WebAuthn Example Application

WebAuthn is a web standard that enables secure, passwordless authentication using public-key cryptography. It replaces traditional passwords with cryptographic credentials stored in authenticators like hardware keys, biometrics, or platform security modules.

WebAuthn terminology

- Relying Party: The server or web application that initiates registration/authentication
- Client Platform: The browser or operating system that mediates between RP and authenticator
- Authenticator: The device or module that generates and stores credentials (e.g. TPM, USB key)
- User verification method (UVM): This refers to a biometric (eg: finger print), a PIN or password, a Touch, gesture or presence check.

It is important to note the following:

WebAuthn credentials are **bound to the authenticator** (e.g. your phone, security key, or platform authenticator like Windows Hello), not to your geographic location. So:

- You can access the same website from **any location**, as long as you have access to the authenticator that holds your credential.
- The **RP ID** (relying party identifier) ensures the credential is scoped to the domain (e.g. example.com), not the IP address or physical location.

By default, credentials are **not portable** across devices. If you register on your laptop in London, that credential won't be available on your phone in New York unless:

- You use a **cross-device authenticator**, like a roaming security key (YubiKey, etc.).
- Or you use a **synchronized platform authenticator**, such as:
  - **Passkeys** on iCloud Keychain (Apple)
  - **Google Password Manager** passkeys
  - **Windows Hello** with cloud sync (if supported)

These allow the same credential to be used across multiple devices, even in different locations.

The above is an annoyance when you have 2 or more browsers accessing the same website using the same user id on the same device and not using a cross-device authenticator or a synchronised platform authenticator.

To overcome this problem, in this example, the devices authenticators GUID is used to qualify the website and the user id. The GUID is stored on the RP with the Credential ID/Public Key associated with the website (rpHost) and the user ID.

WebAuthn has many features and can be coded in many different ways to ensure security and phishing-resistant way to authenticate users. For example, authentication can be done on the client side (browser) or on the Relying Party. It is however recommended to authenticate on the RP which this example does.

This example uses raw javascript and PHP code. No libraries eg. no cbor decoder.

When you register a User verification method; you will sometimes be asked to re–submit it: this is an annoyance but not a problem. This can be caused by the authenticator wanting confirmation.

WebAuthn Example Application

Flow

Client Platform (*PC, smartphone etc*)	Relying Party(RP: Server)
User enters an ID *(email address in this example)*	
The client request a Challenge and Credential ID using User ID	
	• Creates a 32 byte Challenge. • Store the Challenge in a webauthn depository *(eg MYSQL table)* • Send the Challenge and Credential ID *(if it exists)* back to the client
If no Credential ID → *(1st time registration)*	
• Using the Challenge ask the Authenticator to generate a private / public key and a Credential ID • Send the User ID, Challenge, public key and credential ID to the RP. • Note: the credential ID is like a look up to the private key	
	• Check that the Challenge is the same as the one previously sent to the client. • Store the Public Key and Credential ID in the depository and send back website successfully registered
If the Challenge is rejected – abort login	
Otherwise ok for website to continue	
If Credential ID received ->	
• if credential ID does not exist in the Authenticator: abort login • if credential ID does exist get the following values from the Authenticator and send them to the RP: User ID, Challenge, Signature, ClientDataJSON, AuthenticatorData	
	• Check that the Challenge is the same as the one previously sent to the client. • Get the Public Key stored in the depository. • Using openssl: validate the Public Key against those values received from the client. (eg Signature, etc) • respond back to the client accordingly.
If a negative response from the RP – abort login	
If positive then its OK for the website to continue	

WebAuthn Example Application

MySQL Table Structure

```
CREATE TABLE `credentials` (
  `id` int NOT NULL,
  `user_id` varchar(255) NOT NULL,
  `rphost` varchar(253) NOT NULL,
  `aaguid` char(32)  DEFAULT 'unknown',
  `challenge` varbinary(32) NOT NULL,
  `credential_id` varbinary(1023) NOT NULL,
  `signaturecount` bigint UNSIGNED NOT NULL DEFAULT '0',
  `public_key` text NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
)

ALTER TABLE `credentials`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `credentials`
  MODIFY `id` int NOT NULL AUTO_INCREMENT;
```