

商用密码算法实验报告

一、实验目的

1. 掌握国密算法 SM4 的使用方法
2. 掌握国密算法 SM3 的使用方法
3. 掌握国密算法 SM2 的使用方法
4. 掌握 Java 文件读写流的过程

二、相关知识

2.1 SM4 算法

SM4 算法是无线局域网标准的分组数据算法，属于对称密码算法。对称密码算法是一种用相同的密钥进行加密和解密的技术，用于确保消息的机密性，对称密码加密速度快，常用于数据量大的信息加密。常用的国际标准对称密码算法有 DES、3DES、AES、IDEA 等，目前 DES 算法已破解，但该算法仍具有学习价值。SM4 国密算法采用非平衡的 Feistel 模型，是用于无线局域网和可信计算系统的专用分组密码算法，也可以用于其他环境下的数据加密保护，该算法的分组长度为 128 比特，密钥长度为 128 比特。

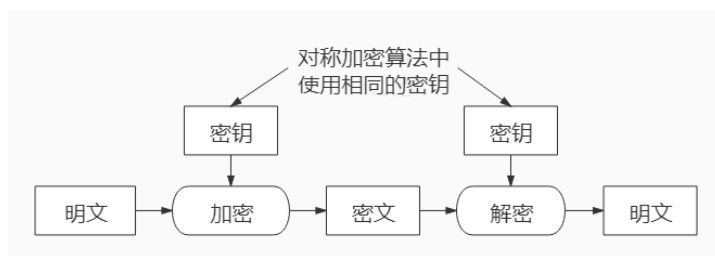


图 1 对称密码算法加解密过程

2.2 SM3 算法

SM3 算法属于消息摘要（单向散列函数）算法，单向散列函数有一个输入和一个输出，其中输入称为消息（message），输出称为散列值（hash code）。单向散列函数输出的散列值也称为消息摘要（message digest）或者指纹（fingerprint）。单向散列函数可以根据消息的内容计算出散列值，而散列值就可以被用来检查消息的完整性。散列值的长度和消息的长度无关，无论消息是 1 比特，还是 100MB，甚至是 100GB，单向散列函数都会计算出固定长度的散列值。

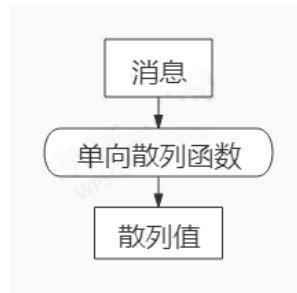


图 2 消息摘要生成流程

2.3 SM2 算法

SM2 算法属于公钥密码（非对称密码）算法，公钥密码算法密钥分为加密密钥和解密密钥两种，发送者用加密密钥对消息进行加密（签名），接收者用解密密钥对密文进行解密（验证签名）。公钥密码算法相对于对称加密算法，算法安全性高，加解密速度慢，主要用于身份认证领域以及对称密钥、重要信息的加密。

国际标准的非对称加密算法主要有 RSA、Elgamal、背包算法、Rabin、D-H、ECC（椭圆曲线加密算法）。目前，RSA-220 即 729 位的数，已经有破解成功的案例，对 RSA-155 即 512 位的数，使用服务器集群很快就可以破解。

SM2 是基于椭圆曲线的数字签名算法，其加密强度为 256 位，其特点是所需的密钥长度比 RSA 短，采用在椭圆曲线上的特定点进行特殊的乘法运算来实现的，利用了这种乘法运算的逆运算非常困难这一特性，SM2 算法在安全性、性能上与 RSA 算法相比都具有优势。

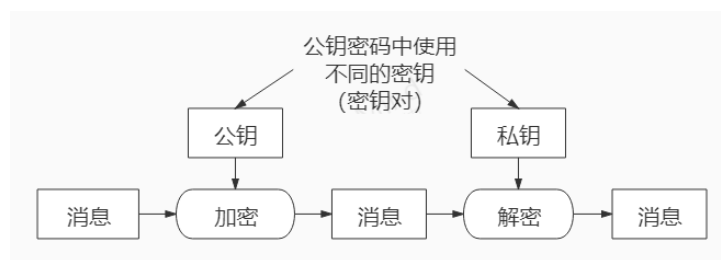


图 3 非对称密钥算法加解密过程

三、实验内容

1. 用 SM4 算法实现对一个 java 源程序文件的加解密操作。
2. 用 SM3 算法计算一个 java 源程序的消息摘要。
3. 用 SM2 算法完成对一个 java 源程序的数字签名与验证。

四、实现思路

4.1 用 SM4 算法实现对一个 java 源程序文件的加解密操作。

1. 这里导入了一个工具包 `hutool.jar` 来更好的实现文件的导入和导出。
2. 根据 SM4 的模板代码，重载了三种生成密钥的方法：不给参数生成默认密钥、用 `String seed` 生成密钥、用 `String seed` 生成指定长度的密钥
3. 生成调用 SM4Core 的加解密算法返回 `byte` 类型的数组的方法
4. 使用文件输入流 `FileInputStream` 将文件读入，并使用 `CipherInputStream` 流将其送入 SM4Core 中进行加密、并将得到的结果通过 `hutool` 中的类方法 `File.writeFromStream` 写入指定文件
5. 同样将加密的文件用 `FileInputStream` 读入并转为字符数组，同样使用 SM4Core 进行解密，将结果通过 `cipherOutputStream` 和 `FileOutputStream` 流的转换得到解密后的文件

4.2 用 SM3 算法计算一个 java 源程序的消息摘要。

1. 使用文件读写流 `FileInputStream` 将 Java 源程序读入
2. 由于 SM3 算法的摘要计算需要先将文件转为 `byte` 型数组，所以我们使用 `InputStream` 和 `ByteArrayOutputStream` 将文件内容转为 `byte` 型数组
3. 调用 `sm3Digest.doFinal` 方法生存 `byte` 类型的摘要并返回
4. 将摘要内容打印

4.3 用 SM2 算法完成对一个 java 源程序的数字签名与验证。

1. 该算法的用法很多，可以用于非对称加密也可以用于数字签名和验证
2. 首先使用标准名称创建 EC 参数生成的参数规范并获取一个椭圆曲线类型的密钥对生成器，使用该生成器获得公钥和私钥
3. 对生成的公钥和私钥进行类型转换，成为 SM2 算法能用的 `BECEPublicKey` 类型
4. 对 Java 源程序进行加密和解密并打印解密结果查看是否成功
5. 使用私钥生成签名并进行 `Sm2Verify` 进行验证

五、 结果展示

5.1 SM4









	SM2_demo	2021/6/13 15:02	JAVA 文件	12 KB
	SM2_Test	2021/6/13 16:04	JAVA 文件	14 KB
	SM3_demo	2021/6/13 14:45	JAVA 文件	3 KB
	SM3_Test	2021/6/13 14:58	JAVA 文件	3 KB
	SM4_demo	2021/6/5 9:33	JAVA 文件	8 KB
	SM4_demo_dp	2021/6/13 16:04	JAVA 文件	8 KB
	SM4_demo_re	2021/6/13 16:04	JAVA 文件	8 KB
	SM4_Test	2021/6/13 15:43	JAVA 文件	9 KB

图 5 生成加密和解密的 Java 文件 SM_demo_dp/re

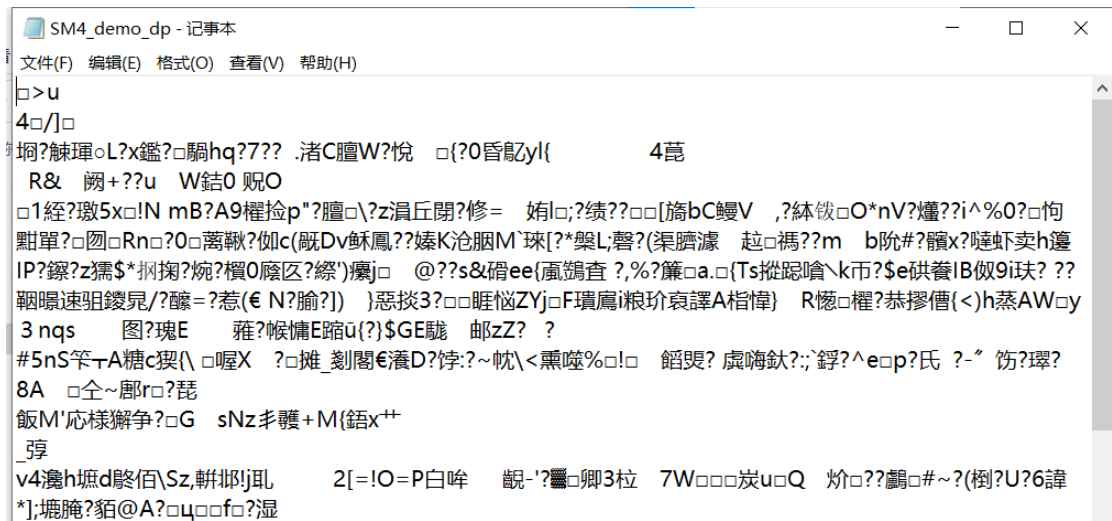
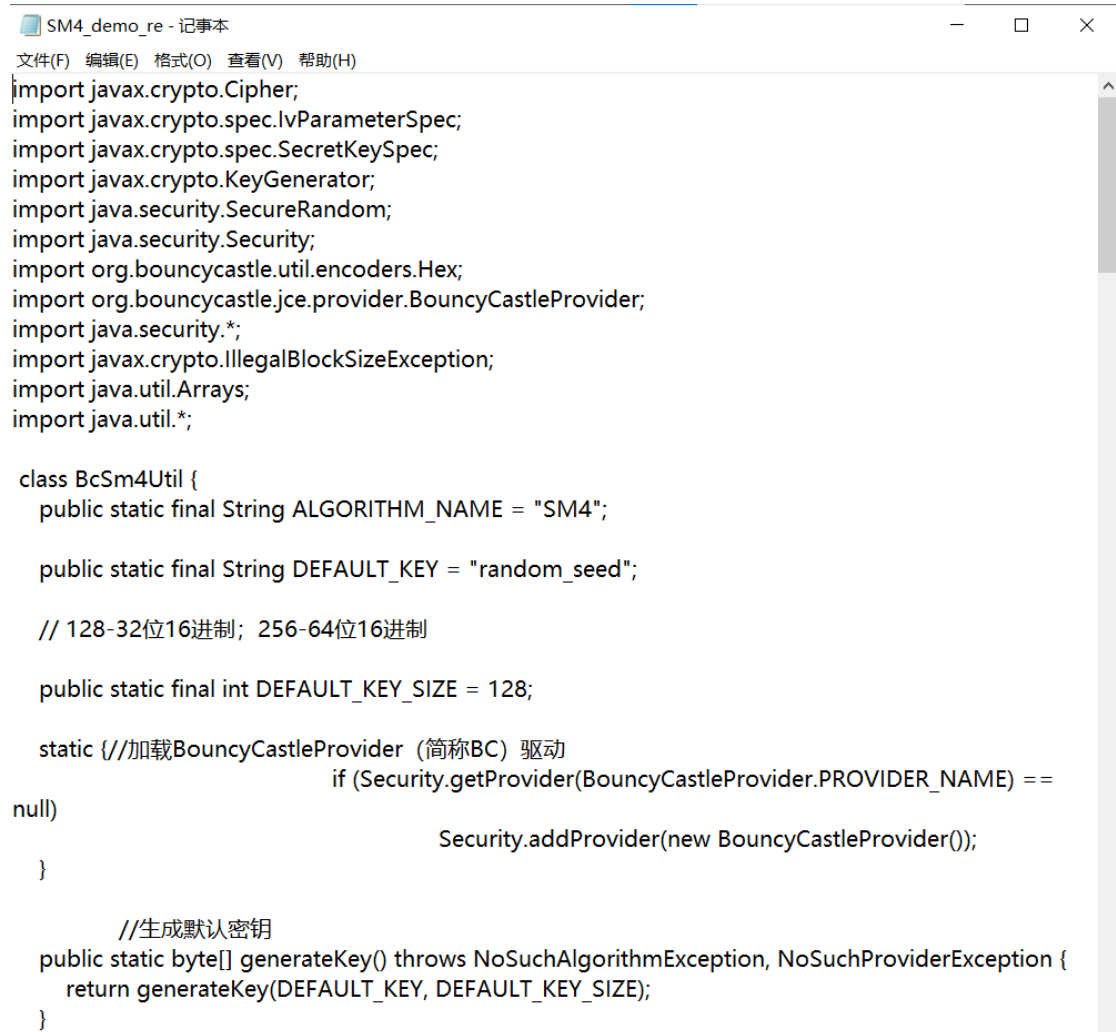


图 5 加密后的 Java 文件



```
SM4_demo_re - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import javax.crypto.KeyGenerator;
import java.security.SecureRandom;
import java.security.Security;
import org.bouncycastle.util.encoders.Hex;
import org.bouncycastle.jce.provider.BouncyCastleProvider;
import java.security.*;
import javax.crypto.IllegalBlockSizeException;
import java.util.Arrays;
import java.util.*;

class BcSm4Util {
    public static final String ALGORITHM_NAME = "SM4";

    public static final String DEFAULT_KEY = "random_seed";

    // 128-32位16进制; 256-64位16进制

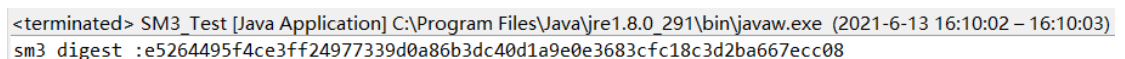
    public static final int DEFAULT_KEY_SIZE = 128;

    static (//加载BouncyCastleProvider (简称BC) 驱动
            if (Security.getProvider(BouncyCastleProvider.PROVIDER_NAME) ==
null)
                Security.addProvider(new BouncyCastleProvider());
    )

    //生成默认密钥
    public static byte[] generateKey() throws NoSuchAlgorithmException, NoSuchProviderException {
        return generateKey(DEFAULT_KEY, DEFAULT_KEY_SIZE);
    }
}
```

图 7 解压后的 Java 文件

5.2 SM3



```
<terminated> SM3_Test [Java Application] C:\Program Files\Java\jre1.8.0_291\bin\javaw.exe (2021-6-13 16:10:02 - 16:10:03)
sm3 digest :e5264495f4ce3ff24977339d0a86b3dc40d1a9e0e3683cfc18c3d2ba667ecc08
```

图 8 生成的摘要打印

5.3 SM2

```
<terminated> SM2_Test [Java Application] C:\Program Files\Java\jre1.8.0_291\bin\javaw.exe (2021-6-13 16:10:56 - 16:10:57)
SM2公钥: 04f8ff99e083388db3f358a3ad0d3be6fc6aaf5ea5b2d441d17808fc14dd2906b4eb8c19befe278f1e8c8be99b25fbb2a243565fab67bd01c10f04480c981!
SM2私钥: 2a019a687ccb57b913c7d6fcc49666e401e6524dfa789d54dabdf3c4724f2d4f

明文为Java文件中的内容
加密结果: 043068db3a45848e138d88d097ecc08145332bb4e9d9865e648ab9a82ad4c58c819f19600035c235552f8aa61de3e098e985751d7c14fd1d7d715591b407!
解密结果:
import java.io.ByteArrayOutputStream;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.math.BigInteger;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.SecureRandom;
import java.security.Security;
import java.security.Signature;
import java.security.SignatureException;
import java.security.cert.CertPathBuilderException;
import java.security.cert.CertificateException;
import java.security.spec.ECGenParameterSpec;
```

图9 公钥私钥以及加解密内容的输出

```
signature: 3045022100c8238efd71f84532aeb63c5f74ac7be7e9
Signature verify result: true
```

图10 签名与验证结果

六、实验拓展与思考

七、实验收获