WebAuthn Reference Implementation

FIDO2: What, Why, How

Julian Stampfli

February 6, 2019

Project source

`https://github.com/Tartori/web-authn-demo`

Table of Contents

Table of Contents

Passwords

Drawbacks

- Hard to remember good passwords
- Weak passwords are chosen
- Passwords are reused

Passwords

Drawbacks

- Hard to remember good passwords
- Weak passwords are chosen
- Passwords are reused

Benefits

- EASY!!
- ...

Passwords

Drawbacks

- Hard to remember good passwords
- Weak passwords are chosen
- Passwords are reused

Benefits

- EASY!!
- ...
- People are used to them

FIDO2

- Standard for Passwordless Authentication
- Easy to use(?)

FIDO2

- Standard for Passwordless Authentication
- Easy to use(?)
- Authenticator (Token) instead of password

FIDO2 vs Passwords

Passwords

- Have it remembered
- Enter it
- Hope it is protected

FIDO2 vs Passwords

Passwords

- Have it remembered
- Enter it
- Hope it is protected

FIDO2

- Bring your Authenticator
- Plug it in
- Press a button
- Like an OTP

FIDO2 vs OTP

OTP

- Generated with shared secret
- Kind of like a password
- Secure against Replay attacks
- Weak against MitM
- Weak against verifier compromise

FIDO2 vs OTP

OTP

- Generated with shared secret
- Kind of like a password
- Secure against Replay attacks
- Weak against MitM
- Weak against verifier compromise

FIDO2

- Generated with public key crypto
- Save(ish) against MitM
- Save against verifier compromise
- Save against remote attacks

FIDO2 vs FIDO U2F

- FIDO U2F predecessor of FIDO2
- FIDO2 used for passwordless and second factor
- FIDO2 keeps the keys on Authenticator
- Interoperable

How does FIDO2 work

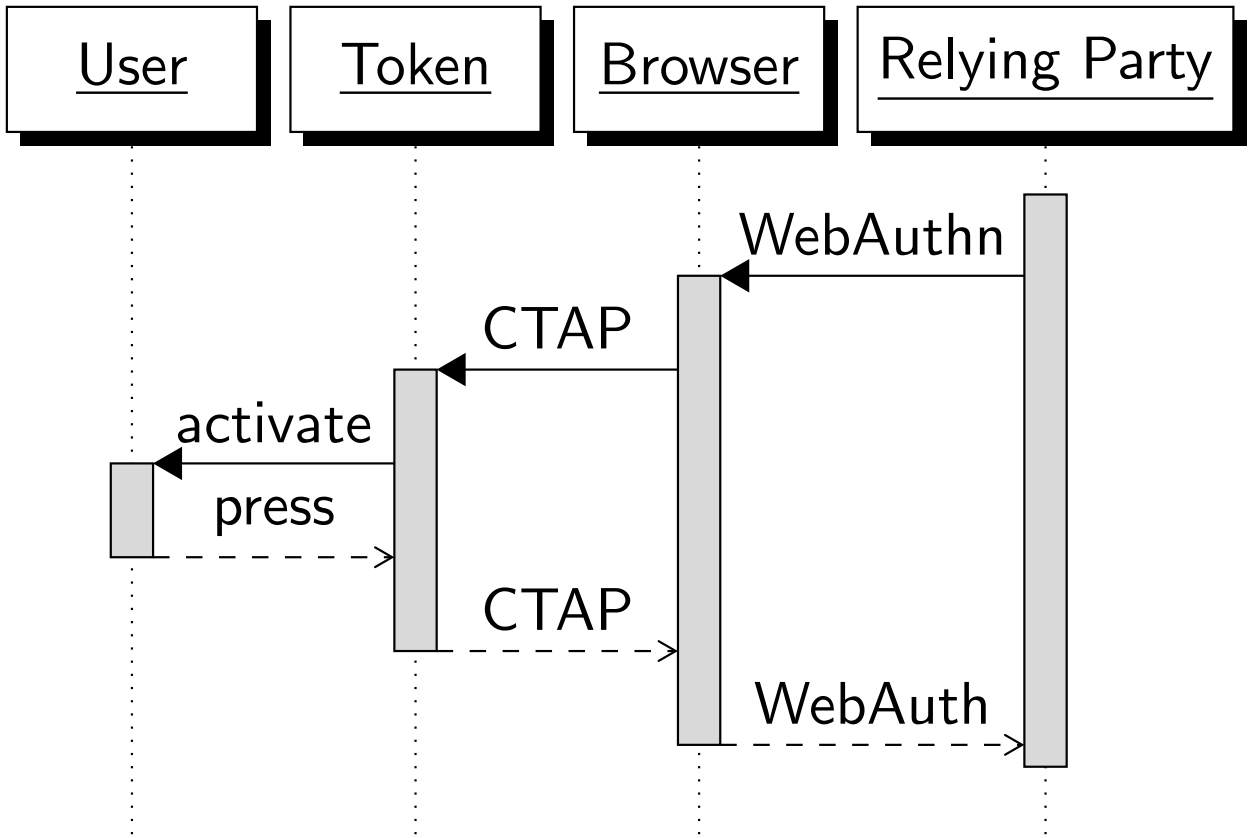
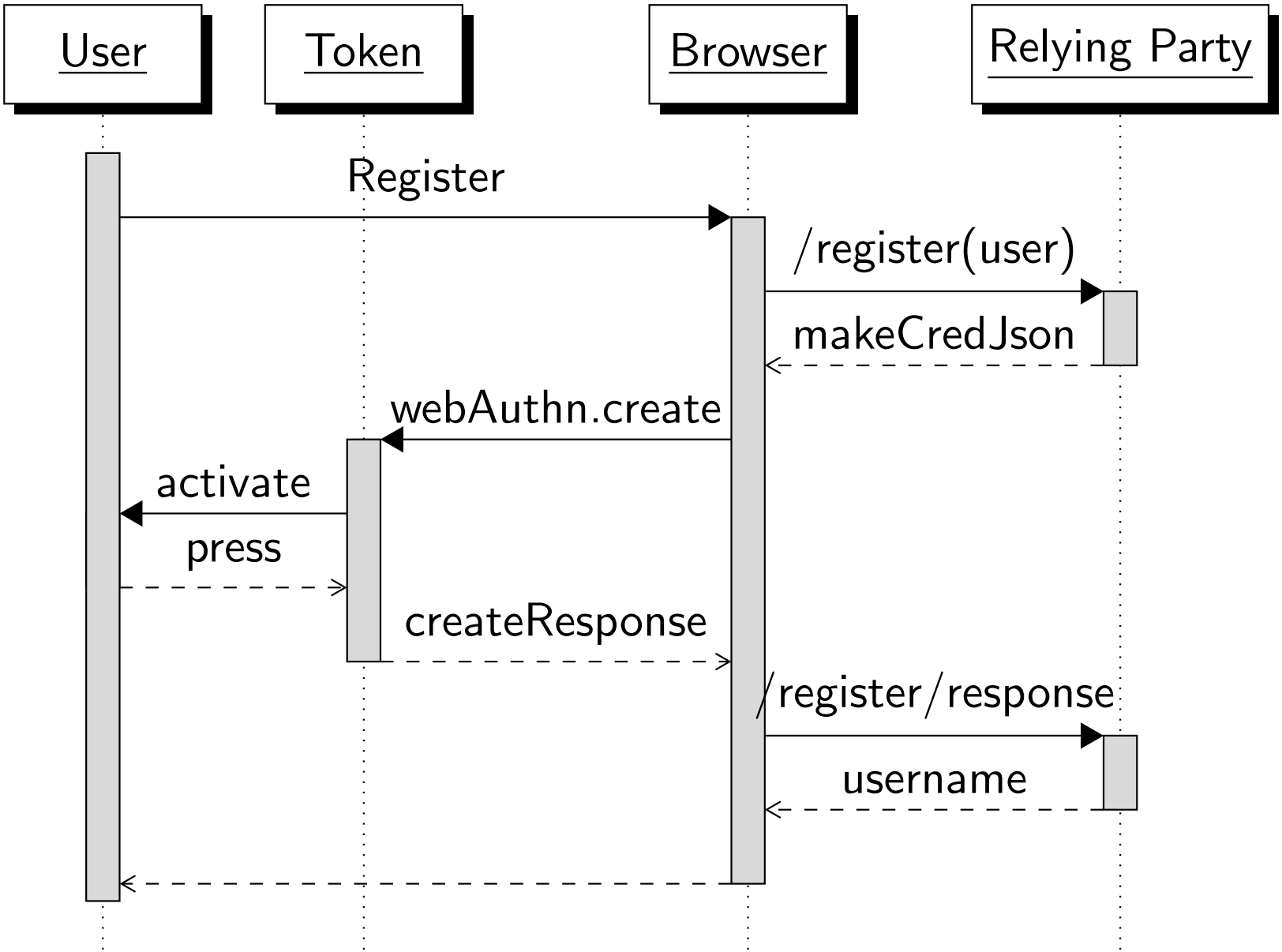


Table of Contents

Registration



Create Registration JSON I

```
{
  "public-key": {
    "challenge": "UvwcjUM5UOASx2AK9...",
    "fidoResponse": "direct",
    "rp": {
      "name": "BFH",
      "id": "dev.webauthn.demo"
    },
    "user": {
      "id": "lPVdjYV97NDuLLMg...",
      "name": "test",
      "displayName": "test"
    },
```

Create Registration JSON II

```
        "pubKeyCredParams": [
          {
            "type": "public-key",
            "alg": -7
          }
        ],
        "attestation": "direct",
        "timeout": 60000,
      }
    }
```

Registration Response

```
{
  "rawId": "qKk7sFYyBRex6K6twW5...",
  "response": {
    "attestationObject": "o2NmbXRm...",
    "clientDataJSON": "eyJjaGF..."
  },
  "getClientExtensionResults": {},
  "id": "qKk7sFYyBRex6K6twW5...",
  "type": "public-key"
}
```

Registration Response

- 19 Steps have to be completed
- 1-7 parse and validate clientDataJSON
- 8-13 parse and validate attestationObject
- 14 verify signature
- 15,16 verify trustworthiness
- 17,18 verify and update user data
- 19 - fail if any step failed

Client Data Parsed

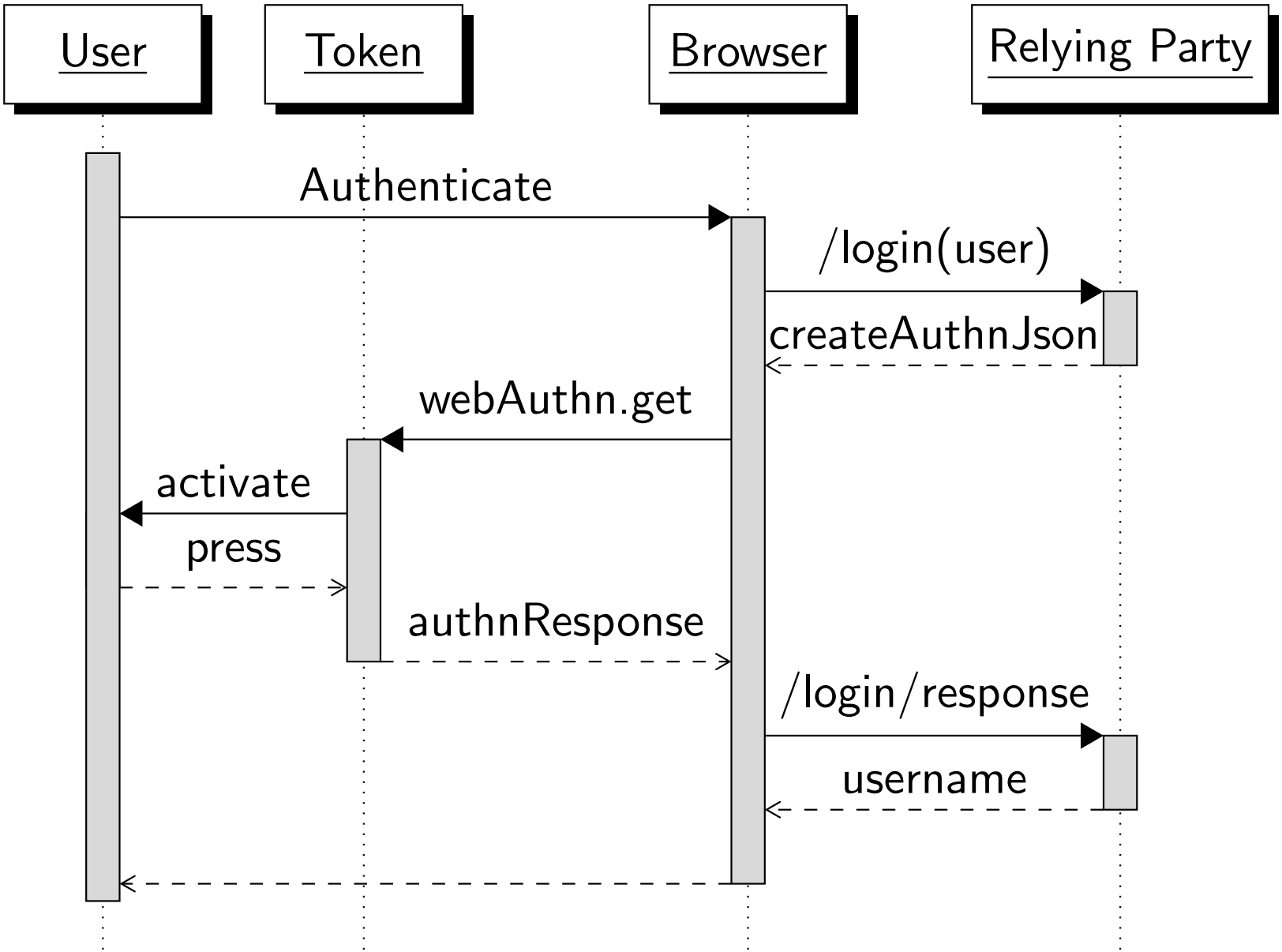
```
1  {
2    "challenge": "UvwcjUM5UOASx2AK9...",
3    "new_keys_may_be_added_here": "do not
       compare clientDataJSON ",
4    "origin": "https://dev.webauthn.demo:8
       888",
5    "type": "webauthn.create"
6  }
```

Attestation Data Parsed

```
{
  "fmt": "packed",
  "attStmt": {
    "alg": -7,
    "sig": "MEQCIBRsm+gm5tY75S/uEk...",
    "x5c": [
      "MIICvDCCAaSgAwIBAgIEA63wEjA..."
    ]
  },
  "authData": "kn8Lq9EV0MBhHa/k+ZE..."
}
```

Table of Contents

Authentication



Create Authentication JSON

```
{
  "public-key": {
    "challenge": "rfmx563tPscMU...",
    "rpId": "dev.webauthn.demo",
    "allowCredentials": [
      {
        "type": "public-key",
        "id": "NWIv8Cd3lXkE8r4_..."
      }
    ],
    "timeout": 60000,
  }
}
```

Authentication Response

```
{
  "rawId": "NWIv8Cd3lXkE8r4_Ondf6f...",
  "response": {
    "authenticatorData": "kn8Lq9EV",
    "signature": "MEUCIAhdPPqzV69p...",
    "userHandle": "",
    "clientDataJSON": "eyJjaGFsbGV..."
  },
  "getClientExtensionResults": {},
  "id": "NWIv8Cd3lXkE8r4_Ondf6fPDe...",
  "type": "public-key"
}
```

Authentication Response Steps

- 18 Steps have to be completed
- 1-3 validate user and authenticator
- 4 parse several fields
- 5-10,15 parse and validate clientDataJSON
- 11-14 parse and validate authenticatorData
- 16,17 verify signature and sign count
- 18 - fail if any step failed

Client Data Parsed

```
1   {
2     "challenge": "pT5Z7r07quoiDUbLxB...",
3     "origin": "https://dev.webauthn.demo:8
        888",
4     "type": "webauthn.get"
5   }
```

Table of Contents

Authenticator Data

- RP ID hash
- Flags
- Sign Count
- Attested Credential Data (Registration)
- Extensions (Optional)

Table of Contents

Conclusion

Summary

- Passwordless
- Registration and authentication very similar
- Rather complex
- Need for libraries

Conclusion

Summary

- Passwordless
- Registration and authentication very similar
- Rather complex
- Need for libraries

Conclusion

- Wide adoption?
- Used in eID?
- Used internally?
- Complexity?
- User verification?

Further Research

- FIDO2 over NFC
- Other authenticators
- Managing authenticators

Table of Contents

Demo

`https://dev.webauthn.demo:8888`

Questions

Questions

`https://github.com/Tartori/web-authn-demo`