```
package auth0

import (
"encoding/json"
"net/http"
"strings"
"sync"

  "github.com/go-errors/errors"
  "gopkg.in/square/go-jose.v2"

)

var (
ErrInvalidContentType = errors.New("Should have a JSON content type for JWKS endpoint.")
ErrNoKeyFound = errors.New("No Keys has been found")
ErrInvalidTokenHeader = errors.New("No valid header found")
ErrInvalidAlgorithm = errors.New("Only RS256 is supported")
)

type JWKClientOptions struct {
URI string
}

type JWKS struct {
Keys []jose.JSONWebKey json:"keys"
}

type JWKClient struct {
keys map[string]jose.JSONWebKey
mu sync.Mutex
options JWKClientOptions
}

func NewJWKClient(options JWKClientOptions) *JWKClient {
return &JWKClient{keys: map[string]jose.JSONWebKey{}, options: options}
}

func (j *JWKClient) GetKey(ID string) (jose.JSONWebKey, bool) {
j.mu.Lock()
defer j.mu.Unlock()
```

```
key, exist := j.keys[ID]

if !exist {
    j.downloadKeys()
}

key, exist = j.keys[ID]
return key, exist
```

}

func (j *JWKClient) downloadKeys() error {
resp, err := http.Get(j.options.URI)

```
if err != nil {
    return err
}
defer resp.Body.Close()

if contentH := resp.Header.Get("Content-Type"); !strings.HasPrefix(contentH, "applic
    return ErrInvalidContentType
}

var jwks = JWKS{}
err = json.NewDecoder(resp.Body).Decode(&jwks)

if err != nil {
    return err
}

if len(jwks.Keys) < 1 {
    return ErrNoKeyFound
}

for _, key := range jwks.Keys {
    j.keys[key.KeyID] = key
}

return nil
```

}

func (j *JWKClient) GetSecret(req *http.Request) (interface{}, error) {
t, err := FromHeader(req)

```go
if err != nil {
    return nil, err
}

if len(t.Headers) < 1 {
    return nil, ErrInvalidTokenHeader
}

header := t.Headers[0]
if header.Algorithm != "RS256" {
    return nil, ErrInvalidAlgorithm
}

webKey, exist := j.GetKey(header.KeyID)
if !exist {
    return nil, ErrNoKeyFound
}

return webKey.Key, nil
```

}