go-oidc

OpenID Connect support for Go

This package enables OpenID Connect support for the golang.org/x/oauth2 package.

```
provider, err := oidc.NewProvider(ctx, "https://accounts.google.com")
if err != nil {
    // handle error
}

// Configure an OpenID Connect aware OAuth2 client.
oauth2Config := oauth2.Config{
    ClientID:     clientID,
    ClientSecret: clientSecret,
    RedirectURL:  redirectURL,

    // Discovery returns the OAuth2 endpoints.
    Endpoint: provider.Endpoint(),

    // "openid" is a required scope for OpenID Connect flows.
    Scopes: []string{oidc.ScopeOpenID, "profile", "email"},
}
```

OAuth2 redirects are unchanged.

```
func handleRedirect(w http.ResponseWriter, r *http.Request) {
    http.Redirect(w, r, oauth2Config.AuthCodeURL(state), http.StatusFound)
}
```

The on responses, the provider can be used to verify ID Tokens.

```
var verifier = provider.Verifier(&oidc.Config{ClientID: clientID})

func handleOAuth2Callback(w http.ResponseWriter, r *http.Request) {
    // Verify state and errors.

    oauth2Token, err := oauth2Config.Exchange(ctx, r.URL.Query().Get("code"))
    if err != nil {
        // handle error
    }

    // Extract the ID Token from OAuth2 token.
    rawIDToken, ok := oauth2Token.Extra("id_token").(string)
    if !ok {
        // handle missing token
```

```
    }

    // Parse and verify ID Token payload.
    idToken, err := verifier.Verify(ctx, rawIDToken)
    if err != nil {
        // handle error
    }

    // Extract custom claims
    var claims struct {
        Email    string `json:"email"`
        Verified bool   `json:"email_verified"`
    }
    if err := idToken.Claims(&claims); err != nil {
        // handle error
    }
}
```