Passkeys Demystified

Macstock 9

Kirschen Seah 2025-07-05

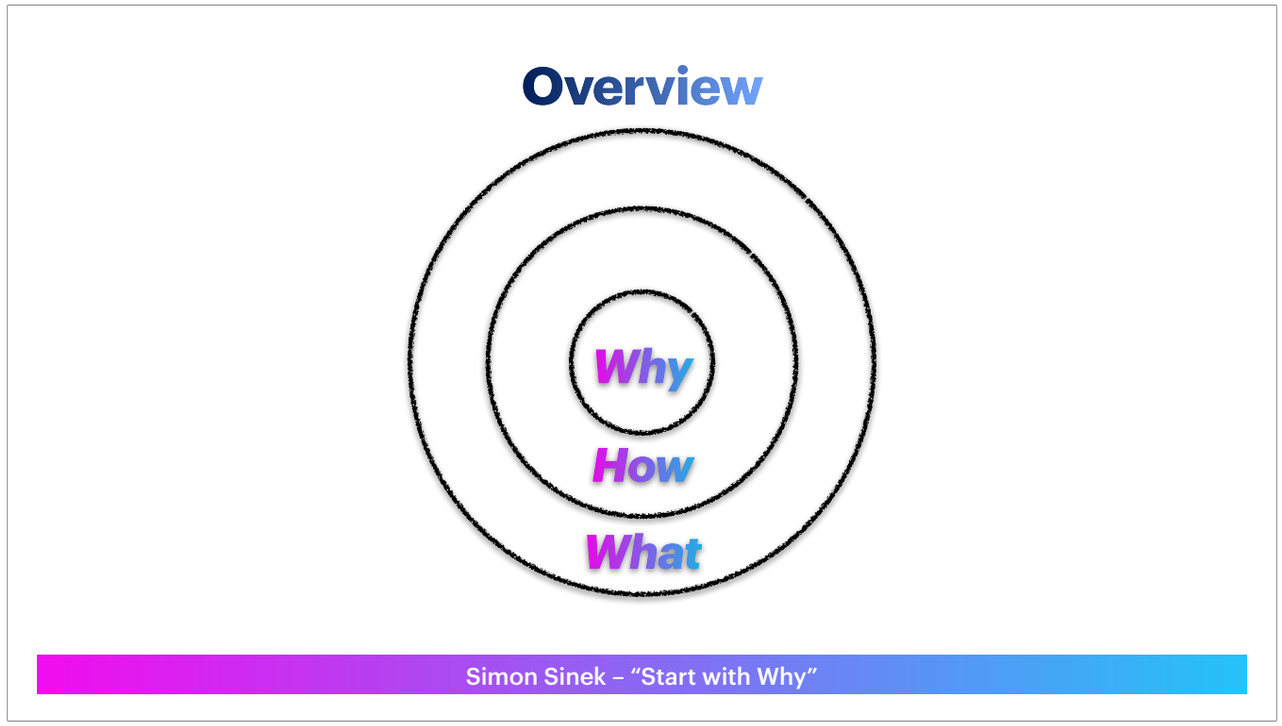
About Kirschen...

Software geekette

Apple aficionado since 1978

Avid bicyclist / Pilot / Retired 😃

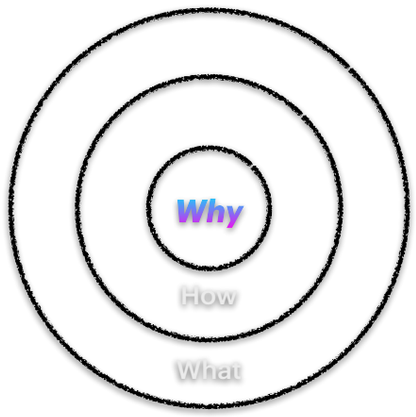
https://www.freerangecoder.com



Simon Sinek - posited The Golden Circle - “Start with Why”
Studied people and companies to distill their attributes for success - including Apple

WHY - Motivation: purpose, cause or belief
* (Allison Sheridan) - the problems we'd like to solve
HOW - Achieve: processes & methods
WHAT - Consequence: results or outcomes

Why



Password management

Many usernames / passwords

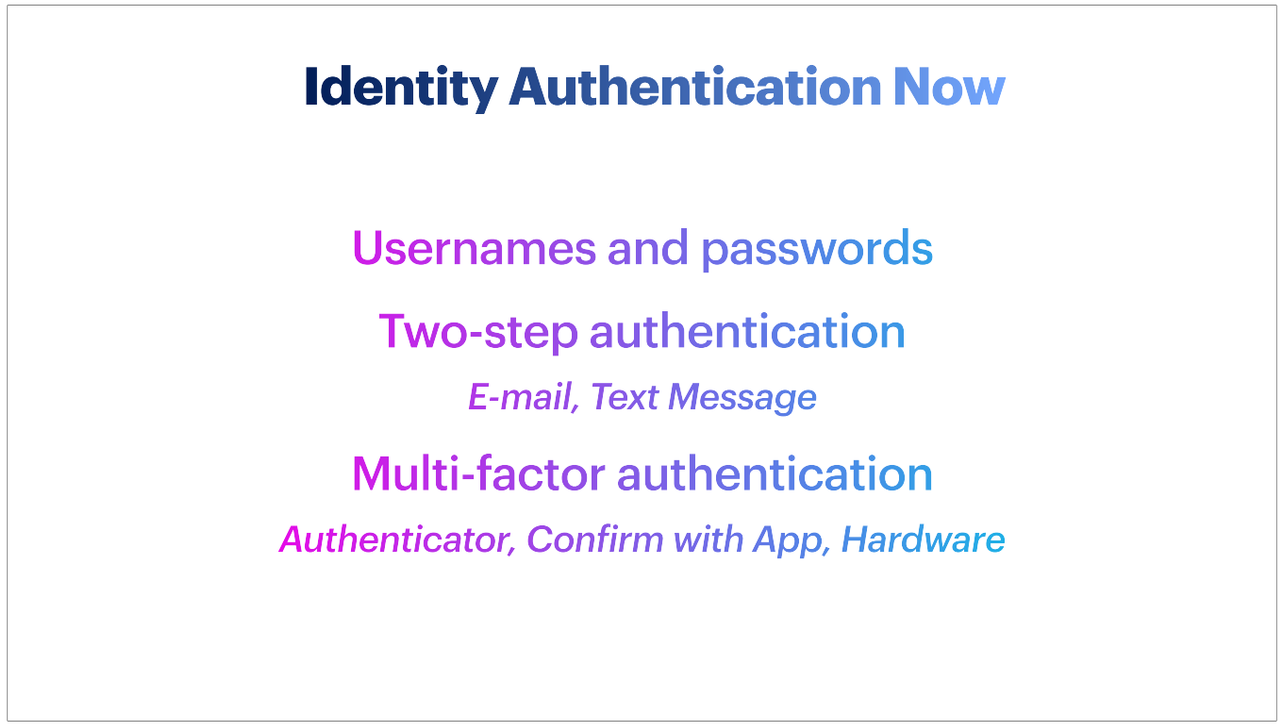
Security issues

Hacking, breaches, phishing

Complicated, especially for non-technically inclined users
* Password re-use, common passwords

Need apps to help manage passwords - e.g. Apple Passwords & 1Password
These apps also manage passkeys

Security issues
* Hacking
* Security breaches
* Social engineering, e.g. Phishing



Remind ourselves - how user authentication is implemented now

Usernames - usually e-mail addresses (pitfall: easily found out)

Passwords - have to meet length and character content criteria (issue: coming up with a strong password)

Two-step authentication
* E-mail - site sends you an e-mail with a link to click
    * Problem if e-mail credentials are compromised
* Text message verification
    * SIM cloning

Multi-factor authentication - might have different devices or apps for different sites. Most sites use the TOTP (Time-based one-time password) standard. Corporate use - hardware device (e.g. Duo, RSA SecurID, or YubiKey)

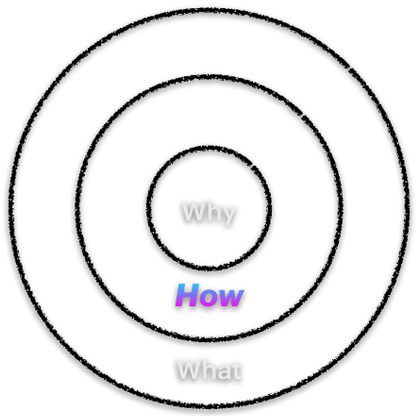
Passkeys

A **secure** and **frictionless** way to **authenticate your identity** to a **web site or app**

Passkeys offer **passwordless login**

What do we mean by
* Secure - prevents security issues (breaches, hacking, insecure passwords) from being a problem, by design
* Frictionless - requires minimal user interaction
* Authenticate your identity - so that user can log in securely to a service
* Web site or app - usual applications where identity authentication is needed

How



Web authentication

Public Key Cryptography

Web authentication standard, WebAuthn published 2019
FIDO - Fast IDentity Online
and
W3C (World Wide Web consortium)

Leverages Public Key Cryptography

How – Passkeys

Web authentication

WebAuthn Standard

FIDO2 Project – FIDO & W3C

Wikipedia: WebAuthn

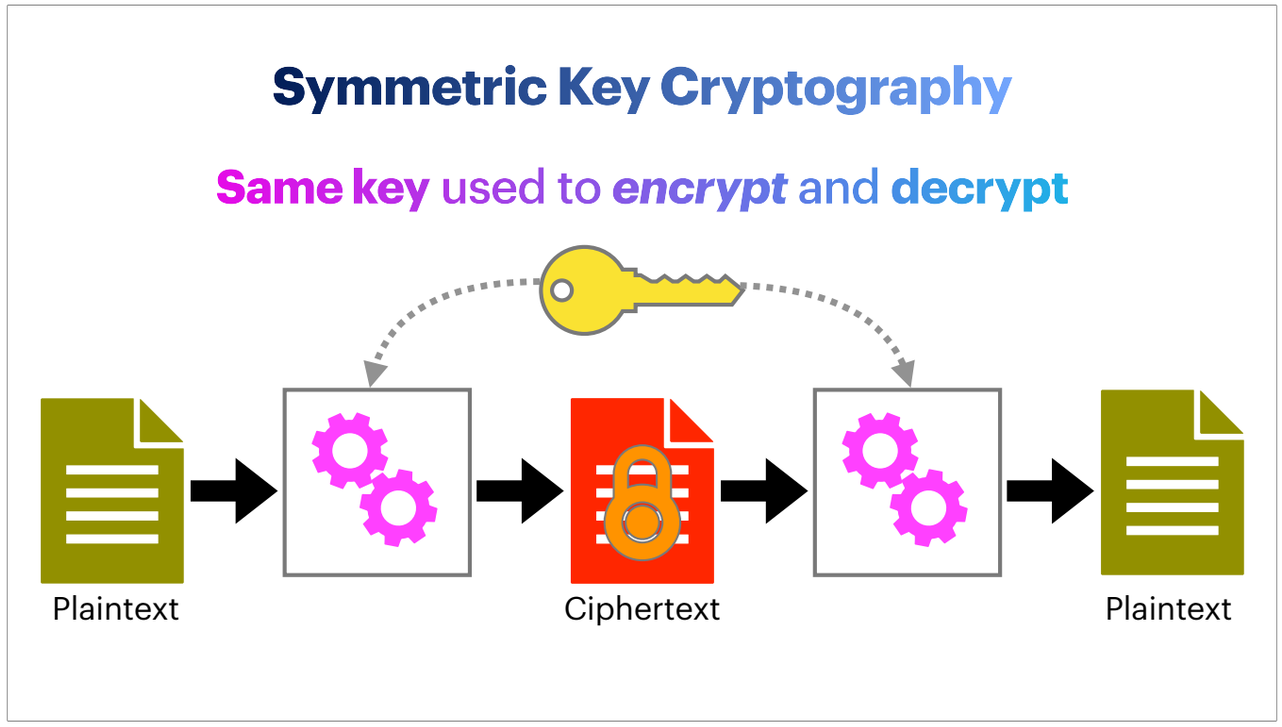
FIDO - Fast Identity Online
W3C - World Wide Web Consortium

How – Passkeys

Public Key Cryptography

Wikipedia: Public-key cryptography

Passkeys rely on a method of encryption/decryption which uses two keys - Public Key Cryptography (PKC).
Before I explain PKC, let’s look at the previous method which uses one key…



Prior to Public Key Cryptosystems

Key - e.g. password

Pros
* Only one key needed
* Fast to encrypt and decrypt

Examples of use - encrypted ZIP / PDF / DMG files

Issues
* Common key is compromised
* How do we share the common key

Public Key Cryptography (PKC)

Asymmetric cryptography

Pair of keys – **private** and **public**

Easy to generate key pair

*Difficult to compute **private key** from **public key***

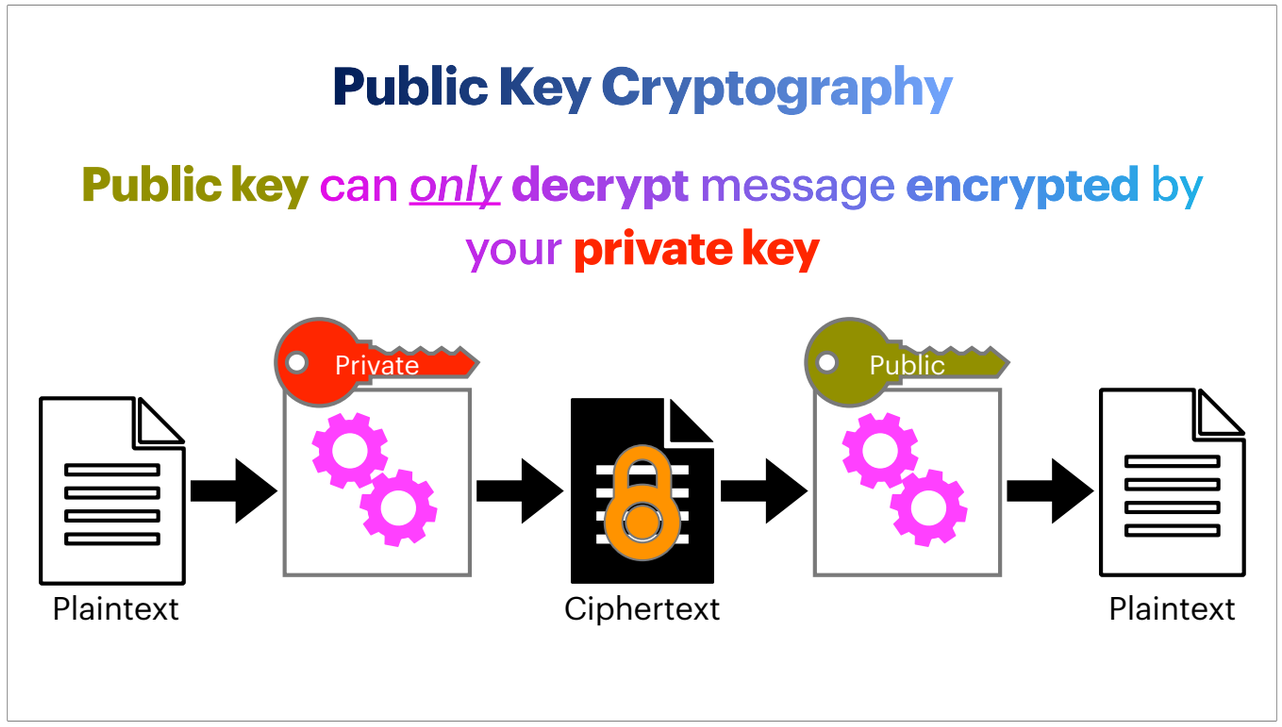
First example was RSA (Rivest / Shamir / Adelman) cryptosystem invented in 1977

Keys are generated in pairs and are very large prime numbers

Private key is kept by user and never revealed
Public key is provided to the public

We'll refer to Public key cryptography as PKC (a lot shorter!)

Note: "Difficult to determine private key from public key" - with existing computer technology, however quantum computers may be able to break current cryptographic algorithms. There are algorithms being developed which are considered "quantum-resistant / safe / proof".

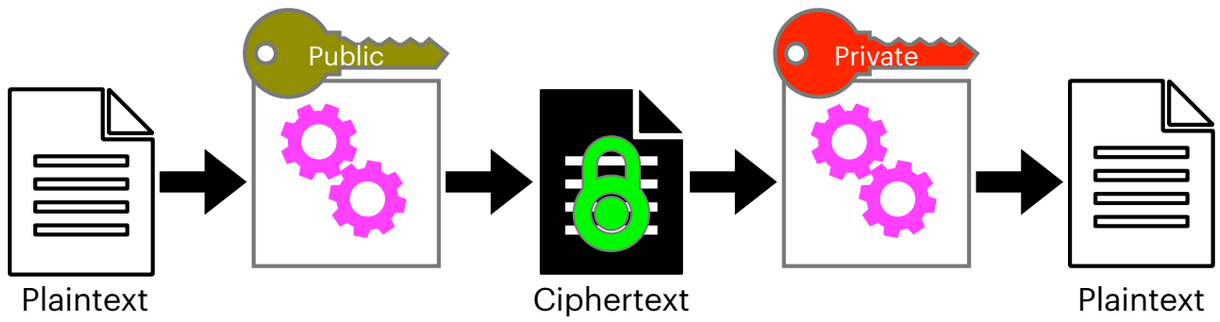


Most important feature of public key cryptography relevant to Passkeys - asymmetric encryption / decryption
Plaintext is the original content
Ciphertext is the encrypted content

Any other Public Key which has not been generated as a pair with the Private Key cannot decrypt the message encrypted by the Private Key.

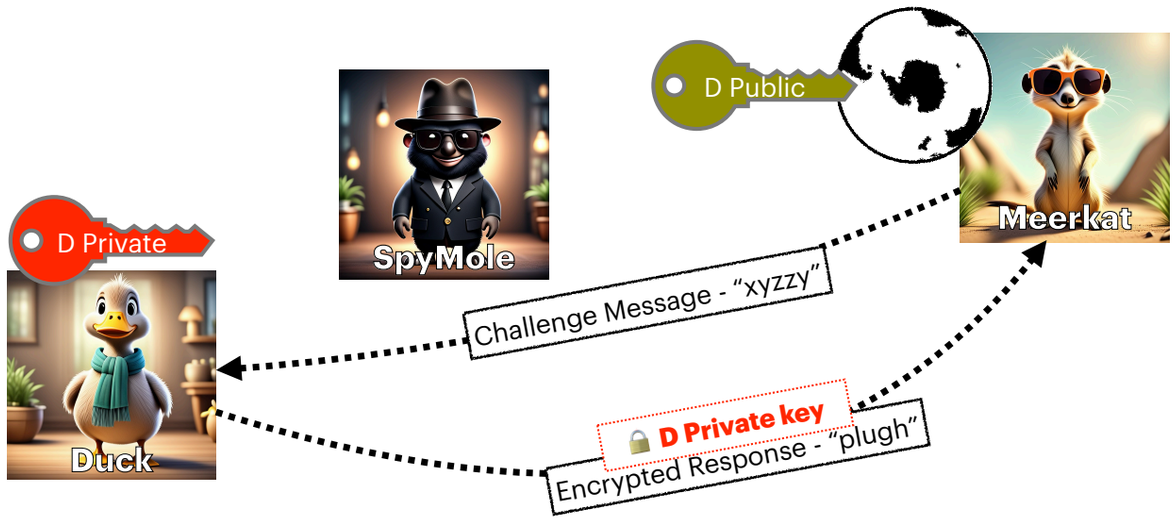
Public Key Cryptography

A message ***encrypted*** by **public key** can be **decrypted** *only* by your **private key**



Ciphertext in this case is different from the one encrypted by private key

Using PKC for Authentication



User authentication is also referred to as a digital signature. This is how Passkeys work.

1. Duck wants to access Meerkat’s site
2. Meerkat’s site sends a challenge message (e.g. “xyzzy”) to Duck - can be sent with no encryption or “in the clear”
3. Duck checks that Meerkat’s site is legitimate (domain, IP address) and corresponds to the site associated with their Private key
4. Duck encrypts the challenge message with their Private key and sends the encrypted response (e.g. “plugh”) back to Meerkat’s site
5. Meerkat’s site then decrypts the encrypted message with Duck’s Public key and compares the result to the challenge message. If the decrypted message and the challenge message match then Duck can be allowed access to Meerkat’s site, Duck has proved their identity.

Why?
Only Duck can encrypt the message with their Private key, and since that encrypted message can only be properly decrypted with their Public key, Meerkat’s site can then verify Duck’s identity is authentic.

If a malicious third party (SpyMole) listens in and tries to send an encrypted response to masquerade as Duck, then Meerkat’s site would see that the result of decrypting SpyMole’s encrypted response with Duck’s public key does not match the challenge message.

Note:
* There has to be a means to verifying that Public Keys are tied to a user / entity
    * Public Key Infrastructure (PKI) - verified repository of public keys, in the general case
    * For Passkeys - site receives the Public key for the authenticated user
* Only considering user authentication here, see Extra Credit for one means of secure two way communications
* After authentication - set up secure two way communications

So... How does it all work?

Websites

rely on

Browsers

interact with

Operating System

provide user authentication via Credential Managers

WebAuthn standard specifies:

* WebAuthn Relying Party - website which uses JavaScript functions for Passkeys implemented by Browser
* WebAuthn Client - Browser: provides JavaScript functions
* WebAuthn / FIDO2 authenticator
    * OS / Credential Manager - generates and manages Passkeys, handles authentication

Credential Managers

Native

Apple Passwords, Windows Credential Manager

Third Party

1Password, Bitwarden, Dashlane, NordPass

6 Best Password Managers (2025), Tested and Reviewed | WIRED

Credential managers
* More than password managers
* Keep track of passwords / Passkeys / authentication tokens etc
* Can be native (Apple Passwords, Windows Credential Manager) or 3rd party (1Password)

Need to have a Credential Manager to use Passkeys

Features
* Cross Platform
* Passkey support
* Free tier & capabilities

Practical Passkey Usage

Initial Setup

Subsequent Access

Practical Usage

Initial Setup

New account for site

Set up account (via e-mail)

Site prompts for Passkey

Existing account

Log in to account

Add Passkey to account

Initial setup also called “registration”

Key thing is initial user authentication
Most sites assume that the user’s e-mail is under the control of the user (i.e. e-mail access is not compromised - choose strong e-mail password!)

New account
* User enters e-mail address
* Site sends e-mail with link to confirm that user controls the e-mail address
* Site *may* ask for password in addition to Passkey (useful, see later)s
* Site prompts for Passkey creation

Existing account
* User logs in using existing means (user id / e-mail and password)
* Site prompts for Passkey creation
* Some credential providers such as 1Password prompt for Passkey creation if the site is known to support Passkeys

Practical Usage

Initial Setup

OS ⇒ Credential Manager

Create Passkey

*Generate **Private** & **Public** Keys*

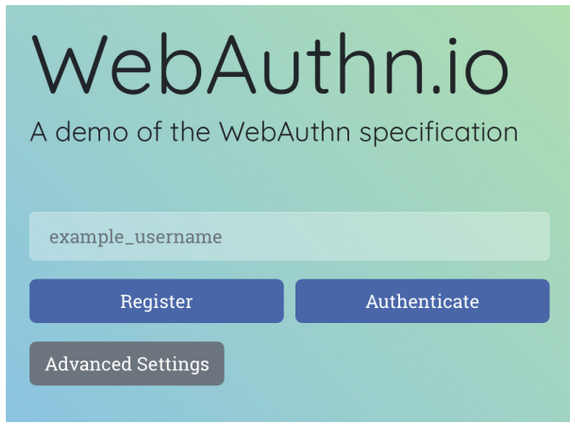
*Store generated **Private Key** & website details*

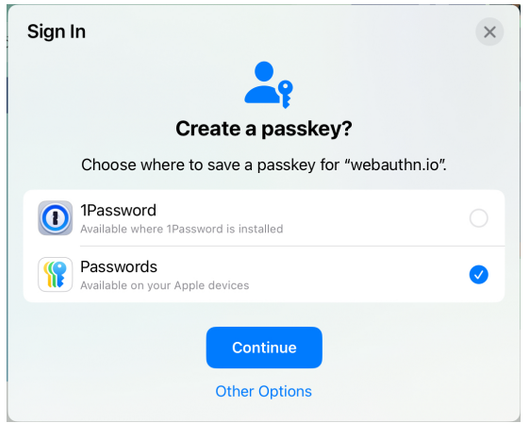
Public Key sent to Site

Passkey's Private Key and Public Key are generated by Credential Manager (acting as "WebAuthn Authenticator")
Credential Manager stores generated Passkey's Private Key & website details (domain, address, name etc)
Passkey's Public Key is sent over to Site

Practical Passkey Usage

Initial Setup





iPadOS 18 / webauthn.io

Slightly different for macOS due to how Autofill & Passwords are integrated.
[Action]
Use drawing mode to highlight buttons, fields, and outputs

Practical Passkey Usage

Subsequent Access

User enters User ID & invokes authenticate via Passkey

Automatic "Use Passkey?" alert

Site **initiates** Passkey authentication

Sends challenge message

Look mum, no passwords!

Sometimes sites automatically trigger "Use Passkey?" alert when Credential Manager detects a Passkey is available
* First step of user ID entry may not be needed if website requests Passkey and WebAuthn authenticator / Credential Manager detects a Passkey exists for that website.

Site calls JavaScript functions for Passkey authentication implemented by browser

Practical Passkey Usage

Subsequent Access

Credential Manager checks **website** address is **valid**

Credential Manager retrieves **Private Key** and **encrypts challenge message**

Browser interacts with Credential Manager via Operating System
* provides challenge message

Credential Manager checks that the request has come from the website associated with the Passkey (part of WebAuthn standard)
Important because a malicious site could impersonate the actual site at a different domain address and start a Passkey authentication

Credential Manager displays choice of Passkeys to use if multiple exist

Authentication for the Credential Manager might need password or biometric to unlock
* Biometric and password are used locally on device

Practical Passkey Usage

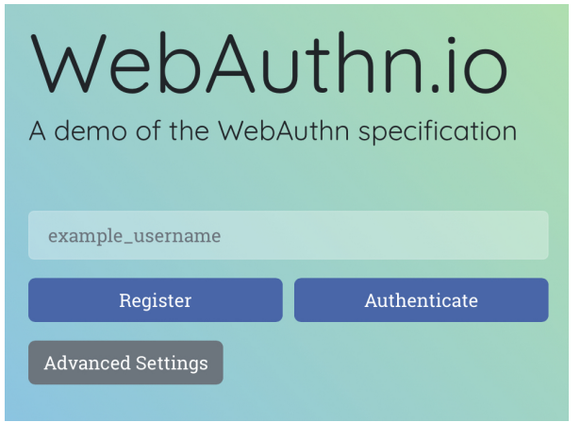
Subsequent Access

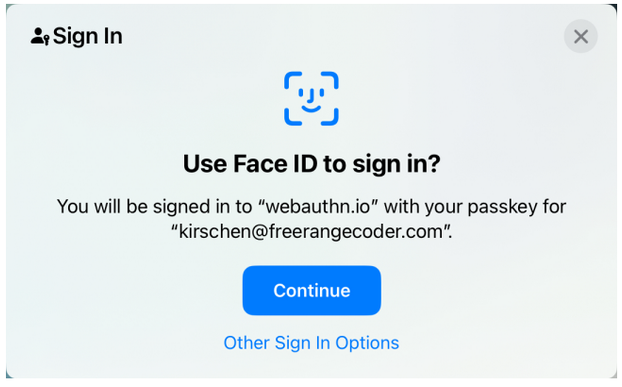
Browser sends **encrypted response message** to Site

Site **decrypts** response and **verifies** challenge message

Practical Passkey Usage

Subsequent Usage





iPadOS 18 / webauthn.io

Slightly different for macOS due to how Autofill & Passwords are integrated.
[Action]
Use drawing mode to highlight buttons, fields, and outputs

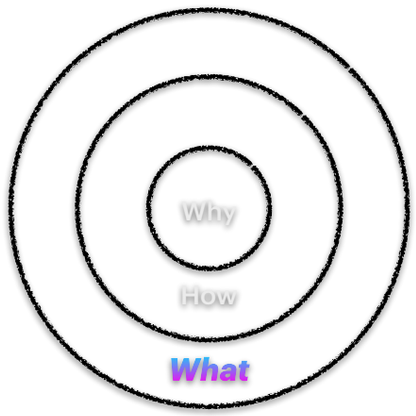
Demo

Passkey Initial Setup

Subsequent Access

https://webauthn.io

What

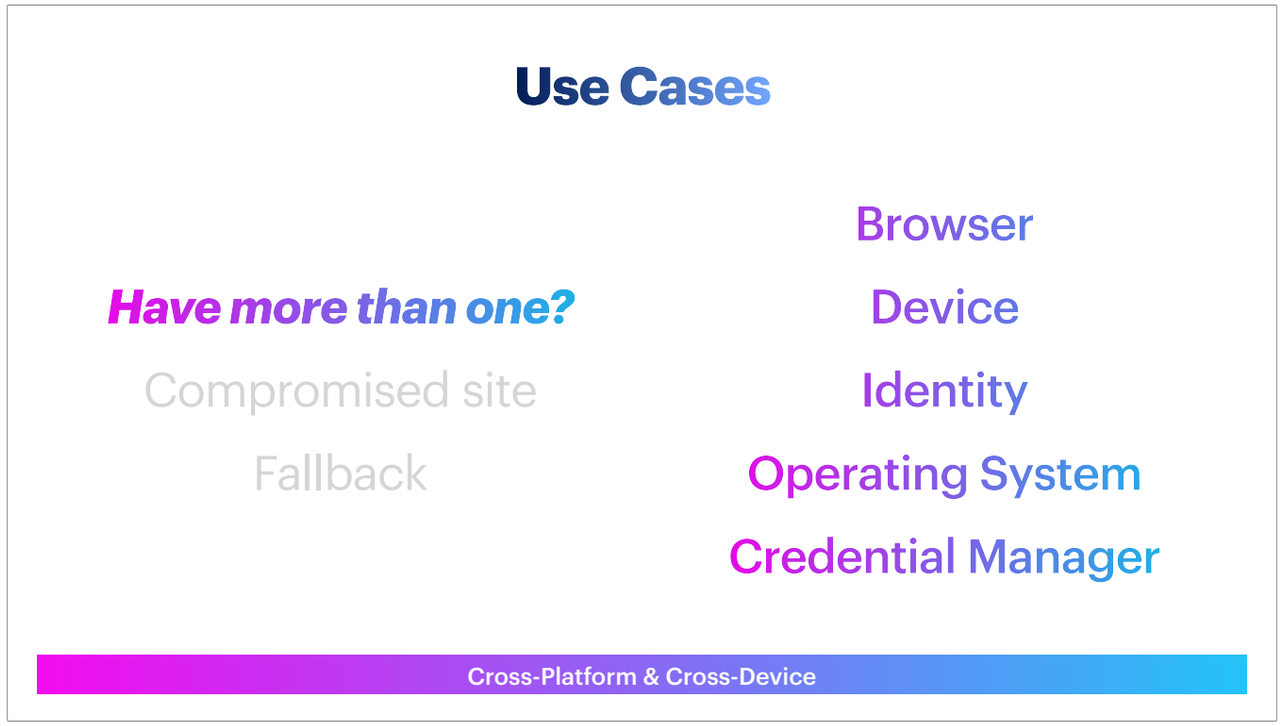


Use Cases

Passkey Support

References

Passkey Support - sites, apps, browsers



Oh! The choices!!

Non-native browsers
* Prompt for access to Passkey (macOS)
* macOS: System Settings > Privacy & Security > Passkeys Access for Web Browsers

Credential Manager - check for cross-device, cross-platform

Passkeys work for multiple identities
* supported by Passwords and 1Password
* just name them differently

Can use multiple Credential Managers concurrently

Use Cases

Have more than one?

Compromised site

Fallback

Delete Passkey from Site

Delete Private Key from Credential Manager

Re-create Passkey

Site compromise → only impacts Public Key shared with that site

Site compromise → only impacts Public Key shared with that site (not other sites)
If malicious entity tries to get user to use Passkey to authenticate, the Credential Manager checks that the requesting domain matches that stored with Passkey. Note: Credential Manager never exposes the Private Key.

Use Cases

Have more than one?

Compromised site

Fallback

Existing authentication methods

Passkey Support

Passkeys.directory

Passkeys.io

Passkeys.com

FIDOAlliance.org

Lots of sites!

https://passkeys.directory (1Password)
https://www.passkeys.io/who-supports-passkeys
https://www.passkeys.com/websites-with-passkey-support-sites-directory
https://fidoalliance.org/passkeys-directory

Passkey Support

Apps

Use embedded browser to authenticate

Browsers

https://www.passkeys.io/compatible-devices

Apps - e.g. Xbox

References

Passkey Standards

https://webauthn.guide

https://fidoalliance.org/passkeys/

https://www.passkeys.com

References

Passkey Overview

CCATP #728 — Bart Busschots on Why FIDO Passkeys Rock - Podfeet Podcasts

How to use Passkeys instead of passwords on iOS 16 | AppleInsider

Breaking Down Passkeys - MacSparky

References

Demo / Test Sites

https://webauthn.io

https://passkey.io

Gotchas

Poor implementation by Sites

Confusing dialogs and alerts

Forced use of Browser's Passkey store

Be aware!

Require different Passkey for each browser, app, device - Passkey Standard specifies that implementer can have this, but it's confusing for users.
Passkey standard has been thought out to be secure
User experience is getting better

Ars Technica article 2024-12-30

“Passkey technology is elegant, but it’s most definitely not usable security”

User experience with Passkeys is confusing but should get better

Ars Technica article from 2024-12-30

Title looks like it’s discouraging the use of Passkey due to “not usable security”.

Passkeys are secure but the user experience with implementations (site, browser integration, credential managers) is frequently confusing. Hopefully there are lessons learnt from poor implementations which lead to better user experience.

Passkeys Scorecard

Secure

Frictionless

Authenticate your identity

Web site or app

Secure - yes, uses Public Key Cryptography

Frictionless
* Enrollment might be tricky - initial proof of identity
* Usage - yes, provided the password store is integrated into the OS

Authenticate your identity - yes

Web site or app
* Adoption by sites and apps - some but growing
* Poor implementations happen
    * Multiple passkeys for different devices or apps

Questions?

Slides

https://www.freerangecoder.com

Slides posted at Free Range Coder

Extra Credit

How it works

Website

WebAuthn Relying Party

Use JavaScript functions for authentication

Web developers use JavaScript functions implemented by the browser for authentication.

How it works

Browser

WebAuthn Client

JavaScript Authentication functions

Operating System integration

Provides JavaScript functions for authentication

How it works

Operating System

WebAuthn / FIDO2 authenticator

Provides system level user authentication

Integrates with **credential managers**

Manages Passkeys

Creation, access, authentication

Operating System interacts with Credential Managers to handle Passkeys generation, access, and authentication

Public Key Cryptography Applications

User authentication / digital signature

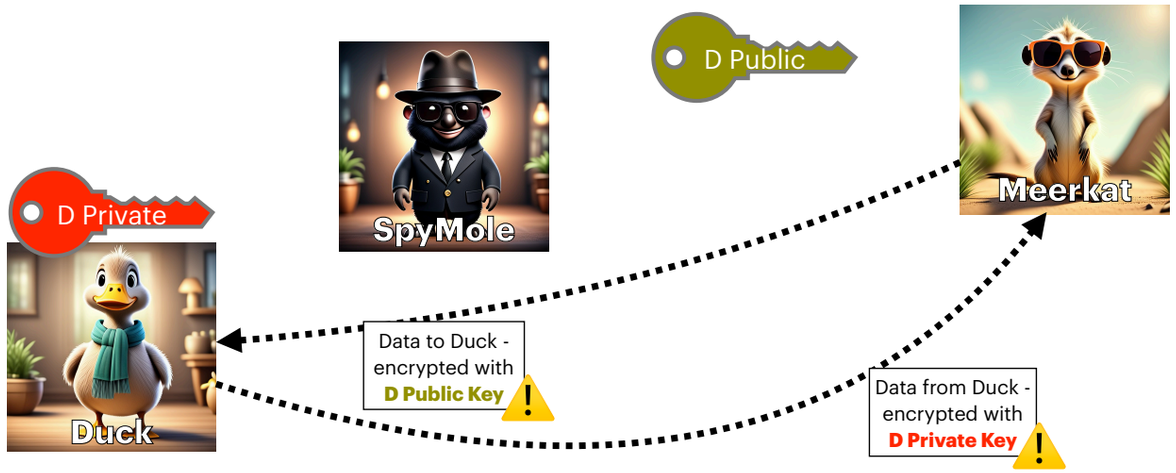
*Encrypt with **private key***

Secure one-way messaging

*Encrypt with **public key***

Single stage encryption

Using PKC for Secure Communications

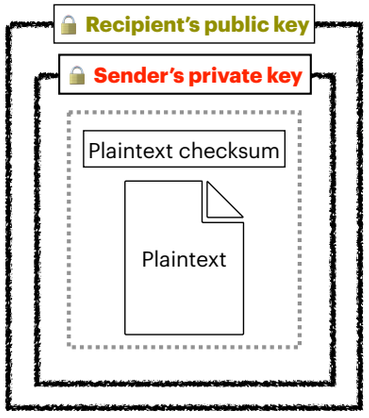


Take 1 - Single Stage Encryption

Data sent to Duck - encrypted with their Public Key, meaning that only Duck can decode the data
* but anyone can send malicious data to Duck by using Duck's public key

Data sent by Duck - encrypted with their Private Key, meaning that message has come from Duck
* but anyone can decrypt what Duck has sent since data is sent out on open networks and their Public Key is out there

Authenticated secure directed messaging



*Compute **plaintext checksum***

*Encrypt plaintext & checksum bundle with **sender's private key***

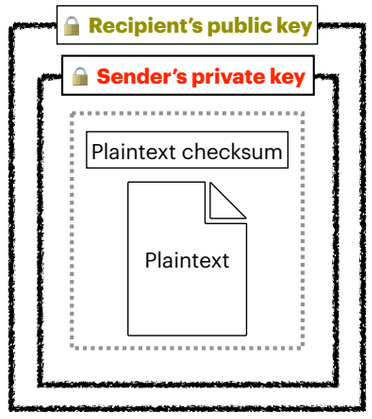
*Encrypt result with **recipient's public key***

Authenticated - receiver can prove that the message originated from the sender
Secure - no external entity can alter the content of the message
Directed - only the receiver can decode the message

Note:
Authenticated secure directed messaging also needs some kind of anti-tamper indicator such as a embedding checksum of the original message plaintext

Simplistic overview 😎

Using PKC for Secure Communications



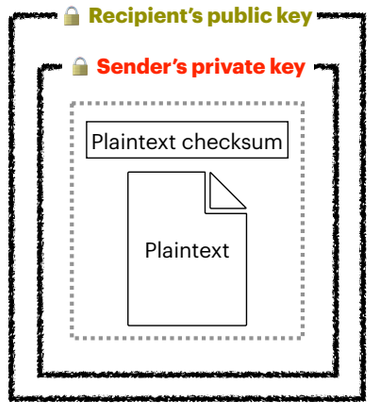
Only **recipient** can use their **private key** to decrypt message to get encrypted plaintext and checksum bundle

⇒ *Encrypted plaintext and checksum bundle securely sent to recipient*

What does this mean?

- Only recipient can use their private key to decrypt the second level encrypted message containing the plaintext and checksum
    - Means that the encrypted plaintext and checksum is securely sent to the recipient

Using PKC for Secure Communications



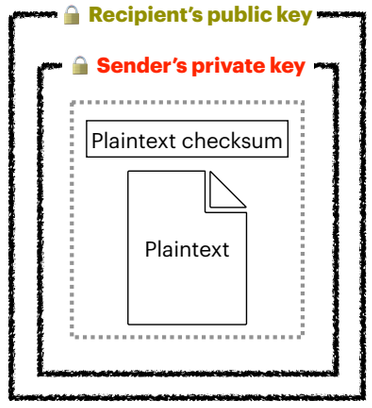
Recipient uses **sender's public key** to decrypt checksum & plaintext bundle

⇒ *Recipient authenticates bundle is from sender*

What does this mean?

- Then the recipient can use the sender's public key to decode the encrypted checksum & plaintext bundle to retrieve checksum & plaintext
    - Means that the recipient can authenticate the sender since the sender used their private key to encrypt

Using PKC for Secure Communications



Recipient computes **decrypted plaintext checksum** and compares to **received checksum**

⇒ *Plaintext has not been tampered with*

What does this mean?

- Then the recipient can calculate the checksum of the decrypted plaintext and compare to the received checksum
    - Means that the decrypted plaintext has not been tampered with

Using PKC for Secure Communications

How to ensure **no eavesdropping** and

no malicious messages

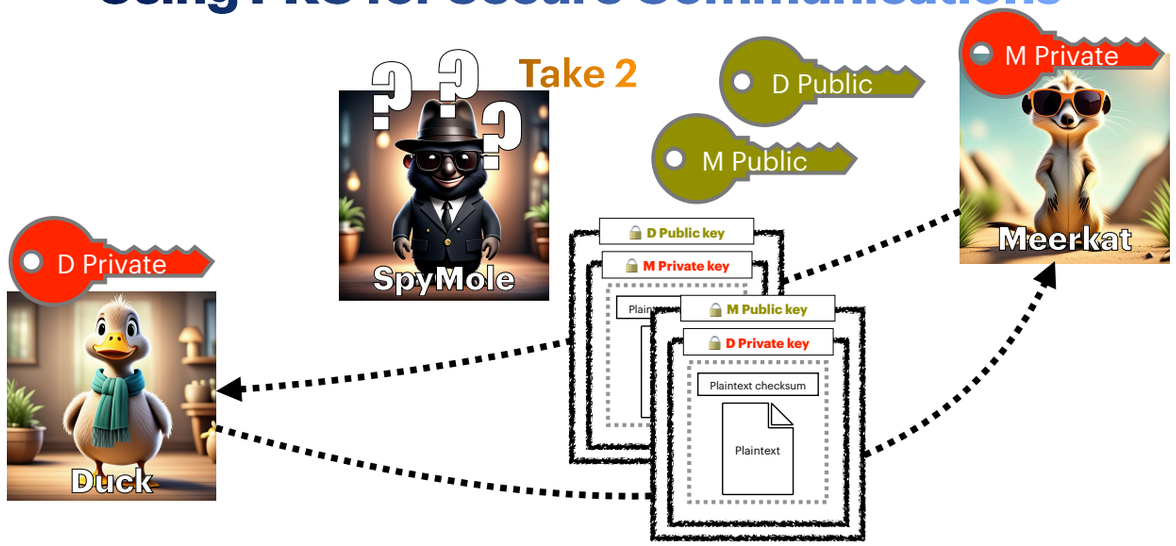
in the data traffic?

No eavesdropping - ensure messages from Duck are only received by Meerkat, and that the message is actually from Duck

No malicious messages - ensure authenticated messages from Meerkat are received by Duck, and only Duck can decrypt them

Other aspects:
* messages can be proved to be authentically from the sender
* only the receiver can decrypt the message
* message has not been tampered with

Using PKC for Secure Communications



Take 2 - Two Stage Encryption

Data sent to Duck - encrypted with M Private Key and then with D Public Key
* Duck can authenticate that entire message has come from Meerkat
* Only Duck can decode the message contents (checksum and plaintext)

Data sent by Duck - encrypted with D Private Key and then with M Public Key
* M can authenticate that entire message has come from Duck
* Only Duck can decode the message contents (checksum and plaintext)

Public Key Cryptography Applications

Securely share a symmetric encryption key

No prior knowledge of shared key

Easier encryption algorithm

Diffie-Hellman key exchange

Diffie–Hellman key exchange - Wikipedia

Use a less computationally intensive means of encryption with the symmetric encryption key

Diffie-Hellman key exchange

Use Cases

More than one...

Browser

Device

Identity

Operating System

Credential Manager

Credential Manager plug-in

Non-native browsers

Firefox, Chrome

Non-native browsers
* Prompt for access to Passkey (macOS)
* macOS: System Settings > Privacy & Security > Passkeys Access for Web Browsers

Use Cases

More than one...

Browser

Device

Identity

Operating System

Credential Manager

Passkeys can be shared

Install Credential Manager

iCloud Drive sync or some other means

Use Cases

More than one...

Browser

Device

Identity

Operating System

Credential Manager

Business & Personal

Passkeys work OK

Use different Credential Managers

Use Cases

More than one...

Browser

Device

Identity

Operating System

Credential Manager

Cross-Platform Credential Managers

Use Cases

More than one...

Browser

Device

Identity

Operating System

Credential Manager

Multiple Credential Managers are OK

You just have to remember which one contains which Passkey

Passkey Support

Developer

webauthn.io

passkeys.dev

passkeys.com

https://www.w3.org/TR/webauthn-2/