Web安全第三次作业

陈铭涛

16340024

## X.509证书描述

在密码学中，X.509是公钥证书的格式标准，在许多互联网协议中得到应用。X.509对公钥证书的格式，撤销证书列表（CRLs），证书验证路径算法等进行了规定。

一个X.509证书中包含了其版本号，证书序列号，签名算法，签发者，证书主体，有效期，公钥，公钥密钥等信息。证书中的信息使用 ASN.1进行编码，ASN.1中数据以tag，长度，值的方式进行编码。证书的基本结构在RFC 5280中4.1节进行了如下的规定：

Certificate ::= SEQUENCE {

tbsCertificate TBSCertificate,

signatureAlgorithm AlgorithmIdentifier,

signatureValue BIT STRING }

（证书主体，签名算法以及签名值）

TBSCertificate ::= SEQUENCE {

version [0] EXPLICIT Version DEFAULT v1,

serialNumber CertificateSerialNumber,

signature AlgorithmIdentifier,

issuer Name,

validity Validity,

subject Name,

subjectPublicKeyInfo SubjectPublicKeyInfo,

issuerUniqueID [1] IMPLICIT UniqueIdentifier OPTIONAL,

-- If present, version MUST be v2 or v3

subjectUniqueID [2] IMPLICIT UniqueIdentifier OPTIONAL,

-- If present, version MUST be v2 or v3

extensions [3] EXPLICIT Extensions OPTIONAL

-- If present, version MUST be v3

}

（证书主体，包含版本号，序列号，签名算法标识，签发者信息，有效期，证书主体，证书公钥信息，签发者 ID，主体 ID 以及扩展段）

Version ::= INTEGER { v1(0), v2(1), v3(2) }

（证书版本，值可为0，1，2，分别代表版本1，2，3）

CertificateSerialNumber ::= INTEGER

（证书序列号）

Validity ::= SEQUENCE {

notBefore Time,

notAfter Time }

（证书有效期，由开始和结束时间组成）

Time ::= CHOICE {

utcTime UTCTime,

generalTime GeneralizedTime }

UniqueIdentifier ::= BIT STRING

SubjectPublicKeyInfo ::= SEQUENCE {

algorithm AlgorithmIdentifier,

subjectPublicKey BIT STRING }

（公钥信息包含公钥算法和公钥数据）

Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension

Extension ::= SEQUENCE {

extnID OBJECT IDENTIFIER,

critical BOOLEAN DEFAULT FALSE,

extnValue OCTET STRING

-- contains the DER encoding of an ASN.1 value

-- corresponding to the extension type identified

-- by extnID

}

AlgorithmIdentifier ::= SEQUENCE {

algorithm OBJECT IDENTIFIER,

parameters ANY DEFINED BY algorithm OPTIONAL }

#### 读取 X.509证书

提交的程序中使用 Go 语言编写读取 X.509的程序，调用了 Go 语言的encoding/asn1库进行 ASN.1编码内容的读取以及crypto/x509/pkix库进行对签发者与证书主体信息的读取。

在代码中根据上面的 X.509证书结构定义了如下的结构体用于ASN.1读取：

1. CertificateData, 对应上述结构中的 Certificate：

type CertificateData struct {

    TBSCertificate tbsCertificate

    SignatureAlgorithm AlgorithmIdentifier

    SignatureValue asn1.BitString

}

1. TbsCertificate, 对应上述结构中的 TBSCertificate:

type tbsCertificate struct {

    Version int `asn1:"optional,explicit,default:0,tag:0"`

    SerialNumber \*big.Int

    Signature AlgorithmIdentifier

    Issuer asn1.RawValue

    Validity timeSpan

    Subject asn1.RawValue

    PublicKey publicKeyInfo

    UniqueId asn1.BitString `asn1:"optional,tag:1"`

    SubjectUniqueId asn1.BitString `asn1:"optional,tag:2"`

    Extensions []extension `asn1:"optional,explicit,tag:3"`

}

1. timeSpan, 对应上述结构的Validity:

type timeSpan struct {

    NotBefore, NotAfter time.Time

}

1. publicKeyInfo, 对应上述的 SubjectPublicKeyInfo:

type publicKeyInfo struct {

    Algorithm AlgorithmIdentifier

    PublicKey asn1.BitString

}

1. extension, 对应上述结构的Extension

type extension struct {

    ExtnID asn1.ObjectIdentifier

    Critical bool `asn1:"default:false"`

    ExtnValue []byte

}

1. AlgorithmIdentifier， 对应上述的 AlgorithmIdentifier:

type AlgorithmIdentifier struct {

    Algorithm asn1.ObjectIdentifier

    Parameters asn1.RawValue `asn1:"optional"`

}

#### 程序结构简述

提交的代码中包含三个代码文件：x509cert/certificate.go 为证书结构体的定义以及对证书中算法的识别函数，并定义了如下的certInfo 结构方便其他代码获取证书信息：

type CertInfo struct {

    Version int

    Serial \*big.Int

    Signature AlgorithmIdentifier

    Issuer IssuerType

    Validity timeSpan

    Subject IssuerType

    PublicKey publicKeyInfo

    UniqueId asn1.BitString

    SubjectUniqueId asn1.BitString

    Extensions []extension

    SignatureAlgorithm AlgorithmIdentifier

    SignatureValue asn1.BitString

}

type IssuerType struct {

    Country string

    Province string

    City string

    Organization string

    Unit string

}

x509cert/static.go 文件包含了证书读取过程中的一些静态数据如算法的名称与 oid 等。

main.go 文件包含程序的主函数，将根据 pem 格式或 DER 格式读取文件并将证书信息打印至标准输出。

#### 程序编译运行结果

程序使用go 进行编译，在代码目录下执行命令go build –o x509即可编译生成命名为 x509的可执行文件。提交的 bin 文件夹中已包含了使用go1.11.1编译获得的 Windows，MacOS 以及 Linux 下的可执行文件。

程序的使用方法为：

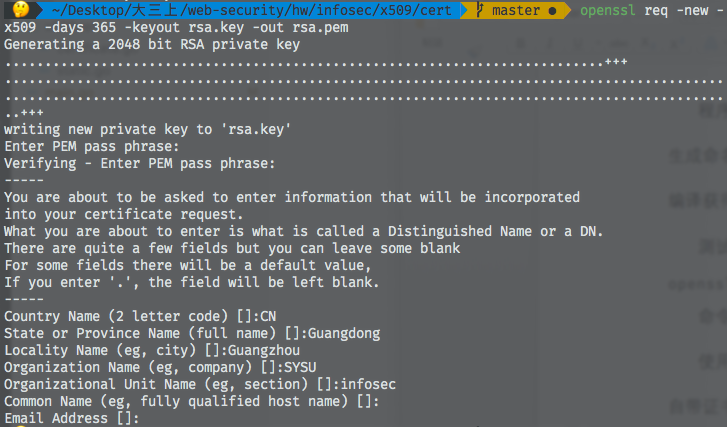
./x509 [--DER] filename

其中若指定了 DER 选项，则程序将会使用 DER 方式读取证书，若未指定则程序会使用 PEM 方式读取证书。

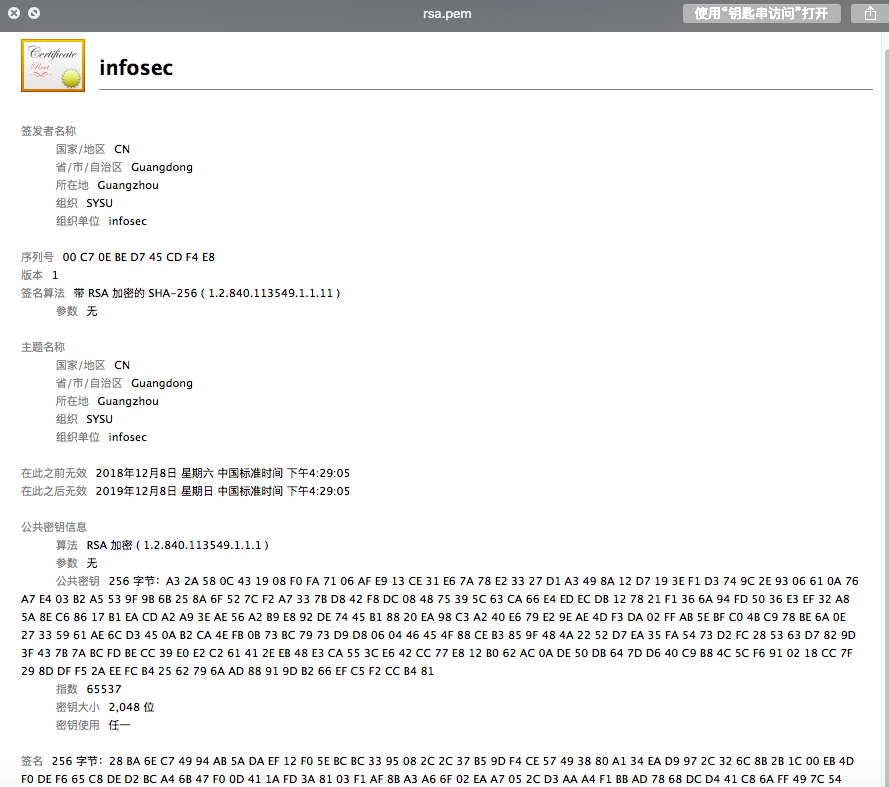
测试运行使用的证书使用 openssl 生成自签名根证书，命令如下：

openssl req -new -x509 -days 365 -keyout rsa.key -out rsa.pem

命令执行后需要输入证书主体信息：



使用该命令会使用RSA 算法签名生成有效期为365天的根证书，使用系统自带证书工具查看证书信息如下：



使用如下命令将上面命令生成的 pem 证书转换为DER 证书进行测试：

openssl x509 -in rsa.pem -outform der -out rsa.crt

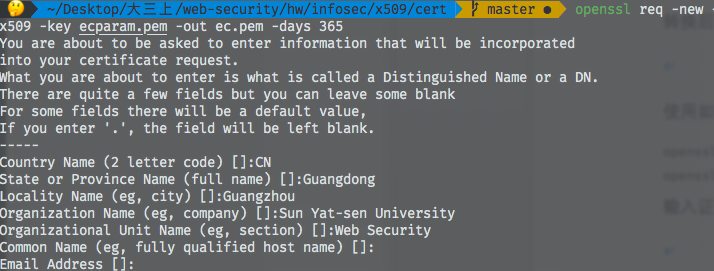
转换后生成 rsa.crt 证书文件，查看结果与原证书相同。

使用如下命令生成ECDSA签名算法的证书用于测试：

openssl ecparam -name secp256k1 -genkey -param\_enc explicit -out ecparam.pem

openssl req -new -x509 -key ecparam.pem -out ec.pem -days 365

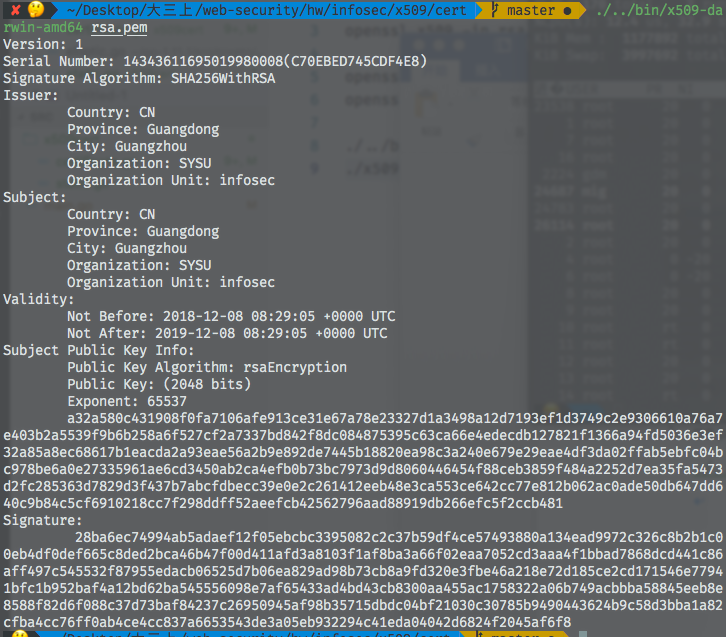
输入证书主体信息后生成证书：



使用编写的程序在证书目录下执行的效果如图：

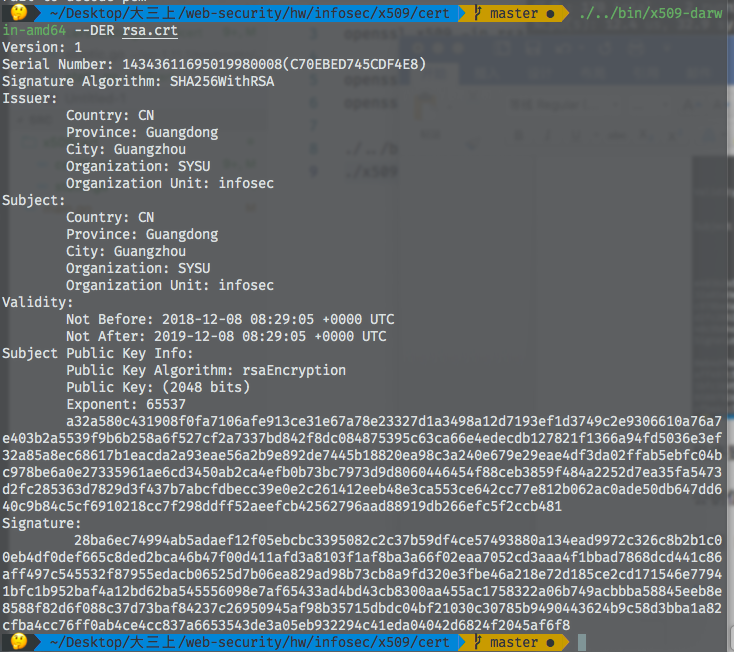
1. rsa.pem

./../bin/x509-darwin-amd64 rsa.pem



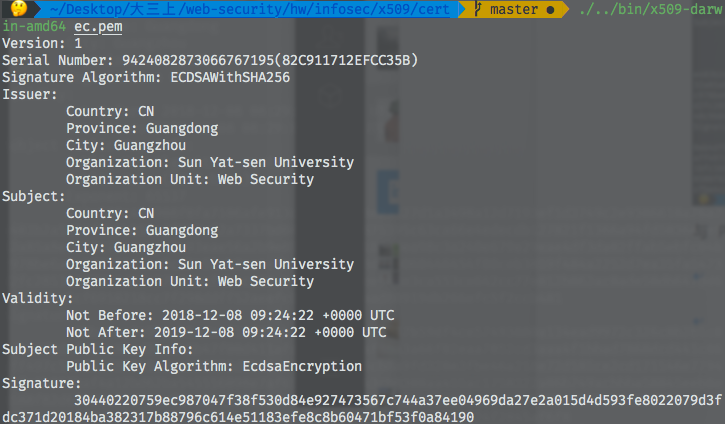
程序显示内容为证书的版本号，序列号，签名算法，签发者，主体，有效期，公钥信息，RSA 公钥数据以及证书签名，与系统工具显示的证书内容相同。

使用 DER 证书文件进行读取的结果如图：



与 PEM 模式下证书读取获得的结果相同。

读取ecdsa 加密的证书结果如下：



与生成证书时的信息相同。