Synced Passkey Deployment: Emerging Practices for Consumer Use Cases

Date: January 2024

Editors:
Hideaki Furukawa, NRI SecureTechnologies, Ltd.
Tatsuya Karino, Mercari

Abstract

This paper explores the emerging practices surrounding the use of synced passkeys which allow passkey use across multiple devices by syncing the passkeys over the cloud, specifically addressing the initial choices and considerations for service providers (aka relying parties or RPs). These practices are in their early stages and are likely to progress, since operating systems, browsers, and passkey providers are still in a phase of enhancing functionality. This document outlines crucial areas such as registration, authentication, passkey management, and accessibility for RPs to consider and presents a range of emerging approaches for adopting this technology. The objective is to guide RPs through these budding strategies, acknowledging that the specifics of ensuring secure and convenient passkey usage may evolve as the digital landscape continues to advance.

This paper is written with independence for each section, allowing readers to read specific topics of interest without the need to read the entire paper from the beginning.

Audience

This white paper is intended for various stakeholders of relying parties, including non-developers, such as information security executives, product owners, identity and access management practitioners, UI/UX designers, and accessibility practitioners.

Contents

Tables

1. Introduction

Synced passkeys, which allow users to use passkeys across multiple devices by syncing the passkeys over the cloud, are still in a relatively early phase of adoption at the time of this paper's publication, yet there are already many emerging deployment practices by RPs. This paper presents these emerging deployment practices to help readers understand various elements concerning passkey deployments. Emerging practices being deployed by some RPs at the time of this writing may become the best practices over time given accumulated experience and evolution in passkey providers. Due to the dynamics of passkey usage, the content in this white paper should be updated periodically to accommodate changes. The FIDO Alliance is already addressing passkey user experiences and has published UX Guidelines for product managers and architects as well as designers. While this paper presents various emerging deployment practices, it also advocates the recommendations of the UX Guideline [1] where possible.

In 2022, FIDO Alliance, Apple, Google, and Microsoft announced their intentions to support synced passkeys [2], FIDO credentials that may be backed up and made available across devices that are registered to the same passkey provider and within its ecosystem. Many OS and browser vendors have already begun supporting passkeys and these passkeys are becoming ubiquitously available on those OSes and browsers [3]. As a result, there are already dozens of consumer services that have adopted passkeys as an authentication method [4].

Note that "passkey" can be categorized as either synced or device-bound. According to "FIDO Deploying Passkeys in the Enterprise - Introduction" (June 2023) [5] passkeys are described as follows:

> *It is important to also note that FIDO Alliance has embraced the term "passkey" to describe any passwordless FIDO credential. This includes synced passkeys (consistent with the original announcement and intent) as well as device-bound passkeys – which are FIDO authentication credentials that cannot leave the issued device (e.g., on a FIDO Security Key).*

In consumer use cases where consumers are using multiple devices and/or lose a device, it is desirable for the majority of RPs to utilize synced passkeys, as users can easily use the same passkey for an RP across different devices. However, this paper also contemplates using device-bound passkeys since consumers may use them instead of, or in addition to, synced passkeys. For example, even if an RP aims to prioritize the deployment of synced passkeys, there are user environments where synced passkeys cannot be utilized. Therefore, though this paper states "synced passkey" in its title, issues and emerging practices regarding device-bound passkeys are also presented.

2. What are the types of passkeys?

Passkeys are based on cryptographic key pairs and have strong phishing resistance. In addition, passkeys enhance user experience as they allow users to choose from a list of available accounts' passkeys instead of typing in an account ID on sign-in. Passkeys surpass passwords in usability in that there is nothing that must be remembered. The advantage of a synced passkey is that a user can use a passkey from multiple devices and restore passkeys to another device if the original device used for registration is lost. At the same time, synced passkeys may invoke additional considerations for RPs. Synced passkeys can provide far better security than password-only authentication and phishable methods such as SMS OTP. Synced passkeys also provide superior usability for a wide range of consumer services, however, since key materials are synchronized through a cloud-based service (passkey provider's sync fabric), for higher-assurance services, RPs may need to consider the security of the cloud-based service as well as export and management of credentials by a third-party passkey provider.

RPs should also be aware that passkeys may be either synced or device bound. Either might be registered to a FIDO server, unless the FIDO server detects whether a passkey is synced or not synced, and decides to accept or deny registration of synced/not synced passkeys as described later in this document.

Furthermore, RPs should be aware that passkeys have different features depending on a user's environment, as shown in Table 1 below. For example, some passkeys can be managed by a passkey provider where the user can see a list of passkeys stored in the passkey provider and then choose to edit a passkey name or delete passkeys. Some OSes and browsers support a user's ability to use passkeys on autofill, meaning that a user can select a username/ID box and a list of available passkeys appears.

Table 1 - Types of passkeys (as of December 15, 2023)

Description and examples	Synced Passkey iOS and iPad OS newer than 16.0 / macOS newer than 13.0 (Safari, Chrome) / Android / third-party passkey provider	"Device-Bound Passkey"- Security key	"Device-Bound Passkey"- Windows 11 / macOS newer than 13.0 (Chrome, on users' choice)	"Device-Bound Passkey"- Windows 10 / iOS and iPadOS older than 16.0 / macOS older than 13.0 / macOS newer than 13.0 (Edge)	Non-discoverable credential Security key (when not set as a discoverable credential)	Non-discoverable credential Android (when not set as a discoverable credential)
Can be synced	Yes	No	No	No	No	No
Bound to a device	No	Yes	Yes	Yes	Yes	Yes
Can be managed on passkey provider	Yes	No	Yes	No	No	No
Can be used on autofill	Yes	No	Yes	No	No	No
Can be used with an empty *allow Credentials*	Yes	Yes	Yes	Yes	No	No
Platform or roaming authenticator	Platform*	Roaming	Platform	Platform	Roaming	Platform

*Roaming when registered using Cross-Device Authentication

2.1 How do types of passkeys impact consumers and RPs?

For consumers, whether passkeys are synced or not (device-bound), differentiates their user experience. If passkeys are synced, users can use the passkey from any device accepted by the same passkey provider. This covers a user's loss of a device, change of a device and use of multiple devices. On the other hand, if the user chooses to use device-bound passkeys, the user is responsible for ensuring they have backup authenticators available.

For RPs, synced passkeys improve the likelihood of account recovery, as compared to device-bound passkeys, since users can easily restore passkeys from a synced device. However, when using passkeys as an authentication method, alternative authentication methods and/or account recovery methods should be provided for users regardless of whether the passkeys are device-bound or synced.

2.2 Why may alternative authentication methods and account recovery methods still be needed for passkeys?

Possible reasons for providing alternative authentication methods and account recovery methods for passkeys are displayed below. When a user is using a device-bound passkey, they may lose access to the passkey just by losing or switching devices. Even when a user is using a synced passkey, they may encounter situations where they are unable to restore the passkey, either by losing access to their passkey provider, or by switching to a device that does not support the previous passkey provider.

Table 2 below lists possible reasons and frequency for a user to lose access to their passkey.

Table 2 - Possible reasons users may lose access to a device-bound or synced passkey

Possible reason for a user to lose access to their passkey	Device-bound passkey	Synced passkey
Lose the device that registered a passkey	Occasionally	N/A
Ban from a passkey provider	N/A	Rarely
Lose authenticator(s) such as password and other multi factor methods to sign-in to a passkey provider	N/A	Rarely
Use (switch to) a different device that does not support the previous passkey provider	N/A	Occasionally
Change passkey provider without passkeys transfer	N/A	Occasionally

2.3 How can RPs detect whether a registered passkey is synced or not synced?

Users have the option to register synced passkeys or device-bound passkeys. How a passkey is registered is important because device bound keys are not recoverable if a user loses access to the device they used to register. In this case, RPs could guide users to register additional passkey(s) and/or add authentication method(s). RPs can inspect the *backupEligible* flag on the passkey management page to determine if an authentication passkey is synced or device bound. If the passkey is device bound, the RP could advise a user to add a passkey or authentication method to prevent them from being locked out of their account upon loss of device. (see section 5.2 for information on additional fields available on the passkey management page).

3. Passkey Registration

This section explains how RPs interface with users regarding how to register, display names, passkey registration security, how to handle device-bound passkeys, and registration of various platform devices using Cross-Device Authentication.

3.1 How to Recommend registering a passkey to a user?

Despite passkeys being secure and convenient, users may not actively register passkeys if the RP only provides a passkey settings page. Many RPs provide various ways to prompt users to register passkeys to existing accounts as shown below in Table 3.

Table 3 - Deployment examples: How to recommend registering a passkey to a user

Deployment case	Possible reason(s) and concern(s)
Enforce passkey registration during the account creation process or as a precondition to using high-security features	This is the most secure approach against phishing attacks, and the passkey registration rate can be significantly increased. However, there is potential for a decrease in the registration success rate with users whose devices do not support passkeys.
Suggest passkey registration during account-related operations such as account creation and account recovery	**This approach is recommended by UX guidelines.** There is little likelihood that users will find passkey registration burdensome because it encourages the use of passkeys to alleviate user friction during account-related operations. It may not significantly boost the passkey registration rate, primarily because opportunities to perform account-related operations are relatively infrequent.

Suggest passkey registration after signing in with a traditional authentication method.	This resolves user friction during sign-in while encouraging passkey registration. However, it may be an unwelcome interruption to users who are used to traditional authentication methods that do not create friction during core site tasks.

Several RPs mandate the registration of a passkey when creating an account. This approach is the most secure approach because the risk of account takeovers through phishing escalates with the elapsed time between account creation and the registration of a phishing-resistant authentication method such as passkeys. Shortening the time between account creation and the registration of a passkey decreases this risk.

The passkey registration rate can be significantly increased among users who are motivated to use a specific service or feature. However, forcing passkey registration during account creation could harm user experience because some user devices do not support passkeys, some users may not have set screen locks and some users believe (incorrectly) that biometrics are stored by the RP. Strong recommendation of passkey registrations during the account creation could drop the registration success rate because many consumers are not yet familiar with passkeys. Therefore, RPs taking this approach should include accompanying information to address potential concerns, per the recommendations of the UX guidelines [1].

Currently, most RPs do not mandate the registration of passkeys due to the likely negative impact on user experience and potential to reduce the rate of account registrations. The UX guidelines recommend, as a best practice, suggesting passkey registration during account-related operations, such as account creation, account recovery, and from the user's *account settings*. In these instances, there is little likelihood that users will find passkey registration burdensome since passkey registration is an optional. Since opportunities to perform account-related operations are relatively infrequent passkey registration rate may not increase.

Some RPs suggest users register a passkey when users access their account profile page. This approach can increase passkey registration rates since accessing the account profile page is more common than account-related operations and it does not significantly disrupt users' activities.

Finally, some RPs suggest users register a passkey after the sign-in process using traditional authentication methods. This approach is known to have a low conversion rate and, as stated earlier, can be perceived as an unwelcome interruption to the user while attempting to accomplish a core task, such as making a purchase.

3.2 What name should be displayed on the autofill and a passkey provider?

Users can select their account in the autofill display page presented upon passkey authentication or find their passkeys with a passkey provider. The name displayed on the autofill display and in the passkey provider is set on passkey registration as *user.name* and *user.displayName*. Prior to the introduction of synced passkeys, these names were only displayed on the passkey authentication UI provided by the OS. With the introduction of synced passkeys, they are now displayed on the autofill page for the user to choose an account to sign into on an RP site. As a result, determining appropriate values used for these names have become important.

Table 4 - Deployment examples: Display names on autofill and passkey provider

Deployment case	Possible reason(s)
Unique and not changeable within an RP *For example, email address, account username*	Users can identify their accounts in the autofill and there is no confusion because the information in an RP and a passkey provider is not shifted.
Unique but changeable within an RP *For example, email address, phone number*	Users can identify their account in the autofill, but the information in an RP and a passkey provider will be misaligned, causing possible confusion.

Possibly not unique and possibly changeable within an RP *For example, masked email address, nickname*	Users may not be able to identify the account in the autofill, or the information in an RP and a passkey provider will be misaligned, which may cause confusion.

It is essential for users to be able to easily distinguish which account is displayed in the autofill. The unique identifier for a user account is preferably user.name and user.displayName. Moreover, an unchangeable identifier is more desirable since there is no way to change the *user.name* and *user.displayName* values registered in a password manager from the RP. If an RP possesses information that is both unique and immutable, utilizing this information is the most effective approach but it is crucial to note that not all RPs have this type of information.

Some RPs use unique but changeable information. In this case, if the values are changed on the RPs site, users may need to change the values in the passkey provider as well, otherwise a different name will be shown on the autofill. Currently, there is no effective way to resolve confusion in such cases; the best that an RP can do is to suggest that the user change the value in their passkey provider.

There are a few RPs using masked emails, nicknames, and similar identifiers that overlap within the overall system but are not commonly used by a specific user. The values will be managed by and may be visible from a user's passkey provider. We expect masking in these situations is done by RPs that are required to treat personally identifiable information (PII) with extreme caution. We do not recommend masking as an approach as it would make account identification more challenging. The consideration of this approach should depend on the RP's privacy decision.

3.3 Re-authentication before passkey registration

Passkeys provide good security to protect accounts and users. The overall security they provide may depend on how they are registered. One of the typical use cases of passkey registration is to add a passkey to an existing account after being authenticated with only a password. In this case, a user is authenticated with a password first to sign in. The question now is, do you need any additional authentication (i.e., re-authentication) for the user before enabling registration of a passkey with the account?

It is a common practice for RPs to initiate re-authentication before allowing users to conduct operations on credentials, such as changing passwords or sign-in methods. If an attacker gains access to a sign-in session, with methods such as phishing attack or Cross-Site Request Forgeries, they might change credentials to completely take over the account. This can occur even when synced passkeys are used.

Table 5 – Re-authentication on passkey registration

Deployment case	Possible reason(s)
No re-authentication required on passkey registration.	This allows users to easily register passkeys.
Require re-authentication with non-phishing-resistant authentication under certain conditions on passkey registration.	This can balance the reduction of risk by phishing and dealing with UX.
Require re-authentication with phishing-resistant authentication on passkey registration.	This can prevent users who have registered a passkey from phishing attacks.

There are different approaches as follows depending on each RP's policy:

- Some RPs do not require re-authentication during passkey registration. These RPs must accept the risk of an attacker completely hijacking the account by gaining access to the sign-in session.
- Most RPs require re-authentication during passkey registration, employing the same methods used to protect other credential management systems. This practice can significantly mitigate the risk of complete account takeover. However, if authentication factors that are not resistant to phishing are permitted, such attacks cannot be entirely prevented.
- RPs who anticipate significant impact on the security from phishing attacks should use phishing-resistant authentication methods for passkey management, such as passkey authentication.
- A small number of RPs may be concerned about the possibility of passkey provider accounts being compromised through phishing. Such RPs may need to verify whether authentication has been performed with factors they control, such as SMS OTP, in addition to synced passkeys. However, this approach can diminish user experience.

3.4 Should RPs intentionally restrict registering synced or device bound passkeys?

It may be difficult for users to understand whether a passkey they created is synced or device bound. Currently, there are browser and OS limits on support for both synced and device bound passkeys. Whether an RP specifies a type of passkey that can be registered may vary based on the RP's tolerance for risk from phishing attacks affecting the account.

Table 6 - Deployment examples: Registration of device-bound or synced passkeys

Deployment case	Possible reason(s)
Accept to register both device-bound and synced passkeys.	This increases the possibility for users to use passkeys for convenience. Since each platform supports different types of passkeys, an RP planning to deploy services across major platforms should support both synced passkeys and device-bound passkeys.
Only accept registration of synced passkeys.	This provides convenience as users do not have to perform passkey credential recovery.
Only accept registration of device-bound passkeys.	This provides higher assurance.

If RPs have strict security policies and do not want to rely solely upon the security of the passkey providers, they can choose to only accept device-bound passkeys which would require them to address the fact that device-bound keys are not offered by all OSs and prevent some users to register.

Some RPs that do not maintain a strict security policy still aim to use passkeys to reduce the risk of phishing attacks against RP accounts. For these RPs that strongly desire to improve usability by making the registration and recovery of passkeys easy, considering support for only synced passkeys might be beneficial. Because users may encounter certain limitations, such as the need for re-authentication during passkey registration, or challenges in falling back to alternative authentication methods when passkeys are not available, RPs prefer to avoid situations where registered passkeys cannot be used. Finally, synced passkeys are easily recovered if lost by authenticating with another device. With device-bound passkeys, if no back up authentication has been set up by the user, then there is no ability to recover a lost passkey, so it might be prudent for these RPs to consider exclusively supporting synced passkeys.

Since not all platforms/passkey providers currently support synced passkeys, supporting only synced passkeys might prevent some users from enjoying the associated user experience and security benefits. RPs in this case need to make a choice. One option is to allow the use of passkeys for only users who can use synced passkeys, which will result in fewer users encountering issues returning after registration and lower customer support operation costs. The other option of allowing both synced and device-bound passkeys results in a greater number of users utilizing passkeys, but may lead to more users experiencing passkey issues and potentially higher customer support operational costs.

3.5 How can RPs let users register passkeys to different platform's devices using Cross-Device Authentication transport?

Synced passkeys are available on any device providing the platform being used remains the same, unless the user has a third-party passkey provider. Users may want or need to register a passkey from a different platform's device and/or passkey provider. In such cases, Cross-Device Authentication transport can be utilized. This enables users to register a passkey with a different device or passkey provider from the one on which the service is currently running. A downside is that Cross-Device Authentication may be difficult for general users since it usually requests users to scan a QR code displayed on a client device with a secondary mobile device housing an authenticator – which must support WebAuthn, a camera for QR scanning, a stable connection for tunneling, and Bluetooth capabilities to register a passkey to the target (client) device. Note that usage of this approach can be restricted by RPs by setting the *authenticatorSelection.authenticatorAttachment* option as "platform".

Table 7 - Deployment examples: Registration with Cross-Device Authentication

Deployment case	Possible reason(s) and explanations
Specify "platform" for *authenticatorAttachment*	RPs can reduce the number of options presented in the passkey provider's dialog by specifying *authenticatorAttachment*="platform", thereby simplifying the process for users who may not have detailed knowledge of passkeys. This is especially useful in consumer applications, where there are very few users with a security key, and interest in using Cross-Device Authentication for passkey registration is limited.
No *authenticatorAttachment* specified	This allows users the option to register their passkey via security key or Cross-Device Authentication.
Choose either "platform" or "cross-platform" for *authenticatorAttachment* after user interaction on RP's website	To clarify the options for users, RPs first inquire through their UI whether users want to register their passkey on this device. Then, depending on the user's choice, they can set *authenticatorAttachment* to "platform" or "cross-platform" to reduce the number of options displayed in the passkey provider's dialog.

Many RPs specify "platform" as the *authenticatorAttachment*. This practice is assumed to be intended to simplify the user interface by reducing the number of options in the dialog presented by the passkey provider. This is particularly true for consumer applications, where very few users possess a security key. Moreover, the subset of users interested in registering via Cross-Device Authentication is considered to be even smaller, which diminishes the incentive for RPs to facilitate cross-platform options for passkey registration. Since most RPs do not mandate the use of a passkey for authentication, users wishing to sign-in to a different platform device can do so using another factor, such as a password or SMS OTP, and subsequently register a passkey on that device.

Other RPs do not specify "platform" as the *authenticatorAttachment*. This approach may be taken to ensure users are not restricted in their choice of authentication methods. We have not found any RPs that actively advocate for the use of Cross-Device Authentication during the registration process.

A few RPs have a user interface that allows choice of device for passkey creation and adjusts the *authenticatorAttachment* value accordingly. This allows RPs to simplify the passkey provider's dialog options without limiting the users' choices. Users are prompted to register the passkey on their current device or another device, and the *authenticatorAttachment* is then set to "platform" or "cross-platform" based on the users' decisions.

4. Passkey Authentication

This section discusses the UIs involved with passkey authentication, whether or not user verification should be required when a passkey is authenticated, whether it is necessary for RPs to combine synced passkeys with other authentication methods, and what RPs can suggest when users sign-in from a device without a registered passkey.

4.1 Which sign-in UI is suitable for passkey authentication?

There are several UI methods that can be used to sign-in to a RP account with passkeys, including using autofill, or the "Sign in with Passkey" button with/without identifier. The method most suitable for an RP depends on the sign-in user interface used by the RP. Since RPs must contemplate existing authentication methods when adopting passkey authentication, it is useful to consider which type of sign-in user interface is suitable for passkeys, while respecting current authentication methods.

Table 8 - Deployment examples: Sign-in UIs

Deployment case	Process to sign-in	Possible reason(s)
Identifier-first approach	User first encounters a sign-in screen with only a user identifier field. If a passkey already exists and autofill is available, the user can use the passkey without having to type an identifier. If a passkey does not exist, if autofill is not available, or if the user wants to sign in as a different user, they enter an identifier (such as an email address or username) on the screen. The system checks the entered identifier and displays the associated authentication methods. Users are then presented with the appropriate authentication elements and are prompted to the user to complete the sign-in process.	**This approach is recommended by FIDO Alliance's UX guidelines.** Autofill is initiated with a touch on the annotated input field with *WebAuthn*. To make the autofill work well, the input field should be in the first view of the sign-in process. Other various authentication elements can be used as alternatives after the identifier is entered.
Identifier and password approach	When a user accesses the sign-in screen, they encounter a field for identifier and another field for password. If a passkey already exists and autofill is available, the user can use the passkey without having to type an identifier. If autofill is not available or the user does not choose to use autofill, the user enters the identifier and the password, then the system prompts additional authentication, such as passkey and SMS OTP.	This approach attempts to prevent confusion for users who are accustomed to a sign-in screen with identifier and password fields by allowing users to choose passkeys using autofill.

Authentication method first approach	Upon accessing the sign-in screen, a user sees various sign-in methods and icons for different authentication providers Users select one available authentication element (e.g., password, biometric authentication, or third-party authentication provider). Based on the user's choice, the corresponding authentication process is initiated. For instance, selecting the passkey icon begins the passkey authentication process.	The sign-in screen may have been inherited and created prior to the introduction of passkeys.

The identifier-first approach can be matched to any passkeys, whether they are discoverable or not. If a user can access a discoverable passkey (discoverable credential), autofill can be used. If a user can only use a non-discoverable credential or they are using a browser which does not support autofill, the user can use the passkey after they've identified themselves. If a user only has a password and no passkey, they will have to navigate two screens: an identifier screen followed by a password input screen.

Some RPs are adopting the identifier + password approach, especially when there is still a need to support password usage. This approach provides a better UX for password users compared to the identifier-first approach, as it consolidates the login process into a single screen.

In each of the first two approaches, the passkey authentication is typically initiated through autofill or automatically initiated after the user's identifier is provided. Since some users prefer to use discoverable credentials even if a browser doesn't support autofill, some RPs will provide the 'sign-in with passkey' buttons on the initial or secondary screen.

If an RP is using the "authentication methods first approach", it's advisable to switch to the identifier-first approach or the identifier + password approach to make the experience more passkey-friendly. With the authentication methods first approach, there are two ways to initiate authentication with a passkey: 1. by pressing the "sign in with passkey button", or 2. by using autofill in the form that appears after selecting the sign-in with the password button. There are, however, several concerns with this approach. Users may forget that they created a passkey, and therefore not select that sign-in option. Also, if an RP offers various social sign-in methods, adding the sign in with a passkey button could worsen the user experience. If a user selects to sign in with a passkey button without having first registered a passkey, they could become confused because they won't be able to proceed having chosen that option.

4.2 Should user verification be forced on passkey authentication or not?

Enforcement of verification of users during passkey authentication can be controlled using the *options.authenticatorSelection.userVerification* parameter. See the table below for the verification options and when you may choose to use them.

Table 9 - Deployment examples: Verification during passkey authentication

Deployment case	Possible reason(s) and explanations
required	This situation satisfies two-factor and phishing-resistant authentication. It tends to be used as the first factor of a sign-in process. This could lead to a sub-optimal user experience, for example, when using an old macOS device that does not support Touch ID.
preferred	This situation satisfies phishing-resistant authentication. It tends to be used as the first factor of a sign-in process. The sub-optimal user experience on old devices can be avoided since verification is not required.
discouraged	This situation satisfies phishing-resistant authentication. It tends to be used with device-bound passkeys as the second factor of the sign in process. The sub-optimal user experience on old devices can be avoided.

RPs using the passkey authentication as the only factor of a sign-in process seem to be setting the *User Verification* option as *"required"* to enforce an authenticator to conduct a user verification during passkey registration and authentication. *User Verification* serves as a method to implement a passkey authentication as two-factor authentication. This setting should prevent security attacks, such as a stolen device being used to authenticate to an RP.

Some RPs set the user verification option as *"preferred"* to allow authenticators to skip a user verification if a device does not support biometric authentication. In this case, passkey authentication still has phishing resistance which makes it a viable option for RPs. A few RPs use the "discouraged" user verification option with device-bound passkeys since they tend to use a second factor, similar to FIDO U2F, for their sign-in process. In this use case, user verification is not necessary.

4.3 Why additional authentication methods may be needed for some use cases?

One of the advantages of passkeys is that they allow for multi-factor authentication without the need to combine with other authentication methods, such as a password plus an SMS OTP. This is because passkeys use two forms of authentication: possession-based authentication through public key cryptography, and user verification, either knowledge-based (e.g., PIN) or biometric authentication (e.g., fingerprint).

Using synced passkeys, however, may increase the attack surface in certain scenarios since passkeys are shared across multiple devices. For example, passkey providers may have different assurance levels; if a user is required to sign in to authenticate, an attacker could successfully compromise a user's passkey provider to steal the user's passkeys and add them to the attacker's device as well. This means the attacker could hijack those passkeys.

Most RPs have a unique situation and unique use cases, so each RP must determine the trade-off of usability and security when deciding whether additional authentication methods are needed.

Table 10 - Deployment Examples: Combining passkeys with other authentication methods

Deployment case	Possible reason(s)
Not combined with other authentication methods	Passkeys alone can fulfill multi-factor authentication security requirements, so no additional authentication method is necessary.
Combined with SMS OTP/TOTP	This solution avoids relying only on the security of passkey providers by combining passkey authentication with other authentication methods that are controlled by RPs themselves.
Combined with a password	This approach involves using the passkey as the second factor in the authentication processes. It is intended to serve as a transitional solution, moving from the traditional password authentication process towards systems that incorporate passkeys.

In general, most RPs do not combine passkey authentication with other authentication methods primarily because a passkey alone can serve as multi-factor authentication that is secure enough for their use cases. Furthermore, for RPs that use social login as one of their sign-in methods, deployment of synced passkeys does not necessarily expand the attack surface because there is already an inherent risk of RP account-takeovers due to compromised external accounts. In terms of usability, passkeys offer advantages over existing multi-factor authentication methods such as SMS OTP and TOTP, because users don't have to wait for an OTP to be sent via SMS or launch a separate application to obtain a TOTP.

On the other hand, there are RPs that may choose to combine passkey authentication with existing SMS OTP/TOTP or other authentication methods managed by the RP. With this additional authentication, even if attackers compromise passkey provider accounts, attackers will not be able to access RP accounts. This may be particularly important for RPs not utilizing social logins. Though this additional authentication approach degrades the usability advantages offered by passkeys, it should be noted that additional risk-based authentication may optimize security and usability.

Other RPs use a combination of passkey and password, primarily for usability rather than security reasons. From a security perspective, there are minimal benefits to combining passkeys and passwords because both can be managed by passkey providers. In the event a passkey provider account is successfully phished an attacker could obtain both sets of credentials. On the other hand, there are usability advantages to replacing traditional MFA with passkeys. Some browsers lack autofill support, leading users to rely on passwords out of habit. In such cases, users are required to perform MFA, and using a passkey is more user-friendly than traditional MFA methods such as SMS OTP or TOTP. As a result, the combination of passkey and password is used. This combination may serve as a transitional solution and become less prevalent in the future, as autofill becomes widely supported in browsers and/or before RPs adopt the identifier-first approach for sign-in processes.

4.4 What do RPs suggest when users sign in from a device without a registered passkey?

When users need to sign in from a device that has no registered passkey, they can use the Cross-Device Authentication method. Cross-Device Authentication allows a user to use one device to authenticate with a passkey on a different device, either running on a different platform or is using a different passkey provider than was originally used. As mentioned earlier in this document, Cross-Device Authentication may be difficult for general users since it usually requests that a user first scan a QR code displayed on one device with another device which holds a permissible passcode. The RP then requires the user to proceed through additional complicated steps to complete the sign in process.

Table 11 - Deployment examples: Using device without a registered passkey

Deployment case	Possible reason(s)
Recommend using alternative authentication methods	Authentication with a passkey is not mandatory, and for some users, traditional authentication methods such as passwords + SMS OTP may be more understandable than using Cross-Device Authentication. Therefore, RPs permit the use of fallback authentication options.
Recommend using Cross-Device Authentication, with providing guidance provided for how to use it	In cases where RPs require authentication with a passkey to access services or features, users must utilize Cross-Device Authentication to avail themselves of these services or features.

Most RPs do not actively promote authentication via Cross-Device Authentication and do not mandate authentication with a passkey. This allows users who cannot use a passkey to authenticate with existing credentials. However, users may register an additional passkey to their account if it is supported in their environment.

Some RPs do recommend the use of Cross-Device Authentication for authentication because these RPs protect certain features or services by requiring authentication via a passkey. However, as stated previously, Cross-Device Authentication can be challenging for the average user. For instance, during autofill, a user must accomplish complex operations to initiate Cross-Device Authentication, which typically only those who are familiar with Cross-Device Authentication would know how to operate. Moreover, when an account is specified with *AllowCredential*, the behavior differs across operating systems. For example, iOS may display a QR code for Cross-Device Authentication if a passkey is not found, whereas Android may show an error.

Addressing the issue might be possible by guiding users on how to utilize Cross-Device Authentication. This guidance could include explanations of compatible devices/OS/versions, instructions for using the Google Lens app on Android versions 13 and below, along with screenshots demonstrating the authentication steps. However, even with such detailed guidance, there remains a possibility that average users may not fully grasp the information, particularly on what steps to take upon encountering a QR code.

5. Passkey Management

Passkey Management is a set of functions for users that allow them to view/edit/delete passkeys registered on a RP's site.

5.1 What passkey names are appropriate as a name to identify a passkey on a passkey management page?

Users can use the passkey names displayed on the passkey management page to distinguish multiple passkeys registered with their account. One possible use case of the passkey name is to allow users to identify suspicious passkeys. The passkey name can be displayed on "passkeys cards" and is recommended in the UX guidelines [1]. Note that the passkey name described in this section is different from *displayName*, as previously described in the section 3.2, which is displayed by the passkey provider and in autofill.

There are several deployment cases that reference the *passkey name* on the passkey management page.

Table 12 - Deployment examples: Passkey name appearance on passkey management page

Passkey name on a passkey management page	Possible reason(s)
Passkey provider name	As synced passkeys can be used on different browsers, displaying a name of a passkey provider allows users to distinguish the passkey by showing the RPs they are synced with.
Environment of the device used for passkey registration (e.g., device name, OS name, and browser name.)	Displaying information about the environment of the device used to register a passkey allows users to distinguish the passkey by showing the device they initially used to register the passkey.
Allowing user to modify	The passkey management page may fill in the passkey provider name, however users can modify the name so they can distinguish the passkeys, such as OS vendor, passkey provider, and device used for registration.

Several RPs display names of the passkey providers to users in order to distinguish registered synced passkeys. This practice makes it easier for users to recognize their passkeys, as each synced passkey is uniquely associated with an RPID and user id. This ensures that no two synced passkeys on an RP's account settings page will have the same passkey provider name. Furthermore, it simplifies the understanding of which devices or platforms can use the passkey. Specifically, RPs can utilize the AAGUID, a unique identifier for each passkey provider, to determine the provider's name and icon. It is important to note that not all passkey providers offer AAGUID, and since AAGUID is not attested, it is susceptible to tampering.

Most RPs display the environmental information of the device used at the time of passkey registration, such as the device, OS, and browser names, as the name of the passkey. This is suitable for device-bound passkeys since they can only be used on the device where they were created. Even RPs that utilize the passkey provider's name for synced passkeys employ the device name for device-bound passkeys. The prevalence of this method among most RPs is probably because use of AAGUID to identify passkey providers is a relatively recent development.

Some RPs allow users to rename their passkeys, facilitating easier identification even when the passkey provider cannot be determined by the RP. This feature enables users to assign names that are most meaningful to them. However, it may be challenging for the average user to manage this effectively. With physical security keys, naming is simple since the item to be named is tangible. This is covered in-depth in the FIDO Security Key UX Guidelines [6]. In contrast, for synced passkeys, users must have a certain level of understanding of the passkey mechanism to assign an appropriate name, which can be a complex task.

5.2 What accompanying information should be displayed on a passkey management page?

In addition to the passkey name mentioned previously, users have other information on the passkey management page to distinguish passkeys registered to their account. They may use this information to review and delete suspicious and/or unused passkeys. Such information can also be displayed on the “passkeys cards" as described in the UX guidelines [1].

Table 13 displays possible deployment cases regarding the information displayed on the passkey management page.

Table 13 - Deployment examples: Accompanying information displayed on the passkey management page

Information displayed	Possible reason(s)
Registration environment	If the registration date, OS, or OS version is not normal, a user can notice suspicious passkeys.
Last used date and time	If the last used date and time is not familiar to the user, it may be a signal that the passkey was compromised. Also, a user can recognize which passkey is no longer used.
Last used device name	If the device name is different from the usual device name used, it may be a signal that the passkey was compromised.
Synced or not synced	This allows a user to know whether or not the registered passkey is backed up
Passkey provider icon	This Allows a user to know where their passkey is stored

Displaying various accompanying information about a registered passkey may allow users to delete passkeys no longer in use and easily recognize and prevent suspicious/unused passkeys use.

5.3 What functions should be provided on a passkey management page?

Management functions on the passkey management page, such as deletion, are available for users to control their passkey registered with the RP. Most RPs provide a passkey deletion function on the management page, and some also provide a passkey name modification function.

Table 14 - Deployment examples: Reasons for deletion and name change on passkey management page

Function	Possible reason(s)
Deletion	A user selects the deletion function to delete suspiciously registered passkeys and/or unused passkeys.
Name modification	A user modifies registered passkey names on the RP's site to make it easier to distinguish from the other passkeys they may have.

The passkey deletion function on the passkey management page may improve security as it can prevent suspicious/unused passkeys use. This is also particularly relevant when users utilize methods such as Airdrop to share passkeys or when future functionalities for sharing passkeys between providers become available. In such cases, removing unused passkeys is more effectively managed on the RP's side rather than through the passkey provider. Note that a public key that is paired to a user's passkey (private key) can be deleted from the RP's page, but this doesn't remove the private passkey on the user devices. This is discussed in the following section in more detail.

5.4 How to handle an inconsistency between an RP and a passkey provider?

Given the inherent characteristics of passkeys, discrepancies can arise between the information held by the RP and that within the passkey provider or device. Such inconsistencies may occur, for example, when a passkey is deleted on the RP's side, and when there is a change in the email address that is used for the *user.name* during passkey registration.

A key on the RP's side that is paired to a user's passkey can be deleted from the RP's page, but this does not remove the private passkey on the user devices. There is no automatic synchronization between the RP server and the user's device and/or passkey provider. The user is expected to delete the passkey in their passkey provider manually themselves. For authentication, if the user attempts to use a passkey that was deleted on the RP server but remains on the user's device and/or passkey provider, authentication will typically fail. How to respond or guide the user in this situation is up to the RP's policy.

Similarly, if the user.name or *user.displayName* specified during passkey registration is editable on the RP's system, changing these values on the RP does not trigger an automatic update across to the user's device or passkey provider. Consequently, this leads to data inconsistencies between the RP server and the user's devices or passkey provider. There is currently no clear solution to this problem.

The best solution for an RP to remain consistent with a user is for them to prompt the user to perform the corresponding actions on their passkey provider that the RP has taken (i.e., if the RP deletes a passkey, they inform the user to delete that passkey in their passkey provider).

5.5 What recovery methods are provided for a user who cannot use their registered passkeys?

There may be cases where users cannot use a passkey account. Users may lose a passkey-enabled device or become locked out from a passkey provider, creating a need for an RP to provide alternative authentication methods.

With most RPs, a user can sign in and add new passkeys by signing in with alternative authentication methods, such as other passkeys, password + SMS OTP or magic link via email, if one of the user's passkeys is unable to be used.

In some RPs, an account recovery process based on robust identity proofing, such as electronic Know Your Customer (eKYC), is necessary. This is particularly true for RPs that require passkey authentication for certain features or services. For these RPs, if users lose access to their passkeys, they cannot authenticate using alternative methods. Therefore, a solid account recovery mechanism is critical, otherwise, the security intended by passkey protection is vulnerable, and the recovery process itself could become a weak point that allows attackers to compromise accounts.

"Recommended Account Recovery Practices for FIDO Relying Parties" [7] provides good insights about account recovery in cases where authenticators are unavailable, lost, or broken.

6. Passkey Accessibility

This section shares findings related to passkey accessibility based on screen reader audits of a limited group of live passkey deployments. This is for informational purposes only. RPs are ultimately responsible for ensuring the passkey deployment is accessible in accordance with the Web Content Accessibility Guidelines (WCAG) [8], the accepted global standards developed by the World Wide Web Consortium (W3C).

6.1 Scope

Screen reader audits of live passkey deployments with the following OS/Browser/Screen reader combinations:

- Windows/Chrome/Jaws
- Windows/Chrome/NVDA
- Windows/Edge/Jaws
- Windows/Edge/NVDA
- Mac/Safari/VoiceOver
- Mac/Chrome/VoiceOver
- iPhone/Chrome/VoiceOver
- iPhone/Safari/VoiceOver

6.2 Key findings

1. Passkey registration and sign-in ceremonies were consistently accessible.

   Using various OS/browser/screen reader combinations with the creation and use of passkeys improved the experience for a blind consumer. The ceremony was largely managed by the platform.

2. Accessibility defects were common before and after the passkey ceremony.

   Audits across several live passkey deployments found that accessibility defects were common in the RP's site as a whole. This is consistent with industry findings on the state of digital accessibility as reported by WebAIM's 2023 Accessibility Report of the Top 1,000,000 Homepages [9].

3. Auto-complete and QR codes introduce accessibility barriers.

   Auto-complete was inconsistently implemented across the live deployments tested yielding inconsistent experiences for screen reader users. The implementation is required to support screen readers by announcing "has popup" to alert the screen reader user that content is available to navigate using the down arrow key. Some sites used the basic HTML auto-complete attribute on a HTML input field and others used a custom input field with ARIA (Accessible Rich Internet Applications Suite), with specialized attributes added to code to make user interactions accessible for assistive technologies such as screen readers, which often is implemented incorrectly. Many browsers now support the basic HTML auto-complete attribute announcing "has popup" on entry into the input field eliminating the need for additional ARIA.

   QR codes introduce barriers for individuals with mobility limitations or with vision limitations. See white paper, Guidance for Making FIDO Deployments Accessible for Users with Disabilities [10] for more information.

Refer to video presentation [11], Foundation for Passkey Accessibility for an overview of the accessibility findings including a screen reader demonstration on a live passkey deployment.

7. Conclusions

This paper has summarized the rising trends and early emerging practices in the area of synced passkeys for RPs. These practices are at the forefront, with the prospect of maturing into established guidelines based on cumulative experience and the evolving ecosystem of passkey providers. RPs are advised to remain agile and continuously reflect on these practices, taking into account the potential shifts in the passkey landscape. This paper serves as an initial guidepost, recognizing the nature of passkey security and the importance of regularly updating and refining deployment strategies.

8. Acknowledgments

The authors acknowledge the following people (in alphabetic order) for their valuable feedback and comments:

- FIDO Japan Working Group volunteers
- John Bradley, Yubico
- Kevin Goldman, Trusona
- Max Hata, NTT Docomo
- Rakan Khalid, Intuit
- Sue Koomen, Amex
- Matthew Miller, Cisco
- Dean H. Saxe, Amazon Web Services
- Joyce Oshita, VMware
- Johannes Stockman, Okta
- Shane Weeden, IBM
- Khaled Zaky, Amazon Web Services

9. Glossary of Terms

Term	Definition
Account Recovery	Conducting identity proofing again to regain ownership of an account, and its associated information and privileges.
Credential Recovery	The process of resetting a credential (e.g., a password) from data that has been stored in or transmitted by a computer system.
Passkey Provider	A service that helps internet users create, save, manage, and use passkeys across different online services.
Restore	The process of returning a key credential for a synced passkey from a passkey provider to a user device.

10. References

[1] FIDO Alliance. “FIDO Alliance UX Guidelines for Passkey Creation and Sign-ins” FIDO Alliance. https://fidoalliance.org/ux-guidelines-for-passkey-creation-and-sign-ins/.

[2] FIDO Alliance. “Apple, Google and Microsoft Commit to Expanded Support for FIDO Standard to Accelerate Availability of Passwordless Sign-Ins.“ FIDO Alliance. May 2022. https://fidoalliance.org/apple-google-and-microsoft-commit-to-expanded-support-for-fido-standard-to-accelerate-availability-of-passwordless-sign-ins/.

[3] Passkeys.dev. “Device Support.” Passkeys.dev. https://passkeys.dev/device-support/.

[4] 1Password. “Passkeys.directory.” 1Password. https://passkeys.directory/.

[5] H. Saxe, Dean. “FIDO Deploying Passkeys in the Enterprise - Introduction.” FIDO Alliance. June 2023. https://fidoalliance.org/wp-content/uploads/2023/06/June-26-FIDO-EDWG-Spring-2023_Paper-1_Introduction-FINAL.docx.pdf.

[6] FIDO Alliance. “FIDO Security Key UX Guidelines” FIDO Alliance. https://fidoalliance.org/ux-guidelines/security-key-ux-guidelines/.

[7] Gomi, Hidehito, Leddy, Bill, and Saxe H., Dean. “Recommended Account Recovery Practices for FIDO Relying Parties.” FIDO Alliance. February 2019. https://fidoalliance.org/wp-content/uploads/2019/02/FIDO_Account_Recovery_Best_Practices-1.pdf.

[8] Henry, Shawn. “WCAG 2 Overview.” W3C Web Accessibility Initiative. October 2023. https://www.w3.org/WAI/standards-guidelines/wcag/.

[9] WebAIM. “The WebAIM Million The 2023 report on the accessibility of the top 1,000,000 home pages.” Institute for Disability Research, Policy, and Practice, Utah State University. March 2023. https://webaim.org/projects/million/.

[10] FIDO Alliance. “Guidance for Making FIDO Deployments Accessible to Users with Disabilities.” FIDO Alliance. August 2022. https://fidoalliance.org/wp-content/uploads/2022/10/Guidance-for-Making-FIDO-Deployments-Accessible-to-Users-with-Disabilities_FINAL.pdf.

[11] FIDO Alliance. “Foundations for Passkey Accessibility.” Video FIDO Alliance. 2023.