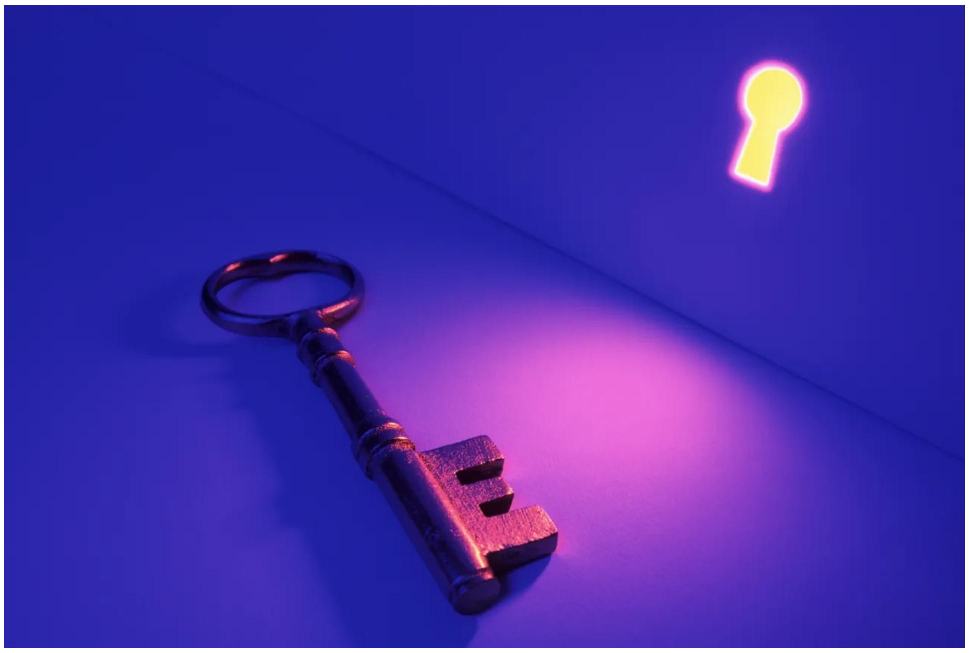
How passkeys work: The complete guide to your inevitable passwordless future

Why are passkeys so much safer than passwords? And how exactly does this sorcery work? We go behind the scenes of this still-evolving authentication process.

Written by David Berlind, Senior Contributing Editor
July 8, 2025 at 7:00 p.m. PT
Reviewed by David Grober
ZDNet



I've been writing a lot about passkeys recently -- and with good reason. This year, some of the world's largest technology companies are doubling down on efforts to convince their billions of global users to start using passkeys instead of passwords when signing into websites, apps, and other services.

Passwords versus passkeys

how passkeys work

Do your favorite sites even support passkeys?

Passkeys are often described as a passwordless technology. In order for passwords to work as a part of the authentication process, the website, app, or other service -- collectively referred to as the "relying party" -- must keep a record of that password in its end-user identity management system. This way, when you submit your password at login time, the relying party can check to see if the password you provided matches the one it has on record for you.

The process is the same, whether or not the password on record is encrypted. In other words, with passwords, before you can establish a login, you must first share your secret with the relying party. From that point forward, every time you go to login, you must send your secret to the relying party again. In the world of cybersecurity, passwords are considered shared secrets, and no matter who you share your secret with, shared secrets are considered risky.

Also: Biometrics vs. passcodes: What lawyers say if you're worried about warrantless phone searches

Many of the largest and most damaging data breaches in history might not have happened had a malicious actor not discovered a shared password.

In contrast, passkeys also involve a secret, but that secret is never shared with a relying party. Passkeys are a form of Zero Knowledge Authentication (ZKA). The relying party has zero knowledge of your secret, and in order to sign in to a relying party, all you have to do is prove to the relying party that you have the secret in your possession.

Here's the big idea behind passkeys: If you never have to share your secret with a legitimate relying party, then you'll never accidentally share your secret with a malicious actor like a phisher or smisher, either. But humans are so programmed to think that we need to share secret passwords that it's difficult for us to fathom how it could work any other way.

So, how exactly is this possible? After all, it seems counterintuitive. And why is it so much safer than passwords?

Public key cryptography is the foundation of passkeys

To understand how passkeys work, it's important to cover some basics of public key cryptography. In the same way that a physical key is matched to a specific lock, public key cryptography involves two digital keys -- a public key and a private key -- that are matched and unique to one another. A set of these uniquely matched keys is called a public/private key pair; behind every unique passkey exists one of these unique pairs.

What's so special about a public/private key pair? Let's say I give you the public key that matches a private (and secret) key in my possession. You can use that public key to scramble (encrypt) a message. When you send that message to me, as long as I'm the only one who has the matching private key, I'm the only one who can unscramble (decrypt) that message.

Also: If we want a passwordless future, let's get our passkey story straight

The public key cannot be used to decrypt that message or derive the private key. The existence of public key cryptography technology means that I can give away my public key to as many people as I want, and they can all use that same key to send me encrypted messages. Meanwhile, I am the only one who can decrypt those messages because I am the only one in possession of the secret private key.

Also, I can use my private key to digitally sign any messages that I might send to the holders of the matching public key. This way, the public key-holding recipients of any such messages can use their public key to validate that the message came from someone in possession of the matching secret private key. This capability plays a special role when it comes to the ZKA aspect of passkeys: convincing a relying party that I have the secret part of the passkey -- the private key -- without having to divulge the secret to them.

Unlike with passkeys, you must risk sharing your secret password with a relying party (as described above) before that password will work as a login credential with that relying party; with passkeys, you never take that risk.

Four passkey workflows everyone should know

For most users, there are four passkey-related processes (workflows) to be aware of.

- Discovery and engagement with the relying party's passkey functionality -- Among those relying parties that support passkeys, there's no standard for how their passkey functionality is advertised or accessed by the user. From one relying party to the next, not only might the user experience be different, but they may also use a different vernacular to refer to the same thing.

- The passkey registration ceremony -- For any given relying party, you establish one or more passkeys that you're going to use in order to sign in to a website or app.

- The passkey-based authentication ceremony -- Once a passkey has been enrolled with a relying party, this is how you use it to authenticate with that relying party.

- The passkey deletion process -- Cleaning up unused or unwanted passkeys.

Also: Your password manager is under attack: How to defend yourself against a new threat

Passkeys depend on standards and hidden complexity

The complexity of these workflows matters. Where the public and private keys come into play, they are significantly embellished by two industry standards that, taken together, form the foundation on which passkeys work. One of these standards -- known as WebAuthn -- comes from the World Wide Web Consortium (W3C) and is an industry standard for web-based passwordless authentication, passkey or not. In other words, passkeys are WebAuthn-compliant credentials. Much of a passkey's unique value proposition as a more secure credential than a username-password combo is rooted in its compliance with the passwordless WebAuthn standard.

However, as complete as the WebAuthn specification is, the innovation of the passkey required the support of an additional foundational standard from another consortium known as the FIDO Alliance. FIDO stands for Fast ID Online, and the need for that additional standard is largely rooted in the need for passkeys to be a friction-free form of authentication involving little more user interaction than the provision of a PIN or biometric (something that most users are already familiar with).

Also: How passkeys work: What really happens during a passwordless login?

After all, we already authenticate to a lot of applications and devices with nothing more than our fingerprint. It stands to reason that we should be able to do the same with any website or app. Moreover, both the process and user experience -- from one relying party to the next -- should be as standard and recognizable as the legacy standard of using usernames and passwords.

To meet that need, the FIDO Alliance published an additional specification called the client-to-authenticator protocol (CTAP). Just the phrase "client to authenticator" is loaded with some of the complexity that most passkey implementations attempt to hide from users (sometimes not so successfully). As you'll learn in this series, a typical passkey-based authentication -- officially referred to as an "authentication ceremony" -- involves a sequence of hand-offs from one entity to another, each one often under the control of a different vendor and each one an opportunity for user confusion or worse, a workflow's failure.

Understanding passkey terminology

As CTAP's meaning suggests, in addition to the aforementioned relying party, two other entities are involved in all passkey workflows: the client and the authenticator. The client, in most circumstances, is the piece of technology (i.e., a web browser) in the end-user's device (i.e., computer or smartphone) that handles incoming and outgoing web traffic. As a handler of your system's web traffic, the client is essentially an intermediary between the relying party and the authenticator. It receives inbound WebAuthn-compliant web-based authentication requests from the relying party, passes them on to an authenticator, and then relays the authenticator's response back to the relying party.

Also: The best password managers: Expert tested

In a WebAuthn and passkey context, an authenticator is a separate piece of technology (also in the user's possession) that's not necessarily the type of authenticator you're likely familiar with -- ones like Google Authenticator or Symantec VIP that are devoted to the generation of one-time passcodes. Authenticators in a WebAuthn and passkey context can handle public key cryptography tasks such as creating public/private key pairs and using a private key to digitally sign or encrypt messages before they are sent to a relying party.

Authenticators come in different forms, and, as discussed in my 10 passkey survival tips, deciding which one to use requires a bit of forward thinking and planning. In addition to their role as credential managers that store and synchronize credentials of all types (user ID/passwords, passkeys, etc.) across all of your devices, password managers such as 1Password, Bitwarden, Dashlane, and LastPass can serve in a secondary role as authenticators (the W3C refers to these third-party authenticators as "virtual authenticators"). Similarly, the platforms and browsers we use -- such as Windows, MacOS, Chrome, and Firefox -- have their own password management and authenticator capabilities (known as "platform authenticators"). And for those who prefer an authenticator in the form of dedicated hardware, you have the option of a roaming authenticator such as one of Yubico's Yubikeys or Google's Titan.

how passkeys work

Let's start the passkey registration process

Your passkey journey can be a strange and inconsistent ordeal. But it doesn't have to be this way. Read now

Given the variety of choices for both clients and authenticators, a standard like CTAP was required to normalize the integration between them as best as possible.

Passkeys are, therefore, both a WebAuthn and CTAP-compliant passwordless credential, or what the FIDO Alliance refers to as a FIDO2-compliant credential. As you can see, the lingo alone can be confusing. A passkey is sometimes called a FIDO2 credential and vice versa. You have password managers that are actually credential managers because not every supported credential type (e.g., a passkey) involves a password. But at the same time, they're also referred to as authenticators, and these authenticators are not necessarily the authenticators you're used to working with. But for these authenticators, there are different types -- platform, virtual, and roaming -- but not all of them are credential managers. Meanwhile, most credential managers are authenticators too.

Is your head spinning yet? Mastering the vocabulary, never mind how to work the technology, is intimidating to say the least.

One of the goals of this six-part ZDNET series on how passkeys work is to make passkeys less intimidating. I'm going to cover the four major workflows in detail, demonstrating what users will typically see as they go through each workflow while also revealing the complexity of what's taking place behind the scenes each time the user clicks or taps on a link or button.

Also: The best security keys: Expert tested

Sussing out the most important passkey concepts

While it's impossible to achieve this goal without going into some of the gory technical details, this series attempts to strike a balance between a high-level primer and a low-level discussion of every configuration, outcome, and parameter that's available to each of the workflows. As such, I focus on the most important of the passkey concepts and implementations, the ones that make a convincing story about the superiority of passkeys over traditional credentials. In doing so, I will gloss over -- or altogether ignore -- some of the finer points (even though they're important too) while taking certain liberties with the vocabulary.

For example, when I say credential manager, I usually mean the password manager performing in its credential manager role. But when I say authenticator, I also mean the password manager, but in its authenticator role. You heard about the client in this introduction to the series, but you won't see that term again. Going forward, I'll go with "browser" to make certain points. Even though the phrase "relying party" technically refers to the party that's the operator of the sites, apps, and other services that support passkeys, it is frequently used interchangeably with words like "site" and "app" to keep the reader focused on the four major entities involved in most passkey workflows.

Speaking of apps, this series would have been twice as long if I had given specific lip service to the mobile passkey user experiences on iOS and Android. I skipped them because the basic workflows and concepts are the same. However, I plan to revisit the mobile use case in future articles because some mobile app adaptations to passkeys are better than others.

With apologies to the people who developed the WebAuthn and CTAP standards, key terminology -- exact field names and conceptually important words like "nonce" and "attestation" -- are abstracted or bypassed altogether so as not to distract readers from the higher-level concepts being explained.

Finally, for demonstration purposes, I use the same four technologies across this entire series:

- Shopify.com for the relying party
- Google Chrome for the browser
- Bitwarden's password management extension (for Chrome) for the authenticator/credential manager
- MacOS for the operating system

I could have just as easily picked Paypal.com, Firefox, NordPass, and Windows 11. There would have been subtle differences in the user experience, but they would have been mostly immaterial -- one of the reasons why passkeys are a better credential than anything to come before them.

Stay ahead of security news with Tech Today, delivered to your inbox every morning.

In the next installment -- Do your favorite sites even support passkeys? -- I use Shopify's website as an example of how to discover if a relying party even supports passkeys -- many do not.

How passkeys work: Overview | Discovery | Selection | Registration | Authentication | Deletion

original article: https://www.zdnet.com/article/how-passkeys-work-the-complete-guide-to-your-inevitable-passwordless-future/?utm_source=iterable&utm_medium=email&utm_campaign=techtoday&zdee=[Contact.email_zdee]