# x509-pkix-full-project

Bu proje `crypto/x509` ile birlikte `crypto/x509/pkix` kullanarak **Root CA → Intermediate CA → Server/Client** sertifika zinciri oluşturur, doğrular ve Docker ile paketlenmiş haliyle çalıştırılabilir.

---

## Proje yapısı

```
x509-pkix-full-project/
├── Dockerfile
├── Makefile
├── README.md
├── generate_pkix_certs.go
├── verify_chain.go
└── certs/ (oluşturulur)
```

---

## generate_pkix_certs.go

Bu dosya `pkix.Name` ve diğer `pkix` tiplerini kullanarak Root CA, Intermediate CA, Server ve Client sertifikalarını üretir.

```go
package main

import (
    "crypto/rand"
    "crypto/rsa"
    "crypto/x509"
    "crypto/x509/pkix"
    "encoding/pem"
    "fmt"
    "math/big"
    "os"
    "time"
)

func savePEM(path, typ string, data []byte) error {
    f, err := os.Create(path)
    if err != nil { return err }
    defer f.Close()
    return pem.Encode(f, &pem.Block{Type: typ, Bytes: data})
}

func main() {
    os.MkdirAll("certs", 0755)
```

```
    // 1) Root CA
    rootKey, _ := rsa.GenerateKey(rand.Reader, 3072)
    rootTmpl := x509.Certificate{
        SerialNumber: big.NewInt(1),
        Subject: pkix.Name{
            CommonName:   "My Root CA",
            Organization: []string{"Example Root Org"},
            Country:      []string{"TR"},
            Locality:     []string{"Istanbul"},
        },
        NotBefore:             time.Now().Add(-time.Hour),
        NotAfter:              time.Now().AddDate(10, 0, 0),
        KeyUsage:              x509.KeyUsageCertSign | x509.KeyUsageCRLSign,
        BasicConstraintsValid: true,
        IsCA:                  true,
        MaxPathLen:            2,
    }
    rootDER, _ := x509.CreateCertificate(rand.Reader, &rootTmpl, &rootTmpl,
&rootKey.PublicKey, rootKey)
    savePEM("certs/rootCA.pem", "CERTIFICATE", rootDER)
    savePEM("certs/rootCA.key", "RSA PRIVATE KEY",
x509.MarshalPKCS1PrivateKey(rootKey))
    fmt.Println("Created root CA")

    // 2) Intermediate CA (signed by Root)
    interKey, _ := rsa.GenerateKey(rand.Reader, 3072)
    interTmpl := x509.Certificate{
        SerialNumber: big.NewInt(2),
        Subject: pkix.Name{
            CommonName:   "Example Intermediate CA",
            Organization: []string{"Example Inter Org"},
        },
        NotBefore:             time.Now().Add(-time.Hour),
        NotAfter:              time.Now().AddDate(5, 0, 0),
        KeyUsage:              x509.KeyUsageCertSign |
x509.KeyUsageDigitalSignature,
        BasicConstraintsValid: true,
        IsCA:                  true,
        MaxPathLen:            1,
    }
    interDER, _ := x509.CreateCertificate(rand.Reader, &interTmpl,
&rootTmpl, &interKey.PublicKey, rootKey)
    savePEM("certs/intermediateCA.pem", "CERTIFICATE", interDER)
    savePEM("certs/intermediateCA.key", "RSA PRIVATE KEY",
x509.MarshalPKCS1PrivateKey(interKey))
    fmt.Println("Created intermediate CA")

    // 3) Server cert (signed by Intermediate) with SANs
    serverKey, _ := rsa.GenerateKey(rand.Reader, 2048)
    serverTmpl := x509.Certificate{
        SerialNumber: big.NewInt(10),
```

```
        Subject: pkix.Name{
            CommonName:   "localhost",
            Organization: []string{"Example Server"},
        },
        DNSNames:    []string{"localhost"},
        IPAddresses: nil,
        NotBefore:   time.Now().Add(-time.Hour),
        NotAfter:    time.Now().AddDate(1, 0, 0),
        KeyUsage:    x509.KeyUsageDigitalSignature |
x509.KeyUsageKeyEncipherment,
        ExtKeyUsage: []x509.ExtKeyUsage{x509.ExtKeyUsageServerAuth},
    }
    serverDER, _ := x509.CreateCertificate(rand.Reader, &serverTmpl,
&interTmpl, &serverKey.PublicKey, interKey)
    savePEM("certs/server.pem", "CERTIFICATE", serverDER)
    savePEM("certs/server.key", "RSA PRIVATE KEY",
x509.MarshalPKCS1PrivateKey(serverKey))
    fmt.Println("Created server cert")

    // 4) Client cert (signed by Intermediate)
    clientKey, _ := rsa.GenerateKey(rand.Reader, 2048)
    clientTmpl := x509.Certificate{
        SerialNumber: big.NewInt(11),
        Subject: pkix.Name{CommonName: "client.local", Organization:
[]string{"Example Client"}},
        NotBefore:   time.Now().Add(-time.Hour),
        NotAfter:    time.Now().AddDate(1, 0, 0),
        KeyUsage:    x509.KeyUsageDigitalSignature,
        ExtKeyUsage: []x509.ExtKeyUsage{x509.ExtKeyUsageClientAuth},
    }
    clientDER, _ := x509.CreateCertificate(rand.Reader, &clientTmpl,
&interTmpl, &clientKey.PublicKey, interKey)
    savePEM("certs/client.pem", "CERTIFICATE", clientDER)
    savePEM("certs/client.key", "RSA PRIVATE KEY",
x509.MarshalPKCS1PrivateKey(clientKey))
    fmt.Println("Created client cert")
}
```

---

## verify_chain.go

```
package main

import (
    "crypto/x509"
    "encoding/pem"
    "fmt"
    "os"
)
```

```
func load(path string) *x509.Certificate {
    b, _ := os.ReadFile(path)
    blk, _ := pem.Decode(b)
    cert, _ := x509.ParseCertificate(blk.Bytes)
    return cert
}

func main() {
    root := load("certs/rootCA.pem")
    inter := load("certs/intermediateCA.pem")
    client := load("certs/client.pem")

    roots := x509.NewCertPool()
    roots.AddCert(root)
    ints := x509.NewCertPool()
    ints.AddCert(inter)

    opts := x509.VerifyOptions{Roots: roots, Intermediates: ints}

    chains, err := client.Verify(opts)
    if err != nil {
        fmt.Println("verify error:", err)
        os.Exit(1)
    }
    fmt.Println("verified chains:")
    for _, ch := range chains {
        for _, c := range ch {
            fmt.Println(" -", c.Subject.CommonName, c.Subject.Organization)
        }
    }
}
```

---

## Dockerfile

```
FROM golang:1.22-alpine
WORKDIR /app
COPY . .
RUN go mod init x509-pkix-full-project || true && go mod tidy
CMD ["sh", "-c", "go run generate_pkix_certs.go && go run verify_chain.go"]
```

---

## Makefile

```
build:
    docker build -t x509-pkix-full-project .
```

```
run:
    docker run --rm x509-pkix-full-project

all: build run
```

---

## README.md

Proje nasıl çalıştırılır, Docker ile nasıl build/run yapılacağı ve hangi dosyaların oluşturulduğu açıklanır.

---

Hazır — istersen bu projeyi şimdi Docker ortamında çalıştırman için adım adım talimat vereyim veya proje dosyalarını bir arşiv halinde sunayım.