

实验四：基于口令的认证过程实现

姓名	石开宇
学号	1162910310
班级	1603201

实验要求

1. 客户端输入用户名，口令，随机产生认证码，使用散列函数计算用户名与口令的散列值1，使用散列值1与认证码计算散列值2，将用户名，散列值2，认证码明文传送到服务器端。
 - $H_1 = \text{Hash}(\text{username} + \text{password})$
 - $H_2 = \text{Hash}(H_1 + \text{rand}_1)$
 - $\text{send}(\text{username}, H_2, \text{rand}_1)$
2. 服务器端以数据库（如access）保存用户名和散列值1的对应关系。收到客户端信息后，以同样的方法计算散列值2'。如散列值2'=散列值2，则认证成功，成功后用散列值1加密认证码发送给客户端。客户端解密后写到指定文件。
 - $H_1' = \text{getH1}(\text{username})$
 - $H_2' = \text{Hash}(H_1' + \text{rand}_1)$
 - $H_2' == H_2 ? \text{send}(\text{Encode}(H_1, \text{rand}_2))$
3. 实验目的，掌握随机函数的使用，掌握散列函数，加解密函数的使用。包的格式与发包的顺序，可以等同于协议的三要素。掌握程序与数据库的连接。

实验内容

- 服务端: 采用node+express+MongoDB作为服务端，密码学工具包：Crypto-JS
 - MongoDB中存放username，hash1
 - 地址 `/post` 用于与客户端收发消息
 - 散列采用SHA256方式
 - 加密采用AES CBC模式，padding方式为Pkcs7，密钥长度256位（即直接使用sha256生成的256bit散列值）
 - iv向量程序随机生成，并发给客户端
- 客户端: 利用python客户端与上面服务器通信：发送接收消息，密码学工具包：PyCrypto
 - 用户输入自己的用户名username，密码passcode，随机或者自行设置认证码rand1
 - 散列函数与加密函数同上，在AES解密时同样要采用CBC模式，iv使用传来的随机iv
 - 简易步骤
 - $H_1 = \text{Hash}(\text{username} + \text{password})$ 利用sha256计算用户名与密码的哈希：hash1
 - $H_2 = \text{Hash}(H_1 + \text{rand}_1)$ 利用sha256计算上面计算的hash1与认证码rand1的哈希：hash2

- send(username, H_2, rand_1) 将username, hash2, rand1发送给服务端
- 成功时，接收到用AES加密的认证码，加密时使用的iv，解密，验证认证码是否正确。

实验结果

- 首先在MongoDB中保存用户名与散列值1

local mongodb (7)

- System
- ToDoApp
- ToDoAppTest
- config
- lab04
 - Collections (2)
 - lab
 - users
 - Functions
 - Users
 - test

db.getCollection('users')...x

local mongodb localhost:27017 lab04

db.getCollection('users').find({})

users 0.001 sec.

Key	Value	Type
(1) Objectid("5cb03422fd91ba750c53f28c")	{ 3 fields }	Object
_id	Objectid("5cb03422fd91ba750c53f28c")	Objectid
username	sky	String
hash1	2165f14179c941df5c317bbcc9a17f6206f0bedb36b7ce06...	String

- 这里用户名是sky，密码是pass01，并不保存密码，直接保存用户名和密码的hash

- 服务端开始监听

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1:

~ / Desktop / REVIEW / network security / lab04 simple-lab • node server / src / app.js

listening at port 3000

- 默认监听地址为127.0.0.1:3000，与客户端通信地址为127.0.0.1:3000/data

- 客户端运行

Auth

username sky

passcode pass01

rand1

Login

decrypted f58ea34849b979d923/dc21ff7732395

~ / Desktop / REVIEW / network security / lab04 simple-lab • python3 client / auth client.py

- 这里rand1为空时会随机生成认证码，也可以自己指定
- 随机生成认证码方式，登录

◦

Auth

username	sky
passcode	pass01
rand1	

```
{'status': 'success', 'rand2_encrypted': '17063429d01410ae0fed5d501358728f872f9d56926a6b4cfd6147f9c9a28b4f0318ed70a753826e916be3338feb6bd6', 'iv': '6d8bb13aa0f0b6a053e145c34cd1ea0c'}  
encrypted: 17063429d01410ae0fed5d501358728f872f9d56926a6b4cfd6147f9c9a28b4f0318ed70a753826e916be3338feb6bd6  
key: 2165f14179c941df5c317bbcc9a17f6206f0bedb36b7ce06e9a4cdfc17430a67  
decrypted: b275a610c20091751ca9921140389482
```

Login

- 这里输出展示4行数据信息
 - 第一行是服务端在收到客户发送请求后返回的消息，status表示认证是否成功，rand2_encrypted表示利用hash1加密的认证码，iv是随机生成的，后面客户端解密将会用到
 - 第二行就是加密消息，跟上面的rand2_encrypted相同
 - 第三行是对称加密使用的密钥，也就是256bit的hash1，这里都是采用16进制展示的
 - 第四行表示解密后的认证码，也就是rand1
- 使用错误密码登录

Auth

username	sky
passcode	pass02
rand1	

```
{'status': 'failed', 'error': 'authentication failed'}  
authentication failed!
```

Login

- 服务端返回status表示认证失败
- 使用指定认证码测试

Auth	
username	sky
passcode	pass01
rand1	123

```
{'status': 'success', 'rand2_encrypted': 'bb036cbb545a1b3ba2263e7aa80613a7', 'iv': '36df126f54b906fa37d9f5c29925717d'}
```

encrypted: bb036cbb545a1b3ba2263e7aa80613a7
key: 2165f14179c941df5c317bbcc9a17f6206f0bedb36b7ce06e9a4cdfc17430a67
decrypted: 123

- 这里使用指定认证码123，可以看到，成功解密出123，认证成功
- 成功解密的认证码也会写到当前目录的test.txt文件

test.txt - Typora

File Edit Paragraph Format View Themes Help

123

- 同样，服务端也可以看到认证过程

```
authentication failed!
~/Desktop/REVIEW/network security/lab04 [simple-lab] [python3 client/auth_client.py
hash1 c0efd4f76eae2c02191ba98499fac99003953864e4620fc93d5e4410d4795ef3
{'username': 'sky', 'hash2': '5ef0b1a7cfeeb20b25dfbdbb712f09793b4810679fac8974a46d760c0d45ba3', 'rand1': 'cb8d9eb9de0a97
4156b735ca42ad0ca7'}
{'status': 'failed', 'error': 'authentication failed'}
authentication failed!
hash1 c0efd4f76eae2c02191ba98499fac99003953864e4620fc93d5e4410d4795ef3
{'username': 'sky', 'hash2': 'a3f6200ff963b3118dfdcf00d2eb51e18890829806985ff50d5103183cdbc526', 'rand1': '123'}
{'status': 'failed', 'error': 'authentication failed'}
authentication failed!
hash1 2165f14179c941df5c317bbcc9a17f6206f0bedb36b7ce06e9a4cdfc17430a67
{'username': 'sky', 'hash2': '016c32e36080b0c50966f483445bebe0f01bed51fa3aa07c4e1333992548e5d2', 'rand1': '123'}
{'status': 'success', 'rand2 encrypted': 'bb036cbb545a1b3ba2263e7aa80613a7', 'iv': '36df126f54b906fa37d9f5c29925717d'}
-----server responds success-----
encrypted: bb036cbb545a1b3ba2263e7aa80613a7
key 2165f14179c941df5c317bbcc9a17f6206f0bedb36b7ce06e9a4cdfc17430a67
decrypted 123
```

困难解决与心得体会

AES加密

- AES加密，因为使用了两种语言，nodejs和python，密码学函数库要表现相同需要多设置一些参数，主要是
 - AES加密模式要统一，实验中选用了CBC模式
 - AES加密模式如果需要padding的话，padding方式也要统一，实验中采用了pkcs7方式，python中没有找到现成的，网上查找资料后，简单实现了pad和unpad函数

```
#pkcs7
BS = AES.block_size
pad = lambda s: s + (BS - len(s) % BS) * chr(BS - len(s) % BS)
unpad = lambda s: s[0:-ord(s[-1])]
```

- AES加密模式如果需要初始向量iv的话，加解密需要统一，实验中采用了密码学随机函数随机生成iv，并且在服务端发回密文的时候，iv也要发回。

代码

参见文件

```
server/: node服务器端，默认地址127.0.0.1:3000
client/: python客户端
```