（1）获得当前支持的加密算法，及其对应的名称，对应的Java类



（2）简单的加密算法演示（MD5加解密）：



（3）对象序列化时使用加密机制：



关键代码：

**注意流的构造方法：**

**try** (ObjectOutputStream oos = **new** ObjectOutputStream(

**new** CipherOutputStream(Files.*newOutputStream*(path), cipher)))

{oos.writeObject(obj);}

报文摘要（防止数据在网络传输时被篡改）：

使用md5进行摘要计算例子：



源码注释：

（1）使用MD5计算摘要：

MessageDigest md = MessageDigest.*getInstance*("MD5");

**byte**[] digest = md.digest("Hello World".getBytes());

（2）使用HMAC计算消息验证码：

KeyGenerator keyGenerator = KeyGenerator.*getInstance*("HmacMD5");

SecretKey key = keyGenerator.generateKey();

Mac mac = Mac.*getInstance*("HmacMD5");

mac.init(key);

**byte**[] result = mac.doFinal("Hello World".getBytes());

在生成了消息验证码之后，一般是添加在元数据的尾部，在同元数据一起加密传输

数字签名java.security.Signature：

公钥是公开的，

****

**如果要对对象进行签名，可以使用SignedObject对象（该对象可以封装Serizable接口的对象）**

Java安全套接字：SSL

解决问题：

（1）身份认证问题

（2）数据加密

（3添加消息验证码，验证消息的正确性

Web安全应用比较不错的例子：



注意：中的注解是在处理的

关键代码：

**public** Object invoke(Object proxy, **final** Method method, **final** Object[] args) **throws** Throwable {

Subject subject = SubjectHolder.*get*();

**return** Subject.*doAsPrivileged*(subject,

**new** PrivilegedExceptionAction<Object>() {

**public** Object run() **throws** Exception {

**try** {

DeclaredPermission annotation = method.getAnnotation(DeclaredPermission.**class**);

**if** (annotation != **null**) {

BasePermission permission = **new** BasePermission(annotation.name(), annotation.actions());

AccessController.*checkPermission*(permission);

}

**return** method.invoke(targetObj, args);

} **catch** (ReflectiveOperationException e) {

**throw** e;

}

}

}, **null**);

}

}