当采用软件系统方式保存私钥时，私钥完全由软件系统进行管理，其安全性完全依赖于软件系统，不同系统下私钥存储形式和安全性可能不同。

11.2.1 PKCS#8 文件形式

PKCS#8 规定了私钥单独以文件形式保存时的具体格式。

私钥密文格式用 ASN.1 描述如下：

```
EncryptedPrivateKeyInfo ::= SEQUENCE {
    encryptionAlgorithm EncryptionAlgorithmIdentifier,
    encryptedData EncryptedData }
EncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
EncryptedData ::= OCTET STRING
```

其中，encryptionAlgorithm 是密码算法标识，用于加密私钥。该密钥通常为对称密钥，可通过口令产生，常用算法为 pbeWithMD2AndDES-CBC 和 pbeWithMD5AndDES-CBC，可参见 PKCS#5。

encryptedData 保存加密后的私钥数据。私钥明文首先以 PrivateKeyInfo 形式编码，由 encryptionAlgorithm 所标识的对称密钥加密后保存到 encryptedData 中。PrivateKeyInfo 用 ASN.1 描述如下：

```
PrivateKeyInfo ::= SEQUENCE {
    version Version,
    privateKeyAlgorithm PrivateKeyAlgorithmIdentifier,
    privateKey PrivateKey,
    attributes [0] IMPLICIT Attributes OPTIONAL }
Version ::= INTEGER
PrivateKeyAlgorithmIdentifier ::= AlgorithmIdentifier
PrivateKey ::= OCTET STRING
Attributes ::= SET OF Attribute
```

其中，version 表示格式版本，缺省值为 0。privateKeyAlgorithm 表示私钥算法标识，如 rsaEncryption。privateKey 包含私钥明文。对于 RSA 算法，为 RSAPrivateKey 类型。attributes 为同私钥一起被加密的属性集合。

11.2.2 PKCS#12 文件形式

当私钥采用 PKCS#12 形式保存时，常用的文件后缀为 pfx 或 p12。

1. PFX

PKCS #12 规定了私钥和证书以单个文件形式保存时的具体格式。用 ASN.1 描述如下：

```
PFX ::= SEQUENCE {
    Version     INTEGER {v3(3)}(v3,...),
    authSafe    ContentInfo,
    macData     MacData OPTIONAL    }
```

其中，version 表示格式版本，缺省值为 3。

2．macData

macData 包含消息认证码 mac 及相关参数 macSalt、iterations，采用口令方式保证 authSafe 数据完整性。用 ASN.1 描述如下：

```
MacData ::= SEQUENCE {
    mac         DigestInfo,
    macSalt     OCTET STRING,
    iterations  INTEGER DEFAULT  1      }
```

其中，mac 表示消息认证码，用于验证 authSafe 数据的完整性。用于计算 mac 的 MAC 密钥基于口令和 macSalt、iterations 产生。mac 值使用 HMAC 算法从 authSafe 和 MAC 密钥产生。iterations 缺省值为 1，只为兼容以前的格式，建议使用大整数，如 1024。

3．authSafe

authSafe 包含私钥和证书数据，采用密码消息 ContentInfo 类型。authSafe 的 contentType 字段可以是 signedData 类型，也可以是 data 类型。当为 signedData 类型时，基于公钥保证数据完整性；当采用 data 类型时，基于口令保证完整性，消息认证码信息保存在 macData 中。authSafe 的 content 字段直接（data 类型）或间接（signedData 类型）包含 AuthenticatedSafe 类型的编码结果。AuthenticatedSafe 用 ASN.1 描述如下：

```
AuthenticatedSafe ::= SEQUENCE OF ContentInfo
```

其中，ContentInfo 的 content 字段可以是明文 Data、加密数据 EncryptedData 或数字信封数据 EnvelopedData 类型。Data 不加密，EncryptedData 基于口令加密，EnvelopedData 基于公钥加密。当使用 EncryptedData 或 EnvelopedData 类型时，其数据的明文定义为 SafeContent 组成。

4．SafeContent

SafeContent 用 ASN.1 描述如下：

```
SafeContents ::= SEQUENCE OF SafeBag
SafeBag ::= SEQUENCE {
    bagId           BAG-TYPE.&id ({PKCS12BagSet})
    bagValue        [0] EXPLICIT BAG-TYPE.&Type({PKCS12BagSet}{@bagId}),
    bagAttributes   SET OF PKCS12Attribute OPTIONAL      }
PKCS12Attribute ::= SEQUENCE {
    attrId          ATTRIBUTE.&id ({PKCS12AttrSet}),
    attrValues SET OF ATTRIBUTE.&Type ({PKCS12AttrSet}{@attrId})   }
PKCS12AttrSet ATTRIBUTE ::= {
    friendlyName |      -- PKCS #9 定义
    localKeyId,         --PKCS #9 定义    }
```

其中，每个 SafeBag 只包含一个密钥或证书，通过 OID 来区分。bagAttributes 字段允许给密钥设定昵称（friendlyName）或标识符（localKeyId），在需要时可显示给用户查看。

5. 常用 SafeBag 类型

常用 SafeBag 类型包括：keyBag、pkcs8ShroudedKeyBag、certBag、crlBag、secretBag、safeContentsBag 等，用 ASN.1 描述如下：

```
KeyBag ::= PrivateKeyInfo
PKCS8ShroudedKeyBag ::= EncryptedPrivateKeyInfo
CertBag ::= SEQUENCE {
    certId      BAG-TYPE.&id    ({CertTypes}),
    certValue   [0] EXPLICIT BAG-TYPE.&Type ({CertTypes}{@certId})    }
CRLBag ::= SEQUENCE {
    crlId       BAG-TYPE.&id   ({CRLTypes}),
    crlValue    [0] EXPLICIT BAG-TYPE.&Type ({CRLTypes}{@crlId})     }
SecretBag ::= SEQUENCE {
    secretTypeId   BAG-TYPE.&id ({SecretTypes}),
    secretValue    [0] EXPLICIT BAG-TYPE.&Type ({SecretTypes}{secretTypeId})   }
```

其中，KeyBag 和 PKCS8ShroudedKeyBag 用于保存私钥。CertBag 用于保存数字证书，通过 OID 来区分证书类型，常用证书类型包括 x509Certificate 和 sdsiCertificate。CRLBag 用于保存 CRL，通过 OID 来区分 CRL 类型，目前只有一种类型 x509CRL。SecretBag 用于保存用户的个人秘密数据。

KeyBag 类型 OID 用 ASN.1 描述如下：

```
bagtypes OBJECT IDENTIFIER ::= {pkcs-12 10 1}
BAG-TYPE ::= TYPE-IDENTIFIER
keyBag          BAG-TYPE ::=     {KeyBag IDENTIFIED BY {bagtypes 1}}
pkcs8ShroudedKeyBag BAG-TYPE ::= {PKCS8ShroudedKeyBag IDENTIFIED BY {bagtypes 2}}
certBag BAG-TYPE ::=       {CertBag IDENTIFIED BY {bagtypes 3}}
crlBag BAG-TYPE ::= {CRLBag IDENTIFIED BY {bagtypes 4}}
secretBag BAG-TYPE ::=          {SecretBag IDENTIFIED BY {bagtypes 5}}
safeContentsBag BAG-TYPE ::=  {SafeContents IDENTIFIED BY {bagtypes 6}}
x509Certificate BAG-TYPE ::=    {OCTET STRING IDENTIFIED BY {certTypes 1}}
sdsiCertificate BAG-TYPE ::=    {IA5String IDENTIFIED BY {certTypes 2}}
x509CRL BAG-TYPE ::=     {OCTET STRING IDENTIFIED BY {certTypes 1}
```

11.2.3 Java Keystore 文件形式

JDK 1.4 已经支持 X.509 数字证书标准。Java 平台将密钥库作为密钥（公私钥对）和证书的资源管理库。从物理上讲，密钥库是个文件，缺省名称为.keystore。通过一个别名来区分密钥和证书，每个别名均由唯一的口令进行保护。密钥库本身也受口令保护。

当私钥采用 Java Keystore 形式保存时，常用的文件后缀为 jks。

Keystore 文件可通过两种方式进行访问或操作：keytool 工具，Java 类。

keytool 是 JDK 自带工具，可方便管理密钥库中的公私钥对和数字证书。主要包括以下命令：

① genkey：产生公私钥对。

② import：导入证书或证书链。例如，keytool -import -alias newroot -file root.cer-keystore server.jks 表示导入别名为 alias 的证书文件 root.cer。

③ export：导出证书。例如，keytool -export -alias myssl -keystore server.jks -rfc -file server.cer 表示将别名为 myssl 的证书导出为 server.cer 文件。

④ certreq：产生 PKCS#10 格式的证书申请请求。

⑤ storepassword：修改密钥库保护口令。

⑥ keypassword：修改私钥保护口令。

⑦ delete：删除证书。例如，keytool -delete -keystore server.jks -alias tomcat 表示删除别名为 tomcat 的证书。

⑧ list：查看密钥库信息。例如，keytool -list -v -keystore c:\server.jks 表示查看密钥库 server.jks 信息。

可通过 java.security.KeyStore 类对密钥库中的公私钥对和数字证书进行管理。主要包括以下方法。

① Enumeration<String> aliases()：列出所有别名。

② boolean containsAlias(String alias)：检查是否存在某别名。

③ void deleteEntry(String alias)：删除别名 alias。

④ Certificate getCertificate(String alias)：获取别名为 alias 的证书。

⑤ String getCertificateAlias(Certficate cert)：获取证书 cert 的别名。

⑥ Certificate[] getCertificateChain(String alias)：获取别名为 alias 的证书链。

⑦ Key getKey(String alias, char[] password)：通过口令 password 获取别名为 alias 的密钥。

⑧ void setCertificateEntry(String alias, Certificate cert)：保存证书 cert 到别名 alias 位置。

⑨ void setKeyEntry(String alias, byte[] key, Certificate[] chain)：保存密钥 key 到别名 alias 位置。

⑩ void setKeyEntry(String alias, Key key, char[] password, Certificate[] chain)：保存密钥 key 到别名 alias 位置，并使用口令 password 保护。

11.2.4　密码设备形式

密码设备是指以硬件形式存在的密码模块。

该模式下，私钥保存在密码设备中，具体包括以下几个方面。

1．密码设备分类

① 基于 IC 卡技术的密码设备。主要包括 IC 卡（接触式和非接触式）、USB Key。IC 卡通过串口与主机连接，采用 ISO 7816 协议进行通信；USB Key 通过 USB 与主机连接，通过 USB 协议进行通信。

② 密码卡。通过主机主板上的扩展槽与主机连接，采用 PCI、PCI-E 等协议进行通信。

③ 密码机。通过网络接口与主机连接，采用 TCP/IP 协议进行通信。

2．私钥存储方式

私钥在不同密码设备内部可能以不同形式存在。如 IC 卡或 USB Key 中，私钥以 EF

形式存在。在加密机或加密卡中，私钥可能存储在 EPROM 芯片或专用密码芯片中。

3．私钥存储安全性

密码设备具有很好的安全性，能有效保护私钥存储的安全性，具体措施包括：

① 不允许私钥以明文形式导出密码设备。

② 密码设备能有效防止非法强制入侵或破坏。如密码机硬件被非法强制打开时，将自动销毁所有密钥。

③ 不允许远程对密码设备进行配置管理，只允许通过密码机自带的管理屏幕进行操作。

④ 配置管理时采用高强度的身份认证手段，如口令、指纹、密钥卡。有些配置管理还要求多人共同操作。

4．私钥访问方式

因私钥保存在密码设备中，为保证安全性，应用系统不允许直接从密码设备中取出私钥后再进行解密或签名操作，而是向密码设备发送解密或签名请求，委托密码设备间接调用私钥完成解密或签名操作。

应用系统可通过以下几种方式访问私钥：

① 报文方式。应用系统直接将待解密或签名的数据组成请求包，发送给密码设备，密码设备完成解密或签名操作后，将响应包返回给应用系统。报文协议可以采用 APDU、XML、TLV 等；APDU 适用于 IC 类密码设备，XML、TLV 等适用于密码机或密码卡。通信协议可以采用 TPDU（T=0 或 T=1）、USB、串口、PCI 或 PCIE、TCP/IP、HTTP、FTP 等。TPDU、USB、串口等适用于 IC 类密码设备，PCI 或 PCIE 等适用于密码卡，TCP/IP、HTTP、FTP 等适用于密码机。

② API 方式。应用系统直接将待解密或签名的数据提交给本地 API 接口模块，由 API 接口模块作为代理，将待解密或签名的数据组成请求包，采用报文方式发送给密码设备，密码设备完成解密或签名操作后，将响应包返回给 API 接口模块，由 API 接口模块解析响应包后，将结果数据返回给应用系统。常用的 API 规范有 PKCS#11、CrptoAPI、CAPICom、JCE、PC/SC、国密接口等。

5．私钥访问安全性

密码设备具有很好的安全性，能有效保护私钥访问的安全性，具体措施包括：

① 应用系统不直接访问私钥，而是通过密码设备间接访问私钥。

② 应用系统访问密码设备需要进行身份认证。常用的认证方式有 IP 地址、口令、数字证书方式等。

③ 应用系统与密码设备之间进行通信时，可对传输数据进行加密保护。

11.2.5 软件系统形式

当采用软件系统方式保存私钥时，私钥完全由软件系统进行管理，其安全性完全依赖于软件系统，不同系统下私钥存储形式和安全性可能不同。私钥可能以单独文件形式存在，也可能与其他数据打包后保存。

Windows 系统采用 CSP（Cryptographic Service Provider）机制保存私钥。每个 CSP 都