FIDO Authentication

# Introduction

FIDO is a **Fast identity Online Universal 2nd Factor (U2F)** authentication. It is a set of standards for fast, simple and strong authentication. FIDO2 is made up of the **World Wide Web Consortium’s (W3C) Web Authentication specification(WebAuthn)** and FIDO’s corresponding **Client-to-Authenticator Protocol (CTAP)**, which collectively enable common FIDO2 compliant devices to authenticate easily to online services in both mobile and desktop environments. FIDO Alliance, an industry association with representatives from a range of organizations including Google, Amazon, Facebook, Microsoft, Intel, Mozilla and Yubico, etc., develops these standards. The standards enable phishing-resistant, passwordless, and multi-factor authentication.

# How does FIDO authentication work?

The FIDO2 / WebAuthn allows user to create and use strong, attested public key based credentials for authenticating users. The API supports the use of BLE (Bluetooth Low Energy), NFC (Near Field Communication) and USB (Universal Serial Bus) roaming authenticators (security keys) as well as a platform authenticator, which allows the user to authenticate using their biometrics such as fingerprint, facial recognition, or security PIN code or screenlock.

FIDO authentication process consists of following components:

1. Relying party – It is the user’s service, composed of a back-end server and a front-end application.
2. Application – User application, which uses client-side APIs like WebAuthn and FIDO2 for Android to create and verify user credentials with the authenticator.
3. Server – The server implements WebAuthn specification required to implement FIDO authentication and stores user’s public key and an identifier for the credential.
4. Authenticator – A FIDO authenticator generates user credentials, a set of public key and private key. It can be part of user’s device, or an external piece of hardware or software. The authenticator is used in two basic interactions – **registration** and **authentication.**

In a FIDO authentication or registration flow, a **relying party** uses APIs to interact with a user’s **authenticator**.

## Registration

* In a registration scenario, when a user is signing up for an account on a website or mobile app, the FIDO authenticator generates user credentials.
* A user credential has both a public and private key component.
* The public key and an identifier for the credential are shared with the server, while the private key is kept secret by the authenticator.

## Authentication

* In an authentication scenario, when a user returns to the service on a new device, or after their session expires, the authenticator much provide proof of the user’s private key. It does this by responding to a cryptographic challenge issued by the server.
* To verify the identity of the user, some types of authenticator use platform authenticators such as fingerprints, facial recognition or security PIN. These are called software authenticators. Some types use hardware authenticators like YubiKey from Yubico, or white-labeled Feitian keys, or Titan security, etc.

# Main use cases

FIDO technology aims to solve three separate use cases for relying parties (or Internet services) by helping to:

1. Prevent phishing during initial login in to a service on a new device;
2. Reverify a user’s identity to a service on a device on which they’ve already logged in to;
3. Confirm that the device a user is connecting from is still the original device where they logged in from previously.

Security-savvy professionals may interpret the third use case as a special instance of use case #2. However, there are some differences, which we break down a bit further below:

* In case #2, the problem that FIDO technology tries to solve is re-verifying a user’s identity by unlocking a private key stored on the device.
* In case #3, FIDO technology helps to determine whether a previously created key is still available on the device without any proof of who the user is.

# FIDO2 API for Android

The **FIDO2 API** allows Android applications to create and use strong, attested public-key based credentials for authenticating users. This API provides **WebAuthn Client** implementation, which supports the use of BLE (Bluetooth Low Energy), NFC (Near Field Communication) and USB (Universal Serial Bus) roaming authenticators (security keys) as well as a platform authenticator, which allows the user to authenticate using their biometrics such as fingerprint, facial recognition, or security PIN code or screenlock.

## Pre-requisites

* Android SDK 26
* Android Build Tools v25.0.3
* Supported on all the devices running Android 7+ i.e. API level 24+

## Integration

* Need to implement WebAuthn specification at server end and host a FIDO server in order to store user’s public key and an identifier for the credential.
* Client application needs to add the FIDO2 dependency i.e. ‘com.google.android.gms:play-services-fido:18.1.0’ into their build.gradle file.
* The FIDO2 API entry point it the FIDO2ApiClient. The API supports two operations:
* Registration is done once per authenticator per account/userid, when the user associates an authenticator with an account.
* Signing is done whenever the relying party wants to authenticate a user.
* Both registration and signing require user interaction.

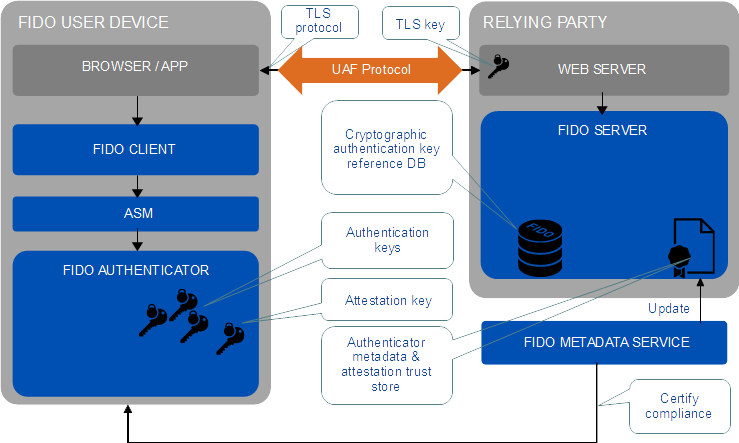
# FIDO Architecture

### Universal Authentication Framework (UAF) Protocol

The UAF protocol allows online services to offer password-less and multi-factor security. The user registers their device to the online service by selecting a local authentication mechanism such as swiping a finger, looking at the camera, etc. Once registered, the user simply repeats the local authentication action whenever they need to authenticate to the service. The user no longer needs to enter their password while authentication. It also allows combining multiple authentication mechanisms such as fingerprint + PIN.

### Universal 2nd Factor (U2F) Protocol

The U2F protocol allows online services to augment the security of their existing password infrastructure by adding a strong second factor to user login. The user logs in with a username and password as before. The service can also prompt the user to present a second factor device at any time it chooses. During registration and authentication, the user presents the second factor by simply pressing a button on a USB device or tapping over NFC.



ASM- Authenticator Specific Module.

TLS- Transport Layer Security

The FIDO-specific components of the reference architecture are described below.

* FIDO Client: It implements the client side of the FIDO protocols, and is responsible for:
* Interacting with specific FIDO Authenticators using FIDO Authenticator Abstraction layer via FIDO Authenticator API.
* Interacting with user agent (e.g. mobile app, browser) using user agent-specific interfaces to communicate with the FIDO Server. The user agent is then responsible for communicating FIDO messages to a FIDO Server at a Relying Party.

The FIDO architecture ensures that FIDO client software can be implemented across a range of system types, operating systems, and Web Browsers. While FIDO client software is typically platform-specific, the interactions between the components should ensure a consistent user experience from platform to platform.

* FIDO Server: It implements the server side of the FIDO protocols, and is responsible for:
* Interacting with the Relying Party web server to communicate FIDO protocol messages to a FIDO Client via a device user agent.
* Validating FIDO authenticator attestations against the configured authenticator metadata to ensure only trusted authenticators are registered for use.
* Manage the association of registered FIDO Authenticators to user accounts at the Relying Party.
* Evaluating user authentication and transaction confirmation responses to determine their validity.

The FIDO server is conceived as being deployable as an on-premise server by Relying Parties or as being outsourced to a FIDO-enabled third party service provider.

* FIDO Protocols: They carry FIDO messages between user devices and Relying Parties. There are protocol messages addressing:
* Authenticator Registration: It enables Relying Parties to:
* Discover the FIDO Authenticators available on a user’s system or device. It will convey FIDO Authenticator attributes to the Relying Party thus enabling policy decisions and enforcement to take place.
* Verify attestation assertions made by the FIDO Authenticators to ensure the authenticator is authentic and trusted. Verification occurs using the attestation public key certificates distributed via authenticator metadata.
* Register the authenticator and associate it with the user’s account at the Relying Party. Once an authenticator attestation has been validate, the Relying Party can provide a unique secure identifier that is specific to the Relying Party and the FIDO Authenticator. This identifier can be used in future interactions between the pair {RP, Authenticator} and is not known to any other devices.
* User Authentication: Authentication is typically based on cryptographic challenge-response authentication protocols and will facilitate user choice regarding which FIDO Authenticators are employed in an authentication event.
* Authenticator Deregistration: It is typically required when user account is removed at the Relying Party. The Relying Party can trigger deregistration by requesting the Authenticator to delete the associated credential with the user account.
* FIDO Authenticator Abstraction Layer: It provides a uniform API to FIDO Clients enabling the use of authenticator-based cryptographic services for FIDO-supported operations. It provides a uniform lower-layer “authenticator plugin” API facilitating the deployment of multi-vendor FIDO Authenticators and their requisite drivers.
* FIDO Authenticator: It is a secure entity, connected to or housed within FIDO user devices, which can create key material associated to a Relying Party. The key can then be used to participate in FIDO strong authentication protocols. For e.g., the FISO Authenticator can provide a response to a cryptographic challenge using the key material thus authenticating itself to the Relying Party.
* FIDO Authenticator Metadata Validation: In the FIDO context, attestation is how Authenticators make claims to a Relying Party during registration that the keys they generate, and/or certain measurements they report, originate from genuine devices with the certified characteristics. The FIDO Server validates an attestation signature carried in a FIDO registration protocol message. FIDO Authenticators are created with attestation private keys used to create the signature and the FIDO Server validates the signature using that authenticator’s attestation public key certificate located in the authenticator metadata.

## WebAuthn

The **Web Authentication API** (WebAuthn) is a specification written by the **W3C** and **FIDO**, with the participation of organizations like Google, Microsoft, Mozilla, and others. The API allows servers to register and authenticate users using public key cryptography instead of a password.

Features:

* It allows servers to integrate with the strong authentication now built into devices. Instead of password, a private-public keypair (known as **credentials)** is created for a website. The private key is stored securely on the user’s device; a public key and randomly generated credential ID is sent to the server for storage. The server can then use that public key to prove user’s identity.
* The public key is not secret, because it is effectively useless without the corresponding private key. The fact that the server receives no secret has far-reaching implications for the security of users and organizations. Databases are no longer as attractive to hackers, because the public key aren’t useful to them.

Web Authentication relies on three major properties:

* **Strong**: Authentication is ideally backed by a Hardware Security Module, which can safely store private keys and perform the cryptographic operations needed for WebAuthn.
* **Scoped**: A keypair is only useful for a specific origin, like browser cookies. A keypair registered at ‘webauthn.guide’ cannot be used at ‘evil-webauthn.guide’, mitigating the threat of phishing.
* **Attested**: Authenticators can provide a certificate that helps servers verify that the public key did in fact come from an authenticator they trust, and not a fraudulent source.

# Considerations while adopting FIDO Technology

### Security

FIDO technology enables passwordless authentication between servers, browsers, and authenticators using a private-public keypair. 81% of all hacking-related breaches leverage stolen or weak passwords. It obviously removes this possibility by leveraging the use of hardware security module. Hence, it is safe to say it is more secure than password based authentication. All the security measures to avoid phishing attack, replay attack and stolen password possibilities are handled in WebAuthn specification and FIDO protocols.

### User Experience

Organizations like Facebook and Google already enabled the FIDO2 authentication function long ago. Since it removes the re-authentication using password and saves the effort to create and memorize long, unique passwords, user experience should be good. The FIDO – Fast Identity Online as the name itself suggests it to be faster, secure, and reliable way of authentication.

### Technology Adoption Challenges

To adopt FIDO2 technology in existing applications, changes need to be incorporated in both client app and server-side implementations. An extra cost overhead will be involved in setting up FIDO server at Relying Party. While adopting FIDO2 changes in client side application, various authentication scenarios need to be properly addressed and eventually implemented. Thorough testing of the application is required after adopting this technology at both client and server end.

### Is Fragmentation a challenge?

In December 2019, with the release of iOS 13.3, Apple has added support for NFC, USB, and Lightening FIDO2-compliant security keys in Safari. It works well with hardware authenticators like Yubico YubiKey 5Ci. Hence both iOS and Android support FIDO2 technology. Google also enabled a built-in FIDO authenticator backed by hardware-based Titan security in Chromebooks.

FIDO is implemented at Android Application Framework layer and it’s part of Google Play Services package i.e. ‘com.google.android.gms.fido’. Every Android device manufacturer vendor should either support or re-implement it as per their device hardware specifications.

Samsung already supports FIDO technology for their devices.

## References

<https://developers.google.com/identity/fido>

<https://developers.google.com/identity/fido/android/native-apps>

<https://webauthn.guide/>

<https://fidoalliance.org/>

<https://www.computerweekly.com/news/252458255/Android-now-Fido2-certified>

<https://searchmobilecomputing.techtarget.com/opinion/What-IT-needs-to-know-about-FIDO2-iOS-and-Android-standards>

<https://www.forbes.com/sites/kateoflahertyuk/2019/12/10/apple-ios-133-drops-with-killer-new-security-feature-heres-how-to-use-it/#7b6af11b125a>

<https://security.googleblog.com/2019/11/using-built-in-fido-authenticator-on.html>