Structure

# SecureField

A control into which people securely enter private text.

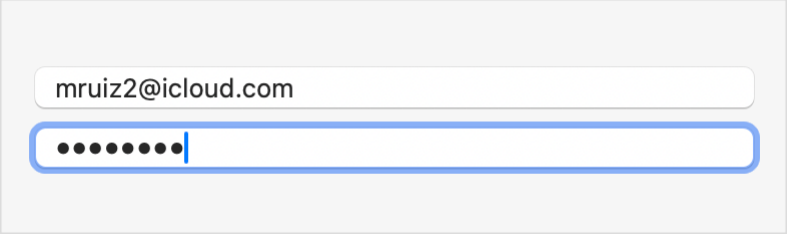
iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst 13.0+ | macOS 10.15+ | tvOS 13.0+ | visionOS 1.0+ | watchOS 6.0+

```
struct SecureField<Label> where Label : View
```

## Overview

Use a secure field when you want the behavior of a `TextField`, but you want to hide the field's text. Typically, you use this for entering passwords and other sensitive information, as the second field in the following screenshot demonstrates:

macOS iOS




The field:

- Displays one dot for each character someone types.
- Hides the dots when someone takes a screenshot in iOS.
- Prevents anyone from cutting or copying the field's contents.
- Displays an indicator when Caps Lock is enabled.

## Bind to a string

A secure field binds to a string value and updates the string on every keystroke or other edit, so you can read its value at any time from elsewhere in your code. The following code shows how to create the above interface, with the secure field bound to a `password` string:

```
@State private var username: String = ""
@State private var password: String = ""

var body: some View {
    VStack {
        TextField("Username", text: $username)
            .autocorrectionDisabled(true)
            #if !os(macOS)
            .textInputAutocapitalization(.never)
            #endif

        SecureField("Password", text: $password)
            .onSubmit {
                handleLogin(username: username, password: password)
            }
    }
    .textFieldStyle(.roundedBorder)
}
```

The field in the above example has an `onSubmit(of:_:)` modifier that sends the `username` and `password` strings to a custom `handleLogin(username:password:)` method if someone presses the Return key while the secure field has focus. You can alternatively provide another mechanism — like a button — to do the same thing.

## Guide people with a prompt

In addition to the string or view that you provide as a label, you can also provide a `Text` view prompt to help guide someone who uses the field, as the following `Form` does:

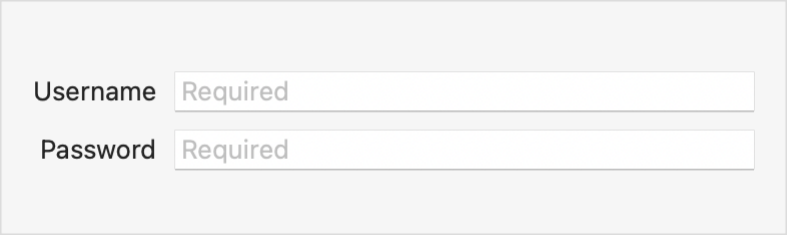
```
Form {
    TextField(text: $username, prompt: Text("Required")) {
        Text("Username")
    }
    .autocorrectionDisabled(true)
    #if !os(macOS)
```

```
    .textInputAutocapitalization(.never)
    #endif

    SecureField(text: $password, prompt: Text("Required")) {
        Text("Password")
    }
}
```

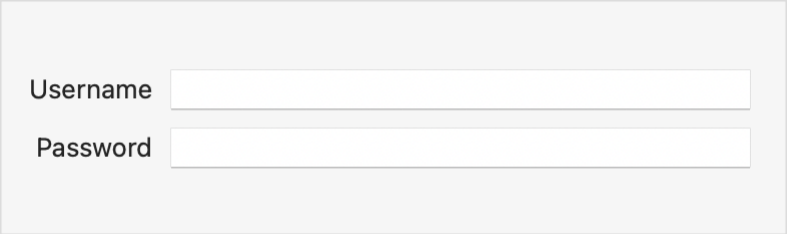
The system uses the label and prompt in different ways depending on the context. For example, a form in macOS places the label against the leading edge of the field and uses the prompt as placeholder text inside the field. The same form in iOS also uses the prompt as placeholder text, but doesn't display the label:

macOS iOS




If you remove the prompt from the previous example, the field keeps the label on the leading edge and omits the placeholder text in macOS, but displays the label as a placeholder in iOS:

macOS iOS




# Topics

## Creating a secure text field

`init(_:text:)`

Creates a secure field with a prompt generated from a `Text`.

`init(_:text:prompt:)`

Creates a secure field with a prompt generated from a `Text`.

`init(text: Binding<String>, prompt: Text?, label: () -> Label)`

Creates a secure field with a prompt generated from a `Text`.

### Deprecated initializers

~~`init(_:text:onCommit:)`~~

Creates an instance.

Deprecated

---

## Relationships

### Conforms To

`View`

---

## See Also

### Getting text input

{} Building rich SwiftUI text experiences

Build an editor for formatted text using SwiftUI text editor views and attributed strings.

`struct TextField`

A control that displays an editable text interface.

`func textFieldStyle<S>(S) -> some View`

Sets the style for text fields within this view.

`struct TextEditor`

A view that can display and edit long-form text.