Article

# Securing Logins with iCloud Keychain Verification Codes

Use time-based codes generated on-device for a secure authentication experience.

## Overview

Apps and services often use verification codes in a two-factor authentication scheme. When used properly, they strengthen account security over just using passwords alone. During sign in, a service asks for a code to verify the user's identity. These codes are often sent over SMS. The user then confirms their identity by entering the code they receive. Delivering codes over SMS entails a number of risks. For example, SMS-based verification codes can be snooped on carrier networks, or redirected to an attacker through SIM swapping attacks.

With iCloud Keychain verification codes, iPhones, iPads, and Macs generate verification codes entirely offline, reducing the risk associated with sending them online. iCloud Keychain synchronizes codes across all of the user's devices.

## Set Up the Verification Code

Both iOS and macOS handle a special URL based on the industry standard otpauth specification. The Apple-specific version of this URL is identical to the standard, but uses the `apple-otpauth:` scheme. Embed this URL in your webpages and apps to allow your users to set up new code generators directly in the iCloud Keychain password manager with minimal effort. On the web, use an <a> tag to create a link. In apps, create an `NSAttributedString` with a `link` attribute, or open the URL in response to a button tap. When the user taps on the URL, the system prompts them to choose a credential. A new verification code is set up for that credential using the parameters specified in the URL.

The host portion of the URL must be `totp`, and the path should be the proper name of your service, followed by a colon and the user name or email of the account. The `otpauth:` specification defines a number of query parameters. The following are the most important:

secret
: An arbitrary key value encoded in Base32. Secrets should be at least 160 bits.

digits
: The length of a one-time passcode. The value is either 6 or 8. The default is 6.

period
: The number of seconds that a verification code is valid for. The default value is 30.

issuer
: The domain of the site or app. The password manager uses this field to suggest credentials when setting up a new code generator.

Here's an example of the URL for an app called Example for the user meichen3@icloud.com:

```
apple-otpauth://totp/Example:meichen3@icloud.com?
secret=HAZDGMBWGE4DOMZYGEYTOMJTG43TMOJRGE4DANJQGE3DGOBRGI4TOOJZGEYTENJT
GIYTEMJRHEZDA&digits=6&period=30&issuer=example.com
```

## Prepare the Text Input Field

AutoFill on iOS and macOS offer to fill verification codes automatically in text fields where you set the content type of the field to a one-time code type. In SwiftUI, set the content type to `oneTimeCode` with the `textContentType(_:)` view modifier. In UIKit, set `textContentType` to `oneTimeCode`. In AppKit, set `contentType` to `oneTimeCode`. For web-based text fields, set the HTML attribute on the input element to `autocomplete=one-time-code`.

# See Also

## Web authentication sessions

Authenticating a User Through a Web Service
: Use a web authentication session to authenticate a user in your app.

`class ASWebAuthenticationSession`
: A session that an app uses to authenticate a user through a web service.

`struct WebAuthenticationSession`
: A SwiftUI environment value that views use to authenticate someone using a web service.

Supporting Single Sign-On in a Web Browser App

Extend your web browser app to handle web authentication requests from other apps.

`class` `ASWebAuthenticationSessionWebBrowserSessionManager`

A session manager that mediates sharing data between an app and a web browser.

`ASWebAuthenticationSessionWebBrowserSupportCapabilities`

A collection of keys that a browser app uses to declare its ability to handle authentication requests from other apps.