

Cryptography

An introduction to OpenSSL, PGP and Let's Encrypt

A Brief intro to cryptography

- Need: To protect applications communication
- Goals: data confidentiality, data integrity, authentication, and non-repudiation
数据保密性、数据完整性、身份验证和不可否认性
- Network-based attacks: eavesdropping, IP spoofing, connection hijacking, and tampering
基于网络的攻击：窃听、IP 欺骗、连接劫持和篡改

Not easy to use cryptographic algorithms in a secure and reliable manner !

Why cryptography is important?

login page

testphp.vulnweb.com/login.php

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB

acunetix acuart

TEST and Demonstration site for **Acunetix Web Vulnerability Scanner**

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo

search art go

Browse categories

Browse artists

Your cart

Signup

Your profile

Our guestbook

AJAX Demo

Links

Security art

PHP scanner

PHP vuln help

Fractal Explorer

If you are already registered please enter your login information below:

Username :

Password :

You can also [signup here](#).

Signup disabled. Please use the username **test** and the password **test**.

About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd

Why we need SSL/TLS?

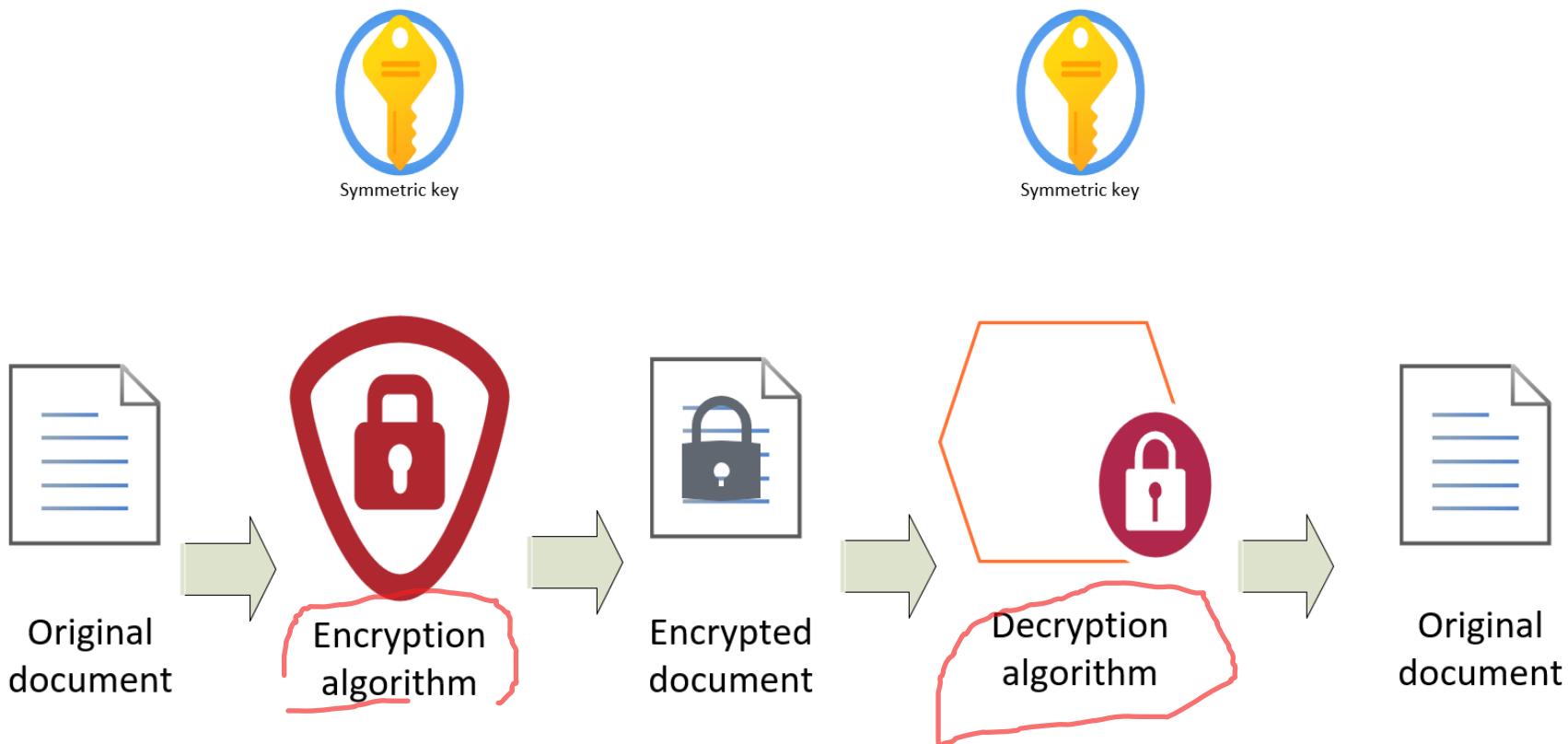
- Cryptographic protocols are difficult to implement
- Not easy to use cryptographic algorithms in a secure and reliable manner
- The algorithms are just the building blocks in the protocols
- Cryptographic protocols need to cover and resist all known attacks
- Attackers can perform tampering to data
- Many cryptographic protocols have limited applicability
- SSL makes the security of network connection easier

SSL/TLS →

provide nowadays the most common security services for TCP-based connection
Adds transparent confidentiality authentication and integrity to TCP connections

Types of cryptographic algorithms: symmetric key encryption

对称密钥加密



Advantages & Disadvantages of symmetric keys

- **Efficient & faster:** suitable for large amount of data (streaming)
- **Simpler:** it includes less computational steps and effort
- **More suitable for embedded systems and IoT industrial devices:** in some case where we have resource-constrained environments
- **Single Point of Failure:** Encrypts and decrypts only with a single key and must keep safe!
- **Limited Authentication:** Only the one with the key can decrypt the message
- **Key Management:** Revoke, rotation hard in large environments with many users

撤销，
旋转困难大
环境中有很多
用户

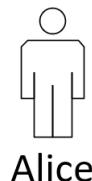
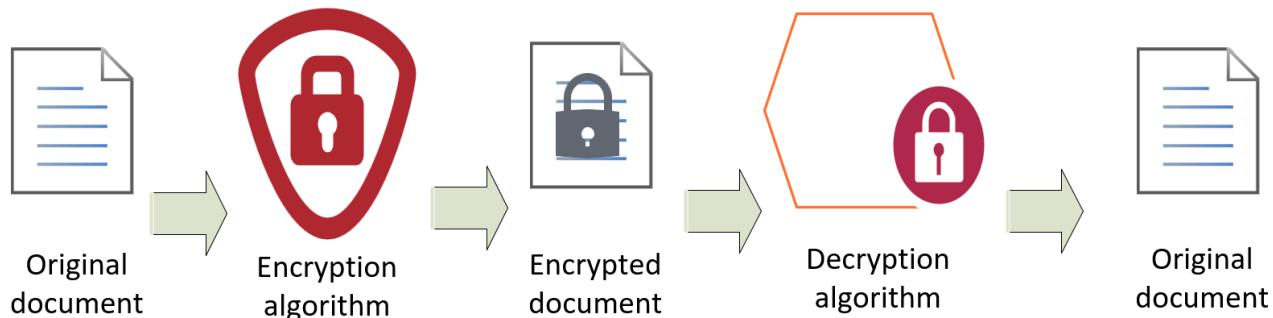
Types of cryptographic algorithms: asymmetric key encryption (Public key encryption)



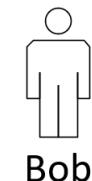
Bob's public key



Bob's private key



Alice



Bob

Advantages & Disadvantages of Public keys

- **Distribution:** use of public key only (relies on trust in the authenticity of public keys)
- **Non-Repudiation**, with certificates verify the authenticity and integrity of messages
不可否认性
- **Authentication:** third parties can validate certificates sent with public keys
- **Slow:** For large messages typically slower and more computationally intensive
- **Not for large data:** used for key exchange protocols
- **Key size:** Produced keys significantly larger than symmetric keys (increased bandwidth and storage requirements)

用于
密钥交换协议

认证：第三
当事人可以验证
与公众一起发送的证书
键

Types of cryptographic algorithms: Cryptographic hash functions

- These are checksum algorithms (i.e. MD5 128bits SHA1 160bits → safer)
- Hash functions converts data into a fixed-size checksum (message digest)
- Any change to the data gives different output (tampering) 篡改
- The output reveals no info about the data
- Impossible to find two inputs to produce same checksum
- Practically impossible to algorithmically reconstruct the input (one-way)
- Output twice as large as the symmetric key algorithm

Hash functions demo

- Password storage solution
- Protect software release

```
import hashlib

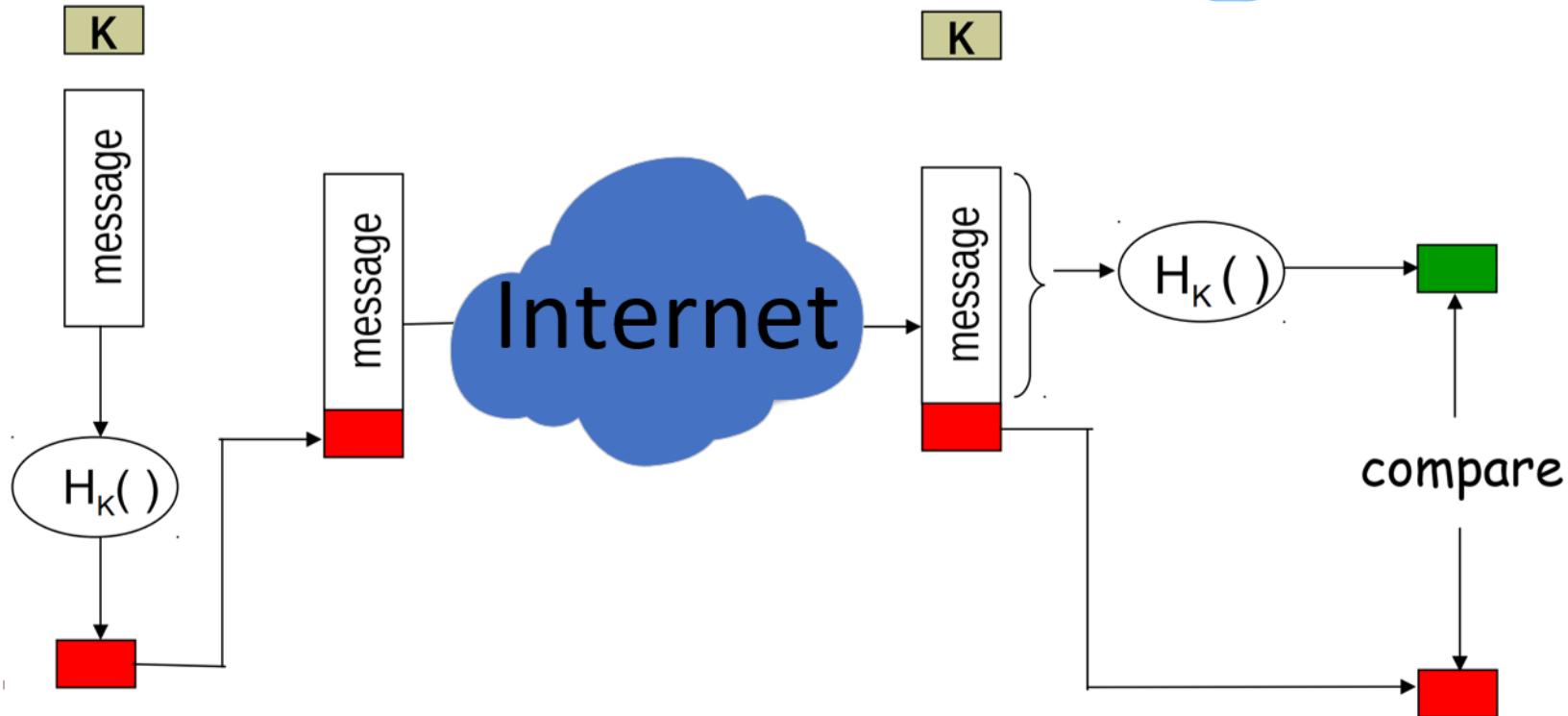
message = input("Enter the message to hash with md5: ")
md5 = hashlib.md5(message.encode())
print ("Hash message with SHA1 128 bits: "+ str(md5))

message = input("Enter the message to hash with sha1: ").encode('utf-8')
sha = hashlib.sha1(message)
sha1 = sha.hexdigest()
print ("Hash message with SHA1 160 bits: "+ sha1)

message = input("Enter the message to hash with sha256: ").encode('utf-8')
sha = hashlib.sha256(message)
sha256 = sha.hexdigest()
print ("Hash message with SHA1 256 bits: "+ sha256)

message = input("Enter the message to hash with sha512: ").encode('utf-8')
sha = hashlib.sha512(message)
sha512 = sha.hexdigest()
print ("Hash message with SHA1 512 bits: "+ sha512)
```

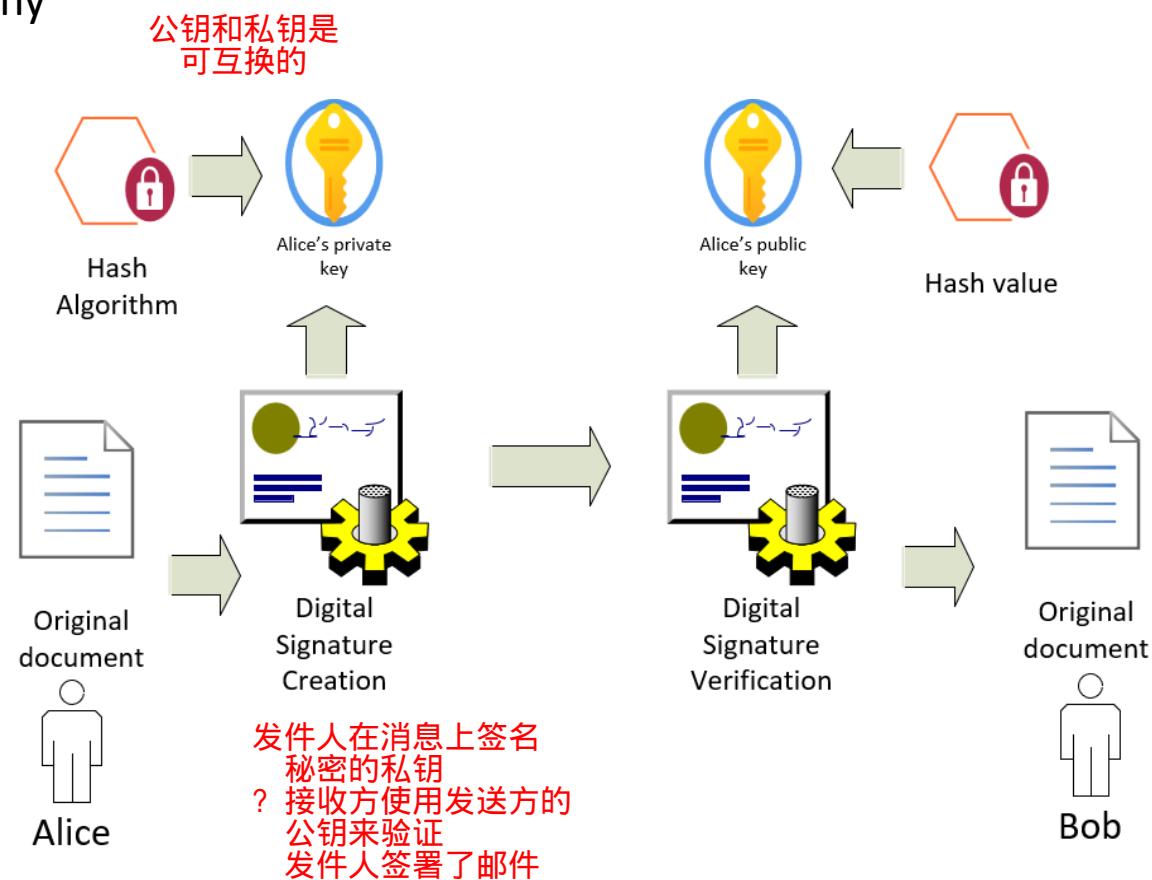
Types of cryptographic algorithms: Message Authentication Codes (MACs)



K = secret key, MAC is supported in SSL and in OpenSSL as only HMAC
Ensure integrity for the "message digest"

Types of cryptographic algorithms: Digital signatures

- A form of public key cryptography
- Used to provide digital identity authentication and encryption
- Public key and private key are interchangeable
- Digital signatures are very slow
- Use 1,024 bits or higher to ensure security
- Sender signs a message with the secret private key
- Receiver use the sender's public key to verify that the sender signed the message



Encryption Algorithms advantages & disadvantages

- Data Encryption Standard (DES)
It was one of the first widely used but not longer considered secure because it holds small key size
- Triple DES (3DES)
Effective as it applies the DES algorithm three times (168-bit key), but it consumes much more time than other
- Advanced Encryption Standard (AES)
Strong security, efficient in terms of computational resources and memory usage
Flexible as it supports many key lengths but vulnerable to side-channel attacks
但容易受到旁路攻击
- RSA (Rivest-Shamir-Adleman)
Widely used to secure key exchange, digital signatures, and public key encryption
It has built-in mechanisms for non-repudiation through digital signatures
To resist attacks large RSA keys, consumes more time for encryption
它具有通过数字签名实现不可否认的内置机制
为了抵御大型RSA密钥的攻击，加密需要花费更多的时间
- Elliptic Curve Cryptography (ECC)
A strong security algorithm with small key sizes.
Efficient in terms of bandwidth and computational resources (IoT devices)
It has Implementation complexity, and need careful implementation
椭圆曲线密码学 (ECC)

How to select the key lengths

- Consider the Encryption Algorithm:

length of keys in public key are large numbers comparable to symmetric algorithms

512-bit keys too weak, 2,048 bits may be too slow

- Security Requirements:

Sensitivity of the data being encrypted and potential threats

- Lifespan of the Data:

Consider for how long you need your data to remain secure (longer key lengths?)

- Regulation:

Ensure to comply with regulatory requirements with minimum key lengths (ISO 27001)

- Maintain balance:

Longer keys provide more security but may increase computational overhead and give slower performance

Overview of SSL/TLS

- SSL is a widely deployed security protocol (HTTPS)
- Secures any protocols over TCP
- Client sends a handshake to the server and the server in the response sends the certificate

Application layer

Presentation layer

Session layer

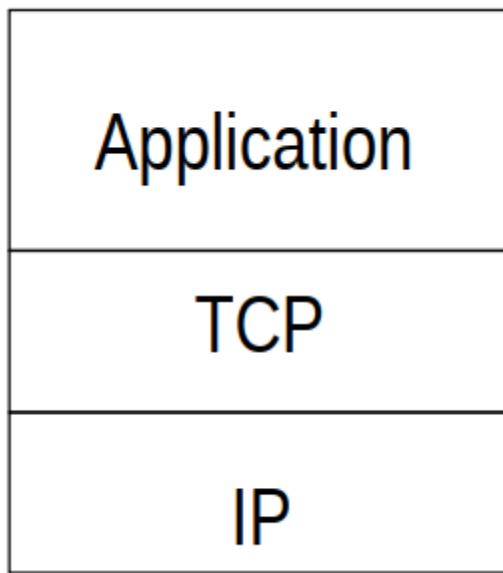
Transport layer

Network layer

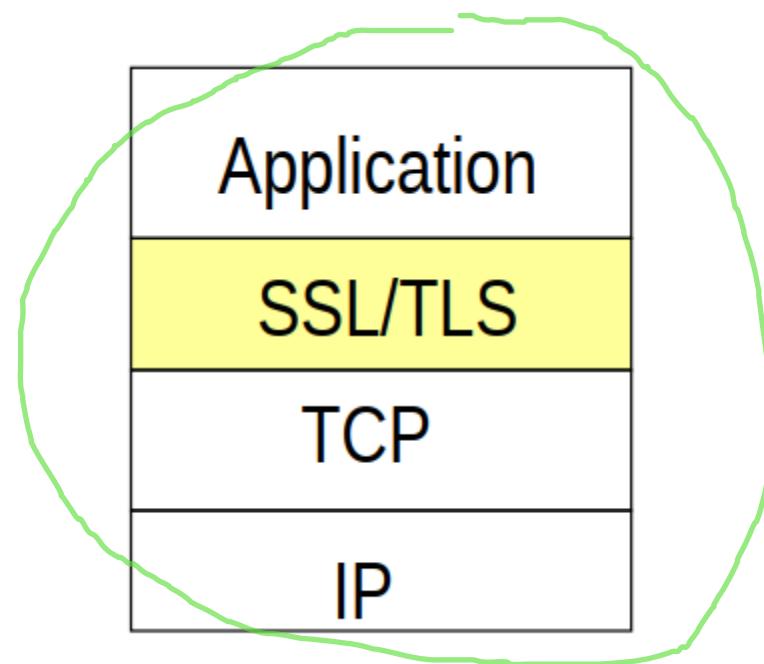
Data link layer

Physical layer

SSL/TLS and TCP/IP

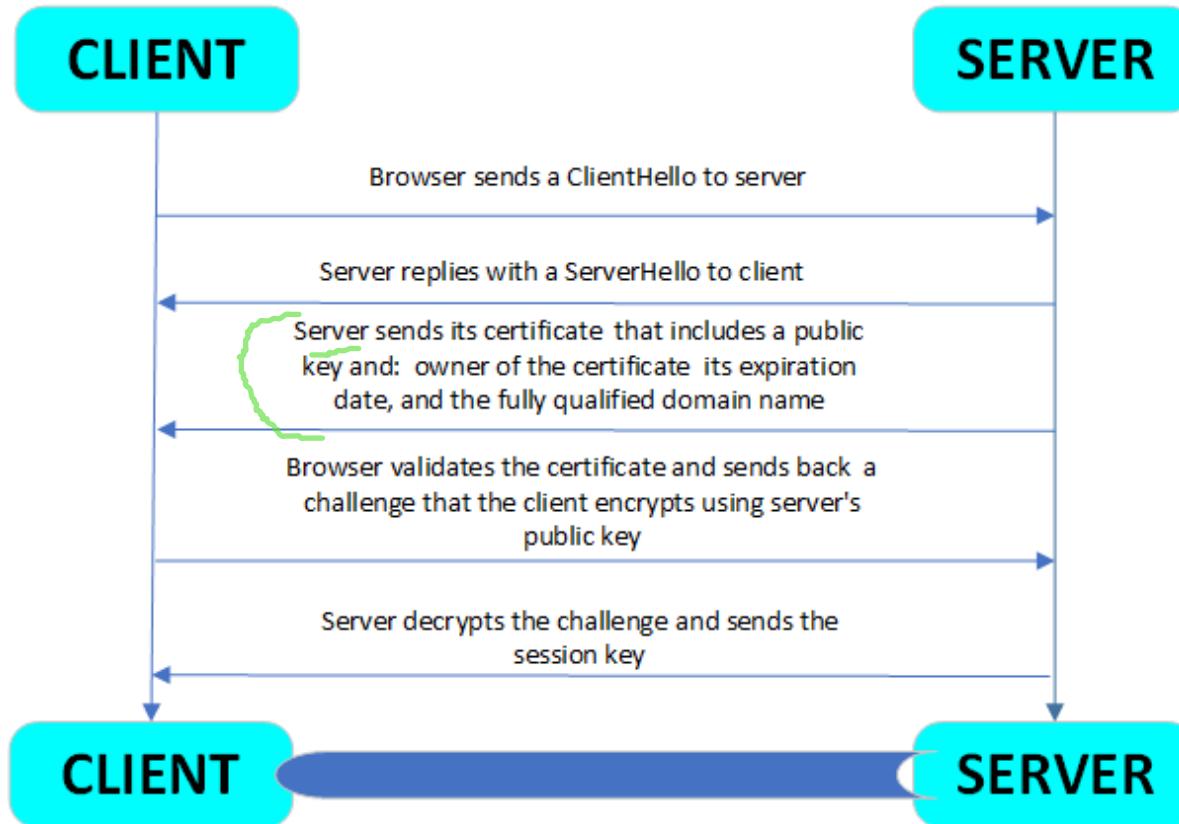


Application



Application with SSL/TLS

SSL/TLS Handshake

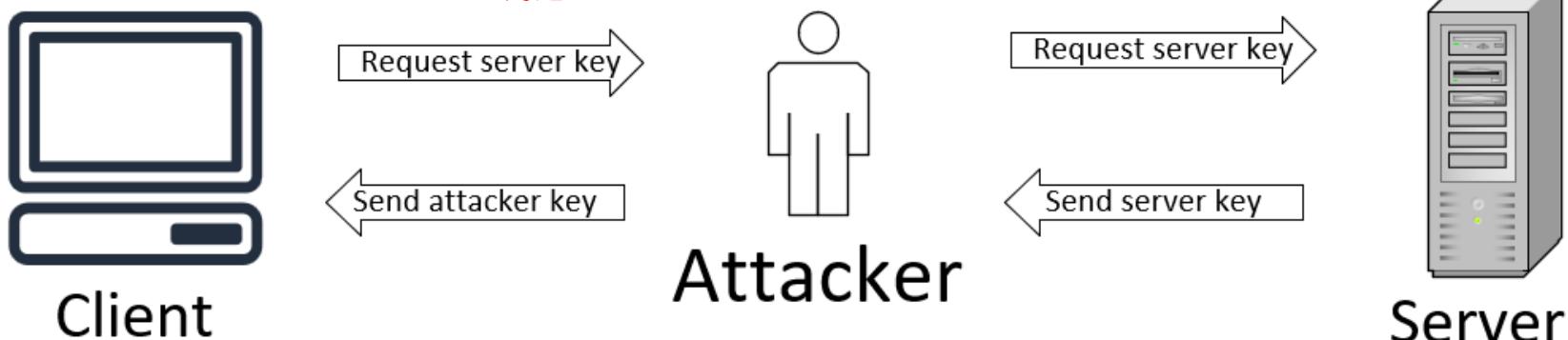


SECURED CONNECTION ESTABLISHED

MitM attack for SSL/TLS

- The attacker needs a copy of the certificate and a private key to masquerade as a known server
- Attacker can sniff the server messages and present the attacker's certificate
- The forged certificate can look like legitimate
- Man-in-the-middle (MitM) attack where the attacker eavesdrops on all communication

攻击者可以嗅探服务器消息并出示攻击者的证书
? 伪造的证书可能看起来像合法的
? 中间人 (MitM) 攻击，攻击者窃听所有信息沟通



What SSL/TLS doesn't do well?

- Using SSL is slower than an HTTP connection (handshake with public key)
- Overhead of encrypting and decrypting the data
- Doesn't work with transport layer protocols not connection-oriented, such as UDP
- SSL has no support for non-repudiation (what if the other party attach a message with invalid signature?)
- SSL doesn't protect against flaws in the application itself i.e. buffer overflow
- SSL cannot protect data before it is sent but only data in transit

Using SSL and the OpenSSL library

- OpenSSL is a cryptographic library able to implement many encryption algorithms, such as DES, AES and RSA
- OpenSSL used to be SSLeay created by Eric A. Young and Tim J. Hudson
- beginning in 1995
- OpenSSL first version was released as 0.9.1c in 1998
- OpenSSL is a cryptographic library and an SSL toolkit
- The SSL library provides all versions of SSL alongside with TLS
- Supports the most popular algorithms for symmetric and public key and hash algorithms
- OpenSSL a free SSL implementation and works on Unix Oss and Windows
- It has a feature of pseudorandom number generator (increase entropy)

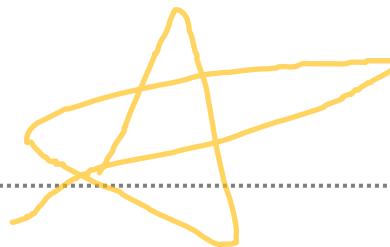


```
int getRandomNumber()
{
    return 4; // chosen by fair dice roll.
              // guaranteed to be random.
}
```

OpenSSL Files

- .KEY
A file that has the private key
- .CSR (Certificate Signing Request)
A file that is sent to the Certificate Authority with information and needs the private key
- .CRT (Certificate abbrev)
It is the security certificate file, created to establish secure connections
- .PEM (Privacy Enhanced Mail)
May include the public certificate or an entire certificate chain (public key, private key, certificate)
- .CRL (Certificate revocation list)
A file used to de-authorize certificates before expiration

这部分看exercise更好



Generating Public and Private Keys

Generating RSA Keys

Generate 2048 bit RSA Private Key saved as KEY1.pem

```
openssl genrsa -out KEY1.pem 2048
```

Generate 4096 bit RSA Private Key, encrypted with AES128

```
openssl genrsa -out KEY2.pem -aes128 4096
```

- Key size must be last argument of command
- Omit -out <FILE> argument to output to StdOut
- Other encryption algorithms are also supported:
-aes128, -aes192, -aes256, -des3, -des

Generating DSA Keys:

Generate DSA Parameters File

```
openssl dsaparam -out DSA-PARAM.pem 1024
```

Generate DSA Keys file with Parameters file

```
openssl gendsa -out DSA-KEY.pem DSA-PARAM.pem
```

Generate DSA Parameters and Keys in one File

```
openssl dsaparam -genkey -out DSA-PARAM-KEY.pem 2048
```

See *Inspecting* section to view file contents.

Generating Certificate Signing Requests (CSRs) and Self-Signed Certificates

Generating CSRs:

Generate CSR with existing Private Key file

```
openssl req -new -key KEY.pem -out CSR.pem
```

Generate CSR and new Private Key file

```
openssl req -new -newkey <alg:opt> -nodes -out CSR.pem
```

Generating Self-Signed Certificates

Generate Certificate with existing Private Key file

```
openssl req -x509 -key KEY.pem -out CERT.pem
```

Generate Certificate and new Private Key file

```
openssl req -x509 -newkey <alg:opt> -nodes -out CERT.pem
```

Inspecting Certificate Signing Requests (CSRs) and Certificates

Viewing contents of Certs and CSRs

Viewing x509 Certificate as human readable Text

```
openssl x509 -in CERT.pem -noout -text
```

Viewing Certificate Signing Request (CSR) contents as Text:

```
openssl req -in CSR.pem -noout -text
```

Extracting Specific Info from Certificates

Extract specific pieces of information from x509 Certificates

```
openssl x509 -in CERT.pem -noout -dates
```

```
openssl x509 -in CERT.pem -noout -issuer -subject
```

Other items you can extract:
-modulus -pubkey -ocsp_uri -ocspid
-serial -startdate -enddate

Some Known attack against SSL/ TLS (CVEs?)

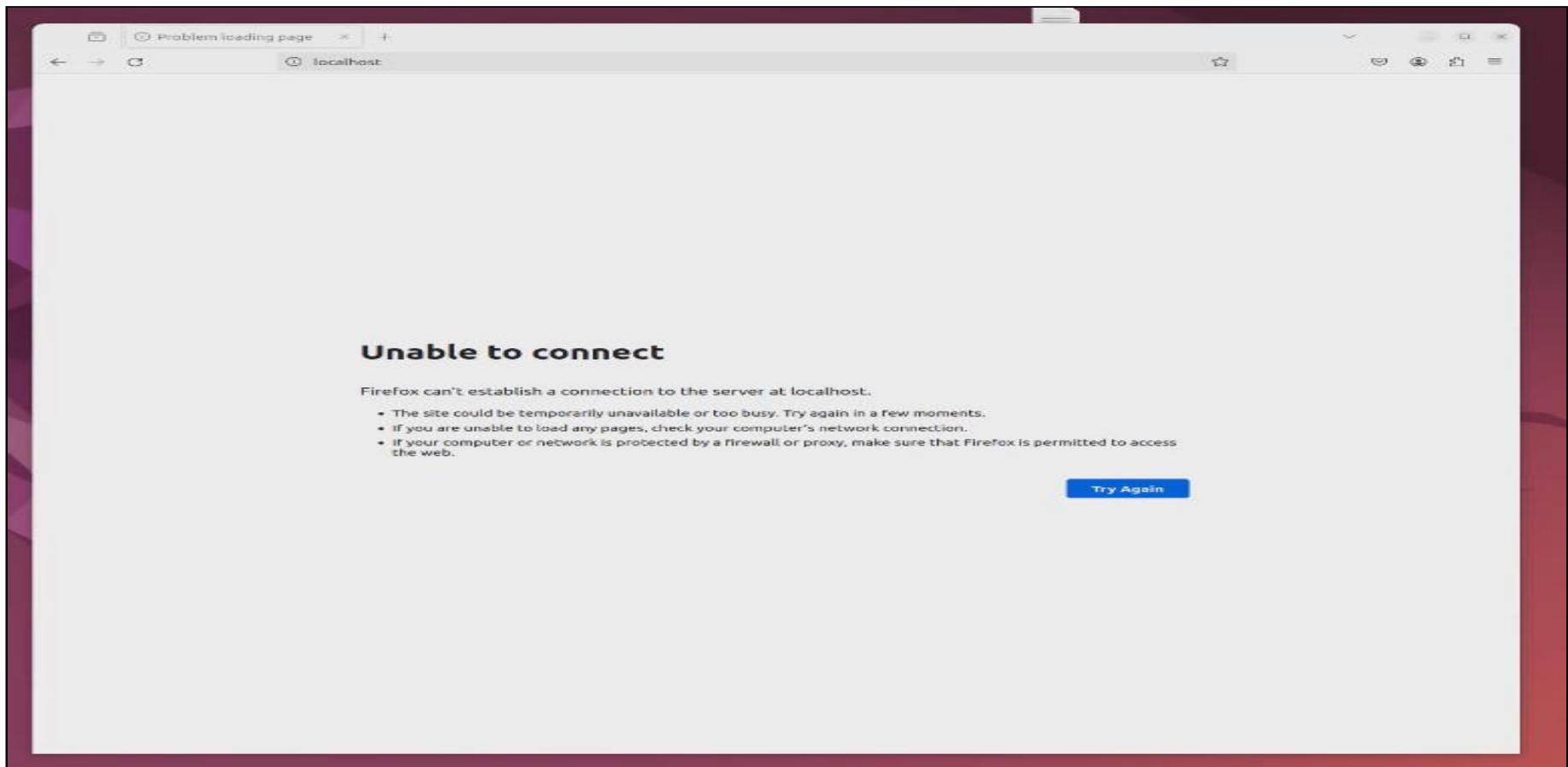
- Downgrade attack

The attacker tries to make the system to use an older insecure version protocol, cryptographic algorithm with known vulnerabilities
- CRIME attack (Compression Ratio Info-leak Made Easy)

The attacker uses a vulnerability that exploits the use of data compression in HTTPS connections, observing the size of compressed HTTPS responses
- BREACH attack (Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext)

The attacker uses a vulnerability to view encrypted traffic and force the victim to send HTTP request to a vulnerable server

OpenSSL basics: Demo with SSL/TLS on Apache web server



TCP sequence prediction attack

s7300-1-nmap.pcap

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.20.3.157	172.20.3.105	S7COMM	87	ROSCTR:[Userdata] Function:[Request] -> [CPU functions] -> [Read S7L] ID=0x0011 Index=0x0001

editedPackets.pcap

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.20.3.24	172.20.3.105	S7COMM	87	ROSCTR:[Userdata] Function:[Request] -> [CPU functions] -> [Read S7L] ID=0x0011 Index=0x0001

TCP sequence prediction attack

15300	221.980033	172.20.3.24	172.20.3.105	S7COMM	85 ROSCTR:[Job] Function:[Read Var]
15304	222.048001	172.20.3.24	172.20.3.105	TCP	54 60973 → 102 [ACK] Seq=414 Ack=339 Win=63661 Len=0

```
> Frame 15304: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{^
  ✓ Ethernet II, Src: VMware_77:50:37 (00:0c:29:77:50:37), Dst: Siemens_a4:9c:d2 (00:0e:8c:a4:9c:d2)
    > Destination: Siemens_a4:9c:d2 (00:0e:8c:a4:9c:d2)
    > Source: VMware_77:50:37 (00:0c:29:77:50:37)
      Type: IPv4 (0x0800)
  > Internet Protocol Version 4, Src: 172.20.3.24, Dst: 172.20.3.105
  ✓ Transmission Control Protocol, Src Port: 60973, Dst Port: 102, Seq: 414, Ack: 339, Len: 0
    Source Port: 60973
    Destination Port: 102
    [Stream index: 3]
    > [Conversation completeness: Incomplete (12)]
      [TCP Segment Len: 0]
      Sequence Number: 414      (relative sequence number)
      Sequence Number (raw): 1357744965
      [Next Sequence Number: 414      (relative sequence number)]
      Acknowledgment Number: 339      (relative ack number)
      Acknowledgment number (raw): 548007402
      0101 .... = Header Length: 20 bytes (5)
    > Flags: 0x010 (ACK)
      Window: 63661
      [Calculated window size: 63661]
      [Window size scaling factor: -1 (unknown)]
      Checksum: 0x7f22 [unverified]
      [Checksum Status: Unverified]
    <   This shows the raw value of the acknowledgment number (tcp.ack_raw), 4 bytes
  ||| Packets: 16
```

0000	00 0e 8c a4 9c d2 00 0c 29 77 50
0010	00 28 9a 8e 40 00 80 06 01 98 ac
0020	03 69 ee 2d 00 66 50 ed 8b 45 20
0030	f8 ad 7f 22 00 00

TCP sequence prediction attack

No.	Time	Source	Destination	Protocol	Length	Info
98353	932.308047	172.20.3.24	172.20.3.195	S7COMM	85	ROSCTR:[Job] Function:[Read Var]
98354	932.309511	172.20.3.105	172.20.3.24	TCP	60	102 → 56286 [ACK] Seq=302 Ack=358 Win=2048 Len=0
98357	932.354784	172.20.3.105	172.20.3.24	S7COMM	82	ROSCTR:[Ack_Data] Function:[Read Var]
98359	932.396988	172.20.3.24	172.20.3.105	TCP	54	56286 → 102 [ACK] Seq=358 Ack=330 Win=63911 Len=0
101515	962.309998	172.20.3.24	172.20.3.105	S7COMM	85	ROSCTR:[Job] Function:[Read Var]
101516	962.310411	172.20.3.105	172.20.3.24	TCP	60	102 → 56286 [ACK] Seq=330 Ack=389 Win=2048 Len=0
101517	962.335453	172.20.3.105	172.20.3.24	S7COMM	82	ROSCTR:[Ack_Data] Function:[Read Var]
101521	962.386158	172.20.3.24	172.20.3.105	TCP	54	56286 → 102 [ACK] Seq=389 Ack=358 Win=63883 Len=0
104649	992.309135	172.20.3.24	172.20.3.105	S7COMM	85	ROSCTR:[Job] Function:[Read Var]
104650	992.310392	172.20.3.105	172.20.3.24	TCP	60	102 → 56286 [ACK] Seq=358 Ack=420 Win=2048 Len=0
104651	992.341365	172.20.3.105	172.20.3.24	S7COMM	82	ROSCTR:[Ack_Data] Function:[Read Var]
104663	992.385994	172.20.3.24	172.20.3.105	TCP	54	56286 → 102 [ACK] Seq=420 Ack=386 Win=63855 Len=0
107848	1822.310143	172.20.3.24	172.20.3.105	S7COMM	85	ROSCTR:[Job] Function:[Read Var]
107849	1822.311358	172.20.3.105	172.20.3.24	TCP	60	102 → 56286 [ACK] Seq=386 Ack=451 Win=2048 Len=0
107851	1822.334887	172.20.3.105	172.20.3.24	S7COMM	82	ROSCTR:[Ack_Data] Function:[Read Var]
107853	1822.385363	172.20.3.24	172.20.3.105	TCP	54	56206 → 102 [ACK] Seq=451 Ack=414 Win=63827 Len=0
111053	1852.310663	172.20.3.24	172.20.3.105	S7COMM	85	ROSCTR:[Job] Function:[Read Var]
111054	1852.312787	172.20.3.105	172.20.3.24	TCP	60	102 → 56286 [ACK] Seq=414 Ack=482 Win=2048 Len=0
111057	1852.339937	172.20.3.105	172.20.3.24	S7COMM	82	ROSCTR:[Ack_Data] Function:[Read Var]
111059	1852.388986	172.20.3.24	172.20.3.105	TCP	54	56206 → 102 [ACK] Seq=482 Ack=442 Win=63799 Len=0
113298	1073.876068	172.20.3.24	172.20.3.105	S7COMM	87	ROSCTR:[Userdata] Function:[Request] → [CPU functions] → [Read SZL] ID=0x0011 Index=0...
113299	1073.877375	172.20.3.105	172.20.3.24	TCP	60	102 → 56286 [ACK] Seq=442 Ack=515 Win=2048 Len=0
113303	1073.912483	172.20.3.105	172.20.3.24	S7COMM	179	ROSCTR:[Userdata] Function:[Response] → [CPU functions] → [Read SZL] ID=0x0011 Index=...
113352	1074.425794	172.20.3.105	172.20.3.24	TCP	179	[TCP Retransmission] 102 → 56286 [PSH, ACK] Seq=442 Ack=515 Win=2048 Len=125
113436	1075.027048	172.20.3.105	172.20.3.24	TCP	179	[TCP Retransmission] 102 → 56286 [PSH, ACK] Seq=442 Ack=515 Win=2048 Len=125
113508	1075.630910	172.20.3.105	172.20.3.24	TCP	179	[TCP Retransmission] 102 → 56286 [PSH, ACK] Seq=442 Ack=515 Win=2048 Len=125
113625	1076.827068	172.20.3.105	172.20.3.24	TCP	179	[TCP Retransmission] 102 → 56286 [PSH, ACK] Seq=442 Ack=515 Win=2048 Len=125
113886	1079.227363	172.20.3.105	172.20.3.24	TCP	179	[TCP Retransmission] 102 → 56286 [PSH, ACK] Seq=442 Ack=515 Win=2048 Len=125
114211	1082.311078	172.20.3.24	172.20.3.105	TCP	85	[TCP Spurious Retransmission] 56286 → 102 [PSH, ACK] Seq=482 Ack=442 Win=63799 Len=31
114213	1082.312108	172.20.3.105	172.20.3.24	TCP	60	[TCP Dup ACK 113299#1] 102 → 56286 [ACK] Seq=567 Ack=515 Win=2048 Len=0
114244	1082.612329	172.20.3.24	172.20.3.105	TCP	85	[TCP Spurious Retransmission] 56286 → 102 [PSH, ACK] Seq=482 Ack=442 Win=63799 Len=31
114245	1082.613344	172.20.3.105	172.20.3.24	TCP	60	[TCP Dup ACK 113299#2] 102 → 56286 [ACK] Seq=567 Ack=515 Win=2048 Len=0
114303	1083.219368	172.20.3.24	172.20.3.105	TCP	85	[TCP Spurious Retransmission] 56286 → 102 [PSH, ACK] Seq=482 Ack=442 Win=63799 Len=31
114304	1083.220820	172.20.3.105	172.20.3.24	TCP	60	[TCP Dup ACK 113299#3] 102 → 56286 [ACK] Seq=567 Ack=515 Win=2048 Len=0
114308	1083.925910	172.20.3.105	172.20.3.24	TCP	60	102 → 56286 [RST] Seq=567 Win=0 Len=0
114381	1083.925985	172.20.3.24	172.20.3.105	TCP	54	56286 → 102 [ACK] Seq=513 Ack=442 Win=63799 Len=0
114382	1083.927906	172.20.3.105	172.20.3.24	TCP	60	102 → 56286 [RST] Seq=442 Win=0 Len=0
<pre>> Source: Siemens_a4:9c:d2 (00:0e:8c:a4:9c:d2) Type: IPv4 (0x0800) > Internet Protocol Version 4, Src: 172.20.3.105, Dst: 172.20.3.24 > Transmission Control Protocol, Src Port: 102, Dst Port: 56286, Seq: 442, Ack: 515, Source Port: 102 Destination Port: 56286 [Stream index: 6] > [Conversation completeness: Complete, WITH_DATA (47)] [TCP Segment Len: 125] Sequence Number: 442 (relative sequence number) Sequence Number (raw): 3712648507 [Next Sequence Number: 567 (relative sequence number)] Acknowledgment Number: 515 (relative ack number) Acknowledgment number (raw): 1751174900 0101 Header Length: 20 bytes (5) > Flags: 0x018 (PSH, ACK) Window: 2048 [Calculated window size: 2048] [Window size scaling factor: -2 (no window scaling used)] Checksum: 0xd9f4 [unverified] [Checksum Status: Unverified] Urgent Pointer: 0 > [Timestamps] > [SEQ/ACK analysis] TCP payload (125 bytes)</pre>						
<pre>^ 0000 00 0c 29 77 50 37 00 0e -8c a4 9c d2 00 05 45 00 0010 00 a5 38 4c 00 00 1e 06 85 5e ac 14 03 69 ac 14 0020 03 18 06 66 db de dd 4a 85 3b 68 60 ce f4 50 18 0030 00 00 d9 f4 00 00 03 00 00 07 d2 f0 80 32 07 00 0040 00 00 00 00 0c 00 00 00 01 12 08 12 84 01 04 00 0050 00 00 00 ff 09 05 c0 00 11 00 00 00 1c 00 03 00 0060 01 36 45 53 37 20 33 31 34 2d 31 41 45 30 34 2d 0070 38 41 42 30 20 00 c0 00 02 00 00 00 36 45 53 0080 37 20 33 31 34 2d 31 41 45 30 34 2d 30 41 42 30 0090 20 00 c0 00 02 00 00 07 20 28 20 20 20 20 20 20 00a0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 00 c0 56 00b0 01 02 01</pre>						
<p style="text-align: right;">Activate Windows Go to Settings to activate Windows. Activate Windows</p>						

Public Key Infrastructure (PKI)

- PKI provides a means to establish trust binding public keys and identities with certificates
- With PKI we are sure that data are decrypted with corresponding private key
- If we combine this with a hash to create a signature, we can be sure that the encrypted data has not been tampered
- Certificates can be signed with the issuer private key with all info to validate the identity

Certification Authorities

- A private CA that issues certificates locally i.e., for an organization trusted by its members
- Public CAs that issue certificates publicly for members and must be trusted by the public (third party CA certificates)
- A CA must be trusted, to extend trust and the certificate includes the public key are freely distributed

CA 必须是可信的，以扩展信任和
证书包含的公钥是自由的
分散式

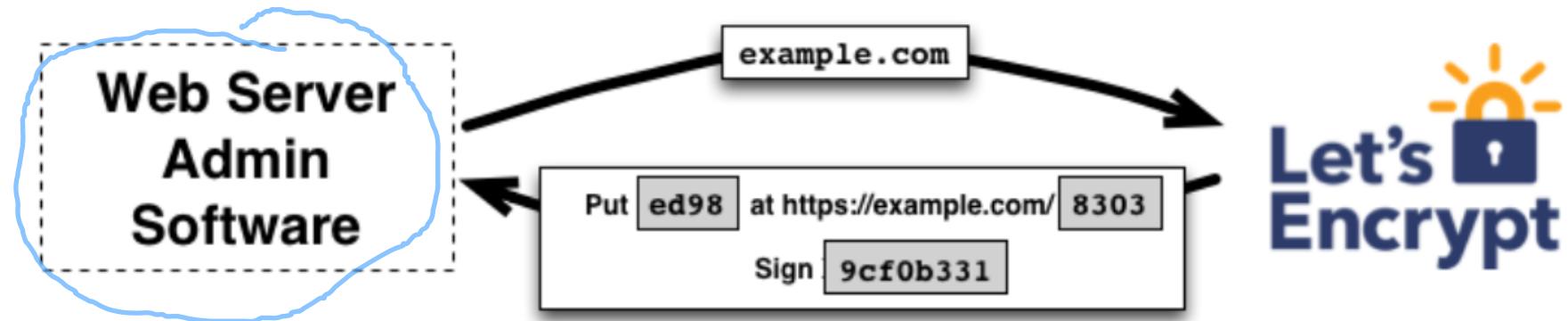
Let's Encrypt key principles

可以自动授予浏览器信任的公共 CA
免费 HTTPS 服务器证书

- A public CA that can automatically grant a browser-trusted certificate for an HTTPS server for free
- The prerequisite is to have a valid registered domain name and install a certificate management agent on the web server (Certbot)
- Free and open certificate authority (CA) by the Internet Security Research Group (ISRG)
- Provides security with TLS security best practices for admins to secure their websites
- Offers transparency, certificates issued will be publicly available for anyone to inspect

Let's Encrypt CA Basics

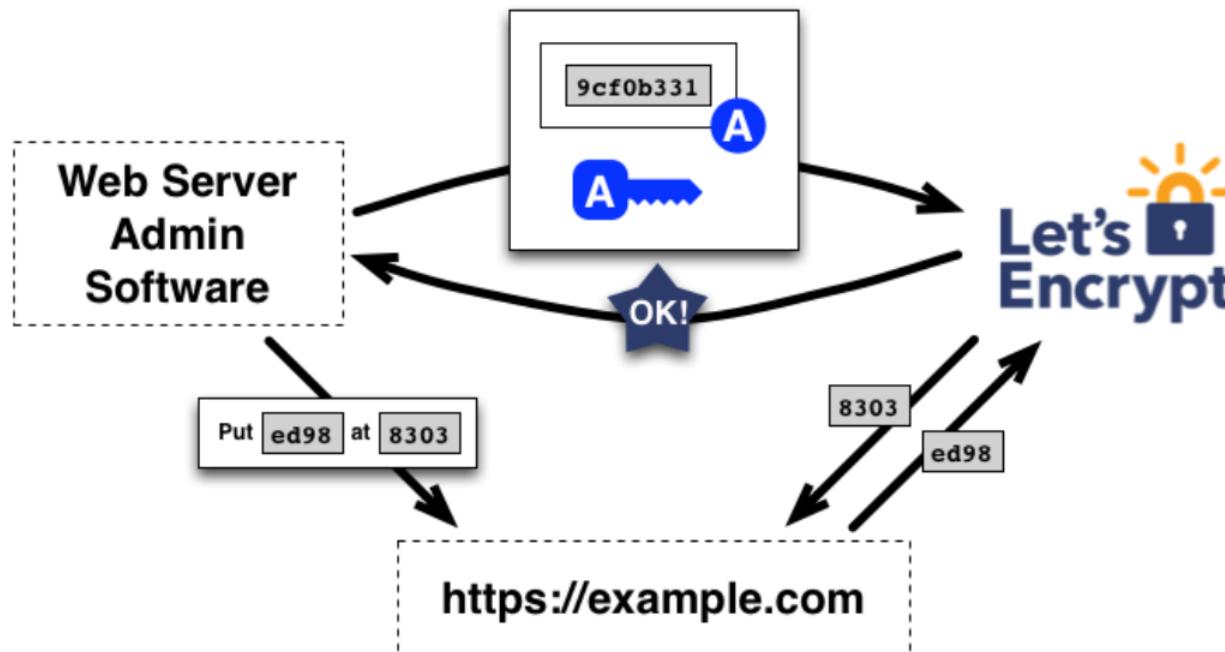
1. Let's Encrypt identifies the server admin with the public key. The installed agent generates a new key pair and informs Let's Encrypt that the server controls a domain
2. Let's Encrypt CA, will issue a set of challenges, for example:
 - Provide the DNS record
 - Provide an HTTP resource
 - Sign an arbitrary number (nonce) with private key



Let's Encrypt CA

4. The agent completes the tasks, and the CA validates the signature of the nonce and the task(s), grants the agent the ability to request, renew and revoke certificates

代理完成任务，CA 验证随机数和任务的签名，
授予代理请求、更新和撤销证书的能力



Let's Encrypt CA

- The agent constructs a **Certificate Signing Request** with a signature (public key) and ask Let's Encrypt to issue a certificate for the domain with it's public key (whole again CSR signed with private key)

代理使用签名（公钥）构建证书签名请求并询问让我们加密以使用其公钥为域颁发证书（再次由 CSR 签名）私钥）

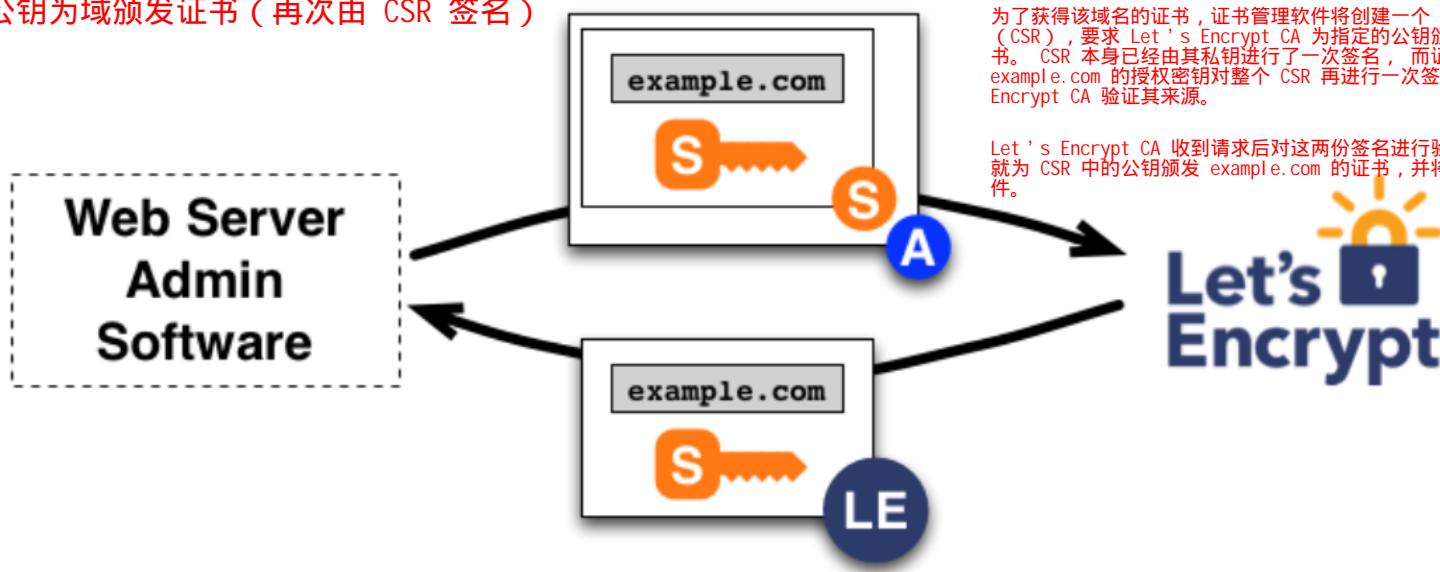
网络资料介绍：

证书颁发和吊销

管理软件具备授权密钥后，证书的申请、续期、吊销操作就简单了，只需将各类证书管理指令用授权密钥签名后发给 CA 即可。

为了获得该域名的证书，证书管理软件将创建一个 PKCS#10 证书签名请求（CSR），要求 Let's Encrypt CA 为指定的公钥颁发 example.com 的证书。CSR 本身已经由其私钥进行了一次签名，而证书管理软件还会用 example.com 的授权密钥对整个 CSR 再进行一次签名，以便 Let's Encrypt CA 验证其来源。

Let's Encrypt CA 收到请求后对这两份签名进行验证，如果全部通过，就为 CSR 中的公钥颁发 example.com 的证书，并将证书文件发给管理软件。



- Let's Encrypt CA gets the request and verifies both signatures and then issues the certificate for the domain and sends it to the server

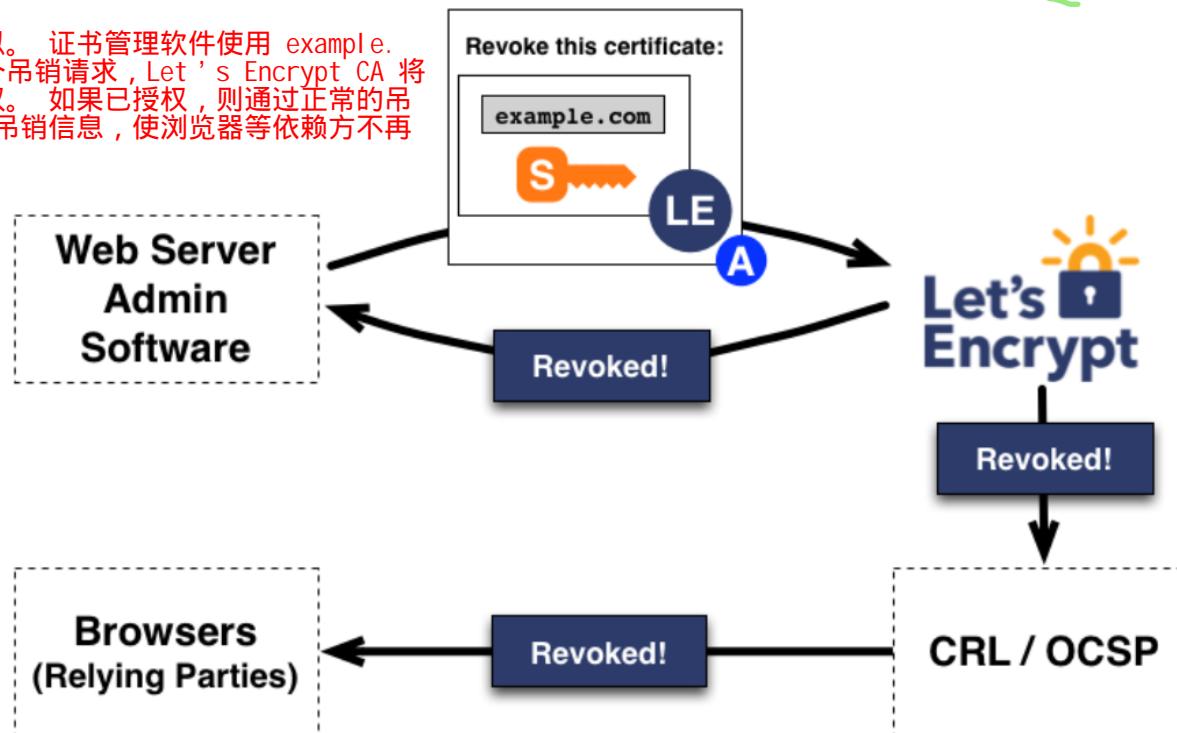
Let's Encrypt CA

撤销的工作原理类似，代理签署撤销请求，然后 Let's Encrypt CA 验证请求并授权，浏览器然后停止接受无效证书

- Revocation works similarly, the agent signs a revocation request and then Let's Encrypt CA verifies the request and authorize, browsers then stop accepting the invalidate certificate

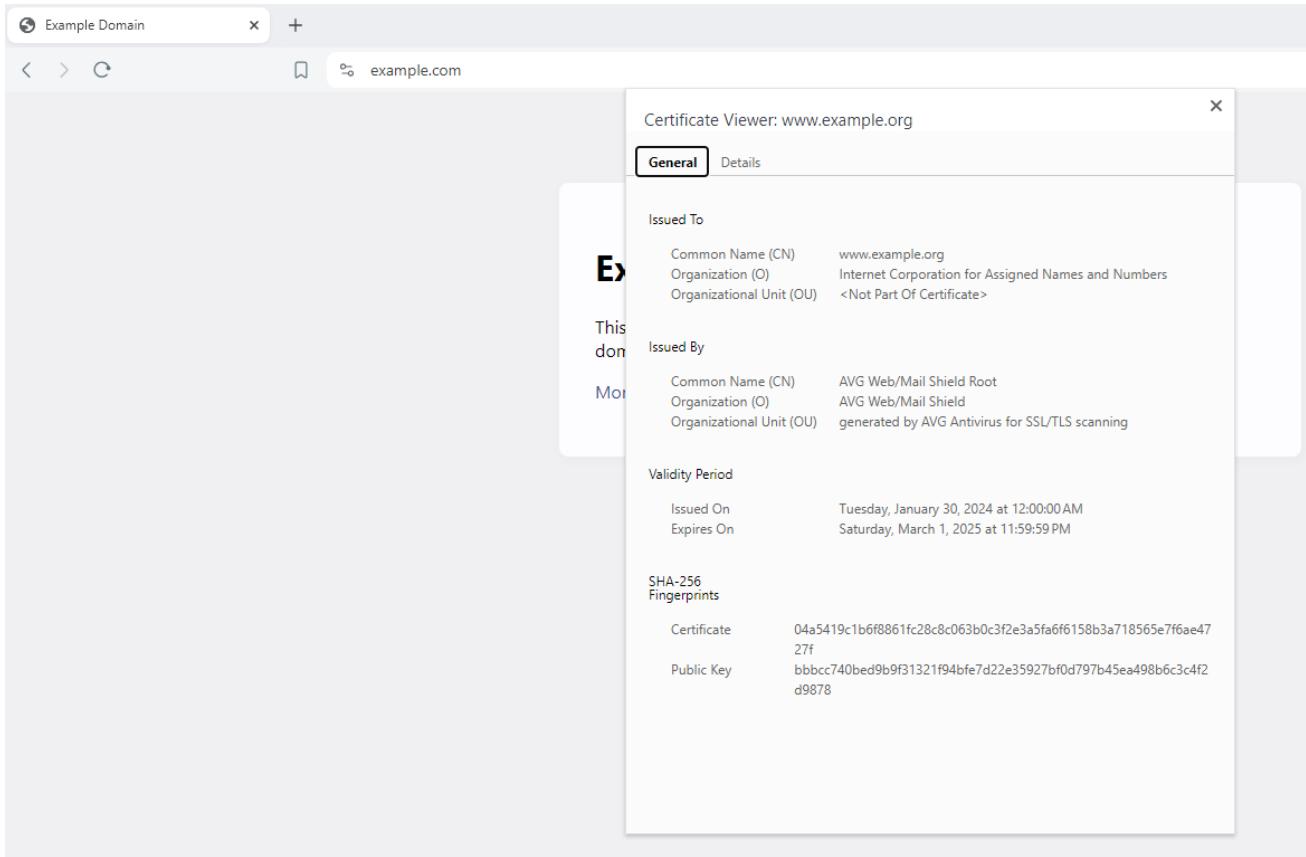
网络的介绍：

申请吊销证书的流程类似。证书管理软件使用 example.e.com 的授权私钥签署一个吊销请求，Let's Encrypt CA 将验证该请求是否已被授权。如果已授权，则通过正常的吊销通道（即 OCSP）发布吊销信息，使浏览器等依赖方不再接受这份证书。



Let's Encrypt CA

Let's see a public example which is implemented this way: <https://example.com/>



Introduction to PGP - Overview

- PGP released in 1991 by Phil Zimmermann → de facto standard for secure exchange of information
- Today PGP has become an open standard known as OpenPGP
- PGP can encrypt messages online: email, plain text files etc.
- Close to military-grade symmetric and asymmetric encryption
- Relies on a private key (kept safe), integrity checking, message authentication and signed certificates
- PGP is slow therefore not considered for use in application

Suggested resources for further reading

Network Security with OpenSSL: Cryptography for Secure Communications

Authors: John Viega, Matt Messier , Pravir Chandra

Have any questions?

