哈尔滨工业大学(深圳)

# 《网络与系统安全》
# 实验报告

实验五

TLS 实验


学　院: 计算机科学与技术学院

姓　名: 房煊梓

学　号: 210010101

专　业: 智能强基-计算机


日　期: 2024 年 5 月 12 日

1.在客户端容器中执行如下命令 ./handshake.py www.baidu.com 根据执行结果回答下面三个问题。

（1）客户端和服务器端使用的加密算法有哪些，分别起什么作用？

**ECDHE 算法：**用于在客户端和服务器之间安全地生成共享密钥。可实现"前向安全"，即使长期密钥或未来的会话密钥被泄露，过去的通信也不会受到影响。

**RSA 算法：**用于身份验证和密钥交换的最后一步。服务器用私钥对密钥进行签名，客户端用公钥验证签名，以此确认服务器的身份并安全地获得共享密钥。

**AES128 算法：**用于对称加密，客户端和服务器使用相同的密钥来加密和解密数据，加快加密和解密速度。

**GCM 算法：**用于数据加密，确保数据完整性校验和防止篡改。

**SHA256 算法：**作为 GCM 模式的一部分来使用，用于生成认证标签。

（2）分析打印出来的服务器端证书

打印出来的服务器端证书如下：

```
root@6267365ac07f:/volumes# ./handshake.py www.baidu.com
After making TCP connection. Press any key to continue ...
=== Cipher used: ('ECDHE-RSA-AES128-GCM-SHA256', 'TLSv1.2', 128)
=== Server hostname: www.baidu.com
=== Server certificate:
{'OCSP': ('http://ocsp.globalsign.com/gsrsaovsslca2018',),
 'caIssuers': ('http://secure.globalsign.com/cacert/gsrsaovsslca2018.crt',),
 'crlDistributionPoints': ('http://crl.globalsign.com/gsrsaovsslca2018.crl',),
 'issuer': ((('countryName', 'BE'),),
            (('organizationName', 'GlobalSign nv-sa'),),
            (('commonName', 'GlobalSign RSA OV SSL CA 2018'),)),
 'notAfter': 'Aug  6 01:51:05 2024 GMT',
 'notBefore': 'Jul  6 01:51:06 2023 GMT',
 'serialNumber': '55E6ACAED1F8A430F9A938C5',
 'subject': ((('countryName', 'CN'),),
             (('stateOrProvinceName', 'beijing'),),
             (('localityName', 'beijing'),),
             (('organizationName',
               'Beijing Baidu Netcom Science Technology Co., Ltd'),),
             (('commonName', 'baidu.com'),)),
 'subjectAltName': (('DNS', 'baidu.com'),
                    ('DNS', 'baifubao.com'),
                    ('DNS', 'www.baidu.cn'),
                    ('DNS', 'www.baidu.com.cn'),
                    ('DNS', 'mct.y.nuomi.com'),
                    ('DNS', 'apollo.auto'),
                    ('DNS', 'dwz.cn'),
                    ('DNS', '*.baidu.com'),
                    ('DNS', '*.baifubao.com'),
                    ('DNS', '*.baidustatic.com'),
                    ('DNS', '*.bdstatic.com'),
                    ('DNS', '*.bdimg.com'),
                    ('DNS', '*.hao123.com'),
                    ('DNS', '*.nuomi.com'),
                    ('DNS', '*.chuanke.com'),
                    ('DNS', '*.trustgo.com'),
                    ('DNS', '*.bce.baidu.com'),
                    ('DNS', '*.eyun.baidu.com'),
                    ('DNS', '*.map.baidu.com'),
                    ('DNS', '*.mbd.baidu.com'),
                    ('DNS', '*.fanyi.baidu.com'),
                    ('DNS', '*.baidubce.com'),
                    ('DNS', '*.mipcdn.com'),
                    ('DNS', '*.news.baidu.com'),
                    ('DNS', '*.baidupcs.com'),
                    ('DNS', '*.aipage.com'),
                    ('DNS', '*.aipage.cn'),
                    ('DNS', '*.bcehost.com'),
                    ('DNS', '*.safe.baidu.com'),
                    ('DNS', '*.im.baidu.com'),
                    ('DNS', '*.baiducontent.com'),
                    ('DNS', '*.dlnel.com'),
                    ('DNS', '*.dlnel.org'),
                    ('DNS', '*.dueros.baidu.com'),
                    ('DNS', '*.su.baidu.com'),
                    ('DNS', '*.91.com'),
                    ('DNS', '*.hao123.baidu.com'),
                    ('DNS', '*.apollo.auto'),
                    ('DNS', '*.xueshu.baidu.com'),
                    ('DNS', '*.bj.baidubce.com'),
                    ('DNS', '*.gz.baidubce.com'),
                    ('DNS', '*.smartapps.cn'),
                    ('DNS', '*.bdtjrcv.com'),
                    ('DNS', '*.hao222.com'),
                    ('DNS', '*.haokan.com'),
                    ('DNS', '*.pae.baidu.com'),
                    ('DNS', '*.vd.bdstatic.com'),
                    ('DNS', '*.cloud.baidu.com'),
                    ('DNS', 'click.hm.baidu.com'),
                    ('DNS', 'log.hm.baidu.com'),
                    ('DNS', 'cm.pos.baidu.com'),
                    ('DNS', 'wn.pos.baidu.com'),
                    ('DNS', 'update.pan.baidu.com')),
 'version': 3}
[{'issuer': ((('organizationalUnitName', 'GlobalSign Root CA - R3'),),
             (('organizationName', 'GlobalSign'),),
             (('commonName', 'GlobalSign'),)),
  'notAfter': 'Mar 18 10:00:00 2029 GMT',
  'notBefore': 'Mar 18 10:00:00 2009 GMT',
  'serialNumber': '040000000000121585308A2',
  'subject': ((('organizationalUnitName', 'GlobalSign Root CA - R3'),),
              (('organizationName', 'GlobalSign'),),
              (('commonName', 'GlobalSign'),)),
  'version': 3}]
After TLS handshake. Press any key to continue ...
root@6267365ac07f:/volumes#
root@6267365ac07f:/volumes#
```

**分析：**根据以上内容可知，该证书：

**①签发机构：**GlobalSign nv-sa，具体名称为 GlobalSign RSA OV SSL CA 2018

**②有效期：**2023 年 7 月 6 日到 2024 年 8 月 6 日

**③序列号：**55E6ACAED1F8A430F9A938C5

**④证书持有者信息：**国家信息为中国，省份信息为北京，城市信息为北京，组织名为“北京百度网讯科技有限公司”，通用名为“baidu.com”

**⑤证书版本：**3

**⑥根证书：**属于 GlobalSign Root CA - R3

（3）抓包分析 TLS 握手协议

抓包如下图：

| | | | | | | |
|---|---|---|---|---|---|---|
| 297 | 2024-05-12 07:0… | 10.0.2.15 | 183.2.172.185 | TLSv1.2 | 571 | Client Hello |
| 298 | 2024-05-12 07:0… | 183.2.172.185 | 10.0.2.15 | TCP | 60 | 443 → 37518 [ACK] Seq=51072002 Ack=2025185034 Win=65535 Len=0 |
| 299 | 2024-05-12 07:0… | 183.2.172.185 | 10.0.2.15 | TLSv1.2 | 1506 | Server Hello |
| 300 | 2024-05-12 07:0… | 10.0.2.15 | 183.2.172.185 | TCP | 54 | 37518 → 443 [ACK] Seq=2025185034 Ack=51073454 Win=63888 Len=0 |
| 301 | 2024-05-12 07:0… | 183.2.172.185 | 10.0.2.15 | TLSv1.2 | 3786 | Certificate, Server Key Exchange, Server Hello Done |
| 302 | 2024-05-12 07:0… | 10.0.2.15 | 183.2.172.185 | TCP | 54 | 37518 → 443 [ACK] Seq=2025185034 Ack=51077186 Win=61320 Len=0 |
| 303 | 2024-05-12 07:0… | 10.0.2.15 | 183.2.172.185 | TLSv1.2 | 180 | Client Key Exchange, Change Cipher Spec, Encrypted Handshake … |
| 304 | 2024-05-12 07:0… | 183.2.172.185 | 10.0.2.15 | TCP | 60 | 443 → 37518 [ACK] Seq=51077186 Ack=2025185160 Win=65535 Len=0 |
| 305 | 2024-05-12 07:0… | 183.2.172.185 | 10.0.2.15 | TLSv1.2 | 280 | New Session Ticket, Change Cipher Spec, Encrypted Handshake M… |
| 306 | 2024-05-12 07:0… | 10.0.2.15 | 183.2.172.185 | TCP | 54 | 37518 → 443 [ACK] Seq=2025185160 Ack=51077412 Win=62780 Len=0 |
| 307 | 2024-05-12 07:0… | 10.0.2.15 | 183.2.172.185 | TCP | 54 | 37518 → 443 [FIN, ACK] Seq=2025185160 Ack=51077412 Win=62780 … |
| 308 | 2024-05-12 07:0… | 183.2.172.185 | 10.0.2.15 | TCP | 60 | 443 → 37518 [ACK] Seq=51077412 Ack=2025185161 Win=65535 Len=0 |
| 309 | 2024-05-12 07:0… | 183.2.172.185 | 10.0.2.15 | TLSv1.2 | 85 | Encrypted Alert |

**分析：**根据 wireshark 工具信息可知：

No.297 时，客户端发送 Client Hello。

No.299 时，服务器发送 Server Hello。

No.301 时，服务器发送服务器公钥证书 Certificate，服务器密钥交换信息 Server Key Exchange 和服务器问候完成信息 Server Hello Done。

No.303 时，客户端发送客户端密钥交换信息 Client Key Exchange，更改密码规范信息 Change Ciper Spec 和一个加密完成的消息 Encrypted Handshake Message。

No.305 时，服务器发送 New Session Ticket，更改密码规范信息 Change Ciper Spec 和一个加密完成的消息 Encrypted Handshake Message。

双方互相发送一个加密完成的消息之后，握手协议成功。

2.更改证书文件路径，请同学们将 www.baidu.com 网站的测试过程截图保存（如果不将证书拷贝过来应该有报错信息，拷贝过来之后应该正常），也可选用其他网站做测试。

在将证书 copy 到./client-cert 之前，将代码中的证书文件路径改成./client-certs，再次在客户端容器运行 ./handshake.py，结果如下两张图：

```
root@98f78c9c60dc:/volumes# ./handshake.py www.baidu.com
After making TCP connection. Press any key to continue ...
Traceback (most recent call last):
  File "./handshake.py", line 29, in <module>
    ssock.do_handshake()   # Start the handshake
  File "/usr/lib/python3.8/ssl.py", line 1309, in do_handshake
    self._sslobj.do_handshake()
ssl.SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verif
y failed: unable to get local issuer certificate (_ssl.c:1123)
```

| | | | | | | |
|---|---|---|---|---|---|---|
| 41 | 2024-05-12 07:1… | 10.0.2.15 | 183.2.172.185 | TCP | 74 | 37642 → 443 [SYN] Seq=3955560804 Win=64240 Len=0 MSS=1460 SAC… |
| 42 | 2024-05-12 07:1… | 183.2.172.185 | 10.0.2.15 | TCP | 60 | 443 → 37642 [SYN, ACK] Seq=106496001 Ack=3955560805 Win=65535… |
| 43 | 2024-05-12 07:1… | 10.0.2.15 | 183.2.172.185 | TCP | 54 | 37642 → 443 [ACK] Seq=3955560805 Ack=106496002 Win=64240 Len=0 |
| 44 | 2024-05-12 07:1… | 10.0.2.15 | 183.2.172.185 | TLSv1.2 | 571 | Client Hello |
| 45 | 2024-05-12 07:1… | 183.2.172.185 | 10.0.2.15 | TCP | 60 | 443 → 37642 [ACK] Seq=106496002 Ack=3955561322 Win=65535 Len=0 |
| 46 | 2024-05-12 07:1… | 183.2.172.185 | 10.0.2.15 | TLSv1.2 | 1506 | Server Hello |
| 47 | 2024-05-12 07:1… | 10.0.2.15 | 183.2.172.185 | TCP | 54 | 37642 → 443 [ACK] Seq=3955561322 Ack=106497454 Win=63888 Len=0 |
| 48 | 2024-05-12 07:1… | 183.2.172.185 | 10.0.2.15 | TLSv1.2 | 3786 | Certificate, Server Key Exchange, Server Hello Done |
| 49 | 2024-05-12 07:1… | 10.0.2.15 | 183.2.172.185 | TCP | 54 | 37642 → 443 [ACK] Seq=3955561322 Ack=106501186 Win=61320 Len=0 |
| 50 | 2024-05-12 07:1… | 10.0.2.15 | 183.2.172.185 | TLSv1.2 | 61 | Alert (Level: Fatal, Description: Unknown CA) |
| 51 | 2024-05-12 07:1… | 183.2.172.185 | 10.0.2.15 | TCP | 60 | 443 → 37642 [ACK] Seq=106501186 Ack=3955561329 Win=65535 Len=0 |

容器报错 CERTIFICATE_VERIFY_FAILED，且其细节内容为 unable to get local issuer certificate；同时 wireshark 也出现了 Alert，其中的 Description 为 Unknown CA，最终没有成功完成 TLS 握手。这是因为证书文件的路径里并没有相应的证书文件。

拷贝证书文件后，生成 hash 并做个软链接，然后再次运行./handshake.py www.baidu.com，发现可以正常运行。(由于打印的证书内容相同，这里不再截图，只展示 wireshark 截图)

| | | | | | | |
|---|---|---|---|---|---|---|
| 345 | 2024-05-12 07:1… | 10.0.2.15 | 183.2.172.42 | TCP | 74 | 33300 → 443 [SYN] Seq=3854666649 Win=64240 Len=0 MSS=1460 SAC… |
| 346 | 2024-05-12 07:1… | 183.2.172.42 | 10.0.2.15 | TCP | 60 | 443 → 33300 [SYN, ACK] Seq=119936001 Ack=3854666650 Win=65535… |
| 347 | 2024-05-12 07:1… | 10.0.2.15 | 183.2.172.42 | TCP | 54 | 33300 → 443 [ACK] Seq=3854666650 Ack=119936002 Win=64240 Len=0 |
| 348 | 2024-05-12 07:1… | 10.0.2.15 | 183.2.172.42 | TLSv1.2 | 571 | Client Hello |
| 349 | 2024-05-12 07:1… | 183.2.172.42 | 10.0.2.15 | TCP | 60 | 443 → 33300 [ACK] Seq=119936002 Ack=3854667167 Win=65535 Len=0 |
| 350 | 2024-05-12 07:1… | 183.2.172.42 | 10.0.2.15 | TLSv1.2 | 2974 | Server Hello |
| 351 | 2024-05-12 07:1… | 10.0.2.15 | 183.2.172.42 | TCP | 54 | 33300 → 443 [ACK] Seq=3854667167 Ack=119938922 Win=62780 Len=0 |
| 352 | 2024-05-12 07:1… | 183.2.172.42 | 10.0.2.15 | TLSv1.2 | 2318 | Certificate, Server Key Exchange, Server Hello Done |
| 353 | 2024-05-12 07:1… | 10.0.2.15 | 183.2.172.42 | TCP | 54 | 33300 → 443 [ACK] Seq=3854667167 Ack=119941186 Win=62780 Len=0 |
| 354 | 2024-05-12 07:1… | 10.0.2.15 | 183.2.172.42 | TLSv1.2 | 180 | Client Key Exchange, Change Cipher Spec, Encrypted Handshake … |
| 355 | 2024-05-12 07:1… | 183.2.172.42 | 10.0.2.15 | TCP | 60 | 443 → 33300 [ACK] Seq=119941186 Ack=3854667293 Win=65535 Len=0 |
| 356 | 2024-05-12 07:1… | 183.2.172.42 | 10.0.2.15 | TLSv1.2 | 280 | New Session Ticket, Change Cipher Spec, Encrypted Handshake M… |
| 357 | 2024-05-12 07:1… | 10.0.2.15 | 183.2.172.42 | TCP | 54 | 33300 → 443 [ACK] Seq=3854667293 Ack=119941412 Win=62780 Len=0 |
| 358 | 2024-05-12 07:1… | 10.0.2.15 | 183.2.172.42 | TCP | 54 | 33300 → 443 [FIN, ACK] Seq=3854667293 Ack=119941412 Win=62780… |
| 359 | 2024-05-12 07:1… | 183.2.172.42 | 10.0.2.15 | TCP | 60 | 443 → 33300 [ACK] Seq=119941412 Ack=3854667294 Win=65535 Len=0 |
| 360 | 2024-05-12 07:1… | 183.2.172.42 | 10.0.2.15 | TLSv1.2 | 85 | Encrypted Alert |

3. 请同学们将修改 www.baidu.com 网站主机名的测试过程截图保存在报告里并分析执行的结果，也可选用其他网站做测试。

首先使用 dig 命令查看 www.baidu.com 的服务器 IP 地址：

```
[05/09/24]seed@VM:~$ dig www.baidu.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.baidu.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26712
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;www.baidu.com.                 IN      A

;; ANSWER SECTION:
www.baidu.com.          111     IN      CNAME   www.a.shifen.com.
www.a.shifen.com.       110     IN      A       183.2.172.42
www.a.shifen.com.       110     IN      A       183.2.172.185

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Thu May 09 04:49:57 EDT 2024
;; MSG SIZE  rcvd: 101
```

发现 183.2.172.42 和 183.2.172.185 均可用，这里选用前者。

用 echo 命令将假的主机名 www.baidu1.com 写入到客户端的/etc/hosts 中，将代码中的主机名检测设置为 True，执行./handshake.py www.baidu1.com 命令，结果如下两张图：

```
root@6267365ac07f:/volumes# ./handshake.py www.baidu1.com
After making TCP connection. Press any key to continue ...
Traceback (most recent call last):
  File "./handshake.py", line 29, in <module>
    ssock.do_handshake()   # Start the handshake
  File "/usr/lib/python3.8/ssl.py", line 1309, in do_handshake
    self._sslobj.do_handshake()
ssl.SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: Hostname mismatch, certificate is not valid for 'ww
w.baidu1.com'. (_ssl.c:1123)
```

```
54 59516 → 443 [ACK] Seq=518 Ack=5185 Win=262656 Len=0
61 Alert (Level: Fatal, Description: Bad Certificate)
```

**分析：**容器报错 CERTIFICATE_VERIFY_FAILED，细节内容为 Hostname mismatch；同时 wireshark 也出现了 Alert，其中的 Description 为 Bad Certificate，最终没有成功完成 TLS 握手。这是因为主机名检测设置为 True，TLS 会校验服务器的主机名，发现主机名不匹配时进行报错。

将代码中的主机名检测设置为 False，执行./handshake.py www.baidu1.com 命令，结果如下图。(由于打印证书的内容基本一致，这里只截取部分)

```
root@6267365ac07f:/volumes# ./handshake.py www.baidu1.com
After making TCP connection. Press any key to continue ...
=== Cipher used: ('ECDHE-RSA-AES128-GCM-SHA256', 'TLSv1.2', 128)
=== Server hostname: www.baidu1.com
=== Server certificate:
{'OCSP': ('http://ocsp.globalsign.com/gsrsaovsslca2018',),
 'caIssuers': ('http://secure.globalsign.com/cacert/gsrsaovsslca2018.crt',),
 'crlDistributionPoints': ('http://crl.globalsign.com/gsrsaovsslca2018.crl',),
 'issuer': ((('countryName', 'BE'),),
            (('organizationName', 'GlobalSign nv-sa'),),
            (('commonName', 'GlobalSign RSA OV SSL CA 2018'),)),
 'notAfter': 'Aug  6 01:51:05 2024 GMT',
 'notBefore': 'Jul  6 01:51:06 2023 GMT',
 'serialNumber': '55E6ACAED1F8A430F9A938C5',
 'subject': ((('countryName', 'CN'),),
             (('stateOrProvinceName', 'beijing'),),
             (('localityName', 'beijing'),),
             (('organizationName',
               'Beijing Baidu Netcom Science Technology Co., Ltd'),),
             (('commonName', 'baidu.com'),)),
 'subjectAltName': (('DNS', 'baidu.com'),
                    ('DNS', 'baifubao.com'),
                    ('DNS', 'www.baidu.cn'),
                    ('DNS', 'www.baidu.com.cn'),
```

**分析：**容器能够正常打印证书内容，且 Server Hostname 为 www.baidu1.com，说明主机名检测设置为 False 时 TLS 不校验服务器的主机名，即使主机名不匹配也可以正常运行。

4.请分析 TLS 客户端编程和 server.py 的代码，说明客户端和服务器程序的关键步骤。

(1)**TLS 客户端编程**

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

首先从命令行参数中读取目标主机名或 IP 地址赋给 hostname，再设置默认端口为 HTTPS 的标准端口 443，然后设置证书路径。

```
# 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
```

创建一个 SSL 上下文并配置 SSL 上下文以验证服务器证书，包括指定证书存储路径、要求证书验证以及开启主机名检查。

```
# 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 ...")
```

创建 TCP 套接字 sock，连接到指定的主机和端口，等待用户输入继续。

```
# 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 ...")
```

将 sock 包装为 ssock，并指定服务器主机名。然后通过 do_handshake()开始握手过程。然后打印使用的加密套件、服务器主机名及服务器证书，等待用户输入继续。

```
# 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)
```

构造并发送 HTTP 请求给服务器，从服务器接收并分段打印 HTTP 响应。

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

最后，关闭 TLS 连接。

(2) **server.py**

```
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/server.crt'
SERVER_PRIVATE = './server-certs/server.key'
```

定义 html，内容为 www.bank32.com 的 HTTP 响应，并且指定服务器证书和私钥的路径。

```
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)
```

创建一个 SSL 上下文，并且加载服务器证书和私钥到 SSL 上下文中。

```
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM, 0)
sock.bind(('10.9.0.43', 443))
sock.listen(5)
```

创建一个 TCP 套接字，再绑定服务器套接字到 IP 地址 10.9.0.43 和端口 443，然后开始监听连接。

```
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
```

while 循环处理客户端连接：

通过 accept()接受一个客户端的连接请求，获得新的套接字和客户端地址。

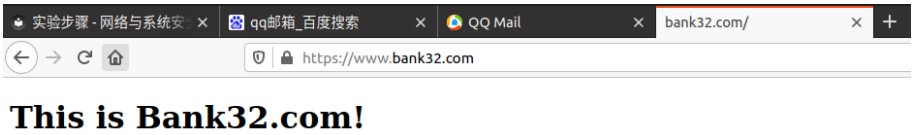
然后使用 try...except Exception：首先包装 newsock 为 ssock，并指定为服务器端并打印连接建立成功的提示；然后通过 ssock 接收客户端发送的数据并格式化打印接收到的请求数据；再将 html 的内容编码后通过 ssock 发送给客户端；最后关闭 ssock 的读写操作，关闭 ssock。如果在处理过程中发生异常，则打印连接建立失败的提示并继续下一个循环。

5.请分别用 client.py 和浏览器两种方式访问服务器，并记录你观察的结果（截图）

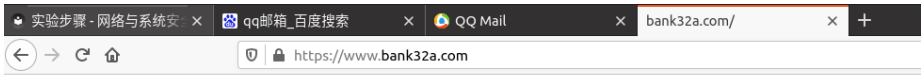
用 client.py 的方式访问服务器截图如下。

```
root@126b271d70b0:/volumes# ./client.py www.bank32.com
After making TCP connection. Press any key to continue ...
=== Cipher used: ('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
=== Server hostname: www.bank32.com
=== Server certificate:
{'issuer': ((('commonName', 'www.modelCA.com'),),
            (('organizationName', 'Model CA LTD.'),),
            (('countryName', 'US'),)),
 'notAfter': 'May  7 13:21:46 2034 GMT',
 'notBefore': 'May  9 13:21:46 2024 GMT',
 'serialNumber': '1000',
 'subject': ((('countryName', 'US'),),
             (('organizationName', 'Bank32 Inc.'),),
             (('commonName', 'www.bank32.com'),)),
 'subjectAltName': (('DNS', 'www.bank32.com'),
                    ('DNS', 'www.bank32A.com'),
                    ('DNS', 'www.bank32B.com')),
 'version': 3}
[{'issuer': ((('commonName', 'www.modelCA.com'),),
             (('organizationName', 'Model CA LTD.'),),
             (('countryName', 'US'),)),
  'notAfter': 'May  7 13:21:06 2034 GMT',
  'notBefore': 'May  9 13:21:06 2024 GMT',
  'serialNumber': '4A6E801155E40654DE16C0F478C64878A7A3F991',
  'subject': ((('commonName', 'www.modelCA.com'),),
              (('organizationName', 'Model CA LTD.'),),
              (('countryName', 'US'),)),
  'version': 3}]
After TLS handshake. Press any key to continue ...
[b'\nHTTP/1.1 200 OK',
 b'Content-Type: text/html',
 b'',
 b'\n<!DOCTYPE html><html><body><h1>This is Bank32.com!</h1></body></html>\n']
root@126b271d70b0:/volumes#
```

用浏览器的方式访问服务器截图如下。第一张是访问 www.bank32.com 的截图，第二张是访问别名 www.bank32a.com 的截图。




**This is Bank32.com!**




**This is Bank32.com!**