1: Programming 2

(a) Algorithm: The implement of PKDF2 and AES-GCM

```
import javax.crypto.*;
import javax.crypto.spec.GCMParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;
import java.security.*;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.KeySpec;
import java.util.Base64;

public Cryptogram(String psd) {

    String password = psd;

    if (password == null || password.isEmpty()) {
        KeyGenerator keyGen = KeyGenerator.getInstance("AES");
        // choose aes-256
        keyGen.init(256);
        password = Base64.getEncoder().encodeToString(keyGen.generateKey().getEncod
    }

    final byte[] salt = new byte[64];
    random = SecureRandom.getInstanceStrong();
    random.nextBytes(salt);

    // use PKDF2 to derive the secret K
    SecretKeyFactory secretKeyFactory = SecretKeyFactory.getInstance("PBKDF2WithHm
    KeySpec passwordBasedEncryptionKeySpec = new PBEKeySpec(password.toCharArray()
    SecretKey secretKeyFromPBKDF2 = secretKeyFactory.generateSecret(passwordBasedE
    this.mSecretKey = new SecretKeySpec(secretKeyFromPBKDF2.getEncoded(), "AES");
    this.mCipher = Cipher.getInstance("AES/GCM/NoPadding");
}

public byte[] encrypt(byte[] data){
    byte[] nonce = new byte[32];
    random.nextBytes(nonce);
    mGCMParameterSpec = new GCMParameterSpec(16 * 8, nonce);
    mCipher.init(Cipher.ENCRYPT_MODE, mSecretKey, mGCMParameterSpec);
    return mCipher.doFinal(data);
}

public byte[] decrypt(byte[] data) {
    mCipher.init(Cipher.DECRYPT_MODE, mSecretKey, mGCMParameterSpec);
    return mCipher.doFinal(data);
}
```

(b) Algorithm: The implement of Client

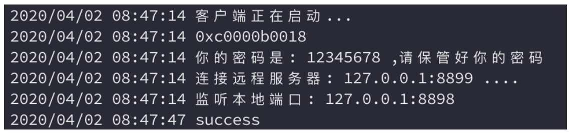
```
import java.net.ServerSocket;
import java.net.Socket;

public void listen() {
    this.localServerSocket = new ServerSocket(this.localPort);
    System.out.println("local listening port: " + serverAddr + ":" + serverPort);
    while (true) {
        try {
            Socket mSocket = localServerSocket.accept();
            new ClientThread(this.serverAddr, this.serverPort, mSocket, this, this.
            } catch (Exception e) {
                e.printStackTrace();
        }
    }
}
```

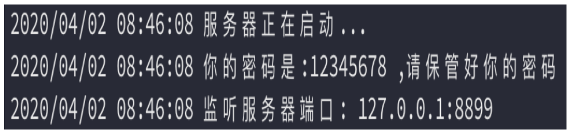
(c) Algorithm: The implement of Server

```
public void listen() {
    System.out.println("server listening port: " + port);
    serverSocket = new ServerSocket(port);
    while (true) {
        try {
            Socket mSocket = serverSocket.accept();
            new ServerThread(mSocket, this, this.pw).start();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

(d) Output:

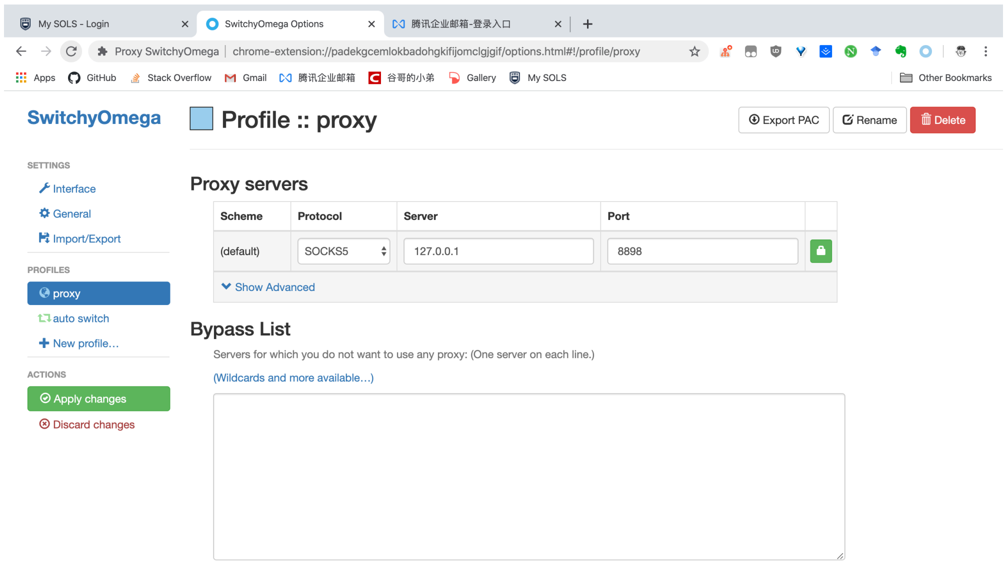


```
2020/04/02 08:47:14 客户端正在启动...
2020/04/02 08:47:14 0xc0000b0018
2020/04/02 08:47:14 你的密码是: 12345678 ,请保管好你的密码
2020/04/02 08:47:14 连接远程服务器: 127.0.0.1:8899 ....
2020/04/02 08:47:14 监听本地端口: 127.0.0.1:8898
2020/04/02 08:47:47 success
```



```
2020/04/02 08:46:08 服务器正在启动...
2020/04/02 08:46:08 你的密码是:12345678 ,请保管好你的密码
2020/04/02 08:46:08 监听服务器端口: 127.0.0.1:8899
```

Figure 1: Client and Server



Profile :: proxy

Proxy servers

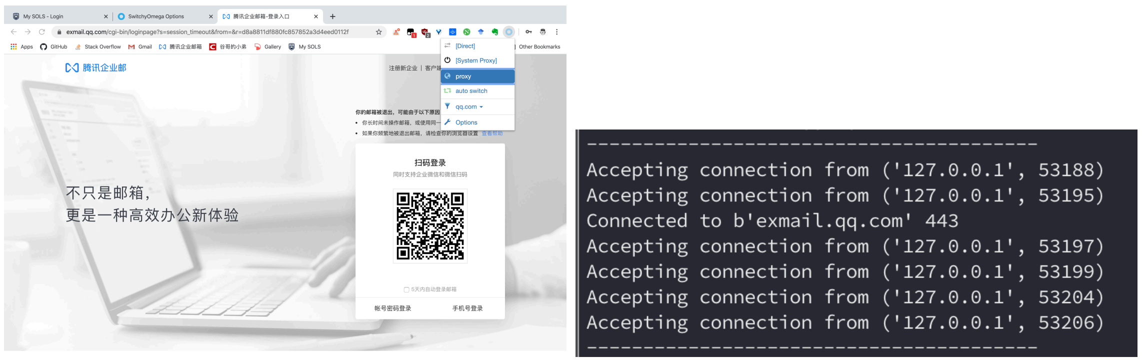
Scheme	Protocol	Server	Port
(default)	SOCKS5	127.0.0.1	8898

Bypass List

Servers for which you do not want to use any proxy: (One server on each line.)

(Wildcards and more available...)

Figure 2: Socks5Proxy



```
-----------------------------------------
Accepting connection from ('127.0.0.1', 53188)
Accepting connection from ('127.0.0.1', 53195)
Connected to b'exmail.qq.com' 443
Accepting connection from ('127.0.0.1', 53197)
Accepting connection from ('127.0.0.1', 53199)
Accepting connection from ('127.0.0.1', 53204)
Accepting connection from ('127.0.0.1', 53206)
-----------------------------------------
```

Figure 3: Output