密码学实验综合实验报告——基于 SM2、SM3、SM4 的安全文件传输协议

张天辰 17377321

2019 年 6 月 29 日

1 协议设计

1.1 需求分析及框架

本方案考虑双方传输文件的场景。在该场景下，双方的文件传输要求保密性、完整性与认证性。考虑到文件可能较大，为了兼顾效率，保密性可以用 SM4 算法提供；完整性和认证性可以由 SM2 数字签名提供。

考虑到并没有实质上可信的第三方来颁发数字证书，本算法并没有设置数字证书，发送方与接收方的公钥是公开发送的。考虑到有数字签名的存在，安全性能够得到保证。本方案使用带数字签名的 Diffie-Hellman 密钥交换协议交换对称加密密钥，保证密钥的安全。

本方案采用密文分组链接模式进行对称加密，则一共需要保密的内容有初始向量、文件名长度、文件名、明文消息及签名。初始向量可以用电码本模式加密，处于发送数据内容的最开始部分即可。

1.2 详细流程

1. 双方首先建立连接，然后双方交换 ID，再交换公钥。这一步类似于数字证书，只不过是没有可信第三方的签名。

2. 在交换这些信息之后，双方就可以开始协商密钥。通过上一步的信息，可以验证数字签名，从而完成密钥的协商。

3. 发送方发送文件长度和给接收方，这样接收方就可以判断接收多少长度的字节。此后，发送方产生初始向量，将这个值用上一步协商的密钥加密；将原始文件进行签名，拼接在文件后面，再使用对称密钥加密。将上述两个部分组合在一起，发送给接收方。

4. 接收方得到发送的字节串，首先提取解密出初始向量，再用初始向量解密明文与签名，最后验证。只要验证通过，传输就结束了。

2 协议实现

本方案基于 Python 的 socket 套接字实现。

1. 交换 ID 与公钥

```python
# sender
socket.listen(5)
sock, addr = socket.accept()
socket.send(ID)
targetID = socket.recv(10)
# signature init
auth = SM2Sign.SM2Sign(targetID)
socket.send(public)
auth.public = sock.recv(65)
```

```python
# receiver
socket.connect((ip, port))
senderID = socket.recv(10)
# signature init
auth = SM2Sign.SM2Sign(senderID)
socket.send(ID)
auth.public = socket.recv(65))
socket.send(public)
```

2. 协商密钥

发送方生成随机数 k_1，然后发送 $k_1 \cdot G$ 以及对其的签名；接收方同理生成 k_2，发送 $k_2 \cdot G$ 以及对其的签名。双方任意一方签名验证失败都会导致整个连接终止。如果双方都验签成功，则 $K = k_1 k_2 \cdot G$。协商成功。

```python
# sender
sign = SM2Sign.SM2Sign(ID)
sign.public = public
sign.private = private
k = randint(3, ECCPoint.n - 1)
pt = point_to_bytes(SM2Sign.
    SM2Sign.g.multi(k))
sig = sign.sign(pt)
socket.send(pt || sig[0] || sig
    [1])
other = socket.recv(130)
otherPt = bytes_to_point(other
    [:65])
otherSig = other[65:]
if not auth.authenticate(other
    [:65], (otherSig[:32],
    otherSig[32:])):
    print("Authentication␣failed␣
        while␣key␣exchanging.")
    socket.close()
    sys.exit(0)
key = point_to_bytes(otherPt.
    multi(k))[1:]
```

```python
# receiver
other = socket.recv(130)
k = randint(3, ECCPoint.n - 1)
otherPt = bytes_to_point(other
    [:65])
otherSig = other[65:]
if not auth.authenticate(other
    [:65], (otherSig[:32],
    otherSig[32:])):
    print("Authentication␣failed␣
        while␣key␣exchanging.")
    socket.close()
    sys.exit(0)
key = point_to_bytes(otherPt.
    multi(k))[1:]
sign = SM2Sign.SM2Sign(ID)
sign.public = public
sign.private = private
pt = point_to_bytes(SM2Sign.
    SM2Sign.g.multi(k))
sig = sign.sign(pt)
socket.send(pt || sig[0] || sig
    [1])
```

3. 消息发送

在发送正式消息前，要先发送加密后消息长度。此后需要发送的消息格式如下：

E_{ecb}(初始向量)||E_{cbc}(文件名长度 || 文件名 || 文件内容 || 签名)

```
# sender
# signature init
sign = SM2Sign.SM2Sign(ID)
sign.public = public
sign.private = private
signature = sign.sign(src)

encrypt = sm4.CryptSM4()
encrypt.set_key(key, ENCRYPT)
encryptIV = encrypt.crypt_ecb(iv)
cipher = encrypt.crypt_cbc(
    iv, bytes([len(name)]) || name || src || signature)
e = encryptIV || cipher
# send the length first
socket.send((len(e)).to_bytes())
# send the rest
for i in range(0, len(e), 1024):
    socket.send(e[i:i+1024])
```

4. 接收文件

```
# receiver
length = socket.recv(3).to_int()
l = 0
target = b''
while l < length:
    t = self.sock.recv(1024)
    target = b''.join([target, t])
    l += len(t)
decrypt = sm4.CryptSM4()
decrypt.set_key(self.key, DECRYPT)
# the first 32 bytes are the encrypted iv
iv = decrypt.crypt_ecb(target[:32])
target = decrypt.crypt_cbc(self.iv, target[32:])
nameLen = target[0]
name = target[1:1+nameLen]
result = auth.authenticate(target[1+nameLen:-64], (target[-64:-32], target
    [-32:]))
if result:
    with open(name.decode(), "wb") as f:
```

```
19            f.write(target[1+nameLen:-64])
20        print("File␣received␣from␣" + sid)
21    else:
22        print("Authentication␣failed␣while␣receiving.")
23        sys.exit(0)
```

3 协议测试

在开始状态时，接收方文件夹只有代码文件，发送方文件夹除了代码文件外还有用于发送的文件（这个文件的位置可以是任意的），如图 1。

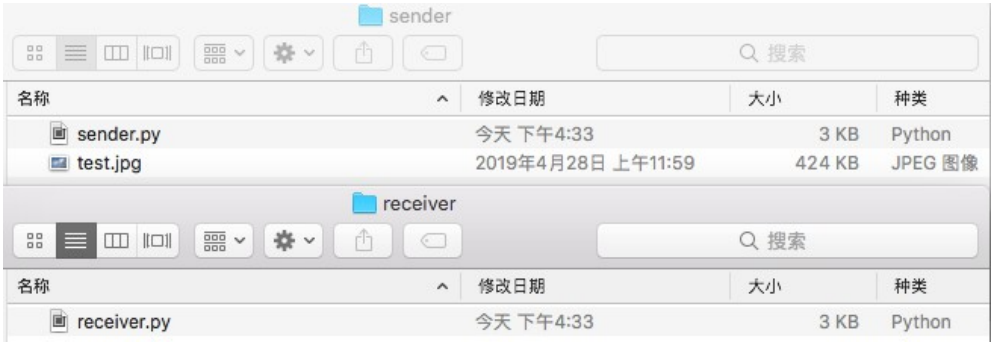

图 1: 初始状态

发送文件时的状态如图 2。

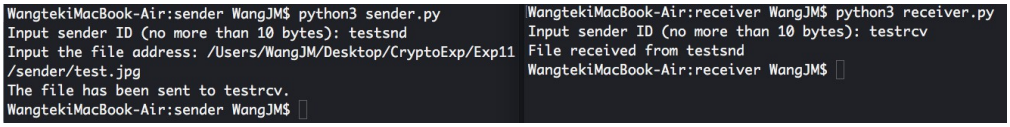

图 2: 发送文件

发送结束且接收结束后，接收方文件夹就出现了新的文件，其与发送方文件完全相同，如图 3。

接收方收到了 jpg 文件可以打开，说明解密完全正确，且数字签名验证成功。综上所述，根据测试结果，本方案的协议可以很好地工作。

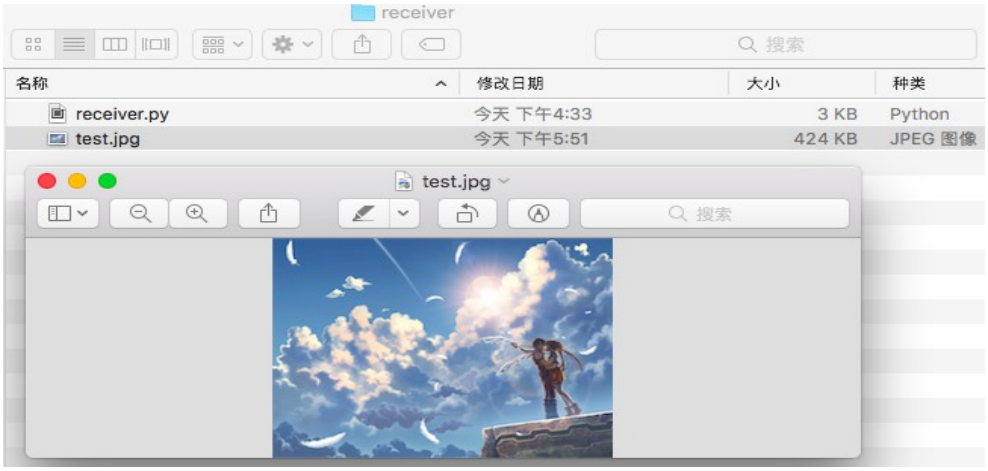
receiver
搜索
名称
修改日期
大小
种类
receiver.py
今天 下午4:33
3 KB
Python
test.jpg
今天 下午5:51
424 KB
JPEG 图像
test.jpg
搜索

图 3: 结束状态