2. RecipientInfo

接收者信息格式用 ASN.1 描述如下：

```
RecipientInfo ::= SEQUENCE {
    version Version,
    issuerAndSerialNumber IssuerAndSerialNumber,
    keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
    encryptedKey EncryptedKey }
KeyEncryptionAlgorithmIdentifier ::=   AlgorithmIdentifier
EncryptedKey ::= OCTET STRING
```

其中，version 表示消息格式版本，缺省值为 0。issuerAndSerialNumber 用于唯一确定接收者证书，由其证书签发者 DN 和证书序列号组成。keyEncryptionAlgorithm 表示密码算法标识，用于加密密钥；被加密的密钥用于加密内容，加密后的内容保存在 encryptedContentInfo→encryptedContent 中。EncryptedKey 表示被加密的密钥。

### 5.6.5 数字签名及信封消息 SignedAndEnvelopedData

数字签名及信封消息格式用 ASN.1 描述如下：

```
SignedAndEnvelopedData ::= SEQUENCE {
    version Version,
    recipientInfos RecipientInfos,
    digestAlgorithms DigestAlgorithmIdentifiers,
    encryptedContentInfo EncryptedContentInfo,
    certificates [0] IMPLICIT ExtendedCertificatesAndCertificates OPTIONAL,
    crls   [1] IMPLICIT CertificateRevocationLists OPTIONAL,
    signerInfos SignerInfos }
```

其中，version 表示消息格式版本，缺省值为 1。recipientInfos 是接收者信息集合，至少包含一个接收者。digestAlgorithms 是摘要算法标识集合，可以包含零或多个摘要算法标识。encryptedContentInfo 表示已加密的内容信息。certificates 包含签名者认证路径中的相关证书。crls 是 CRL 集合。signerInfos 是签名者信息集合，至少包含一个签名者。

### 5.6.6 摘要消息 DigestedData

摘要消息格式用 ASN.1 描述如下：

```
DigestedData ::= SEQUENCE {
    version Version,
    digestAlgorithm DigestAlgorithmIdentifier,
    contentInfo ContentInfo,
    digest Digest }
Digest ::= OCTET STRING
```

其中，version 表示消息格式版本，缺省值为 0。digestAlgorithm 是摘要算法标识。contentInfo 包含待摘要内容。digest 表示摘要值。

### 5.6.7 加密数据消息 EncryptedData

加密数据消息格式用 ASN.1 描述如下：

```
EncryptedData ::= SEQUENCE {
    version Version,
    encryptedContentInfo EncryptedContentInfo }
```

其中，version 表示消息格式版本，缺省值为 0。encryptedContentInfo 表示已加密的内容信息。

### 5.6.8 密钥协商消息 KeyAgreementInfo

密钥协商消息用于两个用户之间为协商共享密钥进行公共参数交换，仅用于 SM2 算法，具体格式用 ASN.1 描述如下：

```
KeyAgreementInfo ::= SEQUENCE {
    version         Version (1),
    tempPublicKeyR      SM2PublicKey,
    userCertificate     Certificate,
    userID              OCTET STRING}
```

其中，version 表示消息格式版本，缺省值为 1。tempPublickeyR 表示临时公钥。userCertificate 表示用户证书。userID 表示用户标识。

### 5.6.9 密码消息类型 OID

密码消息类型的 OID 见表 5-7。

表 5-7 密码消息类型 OID

| / | 消 息 类 型 | OID |
|---|---|---|
| 1 | data | 1.2.840.113549.1.7.1<br>1.2.156.10197.6.1.4.2.1（仅 SM2 算法） |
| 2 | signedData | 1.2.840.113549.1.7.2<br>1.2.156.10197.6.1.4.2.2（仅 SM2 算法） |
| 3 | envelopedData | 1.2.840.113549.1.7.3<br>1.2.156.10197.6.1.4.2.3（仅 SM2 算法） |
| 4 | signedAndEnvelopedData | 1.2.840.113549.1.7.4<br>1.2.156.10197.6.1.4.2.4（仅 SM2 算法） |
| 5 | digestedData | 1.2.840.113549.1.7.5 |
| 6 | encryptedData | 1.2.840.113549.1.7.6<br>1.2.156.10197.6.1.4.2.5（仅 SM2 算法） |
| 7 | keyAgreemengInfo | 1.2.156.10197.6.1.4.2.6（仅 SM2 算法） |

注：pkcs-7 OBJECT IDENTIFIER ::= { iso(1) member-body(2) US(840) rsadsi(113549) pkcs(1) 7 }

## 5.7 Base64 编码

Base64 是一种很常用的编码方式，可以将任意二进制字节编码成 64 个可打印的 ASCII 码字符。Base64 编码的基本原理如下。

### 1. 字符索引

64 个字符包括大小写英文字母各 26 个、10 个数字、加号和左斜杠。等号用于末尾填充。64 个字符所对应的索引如表 5-8 所示。

表 5-8 Base64 字符索引表

| 索引 | 字符 | 索引 | 字符 | 索引 | 字符 | 索引 | 字符 |
|---|---|---|---|---|---|---|---|
| 0 | A | 16 | Q | 32 | g | 48 | w |
| 1 | B | 17 | R | 33 | h | 49 | x |
| 2 | C | 18 | S | 34 | i | 50 | y |
| 3 | D | 19 | T | 35 | j | 51 | z |
| 4 | E | 20 | U | 36 | k | 52 | 0 |
| 5 | F | 21 | V | 37 | l | 53 | 1 |
| 6 | G | 22 | W | 38 | m | 54 | 2 |
| 7 | H | 23 | X | 39 | n | 55 | 3 |
| 8 | I | 24 | Y | 40 | o | 56 | 4 |
| 9 | J | 25 | Z | 41 | p | 57 | 5 |
| 10 | K | 26 | a | 42 | q | 58 | 6 |
| 11 | L | 27 | b | 43 | r | 59 | 7 |
| 12 | M | 28 | c | 44 | s | 60 | 8 |
| 13 | N | 29 | d | 45 | t | 61 | 9 |
| 14 | O | 30 | e | 46 | u | 62 | + |
| 15 | P | 31 | f | 47 | v | 63 | / |

### 2. 编码过程

将二进制字节流按 3 个字节一组进行分组，然后分别对每个 3 字节组进行编码。当二进制字节流长度不是 3 的倍数时，最后一个 3 字节组可能不足 3 个字节，编码时需要进行填充处理。每个 3 字节组可编码成 4 个字符。

将 3 字节组中的每个字节按最高位到最低位顺序排列后组成 24 位，然后按 6 位一组分成 4 组，每组取值范围为 0~63，将每组取值作为索引按照表 5-7 获得对应的字符后，最终形成 4 个字符。

如果最后的 3 字节组只有 1 个字节，则将形成 8 位，右边补足 4 位 0 比特后，可形成 2 个 6 位组，转换成 2 个字符，然后填充 2 个等号，凑成 4 个字符。如果最后的 3 字节组只有 2 个字节，将形成 16 位，右边补足 2 位 0 比特后，可形成 3 个 6 位组，转换成 3 个字符，然后填充 1 个等号，凑成 4 个字符。

Base64 编码后的字符之间允许插入回车换行符，但每行不允许超过 76 个字符。

### 3. 解码过程

将字符流中回车换行符删除后，按 4 个字符一组进行分组，然后分别对每个 4 字符组进行解码。若删除回车换行符后的字符流长度不是 4 的倍数，则该字符流有误。每个 4 字符组可解码成 3 个字节。若最后一个 4 字符组末尾是 1 或 2 个等号，将解码成 2 或 1 个字节。

将 4 字符组中每个字符按表 5-8 获得对应的索引，该索引取值范围为 0～63，将索引转换成 6 位比特后，按最高位到最低位顺序排列后组成 24 位比特；然后按 8 位一组分成 3 组，每组 8 位转换成 1 个字节，最终形成 3 个字节。

如果最后的 4 字符组末尾是 1 个等号，则前 3 个字符的索引将组合成 18 位，取前 16 位转换成 2 个字节。如果最后的 4 字符组末尾是 2 个等号，则前 2 个字符的索引将组合成 12 位，取前 8 位比特转换成 1 个字符。

# 第6章　LDAP技术

## 6.1　目录服务与LDAP概述

### 6.1.1　目录服务简介

在介绍目录服务之前，先回顾一下目录的概念。在现实生活中，我们会接触哪些与目录相关的东西呢？最常见到的目录是电话簿，电话簿中按照字母表次序列出人名以及他们的电话号码和地址，如果是企业，则电话簿中可能会有这些企业的传真号码或营业时间等信息。

另一个目录例子是图书馆的图书检索卡，在图书馆，每本图书都对应一张检索卡，检索卡上记录了图书的名字、作者姓名、出版日期、页数、价格、存放位置等内容。这些检索卡按照一定顺序进行排列，通过检索卡指示的位置，借阅人就可以找到图书。

还有一个常见的使用目录的例子是通过字典检索表查找文字，可以使用拼音进行检索，每个发音对应一个条目，条目列出了对应的文字的页码；也可以使用部首进行检索，依据部首笔画，找出对应的文字及页码。

虽然这些例子各不相同，但可以用一种方式进行统一，即每个目录可以看作一个对象，这个对象包含一个或多个属性。如电话号码簿例子中，电话号码和地址是这个对象的属性；在图书检索卡中，图书的名字、作者姓名、出版日期等都是属性；在字典例子中，页码是属性。

目录的一种更形式化的定义是：一个目录是指一组名字和值的映射，它允许根据一个给出的名字来查找对应的值。与词典相似，每一个词也许会有多个词义，在一个目录中，一个名字也许会与多个不同的信息相关联。类似地，就像一个词会有多个不同的发音和多个不同的词义，目录中的一个名字可能会有多个不同类型的值。

有了目录的定义，目录服务的定义就比较清楚了：目录服务由目录数据库和一套访问协议组成，是一个存储、组织和提供信息访问服务的软件系统，目录服务本质上是一种基于客户/服务器模型的信息查询、管理服务。目录数据库存储目录数据，它以树状的层次结构描述目录数据信息。

目录服务有如下特点：

① 为读操作进行高度优化。

② 可采用分布模式存储信息。

③ 可扩展其存储的信息类型。

④ 具备高级查询能力。

⑤ 支持信息在不同目录服务之间复制。

与关系数据库相比，目录服务更擅长查询。目录数据库中的数据读取和查询效率非常高，比关系型数据库能够快一个数量级，但数据写入效率相对较低，适用于数据不需要经常改动，但需要频繁读出的情况。

目录服务的一个简单实现是名字服务，名字服务将一个网络资源的名字与它的网络地