on.

Of all the sections in the ASVS, the authentication and session management chapters have changed the most. Adoption of effective, evidence-based leading practice will be challenging for many, and that's perfectly okay. We have to start the transition to a post-password future now.

USEFUL REFERENCES

- A primer on Decentralized IDs: https://w3c-ccg.github.io/did-primer/
- A guide on mTLS: https://textslashplain.com/2020/05/04/client-certificate-authentication/
- FIDO2 & WebAuthn: https://fidoalliance.org/fido2/
- WebAuthn implementation guide: https://webauthn.guide/
- OIDC explained: https://connect2id.com/learn/openid-connect
- OWASP ASVS: https://owasp.org/www-project-application-security-verification-standard/
- Additional talks on application security: https://pragmaticwebsecurity.com/talks.html
- Online courses: https://pragmaticwebsecurity.com/courses.html

Thank you for watching!

Connect on social media for more in-depth security content

@PhilippeDeRyck

/in/PhilippeDeRyck