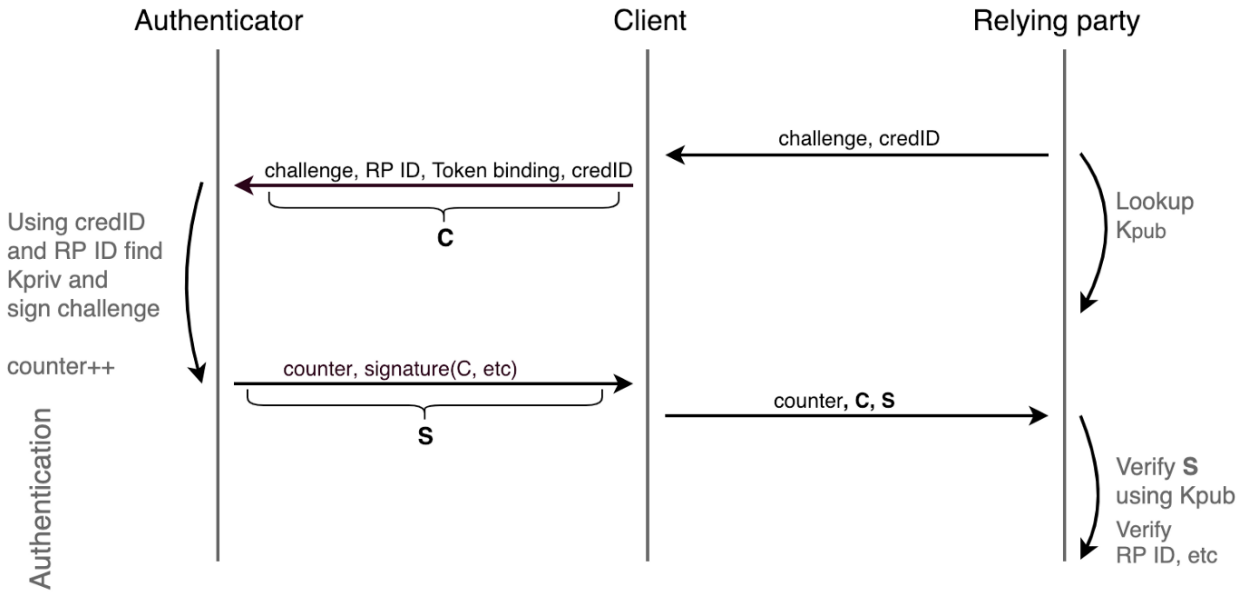
What is FIDO?

FIDO or FIDO Alliance is a consortium that develops secure, open, standard, phishing proof authentication protocols. FIDO Alliance was created in 2013, and now it has over 250 members across the globe. FIDO has three protocols: UAF, U2F and FIDO2. They are the same family of the protocols, since they are all origin based, challenge-response, phishing proof, digital signature authentication protocols.

How does FIDO works?

As I mentioned before, FIDO protocols origin based, challenge-response, phishing proof, digital signature authentication protocols. Here is a diagram that explain the process:



The relying party(the server) send a challenge and credential identifier of the previously registered credential to the client(the browser). The client then attaches relying party information, such as origin of the call, and sends it to the authenticator.

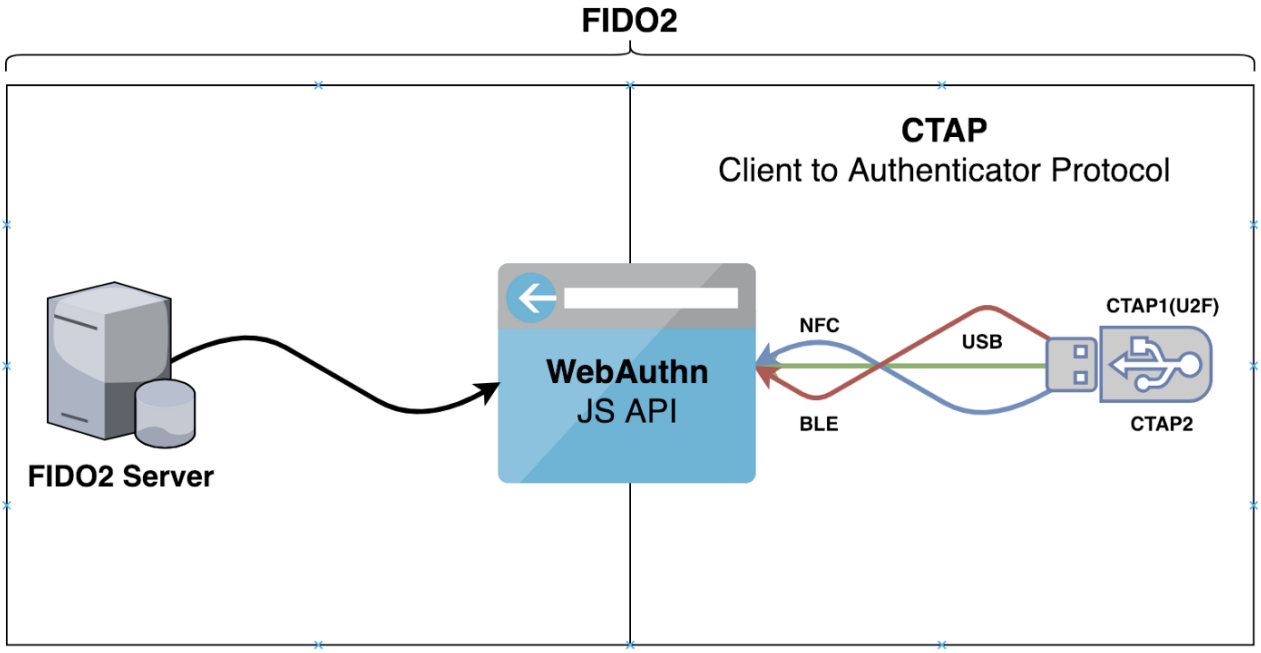
Authenticator first either checks if user is present, by requesting user to press the button, or doing full user verification using on-device pin, biometrics, etc.

When verification is done, the device then signs the payload, using the private key, identified by the credential id, and returns the assertion to the client. Client then attaches information that it has given to the authenticator, as part of the signature payload, and forwards it to the relying party.

The relying party checks that information, and ensures that RP information contains expected origin, and challenge, after which it checks the signature. If any of those steps are failed, we can detect that there was a phishing attack, thus preventing it.

FIDO2 or WebAuthn?

So there is general confusion about the terms. FIDO, FIDO2, WebAuthn, CTAP1, CTAP2... What does this all even mean?



Definitions

- **HASH** - One way encoding (DES/MD5/SHA1/SHA2)
- **PGP** - Pretty Good Privacy - Key Pair(Public Key, Private Key) used for exchanging data by signing data (**encrypting**), unpacking data (**descrypting**) with private and public key or using hash
- **Public Key** - (Certificate) Not secret (**hash**) that is shared with the public that is used to sign data that can be read only by the private key
- **Private Key** - (Certificate) Secret (**hash**) that is never seen by the public that is used to sign data that can be read only by the public key
- **UAF** - Universal Authentication Framework
- **U2F** - Universal Second Factor
- **FIDO2** - Challenge-response, phishing proof, digital signature authentication protocols
- **TLV** - Type - Length - Value
- **RAW** - Raw (TCP/IP) is an insecure communication protocol
- **CBOR** - The Concise Binary Object Representation (CBOR) is a data format whose design goals include the possibility of extremely small code size, fairly small message size, and extensibility without the need for version negotiation
- **FIDO** —**F**ast **ID**entity **O**nline, or FIDO Alliance. As I explained earlier it's a consortium that develops secure, open, phishing proof, passwordless authentication standards. FIDO

Protocol Family is a set of protocol that was developed by FIDO Alliance. UAF — Universal Authentication Framework. U2F — Universal Second Factor, and FIDO2. When I say use **"FIDO"** I generally mean "Use any of the three protocols", as they are all conceptually the same protocol, with the difference being structural (UAF — TLV, U2F — RAW, FIDO2 — CBOR).

- **FIDO2 —** A project name for a new, modern, simple, secure, phishing proof, passwordless authentication protocol. Its core specifications are **WebAuthn**(the client API, the Web Authentication API) and **CTAP(**the authenticator API). Is an improvement over the **U2F** standard, mainly with the ability to now perform **password-less logins** [1][2]. This had to do with a shortcoming in the U2F protocol and/or devices such that they didn't need to have much storage on these devices [3]. To address this, the new **FIDO2 devices are now required to persist your username(s)** for a particular site. The new CTAP2 protocol has also been extended to accommodate more sophisticated authenticators, like those crypto-currency wallets with a display.
- **BLE** - Bluetooth (4.0) Low Energy (**A.K.A. Bluetooth LE, Bluetooth Smart**[)
- **NFC** - Near Field Communication
- **USB** - Universal Serial Bus
- **CTAP** — Client to Authenticator Protocols — A set of low level protocols to communicate with the authenticators over the BLE/NFC/USB. CTAP family includes **CTAP1** and **CTAP2** protocols.
- **CTAP2** - Client to Authenticator Protocol
- **MFA** - Multi Factor Authentication - is a method of granting access to a system only after two or more pieces of evidence have been presented. These pieces of evidence, also known as factors, prove your identity. This could be something you know (secret password or PIN), something you have (mobile phone or security key), or something you are (biometrics: fingerprint or face). You've probably already used MFA in some form if you have withdrawn money from an ATM.
- **2FA** - Two Factor Authentication - is based on a pair of information - What you have and what you know.
- **Client** - Web Browser / Operating System
- **Authenticator** - Hardware based (key) token that proves to the client that you really are who you say you are
- **Relying Party** - The Service that wants to authenticate the user
- **Ceremony** - Set of defined actions used in protocols

FIDO2 is built on top of two open standards: the Web Authentication API (WebAuthn) and the Client to Authenticator Protocol (CTAP2). The two work together and are required to achieve a strong authentication experience. The earlier FIDO U2F protocol working with external authenticators is now renamed CTAP1 in the FIDO specifications. FIDO2 and WebAuthn are backwards compatible with U2F authenticators.

WebAuthn/FIDO2 Credentials

Nested within this FIDO2 are

FIDO2 Server (**F**AST **ID**ENTITY **O**NLINE)
WebAuthn (JS API) - JavaScript (Browser Plugin)
CTAP (Client to Authenticator Protocol - The Authenticator API) (JS Speaking to Authenticator)

FIDO2 Server

Importance of FIDO2/WebAuthn credentials

FIDO2/WebAuthn credentials provide strong authentication that is resistant to phishing, replay, and server breach attacks. Strong authentication is defined as having at least one cryptographically backed factor. FIDO2 provides strong, attested, scoped public key-based credentials for authenticating users.

FIDO2/WebAuthn authentication is decentralized. Users are authenticated locally on an authenticator. The online service validates the authentication ceremony was properly conducted before allowing user access. The online service only needs publicly available information to validate the local authentication. This reduces the value of attacking the server because it has no user authentication secrets and user credentials are also not transferable between services.

Attestation becomes important now that authentication is decentralized. The online service must trust that the authenticator reliably conducts FIDO2 ceremonies. FIDO2 provides a mechanism for authenticators to present an attestation statement so that an online service can verify it is a genuine device.

FIDO2 credentials can provide MFA by employing user **verification authorization gestures**. This could be a PIN or biometric, for example. This user verification stays on the authenticator, it is never sent over the network like a password. If MFA is not required, then a simple test of user presence through an authorization gesture, e.g. tap and go, can show that the user gave consent for the credential to be used.

Public key-based credentials

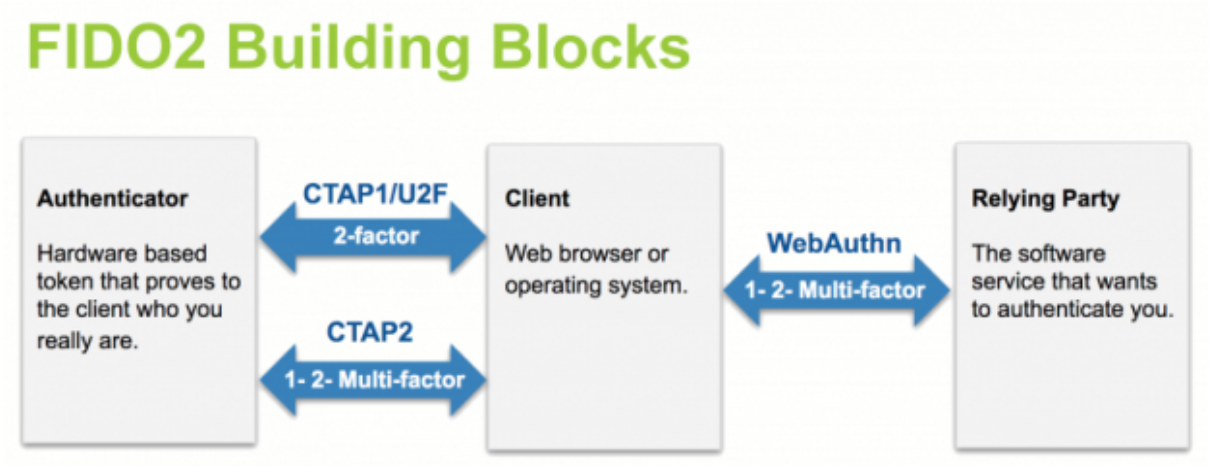
In general, a credential is data that proves a person's identity or qualification. This credential is used to authenticate to an online service, also known as a Relying Party (RP). FIDO2 uses public-key cryptography to authenticate users. An RP-specific credential key pair, i.e., a private key and a public key, is generated on the authenticator. The public key is sent to the RP at registration time. The private key never leaves the authenticator. When the user makes the request to login, the authenticator sends an assertion that proves the user possesses the private key. The RP uses the public key to validate the assertion before allowing the user to login.

FIDO2 introduces the concept of a resident key credential, which is/are stored on the authenticator instead of encrypted and stored on the server. Resident keys also enable passwordless authentication scenarios, at the expense of consuming limited storage space on the authenticator.

Where FIDO2/WebAuthn credentials should be used

FIDO2/WebAuthn credentials with MFA should be used whenever possible, especially when it comes to sensitive data, like your primary email, financial accounts, and health records.

How is FIDO2 implemented? FIDO2 is based on open standards



FIDO2 is built on top of two open standards: the Web Authentication API (WebAuthn) and the Client to Authenticator Protocol (CTAP2). The two work together and are required to achieve a strong authentication experience. The earlier FIDO U2F protocol working with external authenticators is now renamed CTAP1 in the FIDO specifications. FIDO2 and WebAuthn are backwards compatible with U2F authenticators.

How it works

FIDO2/WebAuthn authentication consists of a **registration** ceremony and an **authentication** ceremony.

Registration Ceremony

Assuming a user has already created an **account with the online service**, the user **registers** one or more **authenticators** against that identity. The authenticator could be a mobile phone, laptop, or YubiKey. Any authenticator that is compliant with CTAP1 or CTAP2 will work.

During the registration ceremony the authenticator creates a FIDO2 credential by generating a private key and public key. The private key stays in the hardware's secure element while the public key is given to the online service. Possession of this authenticator in combination with a gesture to unlock the use of the credential is now proof of your identity. Unlike a password which is sent over the network and stored on a remote server, neither the private key nor the gesture ever leave the authenticator. They are not sent over the network nor stored on a remote server. Only the public key is stored by the online service.

Authentication ceremony

Authentication happens locally on the authenticator and is verified by the online service. First, a user will make a request to login. The online service will send the **client** a challenge, which the client wraps in its *client data*. Next, the client requests cryptographic proof of user authentication. With the user's consent, the authenticator will return an assertion. This assertion provides evidence that the authenticator is in possession of the private key by signing the client data with the private key. The server then verifies that the client data contains the correct challenge and uses the public key to verify the signature.

The work of authenicating the user happens locally on the authenticator (**hardware key**) and is **verified by the service(online service provider)** . First, a user will make a request to login. The online service will send the **client** a challenge, which the client wraps in its *client data*. Next, the client requests cryptographic proof of user authentication. With the user's consent, the authenticator will return an assertion. This assertion provides evidence that the authenticator is in possession of the private key by signing the client data with the private key. The server then verifies that the client data contains the correct challenge and uses the public key to verify the signature.

1. User requests to login
2. The Service Provider (Relaying Party) send the client (Browser/Operation System) a challenge, wrapping it in the client data.
3. The client requests that the hardware key take the client data and authenticate against it (**cryptographic proof of user authentication**).
4. The hardware key then prompts the client to request that the user give concent for the authentiation.
5. Authenticator (hardware key) return an assertion to the client. This assertion is the evidence that the authenticator is in possession of the user's priviate key. This assertion is done by signing the client data with the user's private key.
6. The client then sends this signed data to the Service Provider.
7. The Service Provider verifies that the client data, signed by the private key, matches up with the private keys public key pair.
8. The Service Provider then returns a successful message along with a payload that includes a session token that will be used through out the communication with the Service Provider and Client.