Exhibit D

GOOGLE EXHIBIT 1018

U.S. Patent Number US 8,646,042– Preliminary Infringement Contentions[1]

Assignee:	Proxense, LLC
Title:	Hybrid device having a personal digital key and receiver-decoder circuit and methods of use
Filing Date:	2012-04-12
Publication Date:	2014-02-04
Inventor:	Brown, David L.

042 Patent Claim[2]		**Accused Instrumentality And Where Each Claim Element Is Found[3]**
10	A method comprising[4]:	This preamble is not limiting. Google's, Google Cloud, as well as its subscribers (which Google directs and controls with online instructions and direct on-screen prompts to action to receive the benefits of not having to maintain an authentication server and/or permit the use of a Google Account to sign on users) perform the claimed method, literally or by the doctrine of equivalents, for at least the reasons set forth below. Non-limiting examples include the use of Google Identity Passkeys provided by Google and stored on Android devices as directed by Android OS, which is provided and controlled by Google for purposes of accessing services,

[1] The Preliminary Infringement Contentions (PICS) provided herein are based on information obtained to date and may not be exhaustive. Plaintiff's investigation of Defendants' infringement is ongoing. Plaintiff reserves the right to supplement and/or amend these PICS to identify additional instrumentalities and to further identify where each element of each claim is found in each accused instrumentality, including on the basis of discovery obtained from Defendants, and from third parties during the course of this litigation, pursuant to ¶2 of the Order Governing Proceedings – Patent Cases under Hon. Alan D. Albright.

[2] All PICS set forth herein for any independent patent claims are hereby incorporated by reference into the PICS alleged for any dependent patent claims that depend on such independent claims, as if fully set forth therein.

[3] The Accused Instrumentalities and associated exhibits discussed and/or cited for any claim herein are representative in all material respects of all other accused instrumentalities identified for that claim (e.g., a specified device or service may be used as a representative example in these charts since the other accused instrumentalities have immaterial differences in their hardware and/or software configuration, the cited references are believed to be illustrative of all such accused devices).

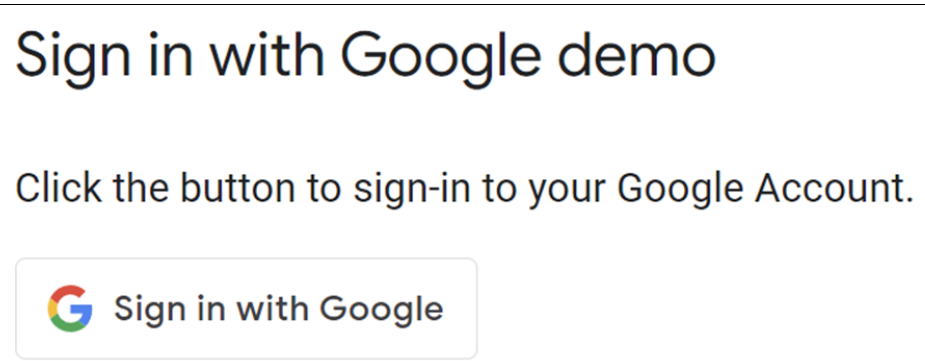
[4] Plaintiff's inclusion of any claim preamble in this claim chart should not be interpreted as an admission that the preamble is limiting. Plaintiff reserves the right to take the position that the claim preambles are limiting or not limiting on a claim-by-claim basis.

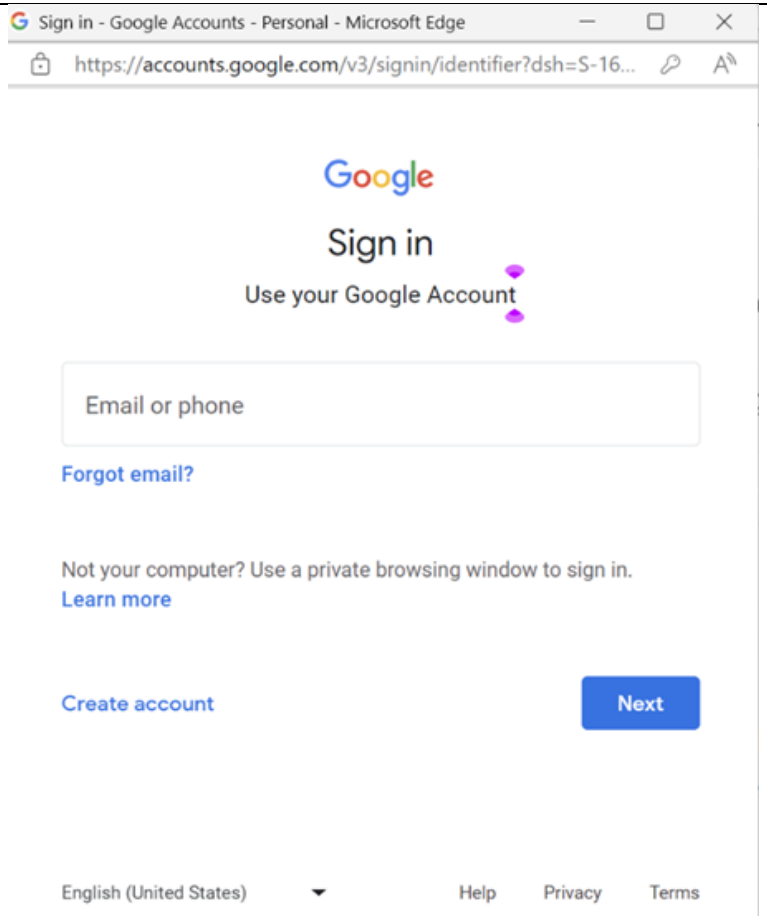
		such as Google Pay for online and in-app payments, Google Cloud services, such as G-mail, and Google Docs, and services provided by subscribers to Google Identity such as Shopify, E-Bay, Kayak, Blizzard Entertainment's Battle.Net, Electronic Arts, PayPal, and numerous others.
	creating a first wireless link between an integrated receiver-decoder circuit (RDC) of a hybrid device and an external personal digital key (PDK), the hybrid device including an integrated PDK and the integrated RDC;	<u>creating a first wireless link between an integrated receiver-decoder circuit (RDC) of a hybrid device and an external personal digital key (PDK)</u> Devices utilizing Android OS communicate passkey signatures over an encrypted connection, through the internet, via Wi-Fi, and/or cellular protocols. PROX_GOOGLE_003192, Google Online Security Blog: So long passwords, thanks for all the phish (googleblog.com) ("The device then verifies that your phone is in proximity using a small anonymous Bluetooth message and sets up an end-to-end encrypted connection to the phone through the internet."). Android devices utilizing Android's native the FIDO compliant native SafetyNet Authenticator, accordingly, include an RDC enabling encrypted communications. Establishing the connection between the Android device and an external device running the Chrome browser requires a first wireless connection between the two devices. <u>the hybrid device including an integrated PDK</u> Google's Android OS natively includes the Android SafetyNet Authenticator, which is FIDO certified, and accessible via the FIDO2 API for Android. *See* FIDO® Certified - FIDO Alliance, **PROX_GOOGLE_003793;** and **PROX_GOOGLE_003775** ("The FIDO2 API allows Android applications to create and use strong, attested public key-based credentials for the purpose of authenticating users.") FIDO compliant authenticators (i.e., security keys) contain a PDK. The API provides a WebAuthn Client implementation, which supports the use of BLE, NFC, and USB roaming authenticators (security keys) as well as a platform authenticator, which allows the user to authenticate using their fingerprint or screen lock."). Accordingly, all Android devices contain an authenticator that has met the standards set by the FIDO Alliance. The 042 Patent defines a PDK as including "an antenna and a transceiver for communicating with an RDC (not shown) and a controller and memory for storing information particular to a user". 042 Patent, 13: 46-49. A component of the integrated PDK of the hybrid device, accordingly, is "a controller and memory for storing information particular to a user". The 042 defines the operation of the controller and memory for storing information particular to the user as entailing the receipt of an access key from an external application that is used to access a specific service block. 042

		Patent, 6:23-39 ("Regardless of how created, once created, external applications (such as applications 120 in FIG. 1) can gain access to specific service block 112 by providing the corresponding access key 118. In FIG. 2., this is shown conceptually by control logic 250.") The integrated PDK of the hybrid device, therefore, includes a controller placed within memory information that can only be accessed by a corresponding access key provided by an external application. Such memory is present in Google's FIDO compliant authenticators because it is required by the standard. The FIDO CTAP specification incorporates the WebAuthn Specification. PROX_GOOGLE_002926, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 1.1 ("This specification is part of the FIDO2 project, which includes this specification and is related to the W3C [WebAuthn] specification."). Under the WebAuthn specification, "compliant authenticators protect public key credentials." PROX_GOOGLE_003287, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 1. A public key credential refers to a public key credential source, which includes a credential ID. PROX_GOOGLE_003287, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 4 (Defining "public key credential" and "public key credential source".). The credential ID uniquely identifies its public key credential source. *See P*ROX_GOOGLE_003287*,* Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 4 (Defining a credential ID as "A probabilistically-unique byte sequence identifying a public key credential source and its authentication assertions."). In addition to the credential ID, each public key credential source contains a "credential private key". PROX_GOOGLE_003287, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 4, (Defining "public key credential source" as data structure including the credential private key and the credential ID.). "The credential private key is bound to a particular authenticator" and part of an asymmetric key pair containing a public key returned to a relying party. PROX_GOOGLE_003287, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 4 (Defining a "credential key pair".). Every FIDO compliant authenticator, therefore, will store within memory a credential comprising a private key of an asymmetric key pair and a credential ID uniquely identifying the private/public key pair to which the private key belongs. The credentials stored within a compliant authenticator can only be accessed with the appropriate access key. "A public key credential can only be used for authentication with the same entity (as identified by the RP ID) it was registered with." PROX_GOOGLE_003287, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 4 ("A public key credential can only be used for authentication with the same entity (as identified by RP ID) it was registered with."). When generating a response, therefore, the authenticator will only retrieve credentials corresponding to the RP ID provided to it by the external FIDO server. PROX_GOOGLE_002926, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 6.2.2 ("7.1 If the allowList parameter is present

		and is non-empty, locate all denoted credentials created by this authenticator and bound to the specified rpId. 7.2 If an allowList is not present, locate all discoverable credentials that are created by this authenticator and bound to the specified rpId.”). As only credentials corresponding to the RP ID will be retrieved, the RP ID is an access key. During a WebAuthn authentication ceremony, an authenticator receives an “authenticatorGetAssertion” request to provide cryptographic proof of user authentication. PROX_GOOGLE_002926, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 6.2 (Defining authenticatorGetAssertion as the method “used by a host to request cryptographic proof of user authentication as well as user consent to a given transaction, using a previously generated credential that is bound to the authenticator and relying party identifier.”). The authenticatorGetAssertion request contains a relying party identifier (RP ID). PROX_GOOGLE_002926, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 6.2 (Defining the input parameters of the authenticatorGetAssertion as including a required relying party identifier.). The authenticatorGetAssertion is called in response to get request issued by the relying party attempting authentication. PROX_GOOGLE_003287, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 5.1.4 (“WebAuthn Relying Parties call navigator.credentials.get({publicKey:..., ...}) to discover and use an existing public key credential, with the user’s consent… If the user picks a credential source, the user agent then uses § 6.3.3 The authenticatorGetAssertion Operation to sign a Relying Party-provided challenge and other collected data into an assertion, which is used as a credential.”). When get() is called by the relying party, “The RP ID defaults to being the caller’s origin's effective domain unless the caller has explicitly set options.rpId when calling get().”PROX_GOOGLE_003287, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 5.1.4. The RP ID, therefore, is provided by the relying party attempting to authenticate the external authenticator. As an authenticator will only return credentials corresponding to the RP ID access key provided by the external relying party, the authenticator has the controller and memory necessary for a minimal embodiment of a PDK. In addition to the controller and memory, a minimal embodiment of PDK is defined by the 042 Patent as including “an antenna and a transceiver for communication with an RDC”. 042 Patent, 13:46-48. While the native Android OS authenticator permits the use of passkey in a variety of situations, each transmits via the internet. When a user wishes to use a passkey stored on their Android device to access a resource via a browser the user is presented with either a QR code or push notification. *See* PROX_GOOGLE_003236, Passkey support on Android and Chrome | Authentication | Google for Developers (“Depending on the desktop operating system (e.g. ChromeOS, iOS, MacOS, Windows) users may be presented with a QR code to securely use a passkey stored on their mobile device or a notification may be displayed prompting the user to unlock their device to use the relevant passkey.”). In the case of a QR code, the user scans a QR code displayed on the screen of a device. PROX_GOOGLE_003192, Google Online Security Blog:

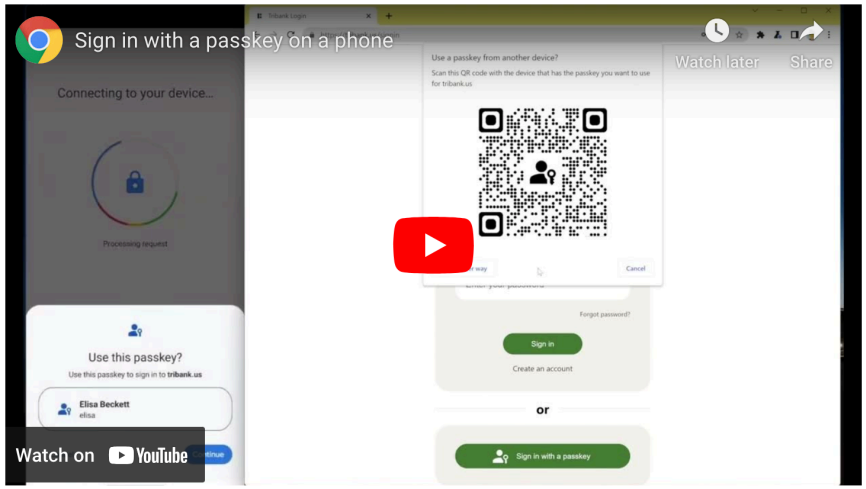
		So long passwords, thanks for all the phish (googleblog.com) ("When you do need to use a passkey from your phone to sign in on another device, the first step is usually to scan a QR code displayed by that device."). Proximity between the devices is then verified using Bluetooth. PROX_GOOGLE_003239, Passkeys use cases | Authentication | Google for Developers ("The two devices verify that they are in proximity with each other using Bluetooth."). The Bluetooth connection is also utilized to establish a secure connection between the devices via the internet. PROX_GOOGLE_003192, Google Online Security Blog: So long passwords, thanks for all the phish (googleblog.com) ("The device then verifies that your phone is in proximity using a small anonymous Bluetooth message and sets up an end-to-end encrypted connection to the phone through the internet.") Utilizing the internet connection, the Android device sends one-time passkey signature, including the device-bound public key, to the other device, which is used to sign in the user. PROX_GOOGLE_003239, Passkeys use cases | Authentication | Google for Developers ("A one-time passkey signature is transferred back to Safari on the Mac, which the website then uses to sign the user in."). Establishing the connection over the internet would require utilizing either the device's Wi-Fi or cellular capabilities – both of which are wireless protocols. Accordingly, tablets and devices implementing the Android OS and native SafetyNet Authenticator have the antenna and transceiver necessary to implement the wireless protocols enabling transmission over the internet. Push notifications can originate with the "Sign in with Google" button. To receive the benefit of quickly and easily managing user authentication utilizing "Sign-in with Google", the website must display a "Sign in with Google" button, which is generated by the Google Identity Services JavaScript library, or provide the user the option to sign in with a passkey. PROX_GOOGLE_003779 ("Put it [*sic*] another way, the Sign in with Google button must be generated by the Google Identity Services JavaScript library now. The button rendering API allows you to customize the color, shape, text, and size to meet the branding requirements of your website, whereas still stick to Google's guidelines."); and PROX_GOOGLE_003239. A functional demonstration of the "Sign in with Google" button is provided in Google Identity documentation, as shown below.

		 *See* PROX_GOOGLE_003232 When this button is clicked, the dialog box pictured below is presented to the user.



The user is prompted by Google to enter an email and then click "next." For all users that have activated "Use your phone to sign in" in their Google Account, a dialog box is presented by Google directing the user to complete the sign-on with their phone, which has received a push notification. The push notification includes a challenge from Google Identity servers for the FIDO client on the user's Android device. The FIDO client will invoke the Android SafetyNet Authenticator to complete the challenge by prompting the user to biometrically unlock the stored passkey, use the passkey to sign a challenge, and return the signature and accompanying device-bound public key to Google Identity, as detailed above. Returning the signed challenge in response to the push notification requires establishing the connection over the internet would require utilizing either the device's Wi-Fi or cellular capabilities – both of which are wireless protocols. Accordingly, tablets and devices implementing the Android OS and native SafetyNet Authenticator have the antenna and transceiver necessary to implement the wireless protocols enabling transmission over the internet.

Having each of the elements of a minimal embodiment of a PDK, Android devices including Android's native FIDO compliant native SafetyNet Authenticator include an integrated PDK. *See also:*



For example, a user visits `example.com` on the Chrome browser on their Windows machine. This user has previously logged into `example.com` on their Android device and generated a passkey. On the Windows machine, the user chooses to sign in with a passkey from another device. The two devices will connect and the user will be prompted to approve the use of their passkey on the Android device, for example, with a fingerprint sensor. After doing so, they're signed in on the Windows machine. Note that the passkey itself isn't transferred to the Windows machine, so typically `example.com` will offer to create a new passkey there. That way, the phone isn't required next time the user wants to sign in. Read Sign-in with a phone to learn more.

PROX_GOOGLE_003243, Passwordless login with passkeys | Authentication | Google for Developers https://developers.google.com/identity/passkeys; Sig n in with a passkey on a phone PROX_GOOGLE_003833, (https://www.youtube.com/watch?v=ywQ8bFla-L8).

the hybrid device including … the integrated RDC

Devices utilizing the Android OS with the native SafetyNet Authenticator must include an RDC to communicate passkey signatures over an encrypted connection, through the internet, via Wi-Fi, and/or cellular protocols. PROX_GOOGLE_003192, Google Online Security Blog: So long passwords, thanks for all the phish (googleblog.com) ("The device then verifies that your phone is in proximity using a small anonymous Bluetooth message and sets up an end-to-end encrypted connection to the phone through the internet."). Android devices utilizing Android's native FIDO

compliant native SafetyNet Authenticator, accordingly, include an RDC enabling encrypted communications.

an external personal digital key (PDK)

External PDKs may include any external device with a supported browser and operating system. At a high level, infringement of the method occurs when a device, like an Android phone with SafetyNet Authenticator (*i.e.*, an integrated RDC of a hybrid device) is wirelessly connected (*e.g.*, through a HTTPS connection) to a browser on another, external device (*i.e.*, an external PDK).

As noted above, FIDO compliant authenticators include the elements of a minimal embodiment of a PDK. Google's Chrome Web Browser supports using passkeys on all desktop platforms. PROX_GOOGLE_003236, Passkey support on Android and Chrome | Authentication | Google for Developers ("Chrome on all desktop platforms supports using passkeys from mobile devices."). As noted in the below chart, this includes signing in with an Android device utilizing Android's native FIDO compliant native SafetyNet Authenticator.

Chrome's passkey support summary

Operating systems	Android	macOS	iOS/iPadOS	Windows	Linux
Local user verification	✓	✓	✓	✓	⊘
Passkey sync	✓	⏲[1]	✓[1]	⊘[3]	⊘
Autofill	✓	✓	✓	✓[2]	⊘
Can sign in with a phone	⏲	✓	✓	✓	✓

✓ : Supported, ⏲: Planned, ⊘ : No plans

[1]: Syncs with iCloud Keychain [2]: Requires Windows 11 22H2 [3]: Depends on Windows Hello

PROX_GOOGLE_003236, Passkey support on Android and Chrome | Authentication | Google for Developers

A Mac (macOS), iPhone (iOS), iPad (iPadOS), and a Windows PC with the Google Chrome web browser, accordingly, are all externals PDK. The use of Passkeys is not restricted to Chrome browser. "As passkeys are built on FIDO standards, all browsers can adopt them." PROX_GOOGLE_003243, Passwordless login with passkeys | Authentication | Google for Developers. In addition to Chrome, browsers currently supporting the use of Google Identity

		Passkeys include Microsoft Edge and Apple Safari. *See* PROX_GOOGLE_003851, Sign in with a passkey instead of a password - Google Account Help (Listing Safari 16 or up and Edge 109 or up as support browsers). Furthermore, "Passkeys are intended to be used through the operating system infrastructure that allows passkey mangers to create, backup, and make passkey available to the applications running on the operation system." PROX_GOOGLE_003243, Passwordless login with passkeys | Authentication | Google for Developers. Accordingly, any external device with a supported browser and operating system may be an external PDK.
	receiving a first signal at the integrated RDC via the first wireless link from the external PDK;	During the authentication process discussed *supra*, an external authenticator (Android OS device) receives an "authenticatorGetAssertion" request to provide cryptographic proof of user authentication. PROX_GOOGLE_002926, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 6.2 (Defining authenticatorGetAssertion as the method "used by a host to request cryptographic proof of user authentication as well as user consent to a given transaction, using a previously generated credential that is bound to the authenticator and relying party identifier."). Browsers, such as Google Chrome, forward the authenticatorGetAssertion to the external authenticator. PROX_GOOGLE_003243, Passwordless login with passkeys | Authentication | Google Developers ("When a user wants to sign in to a service that uses passkeys, their browser or operating system will help them select and use the right passkey."); and PROX_GOOGLE_003192, Google Online Security Blog: So long passwords, thanks for all the phish (googleblog.com) ("The device then verifies that your phone is in proximity using a small anonymous Bluetooth message and sets up an end-to-end encrypted connection to the phone through the internet."). As such, Chrome will connect with the Android device to forward to the SafetyNet Authenticator the authenticatorGetAssertion request received via the Chrome web browser.
	generating an enablement signal enabling one or more of an application, a function and a service on one or more of the hybrid device and a device associated with an external RDC;	<u>generating an enablement signal enabling one or more of an application, a function and a service on … a device associated with an external RDC</u> During a WebAuthn authentication ceremony, an external authenticator (Adnroid OS device) receives an "authenticatorGetAssertion" request to provide cryptographic proof of user authentication. PROX_GOOGLE_002926, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 6.2 (Defining authenticatorGetAssertion as the method "used by a host to request cryptographic proof of user authentication as well as user consent to a given transaction, using a previously generated credential that is bound to the authenticator and relying party identifier."). When successfully invoked, the authenticatorGetAssertion causes the authenticator to return a response including "the credential identifier whose private key was used to generate the assertion." PROX_GOOGLE_002926, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 6.2.2. As noted above, a credential can only be accessed when an authenticator is applied by the appropriate relying party identifier, such that the relying party identifier is an access key. When

		generating a response, therefore, the authenticator will only retrieve credentials corresponding to the RP ID provided to it by the external FIDO server. PROX_GOOGLE_002926, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 6.2.2 ("7.1 If the allowList parameter is present and is non-empty, locate all denoted credentials created by this authenticator and bound to the specified rpId. 7.2 If an allowList is not present, locate all discoverable credentials that are created by this authenticator and bound to the specified rpId."). Authentication is a service provided by a FIDO server operated by Google, and the credential ID is necessary for Google's FIDO server to perform the authentication function. Upon receiving the response (i.e., enablement signal), the WebAuthn/FIDO server will use the credential ID to locate the appropriate public key to verify a signature generated with the private key held by the authenticator. Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 7.2 ("7. Using credential.id (or credential.rawId, if base64url encoding is inappropriate for your use case), look up the corresponding credential public key and let credentialPublicKey be that credential public key… 20. Using credentialPublicKey, verify that sig is a valid signature over the binary concatenation of authData and hash… 22. If all the above steps are successful, continue with the authentication ceremony as appropriate. Otherwise, fail the authentication ceremony.") "[I]f an authenticator returns the wrong credential ID, or if an attacker intercepts and manipulates the credential ID, is that the WebAuthn Relying Party would not look up the correct credential public key with which to verify the returned signed authenticator data (a.k.a., assertion), and thus the interaction would end in an error." PROX_GOOGLE_003287, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 13.1. As the proper credential ID is needed for Google's FIDO server to authenticate a user, and the credential ID is included within a response to a get request having the appropriate relying party ID received from Google's FIDO server, the response to the authenticatorGetAssertion request generated by the authenticator is an enablement signal enabling authentication by Google's FIDO server. Google's FIDO server is associated with either an instance of the Edge or Chrome browser running on the Windows PC linked to the authenticator. PROX_GOOGLE_003243, Passwordless login with passkeys | Authentication | Google Developers ("When a user wants to sign in to a service that uses passkeys, their browser or operating system will help them select and use the right passkey."); and PROX_GOOGLE_003837, All about FIDO2, CTAP2, and WebAuthn - Microsoft Community Hub ("CTAP2 and WebAuthn define an abstraction layer that creates an ecosystem for strongly authenticated credentials. Any interoperable client (such as a native app or browser) running on a given 'client device' can use a standardized method to interact with any interoperable authenticator – which could mean a platform authenticator that is built into the client device or a roaming authenticator that is connected to the client device through USB, BLE, or NFC."). The authenticator, accordingly, generates an enablement signal enabling one or more of an application, a function, and a service on a device associated with an external RDC.

	sending the enablement signal to one or more of the hybrid device and the device associated with an external RDC.	The responsibility for sending responses received from an authenticator falls upon the WebAuthn Client. PROX_GOOGLE_003287, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 4 ("A WebAuthn Client is an intermediary entity typically implemented in the user agent (in whole, or in part). Conceptually, it underlies the Web Authentication API and embodies the implementation of the [[Create]](origin, options, sameOriginWithAncestors) and [[DiscoverFromExternalSource]](origin, options, sameOriginWithAncestors) internal methods. It is responsible for both marshaling the inputs for the underlying authenticator operations and for returning the results of the latter operations to the Web Authentication API's callers."). Google identifies its Chrome Browser as the WebAuthn client. PROX_GOOGLE_003236, Passkey support on Android and Chrome | Authentication | Google for Developers ("Google Password Manager on Chrome helps create and sign in with passkeys. Depending on the desktop operating system (e.g., ChromeOS, iOS, MacOS, Windows) users may be presented with a QR code to securely use a passkey stored on their mobile device, or a notification may be displayed prompting the user to unlock their phone to use the relevant passkey.") and ("Chrome on all desktop platforms supports using passkeys from mobile devices."). In addition to Chrome, browsers currently supporting the use of Google Identity Passkeys include Microsoft Edge and Apple Safari. *See* PROX_GOOGLE_003851, Sign in with a passkey instead of a password - Google Account Help (Listing Safari 16 or up and Edge 109 or up as support browsers). In fact, "As passkeys are built on FIDO standards, all browsers can adopt them." PROX_GOOGLE_003243, Passwordless login with passkeys | Authentication | Google for Developers. As such, Chrome, or another browser supporting the FIOD standards, on the device with external RDC will forward the enablement signal received from a roaming authenticator of the PDK to Google's FIDO server, will be associated with the external RDC for the purpose of signing the user into a website using a passkey signature and credential ID received from the PDK.