信息安全技术-实验手册

2020.11.29

目录

Java 环境配置

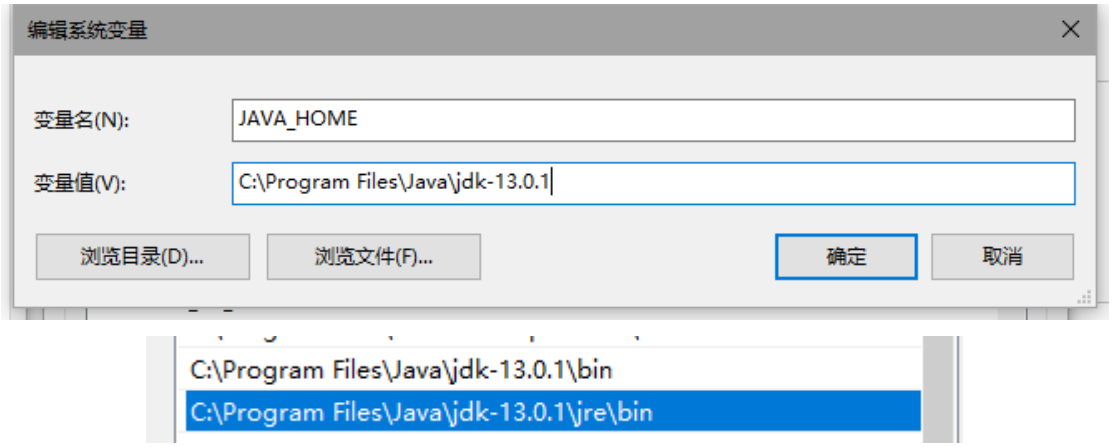
从 ORACLE 官网下载 jdk 安装包

从 Oracle 官网下载 jdk (Java Development Kit)安装包进行安装

https://www.oracle.com/technetwork/java/javase/downloads/index.html

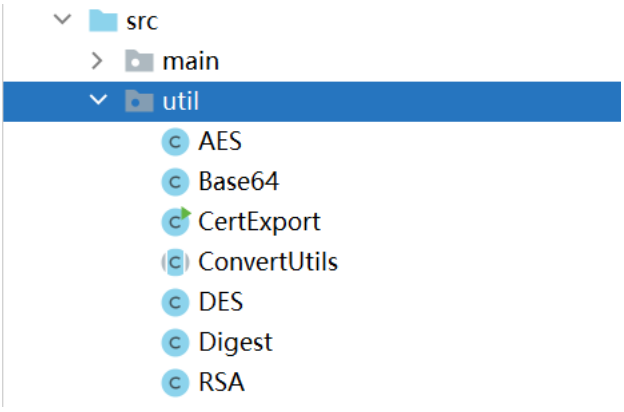
配置环境变量 JAVA_HOME 为 jdk 默认安装目录

环境变量 PATH 中添加%JAVA_HOME%\bin



打开控制台，分别输入 java -version、javac -version 查看是否有信息输出，有信息输出说明环境配置成功。

提供的源代码文件位于 util 包下，组织结构如下图所示：



DES 加解密

DES 全称为 Data Encryption Standard，即数据加密标准，是一种使用密钥加密的块算法，1977 年被美国联邦政府的国家标准局确定为联邦资料处理标准（FIPS），并授权在非密级政府通信中使用，随后该算法在国际上广泛流传开来。

DES 加解密工具代码已给出，使用如下方式进行字符串的加解密操作：

```
public class Main {
    public static void main(String args[]){
        String es=DES.encrypt( password: "aaaaaaaa", data: "asdisadjcnmvsdfdsf.ad");
        System.out.println(es);
        String ds=DES.decrypt( password: "aaaaaaaa",es );
        System.out.println(ds);
    }
}
```

使用如下方式进行文件的加解密操作：

```
DES.encryptFile( password: "aaaaaaaa", srcFile: "C:\\Users\\UbiP Lab - Laptop 01\\Desktop\\Tree.png",
        destFile: "C:\\Users\\UbiP Lab - Laptop 01\\Desktop\\Tree.png.e");
DES.encryptFile( password: "aaaaaaaa", srcFile: "C:\\Users\\UbiP Lab - Laptop 01\\Desktop\\Tree.png.e",
        destFile: "C:\\Users\\UbiP Lab - Laptop 01\\Desktop\\Tree_d.png");
```

对实验结果进行截图，尝试对不同类型的文件，如视频、音频等进行加解密处理。

AES 加解密

AES 全称为 Advanced Encryption Standard，是美国联邦政府采用的一种区块加密标准，用来替代原先的 DES。

AES 加解密工具代码已给出，使用如下方式进行字符串加解密操作：

```
String es=AES.encrypt( key: "aaaaaaaaaaaaaaaa", data: "asdisadjcnmvsdfdsf.ad");
System.out.println(es);
String ds=AES.decrypt( key: "aaaaaaaaaaaaaaaa",es );
System.out.println(ds);
```

注意，这里的密钥必须为 16 字节(128 位)。

使用如下方式对文件进行加解密：

```
AES.encryptFile( password: "aaaaaaaaaaaaaaaa", srcFile: "C:\\Users\\UbiP Lab - Laptop 01\\Desktop\\Tree.png",
        destFile: "C:\\Users\\UbiP Lab - Laptop 01\\Desktop\\Tree.png.e");
AES.decryptFile( password: "aaaaaaaaaaaaaaaa", srcFile: "C:\\Users\\UbiP Lab - Laptop 01\\Desktop\\Tree.png.e",
        destFile: "C:\\Users\\UbiP Lab - Laptop 01\\Desktop\\Tree_d.png");
```

对运行结果进行截图，对实验结果进行截图，尝试对不同类型的文件，如视频、音频等进行加解密处理。

RSA 加解密操作

RSA 加密算法，是世界上第一个非对称加密算法，也是数论的第一个实际应用。它的算法如下：

1. 找两个非常大的质数 p 和 q（通常 p 和 q 都有 155 十进制位或都有 512 十进制位）并计算 n=pq，k=(p-1)(q-1)。
2. 将明文编码成整数 M，保证 M 不小于 0 但是小于 n。
3. 任取一个整数 e，保证 e 和 k 互质，而且 e 不小于 0 但是小于 k。加密钥匙（称作公钥）是(e, n)。
4. 找到一个整数 d，使得 ed 除以 k 的余数是 1（只要 e 和 n 满足上面条件，d 肯定存在）。解密钥匙（称作密钥）是(d, n)。

加密过程： 加密后的编码 C 等于 M 的 e 次方除以 n 所得的余数。
解密过程： 解密后的编码 N 等于 C 的 d 次方除以 n 所得的余数。
只要 e、d 和 n 满足上面给定的条件。M 等于 N。

通过如下方式对字符串进行加解密：

```
try {
    Map<String,String> map= RSA.generateKeyPair();
    String pub=map.get("publicKey");
    String pri=map.get("privateKey");
    String data="testcontent";
    System.out.println(data);
    String en_result=RSA.encrypt(data, pub);
    System.out.println(en_result);
    String de_result=RSA.decrypt(en_result, pri);
    System.out.println(de_result);
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

对实验结果进行截图，比较 RSA 加解密操作与前文中对称加密算法加解密操作的不同。

SHA256 信息摘要算法

SHA256 是 SHA-2 下细分出的一种算法，产生的输出是一个 256-bit 的报文摘要。能够计算出一段信息的摘要，也即数字签名。利用数字签名可以知道消息是否被更改过，可以认证消息是否是确实来自意定的信源，还可以使信源不能否认曾将发送的消息。

Java 中提供了多个信息摘要算法，在工程中引入 java.security.MessageDigest

对一个字符串提取信息摘要的方法如下:

```
try {
    String data="testtest";
    MessageDigest messageDigest=MessageDigest.getInstance("SHA-256");
    messageDigest.update(data.getBytes( charsetName: "UTF-8"));
    String info=new String(Base64.encode(messageDigest.digest()));
    System.out.println(info);
} catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
}
```

对实验结果进行截图，对输入信息进行微小更改并比较信息摘要结果(例如 abcdefghijklmnopqrstuvwxyz 与 abcdefghijklmnopqrstuvwxyx)。

RSA 签名验证操作

RSA 算法与 hash 算法结合可进行签名操作，明文信息与其 hash 值共同使用 RSA 私钥进行签名，任一持有 RSA 公钥者可对签名进行验证。

使用如下方式进行签名与验签操作：

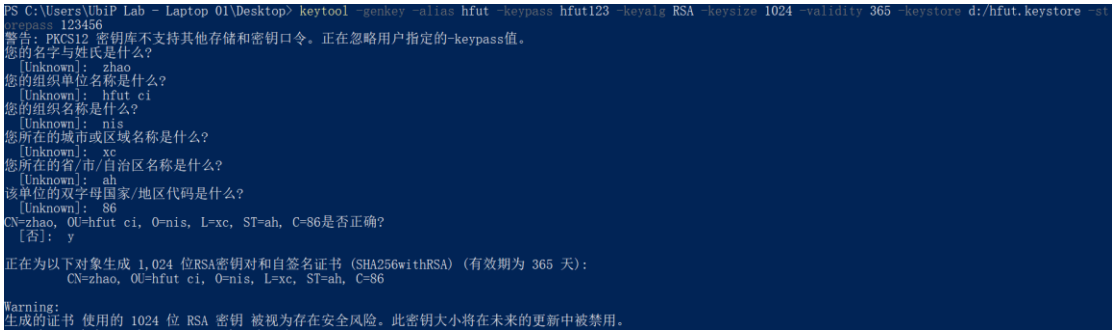
```
try{
    Map<String,String> map=RSA.generateKeyPair();
    String pub=map.get("publicKey");
    String pri=map.get("privateKey");
    String data="Whosyourdaddy";
    String sig=RSA.sign(data,pri);
    System.out.println(sig);
    System.out.println(RSA.checkSign(data,sig,pub));//succeed
    System.out.println(RSA.checkSign( content: data+"?",sig,pub));//failed
} catch (Exception e) {
    e.printStackTrace();
}
```

对实验结果进行截图。

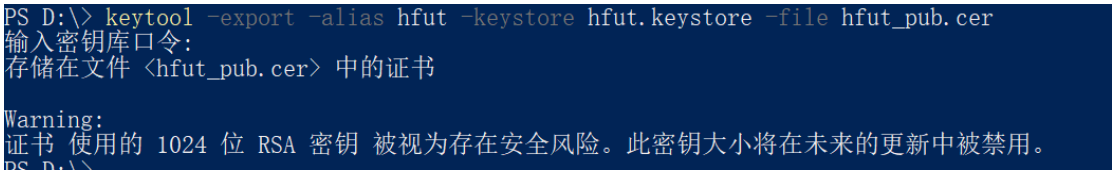
利用 keytool 生成证书

keytool 是一个密钥和证书管理工具。它使用户能够管理自己的公钥/私钥对及相关证书，用于（通过数字签名）自我认证（用户向别的用户/服务认证自己）或数据完整性以及认证服务。它还允许用户储存他们的通信对等者的公钥（以证书形式）。

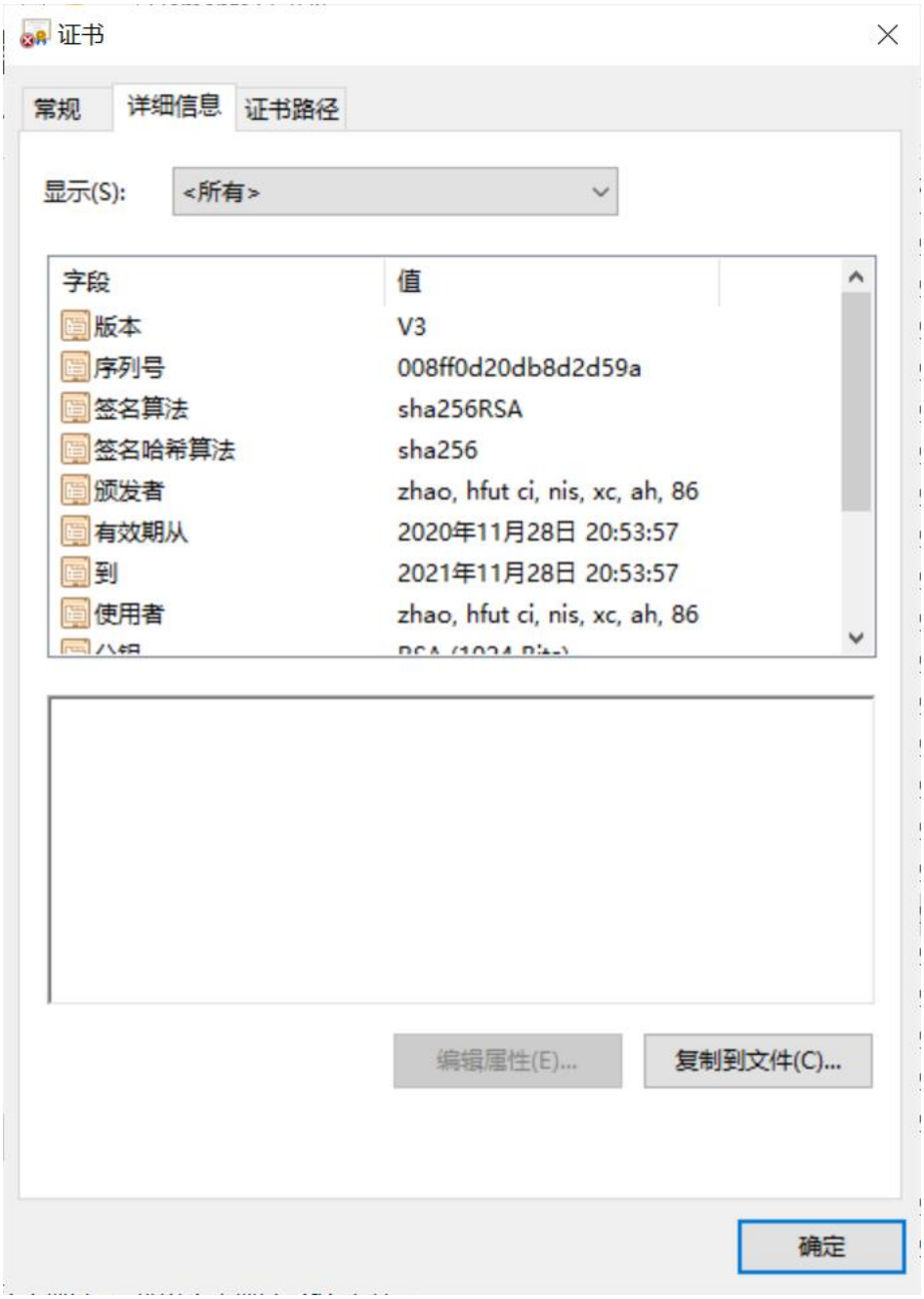
使用以下命令生成 keystore：



切换至 keystore 目录下，使用以下命令导出证书文件：



双击打开可观察到如下信息：


证书
常规
详细信息
证书路径
显示(S):
<所有>
字段
值
版本
V3
序列号
008ff0d20db8d2d59a
签名算法
sha256RSA
签名哈希算法
sha256
颁发者
zhao, hfut ci, nis, xc, ah, 86
有效期从
2020年11月28日 20:53:57
到
2021年11月28日 20:53:57
使用者
zhao, hfut ci, nis, xc, ah, 86
编辑属性(E)...
复制到文件(C)...
确定