保密级别：

保密

传阅范围：

北京永新视博数字电视技术有限公司及与其集成的 SMS 厂商

传阅方式：

带水印的 PDF

CDCAS3.0_SMS 接口说明
<V1.1>

北京永新视博数字电视技术有限公司

Beijing Novel-Super Digital TV Technology Co., LTD

文档说明

本文档详细描述了 CDCAS3.0 同 SMS 的接口，旨在为集成双方顺利实现集成提供指导和依据。

本文档仅限于在北京永新视博数字电视技术有限公司及授权的 SMS 厂商内部公开，任何人不得擅自向外公开。

任何其他人员通过非合法途径得到此文档后，不得擅自使用或泄露给第三方，否则将被追究法律责任。

本文档的解释权属于北京永新视博数字电视技术有限公司(以下简称永新视博)。

目 录

第一章 概 述

1.1 CDCAS3.0 介绍

数字电视在全世界范围内掀起了一场新的信息产业革命，永新视博作为我国数字电视产业技术研究的骨干队伍，充分认识到掌握核心技术从而取得竞争优势的必要性。几年来，永新视博先后投入了大量的人力物力资源，2000 年，CDCAS3.0 在国内第一个通过鉴定，并成为国内第一个实现与国外系统同密运行的产品，成为广电总局推荐的首选国产系统。几年来，CDCAS3.0 系列产品得到了广电总局、中央台、北京台等众多用户的认可。

永新视博乐于同有技术实力的 SMS 厂商进行集成，共同致力于数字电视事业的发展和市场的繁荣。

1.2 定义

CDCAS3.0：永新视博自主知识产权的条件接收系统；

集成：CDCAS3.0 与 SMS 通过接口交换数据，共同完成对订户信息的管理。

1.3 缩略语

CAS：Conditional Access System，条件接收系统；

SMS：Subscriber Management System，订户管理系统。

第二章　通讯协议

2.1 基本工作流程

CDCAS3.0_SMS 接口采用 C/S 模式，CDCAS3.0 作为服务器，SMS 为客户端。为减少资源的使用并提高系统安全性，CDCAS3.0_SMS 接口在 TCP 连接上建立会话，在会话上交换数据。

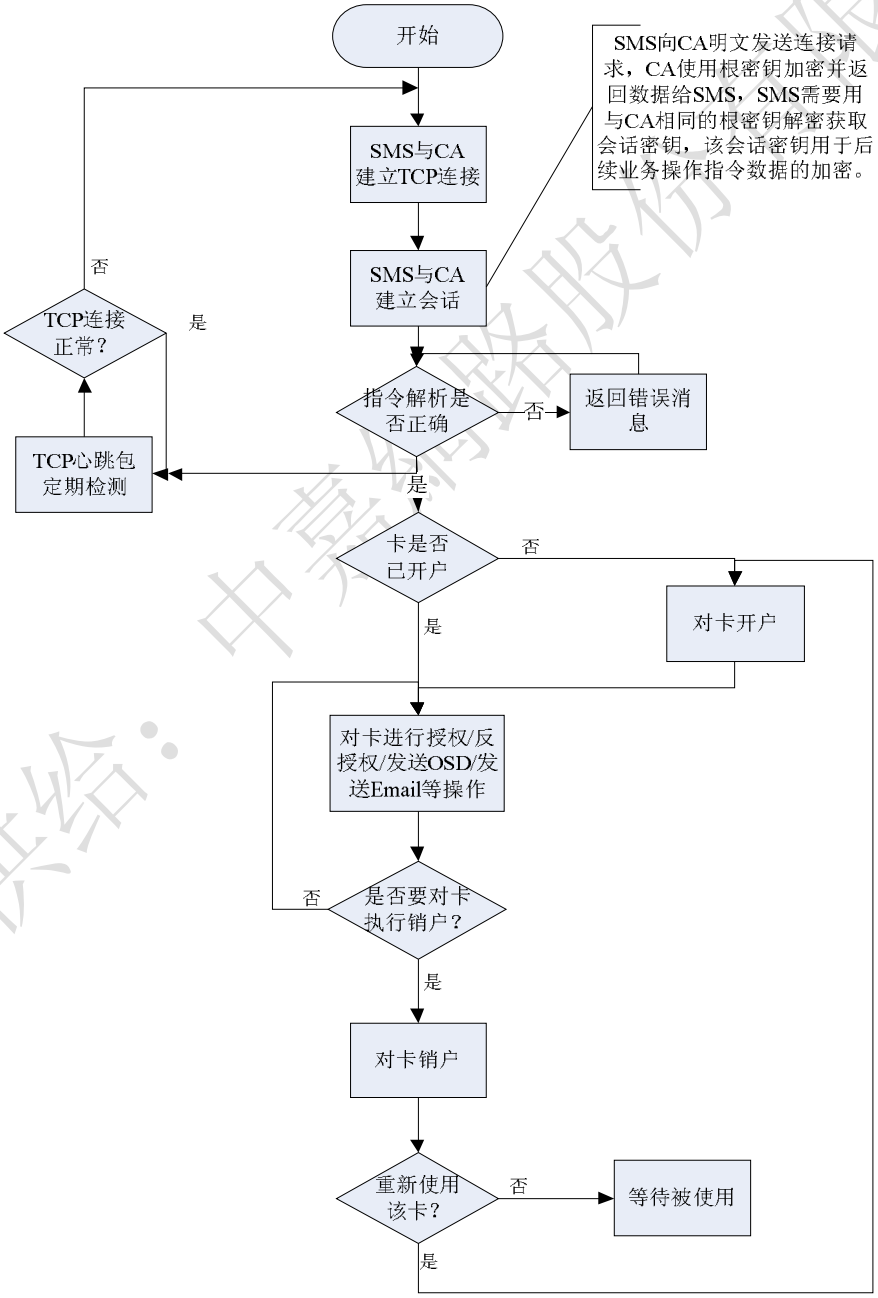


图 2.1 永新视博 CAS 系统 SMS 指令处理流程

CDCAS3.0 采用并发、冗余机制，即多个 SMS 可以连接同一个 CDCAS3.0，但是 CDCAS3.0 不支持一个 TCP 连接上的多个会话。

2.2 TCP 连接

TCP 连接的原则如下：

1. SMS 系统指定 CDCAS3.0 服务器的 IP 和端口，建立 TCP 连接。
2. TCP 心跳包定期检测是指 SMS 每 15 分钟向 CAS 发送心跳数据，内容为：0x00，0x04，0xff，0xff，0xff，0xff，0x00，0x00 ，在网络通畅的情况下，SMS 应能 收到 CA 返回的心跳数据，内容为：0x01，0x06，0xFF，0xFF，0xFF，0xFF，0x00，0x00。发送心跳数据包的目的是为了防止 TCP 连接长时间无操作被防火墙切断或者网络状况不好导致的 TCP 连接中断情况
3. SMS 如发现 TCP 连接中断（未收到心跳响应）需要主动重建连接，并重建会话。
4. SMS 发送数据需对 TCP 数据发送是否成功进行判断，确认数据是否成功发送到 CDCAS3.0。如未成功发送数据，需要重新发送数据。

2.3 会话建立

永新视博 CA 的要求，SMS（Boss）系统在向 CA 发送指令之前，需要先与 CA 建立会话连接，然后再发送指令。建立会话主要目的是获取会话密钥，用来后续业务操作的加密密钥。

创建会话的步骤如下：

1. SMS 向 CA 明文发送连接请求，SMS _CA_CREATE_SESSION_REQUEST。
2. CDCAS3.0 使用根密钥加密并返回数据给 SMS，该数据中包含会话密钥(如果 SMS 发送的数据正确)。
3. SMS 使用 SMS 根密钥解密和检测数据。
4. 数据正常响应，SMS 将获得会话密钥，该会话密钥就用来后续业务操作（即开户、授权等 CA 业务操作）指令数据的加密。

注：在同一 TCP 连接基础上每执行一次功能调用就建立一次会话是不正确的。每次会话连接将保持一段时间，直到 SMS 发送完所有请求。

2.4 数据交换方案

SMS 通过接口向 CDCAS3.0 发送数据包，传递命令请求；CDCAS3.0 执行命令后，通过接口向 SMS 发送数据包，传递命令执行结果。这样一次过程，称为

一次数据交换，而命令请求及命令执行结果均称为消息。

根据不同的情况，SMS 可以通过接口一次向 CDCAS3.0 发送单个的或者成批的命令请求，CDCAS3.0 处理完命令后，会通过接口向 SMS 发送数据包，传递所有命令的执行结果。

同时 CDCAS3.0 具有并发机制，支持多个 SMS 同时连接。

数据包的最大长度为 **4096** 字节，其数据格式是基于二进制的数据模式,且**高位在前（左）**。

CDCAS3.0_SMS 接口交换的数据包采用统一的格式，如表 2.1 所示：

表 2.1 CDCAS3.0_SMS 接口数据包格式

语法	bits	注释
Data_Section(){		
Proto_Ver	8	接口协议版本号
Crypt_Ver	6	加密方案版本号
Key_Type	2	加密密钥类别
OPE_ID	16	运营商编号
SMS_ID	16	SMS 编号
DB_Len	16	Data_Body 长度
Data_Body()		数据体
}		

Proto_Ver：接口协议的版本号，该协议由永新视博发布。当前版本号为 1。

Crypt_Ver：数据加密方案的版本号，该方案由永新视博发布。当前版本号为 1。

Key_Type：使用的密钥类别。密钥有根密钥和会话密钥两种(详见本章 3 加密方案)，共两位（2bits），值分别如下：

“00”：表示根密钥加密

“01”：表示当前会话密钥加密

“10”： 表示不加密

“11”：保留

OPE_ID：运营商编号，以 16 进制表示，比如运营商 ID 是十进制 0001，传入的就是 0x00 0x01。该编号由永新视博分发。

SMS_ID: 在当前运营商下 SMS 编号，每一个 SMS 有一个根密钥。

DB_Len：Data_Body 的长度，以字节为单位。

接口数据体 Data_Body 的格式如表 2.2 所示：

表 2.2 CDCAS3.0_SMS 接口数据体格式

语法	bits	注释
Data_Body(){		
DB_ID	16	Data_Body 编号
Msg_ID	16	消息编号
Data_Len	16	数据长度
Data_Cont()		数据内容
For(i=0; i<N; i++){		

Padding_Byte	8	补齐的字节
}		
MAC	128	数据摘要
}		

DB_ID：Data_Body 编号，由 SMS 生成，唯一标识当前会话上的一次数据交换。

Msg_ID：消息编号，在本协议中规定，表明本次发送数据的意义。消息编号的详细定义，请参见附录 1 消息编号。

Data_Len：数据长度，为数据内容 Data_Cont()长度，以字节为单位。

Data_Cont：数据内容为消息的参数，因消息的不同而不同。命令执行结果的数据内容均包含错误代码一项，其具体定义请参见附录 2 错误代码。

Padding_Byte: 为加密补齐的字节。为实现加密(详见本章 3 加密方案)，Data_Body 长度应为 8 的倍数字节，当长度不足 8 的倍数字节，要用一定的字节补齐。

MAC(Message Authentication Code)：对数据内容作数据摘要。做摘要的数据为整个Data_Body()，MAC字段本身除外。数据摘要的方法由永新视博发布，本版本的摘要方案为MD5(Message Digest5)。如果用户对该字段MD5有不清楚的地方，可以参考永新视博提供的c++编写的代码：MD5（详见附录5），该代码产生MAC数据摘要，用户可以直接利用，或者参照该代码编写MAC数据摘要产生程序。另外在本章第3小节加密方案中，有关于MAC字段的举例，用户可以参照。
附录5中MD5中有一个函数可以直接调用产生数据摘要：
unsigned char TFCA_MD5(unsigned char *pInPut,uint4 dwInput_Length,unsigned char *pOutPut)
参数含义如下：
pInPut：指向需要进行数据摘要的数据首地址。
dwInput_Length：需要进行数据摘要的长度，以字节为单位。
pOutPut：指向返回的MAC的数据首指针。
接口交换的数据包的格式相同，所不同的只有数据内容部分。为简便起见，本文档以下部分以消息编号标识各类数据包，且只给出数据包数据内容即 Data_Cont() 部分的描述。

在 CDCAS3.0_SMS 接口上会发生两种数据非法错误：数据拆包错误,数据解密失败。CDCAS3.0 返回如下格式的数据包：

表 2.3 非法数据返回包格式

语法	bits	注释
Data_Section(){		
Proto_Ver	8	接口协议版本号
Crypt_Ver	6	加密方案版本号
Key_Type	2	加密密钥类别
Reserved	16	固定为 0xFFFF
Reserved	16	固定为 0xFFFF
Reserved	16	固定为 0x0000
}		

具体字段说明与本章的数据交换方案部分相同。

2.5 加密方案

CDCAS3.0 和 SMS 在向对方发送数据包之前，必须对数据内容作摘要，并对 Data_Body()作加密，以保证数据传递的安全和可靠。

CDCAS3.0_SMS 接口使用根密钥和会话密钥两种密钥，分别用来加密会话密钥和其他数据。每个 SMS 有其对应的根密钥，由永新视博颁发。根密钥相对固定，采用双方认同的安全方式传递。（**注：**SMS 系统的根密钥应该可配置，在 CDCAS3.0 更换根密钥以后能够迅速支持根密钥的切换。）会话密钥在每次建立会话时由 CDCAS3.0 产生，通过数据包传递。根密钥用来在会话建立请求反馈时加密会话密钥，利用根密钥解会话建立请求反馈的 Data_Body 得到会话密钥。会话密钥用来加密本文第三章中订户信息管理接口、寻址指令和卡刷新数据接口部分所有的 Data_Body 部分。

在本版本的方案中，当使用根密钥加密时，加密算法为 3DES(Data Encryption Standard)；当使用会话密钥加密时，加密算法为 DES。用户如果对该部分加密算法不清楚，可以参考本文附录 6 d3des 代码。附录 6 中有两个函数可以直接调用进行 3DES 加密和 DES 加密：

bool TFCA_3DES(bool bEnspot, unsigned char* pbyKey,int nLength, unsigned char* pbySource, unsigned char* pbyTarget)
bool TFCA_DES(bool bEnspot, unsigned char* pbyKey,int nLength, unsigned char* pbySource, unsigned char* pbyTarget)

其中TFCA_3DES进行DES加密，TFCA_DES进行DES加密。参数含义如下：
bEnspot：true表示加密，false表示解密
pbyKey：指向进行加密的密钥
nLength：需要进行加密的数据长度，以字节为单位，需要为8的倍数
pbySource：指向需要进行加密的数据首指针
pbyTarget：指向返回的加密后的数据首指针

下面用会话建立来举例本接口中 MAC 校验和加密方案。

创建会话即建立会话请求的数据包明文如下：

```
unsigned char session[32] =
{0x01 ,0x06 ,0x00 ,0x01 ,0x00 ,0x01 ,0x00 ,0x18 ,
0x00 ,0x01 ,0x00 ,0x01 ,0x00 ,0x00 ,0xf0 ,0xf0 ,0xe1 ,0xee ,0x84 ,0x14 ,0x74 ,
0xbb ,0x8e ,0x30 ,0xad ,0xd7 ,0xe8 ,0xb1 ,0xbd ,0x52 ,0x0b ,0xd7};
```

分析如图 2.1 建立会话数据包包头和图 2.2 建立会话 Data_Body()所示。

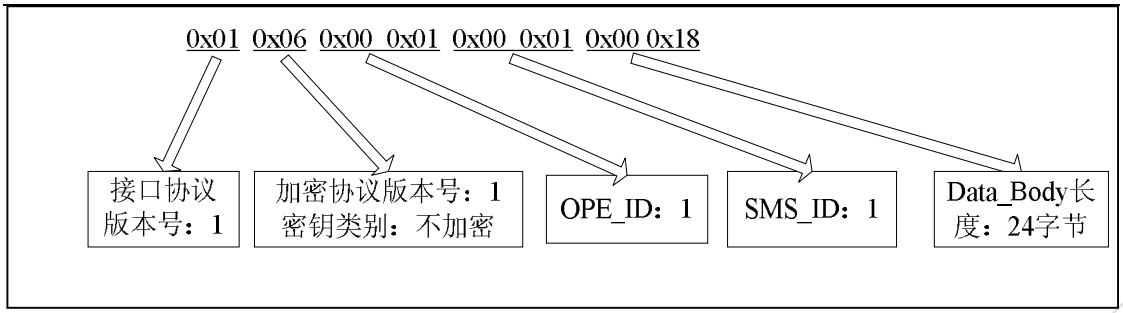


图 2.1 建立会话数据包包头

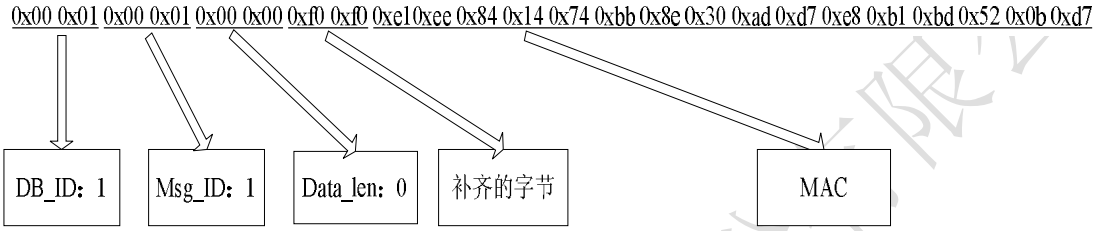


图 2.2 建立会话 Data_Body()

在返回数据包即建立会话请求反馈的数据包的 Data_Body()中将包含会话密钥。

假如根密钥是"ABCDEFGHIJKLMNOP"，返回会话密钥是 “12345678”。那么 返回数据包包头 HEAD 部分仍为明文，内容如下：

unsigned char head[8] =

{0x01,0x04,0x00,0x01,0x00,0x01,0x00,0x28}

分析如图 2.3 建立会话反馈数据包包头所示。

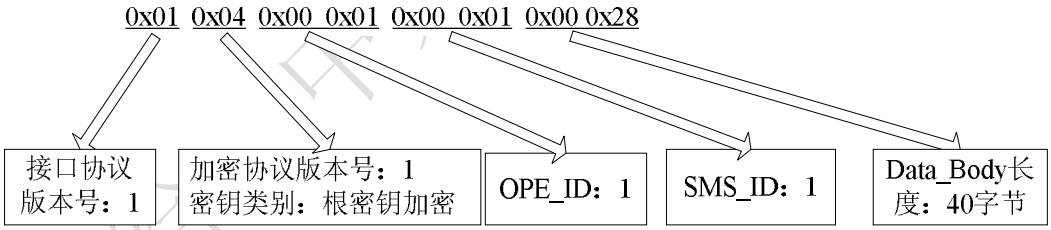


图 2.3 建立会话反馈数据包包头

返回数据包 Data_Body()部分为密文，内容如下：

unsigned char enbody[40] =

{0x0c,0xb8 ,0x8c ,0x3f ,0x86 ,0xce ,0x15 ,0x0d ,0xb5 ,0x22 ,0x65 ,0x9a ,0x56 ,0xee ,0x40 ,0xa5 ,0xef ,0xab ,0xec ,0x4b ,0x71 ,0xf0 ,0xa7 ,0xe5 ,0x7b ,0x0a ,0xd4 ,0xcd ,0x65 ,0xc4 ,0xd3 ,0xe1 ,0xa3 ,0x54 ,0x28 ,0x16 ,0x07 ,0xfa ,0x13 ,0x02}

解密后的带 MD5 的明文应该如下：

unsigned char plainbody[40] =

{0x00,0x01,0x80,0x01,0x00,0x0E,0x00,0x00,0x00,0x00,0x00,0x08,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0xF0,0xF0,0xF0,0xF0 ,0x2c ,0x90 ,0x85 ,0x0d ,0xf3 ,0xfa ,0x1a ,0x64 ,0x93 ,0xcc ,0xad ,0xa2 ,0xb4 ,0x30 ,0x72 ,0xb2}

明文详细解释如图 2.4 所示。

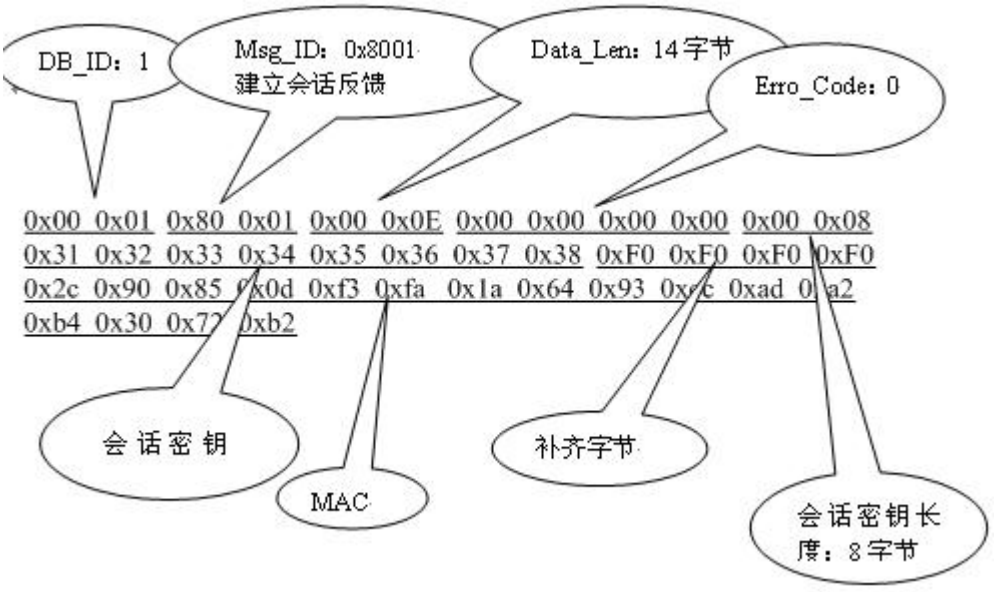


图 2.4 建立会话反馈 Data_Body()示意图

其中，创建会话请求和反馈的数据包格式，请详细参见第三章 1.2 会话建立小节。

2.6 SMS 指令依赖

SMS 指令能成功处理，必须遵守如下规则。

1. SMS 在智能卡开始使用之前必须先开户（相当于启用智能卡），在未开户的情况下，CAS 系统不允许进行授权/反授权等操作，开户是对每张智能卡进行授权、设置特征等 CA 业务的前提。
2. 销户指令是终结对卡的操作标志，CA 收到该指令后，会清空该卡的所有授权信息等，销户后的卡就可以给其他用户使用，但在重新使用的时候需要开户。
3. 每个指令即数据体(Data_Body)都有一个 DB_ID 作为标识。CAS 返回的指令处理结果也以该 DB_ID 作为标识。该标识由 SMS 确定，为了更准确的确定返回结果是哪个指令的，SMS 应该在确保一个 SMS 连接内的消息包的 DB_ID 的唯一性。不同的 SMS 连接之间可以相同。
4. CAS 将每个指令处理完后都会将处理结果返回给 OSS，所以 SMS 只要处理这些返回结果，就可以知道对应的指令是否被 CAS 执行成功，对于 CAS 返回执行结果成功的指令，SMS 就可以确认该指令已经生效，对于 CAS 返回执行结果不成功的指令，则需要 SMS 提供手工维护或自动重发的方式来处理。

第三章 接口详细说明

3.1 会话建立

建立 TCP 连接后，SMS 向 CDCAS3.0 发送建立会话请求，CDCAS3.0 返回建立会话响应。如果 CDCAS3.0 数据校验正确，则返回数据包中包含会话密钥，用以加密其他数据；如果 SMS 数据校验正确，则会话建立，可以交换其他数据。

建立会话请求(SMS_CA_CREATE_SESSION_REQUEST)

该数据包不包含数据内容，Data_Body()结构如表 3.1 所示。该数据包的 Data_Body()不加密，用明文发送。

表 3.1 SMS_CA_CREATE_SESSION_REQUEST 数据体结构

语法	bits	注释
Data_Body(){		
DB_ID	16	Data_Body 编号
Msg_ID	16	消息编号
Data_Len	16	数据长度。为 0
For(i=0; i<N; i++){		
Padding_Byte	8	补齐的字节
}		
MAC	128	数据摘要
}		

DB_ID：Data_Body()编号，由 SMS 生成，唯一标识当前会话上的一次数据交换。

Msg_ID：消息编号，在本协议中规定，表明本次发送数据的意义。消息编号的详细定义，请参见附录 1 消息编号。

Data_Len：数据长度，为数据内容长度，此处为 0。

Padding_Byte：为加密补齐的字节。

MAC(Message Authentication Code)：对数据内容作数据摘要。

建立会话请求的反馈(CA_SMS_CREATE_SESSION_RESPONSE)

该数据包的数据内容如表 3.2 所示。Data_Body()用相应 SMS 的根密钥加密。

表 3.2 CA_SMS_CREATE_SESSION_RESPONSE 数据内容

语法	bits	注释
Data_Cont(){		
Erro_Code	32	错误代码
Key_Len	16	密钥长度
For(i=0; i< Key_Len; i++){		
Key_Char	8	密钥
}		

}

Key_Len：会话密钥的长度，以字节为单位。

Key_Char：会话密钥字节。

3.2 开户

开户是智能卡可以开始使用之前的初始化工作，也是订户和智能卡建立对应关系的过程。

开户请求(SMS_CA_OPEN_ACCOUNT_REQUEST)

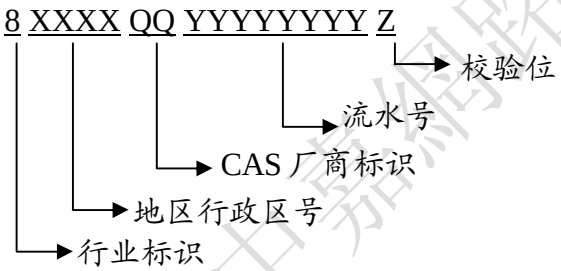
数据内容如表 3.3 所示：

表 3.3 SMS_CA_OPEN_ACCOUNT_REQUEST 数据内容

语法	bits	注释
Data_Cont(){		
CARD_SN	128	智能卡号
}		

CARD_SN：智能卡的卡号。

卡号命名规则为：



【注】：本文以下出现的CARD_SN含义相同，命名规则也相同，不再说明。卡号以ASCII码形式存储，用16进制表示，例如某张卡卡号为800……751，表示为0X38 0X30 0X30 ……0X37 0X35 0X31。

开户请求的反馈(CA_SMS_OPEN_ACCOUNT_RESPONSE)

数据内容如表 3.4 所示：

表 3.4 CA_SMS_OPEN_ACCOUNT_RESPONSE 数据内容

语法	bits	注释
Data_Cont(){		
Erro_Code	32	错误代码
}		

3.3 停户

停户是阻止用户继续收看授权节目的操作，相当于对所有授权节目的反授权。

停户的请求(SMS_CA_STOP_ACCOUNT_REQUEST)

数据内容如表 3.5 所示：

表 3.5 SMS_CA_STOP_ACCOUNT_REQUEST 数据内容

语法	bits	注释
Data_Cont(){		
CARD_SN	128	智能卡号
}		

停户请求的反馈(CA_SMS_STOP_ACCOUNT_RESPONSE)

数据内容如下表 3.6 所示：

表 3.6 CA_SMS _ STOP_ACCOUNT _RESPONSE 数据内容

语法	bits	注释
Data_Cont(){		
Erro_Code	32	错误代码
}		

3.4 冻结智能卡

冻结智能卡操作使卡不进行解扰，这时只能观看非加扰节目，不能观看加扰节目。

冻结智能卡的请求(SMS_CA_SET_LOCK_REQUEST)

数据内容如表 3.7 所示：

表 3.7 SMS_CA_SET_LOCK_REQUEST 数据内容

语法	bits	注释
Data_Cont(){		
CARD_SN	128	智能卡号
}		

冻结智能卡的反馈(CA_ SMS_SET_LOCK_ RESPONSE)

数据内容如下表 3.8 所示：

表 3.8 CA_ SMS_SET_LOCK _RESPONSE 数据内容

语法	bits	注释
Data_Cont(){		
Erro_Code	32	错误代码
}		

3.5 解除冻结智能卡

冻结的智能卡进行解除冻结以后，可以观看加扰节目。

解除冻结智能卡的请求(SMS_CA_SET_UNLOCK_REQUEST)

数据内容如表 3.9 所示：

表 3.9 SMS_CA_SET_UNLOCK_REQUEST 数据内容

语法	bits	注释
Data_Cont(){		
CARD_SN	128	智能卡号
}		

解除冻结智能卡的反馈(CA_ SMS_SET_UNLOCK_ RESPONSE)

数据内容如下表 3.10 所示：

表 3.10 CA_ SMS_SET_UNLOCK _RESPONSE 数据内容

语法	bits	注释
Data_Cont(){		
Erro_Code	32	错误代码
}		

3.6 设置订户特征

当运营商的智能卡与具体某个订户联系时，将订户的具体特征进行设置的操作，只有设置了订户特征，才能使用根据特征的寻址。

设置订户特征的请求(SMS_CA_SET_CHARACTER_REQUEST)

数据内容如表 3.11 所示：

表 3.11 SMS_CA_ SET_CHARACTER _REQUEST 数据内容

语法	bits	注释
Data_Cont(){		
CARD_SN	128	智能卡号
No	8	订户特征序号 0～9
Cha	32	订户特征
}		

No：订户特征的序号，取值范围为 0～9。具体使用情况见附录 3。

cha：订户特征（character）值，由 SMS 自己定义，如邮编等。缺省情况下默认为 0。

设置订户特征请求的反馈(CA_SMS_SET_CHARACTER_RESPONSE)

数据内容如表 3.12 所示：

表 3.12 CA_SMS_ SET_CHARACTER_RESPONSE 数据内容

语法	bits	注释
Data_Cont(){		
Erro_Code	32	错误代码
}		

3.7 机卡对应

CDCAS3.0 支持机顶盒与智能卡的对应关系。通过机卡对应消息，可以使指定智能卡与其所在的机顶盒或者指定的机顶盒进行对应，以达到收看某些要求机

卡对应节目的解密要求。

该消息的请求数据内容中，卡号字段必须有，后面的机顶盒 ID 字段可以有 30 个字节，或者 0 个字节两种选择：

机顶盒 ID 字段 30 个字节表示一个完整的机顶盒 ID 列表，每个机顶盒 ID 长度为 6 个字节，共 5 个机顶盒 ID，少于 5 个时将未使用的字节设置为 0，终端智能卡收到消息后，将与设定的机顶盒对应，前一次设置的机卡对应关系将不再保留。如果这 30 个字节全部设置为 0，智能卡则取消与所有机顶盒的 STBID 对应。

机顶盒 ID 字段 0 个字节，终端智能卡将取消与之前机顶盒的对应。

机卡对应的请求(SMS_CA_REPAIR_REQUEST)

数据内容如表 3.13 所示：

表 3.13 SMS_CA_REPAIR_REQUEST 数据内容

语法	bits	注释
Data_Cont(){		
Card_SN	128	卡号
for (i=0; i<5; i++)		
{		
STBID	48	机顶盒 ID
}		
}		

Card_SN：欲设定机卡对应的智能卡卡号。

STBID: 机顶盒 ID。

机卡对应请求的反馈(CA_SMS_REPAIR_RESPONSE)

数据内容如表 3.14 所示：

表 3.14 CA_SMS_REPAIR_RESPONSE 数据内容

语法	bits	注释
Data_Cont(){		
Erro_Code	32	错误代码
}		

3.8 重置智能卡 PIN 码

CDCAS3.0 智能卡中存有 PIN 码，用于对某些操作权限的保护。通过此消息，SMS 可以重置智能卡的 PIN 码为出厂值。

重置智能卡 PIN 码的请求(SMS_CA_RESETCARDPIN_REQUEST)

数据内容如表 3.15 所示：

表 3.15 SMS_CA_RESETCARDPIN_REQUEST 数据内容

语法	bits	注释
Data_Cont(){		
CARD_SN	128	智能卡号
}		

重置智能卡 PIN 码请求的反馈(CA_SMS_RESETCARDPIN_RESPONSE)

数据内容如表 3.16 所示：

表 3.16 CA_SMS_RESETCARDPIN_RESPONSE 数据内容

语法	bits	注释
Data_Cont(){		
Erro_Code	32	错误代码
}		

3.9 设置钱包

通过设置钱包消息设定用户的钱包内的信用额度。

设置钱包的请求(SMS_CA_SETSLOTMONEY_REQUEST)

数据内容如表 3.17 所示：

表 3.17 SMS_CA_SETSLOTMONEY_REQUEST 数据内容

语法	bits	注释
Data_Cont(){		
CARD_SN	128	智能卡号
Slot_ID	8	钱包号
Cred_Lim	16	信用额度
}		

Slot_ID：欲充值电子钱包 ID，取值范围为 1～4。

Cred_Lim：信用额度，即该电子钱包中历次充值总点数，取值范围为 0～65535。

设置钱包请求的反馈(CA_SMS_SETSLOTMONEY_RESPONSE)

数据内容如表 3.18 所示：

表 3.18 CA_SMS_SETSLOTMONEY_RESPONSE 数据内容

语法	bits	注释
Data_Cont(){		
Erro_Code	32	错误代码
}		

3.10 授权

通过授权消息，SMS 可以给指定的订户批量授权产品，并设置其录像控制属性。

授权的请求(SMS_CA_ENTITLE_REQUEST)

数据内容如表 3.19 所示：

表 3.19 SMS_CA_ENTITLE _REQUEST 数据内容

语法	bits	注释
Data_Cont(){		
CARD_SN	128	智能卡号

ProdCount	8	产品个数
for（I=0;I<Prodcout;I++）{		
Enti_Type	1	授权类型
TF_Reserved	5	永新视博保留
Prod_ID	10	产品 ID
Tape_Ctrl	8	录像控制
Start_Time	32	开始时间
End_Time	32	过期时间
}		
}		

Enti_Type：授权类型，取值范围为 0 和 1。0 为反授权，1 为授权，当为反授权时，Tape_Ctrl、Start_Time 和 End_Time 字段无效。

TF_Rerserved：永新视博保留。

Prod_ID：欲授权的产品 ID，取值范围 为：1～1023。

Tape_Ctrl：录像控制，0x00 表示不可录像，0x01 表示可以录像，0xFF 系统预留，其它数值无意义。

Start_Time：授权开始时间，time_t 格式，CAS 系统的范围限制为 2000 年 1 月 1 日 0 点 0 分 0 秒至 2030 年 12 月 31 日 23 点 59 分 59 秒。

例如：42 2D 4D D4 表示 2005-03-08 15:01:40

72 BB 4A 00 表示 2030-12-31 00:00:00

End_Time：授权结束时间，time_t 格式。CAS 系统的范围限制为 2000 年 1 月 1 日 0 点 0 分 0 秒至 2030 年 12 月 31 日 23 点 59 分 59 秒。

【注】：给每个用户的授权产品数最多为 190 个。

授权请求的反馈(CA_SMS_ENTITLE_RESPONSE)

数据内容如表 3.20 所示：

表 3.20 CA_SMS_ENTITLE _RESPONSE 数据内容

语法	bits	注释
Data_Cont(){		
Erro_Code	32	错误代码
}		

3.11 扩展授权

区别于授权接口，通过扩展授权消息，SMS 可以给指定的订户授权的产品 ID 范围更大，SMS 可根据需求选择“授权”或者“扩展授权”接口。

扩展授权的请求(SMS_CA_ENTITLEEXT_REQUEST)

数据内容如表 3.29 所示：

表 3.29 SMS_CA_ENTITLEEXT_REQUEST 数据内容

语法	bits	注释
Data_Cont(){		

CARD_SN	128	智能卡号
ProdCount	8	产品个数
for（I=0;I<Prodcout;I++）{		
Enti_Type	1	授权类型
TF_Reserved	7	永新视博保留
Prod_ID	16	产品 ID
Tape_Ctrl	8	录像控制
Start_Time	32	开始时间
End_Time	32	过期时间
}		
}		

Enti_Type：授权类型，取值范围为 0 和 1。0 为反授权，1 为授权，当为反授权时，Tape_Ctrl、Start_Time 和 End_Time 字段无效。

TF_Rerserved：永新视博保留。

Prod_ID：欲授权的产品 ID，取值范围 为：1～65533。

Tape_Ctrl：录像控制，0x00 表示不可录像，0x01 表示可以录像，0xFF 系统预留，其它数值无意义。

Start_Time：授权开始时间，time_t 格式，CAS 系统的范围限制为 2000 年 1 月 1 日 0 点 0 分 0 秒至 2030 年 12 月 31 日 23 点 59 分 59 秒。

例如：42 2D 4D D4 表示 2005-03-08 15:01:40

72 BB 4A 00 表示 2030-12-31 00:00:00

End_Time：授权结束时间，time_t 格式。CAS 系统的范围限制为 2000 年 1 月 1 日 0 点 0 分 0 秒至 2030 年 12 月 31 日 23 点 59 分 59 秒。

【注】：给每个用户的授权产品数最多为 190 个。

扩展授权请求的反馈(CA_SMS_ENTITLEEXT_RESPONSE)

数据内容如表 3.30 所示：

表 3.30 CA_SMS_ENTITLEEXT_RESPONSE 数据内容

语法	bits	注释
Data_Cont(){		
Erro_Code	32	错误代码
}		

3.12 设置子卡

通过该消息，SMS 可以将指定的订户卡设置为一个母卡的子卡。

设置子卡的请求(SMS_CA_SET_CHILD_REQUEST)

数据内容如表 3.21 所示：

表 3.21 SMS_CA_SET_CHILD_REQUEST 数据内容

语法	bits	注释

Data_Cont(){		
Card_SN	128	卡号
Parent_Card_SN	128	母卡卡号
Feed_Interval_Hour	16	喂养间隔小时数
}		

Card_SN：欲设置为子卡的卡号。

Parent_Card_SN：母卡的卡号。

Feed_Interval_Hour ：喂养间隔小时数。

设置子卡请求的反馈(CA_SMS_SET_CHILD_RESPONSE)

数据内容如表 3.22 所示：

表 3.22 CA_SMS_SET_CHILD_RESPONSE 数据内容

语法	bits	注释
Data_Cont(){		
Erro_Code	32	错误代码
}		

3.13 解除子卡

通过该消息，SMS 可以给指定的智能卡解除子卡属性，还原为一张普通卡。

解除子卡的请求(SMS_CA_CANCEL_CHILD_REQUEST)

数据内容如表 3.23 所示：

表 3.23 SMS_CA_CANCEL_CHILD_REQUEST 数据内容

语法	bits	注释
Data_Cont(){		
CARD_SN	128	智能卡号
}		

Card_SN：欲解除子卡属性的智能卡卡号。

解除子卡请求的反馈(CA_SMS_CANCEL_CHILD_RESPONSE)

数据内容如表 3.24 所示：

表 3.24 CA_SMS_CANCEL_CHILD_RESPONSE 数据内容

语法	bits	注释
Data_Cont(){		
Erro_Code	32	错误代码
}		

3.14 发送邮件

向指定用户群发送邮件。

发送邮件的请求（SMS_CA_SEND_EMAIL_REQUEST）

数据内容如表 3.31 所示：

表 3.31 SMS_CA_SEND_EMAIL_REQUEST 数据内容

语法	bits	注释
Data_Cont(){		
Exp_Len	8	表达式长度
For(i=0; i< Exp_Len; i++){		
Exp_Char	8	表达式
}		
Title_Len	8	标题长度
For(i=0; i< Title_Len;i++){		
Title_Char	8	标题
}		
Cont_Len	8	正文长度
For(i=0; i< Cont_Len; i++){		
Cont_Char	8	正文
}		
Importance	8	重要性
}		

Exp_Len：寻址表达式的长度。（最长限制为 40char，不能为空）

Exp_Char：寻址表达式字符。

Title_Len：邮件标题长度。（最长限制为 30char，不能为空）

Title_Char：邮件标题字节。

Cont_Len：邮件正文长度。

Cont_Char：邮件正文字节。（最长限制为 160char，不能为空）

Importance：邮件重要性，取值范围为 0 和 1。0 表示普通，1 表示重要。

发送邮件请求的反馈（CA_SMS_SEND_EMAIL_RESPONSE）

数据内容如表 3.32 所示：

表 3.32 CA_SMS_SEND_EMAIL_RESPONSE 数据内容

语法	bits	注释
Data_Cont(){		
Erro_Code	32	错误代码
}		

3.15 发送 OSD

向指定用户群发送 OSD。

发送 OSD 的请求（SMS_CA_SEND_OSD_REQUEST）

数据内容如表 3.33 所示：

表 3.33 SMS_CA_SEND_OSD_REQUEST 数据内容

语法	bits	注释
Data_Cont(){		
Exp_Len	8	表达式长度
For(i=0; i< Exp_Len; i++){		
Exp_Char	8	表达式
}		
Cont_Len	8	正文长度

For(i=0; i< Cont_Len; i++){		
Cont_Char	8	正文
}		
Style	8	显示方式
Duration	32	持续时间
}		

Exp_Len：寻址表达式的长度。（最大限制为 40char，不能为空）

Exp_Char：寻址表达式字节。

Cont_Len：OSD 正文长度。（最大限制为 180char，不能为空）

Cont_Char：OSD 正文字节。

Style：显示方式，取值范围为 1 和 2。1 表示屏幕上方由左到右滚动显示；2 表示屏幕下方由左到右滚动显示。

Duration：持续时间，单位为秒（s），取值范围为 1~900。

发送 OSD 请求的反馈（CA_SMS_SEND_OSD_RESPONSE）

数据内容如表 3.34 所示：

表 3.34 CA_SMS_SEND_OSD_RESPONSE 数据内容

语法	bits	注释
Data_Cont(){		
Erro_Code	32	错误代码
}		

3.16 切换频道

让指定用户切换到某个具体频道。

切换频道的请求（SMS_CA_LOCK_SERVICE_REQUEST）

数据内容如表 3.35 所示：

表 3.35 SMS_CA_ LOCK_SERVICE_REQUEST 数据内容

语法	bits	注释
Data_Cont(){		
Exp_Len	8	表达式长度
For(i=0; i< Exp_Len; i++){		
Exp_Char	8	表达式
}		
LockFlag	8	0：表示只切换不锁定。1：表示锁定。
Reserved	16	保留字段
Frequency	32	频率
Reserved	12	
fec_outer	4	前项纠错外码
Modulation	8	调制方式
symbolrate	28	符号率
fec_inner	4	前项纠错内码
PCR PID	16	时钟同步 PID 码
ComponentNum	8	节目组件个数

For(int I=0;i< ComponentNum;i++){		
CompType	8	基础流类型
CompPID	16	基础流 PID
ECMPID	16	解扰基础流 CW 的 ECM 包的 PID
}		
Reserved	8	固定为 0x00
Reserved	8	固定为 0x00
Reserved	8	固定为 0x00
Reserved	8	固定为 0x00
Reserved	8	固定为 0x00
Reserved	8	固定为 0x00
}		

Exp_Len：表达式长度。（最长限制为 40char，不能为空）
Exp_Char：表达式中的字符。
LockFlag：切换方式。取值范围为 0 和 1。0 为只切换不锁定，1 为切换并且锁定。
Reserved：永新视博保留字段。
Frequency：频率，单位为 MHz，用 BCD 码表示，取值范围为 0～9999.9999。其中前 2 字节为整数部分，后 2 字节为小数部分，不够 4 位就用 0 补。
例如：频率为 4358.59，BCD 码表示为 0x43585900。
频率为 879.564，BCD 码表示为 0x08795640。
Reserved：永新视博保留字段。
fec_outer：前向纠错外码。
Modulation：调制方式，取值范围为 0～255。具体定义如下：

0： 未定义；
1： QAM16；
2： QAM32；
3： QAM64；
4： QAM128；
5： QAM256；
6～255： 未定义。

symbolrate：符号率，单位为 MB，BCD 码表示，取值范围为 0～999.9999。其中前 3 个字节表示整数位，后 4 个字节表示小数位，不够的位数用 0 补。
fec_inner：前项纠错内码。
PCR PID：时钟同步 PID 码。
ComponentNum：组件流个数。
CompType：组件基础流类型。
CompPID：组件基础流 PID。
ECMPID：解扰基础流 CW 的 ECM 包的 PID。

使用切换频道操作时，给定将要切换至频道对应的频率到 Frequency 字段、符号率到字段 symbolrate、调制方式到字段 Modulation。其中频率和符号率在组包中应该转为 BCD 码方式，调制方式选择对应的数值。还需给将 ComponentNum 字段置为 2，然后分别给定要切换至频道的音频流和视频流的类型到字段 CompType、PID 到字段 CompPID、ECM 的 PID 到字段 ECMPID。将这些数据按

照要求写入对应的字段组成数据包传向 CA。

切换频道请求的反馈（CA_SMS_LOCK_SERVICE_RESPONSE）

数据内容如表 3.36 所示：

表 3.36 CA_SMS_ LOCK_SERVICE_RESPONSE 数据内容

语法	bits	注释
Data_Cont(){		
Erro_Code	32	错误代码
}		

3.17 解除锁定频道

当前端对指定用户做了切换频道并且锁定时，必须通过前端来解除。

解除锁定频道的请求（SMS_CA_UNLOCK_SERVICE_REQUEST）

数据内容如表 3.37 所示：

表 3.37 SMS_CA_UNLOCK_SERVICE_REQUEST 数据内容

语法	bits	注释
Data_Cont(){		
Exp_Len	8	表达式长度
For(i=0; i< Exp_Len; i++){		
Exp_Char	8	表达式
}		
}		

Exp_Len：表达式长度。（最长限制为 40char，不能为空）

Exp_Char：表达式字符。

解除锁定频道请求的反馈（CA_SMS_UNLOCK_SERVICE_RESPONSE）

数据内容如表 3.38 所示：

表 3.38 CA_SMS_UNLOCK_SERVICE_RESPONSE 数据内容

语法	bits	注释
Data_Cont(){		
Erro_Code	32	错误代码
}		

3.18 卡刷新数据接口

卡刷新数据是指如果用户没有及时收到数据或者想提高速度，可以刷新此卡对应的数据的发送。

卡刷新请求（SMS_CA_CARD_REFRESH_DATA_REQUEST）

数据内容如表 3.39 所示：

表 3.39 SMS_CA_CARD_REFRESH_DATA_REQUEST 数据内容

语法	bits	注释
Data_Cont(){		
CARD_SN	128	智能卡号
Reserved	8	固定为 0x01
}		

卡刷新请求的反馈（CA_SMS_CARD_REFRESH_DATA_RESPONSE）

数据内容如表 3.40 所示：

表 3.40 CA_SMS_CARD_REFRESH_DATA _RESPONSE 数据内容

语法	bits	注释
Data_Cont(){		
Erro_Code	32	错误代码
}		

3.19 高级预览

通过该指令，SMS 可以对指定的卡进行高级预览相关参数的设置。当参数全为 0 时，则是取消高级预览控制功能。

启用高级预览对应的请求(SMS_CA_SET_ADV_CONTROL_PREVIEW_REQUEST)

数据内容如表 3.42 所示：

表 3.42 SMS_CA_SET_ADV_CONTROL_PREVIEW_REQUEST 数据内容

语法	bits	注释
Data_Cont(){		
CARD_SN	128	智能卡号
PreviewDuration	8	免费预览时间单元片
WatchTime	8	观看时长单元
TotalCount	8	每天允许观看的总次数
TotalTime	16	每天允许观看的总时长
}		

启用高级预览对应的反馈(CA_SMS_SET_ADV_CONTROL_PREVIEW_RESPONSE)

数据内容如下表 3.43 所示：

表 3.43 CA_SMS_SET_ADV_CONTROL_PREVIEW_RESPONSE 数据内容

语法	bits	注释
Data_Cont(){		
Erro_Code	32	错误代码
}		

PreviewDuration：免费预览时间单元片。从用户初次切换到当前频道开始作为一个观看时间单元，在这个单元里可以全部允许观看或者允许观看部分时间，取决于 WatchTime 的内容。单位为分钟，取值范围（0~255）。；

WatchTime：观看时长单元。在一个 PreviewDuration 或 TotalCount 的一次下允许观看的时长。单位为分钟，取值范围（0~255）；

TotalCount：每天允许观看的总次数。从用户首次观看开始，若切换频道则计为 1 次，如果是在[WatchTime, PreviewDuration)代表的时间区间切换则不计次数；若未切换频道，但是连续观看时长达到 WatchTime，则当时长超过 PreviewDuration 时计为 1 次。单位为 4 倍次，取值范围（0~63），说明，当 TotalCount=1 时，实际上是允许观看 4 次，为 2 时，允许 8 次，以此类推。

TotalTime：每天允许观看的总时长，单位为分钟，取值范围（0~1023）。

3.20 发送超级 OSD

通过该指令，SMS 可以发送指定字体、背景颜色、显示区域、持续时间等多种属性的 OSD。

发送超级 OSD 的请求（SMS_CA_SEND_SUPEROSD_REQUEST）

数据内容如表 3.44 所示：

表 3.44 SMS_CA_SEND_SUPEROSD_REQUEST 数据内容

语法	bits	注释
Data_Cont(){		
Exp_Len	8	表达式长度
For(i=0; i< Exp_Len; i++){		
Exp_Char	8	表达式
}		
Cont_Len	16	正文长度
For(i=0; i< Cont_Len; i++){		
Cont_Char	8	正文
}		
Style	8	显示方式
Duration	32	持续时间
ForcedDisplay	8	是否强制显示
FontSize	8	字体大小
FontColor	8	字体颜色
BackgroundColor	8	背景颜色
BackgroundArea	8	显示区域
}		

Exp_Len：寻址表达式的长度。（最大限制为 40char，不能为空）

Exp_Char：寻址表达式字节。

Cont_Len：OSD 正文长度。（最大限制为 256char，不能为空）

Cont_Char：OSD 正文字节。

Style：显示方式，取值范围为 1 和 2。1 表示屏幕上方由左到右滚动显示；2 表

示屏幕下方由左到右滚动显示。

Duration：持续时间，单位为秒（s），取值范围为 1~900。

ForcedDisplay：0：不强制显示，1：强制显示

FontSize：0：默认，1：大号，2：小号

FontColor：支持 256 色

BackgroundColor：支持 256 色

BackgroundArea：表示占屏幕正中央部分的面积百分比，默认为 80，范围（20~80）

发送超级 OSD 请求的反馈（CA_SMS_SEND_SUPEROSD_RESPONSE）

数据内容如表 3.45 所示：

表 3.45 CA_SMS_SEND_SUPEROSD_RESPONSE 数据内容

语法	bits	注释

Data_Cont(){
Erro_Code
}
	32	错误代码

第四章 SMS 业务实现规范建议

4.1 如何提高 CDCAS3.0 与 OSS 系统接口的效率

OSS 可以根据实际情况减少发送给 CDCAS3.0 的指令数，此种情况主要适用于用户续费的情况，例如某用户订购产品 1 的周期为 2007 年 1 月 1 日到 2007 年 12 月 31 日，该用户在 2007 年 12 月 25 日来营业厅续交了下一年的费用，即订购了产品 1 的周期变为 2008 年 1 月 1 日到 2008 年 12 月 31 日，此时对于 OSS 来说，只需发送一条指令即可，即只发送产品 1 的订购周期为 2008 年 1 月 1 日到 2008 年 12 月 31 日，不需要在等到 2007 年 12 月 31 日先发送一个取消授权，然后 2008 年 1 月 1 日再发送一个新授权。

4.2 刷新数据

若用户没有收到运营商给定的数据，运营商可以通过卡刷新数据接口来使用户及时收到数据。卡刷新数据接口内容详见第三章中的卡刷新数据接口说明。

4.3 IPPV 业务的充值

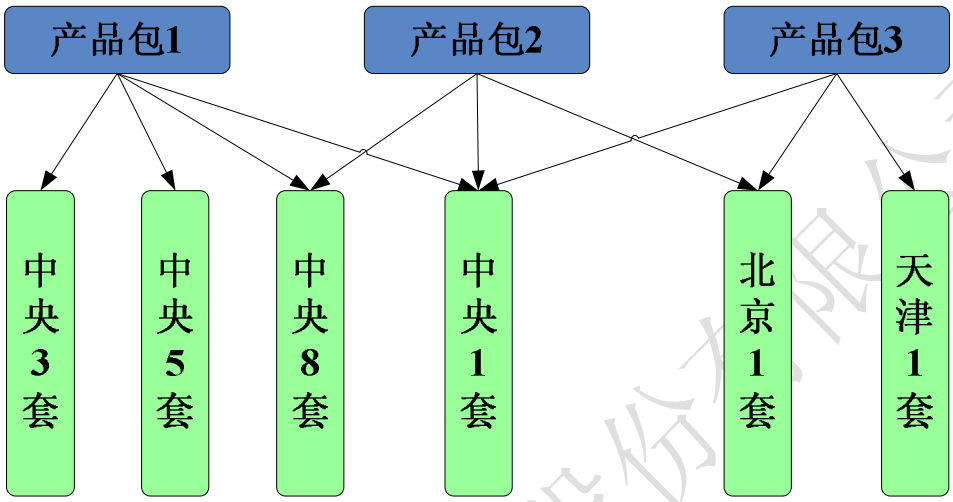
充值主要用于 IPPV 业务。CDCAS3.0 的充值方式是 SMS 根据第三章中设置钱包接口向智能卡中的钱包设置信用额度。信用额度的单位是点数，点数与钱数的汇率由运营商自行定义，建议 1 个点数代表 1 元人民币。SMS 通过接口设置的信用额度值即为智能卡中的信用额度值，因此，SMS 应保证当前设置的信用额度值应为用户历次交费的总和。

举例：假设 1 元代表 1 个点数，某张卡原有信用额度为 200 点，用户当前又交费 800 元（即用户共交费 1000 元），则 SMS 设置的该智能卡的信用额度值为 1000，智能卡收到指令后，卡内的信用额度值变为 1000。

4.4 节目和产品在授权接口中的使用

CDCAS3.0 支持各种灵活的节目打包销售方式，运营商可以对节目进行任意组合的打包，并对节目包进行整体销售。

CDCAS3.0 支持节目组嵌套打包的方式。如：可以把中央台 1、3、5、8 打成一个产品包 1，把中央 1、中央 8 和北京 1 套打成产品包 2，把中央一套、北京一套和天津一套打成产品包 3。如下图所示：



请注意：SMS 授权接口中授权针对产品而不是节目。

根据上述情况，假如用户想观看中央 3 套、中央 5 套、中央 8 套节目，则 SMS 在授权接口中对产品 1 授权即可，授权 3、5、8 的做法是错误的。假如用户想观看中央 1 套节目，则 SMS 在授权接口中可以授权产品 1、2 和 3 任意一个产品即可。

附录 1 消息编号

表附 1.1 消息编号表

代　　码	意　　义	数值
SMS_CA_CREATE_SESSION_REQUEST	建立会话请求	0x0001
CA_SMS_CREATE_SESSION_RESPONSE	建立会话响应	0x8001
SMS_CA_OPEN_ACCOUNT_REQUEST	启用智能卡请求	0x0201
CA_SMS_OPEN_ACCOUNT_RESPONSE	启用智能卡响应	0x8201
SMS_CA_STOP_ACCOUNT_REQUEST	停用智能卡请求	0x0202
CA_SMS_STOP_ACCOUNT_RESPONSE	停用智能卡响应	0x8202
SMS_CA_SET_LOCK_REQUEST	冻结智能卡请求	0x0203
CA_ SMS_SET_LOCK_ RESPONSE	冻结智能卡响应	0x8203
SMS_CA_SET_UNLOCK_REQUEST	解除冻结智能卡请求	0x0204
CA_ SMS_SET_UNLOCK_ RESPONSE	解除冻结智能卡响应	0x8204
SMS_CA_REPAIR_REQUEST	机卡对应请求	0x0206
CA_SMS_REPAIR_RESPONSE	机卡对应响应	0x8206
SMS_CA_RESETCARDPIN_REQUEST	重置智能卡 PIN 码请求	0x0207
CA_SMS_RESETCARDPIN_RESPONSE	重置智能卡 PIN 码响应	0x8207
SMS_CA_SETSLOTMONEY_REQUEST	设置钱包请求	0x0208
CA_SMS_SETSLOTMONEY_RESPONSE	设置钱包响应	0x8208
SMS_CA_SET_CHARACTER_REQUEST	设置订户特征请求	0x020C
CA_SMS_ SET_CHARACTER_RESPONSE	设置订户特征充值响应	0x820C
SMS_CA_ENTITLE_REQUEST	授权请求	0x020D
CA_SMS_ENTITLE_RESPONSE	授权响应	0x820D
SMS_CA_ENTITLEEXT_REQUEST	扩展授权请求	0x020E
CA_SMS_ENTITLEEXT_RESPONSE	扩展授权响应	0x820E
SMS_CA_SET_ADV_CONTROL_PREVIEW_REQUEST	设置高级预览请求	0x020F
CA_SMS_SET_ADV_CONTROL_PREVIEW_RESPONSE	设置高级预览响应	0x820F
SMS_CA_SEND_EMAIL_REQUEST	发送邮件请求	0x0301
CA_SMS_SEND_EMAIL_RESPONSE	发送邮件响应	0x8301
SMS_CA_SEND_OSD_REQUEST	发送 OSD 请求	0x0304
CA_SMS_SEND_OSD_RESPONSE	发送 OSD 响应	0x8304
SMS_CA_LOCK_SERVICE_REQUEST	切换频道请求	0x0306
CA_SMS_LOCK_SERVICE_RESPONSE	切换频道响应	0x8306
SMS_CA_UNLOCK_SERVICE_REQUEST	解除锁定频道请求	0x0307
CA_SMS_UNLOCK_SERVICE_RESPONSE	解除锁定频道响应	0x8307
SMS_CA_SEND_SUPEROSD_REQUEST	发送超级 OSD 请求	0x0308
CA_SMS_SEND_SUPEROSD_ RESPONSE	发送超级 OSD 响应	0x8308
SMS_CA_CARD_REFRESH_DATA_REQUEST	卡刷新数据请求	0x0401
CA_SMS_CARD_REFRESH_DATA_RESPONSE	卡刷新数据响应	0x8401
SMS_CA_SET_CHILD_REQUEST	设置子卡请求	0x0402
CA _SMS _SET_CHILD_RESPONSE	设置子卡响应	0x8402
SMS_CA_CANCEL_CHILD_REQUEST	解除子卡请求	0x0403
CA_SMS _CANCEL_CHILD_RESPONSE	解除子卡响应	0x8403

附录 2 错误代码

表附 2.1 错误代码表

代　　码	数值	意　　义
TFCA_OK	0x0000	执行成功
CARD_NOT_EXIST	0x0002	操作的卡不存在
CARD_NOT_OPEN	0x0003	尚未开户
BATCH_ENTITLE_PID_REPEAT	0x000A	批授权指令中存在重复的产品号
SLOT_NOT_EXIST	0x000B	要充值的钱包不存在
INVILID_PROD	0x000C	非法产品号
INVILID_TIME	0x000D	时间非法
ENTITLE_NOT_EXSIT	0x000E	当反授权时发现此授权不存在
ENTITLE_EXCEED_SECTION_LIMIT	0x000F	达到授权记录数的上限
ADDRCMD_EXPRESSION_TOO_LONG	0x0014	寻址表达式太长
CONENT_TOO_LONG	0x0016	内容太长
TITLE_TOO_LONG	0x0017	标题太长
DURATION_TOO_LONG	0x0018	OSD 持续时间超出范围
ADDRCMD_EXPRESSION_WRONG	0x001D	表达式通用错误（包含左右括号数目不匹配）
INVALID_OSD_STYLE	0x0020	OSD 的显示方式不对
PERSONALMAP_TYPE_INVALID	0x0027	指定的个性化信息非法
TFCAERR_MSG_INVILID	0xBBBB	非法消息
INTERFACE_UNDER_CONSTRUCTION	0xCCCC	接口暂未实现
CASSERVER_INNER_ERROR	0xDDDD	CAS 内部错误
SETCHILD_PARENT_ERR	0x0029	设置子卡时错误：母卡已经是子卡
SETCHILD_CHILD_ERR_HAVEBEPARENT	0x002A	设置子卡时错误：子卡已经是其他卡的母卡
SETCHILD_ CHILD_ERR_HAVEBECHILD	0x002B	设置子卡时错误：子卡已经是其他卡的子卡
CANCELCHILD_ERR	0x002C	解除子卡时错误：子卡已经解除
CARD_NOT_SUPPORT	0x002D	指定的卡不支持当前操作
INVALID_ TOTALCOUNT	0x0031	窗帘每天允许观看的总次数错误
INVALID_ TOTALTIME	0x0032	窗帘每天允许观看的总时长错误
INVALID_ FORCEDISPLAY	0x0033	是否强制显示 OSD 错误
INVALID_ FONTSIZE	0x0034	OSD 字号错误
INVALID_ BACKGROUNDAREA	0x0035	OSD 显示区域错误
SYSTEM_NOT_SUPPORT	0x003C	系统不支持此指令

附录 3 特征关系

特征使用情况如下所示：

元素类型	特征序号	备注：
Area	0	用户所在的区域
Bouquet	1	可以将用户分成不同的 Bouquet 而实现不同的需求，如显示不同的频道列表。
两级映射 1	2	保留
特征 0	3	
特征 1	4	
特征 2	5	
特征 3	6	
特征 4	7	如有需求，可作为机顶盒编号高 4 字节
特征 5	8	如有需求，可作为机顶盒编号低 4 字节
特征 6	9	

附录 4 寻址表达式

1. CDCAS3.0 中寻址元素

元素类型	元素指令标识	备注：
订户卡号	“card”	订户 IC 卡的内部号
订户购买产品号	“prod”	
正在观看的 Service 号	“serv”	针对当前正在观看某个 Service 的订户
特征 0（区域）	“area”	用户所在的区域
Bouquet	“bouq”	可以将用户分成不同的 Bouquet 而实现不同的需求，如显示不同的频道列表
特征 1	“cha1”	
特征 2	“cha2”	
特征 3	“cha3”	
特征 4	“cha4”	
特征 5	“cha5”	
特征 6	“cha6”	

2. 操作符

操作符用字符表示，分别如下：

类型	内容	表示
算术运算操作符	按位与	*
	按位或	+
	大于	>
	小于	<
	等于	=
逻辑运算操作符	AND	&
	OR	|
	NOT	~
	左括号	(
	右括号	)

3. 表达式

1、表达式用字符串来表示，由寻址元素和操作符组成

2、数值用十六进制或十进制表示，当用 16 进制表示时，一定要加上“ 0x”前缀。利用订户卡号寻址时，要求数值要么为 0，要么为一个 CA 中存在的卡号，而且不能利用十六进制表示。

例 1：区域等于 20

“area=20”

“area=0x14”

例 2：

“(~(cha2<234))&(cha3>675)”

例 3：

“card>0”

“card>8000302100016651”

4．指令和寻址元素的匹配表：

寻址指令 / 寻址元素	OSD	E-Mail	锁定频道/解除锁定
订户卡号	√	√	√
订户购买产品号	√	×	×
正在观看的 Service 号	√	×	×
特征	√	√	×

5．寻址元素和算术运算操作符的匹配表：

操作符 / 寻址元素	*	+	>	<	=
订户卡号	√	√	√	√	√
订户购买产品号	×	×	×	×	√
正在观看的 Service 号	×	×	×	×	√
特征	√	√	√	√	√

附录 5 MD5 代码

1. MD5.h 文件

```
/* ============================================================
Copyright (C) 2002  CAS,
R&D Center, yongxin TongFang Co.,Ltd.
All rights reserved.

FileName:                MD5.h
                                    Create Date:          2002.05.16
Author:                             gaozhiyang
description :      这个文件声明做 MAC 的函数
============================================================ */
#ifndef CDCAS3.0_MD5_H
#define CDCAS3.0_MD5_H

#define S11 7
#define S12 12
#define S13 17
#define S14 22
#define S21 5
#define S22 9
#define S23 14
#define S24 20
#define S31 4
#define S32 11
#define S33 16
#define S34 23
#define S41 6
#define S42 10
#define S43 15
#define S44 21
typedef unsigned      long uint4;
typedef unsigned      short  uint2;
typedef unsigned      char uint1;

#ifdef  __cplusplus
extern "C" {
#endif
unsigned char TFCA_MD5(unsigned char *pInPut,uint4 dwInput_Length,unsigned char
*pOutPut);
/*
TFCA_MD5进行数据摘要。参数含义如下：
pInPut：指向需要进行数据摘要的数据首地址
dwInput_Length：需要进行数据摘要的长度，以字节为单位
```

```
pOutPut：指向返回的MAC的数据首指针
*/
#ifdef __cplusplus
}
#endif

#endif //CDCAS3.0_MD5_h
```

MD5.c

2. MD5.c

```
/* ============================================================
Copyright (C) 2002  CAS,
R&D Center, yongxin TongFang Co.,Ltd.
All rights reserved.

FileName:                    MD5.C
                                        Create Date:          2002.05.16
Author:                              gaozhiyang
description :     这个文件声明做 MAC 的函数
============================================================ */

#include <stdio.h>
#include <string.h>
#include "md5.h"

void encode (uint1 *output, uint4 *input, uint4 len)
{
  uint4 i, j;

  for (i = 0, j = 0; j < len; i++, j += 4)
  {
    output[j]   = (uint1)  (input[i] & 0xff);
    output[j+1] = (uint1) ((input[i] >> 8) & 0xff);
    output[j+2] = (uint1) ((input[i] >> 16) & 0xff);
    output[j+3] = (uint1) ((input[i] >> 24) & 0xff);
  }
}
void decode(uint4 *output, uint1 *input, uint4 len)
{
  uint4 i, j;

  for (i = 0, j = 0; j < len; i++, j += 4)
    output[i] = ((uint4)input[j]) | (((uint4)input[j+1]) << 8) |
      (((uint4)input[j+2]) << 16) | (((uint4)input[j+3]) << 24);
}

uint4 rotate_left(uint4 x, uint4 n)
{
  return (x << n) | (x >> (32-n))  ;
}

uint4 F(uint4 x, uint4 y, uint4 z)
{
```

```
  return (x & y) | (~x & z);
}

void FF(uint4 * a, uint4 b, uint4 c, uint4 d, uint4 x,uint4  s, uint4 ac)
{
        *a += F(b, c, d) + x + ac;
        *a = rotate_left (*a, s) +b;
}

uint4 G(uint4 x, uint4 y, uint4 z)
{
  return (x & z) | (y & ~z);
}

void GG(uint4 *a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac)
{
        *a += G(b, c, d) + x + ac;
        *a = rotate_left (*a, s) +b;
}

uint4 H(uint4 x, uint4 y, uint4 z)
{
  return x ^ y ^ z;
}
void HH(uint4 *a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac)
{
        *a += H(b, c, d) + x + ac;
        *a = rotate_left (*a, s) +b;
}

uint4 I(uint4 x, uint4 y, uint4 z)
{
  return y ^ (x | ~z);
}

void II(uint4 *a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac)
{
        *a += I(b, c, d) + x + ac;
        *a = rotate_left (*a, s) +b;
}

void transform (uint1 block[64],uint4 *pstate,uint1 *pfinalized){

  uint4 a = pstate[0], b = pstate[1], c = pstate[2], d = pstate[3], x[16];
  decode (x, block, 64);

  /* Round 1 */
  FF(&a, b, c, d, x[ 0], S11, 0xd76aa478); /* 1 */
  FF(&d, a, b, c, x[ 1], S12, 0xe8c7b756); /* 2 */
  FF(&c, d, a, b, x[ 2], S13, 0x242070db); /* 3 */
  FF(&b, c, d, a, x[ 3], S14, 0xc1bdceee); /* 4 */
  FF(&a, b, c, d, x[ 4], S11, 0xf57c0faf); /* 5 */
  FF(&d, a, b, c, x[ 5], S12, 0x4787c62a); /* 6 */
  FF(&c, d, a, b, x[ 6], S13, 0xa8304613); /* 7 */
  FF(&b, c, d, a, x[ 7], S14, 0xfd469501); /* 8 */
  FF(&a, b, c, d, x[ 8], S11, 0x698098d8); /* 9 */
  FF(&d, a, b, c, x[ 9], S12, 0x8b44f7af); /* 10 */
```

```
  FF(&c, d, a, b, x[10], S13, 0xffff5bb1); /* 11 */
  FF(&b, c, d, a, x[11], S14, 0x895cd7be); /* 12 */
  FF(&a, b, c, d, x[12], S11, 0x6b901122); /* 13 */
  FF(&d, a, b, c, x[13], S12, 0xfd987193); /* 14 */
  FF(&c, d, a, b, x[14], S13, 0xa679438e); /* 15 */
  FF(&b, c, d, a, x[15], S14, 0x49b40821); /* 16 */

 /* Round 2 */
  GG(&a, b, c, d, x[ 1], S21, 0xf61e2562); /* 17 */
  GG(&d, a, b, c, x[ 6], S22, 0xc040b340); /* 18 */
  GG(&c, d, a, b, x[11], S23, 0x265e5a51); /* 19 */
  GG(&b, c, d, a, x[ 0], S24, 0xe9b6c7aa); /* 20 */
  GG(&a, b, c, d, x[ 5], S21, 0xd62f105d); /* 21 */
  GG(&d, a, b, c, x[10], S22,  0x2441453); /* 22 */
  GG(&c, d, a, b, x[15], S23, 0xd8a1e681); /* 23 */
  GG(&b, c, d, a, x[ 4], S24, 0xe7d3fbc8); /* 24 */
  GG(&a, b, c, d, x[ 9], S21, 0x21e1cde6); /* 25 */
  GG(&d, a, b, c, x[14], S22, 0xc33707d6); /* 26 */
  GG(&c, d, a, b, x[ 3], S23, 0xf4d50d87); /* 27 */
  GG(&b, c, d, a, x[ 8], S24, 0x455a14ed); /* 28 */
  GG(&a, b, c, d, x[13], S21, 0xa9e3e905); /* 29 */
  GG(&d, a, b, c, x[ 2], S22, 0xfcefa3f8); /* 30 */
  GG(&c, d, a, b, x[ 7], S23, 0x676f02d9); /* 31 */
  GG(&b, c, d, a, x[12], S24, 0x8d2a4c8a); /* 32 */

  /* Round 3 */
  HH(&a, b, c, d, x[ 5], S31, 0xfffa3942); /* 33 */
  HH(&d, a, b, c, x[ 8], S32, 0x8771f681); /* 34 */
  HH(&c, d, a, b, x[11], S33, 0x6d9d6122); /* 35 */
  HH(&b, c, d, a, x[14], S34, 0xfde5380c); /* 36 */
  HH(&a, b, c, d, x[ 1], S31, 0xa4beea44); /* 37 */
  HH(&d, a, b, c, x[ 4], S32, 0x4bdecfa9); /* 38 */
  HH(&c, d, a, b, x[ 7], S33, 0xf6bb4b60); /* 39 */
  HH(&b, c, d, a, x[10], S34, 0xbebfbc70); /* 40 */
  HH(&a, b, c, d, x[13], S31, 0x289b7ec6); /* 41 */
  HH(&d, a, b, c, x[ 0], S32, 0xeaa127fa); /* 42 */
  HH(&c, d, a, b, x[ 3], S33, 0xd4ef3085); /* 43 */
  HH(&b, c, d, a, x[ 6], S34,  0x4881d05); /* 44 */
  HH(&a, b, c, d, x[ 9], S31, 0xd9d4d039); /* 45 */
  HH(&d, a, b, c, x[12], S32, 0xe6db99e5); /* 46 */
  HH(&c, d, a, b, x[15], S33, 0x1fa27cf8); /* 47 */
  HH(&b, c, d, a, x[ 2], S34, 0xc4ac5665); /* 48 */

  /* Round 4 */
  II(&a, b, c, d, x[ 0], S41, 0xf4292244); /* 49 */
  II(&d, a, b, c, x[ 7], S42, 0x432aff97); /* 50 */
  II(&c, d, a, b, x[14], S43, 0xab9423a7); /* 51 */
  II(&b, c, d, a, x[ 5], S44, 0xfc93a039); /* 52 */
  II(&a, b, c, d, x[12], S41, 0x655b59c3); /* 53 */
  II(&d, a, b, c, x[ 3], S42, 0x8f0ccc92); /* 54 */
  II(&c, d, a, b, x[10], S43, 0xffeff47d); /* 55 */
  II(&b, c, d, a, x[ 1], S44, 0x85845dd1); /* 56 */
  II(&a, b, c, d, x[ 8], S41, 0x6fa87e4f); /* 57 */
  II(&d, a, b, c, x[15], S42, 0xfe2ce6e0); /* 58 */
  II(&c, d, a, b, x[ 6], S43, 0xa3014314); /* 59 */
  II(&b, c, d, a, x[13], S44, 0x4e0811a1); /* 60 */
  II(&a, b, c, d, x[ 4], S41, 0xf7537e82); /* 61 */
```

```
 II(&d, a, b, c, x[11], S42, 0xbd3af235); /* 62 */
 II(&c, d, a, b, x[ 2], S43, 0x2ad7d2bb); /* 63 */
 II(&b, c, d, a, x[ 9], S44, 0xeb86d391); /* 64 */

 pstate[0] += a;
 pstate[1] += b;
 pstate[2] += c;
 pstate[3] += d;
  memset ( (uint1 *) x, 0, sizeof(x));
}
 void update (uint1 *input, uint4 input_length,uint4 *pstate,uint1 *pfinalized,uint4 *pcount,uint1
*pbuffer) {

 uint4 input_index, buffer_index;
 uint4 buffer_space;

 if (*pfinalized)
   return;
 buffer_index = (uint4)((pcount[0] >> 3) & 0x3F);

 if (  (pcount[0] += ((uint4) input_length << 3))<((uint4) input_length << 3) )
   pcount[1]++;

 pcount[1] += ((uint4)input_length >> 29);

 buffer_space = 64 - buffer_index;

 if (input_length >= buffer_space)
 {
   memcpy (pbuffer + buffer_index, input, buffer_space);
   transform (pbuffer,pstate,pfinalized);

   for (input_index = buffer_space; input_index + 63 < input_length;  input_index += 64)
     transform (input+input_index,pstate,pfinalized);

   buffer_index = 0;
 }
 else
   input_index=0;

 memcpy(pbuffer+buffer_index, input+input_index, input_length-input_index);
}

void finalize(uint4 *pstate,uint1 *pfinalized,uint4 *pcount,uint1 *pbuffer,uint1 *pOutPut)
{
 unsigned char bits[8];
 uint4 index, padLen;
 uint1 PADDING[64]=
 {
   0x80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
 };

 if (*pfinalized)
   return;
```

```
  encode (bits, pcount, 8);
  index = (uint4) ((pcount[0] >> 3) & 0x3f);
  padLen = (index < 56) ? (56 - index) : (120 - index);
  update (PADDING, padLen,pstate,pfinalized,pcount,pbuffer);
  update (bits, 8,pstate,pfinalized,pcount,pbuffer);
  encode (pOutPut, pstate, 16);
  memset (pbuffer, 0, sizeof(*pbuffer));
  *pfinalized=1;
}

unsigned char  TFCA_MD5(unsigned char *pInPut,uint4 dwInput_Length,unsigned char
*pOutPut)
{
        uint4 state[4];
        uint4 count[2];
        uint1 buffer[64];
        uint1 finalized;

        finalized=0;
        count[0] = 0;
        count[1] = 0;

        state[0] = 0x67452301;
        state[1] = 0xefcdab89;
        state[2] = 0x98badcfe;
        state[3] = 0x10325476;

        update(pInPut, dwInput_Length,state,&finalized,count,buffer);
        finalize(state,&finalized,count,buffer,pOutPut);
        return 0x00;
}
```

附录 6 d3des 代码

1. d3des.h 文件

```
/* d3des.h -
 *
 *	Headers and defines for d3des.c
 *	Graven Imagery, 1992.
 *
 * Copyright (c) 1988,1989,1990,1991,1992 by Richard Outerbridge
 *	(GEnie : OUTER; CIS : [71755,204])
 */

#define EN0	0	/* MODE == encrypt */
#define DE1	1	/* MODE == decrypt */

#ifdef  __cplusplus
extern "C"{
#endif
/* A useful alias on 68000-ish machines, but NOT USED HERE. */

typedef union {
	unsigned long blok[2];
	unsigned short word[4];
	unsigned char byte[8];
	} M68K;

extern void deskey(unsigned char *, short);
/*			hexkey[8]     MODE
 * Sets the internal key register according to the hexadecimal
 * key contained in the 8 bytes of hexkey, according to the DES,
 * for encryption or decryption according to MODE.
 */

extern void usekey(unsigned long *);
/*			cookedkey[32]
 * Loads the internal key register with the data in cookedkey.
 */

extern void cpkey(unsigned long *);
/*			cookedkey[32]
 * Copies the contents of the internal key register into the storage
 * located at &cookedkey[0].
 */

extern void des(unsigned char *, unsigned char *);
/*			from[8]		to[8]
 * Encrypts/Decrypts (according to the key currently loaded in the
 * internal key register) one block of eight bytes at address 'from'
 * into the block at address 'to'.  They can be the same.
 */
```

```
extern void des2key(unsigned char *, short);
/*                          hexkey[16]     MODE
 * Sets the internal key registerS according to the hexadecimal
 * keyS contained in the 16 bytes of hexkey, according to the DES,
 * for DOUBLE encryption or decryption according to MODE.
 * NOTE: this clobbers all three key registers!
 */

extern void Ddes(unsigned char *, unsigned char *);
/*                          from[8]               to[8]
 * Encrypts/Decrypts (according to the keyS currently loaded in the
 * internal key registerS) one block of eight bytes at address 'from'
 * into the block at address 'to'.  They can be the same.
 */
bool TFCA_3DES(bool bEnspot, unsigned char* pbyKey,int nLength, unsigned char*
pbySource, unsigned char* pbyTarget)
bool TFCA_DES(bool bEnspot, unsigned char* pbyKey,int nLength, unsigned char*
pbySource, unsigned char* pbyTarget)
/*
TFCA_3DES进行3DES加密，TFCA_DES进行DES加密。参数含义如下：
bEnspot：true表示加密，false表示解密
pbyKey：指向进行加密的密钥
nLength：需要进行加密的数据长度，以字节为单位，需要为8的倍数
pbySource：指向需要进行加密的数据首指针
pbyTarget：指向返回的加密后的数据首指针
*/

#ifdef  __cplusplus
}
#endif
```

2. d3des.c

```
/* D3DES (V5.09) -
 *
 * A portable, public domain, version of the Data Encryption Standard.
 *
 * Written with Symantec's THINK (Lightspeed) C by Richard Outerbridge.
 * Thanks to: Dan Hoey for his excellent Initial and Inverse permutation
 * code;  Jim Gillogly & Phil Karn for the DES key schedule code; Dennis
 * Ferguson, Eric Young and Dana How for comparing notes; and Ray Lau,
 * for humouring me on.
 *
 * Copyright (c) 1988,1989,1990,1991,1992 by Richard Outerbridge.
 * (GEnie : OUTER; CIS : [71755,204]) Graven Imagery, 1992.
 */

#include "d3des.h"

static void scrunch(unsigned char *, unsigned long *);
static void unscrun(unsigned long *, unsigned char *);
static void desfunc(unsigned long *, unsigned long *);
static void cookey(unsigned long *);
```

```
static unsigned long KnL[32] = { 0L };
static unsigned long KnR[32] = { 0L };
static unsigned long Kn3[32] = { 0L };
static unsigned char Df_Key[24] = {
        0x01,0x23,0x45,0x67,0x89,0xab,0xcd,0xef,
        0xfe,0xdc,0xba,0x98,0x76,0x54,0x32,0x10,
        0x89,0xab,0xcd,0xef,0x01,0x23,0x45,0x67 };

static unsigned short bytebit[8]    = {
        0200, 0100, 040, 020, 010, 04, 02, 01 };

static unsigned long bigbyte[24] = {
        0x800000L,      0x400000L,      0x200000L,      0x100000L,
        0x80000L,       0x40000L,       0x20000L,       0x10000L,
        0x8000L,        0x4000L,        0x2000L,        0x1000L,
        0x800L,         0x400L,         0x200L,         0x100L,
        0x80L,          0x40L,          0x20L,          0x10L,
        0x8L,           0x4L,           0x2L,           0x1L    };

/* Use the key schedule specified in the Standard (ANSI X3.92-1981). */

static unsigned char pc1[56] = {
        56, 48, 40, 32, 24, 16,  8,  0, 57, 49, 41, 33, 25, 17,
         9,  1, 58, 50, 42, 34, 26,  18, 10,  2, 59, 51, 43, 35,
        62, 54, 46, 38, 30, 22, 14, 6, 61, 53, 45, 37, 29, 21,
        13,  5, 60, 52, 44, 36, 28, 20, 12,  4, 27, 19, 11,  3 };

static unsigned char totrot[16] = {
        1,2,4,6,8,10,12,14,15,17,19,21,23,25,27,28 };

static unsigned char pc2[48] = {
        13, 16, 10, 23,  0,  4,  2, 27, 14,  5, 20,  9,
        22, 18, 11,  3, 25,  7, 15,  6, 26, 19, 12,  1,
        40, 51, 30, 36, 46, 54, 29, 39, 50, 44, 32, 47,
        43, 48, 38, 55, 33, 52, 45, 41, 49, 35, 28, 31 };

void deskey(key, edf)       /* Thanks to James Gillogly & Phil Karn! */
unsigned char *key;
short edf;
{
        register int i, j, l, m, n;
        unsigned char pc1m[56], pcr[56];
        unsigned long kn[32];

        for ( j = 0; j < 56; j++ ) {
                l = pc1[j];
                m = l & 07;
                pc1m[j] = (key[l >> 3] & bytebit[m]) ? 1 : 0;
                }
        for( i = 0; i < 16; i++ ) {
                if( edf == DE1 ) m = (15 - i) << 1;
                else m = i << 1;
                n = m + 1;
                kn[m] = kn[n] = 0L;
                for( j = 0; j < 28; j++ ) {
                        l = j + totrot[i];
```

```
                        if( l < 28 ) pcr[j] = pc1m[l];
                        else pcr[j] = pc1m[l - 28];
                        }
                for( j = 28; j < 56; j++ ) {
                   l = j + totrot[i];
                   if( l < 56 ) pcr[j] = pc1m[l];
                   else pcr[j] = pc1m[l - 28];
                   }
                for( j = 0; j < 24; j++ ) {
                        if( pcr[pc2[j]] ) kn[m] |= bigbyte[j];
                        if( pcr[pc2[j+24]] ) kn[n] |= bigbyte[j];
                        }
                }
        cookey(kn);
        return;
        }

static void cookey(raw1)
register unsigned long *raw1;
{
        register unsigned long *cook, *raw0;
        unsigned long dough[32];
        register int i;

        cook = dough;
        for( i = 0; i < 16; i++, raw1++ ) {
                raw0 = raw1++;
                *cook    = (*raw0 & 0x00fc0000L) << 6;
                *cook   |= (*raw0 & 0x00000fc0L) << 10;
                *cook   |= (*raw1 & 0x00fc0000L) >> 10;
                *cook++ |= (*raw1 & 0x00000fc0L) >> 6;
                *cook    = (*raw0 & 0x0003f000L) << 12;
                *cook   |= (*raw0 & 0x0000003fL) << 16;
                *cook   |= (*raw1 & 0x0003f000L) >> 4;
                *cook++ |= (*raw1 & 0x0000003fL);
                }
        usekey(dough);
        return;
        }

void cpkey(into)
register unsigned long *into;
{
        register unsigned long *from, *endp;

        from = KnL, endp = &KnL[32];
        while( from < endp ) *into++ = *from++;
        return;
        }

void usekey(from)
register unsigned long *from;
{
        register unsigned long *to, *endp;

        to = KnL, endp = &KnL[32];
        while( to < endp ) *to++ = *from++;
```

```
        return;
        }

void des(inblock, outblock)
unsigned char *inblock, *outblock;
{
        unsigned long work[2];

        scrunch(inblock, work);
        desfunc(work, KnL);
        unscrun(work, outblock);
        return;
        }

static void scrunch(outof, into)
register unsigned char *outof;
register unsigned long *into;
{
        *into      = (*outof++ & 0xffL) << 24;
        *into     |= (*outof++ & 0xffL) << 16;
        *into     |= (*outof++ & 0xffL) << 8;
        *into++ |= (*outof++ & 0xffL);
        *into      = (*outof++ & 0xffL) << 24;
        *into     |= (*outof++ & 0xffL) << 16;
        *into     |= (*outof++ & 0xffL) << 8;
        *into     |= (*outof   & 0xffL);
        return;
        }

static void unscrun(outof, into)
register unsigned long *outof;
register unsigned char *into;
{
        *into++ = (*outof >> 24) & 0xffL;
        *into++ = (*outof >> 16) & 0xffL;
        *into++ = (*outof >>  8) & 0xffL;
        *into++ =  *outof++      & 0xffL;
        *into++ = (*outof >> 24) & 0xffL;
        *into++ = (*outof >> 16) & 0xffL;
        *into++ = (*outof >>  8) & 0xffL;
        *into     =  *outof      & 0xffL;
        return;
        }

static unsigned long SP1[64] = {
        0x01010400L, 0x00000000L, 0x00010000L, 0x01010404L,
        0x01010004L, 0x00010404L, 0x00000004L, 0x00010000L,
        0x00000400L, 0x01010400L, 0x01010404L, 0x00000400L,
        0x01000404L, 0x01010004L, 0x01000000L, 0x00000004L,
        0x00000404L, 0x01000400L, 0x01000400L, 0x00010400L,
        0x00010400L, 0x01010000L, 0x01010000L, 0x01000404L,
        0x00010004L, 0x01000004L, 0x01000004L, 0x00010004L,
        0x00000000L, 0x00000404L, 0x00010404L, 0x01000000L,
        0x00010000L, 0x01010404L, 0x00000004L, 0x01010000L,
        0x01010400L, 0x01000000L, 0x01000000L, 0x00000400L,
        0x01010004L, 0x00010000L, 0x00010400L, 0x01000004L,
        0x00000400L, 0x00000004L, 0x01000404L, 0x00010404L,
```

```
        0x01010404L, 0x00010004L, 0x01010000L, 0x01000404L,
        0x01000004L, 0x00000404L, 0x00010404L, 0x01010400L,
        0x00000404L, 0x01000400L, 0x01000400L, 0x00000000L,
        0x00010004L, 0x00010400L, 0x00000000L, 0x01010004L };

static unsigned long SP2[64] = {
        0x80108020L, 0x80008000L, 0x00008000L, 0x00108020L,
        0x00100000L, 0x00000020L, 0x80100020L, 0x80008020L,
        0x80000020L, 0x80108020L, 0x80108000L, 0x80000000L,
        0x80008000L, 0x00100000L, 0x00000020L, 0x80100020L,
        0x00108000L, 0x00100020L, 0x80008020L, 0x00000000L,
        0x80000000L, 0x00008000L, 0x00108020L, 0x80100000L,
        0x00100020L, 0x80000020L, 0x00000000L, 0x00108000L,
        0x00008020L, 0x80108000L, 0x80100000L, 0x00008020L,
        0x00000000L, 0x00108020L, 0x80100020L, 0x00100000L,
        0x80008020L, 0x80100000L, 0x80108000L, 0x00008000L,
        0x80100000L, 0x80008000L, 0x00000020L, 0x80108020L,
        0x00108020L, 0x00000020L, 0x00008000L, 0x80000000L,
        0x00008020L, 0x80108000L, 0x00100000L, 0x80000020L,
        0x00100020L, 0x80008020L, 0x80000020L, 0x00100020L,
        0x00108000L, 0x00000000L, 0x80008000L, 0x00008020L,
        0x80000000L, 0x80100020L, 0x80108020L, 0x00108000L };

static unsigned long SP3[64] = {
        0x00000208L, 0x08020200L, 0x00000000L, 0x08020008L,
        0x08000200L, 0x00000000L, 0x00020208L, 0x08000200L,
        0x00020008L, 0x08000008L, 0x08000008L, 0x00020000L,
        0x08020208L, 0x00020008L, 0x08020000L, 0x00000208L,
        0x08000000L, 0x00000008L, 0x08020200L, 0x00000200L,
        0x00020200L, 0x08020000L, 0x08020008L, 0x00020208L,
        0x08000208L, 0x00020200L, 0x00020000L, 0x08000208L,
        0x00000008L, 0x08020208L, 0x00000200L, 0x08000000L,
        0x08020200L, 0x08000000L, 0x00020008L, 0x00000208L,
        0x00020000L, 0x08020200L, 0x08000200L, 0x00000000L,
        0x00000200L, 0x00020008L, 0x08020208L, 0x08000200L,
        0x08000008L, 0x00000200L, 0x00000000L, 0x08020008L,
        0x08000208L, 0x00020000L, 0x08000000L, 0x08020208L,
        0x00000008L, 0x00020208L, 0x00020200L, 0x08000008L,
        0x08020000L, 0x08000208L, 0x00000208L, 0x08020000L,
        0x00020208L, 0x00000008L, 0x08020008L, 0x00020200L };

static unsigned long SP4[64] = {
        0x00802001L, 0x00002081L, 0x00002081L, 0x00000080L,
        0x00802080L, 0x00800081L, 0x00800001L, 0x00002001L,
        0x00000000L, 0x00802000L, 0x00802000L, 0x00802081L,
        0x00000081L, 0x00000000L, 0x00800080L, 0x00800001L,
        0x00000001L, 0x00002000L, 0x00800000L, 0x00802001L,
        0x00000080L, 0x00800000L, 0x00002001L, 0x00002080L,
        0x00800081L, 0x00000001L, 0x00002080L, 0x00800080L,
        0x00002000L, 0x00802080L, 0x00802081L, 0x00000081L,
        0x00800080L, 0x00800001L, 0x00802000L, 0x00802081L,
        0x00000081L, 0x00000000L, 0x00000000L, 0x00802000L,
        0x00002080L, 0x00800080L, 0x00800081L, 0x00000001L,
        0x00802001L, 0x00002081L, 0x00002081L, 0x00000080L,
        0x00802081L, 0x00000081L, 0x00000001L, 0x00002000L,
        0x00800001L, 0x00002001L, 0x00802080L, 0x00800081L,
        0x00002001L, 0x00002080L, 0x00800000L, 0x00802001L,
```

```
        0x00000080L, 0x00800000L, 0x00002000L, 0x00802080L };

static unsigned long SP5[64] = {
        0x00000100L, 0x02080100L, 0x02080000L, 0x42000100L,
        0x00080000L, 0x00000100L, 0x40000000L, 0x02080000L,
        0x40080100L, 0x00080000L, 0x02000100L, 0x40080100L,
        0x42000100L, 0x42080000L, 0x00080100L, 0x40000000L,
        0x02000000L, 0x40080000L, 0x40080000L, 0x00000000L,
        0x40000100L, 0x42080100L, 0x42080100L, 0x02000100L,
        0x42080000L, 0x40000100L, 0x00000000L, 0x42000000L,
        0x02080100L, 0x02000000L, 0x42000000L, 0x00080100L,
        0x00080000L, 0x42000100L, 0x00000100L, 0x02000000L,
        0x40000000L, 0x02080000L, 0x42000100L, 0x40080100L,
        0x02000100L, 0x40000000L, 0x42080000L, 0x02080100L,
        0x40080100L, 0x00000100L, 0x02000000L, 0x42080000L,
        0x42080100L, 0x00080100L, 0x42000000L, 0x42080100L,
        0x02080000L, 0x00000000L, 0x40080000L, 0x42000000L,
        0x00080100L, 0x02000100L, 0x40000100L, 0x00080000L,
        0x00000000L, 0x40080000L, 0x02080100L, 0x40000100L };

static unsigned long SP6[64] = {
        0x20000010L, 0x20400000L, 0x00004000L, 0x20404010L,
        0x20400000L, 0x00000010L, 0x20404010L, 0x00400000L,
        0x20004000L, 0x00404010L, 0x00400000L, 0x20000010L,
        0x00400010L, 0x20004000L, 0x20000000L, 0x00004010L,
        0x00000000L, 0x00400010L, 0x20004010L, 0x00004000L,
        0x00404000L, 0x20004010L, 0x00000010L, 0x20400010L,
        0x20400010L, 0x00000000L, 0x00404010L, 0x20404000L,
        0x00004010L, 0x00404000L, 0x20404000L, 0x20000000L,
        0x20004000L, 0x00000010L, 0x20400010L, 0x00404000L,
        0x20404010L, 0x00400000L, 0x00004010L, 0x20000010L,
        0x00400000L, 0x20004000L, 0x20000000L, 0x00004010L,
        0x20000010L, 0x20404010L, 0x00404000L, 0x20400000L,
        0x00404010L, 0x20404000L, 0x00000000L, 0x20400010L,
        0x00000010L, 0x00004000L, 0x20400000L, 0x00404010L,
        0x00004000L, 0x00400010L, 0x20004010L, 0x00000000L,
        0x20404000L, 0x20000000L, 0x00400010L, 0x20004010L };

static unsigned long SP7[64] = {
        0x00200000L, 0x04200002L, 0x04000802L, 0x00000000L,
        0x00000800L, 0x04000802L, 0x00200802L, 0x04200800L,
        0x04200802L, 0x00200000L, 0x00000000L, 0x04000002L,
        0x00000002L, 0x04000000L, 0x04200002L, 0x00000802L,
        0x04000800L, 0x00200802L, 0x00200002L, 0x04000800L,
        0x04000002L, 0x04200000L, 0x04200800L, 0x00200002L,
        0x04200000L, 0x00000800L, 0x00000802L, 0x04200802L,
        0x00200800L, 0x00000002L, 0x04000000L, 0x00200800L,
        0x04000000L, 0x00200800L, 0x00200000L, 0x04000802L,
        0x04000802L, 0x04200002L, 0x04200002L, 0x00000002L,
        0x00200002L, 0x04000000L, 0x04000800L, 0x00200000L,
        0x04200800L, 0x00000802L, 0x00200802L, 0x04200800L,
        0x00000802L, 0x04000002L, 0x04200802L, 0x04200000L,
        0x00200800L, 0x00000000L, 0x00000002L, 0x04200802L,
        0x00000000L, 0x00200802L, 0x04200000L, 0x00000800L,
        0x04000002L, 0x04000800L, 0x00000800L, 0x00200002L };

static unsigned long SP8[64] = {
```

```
        0x10001040L, 0x00001000L, 0x00040000L, 0x10041040L,
        0x10000000L, 0x10001040L, 0x00000040L, 0x10000000L,
        0x00040040L, 0x10040000L, 0x10041040L, 0x00041000L,
        0x10041000L, 0x00041040L, 0x00001000L, 0x00000040L,
        0x10040000L, 0x10000040L, 0x10001000L, 0x00001040L,
        0x00041000L, 0x00040040L, 0x10040040L, 0x10041000L,
        0x00001040L, 0x00000000L, 0x00000000L, 0x10040040L,
        0x10000040L, 0x10001000L, 0x00041040L, 0x00040000L,
        0x00041040L, 0x00040000L, 0x10041000L, 0x00001000L,
        0x00000040L, 0x10040040L, 0x00001000L, 0x00041040L,
        0x10001000L, 0x00000040L, 0x10000040L, 0x10040000L,
        0x10040040L, 0x10000000L, 0x00040000L, 0x10001040L,
        0x00000000L, 0x10041040L, 0x00040040L, 0x10000040L,
        0x10040000L, 0x10001000L, 0x10001040L, 0x00000000L,
        0x10041040L, 0x00041000L, 0x00041000L, 0x00001040L,
        0x00001040L, 0x00040040L, 0x10000000L, 0x10041000L };

static void desfunc(block, keys)
register unsigned long *block, *keys;
{
        register unsigned long fval, work, right, leftt;
        register int round;

        leftt = block[0];
        right = block[1];
        work = ((leftt >> 4) ^ right) & 0x0f0f0f0fL;
        right ^= work;
        leftt ^= (work << 4);
        work = ((leftt >> 16) ^ right) & 0x0000ffffL;
        right ^= work;
        leftt ^= (work << 16);
        work = ((right >> 2) ^ leftt) & 0x33333333L;
        leftt ^= work;
        right ^= (work << 2);
        work = ((right >> 8) ^ leftt) & 0x00ff00ffL;
        leftt ^= work;
        right ^= (work << 8);
        right = ((right << 1) | ((right >> 31) & 1L)) & 0xffffffffL;
        work = (leftt ^ right) & 0xaaaaaaaaL;
        leftt ^= work;
        right ^= work;
        leftt = ((leftt << 1) | ((leftt >> 31) & 1L)) & 0xffffffffL;

        for( round = 0; round < 8; round++ ) {
                work  = (right << 28) | (right >> 4);
                work ^= *keys++;
                fval  = SP7[ work                & 0x3fL];
                fval |= SP5[(work >>  8) & 0x3fL];
                fval |= SP3[(work >> 16) & 0x3fL];
                fval |= SP1[(work >> 24) & 0x3fL];
                work  = right ^ *keys++;
                fval |= SP8[ work                & 0x3fL];
                fval |= SP6[(work >>  8) & 0x3fL];
                fval |= SP4[(work >> 16) & 0x3fL];
                fval |= SP2[(work >> 24) & 0x3fL];
                leftt ^= fval;
                work  = (leftt << 28) | (leftt >> 4);
```

```
                work ^= *keys++;
                fval  = SP7[ work                & 0x3fL];
                fval |= SP5[(work >>  8) & 0x3fL];
                fval |= SP3[(work >> 16) & 0x3fL];
                fval |= SP1[(work >> 24) & 0x3fL];
                work  = leftt ^ *keys++;
                fval |= SP8[ work                & 0x3fL];
                fval |= SP6[(work >>  8) & 0x3fL];
                fval |= SP4[(work >> 16) & 0x3fL];
                fval |= SP2[(work >> 24) & 0x3fL];
                right ^= fval;
                }

        right = (right << 31) | (right >> 1);
        work = (leftt ^ right) & 0xaaaaaaaaL;
        leftt ^= work;
        right ^= work;
        leftt = (leftt << 31) | (leftt >> 1);
        work = ((leftt >> 8) ^ right) & 0x00ff00ffL;
        right ^= work;
        leftt ^= (work << 8);
        work = ((leftt >> 2) ^ right) & 0x33333333L;
        right ^= work;
        leftt ^= (work << 2);
        work = ((right >> 16) ^ leftt) & 0x0000ffffL;
        leftt ^= work;
        right ^= (work << 16);
        work = ((right >> 4) ^ leftt) & 0x0f0f0f0fL;
        leftt ^= work;
        right ^= (work << 4);
        *block++ = right;
        *block = leftt;
        return;
        }

void des2key(hexkey, mode)              /* stomps on Kn3 too */
unsigned char *hexkey;                  /* unsigned char[16] */
short mode;
{
        short revmod;

        revmod = (mode == EN0) ? DE1 : EN0;
        deskey(&hexkey[8], revmod);
        cpkey(KnR);
        deskey(hexkey, mode);
        cpkey(Kn3);                                     /* Kn3 = KnL */
        return;
        }

void Ddes(from, into)
unsigned char *from, *into;             /* unsigned char[8] */
{
        unsigned long work[2];

        scrunch(from, work);
        desfunc(work, KnL);
        desfunc(work, KnR);
```

```
        desfunc(work, Kn3);
        unscrun(work, into);
        return;
        }

bool TFCA_3DES(bool bEnspot, unsigned char* pbyKey,int nLength, unsigned char*
pbySource, unsigned char* pbyTarget)
{
       if(nLength==0 || nLength%8)
              return false;
       int section = nLength/8;
       des2key(pbyKey,bEnspot?0:1);
       for(int i = 0 ;i<section;i++)
              Ddes(pbySource+8*i,pbyTarget+8*i);
       return true;
}

bool TFCA_DES(bool bEnspot, unsigned char* pbyKey,int nLength, unsigned char*
pbySource, unsigned char* pbyTarget)
{
       if(nLength==0 || nLength%8)
              return false;
       int section = nLength/8;
       deskey(pbyKey,bEnspot?0:1);
       for(int i = 0 ;i<section;i++)
              des(pbySource+8*i,pbyTarget+8*i);;
       return true;
}

/* Validation sets:
 *
 * Single-length key, single-length plaintext -
 * Key     : 0123 4567 89ab cdef
 * Plain   : 0123 4567 89ab cde7
 * Cipher : c957 4425 6a5e d31d
 *
 * Double-length key, single-length plaintext -
 * Key     : 0123 4567 89ab cdef fedc ba98 7654 3210
 * Plain   : 0123 4567 89ab cde7
 * Cipher : 7f1d 0a77 826b 8aff
 * d3des V5.0a rwo 9208.07 18:44 Graven Imagery
 ********************************************************************/
```