

# FIDO on macOS

How it works, attack vectors, and other learnings

Joel Rennich  
Identity Things  
Jamf  
@mactroll

**FIDO is more secure in every way than using a password**

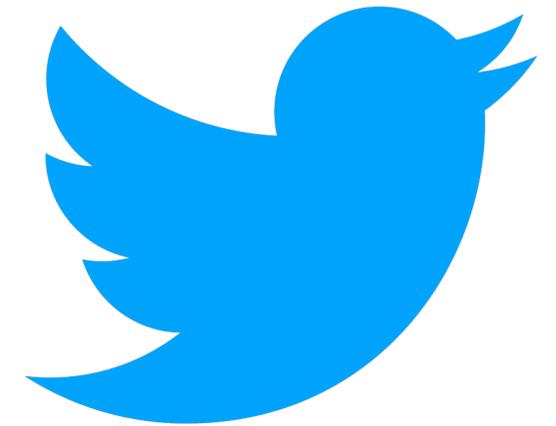
**Focus on macOS, but almost  
all of this carries over to iOS**

**FIDO in 30 secs...**

**As seen in...**



As seen in...



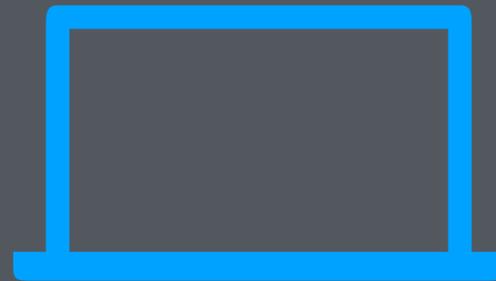
# FIDO Flow

<https://youridp.com>



**Relying Party**

**Mac**



**Platform**

**FIDO Key**



**Authenticator**

# FIDO Flow

<https://youridp.com>

Mac

FIDO Key



Relying Party

Platform

Authenticator



WebAuthn

JSON

# FIDO Flow

<https://youridp.com>

Mac

FIDO Key



Relying Party

Platform

Authenticator



WebAuthn

CTAP

JSON

CBOR



FEITIAN  
948174  
VJPDVQ

GoTrust  
fido CERTIFIED  
FOUO  
NFC  
N  
Bluetooth  
**Idem Card**  
BLE enabled NFC smart card  
Idem0774

THETIS

THETIS

 **TREZOR**



**coinstop.io**





CRAYONIC



# Leaks and Curiosities

# Type Casting

- FIDO has a strong focus on anonymity
  - Each key is unique
  - Attestations and AAGUIDs are done by batches of devices
- However... vendors do unique things
  - KeyID length
  - Sign Count
  - Adherence to standards

Demo

# Key Characteristics

	<b>Attestation</b>	<b>ID Length</b>	<b>Sign Count</b>	<b>AAGUID w/out Attestation</b>
Yubico	Direct	64 bytes	Device based	000000000000000000
Yubico Resident Key	Direct	16 bytes	Device based	000000000000000000
Thetis	Direct	96/129 bytes	Device based	000000000000000000
Thetis RK	Direct	16	Key based	000000000000000000
Feitian	Direct	96	Device based	000000000000000000
Feitian RK	Direct	32	Device based	000000000000000000
Apple Platform	It's complicated	20 bytes	Always 0	000000000000000000
Chrome Platform	Self	80 bytes	Unix time stamp	adce0235bcc6a648bb25f1f0553

Non-scientific, subject to change, not for use with nuclear reactors, medical devices or anything you care about.  
Author is not responsible for anything, including himself.

# Syncing Platform Authenticator

# Syncing Platform Authenticator

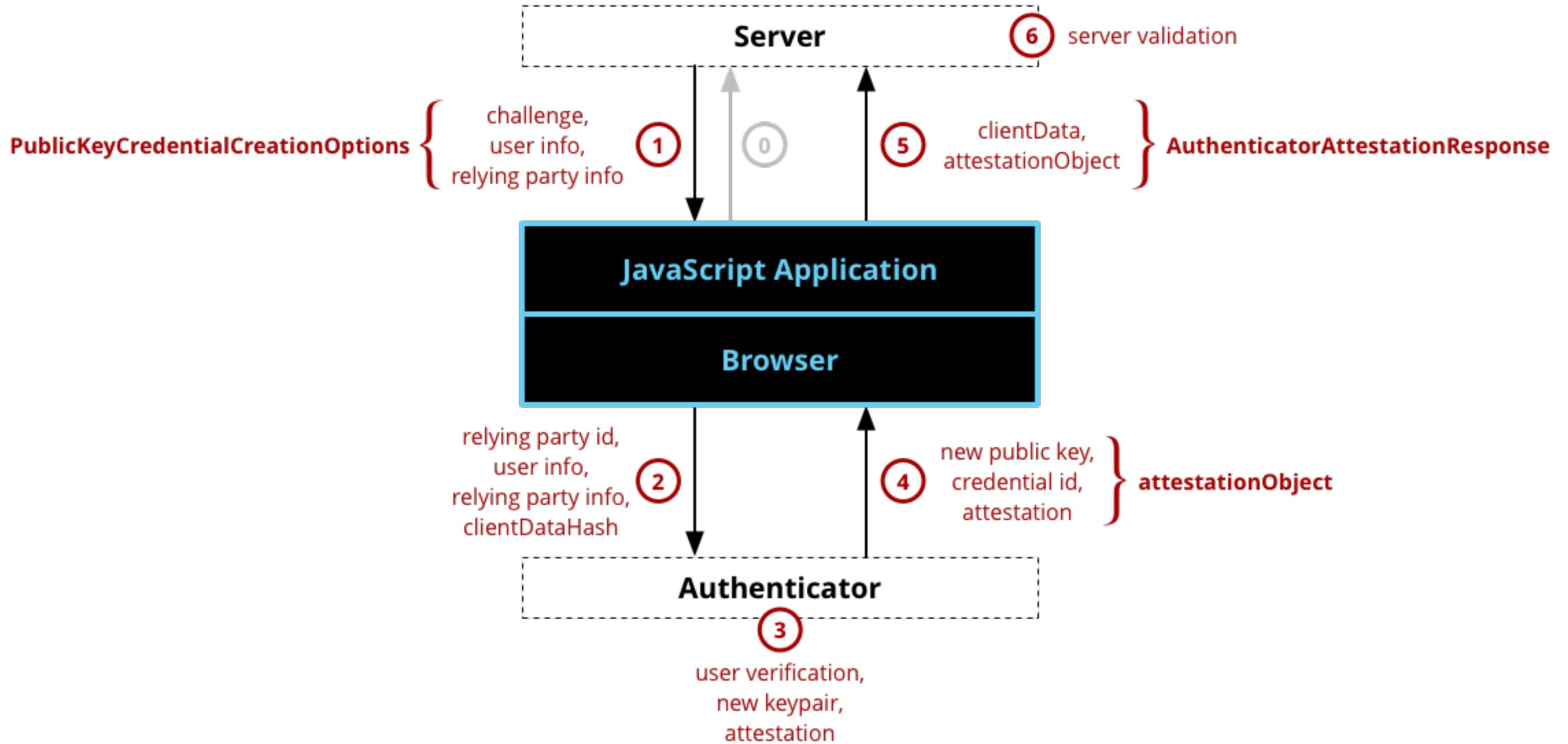
- “FIDO” keys kept in iCloud Keychain
- iOS 15 macOS Monterey
- Have to be turned on, and require an iCloud account
- Currently identify as a “platform” authenticator
  - 🤔
  - Will sync between devices

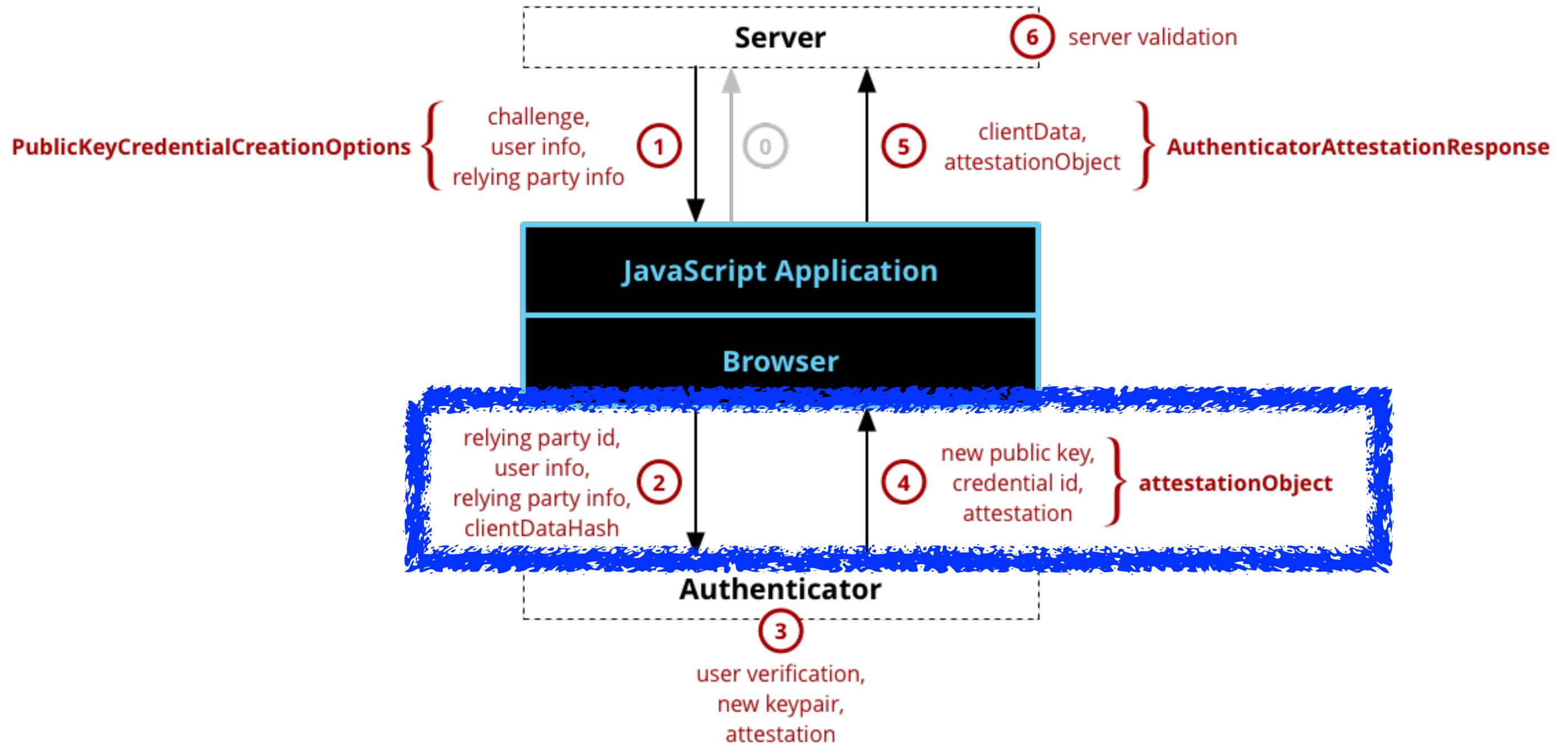
# Better Understanding the Communication

**Let's get wedged!**

# Interrupting the Flow

- Override `navigator.credentials.create()` and `navigator.credentials.get()`
  - Browser Extension
  - Custom WKWebView or other web window
- Once you have control
  - Change Attestation requirements
  - Change platform vs. cross-platform authenticator
  - Add attestation







`PublicKeyCredential.isUserVerifyingPlatformAuthenticatorAvailable()`

Lets the relaying party know that FIDO is available.

1

`PublicKeyCredential.isUserVerifyingPlatformAuthenticatorAvailable()`

Lets the relaying party know that FIDO is available.

2

`navigator.credentials.create()`

Takes a challenge and some other information and creates a new FIDO key.

1

`PublicKeyCredential.isUserVerifyingPlatformAuthenticatorAvailable()`

Lets the relaying party know that FIDO is available.

2

`navigator.credentials.create()`

Takes a challenge and some other information and creates a new FIDO key.

3

`navigator.credentials.get()`

Proves you have access to the private key associated with the account.

```
var real_create = navigator.credentials.create.bind(navigator.credentials);

navigator.credentials.create = function(options) {
  console.log("Credentials Create Logging Wrapper Engaged");
  var cleanedArgs = JSON.stringify(options, stringifyArrayCleaner);
  console.log(cleanedArgs);
  return new Promise(
    function (resolve, reject) {
      real_create(newOptions).then(value => {
        console.log("Credentials Create Response");
        console.log(value);
        resolve(value);
      }, reason => {
        console.log(reason);
        reject(reason);
      });
    });
};
}
```

```
var real_create = navigator.credentials.create.bind(navigator.credentials);

navigator.credentials.create = function(options) {
  console.log("Credentials Create Logging Wrapper Engaged");
  var cleanedArgs = JSON.stringify(options, stringifyArrayCleaner);
  console.log(cleanedArgs);
  return new Promise(
    function (resolve, reject) {
      real_create(newOptions).then(value => {
        console.log("Credentials Create Response");
        console.log(value);
        resolve(value);
      }, reason => {
        console.log(reason);
        reject(reason);
      });
    });
};
}
```

```
var real_create = navigator.credentials.create.bind(navigator.credentials);

navigator.credentials.create = function(options) {
  console.log("Credentials Create Logging Wrapper Engaged");
  var cleanedArgs = JSON.stringify(options, stringifyArrayCleaner);
  console.log(cleanedArgs);
  return new Promise(
    function (resolve, reject) {
      real_create(newOptions).then(value => {
        console.log("Credentials Create Response");
        console.log(value);
        resolve(value);
      }, reason => {
        console.log(reason);
        reject(reason);
      });
    });
});
}
```

Demo

**FIDO is more secure in every way than using a password**

**Keys can lie**

# Your Lying Keys

User Presence, User Verification and other aspects of assertions  
require trusting the key

Demo

# Your Lying Keys

User Presence, User Verification and other aspects of assertions  
require trusting the key

Only trust keys with attestation

# Your Lying Keys

User Presence, User Verification and other aspects of assertions  
require trusting the key

Only trust keys with attestation

Chrome and Safari Platform Authenticators don't do direct  
attestations\*

**Speaking of Attestation...**

```
attestationObject: {
  "fmt": "packed",
  "attStmt": {
    "alg": -7,
    "sig": <<array buffer>>,
    "x5c": [ <<array buffer>>]
  },
  "authData": {
    "rpIdHash": "f95bc73828ee21f9fd3bbe72d97908013b0a3759e9aea3dae318766cd2e1ad",
    "flags": {
      "userPresent": true,
      "reserved1": false,
      "userVerified": true,
      "reserved2": "0",
      "attestedCredentialData": true,
      "extensionDataIncluded": false
    },
    "signCount": 0,
    "attestedCredentialData": {
      "aaguid": "0000000000000000",
      "credentialIdLength": 20,
      "credentialId": "586dcf643e8d88fa8d33f77383b29885d88c0fa",
      "credentialPublicKey": {
        "kty": "EC",
        "alg": "ECDSA_w_SHA256",
        "crv": "P-256",
        "x": "Ec5m1WYnzTUGx7K8d03jYzpfXQJzA6EpqcYvxrmoQLg=",
        "y": "qxK7/ZErVjQ/gadyPGRHZlJx32Svaz60baxpFiGQ8B4="
      }
    }
  }
}
```

```
attestationObject: {
  "fmt": "packed",
  "attStmt": {
    "alg": -7,
    "sig": <<array buffer>>,
    "x5c": [ <<array buffer>>]
  },
  "authData": {
    "rpIdHash": "f95bc73828ee21f9fd3bbe72d97908013b0a3759e9aea3dae318766cd2e1ad",
    "flags": {
      "userPresent": true,
      "reserved1": false,
      "userVerified": true,
      "reserved2": "0",
      "attestedCredentialData": true,
      "extensionDataIncluded": false
    },
    "signCount": 0,
    "attestedCredentialData": {
      "aaguid": "0000000000000000",
      "credentialIdLength": 20,
      "credentialId": "586dcf643e8d88fa8d33f77383b29885d88c0fa",
      "credentialPublicKey": {
        "kty": "EC",
        "alg": "ECDSA_w_SHA256",
        "crv": "P-256",
        "x": "Ec5m1WYnzTUGx7K8d03jYzpfXQJzA6EpqcYvxrmoQLg=",
        "y": "qxK7/ZervjQ/gadyPGRHZlJx32Svaz60baxpFiGQ8B4="
      }
    }
  }
}
```

**<- Flags**

**<- Key ID**

**<- P256 key**

```
attestationObject: {
  "fmt": "packed",
  "attStmt": {
    "alg": -7,
    "sig": <<array buffer>>,
    "x5c": [ <<array buffer>>]
  },
  "authData": {
    "rpIdHash": "f95bc73828ee21f9fd3bbe72d97908013b0a3759e9aea3dae318766cd2e1ad",
    "flags": {
      "userPresent": true,
      "reserved1": false,
      "userVerified": true,
      "reserved2": "0",
      "attestedCredentialData": true,
      "extensionDataIncluded": false
    },
    "signCount": 0,
    "attestedCredentialData": {
      "aaguid": "0000000000000000",
      "credentialIdLength": 20,
      "credentialId": "586dcf643e8d88fa8d33f77383b29885d88c0fa",
      "credentialPublicKey": {
        "kty": "EC",
        "alg": "ECDSA_w_SHA256",
        "crv": "P-256",
        "x": "Ec5m1WYnzTUGx7K8d03jYzpfXQJzA6EpqcYvxrmoQLg=",
        "y": "qxK7/ZErVjQ/gadyPGRHZlJx32Svaz60baxpFiGQ8B4="
      }
    }
  }
}
```

## ← Attestation

Demo

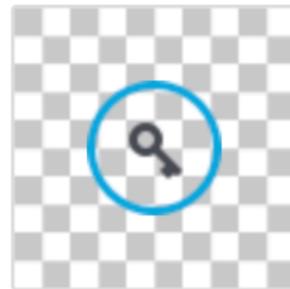
# Enterprise Attestation - Okta

✔ Okta Verify	
SMS Authentication	
Voice Call Authentication	
Google Authenticator	
✔ FIDO2 (WebAuthn)	<h2>FIDO2 (WebAuthn) <span>Active ▾</span></h2> <p>Once this factor is configured, additional verification will be required when users sign in to Okta.</p> <p>If the user selects 'Security key or biometric authenticator', they will be prompted to register an authenticator via Web Authentication in order to sign in to Okta successfully. Users can follow the on-screen prompts for browser or OS instructions in order to gain access. Learn more in <a href="#">documentation</a> .</p> <p>Web Authentication supports two authentication methods:</p> <ol style="list-style-type: none"><li>1. Security keys such as YubiKeys or Google Titan</li><li>2. Biometric authenticators such as Windows Hello or Apple Touch ID</li></ol>
YubiKey	
Duo Security	

# Enterprise Attestation - OneLogin

## Edit WebAuthn

Square Icon



 Square icon at least 96x96px as either a transparent .PNG or .SVG

User description

WebAuthn

Cancel

Delete

Save

# Enterprise Attestation - Azure

Save

Discard

## ENABLE

Yes

No

## USE FOR:

- Sign in
- Strong authentication

## TARGET

All users

Select users

Name	Type	Registration
All users	Group	Optional <span>⌵</span> <span>⋮</span>

## GENERAL

Allow self-service set up

Yes

No

Enforce attestation

Yes

No

## KEY RESTRICTION POLICY

Enforce key restrictions

Yes

No

Restrict specific keys

Allow

Block

# Enterprise Attestation - Azure

Save

Discard

## ENABLE

Yes

No

## USE FOR:

- Sign in
- Strong authentication

## TARGET

All users

Select users

Name	Type	Registration
All users	Group	Optional

## GENERAL

Allow self-service set up

Yes

No

Enforce attestation

Yes

No

## KEY RESTRICTION POLICY

Enforce key restrictions

Yes

No

Restrict specific keys

Allow

Block

**Always require attestation...?**



**If you're not so nice...**



## Block existing keys from working

Confuse user into thinking it's their fault. Can apply to platform and cross-platform.

1

Block existing keys from working

Confuse user into thinking it's their fault. Can apply to platform and cross-platform.

2

Have user register new key

Allow the key the user thinks they are using to go through the ceremony.

1

Block existing keys from working

Confuse user into thinking it's their fault. Can apply to platform and cross-platform.

2

Have user register new key

Allow the key the user thinks they are using to go through the ceremony.

3

Profit!

Exfiltrate key

**The moral of the story...**

**1. Don't blindly trust the browser.**

**2. Attestation can keep some of the riffraff out, but it will increase complexity.**

And...

**FIDO is more secure in every way than using a password**

**Intrigued?**



**Thanks!**