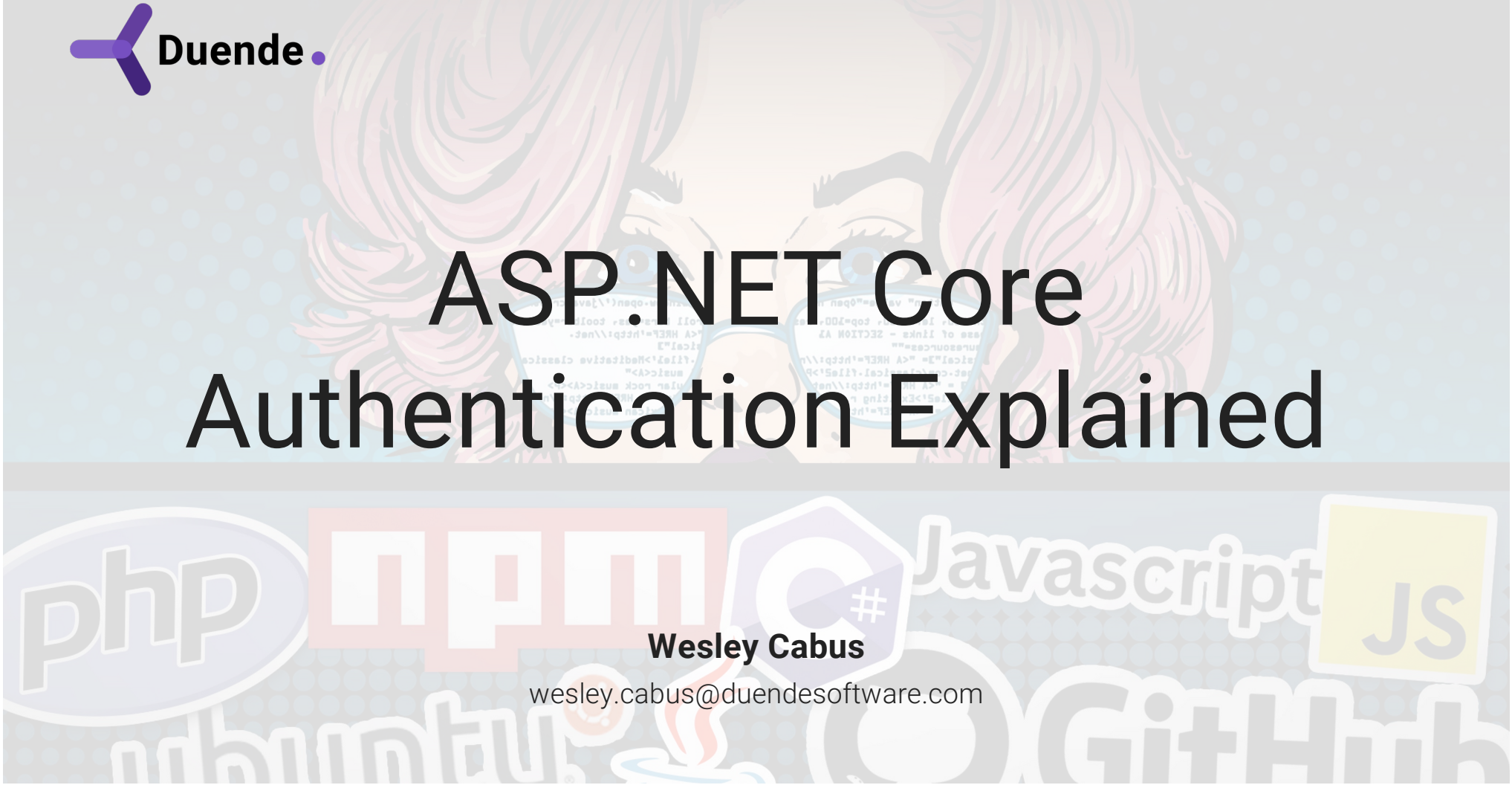
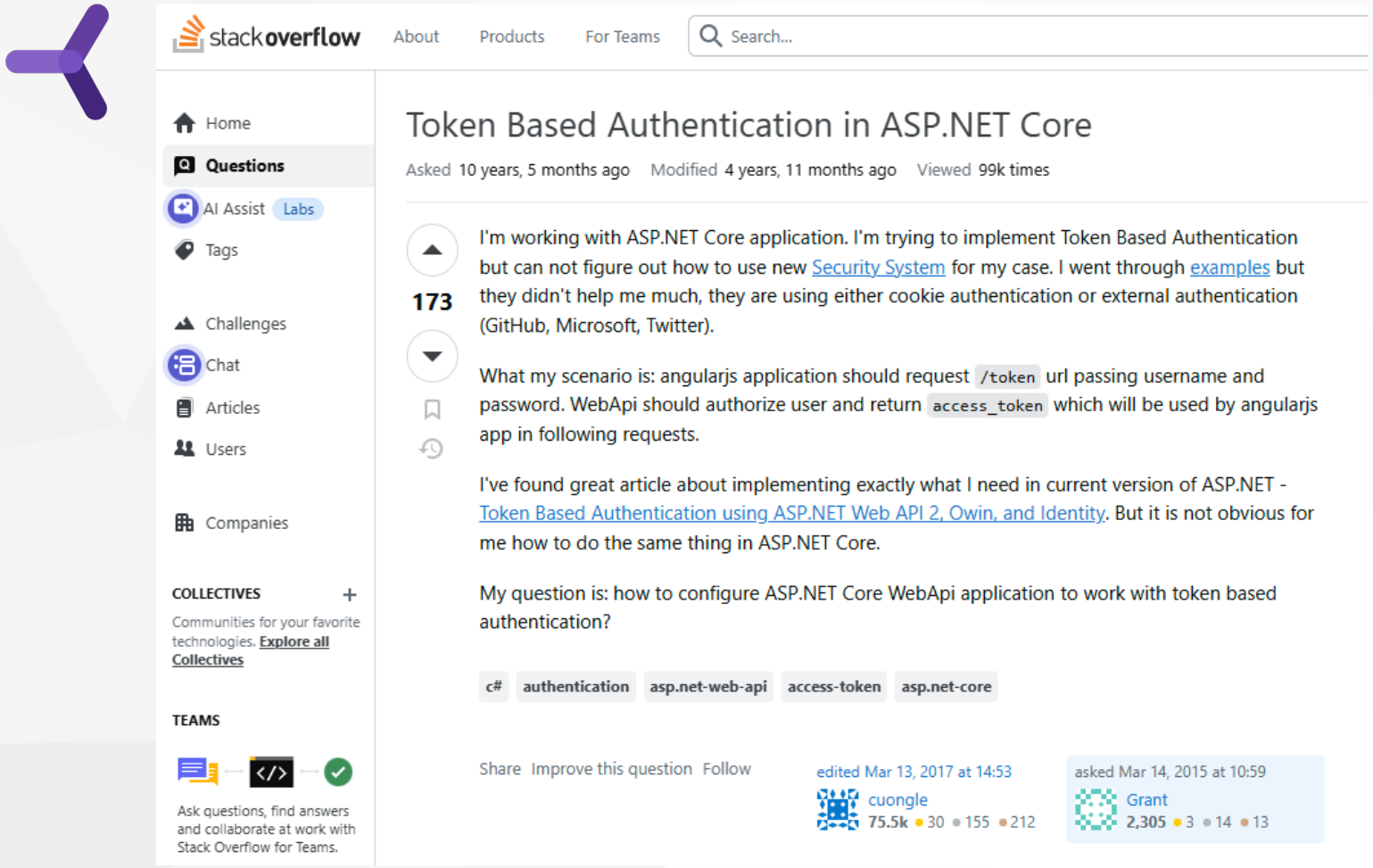
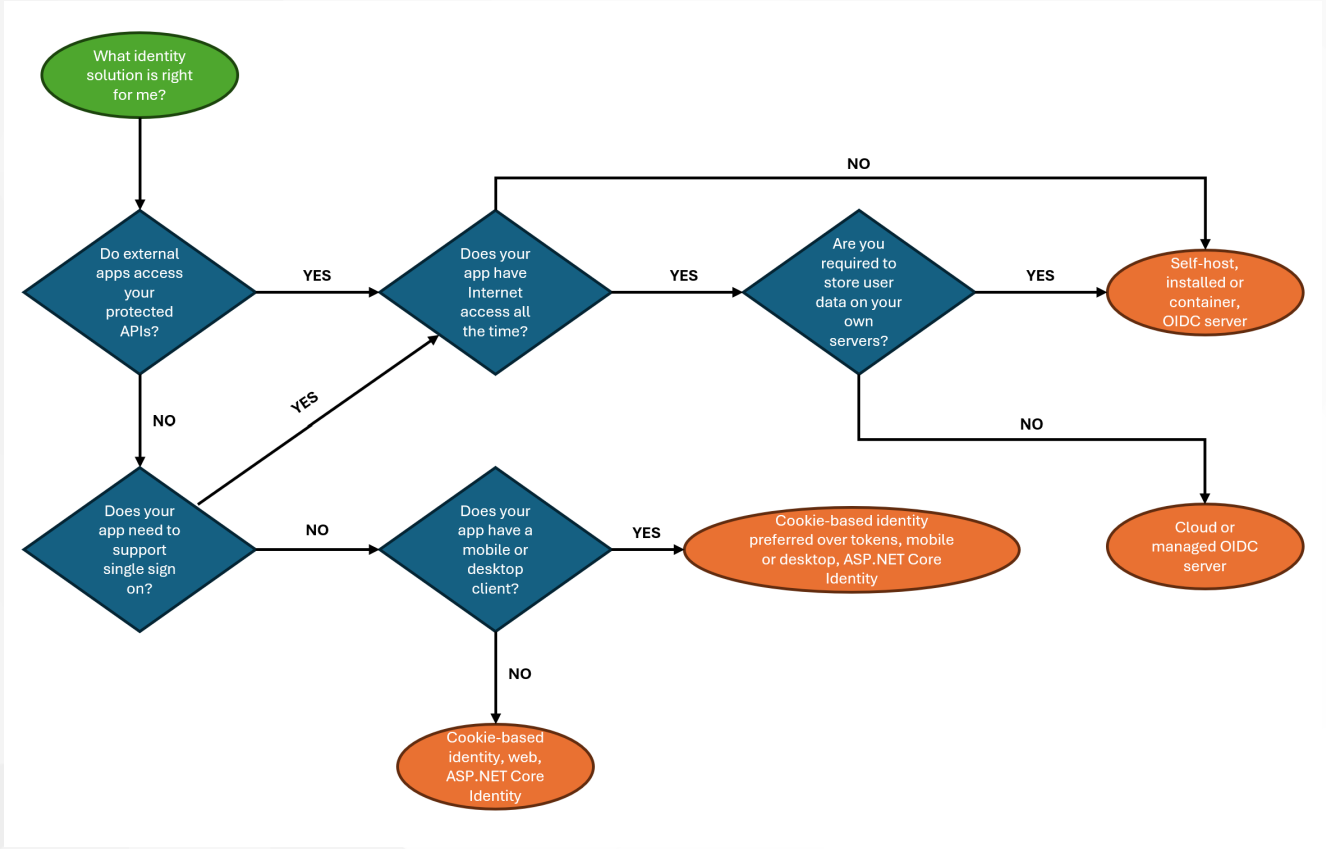


ASP.NET Core Authentication Explained

Wesley Cabus
wesley.cabus@duendesoftware.com



Source: https://stackoverflow.com/questions/29048122/token-based-authentication-in-asp-net-core



Source: https://learn.microsoft.com/en-us/aspnet/core/security/how-to-choose-identity-solution?view=aspnetcore-9.0#identity-vs-oidc-server

Learn / .NET / ASP.NET Core / Ask Learn Focus mode

Secure an ASP.NET Core Blazor Web App with OpenID Connect (OIDC)

07/29/2025

Choose the app specification

Non-BFF pattern (Interactive Auto) | Non-BFF pattern (Interactive Server) | BFF pattern (Interactive Auto)

This article describes how to secure a Blazor Web App with OpenID Connect (OIDC) using a sample app in the dotnet/blazor-samples GitHub repository (.NET 8 or later) (how to download).

For Microsoft Entra ID or Azure AD B2C, you can use AddMicrosoftIdentityWebApp from Microsoft Identity Web (Microsoft.Identity.Web NuGet package, API documentation), which adds both the OIDC and Cookie authentication handlers with the appropriate defaults. The sample app and the guidance in this article don't use Microsoft Identity Web. The guidance demonstrates how to configure the OIDC handler *manually* for any OIDC provider. For more information on implementing Microsoft Identity Web, see Secure an ASP.NET Core Blazor Web App with Microsoft Entra ID.

This version of the article covers implementing OIDC without adopting the Backend for Frontend (BFF) pattern with an app that adopts global Interactive Auto rendering (server and `.Client` projects). The BFF pattern is useful for making authenticated requests to external services. Change the article version selector to **BFF pattern** if the app's specification calls for adopting the BFF pattern.

The following specification is adopted:

- The Blazor Web App uses the Auto render mode with global interactivity.
- Custom auth state provider services are used by the server and client apps to capture the user's authentication state and flow it between the server and client.

Source: https://learn.microsoft.com/en-us/aspnet/core/blazor/security/blazor-web-app-with-oidc?view=aspnetcore-9.0&pivots=non-bff-pattern

CHAT

I need to set up authentication using OAuth and OpenID Connect, can you help me please?

Program.cs

Workspace (rerun without)

Used 31 references

Collecting workspace information

To set up authentication using OAuth and OpenID Connect in your ASP.NET Core MVC app, you typically use the `AddAuthentication` and `AddOpenIdConnect` methods in your Program.cs. Here's how you can get started:

```
// ...existing code...

// Add services to the container.
builder.Services.AddControllersWithViews();

// Add authentication services
builder.Services.AddAuthentication(options =>
{
    options.DefaultScheme = "Cookies";
    options.DefaultChallengeScheme = "oidc";
})
.AddCookie("Cookies")
.AddOpenIdConnect("oidc", options =>
{
    options.Authority = "https://your-identity-provider.com"; // Set your authority URL
    options.ClientId = "your-client-id";
    options.ClientSecret = "your-client-secret";
    options.ResponseType = "code";
```

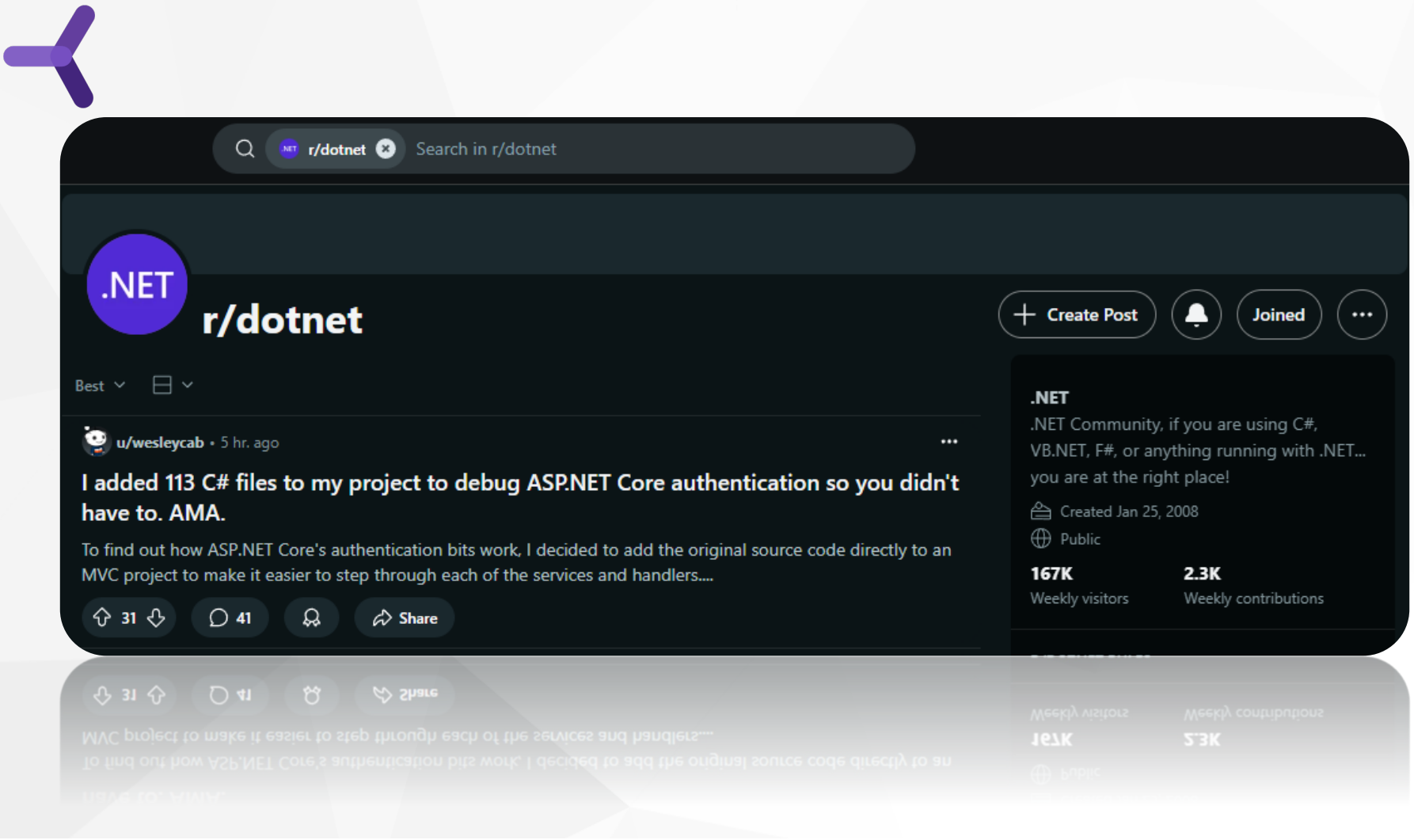

Let’s see how it actually works

Hi, I’m Wesley

Customer Success Engineer
at Duende Software

Coffee enthusiast

Beer aficionado


r/dotnet
Search in r/dotnet
.NET
r/dotnet
Create Post
Joined
Best
u/wesleycab • 5 hr. ago
I added 113 C# files to my project to debug ASP.NET Core authentication so you didn't have to. AMA.
To find out how ASP.NET Core's authentication bits work, I decided to add the original source code directly to an MVC project to make it easier to step through each of the services and handlers....
31
41
Share
.NET
.NET Community, if you are using C#, VB.NET, F#, or anything running with .NET... you are at the right place!
Created Jan 25, 2008
Public
167K
Weekly visitors
2.3K
Weekly contributions


Part 1:
The Grand Scheme of Things

AddAuthentication()

Handlers / Schemes

- Cookie
- OpenIdConnect
- JWT Bearer

Actions

- Authenticate
- Challenge
- SignIn

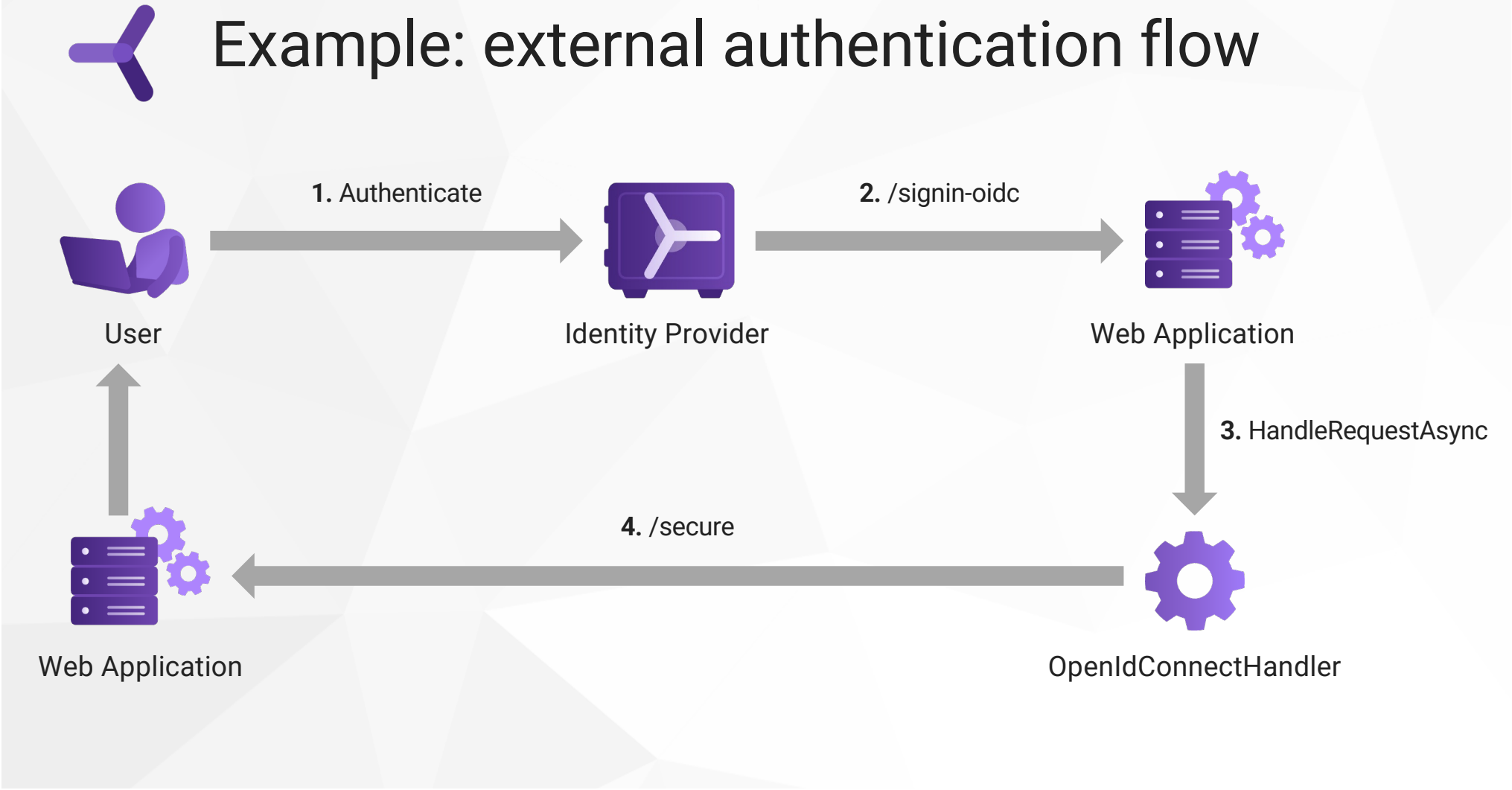
UseAuthentication()

AuthenticationMiddleware

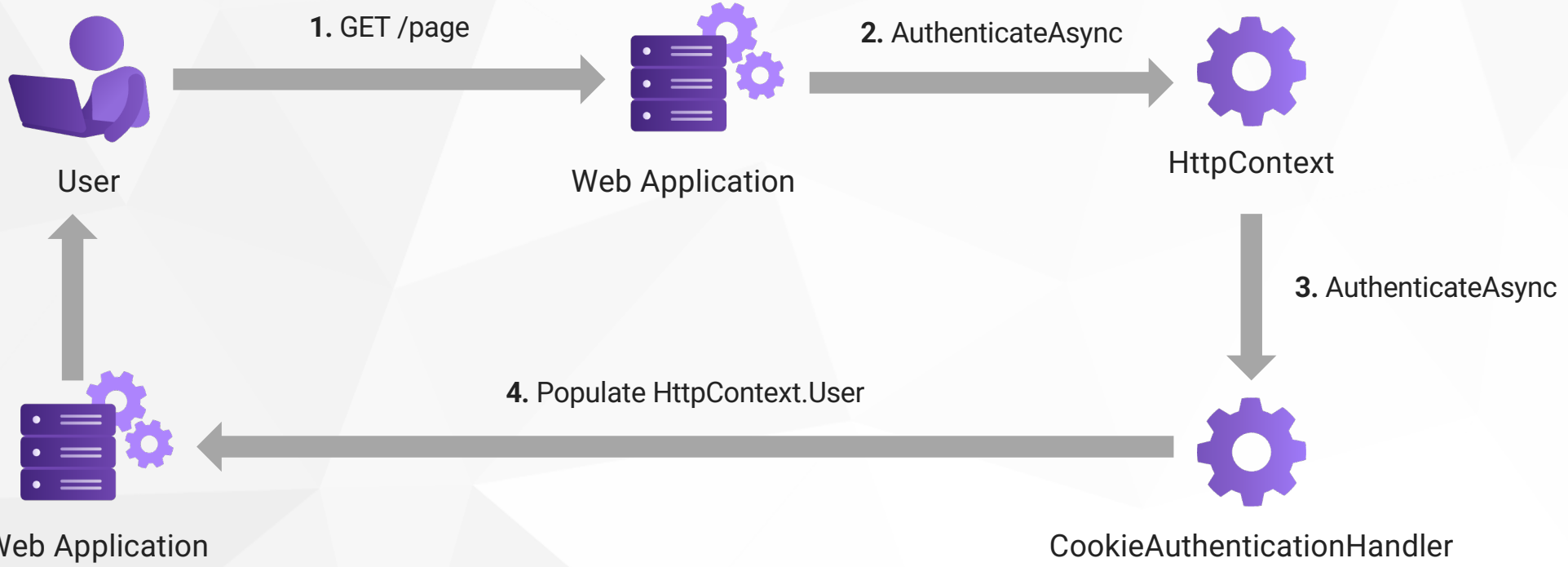
IAuthenticationHandler.
HandleRequestAsync()

DefaultAuthenticateScheme +
AuthenticateAsync()

Example: external authentication flow



Example: cookies



AddAuthentication()

- Adds the authentication middleware services
  - IAuthenticationService
  - Scheme and handler providers
  - Claims transformation
  - Data protection
  - Web encoders
- Returns an AuthenticationBuilder
  - Allowing you to register schemes
  - And a scheme = a handler + its settings

UseAuthentication()

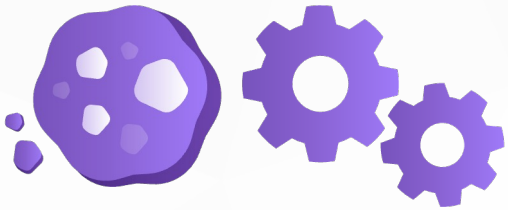
- Registers the authentication middleware in the HTTP pipeline
- Calls authentication handlers for each HTTP request
  - Short-circuit via IAuthenticationRequestHandler.HandleRequestAsync
  - Default authenticate scheme
  - HttpContext.AuthenticateAsync
- Sets HttpContext.User

Part 2:

Authentication Scheme

Authentication Scheme

"Cookies"



Authentication Handler
&
Specific Configuration

Authentication Scheme

```
services.AddAuthentication( ... )
.AddCookie(IdentityConstants.ApplicationScheme, o ⇒
{
    o.LoginPath = new PathString("/Account/Login");
})
.AddCookie(IdentityConstants.ExternalScheme, o ⇒
{
    o.Cookie.Name = IdentityConstants.ExternalScheme;
    o.ExpireTimeSpan = TimeSpan.FromMinutes(5);
})
.AddCookie(IdentityConstants.TwoFactorRememberMeScheme, o ⇒
{
    o.Cookie.Name = IdentityConstants.TwoFactorRememberMeScheme;
})
.AddCookie(IdentityConstants.TwoFactorUserIdScheme, o ⇒
{
    o.Cookie.Name = IdentityConstants.TwoFactorUserIdScheme;
    o.ExpireTimeSpan = TimeSpan.FromMinutes(5);
});
```

Authentication Scheme

```
services.AddAuthentication( ... )
.AddCookie(IdentityConstants.ApplicationScheme, o =>
{
    o.LoginPath = new PathString("/Account/Login");
})
.AddCookie(IdentityConstants.ExternalScheme, o =>
{
    o.Cookie.Name = IdentityConstants.ExternalScheme;
    o.ExpireTimeSpan = TimeSpan.FromMinutes(5);
})
.AddCookie(IdentityConstants.TwoFactorRememberMeScheme, o =>
{
    o.Cookie.Name = IdentityConstants.TwoFactorRememberMeScheme;
})
.AddCookie(IdentityConstants.TwoFactorUserIdScheme, o =>
{
    o.Cookie.Name = IdentityConstants.TwoFactorUserIdScheme;
    o.ExpireTimeSpan = TimeSpan.FromMinutes(5);
});
```

Authentication Scheme

```
services.AddAuthentication( ... )
.AddCookie(IdentityConstants.ApplicationScheme, o =>
{
    o.LoginPath = new PathString("/Account/Login");
})
.AddCookie(IdentityConstants.ExternalScheme, o =>
{
    o.Cookie.Name = IdentityConstants.ExternalScheme;
    o.ExpireTimeSpan = TimeSpan.FromMinutes(5);
})
.AddCookie(IdentityConstants.TwoFactorRememberMeScheme, o =>
{
    o.Cookie.Name = IdentityConstants.TwoFactorRememberMeScheme;
})
.AddCookie(IdentityConstants.TwoFactorUserIdScheme, o =>
{
    o.Cookie.Name = IdentityConstants.TwoFactorUserIdScheme;
    o.ExpireTimeSpan = TimeSpan.FromMinutes(5);
});
```

Authentication Scheme

```
services.AddAuthentication( ... )
.AddCookie(IdentityConstants.ApplicationScheme, o ⇒
{
    o.LoginPath = new PathString("/Account/Login");
})
.AddCookie(IdentityConstants.ExternalScheme, o ⇒
{
    o.Cookie.Name = IdentityConstants.ExternalScheme;
    o.ExpireTimeSpan = TimeSpan.FromMinutes(5);
})
.AddCookie(IdentityConstants.TwoFactorRememberMeScheme, o ⇒
{
    o.Cookie.Name = IdentityConstants.TwoFactorRememberMeScheme;
})
.AddCookie(IdentityConstants.TwoFactorUserIdScheme, o ⇒
{
    o.Cookie.Name = IdentityConstants.TwoFactorUserIdScheme;
    o.ExpireTimeSpan = TimeSpan.FromMinutes(5);
});
```

Authentication Scheme

```
services.AddAuthentication( ... )
.AddCookie(IdentityConstants.ApplicationScheme, o ⇒
{
    o.LoginPath = new PathString("/Account/Login");
})
.AddCookie(IdentityConstants.ExternalScheme, o ⇒
{
    o.Cookie.Name = IdentityConstants.ExternalScheme;
    o.ExpireTimeSpan = TimeSpan.FromMinutes(5);
})
.AddCookie(IdentityConstants.TwoFactorRememberMeScheme, o ⇒
{
    o.Cookie.Name = IdentityConstants.TwoFactorRememberMeScheme;
})
.AddCookie(IdentityConstants.TwoFactorUserIdScheme, o ⇒
{
    o.Cookie.Name = IdentityConstants.TwoFactorUserIdScheme;
    o.ExpireTimeSpan = TimeSpan.FromMinutes(5);
});
```

Authentication Scheme

```
services.AddAuthentication( ... )
.AddCookie(IdentityConstants.ApplicationScheme, o ⇒
{
    o.LoginPath = new PathString("/Account/Login");
})
.AddCookie(IdentityConstants.ExternalScheme, o ⇒
{
    o.Cookie.Name = IdentityConstants.ExternalScheme;
    o.ExpireTimeSpan = TimeSpan.FromMinutes(5);
})
.AddCookie(IdentityConstants.TwoFactorRememberMeScheme, o ⇒
{
    o.Cookie.Name = IdentityConstants.TwoFactorRememberMeScheme;
})
.AddCookie(IdentityConstants.TwoFactorUserIdScheme, o ⇒
{
    o.Cookie.Name = IdentityConstants.TwoFactorUserIdScheme;
    o.ExpireTimeSpan = TimeSpan.FromMinutes(5);
});
```

Authentication Scheme

```
services
    .AddAuthentication()
    .AddCookie();
```

Authentication Scheme

```
services
    .AddAuthentication()
    .AddCookie(
        authenticationScheme:
        CookieAuthenticationDefaults.AuthenticationScheme,
        displayName: null,
        configureOptions: _ ⇒ { });
```

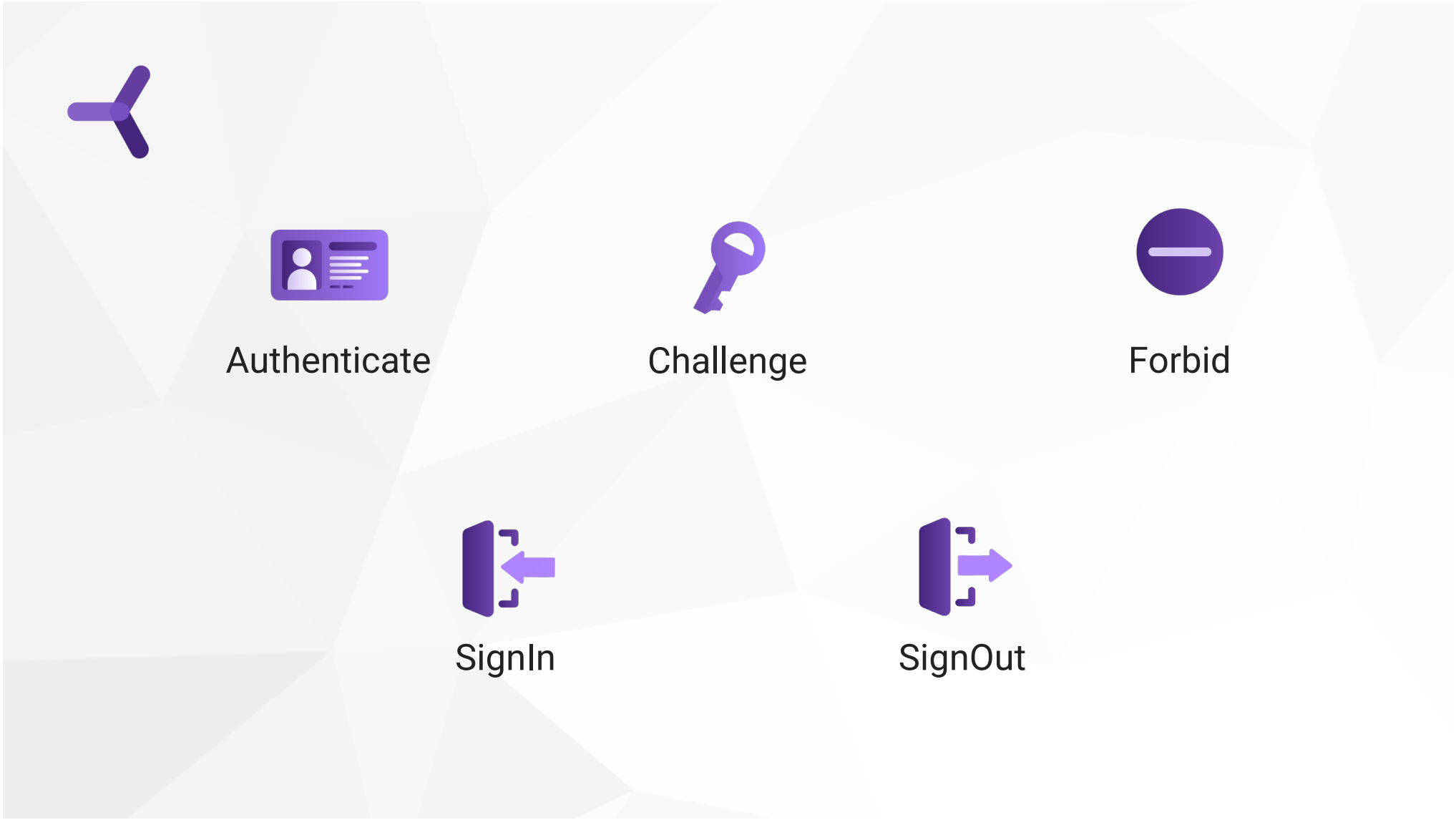
Authentication Scheme

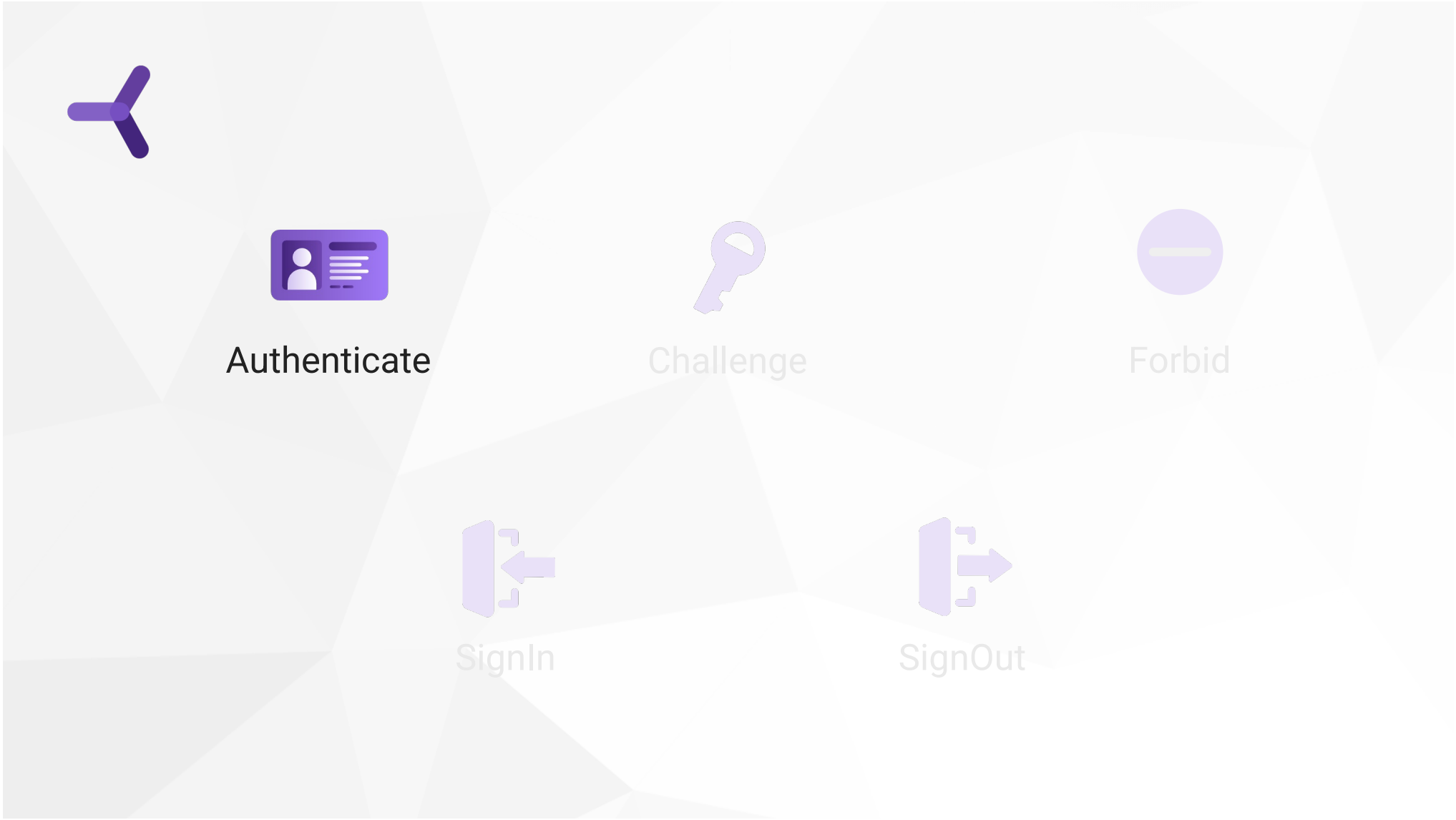
```
services
    .AddAuthentication(o =>
    {
        o.DefaultScheme =
        CookieAuthenticationDefaults.AuthenticationScheme;
    })
    .AddCookie(
        authenticationScheme:
        CookieAuthenticationDefaults.AuthenticationScheme,
        displayName: null,
        configureOptions: _ => { });
```

Authentication Scheme

```
services
    .AddAuthentication(o ⇒
    {
        // o.DefaultScheme = "Cookies";
        o.DefaultAuthenticateScheme = "Cookies";
        o.DefaultChallengeScheme = "Cookies";
        o.DefaultForbidScheme = "Cookies";
        o.DefaultSignInScheme = "Cookies";
        o.DefaultSignOutScheme = "Cookies";
    })
    .AddCookie("Cookies");
```

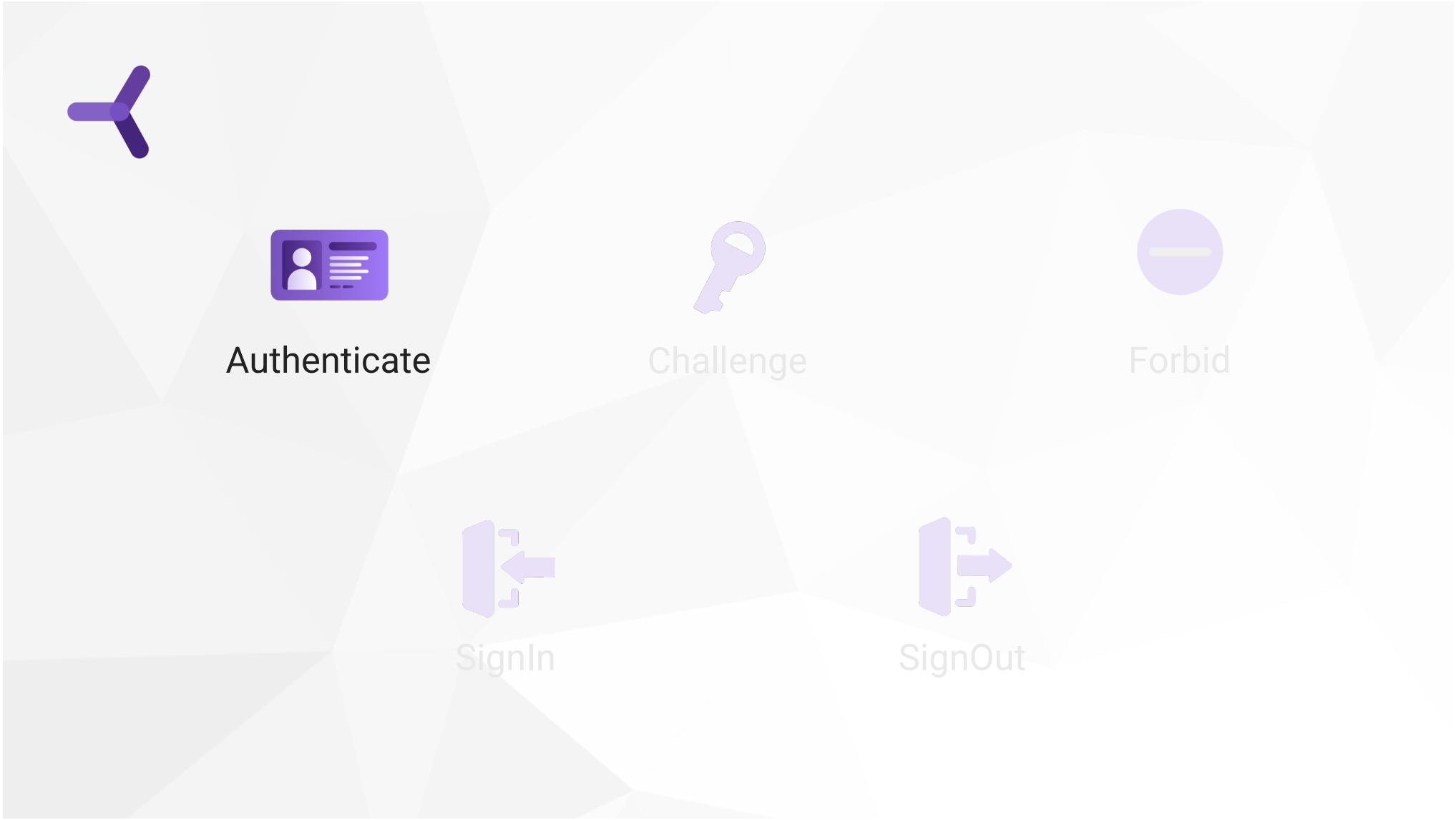

Part 3:
Authentication Actions

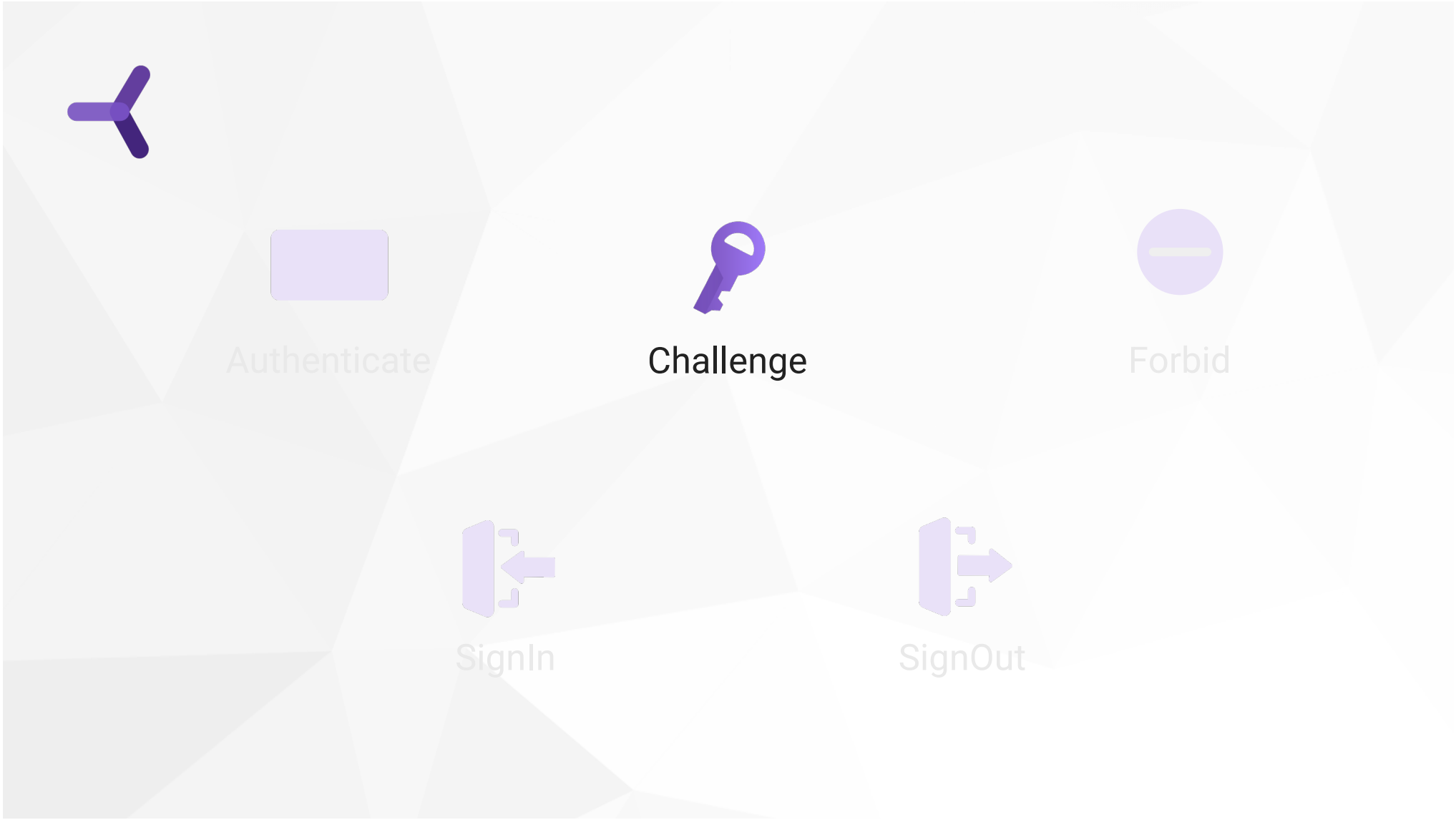

Authenticate
Challenge
Forbid
SignIn
SignOut

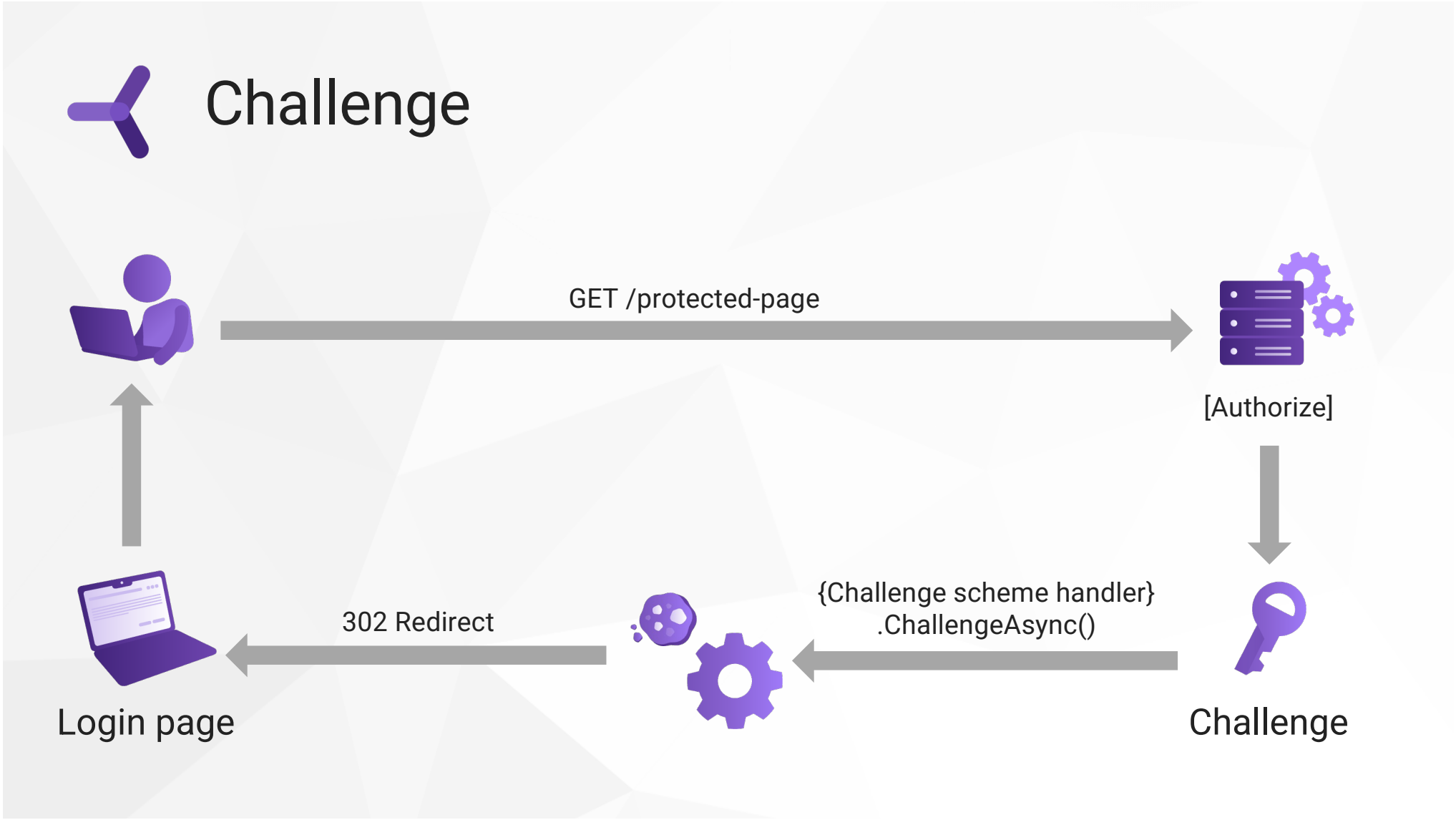

Authenticate
Challenge
Forbid
SignIn
SignOut

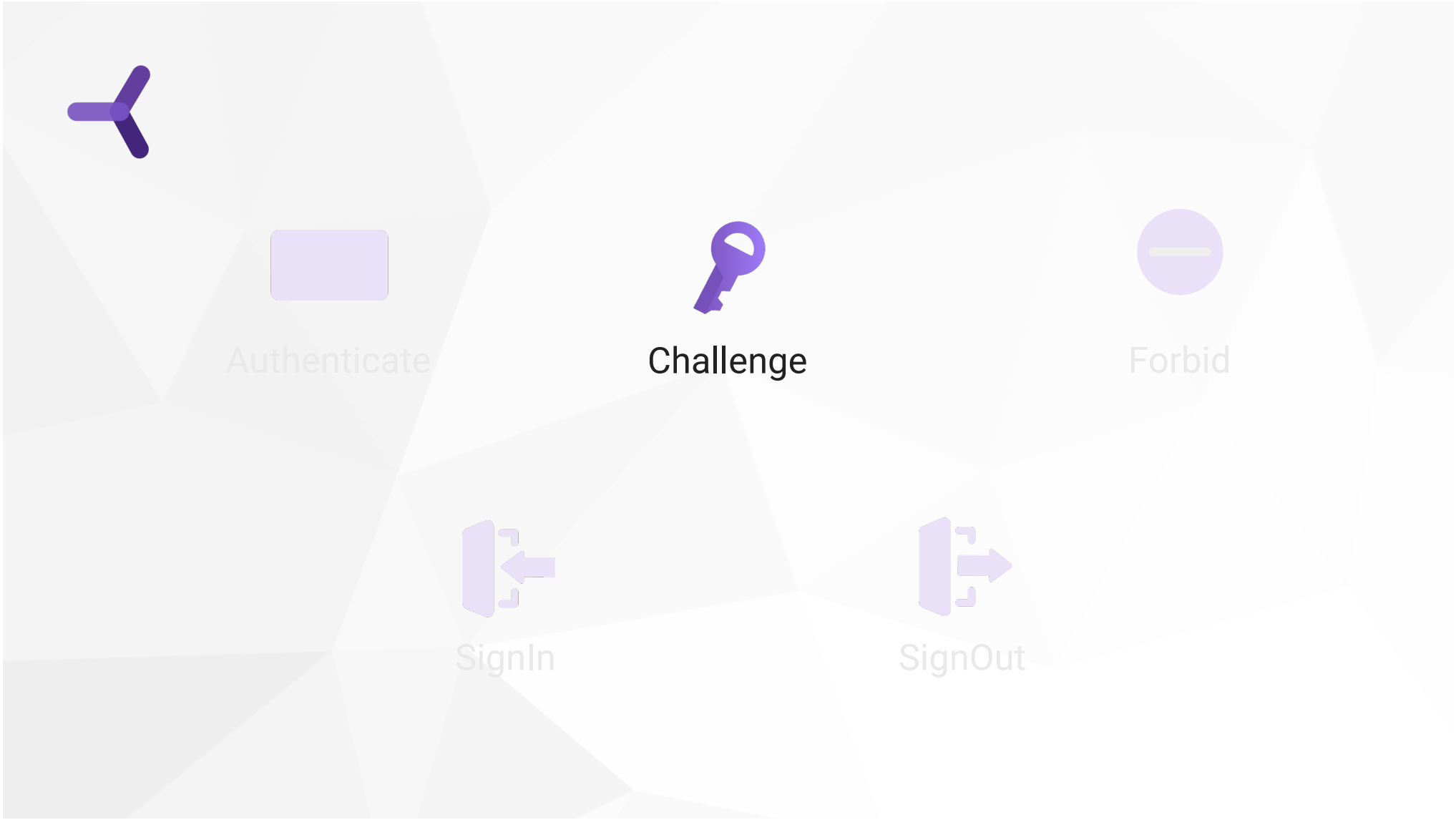
Authenticate with Cookie

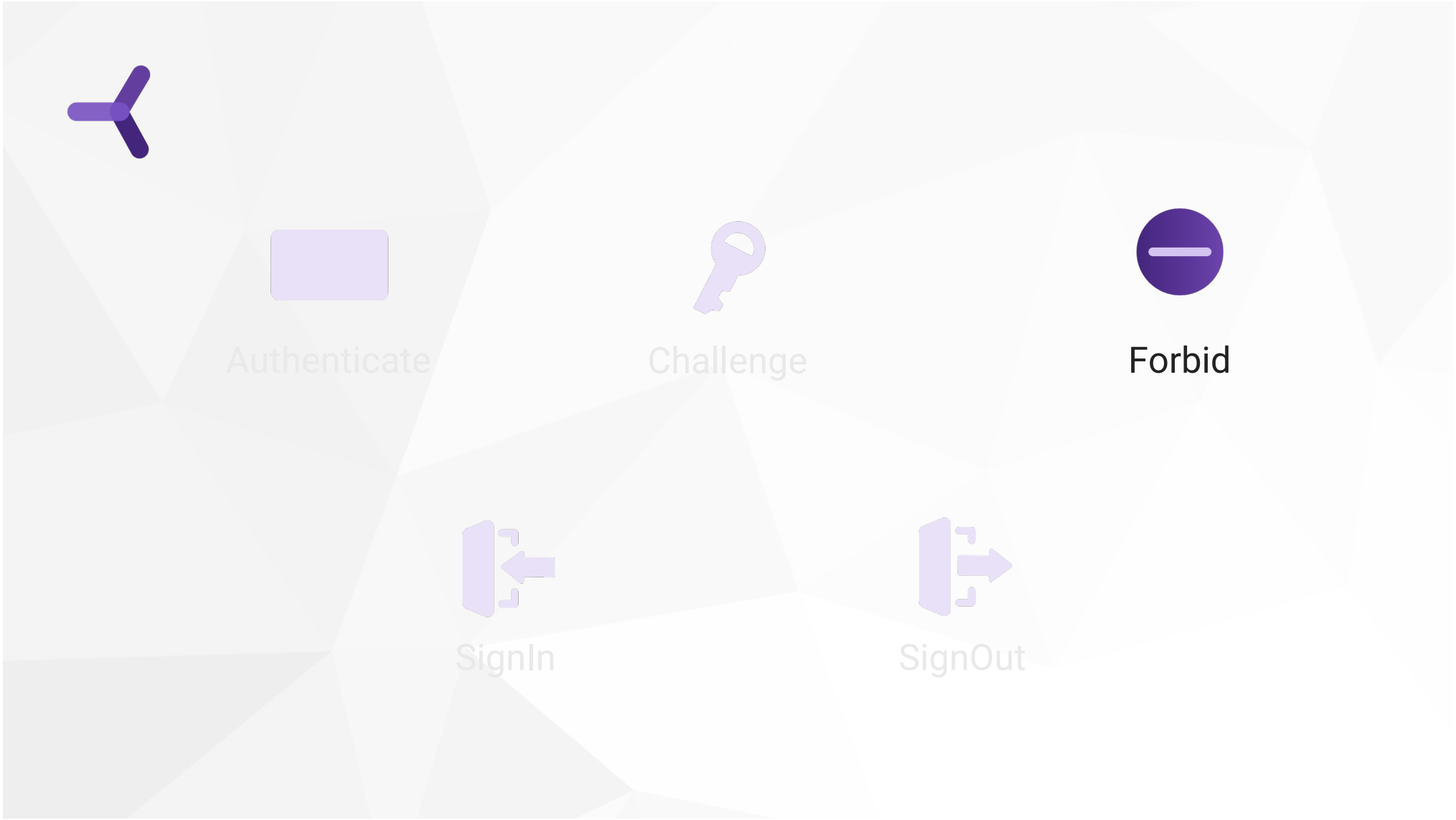
Authenticate with JWT Bearer

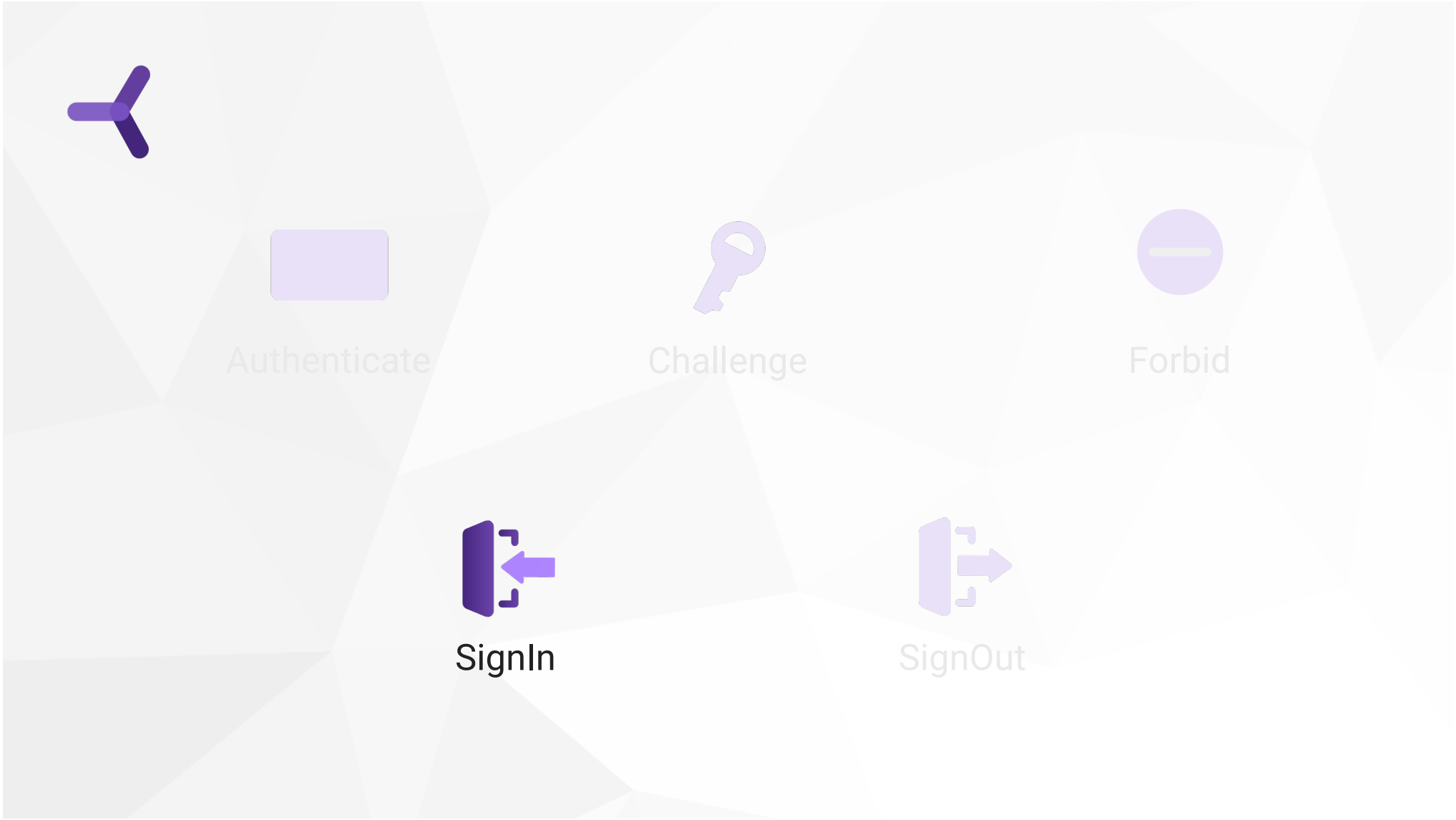

Authenticate
Challenge
Forbid
SignIn
SignOut

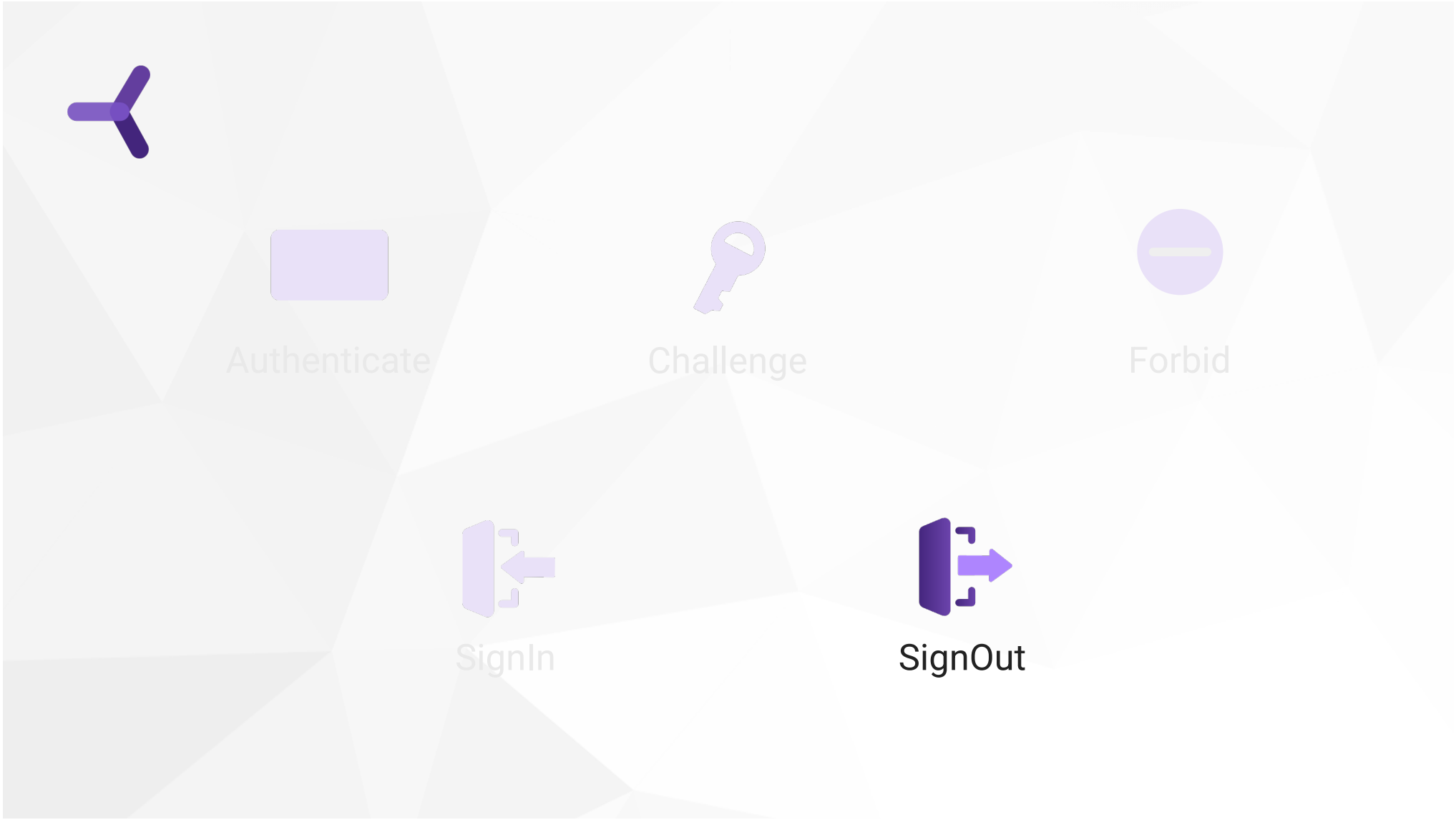

Authenticate
Challenge
Forbid
SignIn
SignOut


Challenge
GET /protected-page
[Authorize]
{Challenge scheme handler}
.ChallengeAsync()
302 Redirect
Challenge
Login page


Authenticate
Challenge
Forbid
SignIn
SignOut


Authenticate
Challenge
Forbid
SignIn
SignOut


Authenticate
Challenge
Forbid
SignIn
SignOut


Authenticate
Challenge
Forbid
SignIn
SignOut

Part 4:

Authentication Handlers

IAuthenticationHandler

IAuthenticationSignOutHandler

IAuthenticationRequestHandler

IAuthenticationSignInHandler

AuthenticationHandler<>

SignOutAuthenticationHandler<>

RemoteAuthenticationHandler<>

SignInAuthenticationHandler<>

OAuthHandler<>

OpenIdConnectHandler

CookieAuthenticationHandler


Duende.
Thank you!