实例：用Java的加密机制来保护你的数据Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/644/2021_2022__E5_AE_9E_E4_BE_8B_EF_BC_9A_E7_c104_644715.htm Java开发工具包(JDK)对加密和安全性有很好的支持。其中一个优势就是其内置的对Socket通信的支持。因此，很容易做到在服务器和客户之间建立安全的数据流。 Java streams 是一个强大的编程工具。java.io包提供了很多标准的流类型，并能很容易的建立自己的流类型。流的一个有用的特点是和链表一样的简单处理过程。将 FileReader和 BufferedReader链接起来。我们在用客户机/服务器应用程序的时候也会用到类似的概念。 关键字 对于验证来说，关键字很重要，运行KeyGen来产生一个关键字。我们采用同步方法，所以客户机和服务器必须用相同的关键字。 安全socket 我们从一个简单的类开始，它提供我们在普通socket对象之上的加密。构造器创建了变量并初始化了密码： outCipher = Cipher.getInstance(algorithm). outCipher.init(Cipher.ENCRYPT_MODE, key). inCipher = Cipher.getInstance(algorithm). inCipher.init(Cipher.DECRYPT_MODE, key). 因为socket是双向的通信，所以我们采用两个密码。加密输出的数据并解密输入的数据。我们使用getInputStream()和 getOutputStream(),这两种方法来加密合解密通用的输入和输出的经过包装的数据流。 在JCE的javax.crypto包中包含CipherInputStream和CipherOutputStream这两种流类型。他们接收输入输出的流对象和密码对象。 Socket 服务器 开始写我们的socket服务器类吧。SecretSocketServer在一个端口打开ServerSocket，当接收到

连接时，使用SocketHandler产生一个线程来操作连接。
Socket 句柄 通过KeyGen来定位关键字，并建立一个SecretSocket 对象。 Key key = KeyGen.getSecretKey(). this.ss = new SecretSocket(s, key). 所有的socket 处理都是通过SecretSocket而不是Socket对象。然后我们使用下面的代码：in = ss.getInputStream(). 记住，在SecretSocket中，getInputStream是和CipherInputStream以及 InputStream相结合的。因为SocketHandler 是一个可执行的界面，我们为它生成一个 run()方法。这个方法只是在等待socket的数据。