

Move over  
passwords, passkeys  
are my new best  
friend

~~TOP SECRET~~

DAPHNE LIU





Daphne  
Liu

**CONFIDENTIAL**

- Frontend engineer  
@ Grow Therapy
- Previously Lyft,  
Shopify, Yelp
- Seattle, WA
- Improv & sketch  
comedy

@thebetterdaphne



# CRIME SCENE

1. Lack of security.
2. Human forgetfulness.

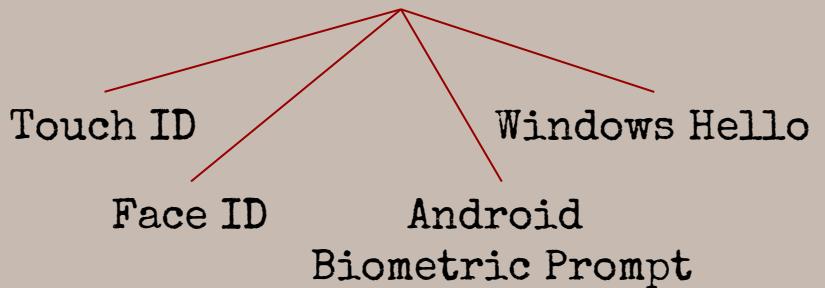
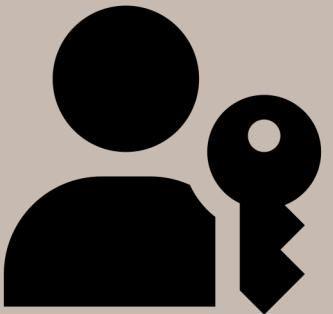
**TOP SECRET**





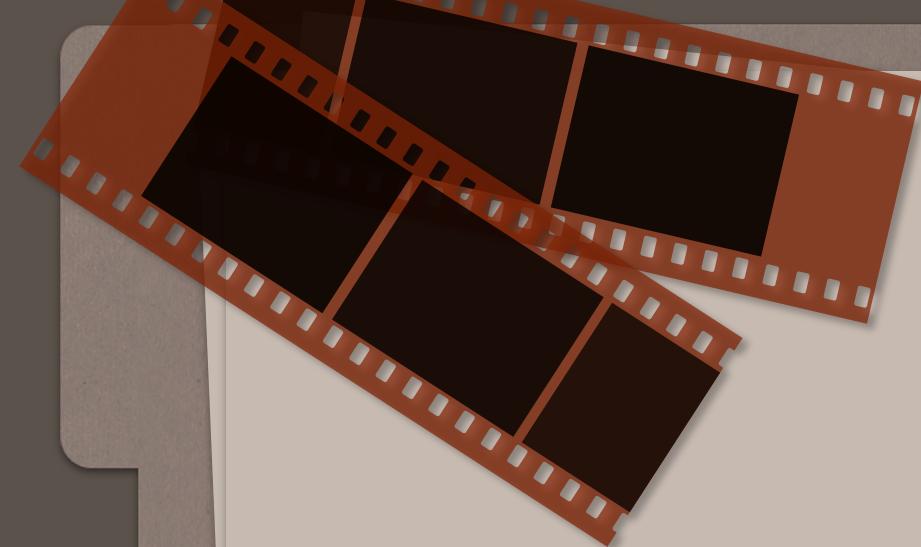
MURDER  
WEAPON:  
Passkeys.

# PASSKEYS.



Passkeys are a replacement for passwords that let users sign in with their device lock screen.



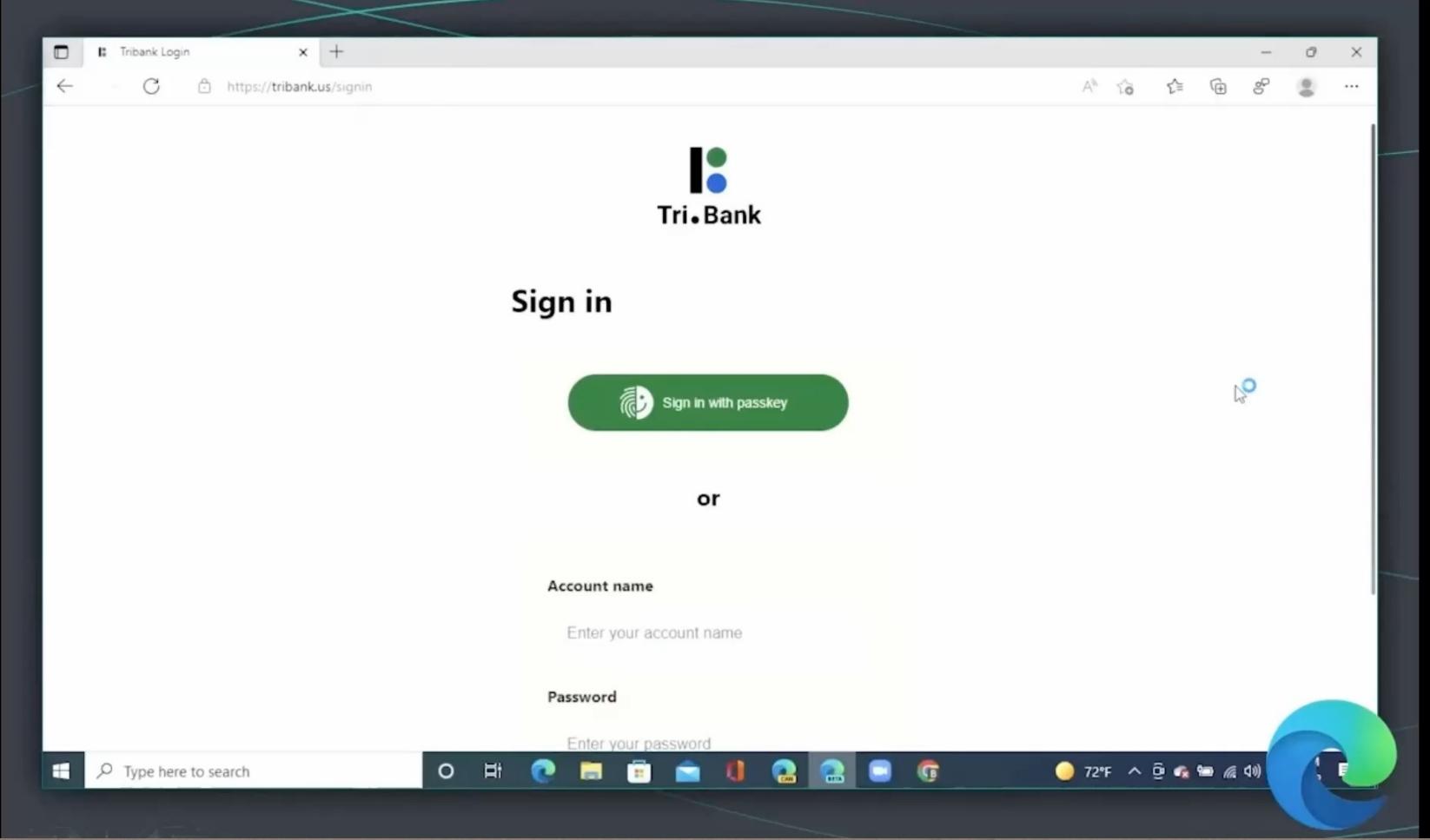


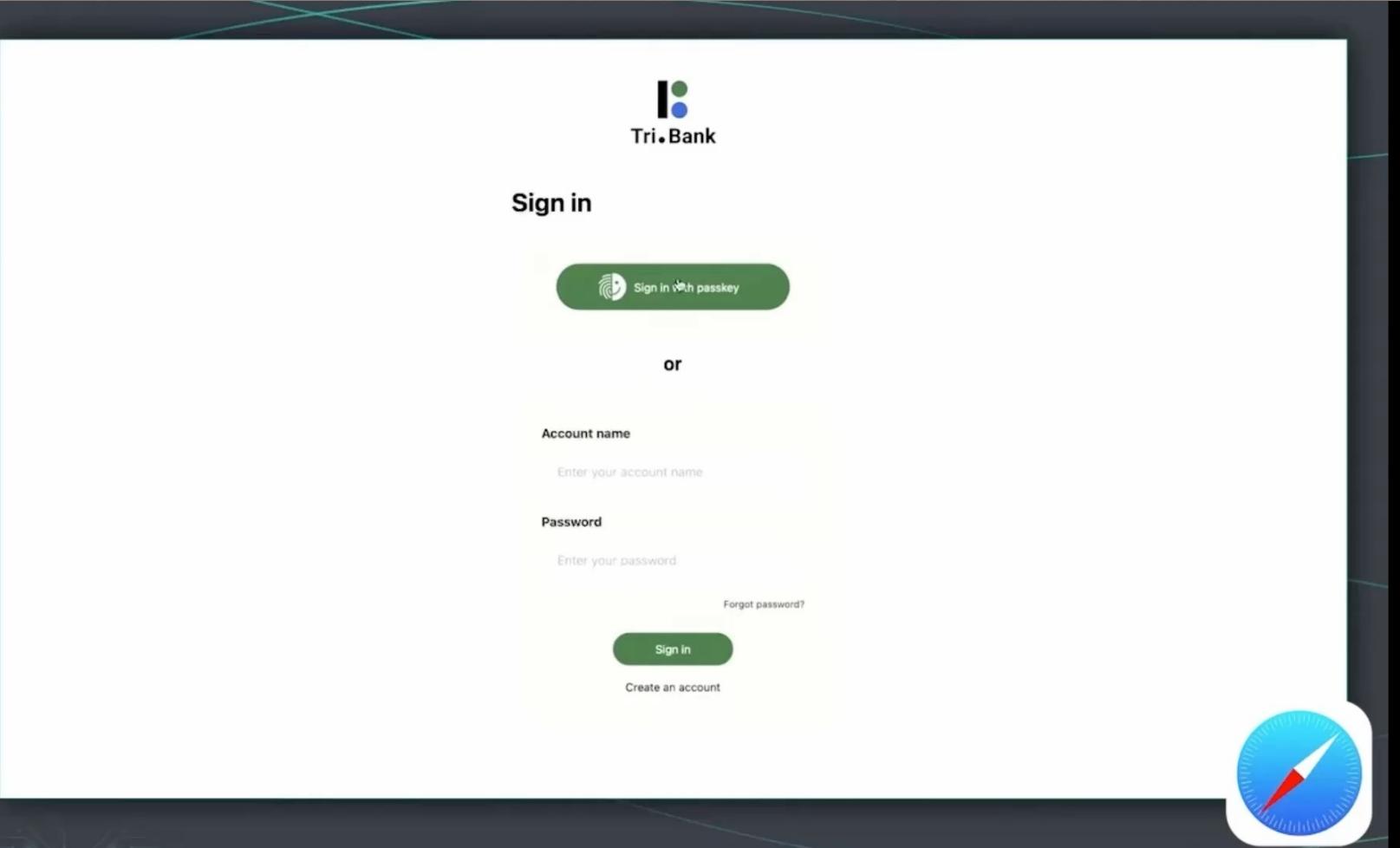
TOP SECRET

# Security Footage











# CLOUD SYNC



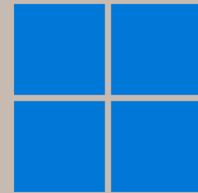
## iCloud Passwords

- Syncs iOS, iPadOS, macOS



## Google Password Manager

- Syncs Android
- No ChromeOS sync yet



## Windows Hello

- No Windows sync yet



## 1Password

- Syncs Windows, Mac, Linux



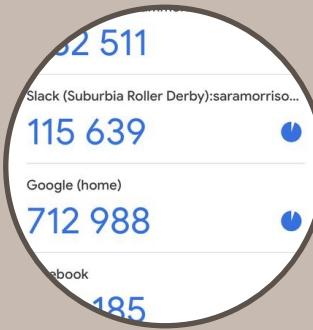
## Bitwarden

- Syncs Windows, Mac, Linux

# SECURITY COMPARISON.



Password



Password +  
2FA code

- Google  
Authenticator,  
Authy
- SMS or email  
link



Passkeys

- Biometrics
- Apple Touch ID,  
Face ID
- Windows Hello
- In password  
manager



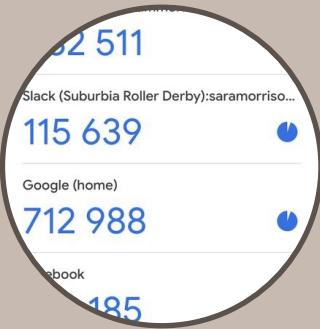
Hardware  
security key

- Yubikey
- In physical  
device

Least secure

Most secure

# CONVENIENCE COMPARISON.



## Password + 2FA code

- Need to lookup code in app



## Password

- Need to memorize unique passwords



## Hardware security key

- Need to carry Yubikey



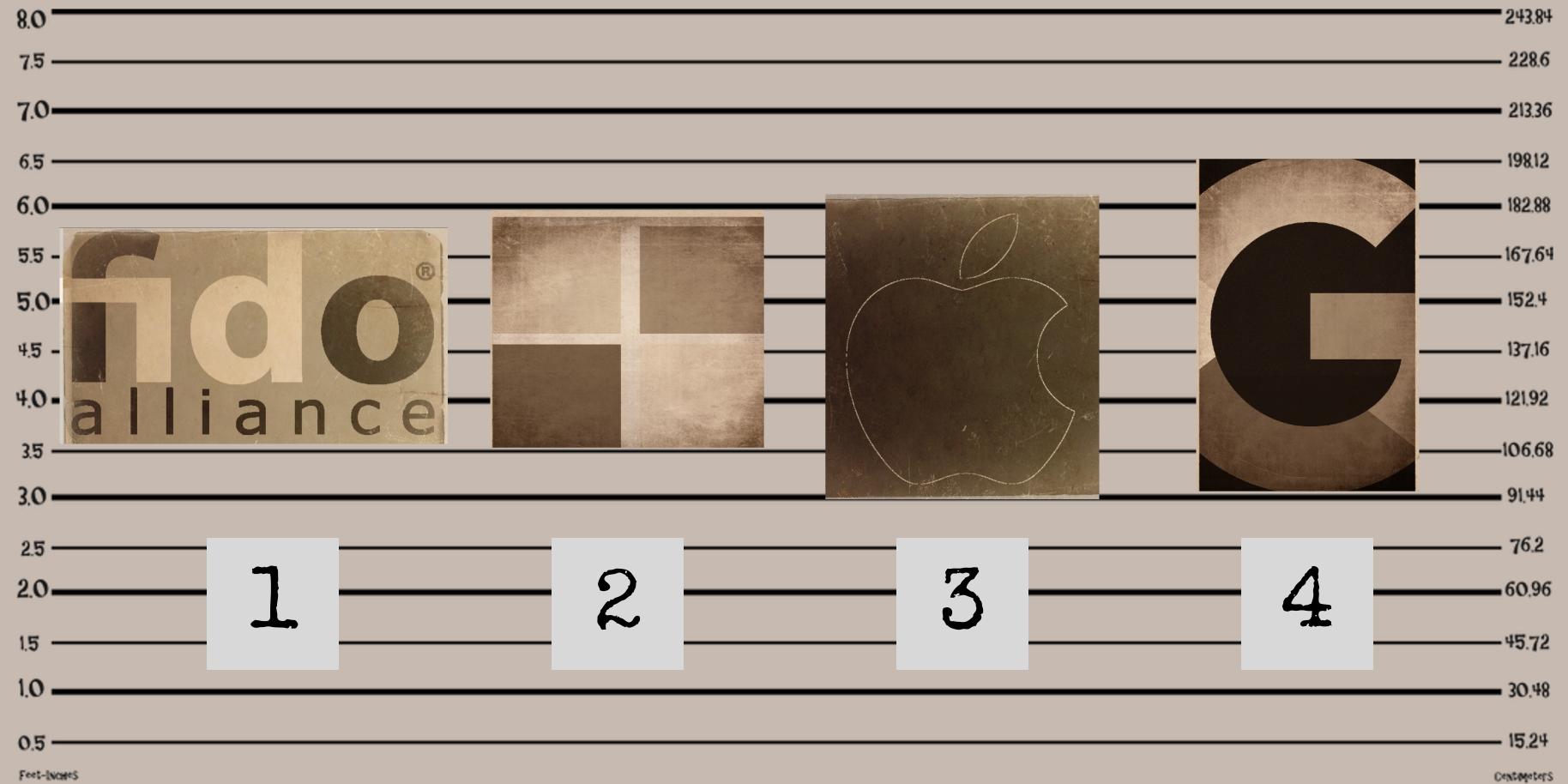
## Passkeys

- Built into phone/computer

Least convenient

Most convenient

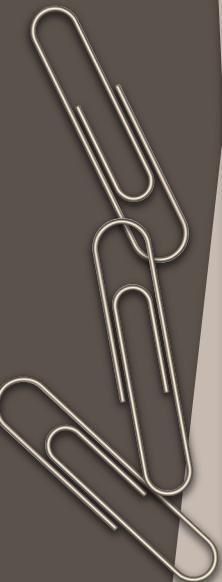
# SUSPECTS

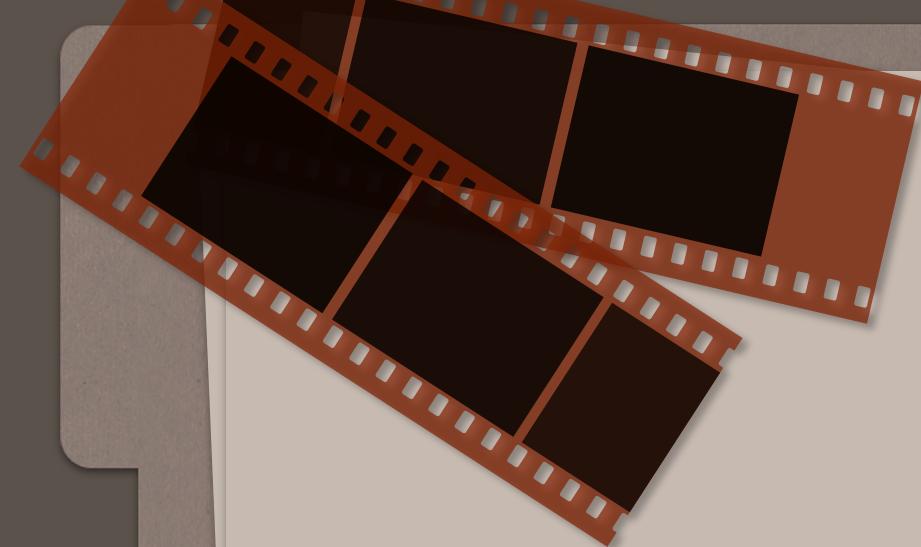


**CLASSIFIED**

*Definitely*

~~-Don't~~ try this at home!





TOP SECRET

Let's learn about  
how passkeys work

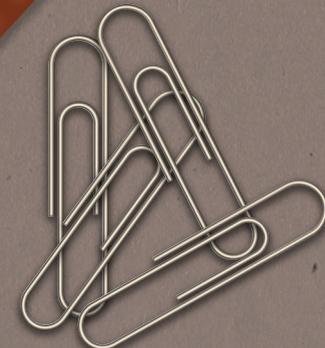
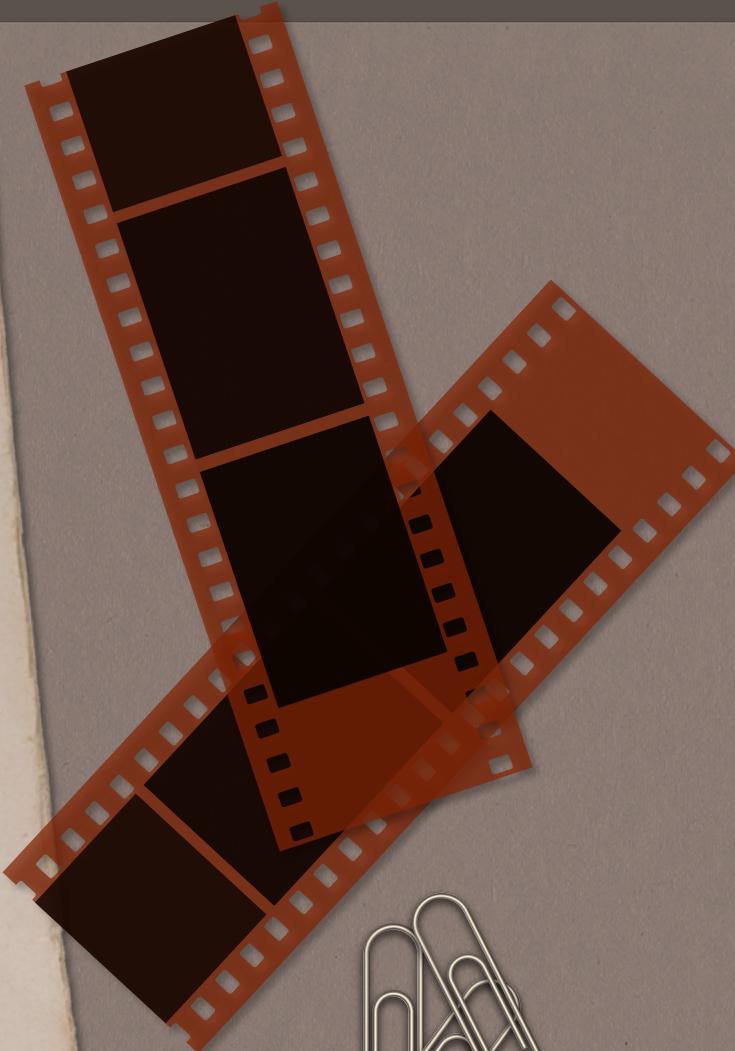


ALSO  
KNOWN AS.

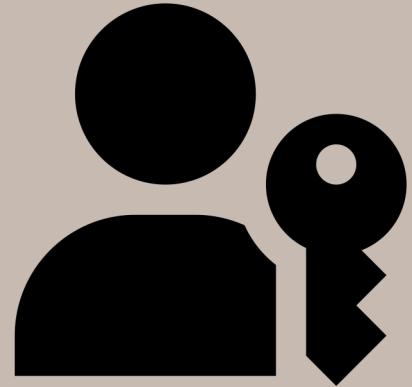
“Passkeys”

A.K.A.

Discoverable Credential

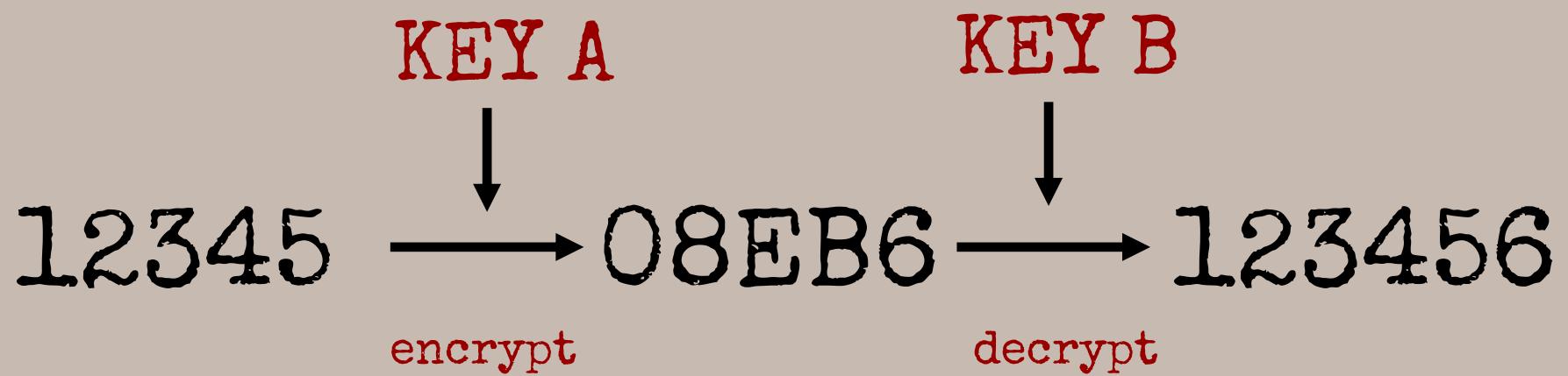


# PASSKEYS



Passkeys are secure as they're built with  
public key cryptography.

# ENCRYPTION.



# PUBLIC KEY CRYPTOGRAPHY.

A.K.A. Asymmetric encryption

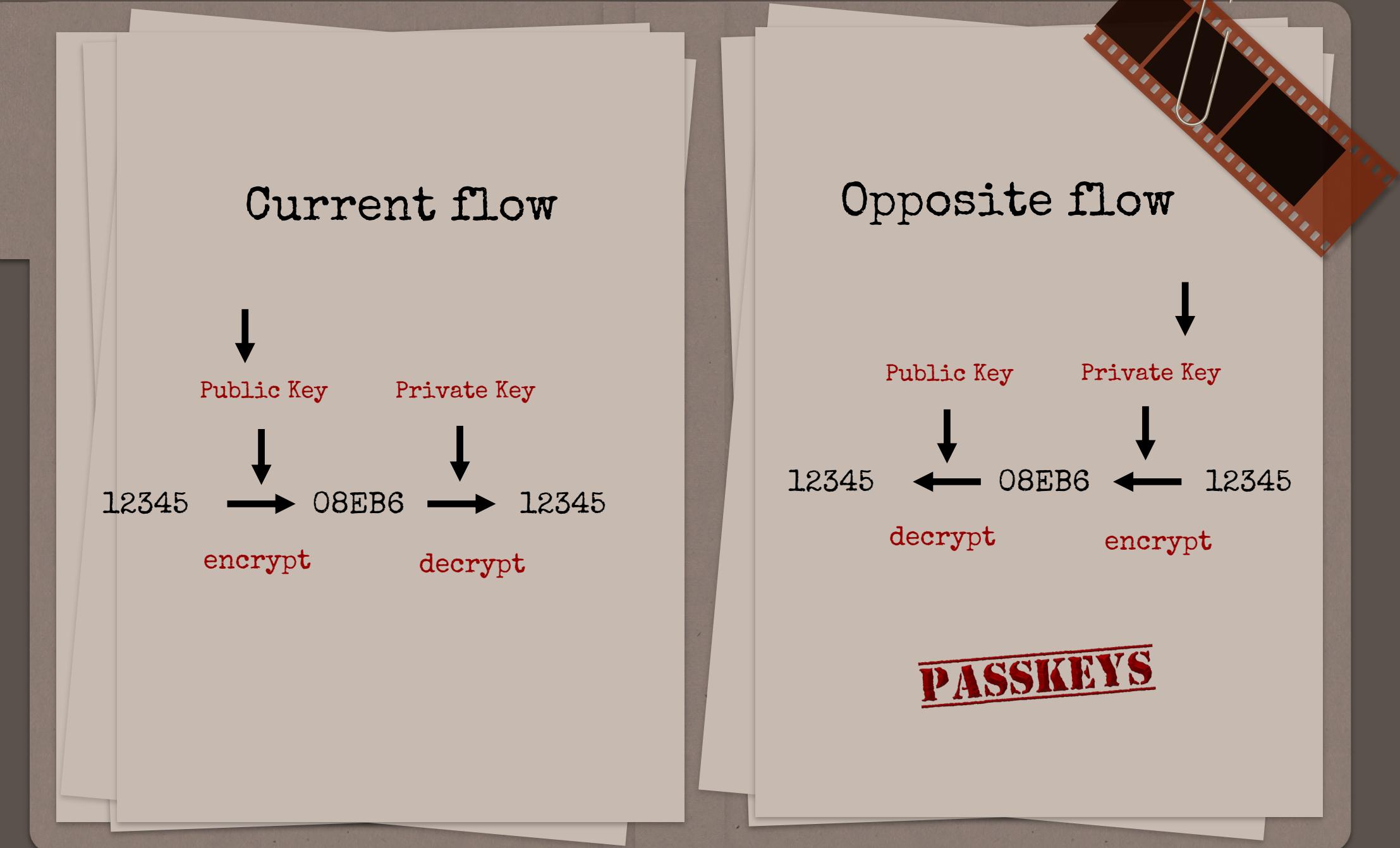
“Mailing address”

Public Key

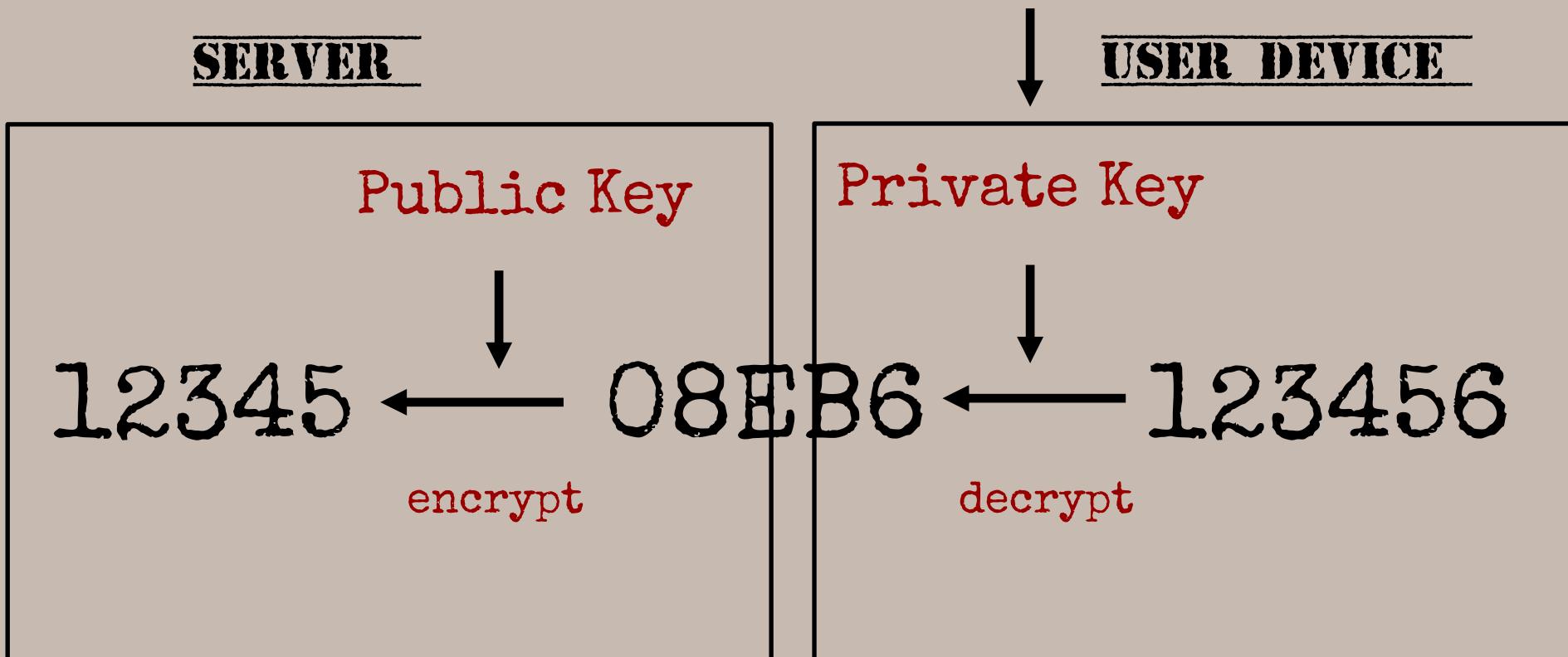
“physical mailbox key”

Private Key





# PUBLIC KEYS AND PASSKEYS



**Love Bytes: Me and a Server's Heart**  
A fanfic by Daphne

# WEB AUTHENTICATION API.

A.K.A.

# WebAuthn

Built-in API used  
to generate  
public and  
private keys for  
login.



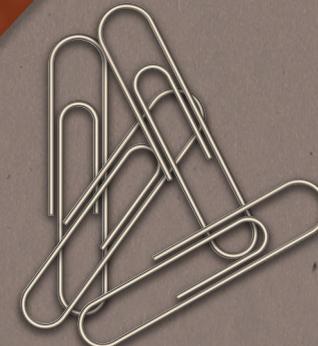
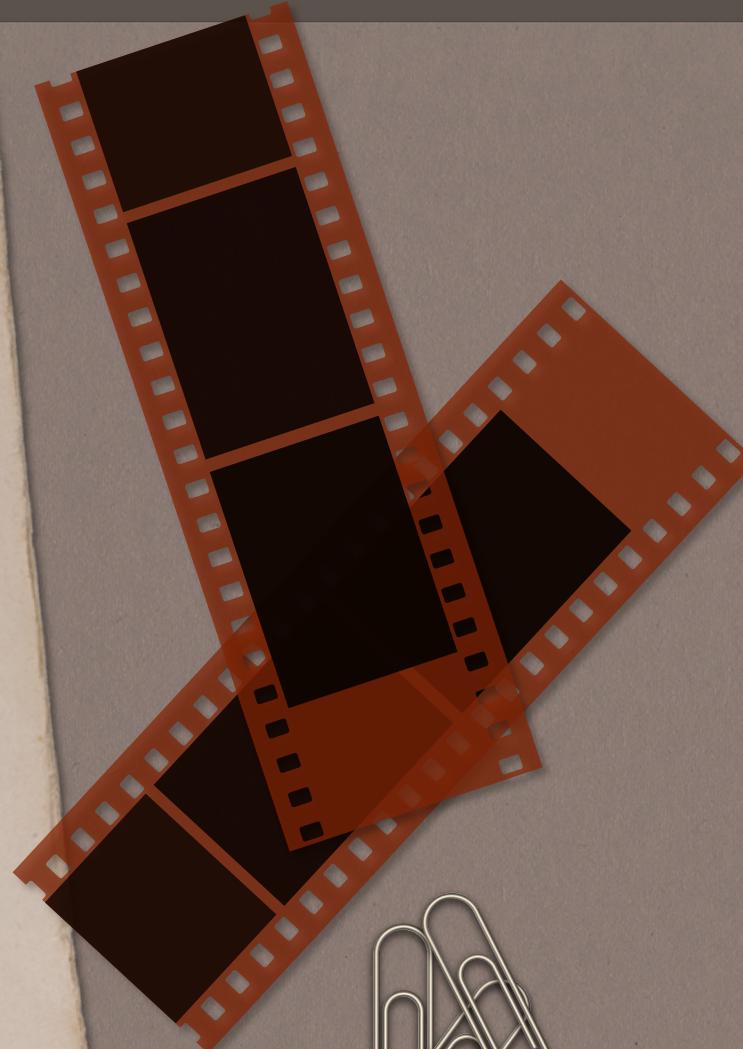
# WEBAUTHN USE CASES.

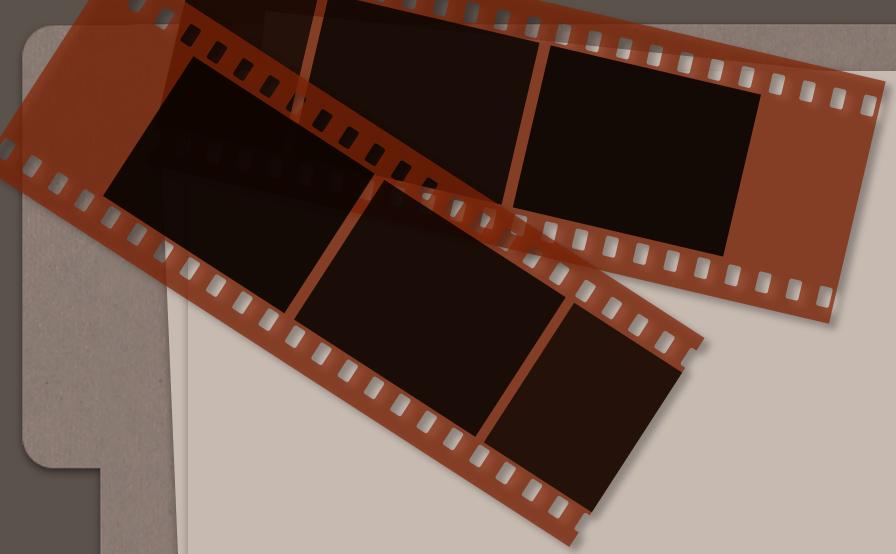
1. Supplement passwords – using a hardware device as second factor (ex. Yubikey)
2. Replace passwords – using a hardware device as a single factor (passkeys)

# PASSKEY FLOWS.

1. Registration
2. Authentication

**TOP SECRET**





TOP SECRET

# Registration

A.K.A. Attestation

## Creating a new passkey

# REGISTRATION.

Passkey Starter Kit [Home](#) [Sign In](#) [Sign up](#) [Test Panel](#)

Welcome myuser

My passkeys



Security Key  
Reg time: 3/30/2023, 3:31:41 PM  
Last used: 3/30/2023, 3:31:41 PM

[Update](#) [Delete](#)

[Add a new passkey](#)

Logout

[Click to sign out of your account](#)

Created by **yubico®**

Made with <3 by Yubico's Developer Program

Useful passkey resources

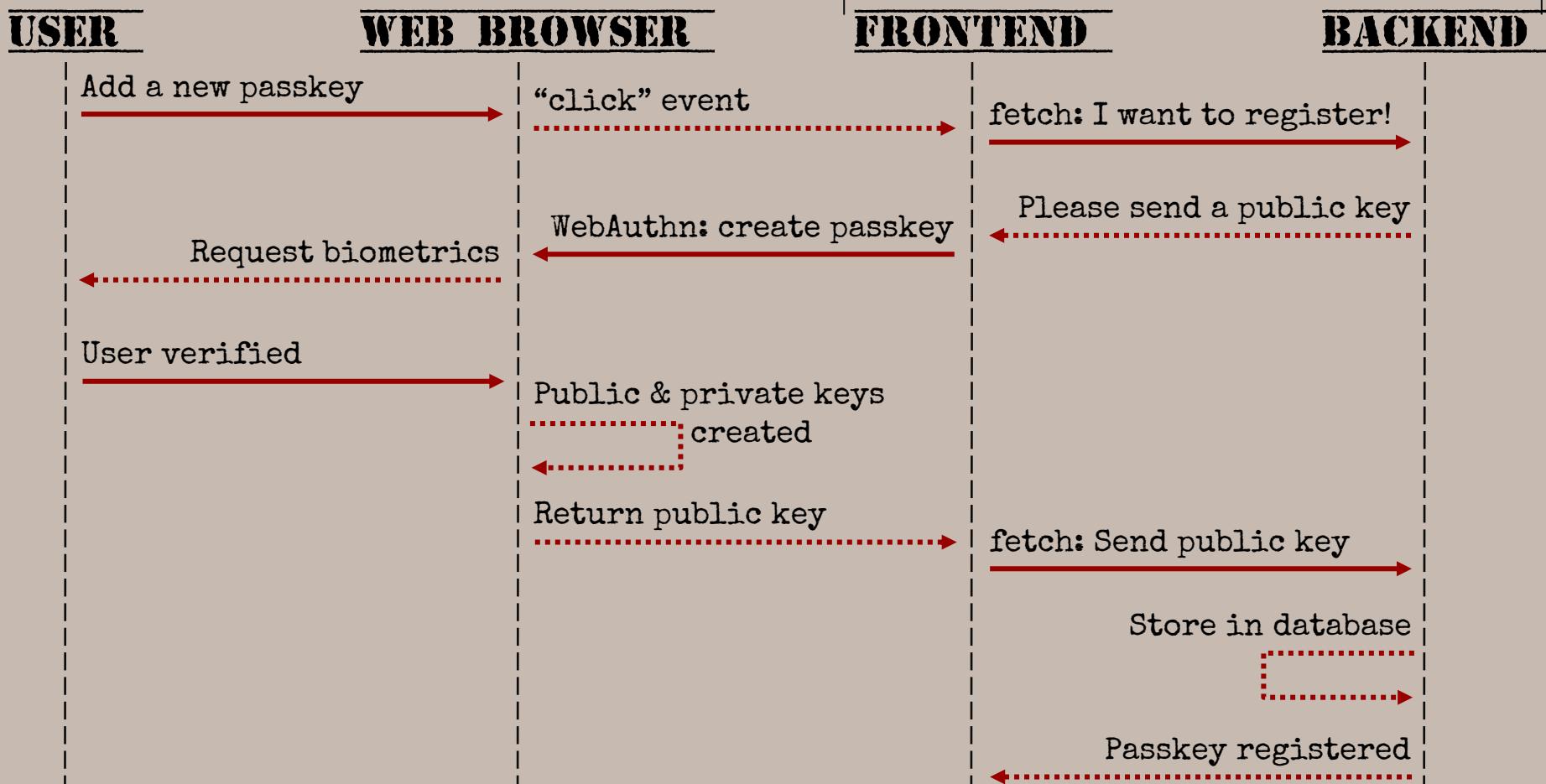
- [Passkey developer guidance](#)
- [Passkey demo](#)
- [passkeys.dev](#)

[yubicolabs](#)

# REGISTRATION.

Your Website

A.K.A. Relying Party



## FRONTEND

```
/**  
 * API call to your server  
 * Should return a 'PublicKeyCredentialCreationOptions'  
 */  
async function startRegistration(): Promise<PublicKeyCredentialCreationOptions> {  
    await fetch(/*...*/);  
}  
  
/**  
 * API call to your server  
 * Should send newPublicKey to your server for storage  
 */  
async function postNewPublicKey(newPublicKey: AuthenticatorAttestationResponse) {  
    await fetch(/*...*/, { method: "POST", body: newPublicKey });  
}  
  
async function registerPasskey() {  
    const options = await startRegistration();  
    const newPublicKey = await navigator.credentials.create({  
        publicKey: options,  
    });  
    await postNewPublicKey(newPublicKey.response);  
}  
  
<button onclick="registerPasskey()">Add a new passkey</button>
```

## BACKEND

```
const options: PublicKeyCredentialCreationOptions = {
  // Your website origin and name
  rp: { id: "themysterymachine.com", name: "The Mystery Machine" },
  user: {
    id: new Uint8Array([79, 252, 83, 72, 214, 7, 89, 26]), // database ID
    displayName: "Daphne Liu", // human friendly name
    name: "daphneliu", // user name to distinguish between same display name
  },
  // randomly generated byte array
  challenge: new Uint8Array([117, 61, 252, 231, 191, 241, ...]),
  pubKeyCredParams: [{ type: "public-key", alg: -7 }]
};
```



# Authentication

A.K.A.  
*Assertion*

Log in with a passkey

# AUTHENTICATION.

yubico labs

Welcome

Log in to the WebAuthn Starter Kit to continue

@ Username

Continue

Forgot Your Security Key?

OR

Continue with Trusted Device or Security Key

Don't have an account? [Sign Up](#)

WebAuthn Starter Kit Reference Architecture

[yubico.com](#)

[yubicolabs](#)

# AUTOFILL AUTHENTICATION.

Passkey Starter Kit Home Sign In Sign up Test Panel

Sign In

Username

Enter username

Sign in with passkey ?

Need an account?

Click here to register for a new account

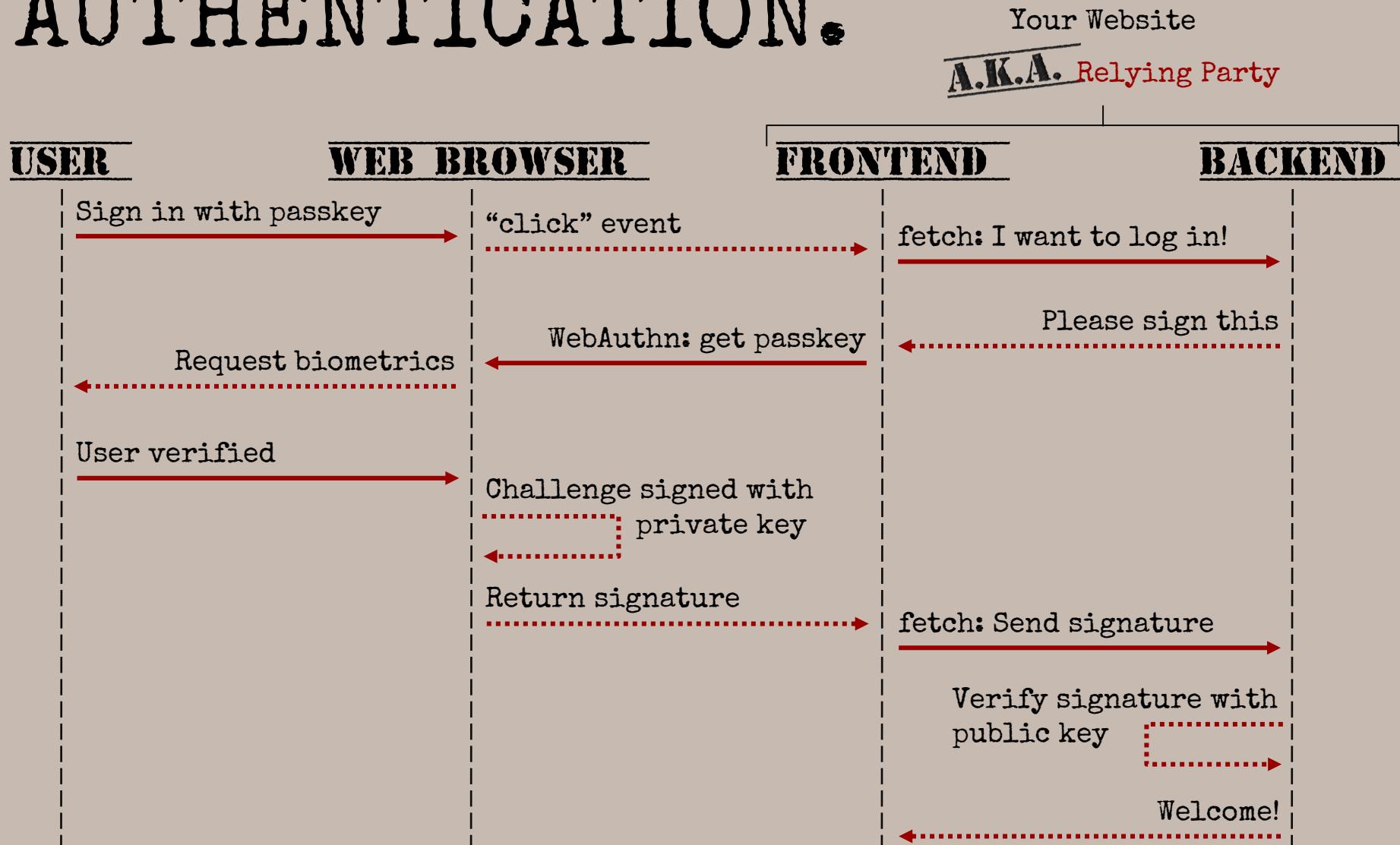
Created by **yubico**

Useful passkey resources

- [Passkey developer guidance](#)
- [Passkey demo](#)
- [passkeys.dev](#)

Made with <3 by Yubico's Developer Program

# AUTHENTICATION.



# PASSKEY MODAL.

- Pops up over web UI, so should be behind a button.
- Can be setup when user clicks button.

# PASSKEY AUTOFILL.

A.K.A. Conditional UI

- Appears as dropdown under username input.
- Needs to be setup ahead of time.

## FRONTEND

```
/**  
 * API call to your server  
 * Should return a 'PublicKeyCredentialRequestOptions'  
 */  
async function startAuthentication(): Promise<PublicKeyCredentialRequestOptions> {  
    await fetch(/*...*/);  
}  
  
/**  
 * API call to your server  
 * Should send signature to your server to check if it's valid  
 */  
async function postNewSignature(signatureResult: AuthenticatorAssertionResponse):  
Promise<boolean> {  
    await fetch(/*...*/, { method: "POST", body: signatureResult });  
}  
  
async function loginWithPasskey() {  
    const options = await startAuthentication();  
    const signatureResult = await navigator.credentials.get({ publicKey: options });  
    const valid = await postNewSignature(signatureResult.response);  
    if (valid) {  
        // User is logged in  
    }  
}  
  
<button onclick="loginWithPasskey()">Sign in with passkey</button>
```

## FRONTEND

```
async function waitForAutofillPasskeyLogin() {
  const isSupported = await PublicKeyCredential.isConditionalMediationAvailable?.();
  if (!isSupported) return;

  const options = await startAuthentication();
  try {
    // Will be waiting until autofill is called
    const signatureResult = await navigator.credentials.get({
      publicKey: options,
      mediation: "conditional",
    });
    const valid = await postNewSignature(signatureResult.response);
    if (valid) {
      // User is logged in
    }
  } catch {
    // Autofill was cancelled
  }
}

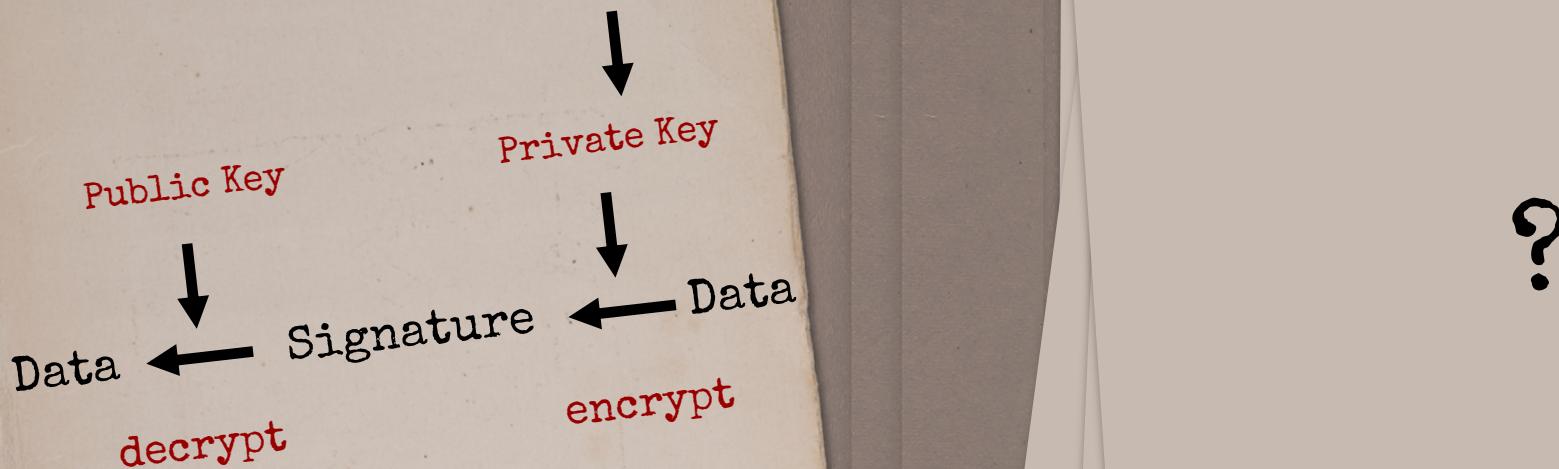
window.onload = waitForAutofillPasskeyLogin;

<input type="text" id="username-field" autocomplete="username webauthn" />
```

## BACKEND

```
const options: PublicKeyCredentialRequestOptions = {  
  // Your website origin  
  rpId: "themysterymachine.com",  
  // randomly generated byte array  
  challenge: new Uint8Array([117, 61, 252, 231, 191, 241, ...]),  
};
```

# WHAT GETS ENCRYPTED?



# OVERSIMPLIFICATION

```
interface AuthenticatorAssertionResponse {  
    clientDataJSON: ArrayBuffer; // unencrypted stuff that was hashed then signed  
    authenticatorData: ArrayBuffer; // more unencrypted stuff that was signed  
  
    signature: ArrayBuffer; // encrypted signature  
  
    userHandle: ArrayBuffer; // database ID, same as user.id  
}
```

# WebAuthn.io

## Using WebAuthn

Add WebAuthn to your site with one of these libraries:

### Python

-  [duo-labs/py\\_webauthn](#)
-  Duo Labs
-  Library

### TypeScript

-  [SimpleWebAuthn](#)
-  Matthew Miller
-  Library

### TypeScript

-  [passwordless-id/webauthn](#)
-  Arnaud Dagnelies
-  Library

### Ruby

-  [cedarcode/webauthn-ruby](#)
-  Cedarcodes
-  Library

### Ruby

-  [ruby-passkeys/devise-passkeys](#)
-  Ruby passkeys
-  Library

### Ruby

-  [ruby-passkeys/warden-webauthn](#)
-  Ruby passkeys
-  Library

### Java

-  [webauthn4j/webauthn4j](#)
-  Yoshikazu Nojima
-  Library

### Java

-  [java-webauthn-server](#)
-  Yubico
-  Library

### Java

-  [vertx-auth/webauthn](#)
-  Eclipse Vert.x
-  Library



# WHY PASSKEYS?

1. More secure
  - Passkeys are bound to website origin and phishing resistant.
2. Easier
  - Familiar UI, don't need separate hardware.
3. Faster
  - 7 seconds to register and 3 seconds to authenticate.





# RESOURCES.

- <https://webauthn.io>
- <https://passkeys.dev/device-support>
- <https://passwordless.id>
- [https://developer.mozilla.org/en-US/docs/Web/API/Web.Authentication\\_API](https://developer.mozilla.org/en-US/docs/Web/API/Web.Authentication_API)
- <https://yubicolabs.github.io/passkey-workshop/>
- <https://github.com/w3c/webauthn/wiki/Explainer:-WebAuthn-Conditional-UI>
- <https://web.dev/passkey-form-autofill/>
- <https://developers.google.com/codelabs/passkey-form-autofill>
- [https://www.youtube.com/watch?v=wN5lpttf\\_Hc](https://www.youtube.com/watch?v=wN5lpttf_Hc)
- <https://www.youtube.com/watch?v=SWocv4BhCNg>
- <https://arstechnica.com/information-technology/2023/05/passkeys-may-not-be-for-you-but-they-are-safe-and-easy-heres-why/>

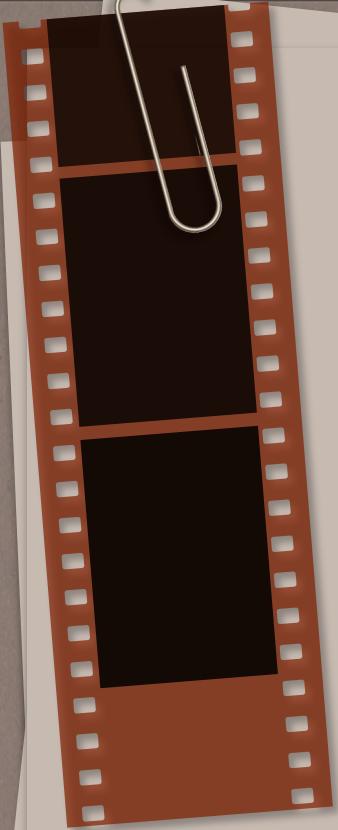


# PASSWORDS

\* \* \* \* \*

# PERPETRATORS





CASE CLOSED

THANK YOU!



@thebetterdaphne

[linkedin.com/in/daphliu](https://www.linkedin.com/in/daphliu)

[daphneliu.com/passkeys.pdf](http://daphneliu.com/passkeys.pdf)

# CREDITS.

Presentation  
Template:  
[SlidesMania](#)

Images: [Unsplash](#)

To get the old photo effect I've used a website called [funny.photo/old-photo-effect/](http://funny.photo/old-photo-effect/) and then removed the white background using [remove.bg](http://remove.bg)

And to generate the TOP SECRET, CLASSIFIED and CONFIDENTIAL texts I've used a website called [cooltextr.com](http://cooltextr.com)

