东南大学 SOUTHEAST UNIVERSITY
网络空间安全学院
School of Cyber Science and Engineering

网络空间安全实验基础

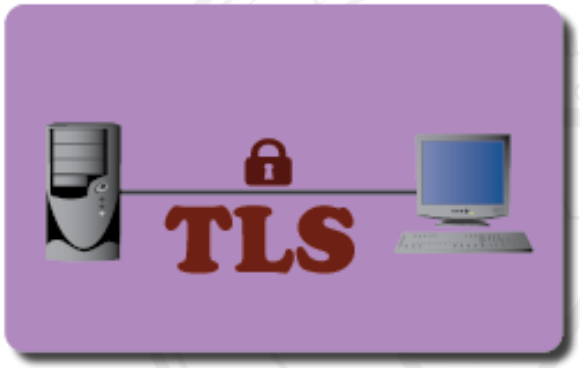
Hands-on Cybersecurity – Fundamentals

传输层安全协议（TLS）

主讲人：张玉健

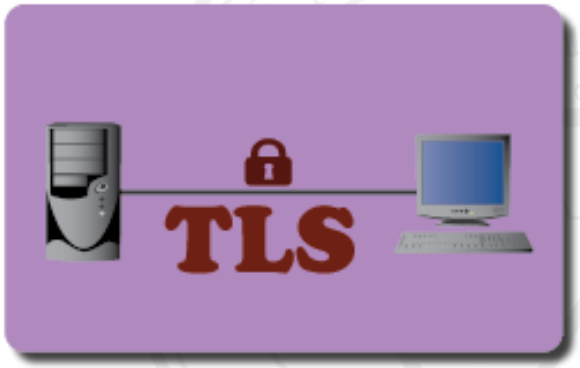
课程提纲

- TLS协议概览
- TLS握手
- TLS数据传输
- TLS客户端编程
- TLS服务端编程
- TLS代理编程



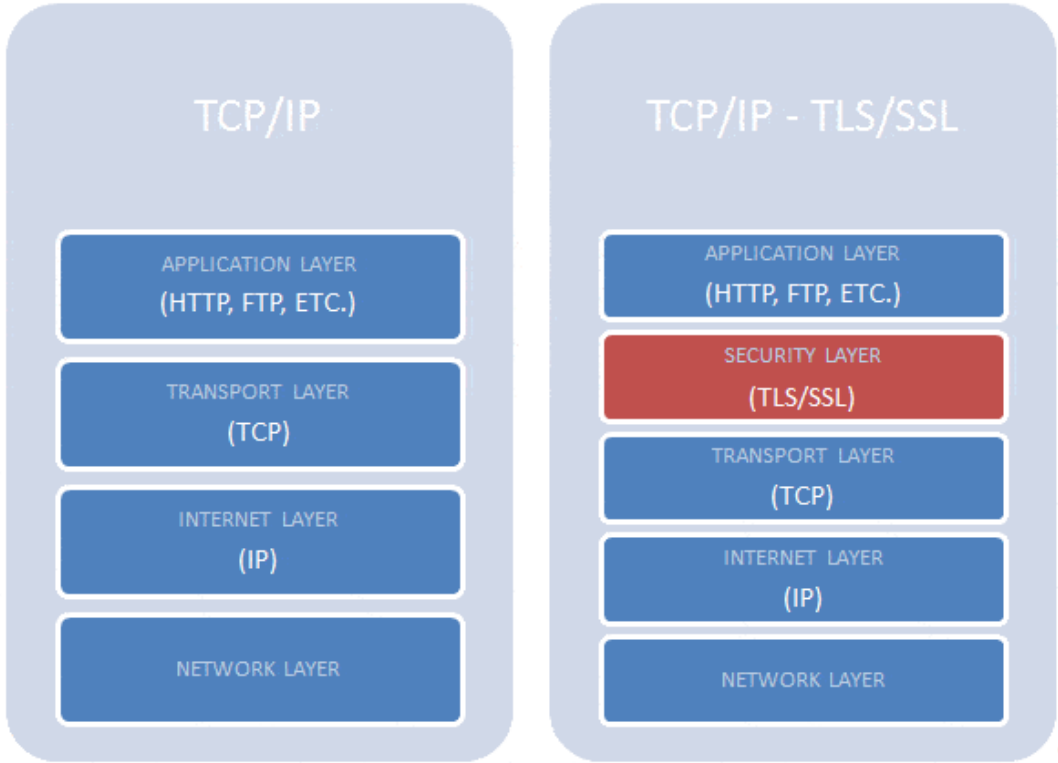
课程提纲

- TLS协议概览
- TLS握手
- TLS数据传输
- TLS客户端编程
- TLS服务端编程
- TLS代理编程



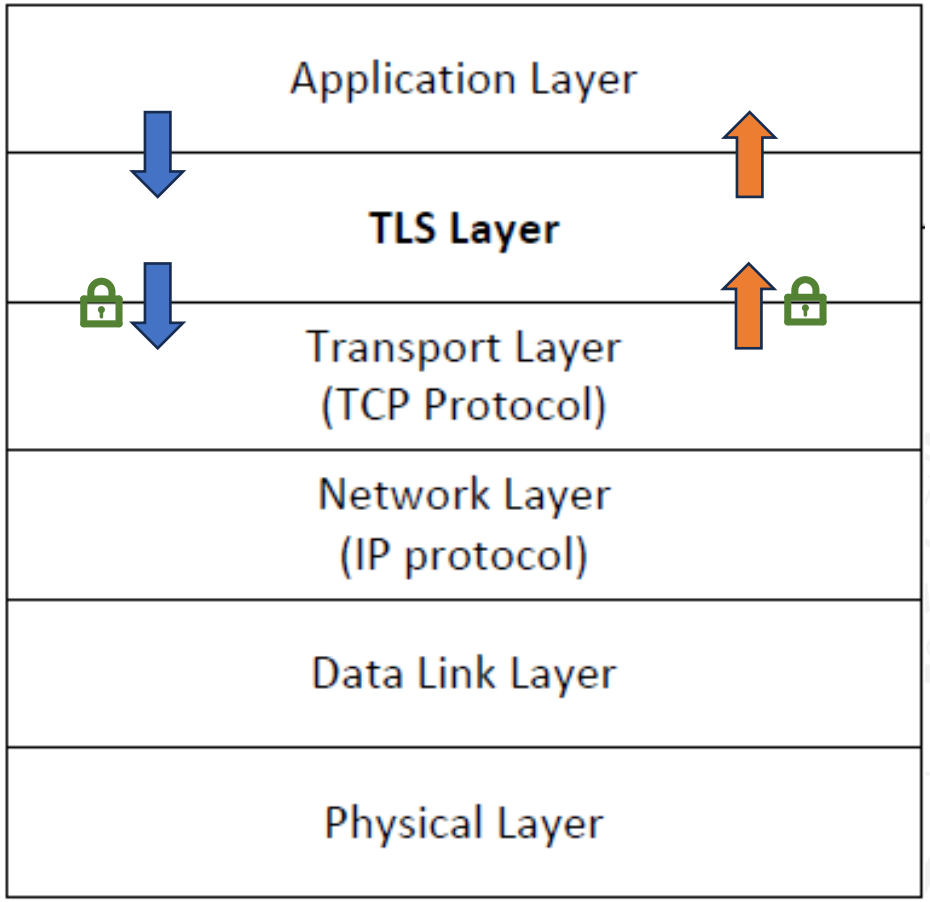
引入TLS目的

- **TLS（Transport Layer Security）的目的**
  - TCP/IP协议栈的问题：payload明文，极易被窃听、篡改、仿冒等
  - TLS的作用：为网络通信提供机密性（Confidentiality）、认证性（Authentication）及数据完整性（Integrity）保障



TLS层的工作

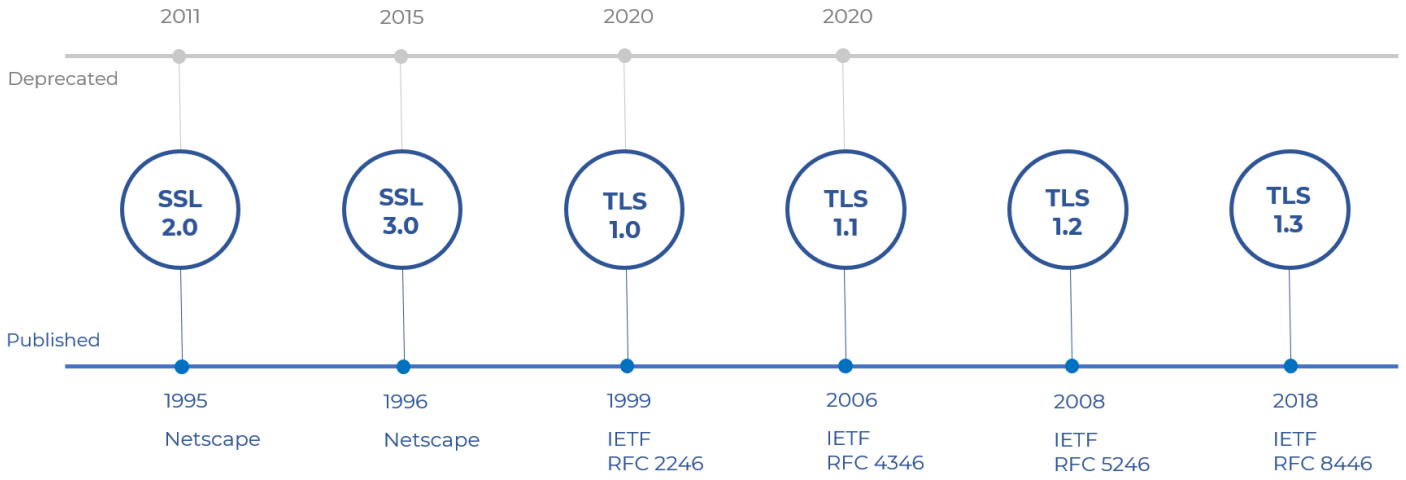
- **TLS介于传输层和应用层之间**
  - 从应用层看：向/从TLS层发送/收取数据明文
  - 从TLS层看：处理数据加密/解密、完整性验证等
  - 从传输层看：向/从TLS层发送/收取数据密文



TLS的发展历程

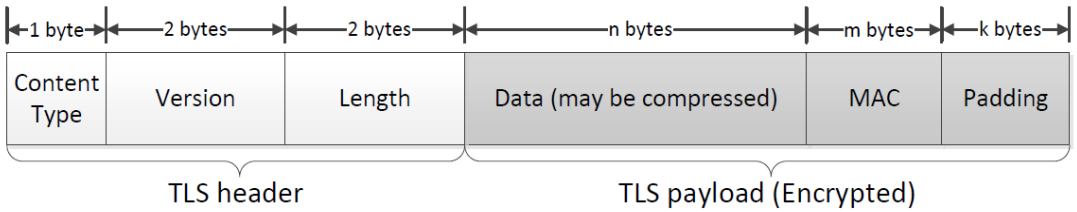
➢ TLS的发展时间线

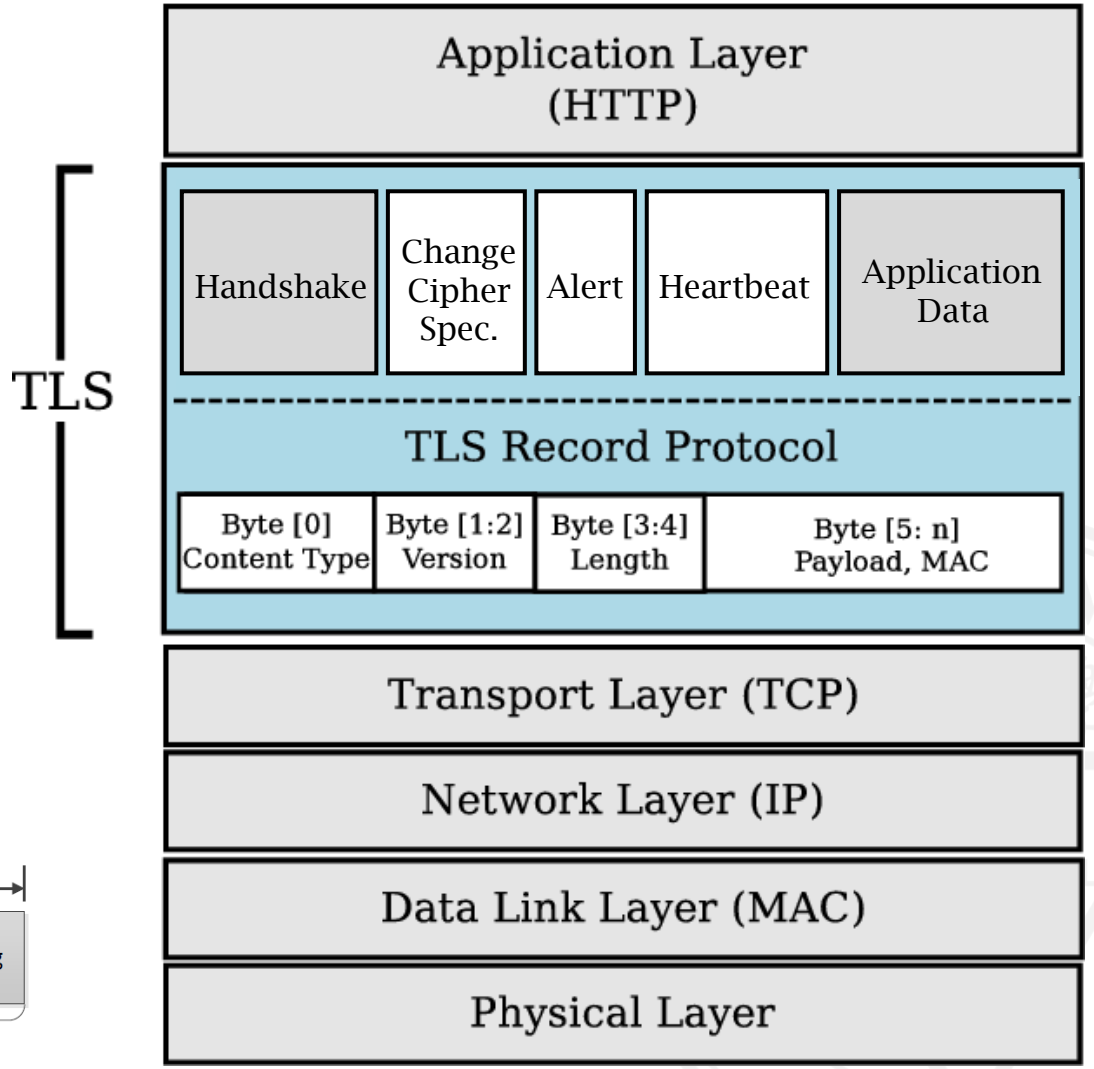
- 1994年，网景（NetScape）公司设计了 SSL 1.0
- 1995年，SSL 2.0，存在严重漏洞
- 1996年，SSL 3.0，得到大规模应用
- 1999年，IETF 对 SSL 进行标准化，发布了 TLS 1.0
- 2006年和2008年，TLS 进行了两次升级，分别为 TLS 1.1 和 TLS 1.2
- 2018年，TLS 1.3作为建议标准发布于RFC 8446



TLS协议框架

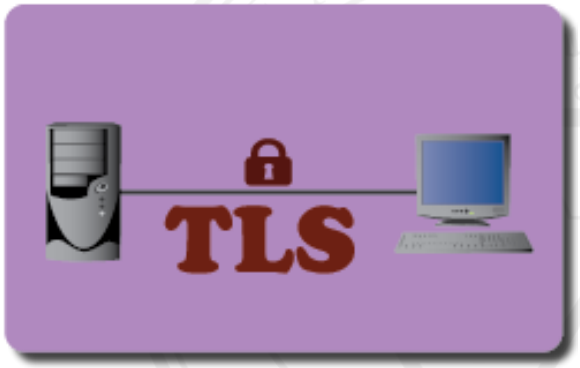
- **TLS的子协议**
  - Handshake
  - Change Cipher Spec.(deprecated in TLS 1.3)
  - Alert
  - Heartbeat (extended by RFC6520)
  - Application Data
- **TLS记录层（Record）**
  - 作用：提供数据封装、压缩、加/解密等功能
  - 格式：type、version、length、payload
  - payload长度：最大2^{14}





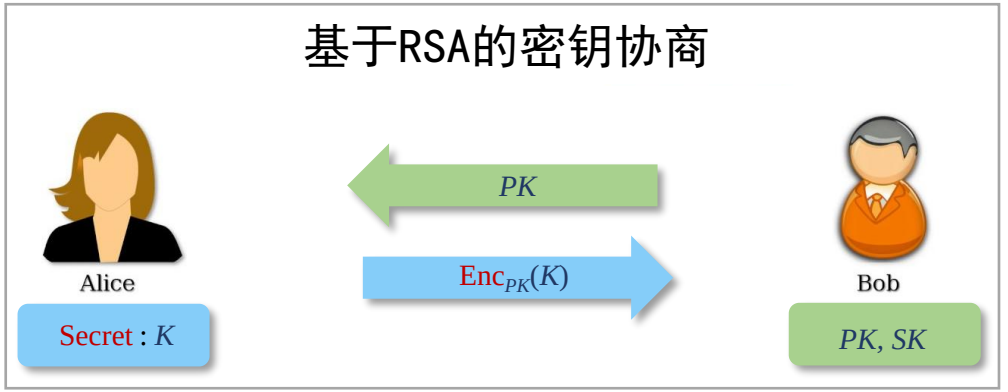
课程提纲

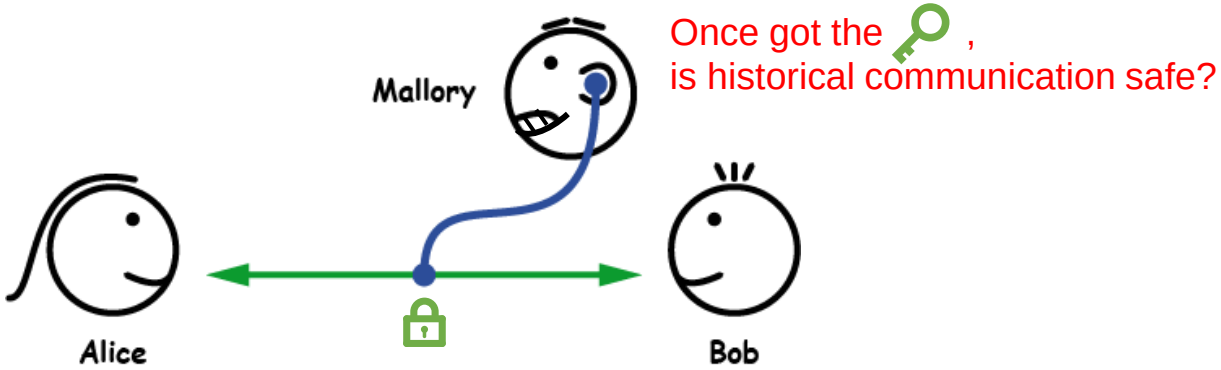
- TLS协议概览
- TLS握手
- TLS数据传输
- TLS客户端编程
- TLS服务端编程
- TLS代理编程

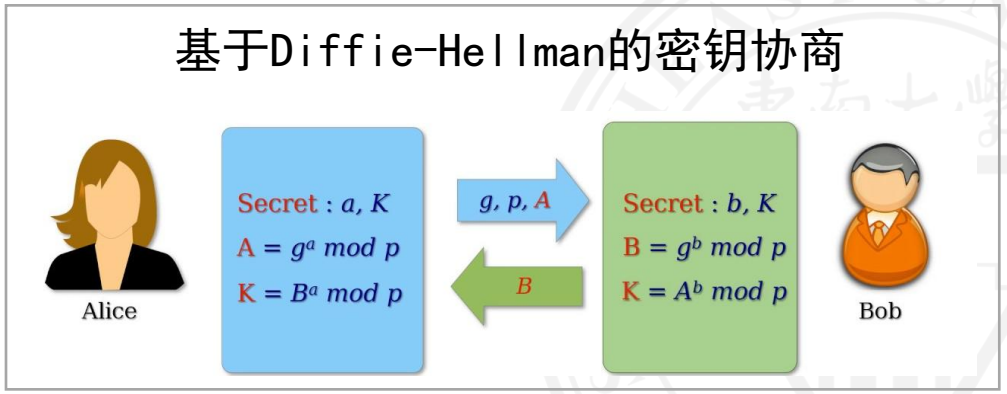


TLS握手的目的和选择

- **密钥协商**
  - 基于RSA
  - 基于Diffie-Hellman（DH）
- **认证**
  - RSA已提供认证功能，直接使用？
  - DH未提供（基于DH的证书几乎没有）







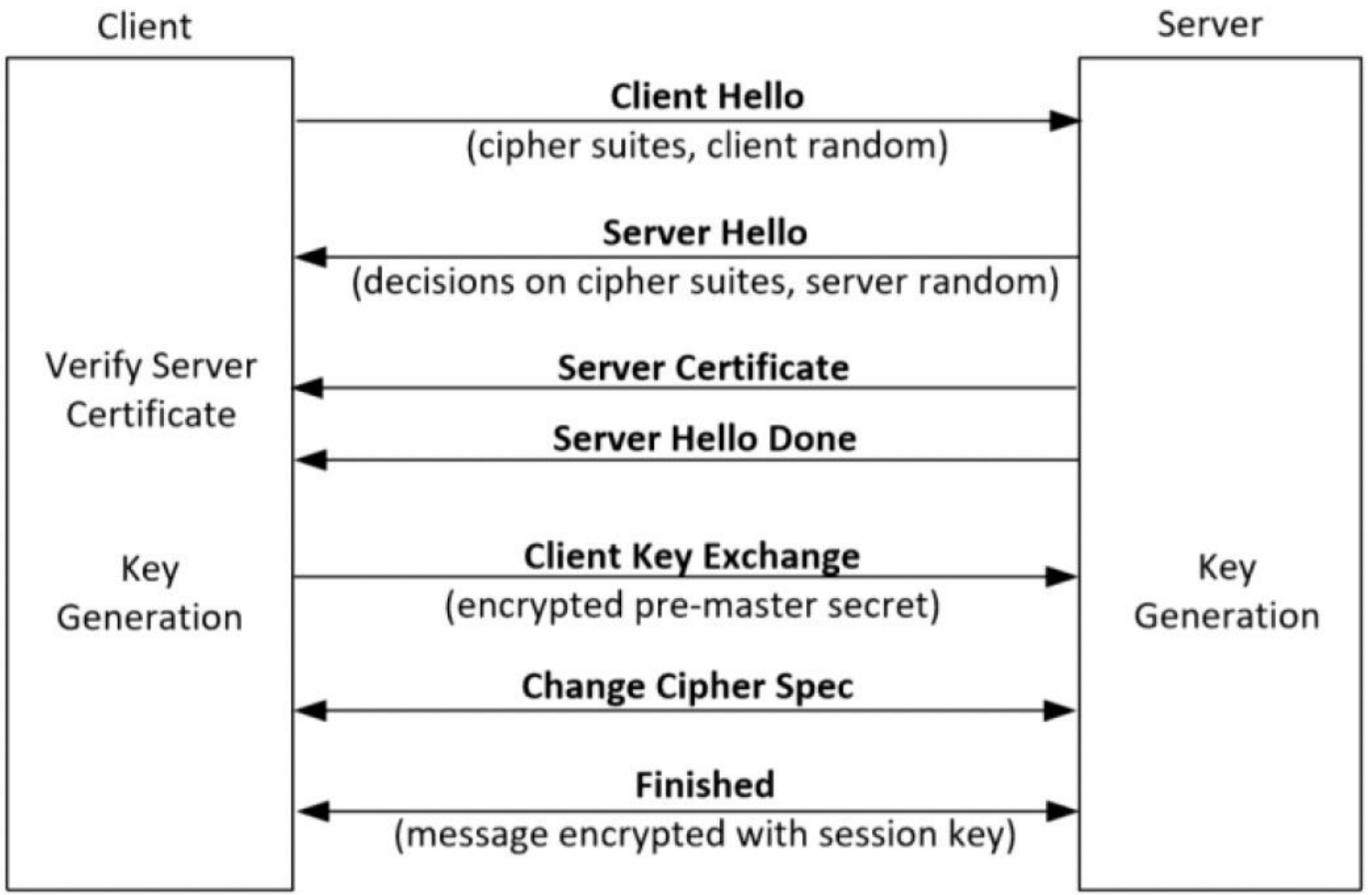
完美前向保密（Perfect Forward Secrecy）问题

TLS的加密套件

- **加密套件（Cipher Suite）**
  - Key Exchange：密钥协商算法
  - Authentication：身份认证算法
  - Encryption：对称加密算法
  - MAC(Message Authentication Code)：消息认证码算法

```
Handshake Type: Client Hello (1)
Length: 311
Version: TLS 1.2 (0x0303)
Random
Session ID Length: 0
Cipher Suites Length: 158
Cipher Suites (79 suites)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xc024)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
    Cipher Suite: TLS_SRP_SHA_DSS_WITH_AES_256_CBC_SHA (0xc022)
```

TLS握手协议（v1.2）


Client
Server
Client Hello
(cipher suites, client random)
Server Hello
(decisions on cipher suites, server random)
Verify Server Certificate
Server Certificate
Server Hello Done
Key Generation
Client Key Exchange
(encrypted pre-master secret)
Key Generation
Change Cipher Spec
Finished
(message encrypted with session key)

TLS握手协议（v1.2）报文

TCP握手

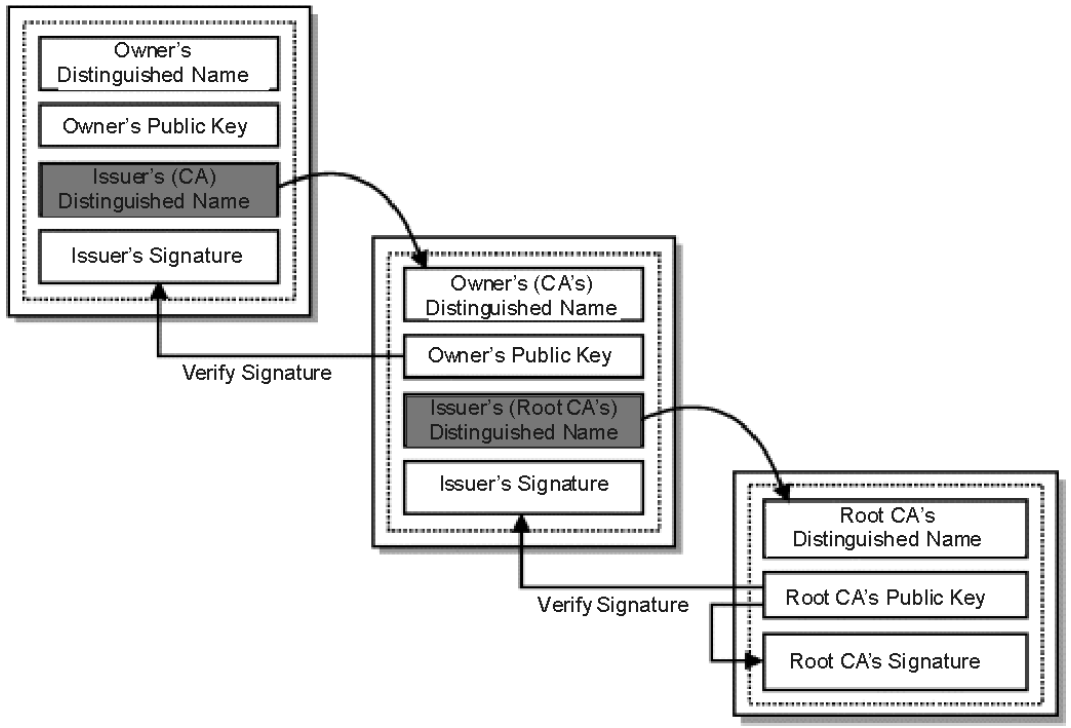
```
10.0.5.5          93.184.216.34     TCP       74 36460 → 443 [SYN] Seq=1634118127 Win=64240 …
93.184.216.34     10.0.5.5          TCP       60 443 → 36460 [SYN, ACK] Seq=1590143 Ack=1634…
10.0.5.5          93.184.216.34     TCP       54 36460 → 443 [ACK] Seq=1634118128 Ack=159014…
```

```
10.0.5.5          93.184.216.34     TLSv1.2   272 Client Hello
93.184.216.34     10.0.5.5          TLSv1.2  1514 Server Hello
93.184.216.34     10.0.5.5          TLSv1.2   654 Certificate, Server Key Exchange, Server Hello Done
10.0.5.5          93.184.216.34     TLSv1.2   180 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
93.184.216.34     10.0.5.5          TLSv1.2   280 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
```

TLS握手

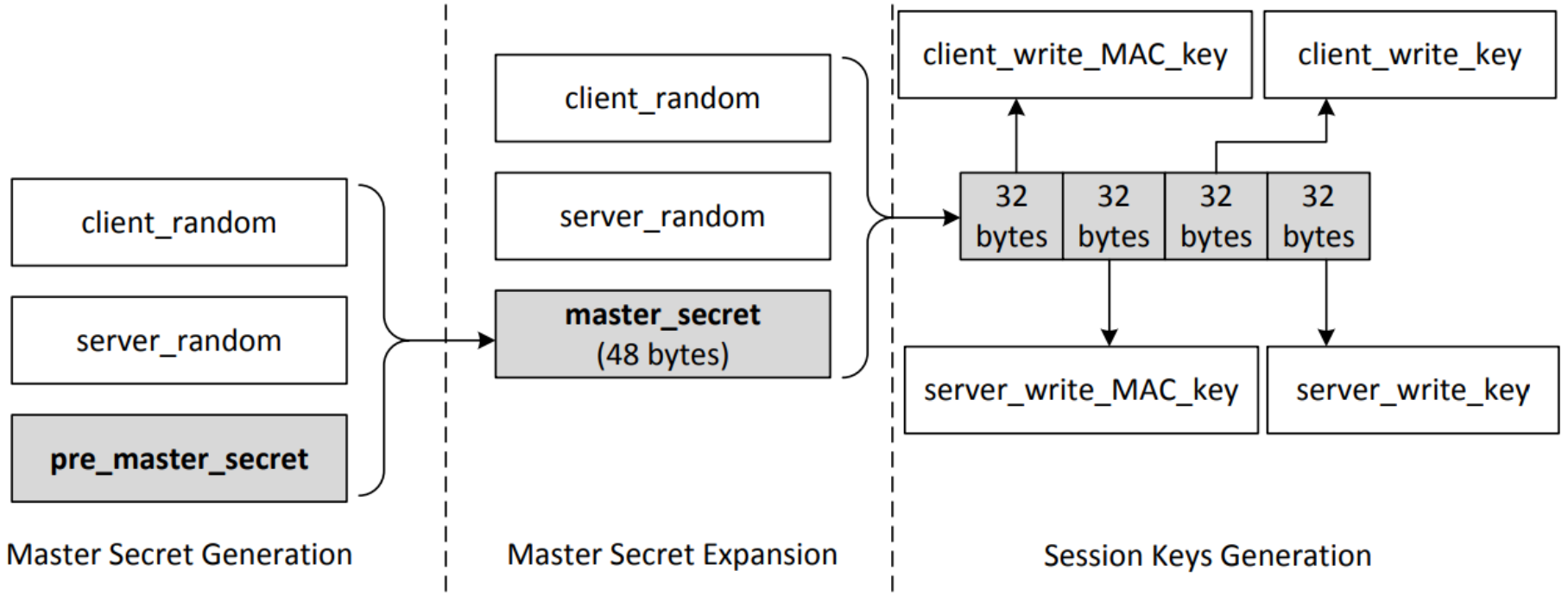
TLS握手中的证书验证

- **客户端验证服务端的证书**
  - 验证证书的有效期、签名等
  - 验证访问的域名与证书中的名称是否一致
  - 客户端需要预安装根证书（参见PKI）

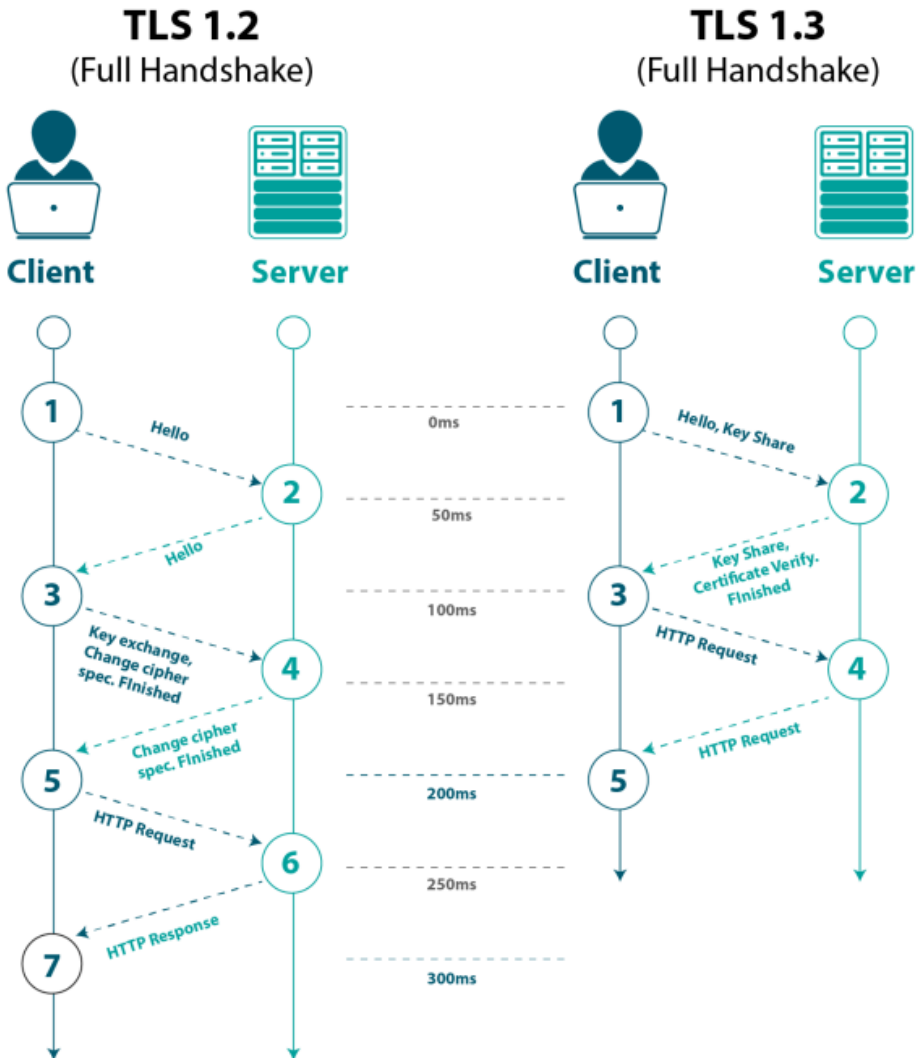


TLS握手中的密钥生成

- **会话密钥的生成**
  - 用途：后续通信过程的对称加密
  - 来源：三个随机数（client_random、server_random、pre_master_secret）



TLS握手协议（v1.3）的变化



- 密钥交换：取消基于RSA的方法，使用ECDHE
- 减少步骤：客户端提前预测并使用Cipher Suite

TLS 1.2报文

```
10.0.5.5          93.184.216.34     TCP       74 36460 → 443 [SYN] Seq=1634118127 Win=64240 …
93.184.216.34     10.0.5.5          TCP       60 443 → 36460 [SYN, ACK] Seq=1590143 Ack=1634…
10.0.5.5          93.184.216.34     TCP       54 36460 → 443 [ACK] Seq=1634118128 Ack=159014…
```

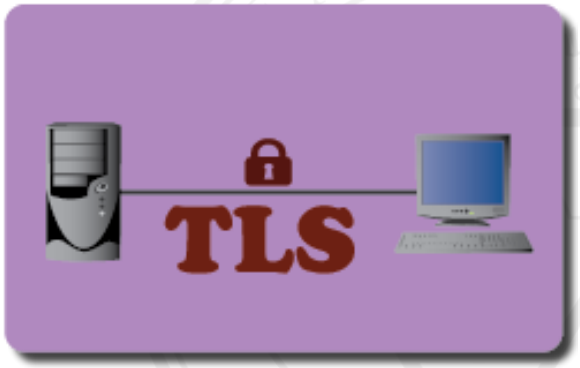
```
10.0.5.5          93.184.216.34     TLSv1.2   272 Client Hello
93.184.216.34     10.0.5.5          TLSv1.2  1514 Server Hello
93.184.216.34     10.0.5.5          TLSv1.2   654 Certificate, Server Key Exchange, Server Hello Done
10.0.5.5          93.184.216.34     TLSv1.2   180 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
93.184.216.34     10.0.5.5          TLSv1.2   280 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
```

TLS 1.3报文

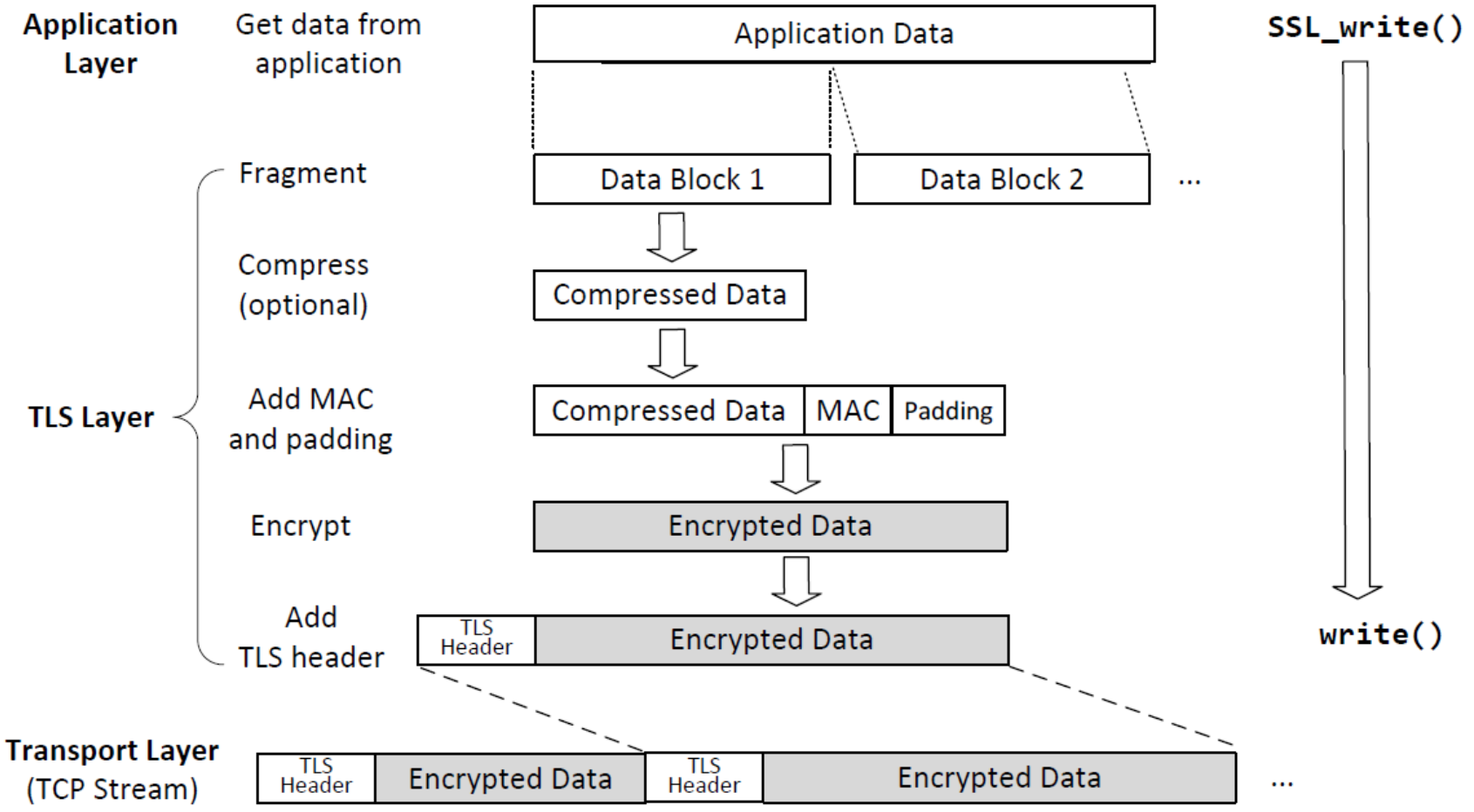
```
10.0.5.5          93.184.216.34     TLSv1.3   571 Client Hello
93.184.216.34     10.0.5.5          TLSv1.3  1514 Server Hello, Change Cipher Spec, Applicati…
93.184.216.34     10.0.5.5          TLSv1.3  1301 Application Data, Application Data, Applica…
10.0.5.5          93.184.216.34     TLSv1.3   134 Change Cipher Spec, Application Data
10.0.5.5          93.184.216.34     TLSv1.3   224 Application Data
93.184.216.34     10.0.5.5          TLSv1.3   691 Application Data, Application Data, Applica…
10.0.5.5          93.184.216.34     TLSv1.3    85 Application Data
```

课程提纲

- TLS协议概览
- TLS握手
- TLS数据传输
- TLS客户端编程
- TLS服务端编程
- TLS代理编程

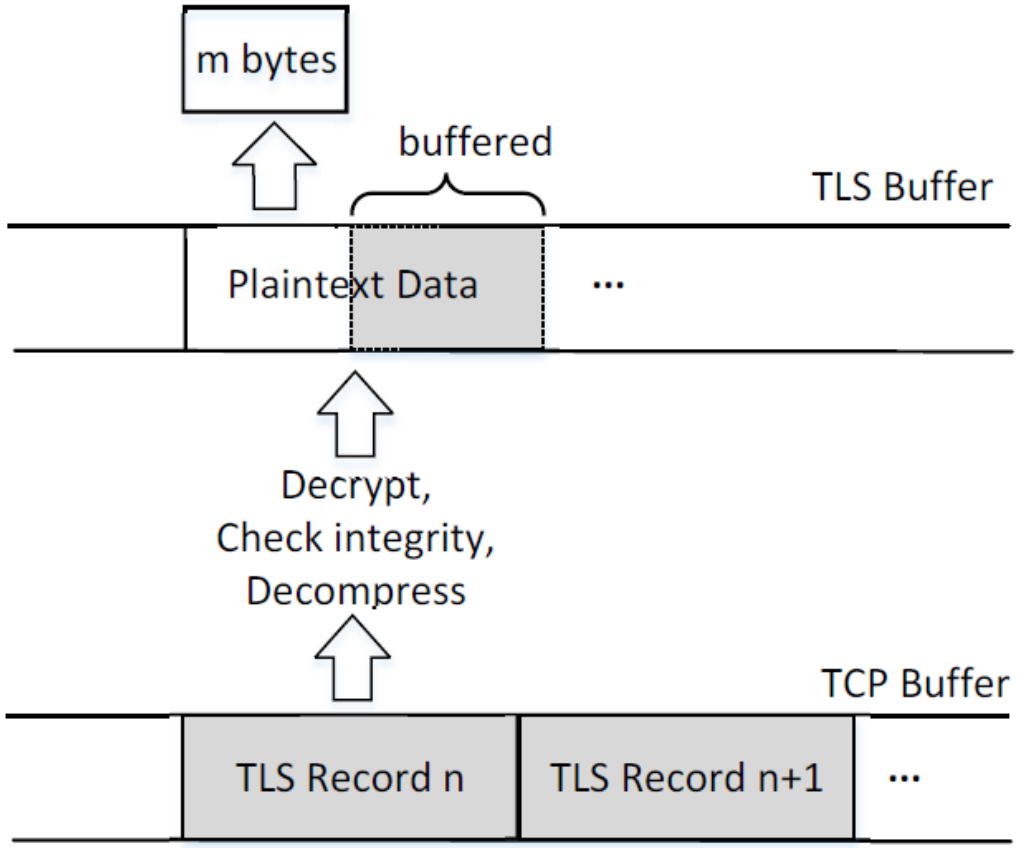


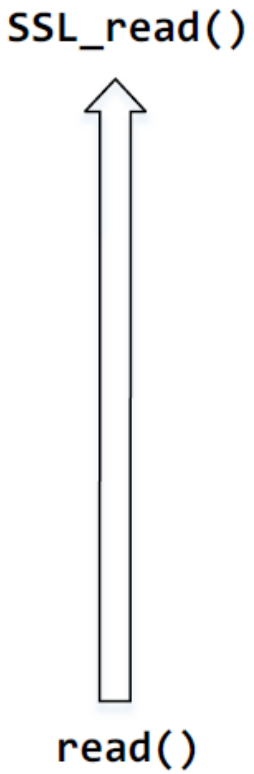
通过TLS发送数据



- 封装
- 压缩
- 加密

通过TLS接收数据

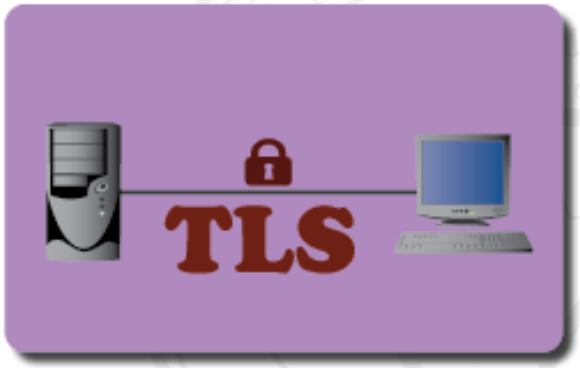




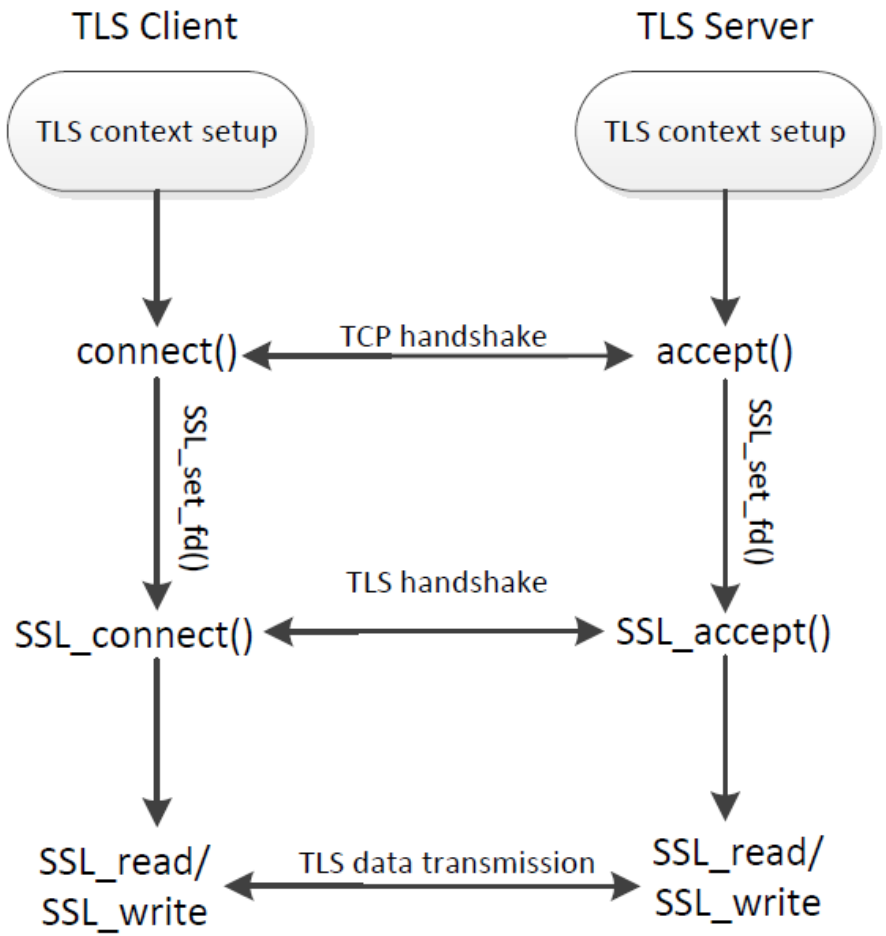
- 解密
- 解压缩
- 完整性校验
- 用户态缓存

课程提纲

- TLS协议概览
- TLS握手
- TLS数据传输
- TLS客户端编程
- TLS服务端编程
- TLS代理编程



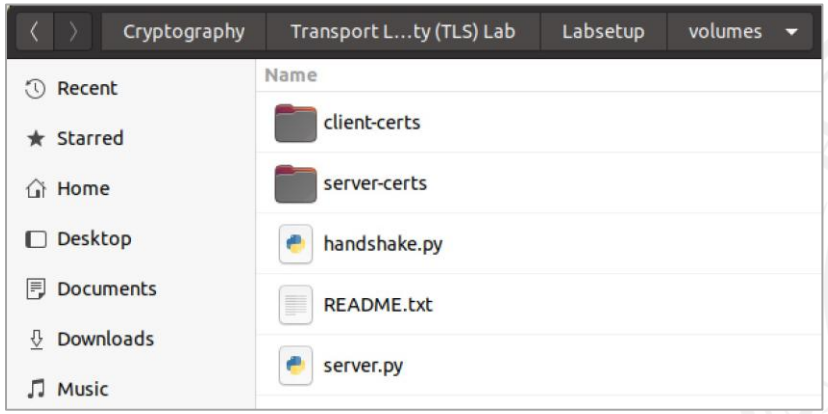
TLS编程总览



启动容器

```
seed@VM: ~/.../Labsetup
[08/20/23]seed@VM:~/.../Labsetup$ docker-compose up
Starting server-10.9.0.43      ... done
Starting mitm-proxy-10.9.0.143 ... done
Starting client-10.9.0.5       ... done
Attaching to server-10.9.0.43, client-10.9.0.5, mitm-proxy-10.9.0.143
```

为降低实验难度，大部分代码已给出



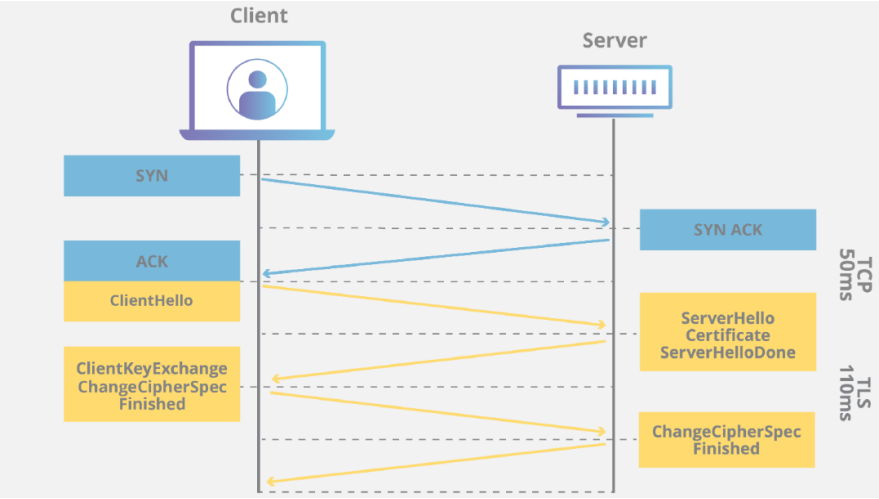
与真实网站TLS握手（实验任务一. a）

- **与真实网站（任选HTTPS）建立TLS握手连接**
  - 实验步骤

    ① 在虚拟机VM的当前实验目录下运行hankshake. py

    ```
    ./handshake.py www.alipay.com
    ```

  - 要求：记录程序、操作命令和输出结果，写入实验报告，对实验现象进行解释。




```python
#!/usr/bin/env python3

import socket
import ssl
import sys
import pprint

hostname = sys.argv[1]
port = 443
cadir = '/etc/ssl/certs'
#cadir = './client-certs'

# Set up the TLS context
context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)  # For Ubuntu 20.04 VM
# context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)      # For Ubuntu 16.04 VM

context.load_verify_locations(capath=cadir)
context.verify_mode = ssl.CERT_REQUIRED
context.check_hostname = True

# Create TCP connection
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((hostname, port))
input("After making TCP connection. Press any key to continue ...")

# Add the TLS
ssock = context.wrap_socket(sock, server_hostname=hostname,
                            do_handshake_on_connect=False)
ssock.do_handshake()   # Start the handshake
print("=== Cipher used: {}".format(ssock.cipher()))
print("=== Server hostname: {}".format(ssock.server_hostname))
print("=== Server certificate:")
pprint.pprint(ssock.getpeercert())
pprint.pprint(context.get_ca_certs())
input("After TLS handshake. Press any key to continue ...")

# Close the TLS Connection
ssock.shutdown(socket.SHUT_RDWR)
ssock.close()

```

与真实网站TLS握手（实验任务一.b）

➢ 更换根证书目录

- 实验步骤

① 修改hankshake.py的证书目录

```
cadir = './client-certs'
```

注意：此时该目录为空

② 将此前访问网站验证时需要的根证书复制到./client-certs中，并生成哈希索引

```
$ openssl x509 -in someCA.crt -noout -subject_hash
4a6481c9

$ ln -s someCA.crt 4a6481c9.0
$ ls -l
total 4
lrwxrwxrwx 1 ... 4a6481c9.0 -> someCA.crt
-rw-r--r-- 1 ... someCA.crt
```

Root CA Certificate

in case of hash collision

③ 再次运行hankshake.py观察结果

- 要求：记录程序、操作命令和输出结果，写入实验报告，对实验现象进行解释。

与真实网站TLS握手（实验任务一.c）

➢ 体会检查主机名称的重要性

- 实验步骤

① 获取目标网站的IP地址

```
$ dig www.example.com
...
;; ANSWER SECTION:
www.example.com.   403    IN A   93.184.216.34
```

② 通过修改/etc/hosts新建一个域名指向该IP地址（也可通过DNS劫持）

```
93.184.216.34     www.alipay.com
```

③ 修改handshake.py中的check_hostname选项

```
context.check_hostname = False  # try both True and False
```

④ 运行hankshake.py观察结果

思考：如果www.example.com是钓鱼网站将会怎样？

- 要求：记录程序、操作命令和输出结果，写入实验报告，对实验现象进行解释。

与真实网站TLS握手（实验任务一. d）

➢ 与网站进行数据传输

- 实验步骤

① 修改handshake. py，在TLS握手后增加收发数据代码

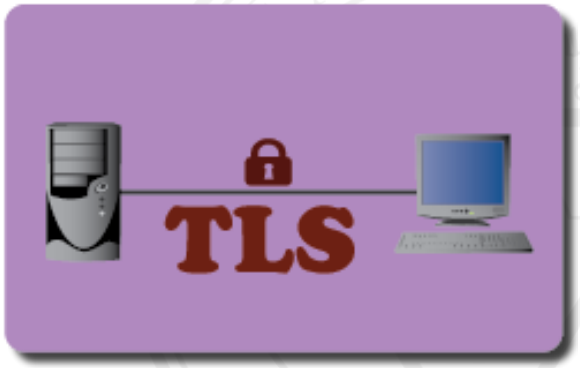
```
# Send HTTP Request to Server
request = b"GET / HTTP/1.0\r\nHost: " + \
          hostname.encode('utf-8') + b"\r\n\r\n"
ssock.sendall(request)

# Read HTTP Response from Server
response = ssock.recv(2048)
while response:
  pprint.pprint(response.split(b"\r\n"))
  response = ssock.recv(2048)
```

② 运行hankshake. py观察结果

- 要求：记录程序、操作命令和输出结果，写入实验报告，对实验现象进行解释。

课程提纲

- TLS协议概览
- TLS握手
- TLS数据传输
- TLS客户端编程
- TLS服务端编程
- TLS代理编程



搭建一个TLS服务端（实验任务二.a）

➢ 搭建一个TLS服务端

- 实验步骤

① 为服务端生成证书，放置./server-certs/目录

会用到PKI实验中的内容

② 在server容器中运行server.py

```
seed@VM: ~/.../volumes
root@54ea301ee99d:/volumes# ./server.py
Enter PEM pass phrase:
```

③ 在虚拟机上修改/etc/hosts

④ 利用实验任务一中client与服务端对话

注意：需将根证书导入到./client-certs中

- 要求：记录程序、操作命令和输出结果，写入实验报告，对实验现象进行解释。

```python
#!/usr/bin/env python3

import socket
import ssl
import pprint

html = """
HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n
<!DOCTYPE html><html><body><h1>This is Bank32.com!</h1></body></html>
"""

SERVER_CERT = './server-certs/mycert.crt'
SERVER_PRIVATE = './server-certs/mycert.key'


context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)  # For Ubuntu 20.04 VM
# context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)      # For Ubuntu 16.04 VM
context.load_cert_chain(SERVER_CERT, SERVER_PRIVATE)

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM, 0)
sock.bind(('0.0.0.0', 4433))
sock.listen(5)

while True:
    newsock, fromaddr = sock.accept()
    try:
        ssock = context.wrap_socket(newsock, server_side=True)
        print("TLS connection established")
        data = ssock.recv(1024)              # Read data over TLS
        pprint.pprint("Request: {}".format(data))
        ssock.sendall(html.encode('utf-8'))  # Send data over TLS

        ssock.shutdown(socket.SHUT_RDWR)     # Close the TLS connection
        ssock.close()

    except Exception:
        print("TLS connection fails")
        continue
```

搭建一个TLS服务端（实验任务二. b）

➢ 用浏览器打开测试

- 实验步骤

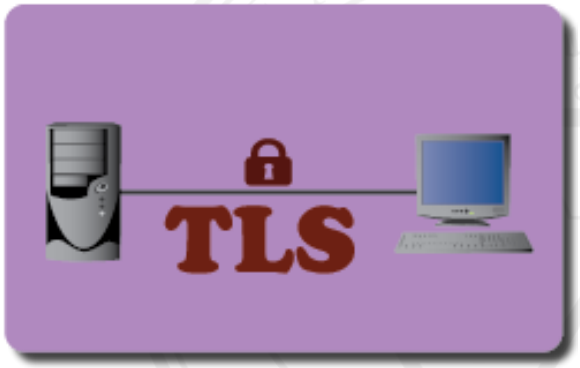
① 在虚拟机上打开浏览器，输入地址（注意端口号是4433）

② 检查出现的问题，尝试进一步修复（PKI实验中的内容）

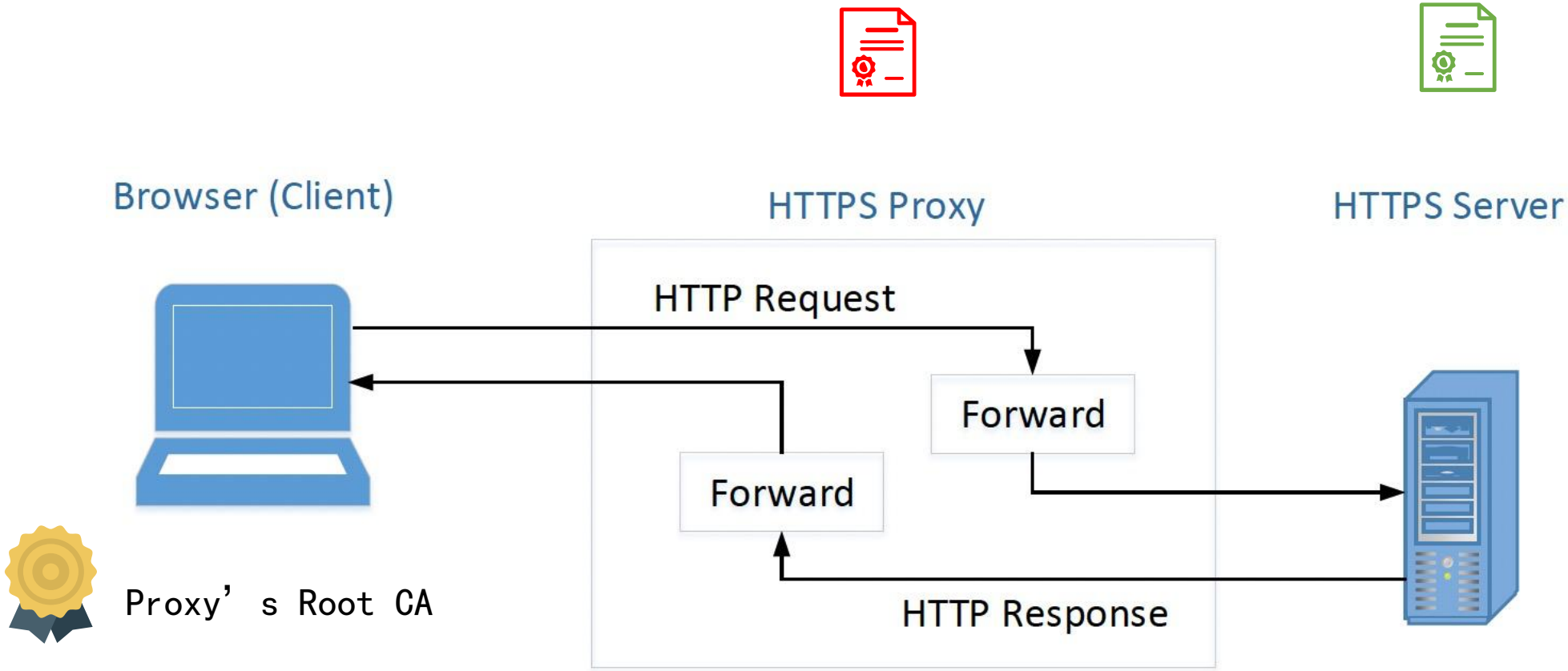
- 要求：记录程序、操作命令和输出结果，写入实验报告，对实验现象进行解释。

课程提纲

- TLS协议概览
- TLS握手
- TLS数据传输
- TLS客户端编程
- TLS服务端编程
- TLS代理编程



TLS代理（MITM代理）



搭建一个TLS代理（实验任务三）

➢ 搭建一个TLS代理

- 实验步骤

① 为目标网站（如www.example.com）颁发一个证书

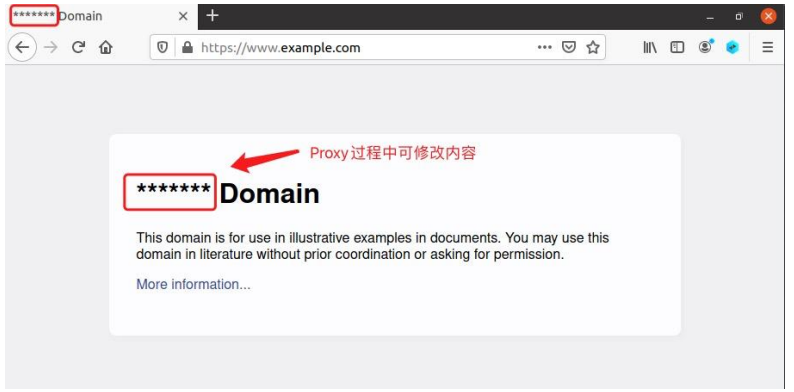
② 修改客户端（可直接用虚拟机）的DNS配置文件/etc/hosts

```
10.9.0.143        www.example.com
```

③ 在mitm-proxy容器内运行TLS代理程序

```
root@2cd59554d398:/volumes# ./proxy.py
Enter PEM pass phrase:
```

④ 通过浏览器访问目标网站（注意：需导入步骤 1中签发者的根证书）



- 要求：记录程序、操作命令和输出结果，写入实验报告，对实验现象进行解释。

小结

- TLS握手（v1.2 v.s. v1.3）
- 密钥协商和服务器认证
- TLS数据传输
- TLS客户端和服务端编程（Python）
- MITM代理

注意：为避免影响后续实验，请清除本实验用过的VM相关配置（恢复/etc/hosts，删除ModelCA根证书）