# Exp4-2 基于口令的身份验证协议

## 环境设置

本实验的环境设置为macOS 13.6.7, python 3.10.5, pycryptodome 3.20.0

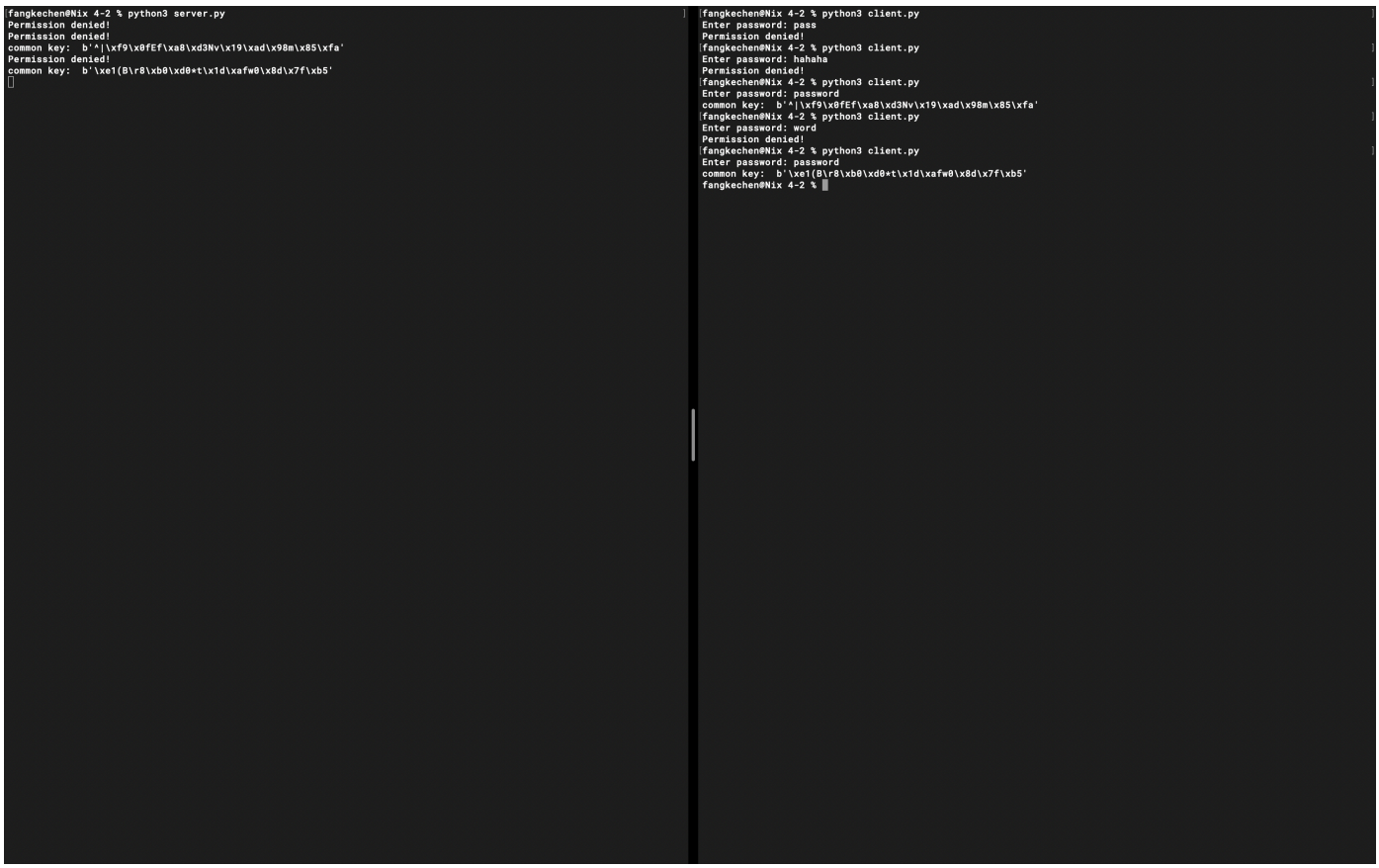
## 关键步骤

代码分为三部分，`server.py, client.py, utils.py`。其中`utils.py`中放了若干加密解密等操作的函数以便服务端和客户端调用。服务端`client.py`中存有明文密码`password`且将其使用SHA256进行Hash后取前 $16$ 字节。使用SHA256进行加密的原因是，若直接将密码用于AES加解密会有长度要求错误，这会使破解者轻易获取密码长度，因此使用Hash将任意长度密码映射到固定长度的口令。在客户端中会要求用户输入密码并使用同样的操作得到 $16$ 字节密码，这即为事先共享的口令 $pw$。之后便根据教材上的步骤 $1-6$，在两端进行相应的加解密、收发操作。当客户端通过验证，两方会输出最终采用的 $K_s$ 密钥。

## 影响因素分析

如果在两台机器上测试，可能需要考虑网络延迟、丢包等因素。

## 实验结果




## 关键源代码

其中`client.py`的关键源代码为：

```
if __name__ == "__main__":
    password = input("Enter password: ").encode()
    password = aes_transform_key(password)

    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect(("localhost", 1234))

    # Stage 2
    try:
        public_key = aes_decode(password, sock.recv(1024))
    except:
        print("Permission denied!")
        sock.close()
        sys.exit(0)
    K_s = aes_keygen()
    sock.send(aes_encode(password, rsa_encode(public_key, K_s)))

    # Stage 4
    try:
        N_A = aes_decode(K_s, sock.recv(1024))
    except:
        print("Permission denied!")
        sock.close()
        sys.exit(0)
    N_B = aes_keygen()
    sock.send(aes_encode(K_s, N_A + N_B))

    # Stage 6
    try:
        N2 = aes_decode(K_s, sock.recv(1024))
    except:
        print("Permission denied!")
        sock.close()
        sys.exit(0)
    if N2 != N_B:
        print("Permission denied!")
        sock.close()
        sys.exit(0)
    print("common key: ", K_s)
```

`server.py` 的关键源代码为：

```
password = b'password'
password = aes_transform_key(password)

if __name__ == "__main__":
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.bind(("localhost", 1234))
    while (True):
```

```
        sock.listen(1)
        conn, addr = sock.accept()

        # Stage 1
        public_key, private_key = rsa_keygen()
        conn.send(aes_encode(password, public_key))

        # Stage 3
        try:
            K_s = rsa_decode(private_key, aes_decode(password, conn.recv(1024)))
        except:
            print("Permission denied!")
            conn.close()
            continue
        N_A = aes_keygen()
        conn.send(aes_encode(K_s, N_A))

        # Stage 5
        try:
            N = aes_decode(K_s, conn.recv(1024))
        except:
            print("Permission denied!")
            conn.close()
            continue
        N1 = N[:16]
        N2 = N[16:]
        if N1 == N_A:
            conn.send(aes_encode(K_s, N2))
        else:
            print("Permission denied!")
            conn.close()
            continue
        print("common key: ", K_s)
```