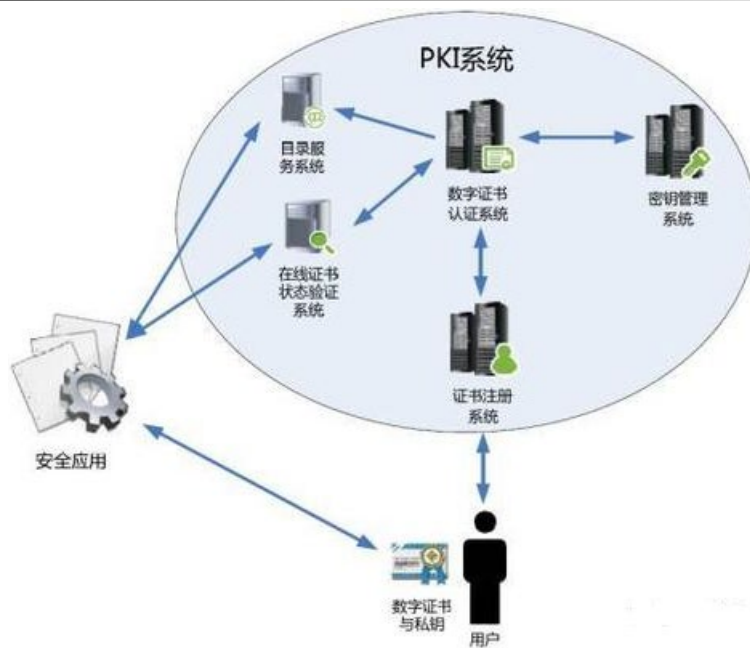




声 明

- 本PPT是电子工业出版社出版的教材《计算机网络安全原理》配套教学PPT（部分内容的深度和广度在教材的基础上有所扩展），作者：吴礼发
 - 本PPT可能直接或间接采用了网上资源、公开学术报告中的部分PPT页面、图片、文字，引用时我们力求在该PPT的备注栏或标题栏中注明出处，如果有疏漏之处，敬请谅解。同时对被引用资源或报告的作者表示诚挚的谢意！
 - 本PPT可免费使用、修改，使用时请保留此页。
-

第四章 PKI与数字证书





内容提纲

1

密钥管理

2

数字证书

3

PKI

4

证书透明性





密钥管理

- 基于**密钥保护**的安全策略
- 密钥管理包括密钥的**产生、存储、分发、组织、使用、停用、更换、销毁**等一系列问题，涉及每个密钥的从产生到销毁的整个生命周期。





密钥分发

- 人工方法：
 - 选择一个密钥后以物理的方式传送给B。
 - 第三方选择密钥后以物理的方式传送给A和B。
 - 如果A和B先前或者最近使用过一个密钥，则一方可以将新密钥用旧密钥加密后发送给另一方。
 - 如果A和B到第三方C（密钥分配中心，KDC）有加密连接，C可以通过该加密连接将密钥传送给A和B。
-



公开密钥的管理

- 公开密码体制的密钥管理：将产生的公开密钥安全地发送给相关通信参与方的技术和方法
 - **问题：**如果公钥的完整性和真实性受到危害，则基于公钥的各种应用的安全性将受到危害。
 - 在公钥管理的过程中采取了**将公钥和公钥所有人信息绑定**的方法，这种绑定产生的就是用户**数字证书**（Digital Certificate, DC）





内容提纲

1

密钥管理

2

数字证书

3

PKI

4

证书透明性





数字证书

- 数字证书是一种由一个可信任的权威机构（CA）签署的信息集合。在不同的应用中有不同的证书，如公钥证书（Public Key Certificate, PKC）、PGP证书、SET证书等。





数字证书

■ 公钥证书

- 公钥证书主要用于确保公钥及其与用户绑定关系的安全，一般包含持证主体身份信息、主体的公钥信息、CA信息以及附加信息，再加上用CA私钥对上述信息的数字签名。目前应用最广泛的证书格式是国际电信联盟（International Telecommunication Union, ITU）制定的X.509标准中定义的格式



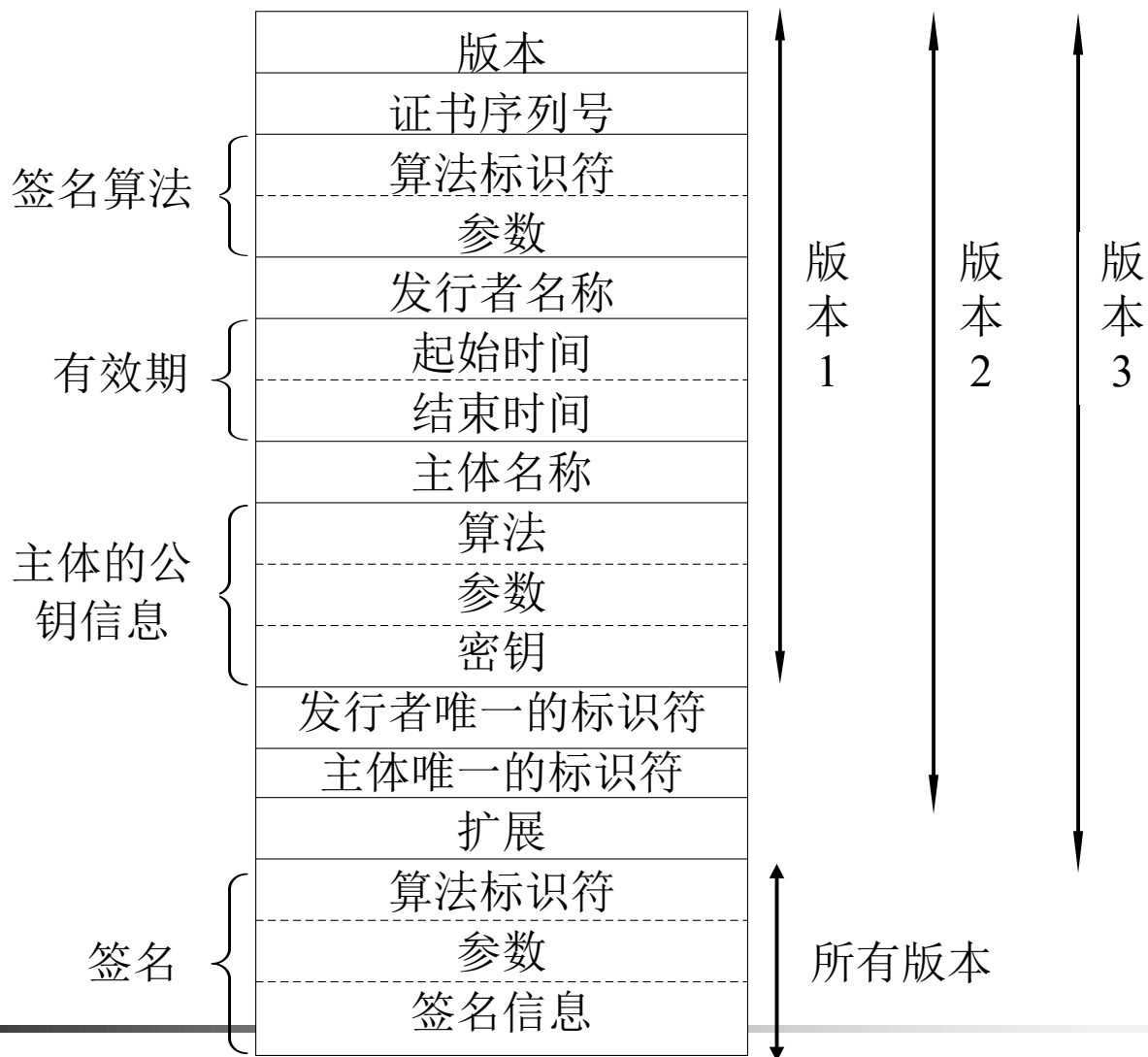


X.509证书

- X.509最初是在1988年的7月3日发布的，版本是X.509 v1，当时是作为ITU X.500目录服务标准的一部分。在此之后，ITU分别于1993年和1995年进行过两个修改，分别形成了X.509 版本2 (X.509 v2)和版本3 (X.509 v3)，其中v2证书并未得到广泛使用



X.509 数字证书



360浏览器中的数字证书

证书

×

预期目的(N):

<所有>

个人 其他人 中间证书颁发机构 受信任的根证书颁发机构 受信任的发布者 未受信任的发布者

颁发给	颁发者	截止日期	友好名称
GlobalSign Orga...	GlobalSign Root CA	2024/2/...	<无>
Go Daddy Secur...	Go Daddy Root C...	2031/5/3	<无>
GoGetSSL RSA D...	USERTrust RSA Ce...	2028/9/6	<无>
ICBC Corporate ...	ICBC Root CA	2030/5/...	<无>
ICBC Corporate ...	ICBC Root CA	2030/5/...	<无>
ICBC Personal S...	ICBC Root CA	2030/5/...	<无>
ICBC Personal S...	ICBC Root CA	2030/5/...	<无>
InCommon RSA ...	USERTrust RSA Ce...	2024/10...	<无>
Microsoft Secur...	Microsoft Root Ce...	2026/10...	<无>
Microsoft TPM ...	Microsoft TPM Ro...	2039/12...	<无>

证书管理器

×

信任的根证书

360浏览器目前已信任如下证书

证书名称	有效期
Global Digital Cybersecurity Authority Co., Ltd.	
GlobalSign	
GlobalSign	2009/03/18-2029/03/18
GlobalSign	2012/11/13-2038/01/19
GlobalSign	2014/12/10-2034/12/10
GlobalSign	2006/12/15-2021/12/15
GlobalSign	2012/11/13-2038/01/19
GlobalSign nv-sa	
Google Trust Services LLC	

导入(I)...

导出(E)...

删除(R)

证书的预期目的

服务器身份验证, 客户端身份验证

证书保存格式

← 证书导出向导

导出文件格式

可以用不同的文件格式导出证书。

选择要使用的格式:

☒ DER 编码二进制 X.509 (.CER)(D)

☐ Base64 编码 X.509(.CER)(S)

☐ 加密消息语法标准 - PKCS #7 证书(.P7B)(C)

☐ 如果可能, 则包括证书路径中的所有证书(I)

☐ 个人信息交换 - PKCS #12(.PFX)(P)

☐ 如果可能, 则包括证书路径中的所有证书(U)

☐ 如果导出成功, 删除私钥(K)

☐ 导出所有扩展属性(A)

☐ 启用证书隐私(E)

☐ Microsoft 系列证书存储(.SST)(T)

X.509 certificates, as well as many other things in the X.509 standard, are described using [Abstract Syntax Notation One \(ASN.1\)](#). ASN.1 is a standard used to exchange information between systems independently of the systems' encoding techniques. ASN.1 have several encoding rules:

- Basic Encoding Rules (BER)
- Canonical Encoding Rules (CER)
- Distinguished Encoding Rules (DER)
- XML Encoding Rules (XER)
- Canonical XML Encoding Rules (CXER)
- Extended XML Encoding Rules (E-XER)
- Packed Encoding Rules (PER, unaligned: UPER, canonical: CPER)
- Generic String Encoding Rules (GSER)

The original rules laid out for the ASN.1 standard were Basic Encoding Rules (BER), and CER and DER are more strict variants of BER. Digital certificates are usually stored in the file system as raw binary data, so DER (binary) is the most common. Certificates stored as raw binary usually have a .cer extension, but .der is also in use. Often the binary data is converted to Base64 ASCII files. This is called Privacy Enhanced Email (PEM), and these files commonly have one of these extensions: .pem, .crt, .cer, and .key.

下一步(N)

取消

证书保存格式

a DER encoded
certificate
opened in a HEX editor

```
0...0..ñ .....n'5F.0µ.MYùññiæ0..  
.+. ....0.1.0...U....Morgan Simons  
en0...130416085717Z..391231235959Z  
0.1.0...U....Morgan Simonsen0.."0.  
..*.H÷.....0.....~íÄ...c  
0%P..ÄÄö.#É qxnÖÈ)..v¶..3¿É.=É...`  
äðw@á'Go±5.ÓÐRÇ`..él<aÄ6=§ð2.P<Äy  
b'°K$xc.j)@í.zð...{²SÑ...Üëû.MÄ@Ba  
~+Ý.µe³¼'¼XÉí.í.&.yÄ..'.bóp.«3'@i{  
.±qv'.{Éa.p.. 'IÉFBüDgë.ê|)1ÉL2.'l  
Ý.Y.Qjæ@úêN²@d!.k..ð|&n-l..6Ü.aéf(  
.ø..ííÄîTî²f.ruqYüüfMS>"qG.$âQ(6..  
k.É7.àÑ.g.....f00M0K...U...D0B..'`  
Dìµ._Tøîè{ð.É.'j.0.1.0...U....Morg  
an Simonsen..n'5F.0µ.MYùññiæ0...+  
.....«É|.8.ÉAe.¿..9ÿ-§çQ...o  
/.jsäPíí..ÿ...@|'.íÿ.ðñfí÷.Cçý..G.  
¶'Q7JLc-1.é.²ð%.Bw%.0²¶.].C.4sîð-  
.ðj¿".KMÜ*²C5G..ßñ|.U=f=Y...Cäi-  
n÷..NNâ/&.. @4âT.â²_üthò!xu.âvÑ¶«7  
]Nà-ß.güó.i1ðç-ç>jµ)..6|x2;».a.~+Ü  
.Äç§"6m.A=+hÜ&mcü`.¼fdð¼(1Ä.'...8x.  
..^..Ý.vj>...bN.'î.é6.-²ð_¥U.
```

the same cert encoded as Base64
also opened in a HEX editor

```
-----BEGIN CERTIFICATE----- MIIDBTCC  
AFGgAwIBAgIQbpI1Rg7btZRNWfnxqPHP5jAJ  
BgUrDgMCHQUAMBoxGDAw.BgNVBAMTD01vcmd  
hbiBTaw1vbnNlbjAeFw0xMZA0MTYwODU3MTd  
aFw0ZOTEyMZEy.MZU5NTlambBoxGDAwBgNVBA  
MTD01vcmdhbiBTaw1vbnNlbjCCASIwDQYJKo  
ZIhvcN.AQEBBQADggEPADCCAQoCggEBAKztw  
x0Rf2PbJVAumsbB9bcjyKBxpG7wyCkXj3a2.  
jIgzv8kOPcgNhxFg5PB3ruw0R2+XNzjTRNBS  
x2Auf+lsPGHCNj2n9TKI3jzEewKR.sEskeKI  
uaimp7g562A2eEnuyU9EXjAHC6/sYTCcu32F  
+K90VtwwzVklWmntnu+f.Jpt5w44Tkp5i8/  
6Nqz00QKF7DrFxvrSde8thnXAdnBRjYUzC/G  
REZ+uL6nwpMctM.MhKRBN0EwQdRauZA+upos  
q5kIS5rAJnwfcZurwvGDbcgwHPzih/+ImC7  
u3F7lTu.qs0BcnVxwf38zu1TPiJxR38k5VEo  
NhIJaw2vyTeb4NEAZxECawEAAanPMPE0wSwYD  
.VR0BBEQwQoAQETstZdfVPju6HvQHsmbKqE  
cmBoxGDAwBgNVBAMTD01vcmdhbiBT.aw1vbn  
NlboIQbpI1Rg7btZRNWfnxqPHP5jAJBgUrDg  
MCHQUAA4IBAQCry3yTOILJ.Qww4v4yEOf+sp  
+dRG46vby8DanPjUM/NC5T/uAUIQKawn8//A  
NDxZu73gkPn/ZSo.R4G2tFE3SkxjLTEO6Zqy  
Or0KQnc1k9uqtp5dgkOZNHPu0K0dhvwhvyke  
S03ckPeY.QzVHHQDf8XwRA1U9oz1ZnQseQ+D  
vrG73hphoTuIvJp4IoK404lQS5bjf3Hro0ih  
X.VZbiVtGkqzddTuAt37dn/PMfaTHQxy2iPm  
q1KYEINqZ4Mju7EGEYryvcFCTnpyI2.byjAP  
Sto3CZtY/xgBL6jZNa+KGZDGLkfntjXgLCox  
pSE3ZN2ft4ZGRhiThaSzJzp.Nhkts/JfpvWL  
-----END CERTIFICATE-----.
```

证书保存格式

Finally here is the same certificate in ASN.1 human readable form (this isn't the whole cert)

```
0000: 30 82 03 05          ; SEQUENCE (305 Bytes)
0004: 30 82 01 f1          ; SEQUENCE (1f1 Bytes)
0008: | a0 03              ; OPTIONAL[0] (3 Bytes)
000a: | | 02 01            ; INTEGER (1 Bytes)
000c: | | 02              ;
000d: | 02 10              ; INTEGER (10 Bytes)
000f: | | 6e 92 35 46 0e db b5 94 4d 59 f9 f1 a8 f1 cf e6
001f: | 30 09              ; SEQUENCE (9 Bytes)
0021: | | 06 05            ; OBJECT_ID (5 Bytes)
0023: | | | 2b 0e 03 02 1d
      | | |           ; 1.3.14.3.2.29 sha1RSA (shaRSA)
0028: | | 05 00            ; NULL (0 Bytes)
002a: | 30 1a              ; SEQUENCE (1a Bytes)
002c: | | 31 18            ; SET (18 Bytes)
002e: | | 30 16            ; SEQUENCE (16 Bytes)
0030: | | 06 03            ; OBJECT_ID (3 Bytes)
0032: | | | 55 04 03
      | | |           ; 2.5.4.3 Common Name (CN)
0035: | | 13 0f            ; PRINTABLE_STRING (f Bytes)
0037: | | 4d 6f 72 67 61 6e 20 53 69 6d 6f 6e 73 65 6e      ; Morgan Simonsen
      | |           ; "Morgan Simonsen"
0046: | 30 1e              ; SEQUENCE (1e Bytes)
0048: | | 17 0d            ; UTC_TIME (d Bytes)
004a: | | | 31 33 30 34 31 36 30 38 35 37 31 37 5a          ; 130416085717Z
      | | |           ; 16.04.2013 10:57
0057: | | 17 0d            ; UTC_TIME (d Bytes)
```

数字证书

证书管理器



信任的根证书

360浏览器目前已信任如下证书

证书名称	有效期
SSL.com Root Certification Authority RSA	2016/02/13-2041/02/13
> SecureTrust Corporation	
> The USERTRUST Network	
> UniTrust	
> Unizeto Technologies S.A.	
v Verizon Business	
Verizon Global Root CA	2009/07/30-2034/07/30
> WiSeKey	
> XRamp Security Services Inc	

数字证书

证书

常规

详细信息

证书路径



证书信息

这个证书的目的如下:

- 保证远程计算机的身份
- 向远程计算机证明你的身份
- 保护电子邮件消息
- 确保软件来自软件发布者

颁发给: Verizon Global Root CA

颁发者: Verizon Global Root CA

有效期从 2009/7/30 到 2034/7/30

证书

常规

详细信息

证书路径

显示(S):

<所有>

字段

值

版本

V3

序列号

01

签名算法

sha256RSA

签名哈希算法

sha256

颁发者

Verizon Global Root CA, ...

有效期从

2009年7月30日 22:27:04

到

2034年7月30日 22:27:04

使用者

Verizon Global Root CA, ...

密钥

RSA (2048 Bits)

编辑属性(E)...

复制到文件(C)...

数字证书

证书

常规

详细信息

证书路径

显示(S): <所有>

字段	值
到期	2034年7月30日 22:27:04
使用者	Verizon Global Root CA, ...
公钥	RSA (2048 Bits)
公钥参数	05 00
使用者密钥标识符	4c3811b898005b5a2b703...
基本约束	Subject Type=CA, Path Le...
密钥用法	Certificate Signing, Off-lin...
指纹	912198eef23dcac4093931...

30 82 01 0a 02 82 01 01 00 8a 00 0c 70 1d bf eb 34 86 c3 99 45 35
1e 7f 43 f7 ab 6f 24 2d cd 19 c2 10 bb b0 ca 29 5b a9 20 ab ab 72
2c c4 e2 02 39 6d 82 b8 c5 11 ea f8 fb b3 9e 62 f8 33 1e b0 1f c9 e3
f6 37 db 04 c8 3b 63 4f 36 e2 85 a4 25 1d c7 69 1f 04 bd 68 45 13
96 07 1f 94 50 f5 3e c5 27 54 9e c0 49 57 44 8e 07 63 d4 a6 ae ed
22 99 cc 4d 96 69 04 13 6e 76 89 9f 74 16 94 f9 1d 54 bd a2 b9 d2
83 01 22 0c 4d 44 80 aa fe 35 89 27 25 a7 86 89 c6 d5 1a 92 e3 8f
c5 95 a0 14 72 9a e8 56 c5 02 55 1c 97 f9 20 2e d0 f5 3c 13 19 5a f6
e1 f9 0b 03 82 69 a7 8c b7 d6 6f 9c 56 3e 9d e8 2a 09 60 6d 4b e6

编辑属性(E)...

复制到文件(C)...

证书

常规

详细信息

证书路径

显示(S): <所有>

字段	值
到期	2034年7月30日 22:27:04
使用者	Verizon Global Root CA, ...
公钥	RSA (2048 Bits)
公钥参数	05 00
使用者密钥标识符	4c3811b898005b5a2b703...
基本约束	Subject Type=CA, Path Le...
密钥用法	Certificate Signing, Off-lin...
指纹	912198eef23dcac4093931...

912198eef23dcac40939312fee97dd560bae49b1

编辑属性(E)...

复制到文件(C)...

数字证书

证书管理器

信任的根证书

360浏览器目前已信任如下证书

证书名称	有效期
▼ SSL Corporation	
SSL.com EV Root Certification Authority RSA R2	2017/06/01-2042/05/31
SSL.com EV Root Certification Authority ECC	2016/02/13-2041/02/13
SSL.com Root Certification Authority ECC	2016/02/13-2041/02/13
SSL.com Root Certification Authority RSA	2016/02/13-2041/02/13
> SecureTrust Corporation	
> The USERTRUST Network	
> UniTrust	
> Unizeto Technologies S.A.	
▼ Verizon Business	

数字证书

证书

常规 详细信息 证书路径

证书信息

这个证书的目的如下:

- 保证远程计算机的身份
- 向远程计算机证明你的身份
- 保护电子邮件消息
- 确保软件来自软件发布者

颁发给: SSL.com EV Root Certification Authority ECC

颁发者: SSL.com EV Root Certification Authority ECC

有效期从 2016/2/13 到 2041/2/13

证书

常规 详细信息 证书路径

显示(S): <所有>

字段	值
版本	V3
序列号	2c299c5b16ed0595
签名算法	sha256ECDSA
签名哈希算法	sha256
颁发者	SSL.com EV Root Certifica...
有效期从	2016年2月13日 2:15:23
到	2041年2月13日 2:15:23
使用者	SSL.com EV Root Certifica...
公钥	ECC (256 Bit)

编辑属性(E)...

复制到文件(C)...

数字证书

证书

常规

详细信息

证书路径

显示(S): <所有>

字段	值
使用者	SSL.com EV Root Certifica...
公钥	ECC (384 Bits)
公钥参数	ECDSA_P384
使用者密钥标识符	5bca5ee5ded281aacda82...
授权密钥标识符	KeyID=5bca5ee5ded281a...
基本约束	Subject Type=CA, Path Le...
密钥用法	Digital Signature, Certifica...
指纹	4cdd51a3d1f5203214b0c...

04 aa 12 47 90 98 1b fb ef c3 40 07 83 20 4e f1 30 82 a2 06 d1 f2 92 86 61 f2 f6 21 68 ca 00 c4 c7 ea 43 00 54 86 dc fd 1f df 00 b8 41 62 5c dc 70 16 32 de 1f 99 d4 cc c5 07 c8 08 1f 61 16 07 51 3d 7d 5c 07 53 e3 35 38 8c df cd 9f d9 2e 0d 4a b6 19 2e 5a 70 5a 06 ed be f0 a1 b0 ca d0 09 29

编辑属性(E)...

复制到文件(C)...

证书

常规

详细信息

证书路径

显示(S): <所有>

字段	值
使用者	SSL.com EV Root Certifica...
公钥	ECC (384 Bits)
公钥参数	ECDSA_P384
使用者密钥标识符	5bca5ee5ded281aacda82...
授权密钥标识符	KeyID=5bca5ee5ded281a...
基本约束	Subject Type=CA, Path Le...
密钥用法	Digital Signature, Certifica...
指纹	4cdd51a3d1f5203214b0c...

4cdd51a3d1f5203214b0c6c532230391c746426d

编辑属性(E)...

复制到文件(C)...

数字证书

证书

常规 详细信息 证书路径

显示(S): <所有>

字段	值
版本	V3
序号	2f4b0005
签名算法	sha1RSA
签名哈希算法	sha1
颁发者	Industrial and Commercia...
有效期从	2010年5月17日 18:00:00

字段	值
公钥参数	05 00
Netscape Cert Type	SSL CA, SMIME CA, 签名 ...
授权密钥标识符	KeyID=5f095f2aa344bd5...
CRL 分发点	[1]CRL Distribution Point: ...
密钥用法	Digital Signature, Non-Re...
使用者密钥标识符	1f0db518d378fb9e6dbe8...
基本约束	Subject Type=CA, Path Le...
指纹	2b3edc77574d73a2afcf5a...

2b3edc77574d73a2afcf5a1b031944c1218635bb

证书

常规 详细信息 证书路径

显示(S): <所有>

字段	值
版本	V3
序号	00938bb08e62987b4f75f...
签名算法	sha384RSA
签名哈希算法	sha384
颁发者	USERTrust RSA Certificati...
有效期从	2018年9月6日 8:00:00

字段	值
使用者密钥标识符	f9fb50c48b67bb6764fe83...
增强型密钥用法	服务器身份验证 (1.3.6.1.5.5...
证书策略	[1]Certificate Policy:Policy ...
CRL 分发点	[1]CRL Distribution Point: ...
授权信息访问	[1]Authority Info Access: ...
密钥用法	Digital Signature, Certifica...
基本约束	Subject Type=CA, Path Le...
指纹	070a726c6e4418dcf02138...

070a726c6e4418dcf0213874f0c16d93b041e935

?

关于证书指纹

字段	值
公钥参数	05 00
Netscape Cert Type	SSL CA, SMIME CA, 签名 ...
授权密钥标识符	KeyID=5f095f2aa344bd5...
CRL 分发点	[1]CRL Distribution Point: ...
密钥用法	Digital Signature, Non-Re...
使用者密钥标识符	1f0db518d378fb9e6dbe8...
基本约束	Subject Type=CA, Path Le...
指纹	2b3edc77574d73a2afcf5a...

2b3edc77574d73a2afcf5a1b031944c1218635bb

- 1、浏览器显示的指纹就是证书签名吗？
- 2、**YES**，那为什么长度都是160位？
- 3、**NO**，那它跟证书签名有什么关系？
是证书签名值的一部分还是完全独立的？

关于证书指纹

用openssl读出来的google的证书的数字签名算法和数字签名,证书里没有“指纹”这个字段

Signature Algorithm: sha1WithRSAEncryption

5c:36:99:4c:2d:78:b7:ed:8c:9b:dc:f3:77:9b:f2:76:d2:77:
30:4f:c1:1f:85:83:85:1b:99:3d:47:37:f2:a9:9b:40:8e:2c:
d4:b1:90:12:d8:be:f4:73:9b:ee:d2:64:0f:cb:79:4f:34:d8:
a2:3e:f9:78:ff:6b:c8:07:ec:7d:39:83:8b:53:20:d3:38:c4:
b1:bf:9a:4f:0a:6b:ff:2b:fc:59:a7:05:09:7c:17:40:56:11:
1e:74:d3:b7:8b:23:3b:47:a3:d5:6f:24:e2:eb:d1:b7:70:df:
0f:45:e1:27:ca:f1:6d:78:ed:e7:b5:17:17:a8:dc:7e:22:35:
ca:25:d5:d9:0f:d6:6b:d4:a2:24:23:11:f7:a1:ac:8f:73:81:
60:c6:1b:5b:09:2f:92:b2:f8:44:48:f0:60:38:9e:15:f5:3d:
26:67:20:8a:33:6a:f7:0d:82:cf:de:eb:a3:2f:f9:53:6a:5b:
64:c0:63:33:77:f7:3a:07:2c:56:eb:da:0f:21:0e:da:ba:73:
19:4f:b5:d9:36:7f:c1:87:55:d9:a7:99:b9:32:42:fb:d8:d5:
71:9e:7e:a1:52:b7:1b:bd:93:42:24:12:2a:c7:0f:1d:b6:4d:
9c:5e:63:c8:4b:80:17:50:aa:8a:d5:da:e4:fc:d0:09:07:37:
b0:75:75:21

关于证书指纹

用openssl读出来的apple的证书的数字签名算法和数字签名,证书里也没有“指纹”这个字段

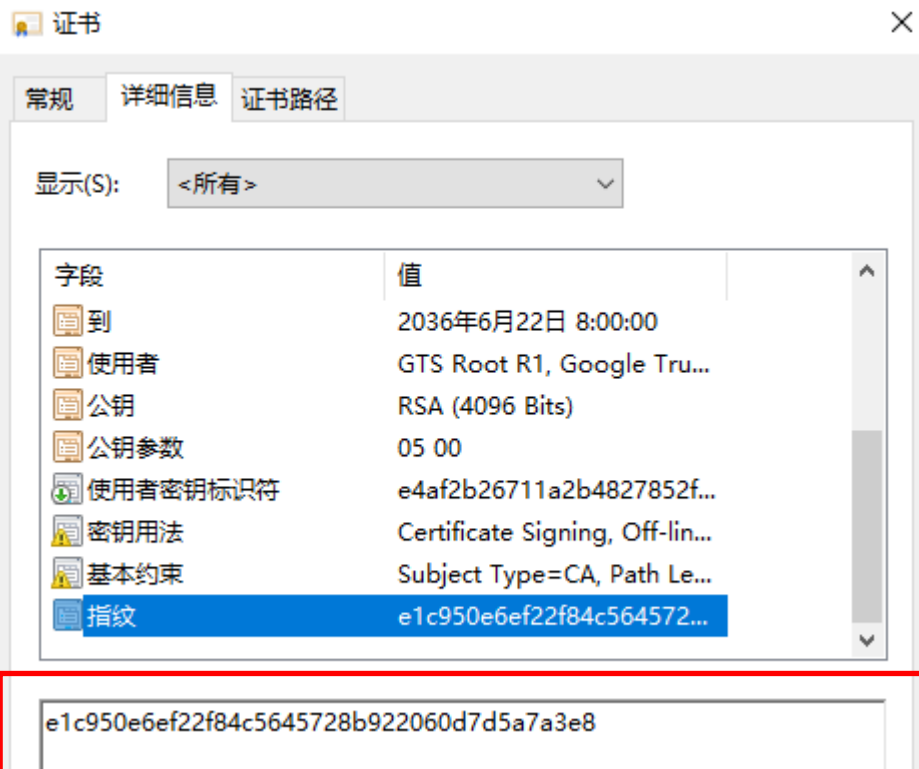
Signature Algorithm: sha384WithRSAEncryption

```
38:96:0a:ee:3d:b4:96:1e:5f:ef:9d:9c:0b:33:9f:2b:e0:ca:
fd:d2:8e:0a:1f:41:74:a5:7c:aa:84:d4:e5:f2:1e:e6:37:52:
32:9c:0b:d1:61:1d:bf:28:c1:b6:44:29:35:75:77:98:b2:7c:
d9:bd:74:ac:8a:68:e3:a9:31:09:29:01:60:73:e3:47:7c:53:
a8:90:4a:27:ef:4b:d7:9f:93:e7:82:36:ce:9a:68:0c:82:e7:
cf:d4:10:16:6f:5f:0e:99:5c:f6:1f:71:7d:ef:ef:7b:2f:7e:
ea:36:d6:97:70:0b:15:ee:d7:5c:56:6a:33:a5:e3:49:38:0c:
b8:7d:fb:8d:85:a4:b1:59:5e:f4:6a:e1:dd:a1:f6:64:44:ae:
e6:51:83:21:66:c6:11:3e:f3:ce:47:ee:9c:28:1f:25:da:ff:
ac:66:95:dd:35:0f:5c:ef:20:2c:62:fd:91:ba:a9:cc:fc:5a:
9c:93:81:83:29:97:4a:7c:5a:72:b4:39:d0:b7:77:cb:79:fd:
69:3a:92:37:ed:6e:38:65:46:7e:e9:60:bd:79:88:97:5f:38:
12:f4:ee:af:5b:82:c8:86:d5:e1:99:6d:8c:04:f2:76:ba:49:
f6:6e:e9:6d:1e:5f:a0:ef:27:82:76:40:f8:a6:d3:58:5c:0f:
2c:42:da:42:c6:7b:88:34:c7:c1:d8:45:9b:c1:3e:c5:61:1d:
d9:63:50:49:f6:34:85:6a:e0:18:c5:6e:47:ab:41:42:29:9b:
f6:60:0d:d2:31:d3:63:98:23:93:5a:00:81:48:b4:ef:cd:8a:
cd:c9:cf:99:ee:d9:9e:aa:36:e1:68:4b:71:49:14:36:28:3a:
3d:1d:ce:9a:8f:25:e6:80:71:61:2b:b5:7b:cc:f9:25:16:81:
e1:31:5f:a1:a3:7e:16:a4:9c:16:6a:97:18:bd:76:72:a5:0b:
9e:1d:36:e6:2f:a1:2f:be:70:91:0f:a8:e6:da:f8:c4:92:40:
6c:25:7e:7b:b3:09:dc:b2:17:ad:80:44:f0:68:a5:8f:94:75:
ff:74:5a:e8:a8:02:7c:0c:09:e2:a9:4b:0b:a0:85:0b:62:b9:
ef:a1:31:92:fb:ef:f6:51:04:89:6c:e8:a9:74:a1:bb:17:b3:
b5:fd:49:0f:7c:3c:ec:83:18:20:43:4e:d5:93:ba:b4:34:b1:
1f:16:36:1f:0c:e6:64:39:16:4c:dc:e0:fe:1d:c8:a9:62:3d:
40:ea:ca:c5:34:02:b4:ae:89:88:33:35:dc:2c:13:73:d8:27:
f1:d0:72:ee:75:3b:22:de:98:68:66:5b:f1:c6:63:47:55:1c:
ba:a5:08:51:75:a6:48:25
```

关于证书指纹

用openssl的命令查看google证书的sha1指纹：

```
[xuweiguangdeMacBook-Pro:x509 xuweiguang$ openssl x509 -sha1 -in google.pem -noout -fingerprint  
SHA1 Fingerprint=E1:C9:50:E6:EF:22:F8:4C:56:45:72:8B:92:20:60:D7:D5:A7:A3:E8
```



与浏览器中看到的指纹一致！

关于证书指纹

指纹并不是证书本身的一个字段，而是浏览器额外计算出来显示的，**进一步验证**

```
with open('apple.pem', 'r') as f:
    s = f.read()
print s
s = s[len('-----BEGIN CERTIFICATE-----') : -len('-----END CERTIFICATE-----')].strip()
print s
s = s.decode('base_64')
from Crypto.Hash import SHA
print SHA.new(s).hexdigest()
```

#输出结果: 611e5b662c593a08ff58d14ae22452d198df6c60

指纹

SHA-256	B0 B1 73 0E CB C7 FF 45 05 14 2C 49 F1 29 5E 6E DA 6B CA ED 7E 2C 68 C5 BE 91 B5 A1 10 01 F0 24
SHA-1	61 1E 5B 66 2C 59 3A 08 FF 58 D1 4A E2 24 52 D1 98 DF 6C 60

结果和浏览器显示的指纹一致 = 整个证书的sha1散列值



关于证书指纹

- 1、浏览器显示的指纹就是证书签名吗？
- 2、**YES**，那为什么长度都是160位？
- 3、**NO**，那它跟证书签名有什么关系？
是证书签名值的一部分还是完全独立的？

指纹并不是证书本身的一个字段，而是浏览器额外计算出来显示的，IE/360浏览器默认用sha1计算指纹(160位)，而google的chrome默认计算了sha1和sha256指纹



关于证书指纹

1、浏览器为什么要计算整个证书的指纹？



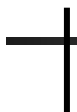


关于证书指纹

1、浏览器为什么要计算整个证书的指纹？

So now we have the answer to why you cannot request a new certificate, or renew an existing one, with the same thumbprint. Changing anything in the certificate data will produce a completely different hash result and thus a completely different thumbprint.

The thumbprints purpose is actually to make it easy to locate a particular certificate in the certificate store of a system. Let's say you have a webserver that needs a certificate. Instead of specifying a certificate by subject name, validity or anything else you just supply the thumbprint to the webserver.





关于证书指纹

2、证书中哪些字段参与哈希签名计算？





关于证书指纹

2、证书中哪些字段参与哈希签名计算？

Field	Definition from RFC 5280
tbsCertificate	The sequence TBSCertificate contains information associated with the subject of the certificate and the CA that issued it. Every TBSCertificate contains the names of the subject and issuer, a public key associated with the subject, a validity period, a version number, and a serial number; some MAY contain optional unique identifier fields.
signatureAlgorithm	The signatureAlgorithm field contains the identifier for the cryptographic algorithm used by the CA to sign this certificate.
signatureValue	The signatureValue field contains a digital signature computed upon the ASN.1 DER encoded tbsCertificate. The ASN.1 DER encoded tbsCertificate is used as the input to the signature function. This signature value is encoded as a BIT STRING and included in the signature field.

The **tbsCertificate** field is by far the largest containing also any extensions the certificate may have like key usage, alternate names etc. RFC 5280 lists all the possible extensions. **signatureAlgorithm** contains only one piece of data; the hashing algorithm used by the signing authority to sign this particular certificate. **signatureValue** contains the signature itself, calculated with the hashing algorithm from **signatureAlgorithm**.

The signature

To produce the certificate signature the signing authority takes the **tbsCertificate** field in ASN.1 DER encoded form (binary data) and applies the hashing algorithm to it. Inside the **tbsCertificate** field are some important fields. Specifically the subject name (CN), the hashing algorithm the signing authority used to sign the certificate and the subject's public key. By signing all these fields the signing authority certifies that the subject in question does in fact own the public key in the certificate. It is a requirement that the **signature** field within the **tbsCertificate** field match the **signatureAlgorithm** field in the certificate. The important distinction here is that it is only the **signature** field *inside* the **tbsCertificate** field that is included in the signature, not the **signatureAlgorithm** field.

关于证书指纹

2、证书中哪些字段参与哈希签名计算？

Field

tbsCertificate

Network Working Group
Request for Comments: 5280
Obsoletes: 3280, 4325, 4630
Category: Standards Track

D. Cooper
NIST
S. Santesson
Microsoft
S. Farrell
Trinity College Dublin
S. Boeyen
Entrust
R. Housley
Vigil Security
W. Polk
NIST
May 2008

largest containing also any
like key usage, alternate names
xtensions. **signatureAlgorithm**
hashing algorithm used by the
lar certificate. **signatureValue**
ted with the hashing algorithm

signatureAlgorithm

Internet X.509 Public Key Infrastructure Certificate
and Certificate Revocation List (CRL) Profile

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

This memo profiles the X.509 v3 certificate and X.509 v2 certificate revocation list (CRL) for use in the Internet. An overview of this approach and model is provided as an introduction. The X.509 v3 certificate format is described in detail, with additional information regarding the format and semantics of Internet name

the signing authority takes the
coded form (binary data) and
inside the **tbsCertificate** field are
the subject name (CN), the
rity used to sign the certificate
ng all these fields the signing
question does in fact own the
quirement that the **signature**
match the **signatureAlgorithm**
t distinction here is that it is only
ificate field that is included in
rithm field.



关于证书指纹

3、证书中哪些字段参与**指纹**计算？

Field	Definition from RFC 5280
tbsCertificate	The sequence TBSCertificate contains information associated with the subject of the certificate and the CA that issued it. Every TBSCertificate contains the names of the subject and issuer, a public key associated with the subject, a validity period, a version number, and a serial number; some MAY contain optional unique identifier fields.
signatureAlgorithm	The signatureAlgorithm field contains the identifier for the cryptographic algorithm used by the CA to sign this certificate.
signatureValue	The signatureValue field contains a digital signature computed upon the ASN.1 DER encoded tbsCertificate. The ASN.1 DER encoded tbsCertificate is used as the input to the signature function. This signature value is encoded as a BIT STRING and included in the signature field.

So what is the thumbprint a hash of? Turns out it is actually the whole certificate, i.e. the binary data representing the three required fields (tbsCertificate, signatureAlgorithm and signature). You can verify this by using a tool that can generate hashes directly from the certificate binary DER file in the file system. In the



关于证书指纹

<https://morgansimonsen.com/2013/04/16/understanding-x-509-digital-certificate-thumbprints/>

Morgan Simonsen's Blog

Information wants to be free!

RECENT POSTS

How to add IIS Request
Filtering Hidden Segments
with PowerShell

Packer for Azure Peculiarities

Migrating blog database from
ClearDB to Azure DB for
MySQL

But I _am_ on the Internet!

Copying Azure Managed disks
between regions

RECENT COMMENTS

sudam on Packer for Azure
Peculiarities

oscar on About

Morgan Simonsen on Backing
up your Windows profile using

[ABOUT](#) [SAMPLE PAGE](#)



SECURITY

UNDERSTANDING X.509 DIGITAL CERTIFICATE THUMBPRINTS

🕒 16/04/2013 👤 MORGAN SIMONSEN 💬 4 COMMENTS

Introduction

I got an interesting question about X.509 certificate thumbprints today from a colleague. Specifically, he wanted to know if you could renew a certificate and keep the thumbprint. The answer is no, unfortunately. So I thought I would explain why you can't.

Certificate storage

关于证书指纹

如何检查数字证书的指纹 - SSL证书数字签名网

我来答

分享

举报

浏览 2231 次

1个回答

#热议# 本科应届生真的人均都月薪过万吗?



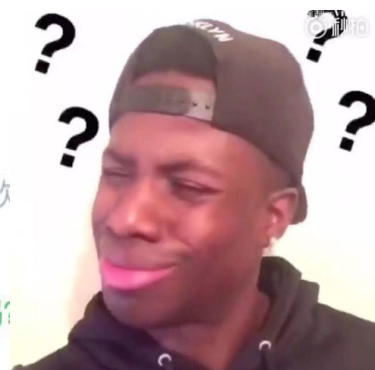
llzcc66

知道合伙人数码行家 2016-12-21

证书的指纹（certificate's thumbprint）是独一无二的标识符。微软Internet Explorer调用它按手印。浏览器倾向于以显示它，就好像它是一个部分的证书。它不是一个证书的一部分，但它是从它计算。

指纹是DER编码的证书信息，这是一个ASN.1类型的X.509规范的一部分指定为MD5摘要。

证书指纹是摘要（散列函数）的X509证书的二进制格式。它可以计算出不同的算法，如SHA1为Microsoft Internet Explorer，Netscape Navigator为MD5。



关于证书指纹



TLS/SSL数字证书里的指纹算法、签名算法和签名哈希算法各是做什么用的？

看到某数字证书里有三个算法：指纹算法sha1、签名算法sha256RSA和签名哈希算法sha256，具体签名过程是怎样的？我的理解是先用指纹算法(也是哈希算法之一)算出证书内容的摘要指纹，再用签名哈希算法对指纹进行二次哈希，最后用颁发机构的RSA私钥签名，是这样么？

字段	值
版本	V3
序列号	04 00 00 00 00 01 44 4e...
签名算法	sha256RSA
签名哈希算法	sha256
颁发者	GlobalSign Root CA, Roo...
有效期从	2014年2月20日 18:00:00

字段	值
颁发机构信息访问	[1]Authority Info Acces...
颁发机构密钥标识符	KeyID=60 7b 66 1a 45 0d...
密钥用法	Certificate Signing, Of...
基本约束	Subject Type=CA, Path L...
指纹算法	sha1
指纹	4c 27 43 17 17 56 5a 3a...

关于证书指纹



数字证书的签名哈希算法跟指纹算法都是指对摘要（指纹...

我来答



推荐于2017-09-25

证书签名使用的算法是发布者自己规定的 使用自己的私钥对证书编码的哈希值进行加密 一般算法为md5withrsa或者sha256withrsa。哈希算法是唯一的 就是把证书编码转换为固定长度的2进制 这个过程不可逆 就是说无法通过哈希值还原证书编码。指纹算法就是哈希算法 一般都是sh1。证书认证的流程是证书所有者把证书和指纹（证书的哈希值并用私钥加密）发给用户 用户根据证书计算出一个哈希值 用公钥解密指纹得到一个哈希值 看一下两者是否相同 相同及证明证书未被篡改。算法是由所有者的私钥加密的。ca的作用是ca是可以认证一个证书链，源头就是ca 一旦你信任了这个ca 就是信任了ca发布的证书，这样你与ca发布的证书的所有者通信时可以根据证书链找到ca ca可信任了则这个发布者就是可信任的

关于证书指纹

TLS/SSL数字证书里的指纹算法、签名算法和签名哈希算...

我来答

分享

举报

浏览 56

3个回答

#热议# 英雄联盟手游出了以后王者荣耀会凉



匿名用户

2019-06-28

你好!

签名哈希算法：签名之前对证书主体部分进行哈希的算法，它和签名算法结合，是签名及认证的一部分。

指纹算法：是对签名之后的证书文件计算一下散列值，只是用于检测证书是否被篡改，类似于去网站下载一个安装包，严谨一点的网站会给一个MD5的值，便于你下载之后再MD5一下核对。

证书指纹何时用到？举个例子，在导入某CA根证书或证书链时，会去该CA官方网站下载相关证书，下载好了之后可以手工检查一下指纹，确认无误再导入。

说的好有道理



撤销证书列表 (CRL)

版本号	
签名算法标识符	----- 算 法 参 数
颁发者名称	
本次更新时间 (日期/时间)	
下次更新时间 (日期/时间)	
撤销证书	----- 用户证书序列号 # 撤销时间 (日期/时间)
CRL 条目扩展项	
	.
	.
	.
撤销证书	----- 用户证书序列号 # 撤销时间 (日期/时间)
CRL 条目扩展项	
CRL 扩展项	
颁发者签名	----- 算 法 参 数 ----- 已加密的 Hash 值

证书有效性验证

现有许多开源的程序帮助你做复杂的证书校验过程



SSL/TLS 协议的实现程序说“该证书有效” == 该证书有效?

基于深度学习的SSL/TLS证书验证程序的自动化测试

Deep Learning-based Automated Testing of Certificate Verification in SSL/TLS Implementations

陈超

山东大学网络与信息安全研究所成员

ISC 互联网安全大会 | 中国·北京
2018 Internet Security Conference | China · Beijing





内容提纲

1

密钥管理

2

数字证书

3

PKI

4

证书透明性



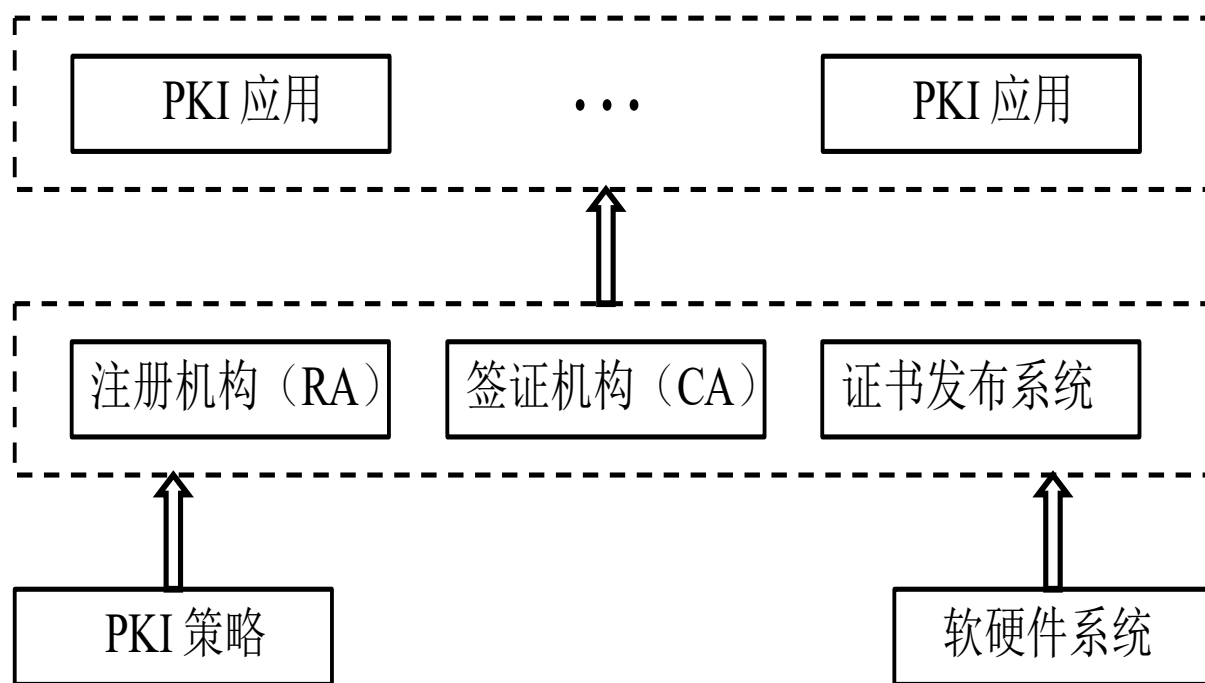


PKI

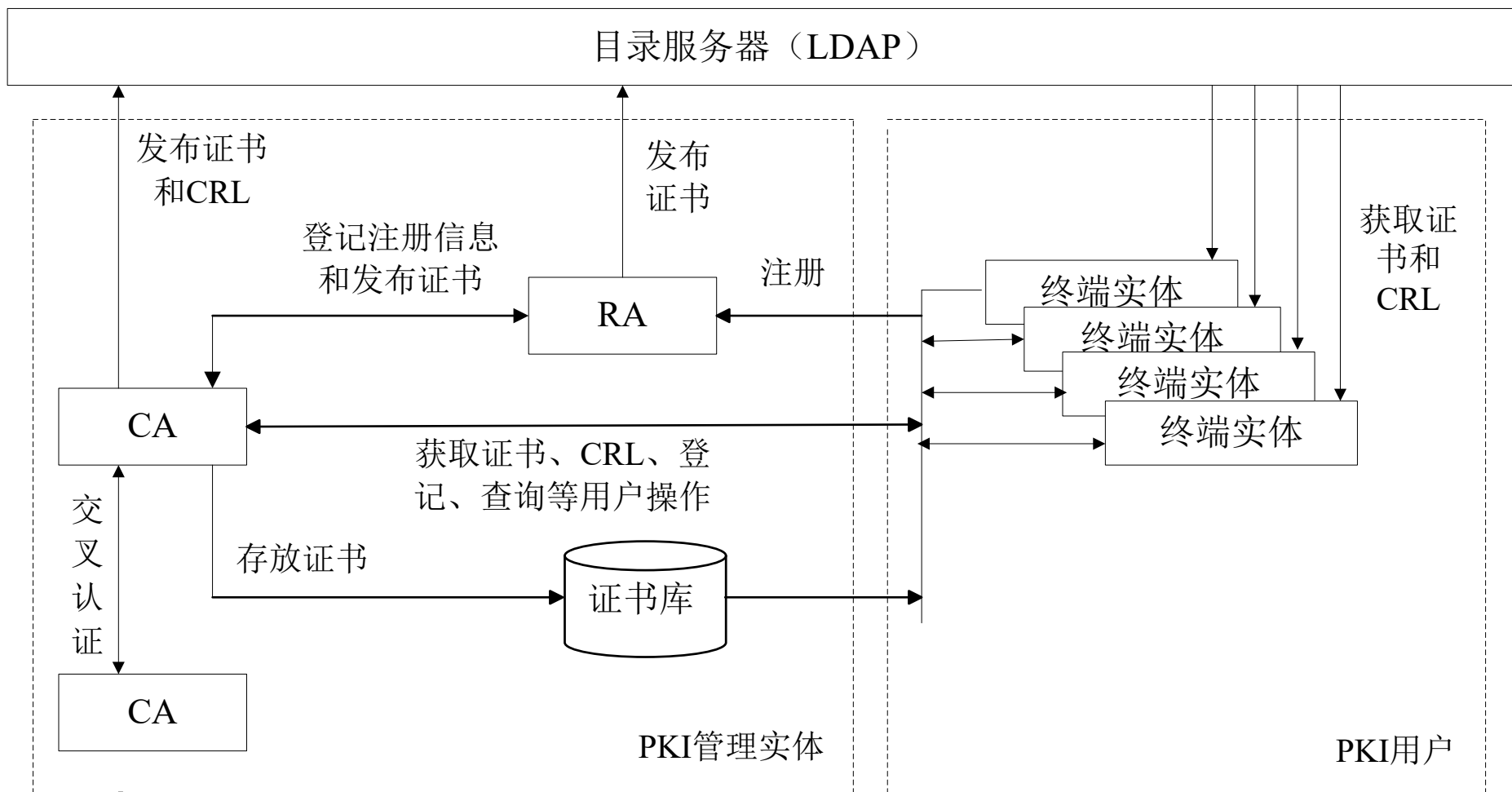
- 有了证书以后，将涉及证书的申请、发布、查询、撤销等一系列管理任务，因此需要一套完整的软硬件系统、协议、管理机制来完成这些任务，由此产生了公钥基础设施（PKI）



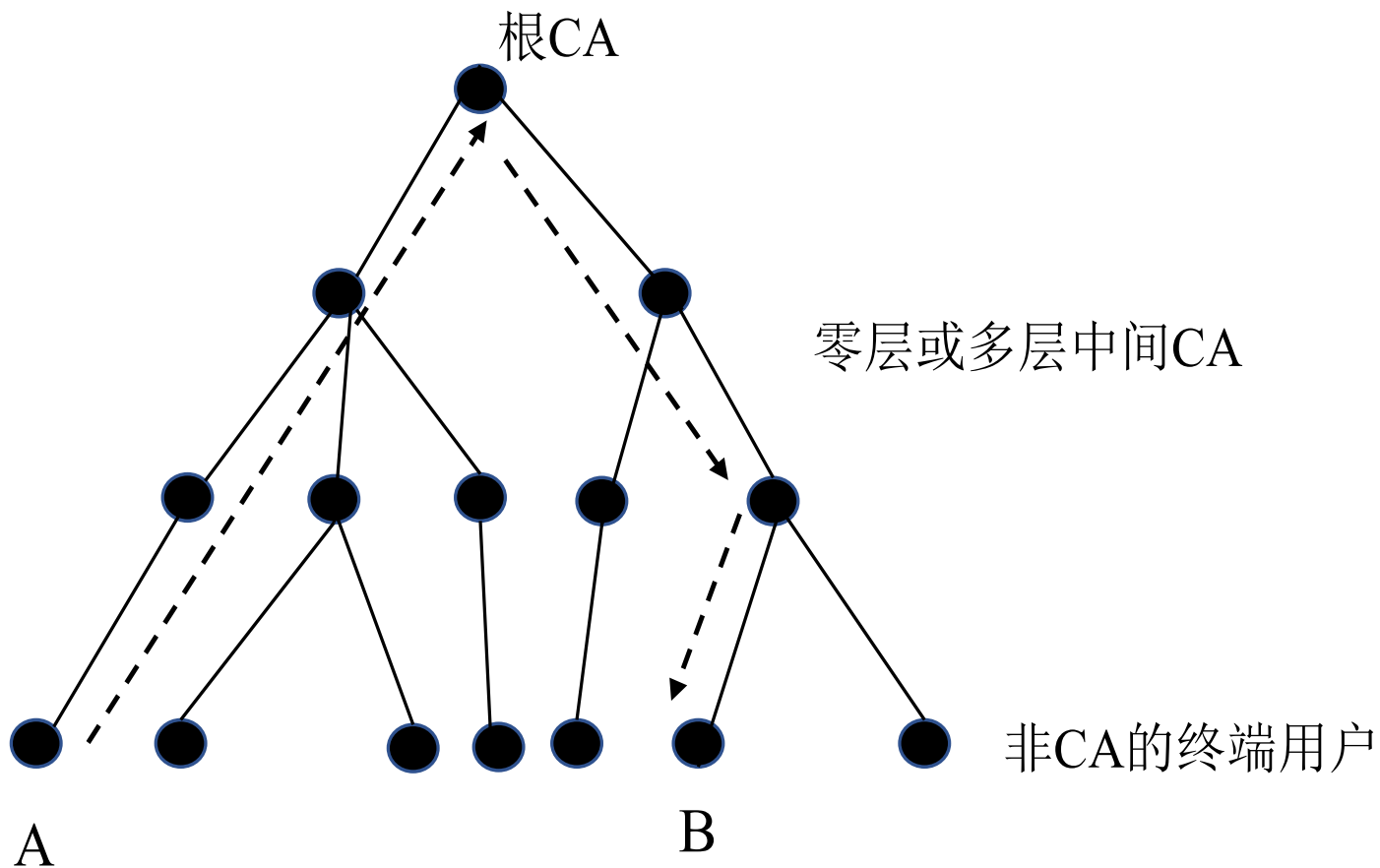
PKI系统组成



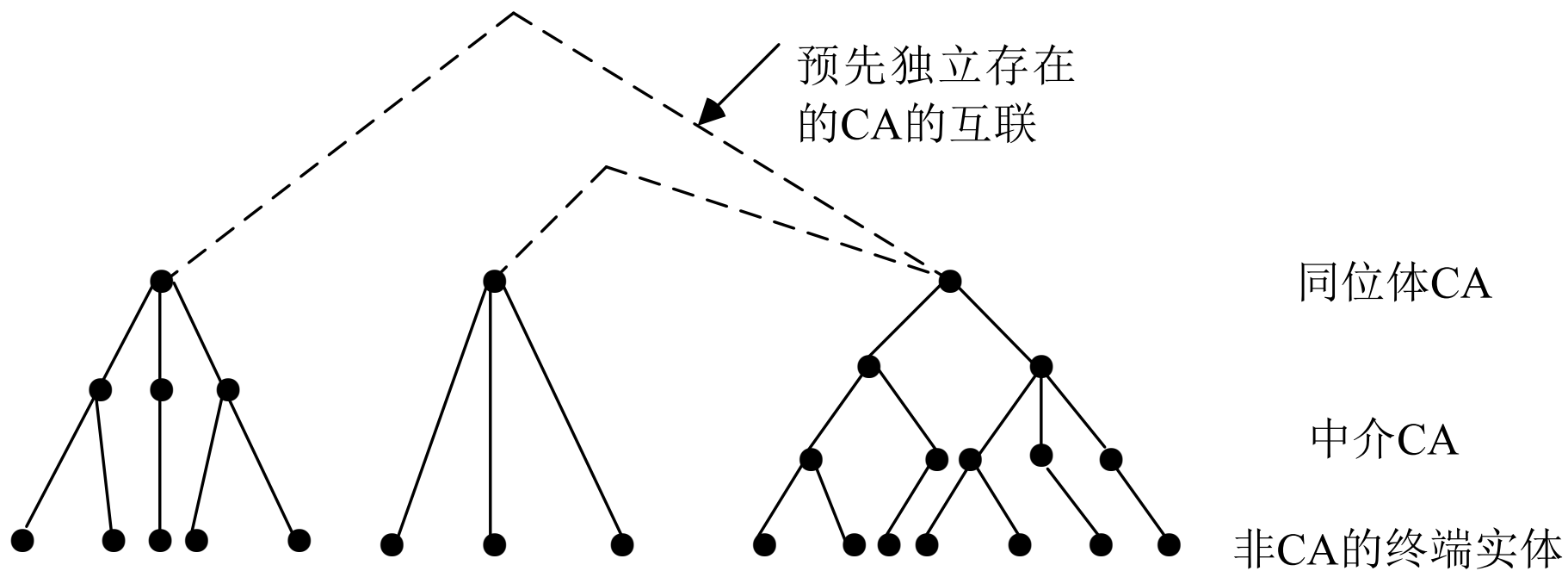
PKI认证体系



CA树型信任模型



CA交叉信任模型





CA交叉信任模型

CA1目录实体

交叉证书对

正向交叉证书 主体=CA1 颁发者=CA2
反向交叉证书 主体=CA2 颁发者=CA1

CA2目录实体

交叉证书对

正向交叉证书 主体=CA2 颁发者=CA1
反向交叉证书 主体=CA1 颁发者=CA2

假设Alice有CA1公钥，Bob有CA2公钥，交叉认证后，Alice的信任能扩展到CA2的主体群（包括Bob），反之亦然。



交叉认证

- Let's Encrypt 提供免费加密证书服务，在获得 IdenTrust 交叉签名认证后为主流浏览器支持。
- Let's Encrypt 证书已经被黑客用的极度泛滥了

[Documentation](#)[Get Help](#)[Donate ▾](#)[About Us ▾](#)[Languages !\[\]\(003082e50e3009141f59bd5df831749f_img.jpg\) ▾](#)

A nonprofit Certificate Authority providing TLS certificates to **200 million** websites.

Read our 2019 Annual Report ([Desktop](#), [Mobile](#))

[Get Started](#)[Sponsor](#)

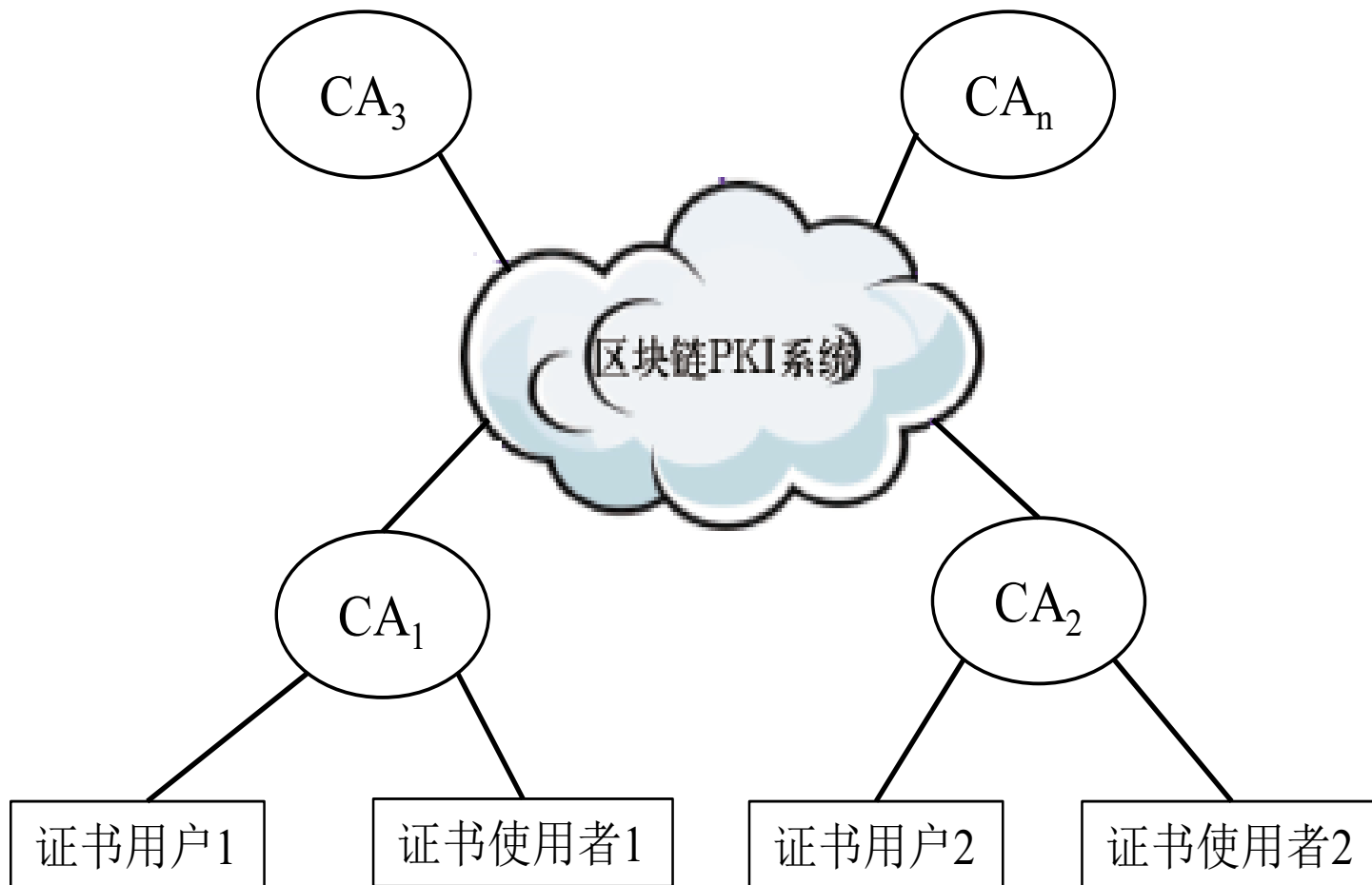


CA分布式信任模型

- 双向交叉认证有什么问题？
 - 只适用于CA 数量较少的情况，但当CA数量较大时，大量CA 两两进行交叉认证就会形成复杂的网状结构，且证书策略经过多次映射之后会使证书用途大大受限

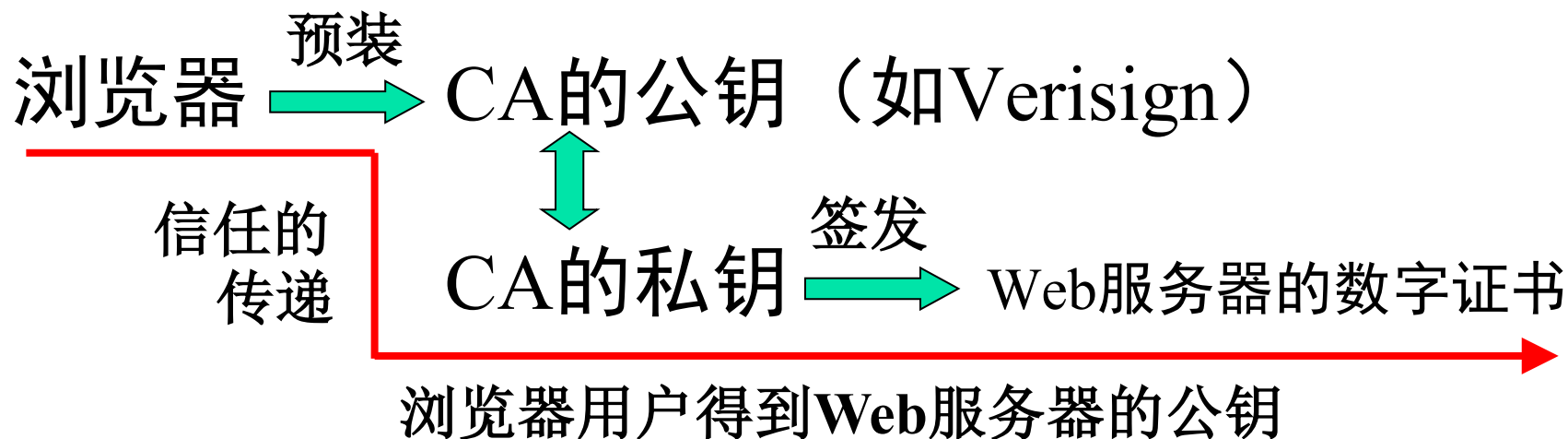


CA分布式信任模型




Web浏览器中的数字证书

★ Web模型依赖于流行的浏览器



浏览器厂商起到信任锚的作用，预装公钥的CA就是它所认证的CA——有隐含根严格层次结构。

用自签名证书伪造知名网站证书

- 2020.3.26 国内部分地区网络出现中间人攻击（通过骨干网络进行劫持 HTTPS的443端口）：GitHub、京东等被劫持。因证书不对，被浏览器阻止访问 

您的连接不是私密连接

攻击者可能会试图从 **z.github.io** 窃取您的信息（例如：密码、通讯内容或信用卡信息）。
[了解详情](#)

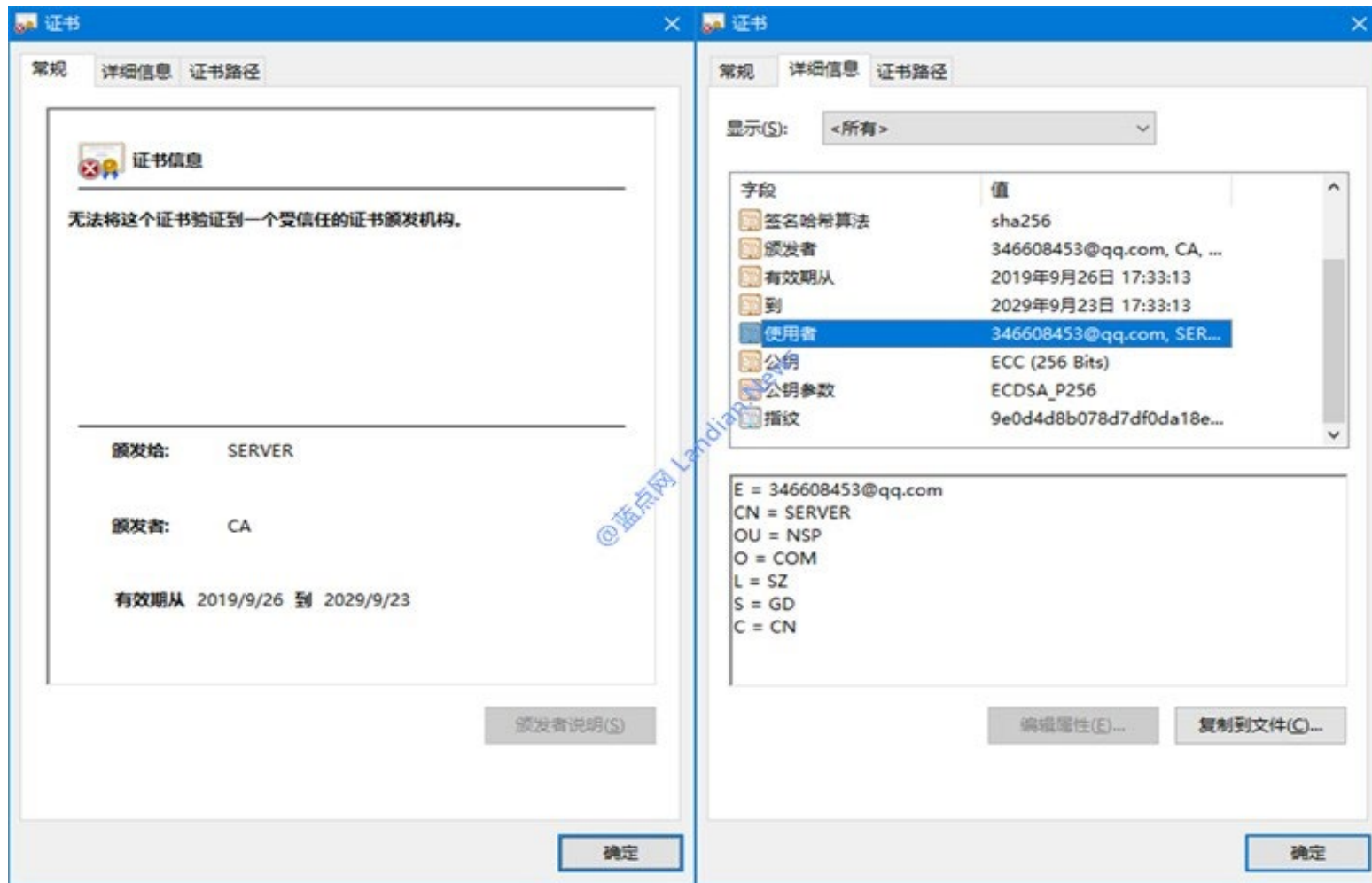
NET::ERR_CERT_AUTHORITY_INVALID

☐ 将您访问的部分网页的网址、有限的系统信息以及部分网页内容发送给 Google，以帮助我们提升 Chrome 的安全性。[隐私权政策](#)

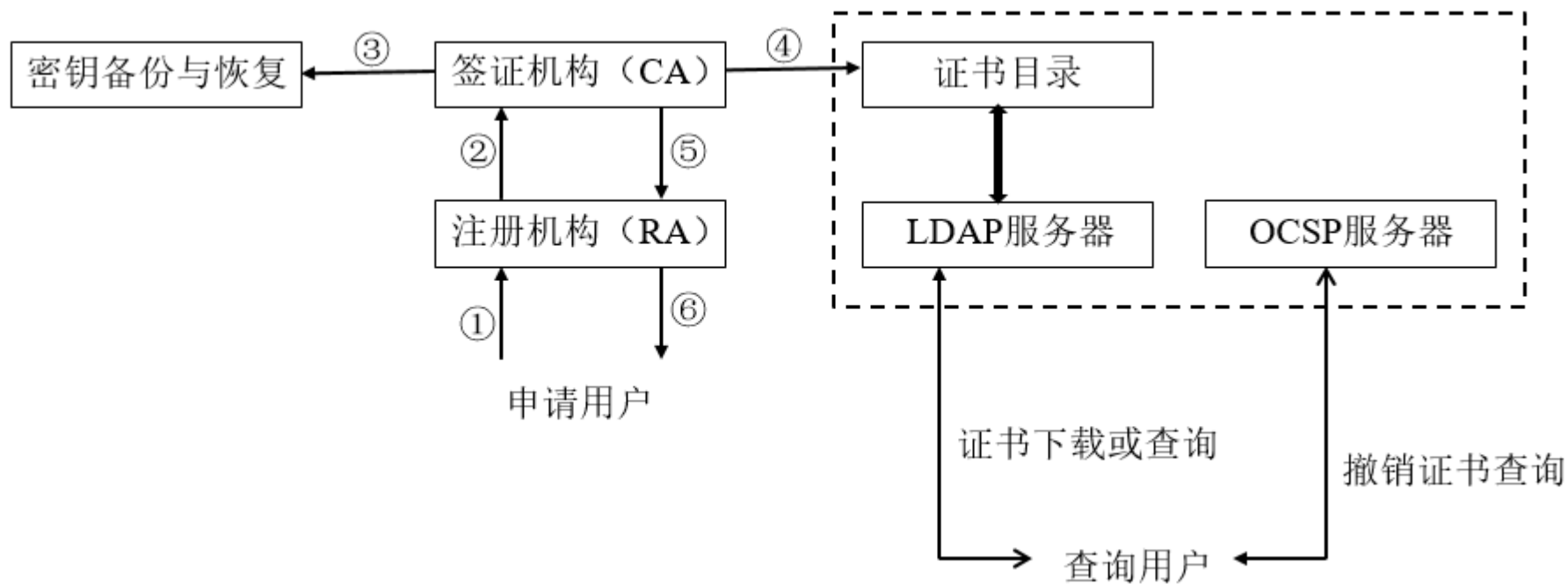
隐藏详情

返回安全连接

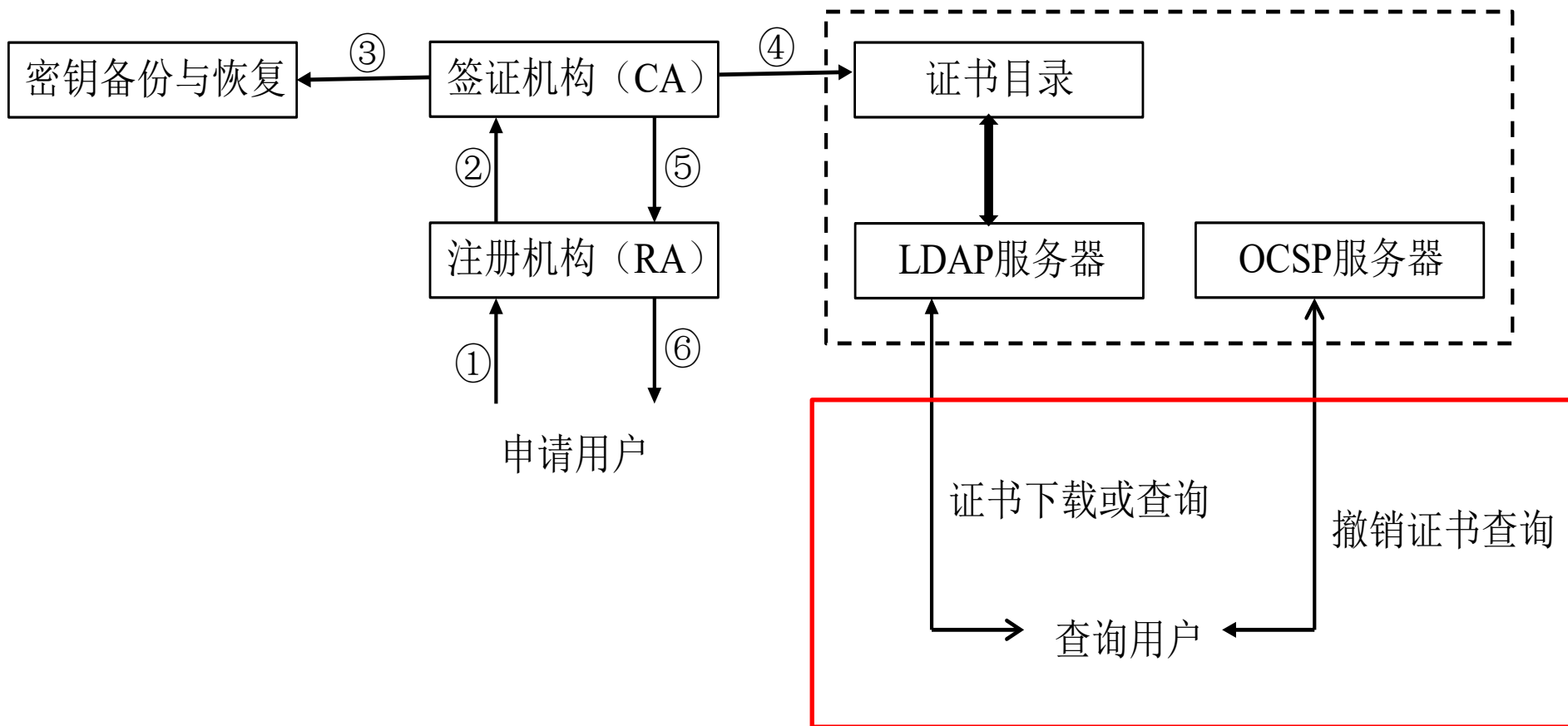
用自签名证书伪造知名网站证书



证书签发和撤销流程



证书使用查询





PKIX

- IETF成立了PKI工作组，制定了PKIX系列标准（Public Key Infrastructure on X.509, PKIX）。

PKIX定义了X.509证书在Internet上的使用规范，包括证书的产生、发布、获取、撤销，各种产生和发布密钥的机制，以及怎样实现这些标准的框架结构等



■ 标准

表 4-1 PKIX 基础标准列表

标准编号	标准内容
RFC 5280	定义了 X.509 v3 公钥证书和 X.509 v2 CRL 格式、结构。本标准替代了早期的 RFC 2459, 3280, 4325, 4630
RFC 2528	基于 X.509 的密钥交换算法 KEA (Key Exchange Algorithm)
RFC 3039	描述用于防抵赖的高可信证书的格式和相关内容
RFC 3279	描述了 X.509 v3 公钥证书和 X.509 v2 CRL 中使用基于 ASN.1 的算法标志和算法的编码格式



■ 标准

表 4-2 PKIX 证书操作标准列表

标准编号	标准内容
RFC 2559	使用 LDAP v2 作为 PKI 实体发布和获取证书及 CRL 的协议。该标准后被 RFC 3494 替代，以在开放环境下提供足够强度的完整性和机密性的支持（使用 LDAP v3 标准）
RFC 2560	在线证书状态查询协议(OCSP)，从而可以通过在线证书状态服务器，而不是使用 CRL 获得证书的当前状态
RFC 2585	通过 FTP 和 HTTP 从 PKI 系统中获取证书和 CRL 的操作协议
RFC 2587	使用 LDAP v2 获取公钥证书和 CRL 的一个最小模型

■ 标准

表 4-3 PKIX 证书管理协议标准列表

标准编号	标准内容
RFC 2510	X.509 PKI 用于实体间传递消息的证书管理协议 CMP (Certificate Manager Protocol), 来提供完整的 PKI 管理服务
RFC 2511	证书请求报文格式 CRMF (Certificate Request Message Format)
RFC 2527	证书策略和 CPS 相关信息的政策大纲
RFC 2797	描述了 X.509 PKI 客户端和服务端之间采用 CMS (Cryptography Message Syntax)加密消息语法作为消息封装的方法





内容提纲

1

密钥管理

2

数字证书

3

PKI

4

证书透明性





CA有问题怎么办？

- PKI体系中，用户无条件信任由可信第三方（CA）签发的证书。但是，如果CA服务器被攻击或CA在签发证书时没有对申请者进行严格的尽职调查，就会产生严重的安全问题。
 - 几起典型CA问题事件



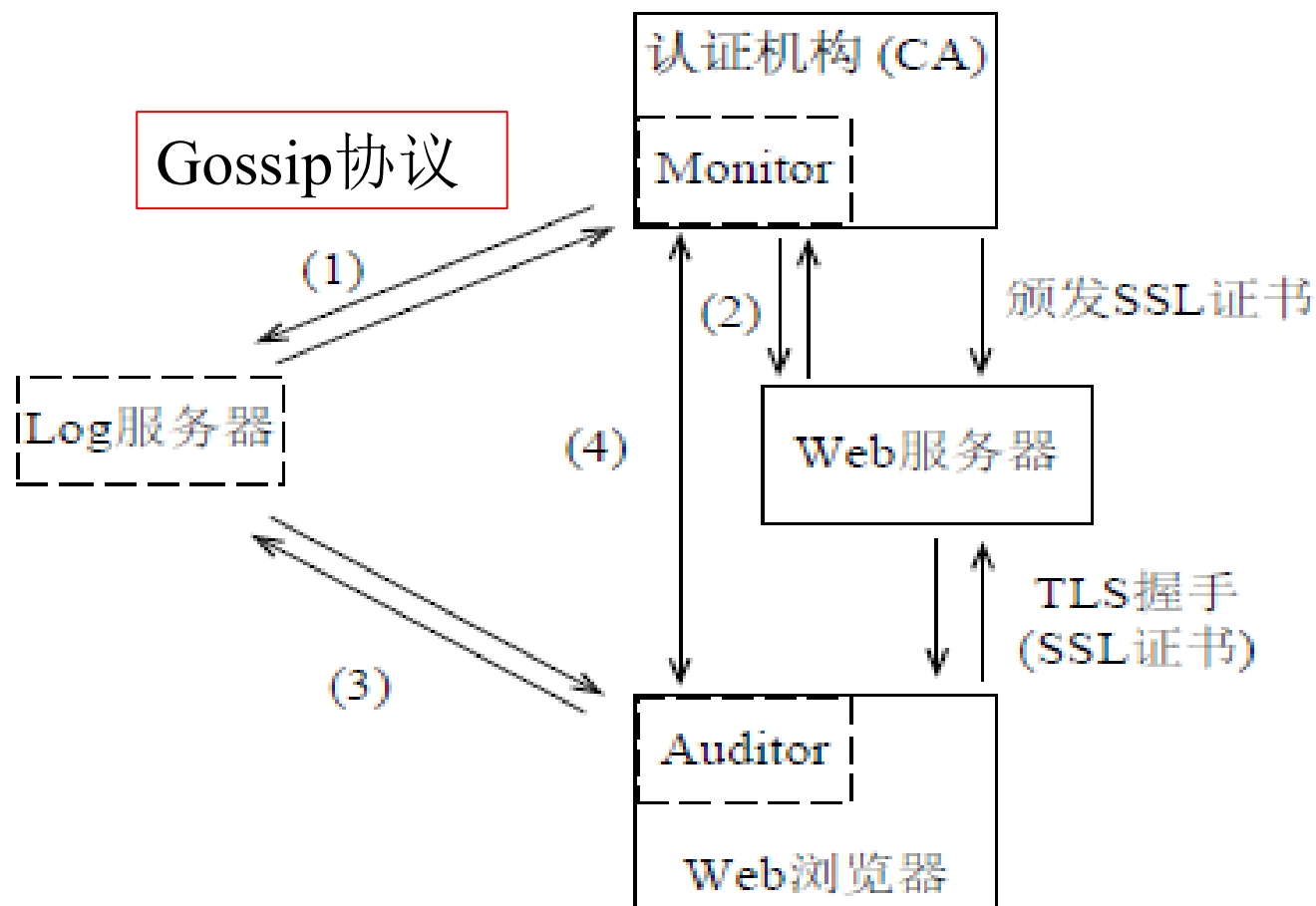


CA有问题怎么办？

- 为了解决盲目信任CA 签发的证书所存在的潜在风险，Google于2013年3月提出了**数字证书透明性**（Certificate Transparency, CT)技术，用于提升服务器证书的可信性，从而提高使用证书的系统的安全性。
- 同年6月，IETF发布CT试验性标准：RFC6962 (Certificate Transparency)。2014年1月，IETF 成立Public Notary Transparency (TRANS) 工作组，专门讨论设计、部署、使用CT时碰到的各种问题



证书透明性





证书透明性

- CT能够对证书进行公开审计，确保网站访问者不受恶意或者错误的证书所害。
- 安全风险：CT也引入了新的运行风险，如Log服务器为虚假证书创建了一条日志，Monitor不通知域名所有者存在针对其域名的虚假证书等



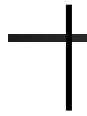


本章小结





讨论：Web信任模型的安全问题

- 当前，CA的信任体系没有唯一的信任根（信任锚点，Trust Anchor），预装到浏览器或操作系统中的可信根CA证书有一百多个，他们又通过成千上万个二级或三级CA签发最终的服务器证书。
 - 这种信任模型最大的安全问题是，**任何一个CA都可以为任何一个网站（域名）签发公钥证书，而不需要该网站的授权**
- 



讨论：Web信任模型的安全问题

- 如果CA出了问题，如私钥泄露了该怎么办？
 - 2011年8月，荷兰CA安全证书提供商DigiNotar的服务器被发现遭黑客入侵。黑客为包括Google.com在内的20多个领域的531个网站发行了伪造的CA证书，经分析DigiNotar的8台证书服务器均遭入侵。2011年7月19被发现入侵，但外界直到8月份才知道。
 - 出了这种事，后果是什么？用户如何应对？



讨论：Web信任模型的安全问题

- 如果根CA不可信，会有什么后果？
- 如果互联网接入服务提供商（ISP）能作为CA签发证书，它能做什么？



讨论：证书共享的安全问题

- 证书共享：随着技术的发展，多个实体（如网站）共享一个证书的情况已经比

比

■ C

■ 集

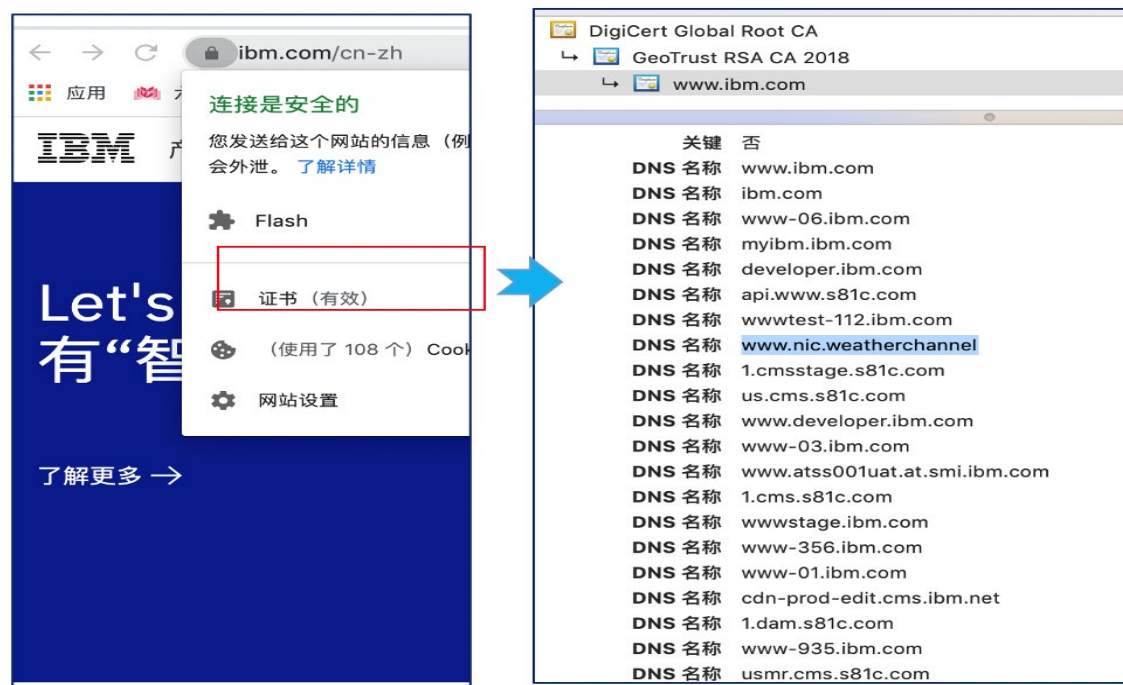


图6. 共享证书实例：ibm.com与45个域名共享证书



讨论：证书共享的安全问题

- 类似IBM 和nic.weatherchannel的情况非常普遍，由于Web网站的HTTPS部署比较复杂而且技术仍在进一步发展，共享证书的多个网站之间难免会出现配置故障或者策略不一致的现象。



讨论：证书共享的安全问题

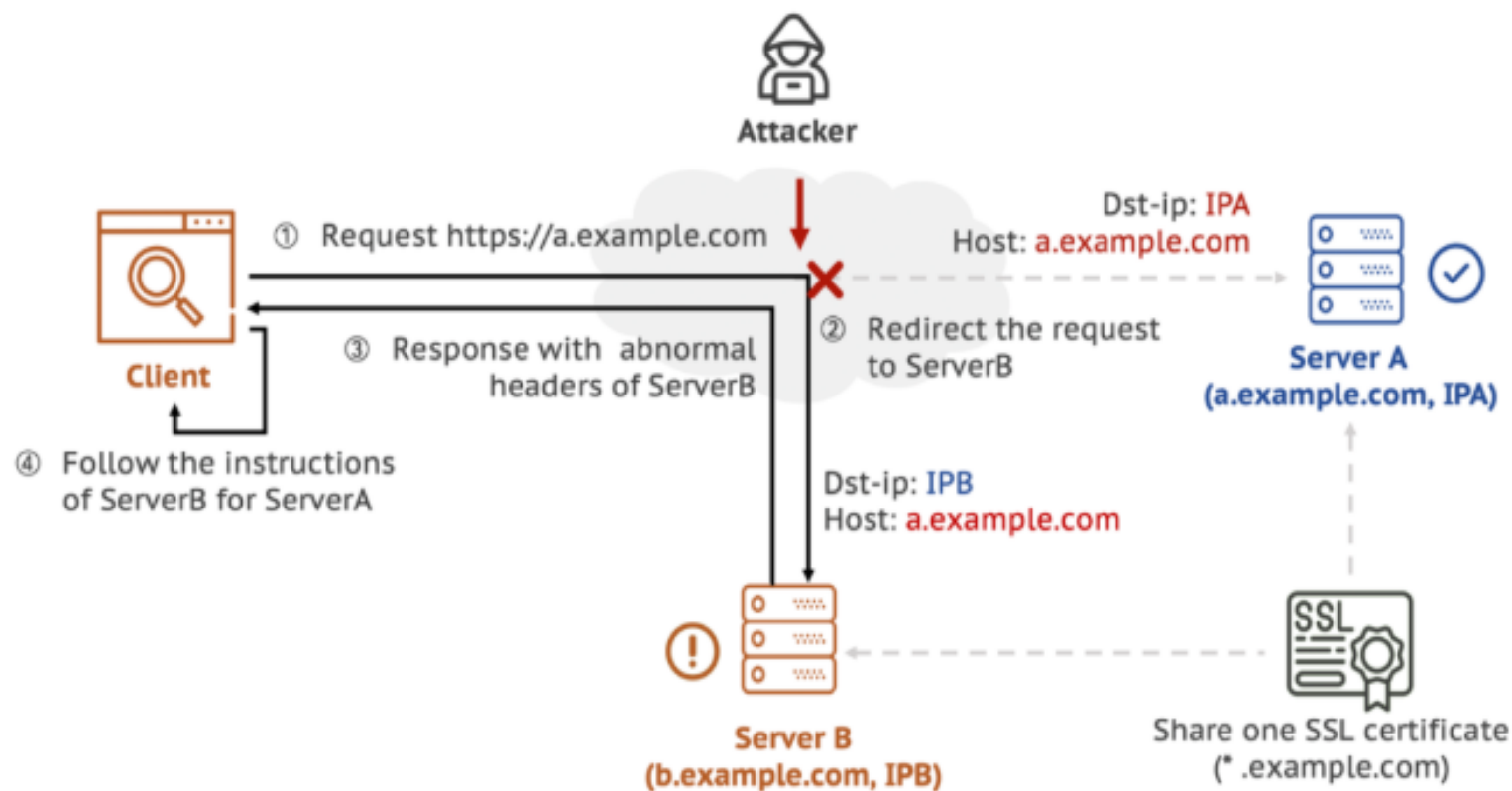


图7. 熟悉的陌生人攻击——证书共享导致的HTTPS中间人劫持



作业

