基于 CRT 的证书状态服务的研究与实现*

薛 源[1,2]，郭建锋[1,3]，倪惜珍[2]

(1. 中国科学院 研究生院，北京 100039；2. 中国科学院 软件研究所 信息安全技术工程研究中心，北京 100080；3. 中国科学院 计算技术研究所，北京 100080)

摘 要：给出了 CRT 系统实现的逻辑结构模型；基于 ASN.1 的语法提出了一种证书状态证据(CRT-based Proof of Certificate Status, CRT-PCS)请求协议，能够有效地实现用户 - 目录之间的证书 CRT-PCS 请求服务，具有通用、高效的优点。针对协议实现的技术细节，介绍了利用 OpenSSL 实现 ASN.1 的 DER 编解码方法。

关键词：证书状态服务；证书撤销树；ASN.1；OpenSSL

中图法分类号：TP393.08 **文献标识码**：A **文章编号**：1001-3695(2005)02-0096-04

Research and Implementation of CRT-based Certificate Status Service

XUE Yuan[1,2], GUO Jian-feng[1,3], NI Xi-zhen[2]

(1. *School of Graduate, Chinese Academy of Sciences, Beijing* 100039, *China*; 2. *Engineering Research Center for Information Security Technology, Institute of Software, Chinese Academy of Sciences, Beijing* 100080, *China*; 3. *Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100080, *China*)

Abstract: Present a logical model of CRT system, and propose a CRT Proof of Certificate Status (CRT-PCS) request protocol based on ASN.1, which can efficiently realize the CRT-PCS service between directory and end entity. This protocol is also general and efficient. To deal with the technical problems of protocol implementation, it introduces how to do the AS N.1 DER encoding and decoding in utilizing the OpenSSL.

Key words: Certificate Status Service; Certificate Revocation Tree; ASN.1; OpenSSL

公钥基础设施(Public Key Infrastructure, PKI)作为解决网络环境中安全问题的技术，在过去十多年间得到了迅速的发展以及广泛的关注。在 PKI 系统中，CA(Certification Authority)是一个域中的信任中心，终端实体之间的安全通信依赖于 CA 所颁发的证书。证书具有一个有效期限，当证书过期后，该证书将不能正常使用；然而，在有效期范围内，可能由于密钥泄漏、从属关系改变等原因，需要撤销该证书。

在 PKI 系统中，证书撤销机制的选择和实施直接影响着应用系统的效率[1]。ITU-T(前身 CCITT)在 1988 年提出的证书撤销列表 CRL(Certificate Revocation List)[2]是一种广泛使用的证书撤销机制。该方案的优点是实现简单，但随着系统的使用，CRL 会不断增大，加重 CA - 目录、目录 - 用户间的通信负担，另外 CRL 有限的及时性也不能满足高实时性安全应用的需要。

证书撤销树 CRT(Certificate Revocation Trees)[3,4]是一种高效的证书撤销机制，由 Paul Kocher 在 1998 年基于二叉杂凑树提出。相比 CRL，CRT 有简短的证书状态证据(Proof of Certificate Status)，用户 - 目录间的通信量较小。当前对 CRT 的研究大多集中在理论模型上，相关实现的研究很少，其应用还未进行标准化。

1 证书撤销树 CRT

为了叙述的方便，做如下一些符号的约定[6]。设证书的序列号是有序的，Low 和 High 分别为证书序列号的下界和上界，也就是对于系统中任意的证书序列号 i，满足 $Low < i < High$。证书序列号为 i 的证书用 C_i 表示，数对 (j, k) 表示证书 C_j 和 C_k 被撤销，而对任意 $m(j < m < k)$，C_m 未被撤销。设被撤销的证书总数为 N，则序列号空间被分为 $N+1$ 个区域，用数据结构 $L_0, L_1, \cdots, L_N$ 表示这些区域，$L_i(0 \leq i \leq N)$ 中除记录数对 (j, k) 外，还可包含证书的撤销时间和原因等附加信息。二叉杂凑树的叶子节点 $N_{0,i} = H(L_i)$，其中 H 是一个杂凑函数。为方便讨论，下面将假设 $N+1$ 是 2 的幂次，这样得到的树将是一个完全二叉树，树的高度 $r = \log_2(N+1)$。

用 N_{ij} 表示树的节点，其中 i, j 满足 $0 \leq i \leq r$，$0 \leq j < (N+1)/2^i$，如前所定义，$N_{0,j}$ 表示叶子节点，$N_{1,j}$ 表示叶子节点的父节点，直到根节点 $N_{r,0}$。每一个内部节点的值都是通过它的儿子节点的值计算得出，节点 $N_{i,j}(1 \leq i \leq r,\ 0 \leq i < (N+1)/2^i)$ 的左右儿子分别为 $N_{i-1,2j}$ 和 $N_{i-1,2j+1}$，$N_{i,j} = H(N_{i-1,2j} \| N_{i-1,2j+1})$(符号 ‖ 表示连接)。根节点 $N_{r,0}$ 连同其他一些信息(如 CRT 的有效期)由 CRT 签发者签名。最后整棵树和根节点签名将会被发布到目录服务器。

用户在使用时，向目录服务器发送一个含有待验证证书序列号的请求。服务返回的数据包括：对应序列号的 L_k，从 $N_{0,k}$ 到根节点路径上所有节点的兄弟节点的值，以及根节点及其签名。把验证证书状态所需的这些数据统称为基于 CRT 的证书状态证据(CRT-based Proof of Certificate Status)，下面用 CRT-PCS 来表示。用户根据收到的 CRT-PCS 重新计算根节点，判断是否与收到的根节点值相同，并验证其签名。签名验证成功后，查看 L_k 的内容来判断待验证证书是否被撤销，如果待验证

收稿日期：2004-03-26；修返日期：2004-05-19

基金项目：国家重点基础研究发展规划基金资助项目(G1999035810)；国家自然科学基金资助项目(60083007)

证书是 L_k 的边界, 则被撤销; 否则未被撤销。

图 1 是一个 CRT 的例子, 其中 $N = 7$, 被撤销证书的序列号依次为 3, 4, 7, 10, 13, 17 和 21。其中 $N_{1,1} = H(N_{0,2} \| N_{0,3})$, 而 $N_{0,2} = H(L_2)$, $N_{0,2} = H(L_3)$。验证证书序列号为 15 的证书的状态需要的节点信息, 也就是其 CRT-PCS 为 L_5, $N_{0,4}$, $N_{1,3}$, $N_{2,0}$ 以及签名的根节点 $N_{3,0}$。

CRT 具有短的证书状态证据 CRT-PCS, 因此有较小的用户 - 目录通信复杂度, 能够提供较高的及时性。

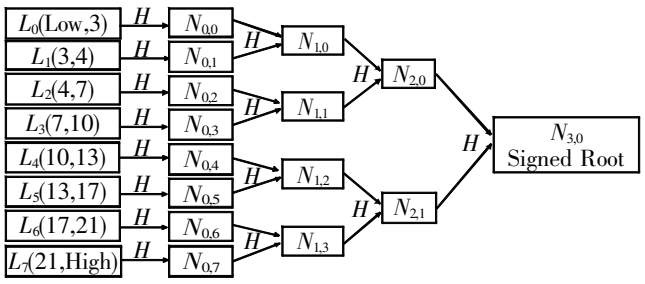

图 1　CRT 的一个例子

2　CRT 系统的逻辑结构

CRT 实现的逻辑结构(图 2) 有如下几部分:

(1) CRT 发布者。向目录服务控制器发布 CRT 的一方, 一般是 CA, 或者 CA 授权的 CRT 签发者。它所发布的消息通常包括增加(或删除) 叶子节点的信息, CRT 根节点信息及其签名值。

(2) 目录服务控制器。维护目录服务器, 根据收到的 CRT 更新信息和目录服务器中已有的 CRT 构造出与 CRT 签发者一致的二叉杂凑树, 并将最新的 CRT 存入目录服务器。

(3) 目录服务器。方便快速的数据库系统, 一般采用 LDAP, 用于存取当前最新的 CRT。

(4) CRT-PCS 响应器。处理客户端的证书状态查询请求, 从目录服务器的最新 CRT 中抽取出对应证书的 CRT-PCS, 作为应答返回给客户端。

(5) 客户端应用程序。依赖于证书的应用程序。

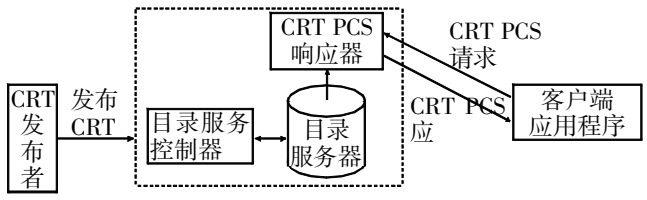

图 2　CRT 实现的逻辑框图

上面提到的前四个组件, 每一个采用 CRT 作为证书撤销机制的 PKI 系统都可以有自己的实现方式和专有的数据流设计; 但是 CRT-PCS 响应器与客户端应用程序间的数据流必须具有通用性, 并且进行标准化, 使客户端应用程序能以统一的方式访问各个实现 CRT 的 PKI 系统, 正确解析 PCS, 以保证各个 PKI 系统的证书在大型网络环境中的有效使用。本文不讨论特定于每个 CRT 系统中的实现细节, 而是设计一种 CRT-PCS 响应器与客户端应用程序间的通用的服务请求协议, 称之为 CRT-PCS 请求协议。

3　CRT-PCS 请求协议

CRT-PCS 请求协议定义了客户端应用程序与 CRT-PCS 响应器之间所交换数据的格式, 包括请求消息和应答消息。消息格式采用 ASN.1 定义, 定义中参考了 IETF PKIX 工作组定义的 OCSP[7] 的消息格式和 ITU-T 定义的 CRL[2] 的消息格式。另外在消息的定义中, 会用到一些从相关标准(如 RFC2459[2]) 中导入的类型, 如 GeneralName, Extensions, AlgorithmIdentifier, Certificate, CertificateSerialNumber, GeneralizedTime, CRLReason, Name 等, 对这些类型, 可以参看相应的 RFC。

3.1　请求消息格式

请求消息对应的类型为 CRTRequest, 其 ASN.1 表示如下:

```
CRTRequest: : = SEQUENCE {
tbsRequest          TBSRequest,
optionalSignature   [0] EXPLICIT Signature OPTIONAL }
TBSRequest: : = SEQUENCE {
requestorName       [0] EXPLICIT GeneralName OPTIONAL,
requestCertID       RequestCertID,
requestExtensions   [1] EXPLICIT Extensions OPTIONAL}
RequestCertID: : = SEQUENCE {
hashAlgorithm       AlgorithmIdentifier,
issuerNameHash      OCTET STRING,
issuerKeyHash       OCTET STRING,
serialNumber        CertificateSerialNumber }
Signature: : = SEQUENCE {
signatureAlgorithmAlgorithmIdentifier,
signature           BIT STRING,
certs               [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL }
```

请求者可以选择对消息进行签名, 签名是对结构 tbsRequest 进行, Signature 结构中的 SignatureAlgorithm 和 Signature 分别表示签名算法和签名值。如果请求消息被签名, 则请求者应在 TBSRequest 的 RequestorName 域中放置自己的名字, 以使响应器能对签名进行验证。另外请求者也可在 Signature 的 Certs 域中放入相关的证书以方便响应器验证签名。

TBSRequest 的 RequestExtensions 域可用来放置扩展信息, 目前未定义任何扩展, 扩展可根据实际应用的需要加入。

结构 RequestCertID 表示待验证的证书, 其中 issuerNameHash 是证书签发者 Name DER 编码的杂凑值, 而 issuerKeyHash 是证书签发者公钥的杂凑值。HashAlgorithm 用来表示所使用的杂凑算法。SerialNumber 是待验证证书的序列号。

3.2　应答消息格式

应答消息对应的类型为 CRTResponse, 其 ASN.1 表示如下:

```
CRTResponse: : = SEQUENCE {
responseStatus      CRTResponseStatus,
responseBytes       [0] EXPLICIT ResponseBytes OPTIONAL }
CRTResponseStatus: : = INTEGER {
successful          (0),
malformedRequest    (1),
internalError       (2),
tryLater            (3),
sigRequired         (4),
unauthorized        (5)}
ResponseBytes: : = SEQUENCE {
hashAlgorithm       AlgorithmIdentifier,
leafNode            LeafNode,
siblingList         [0] EXPLICIT SEQUENCE OF SiblingNode OPTIONAL,
root                CRTRoot }
LeafNode: : = SEQUENCE {
leftBoundCert       BoundCert,
rightBoundCert      BoundCert}
BoundCert: : = SEQUENCE {
certSerialNumber    CertificateSerialNumber,
revocationTime      GeneralizedTime,
revocationReason    [0] EXPLICIT CRLReason OPTIONAL,
extensions          [1] EXPLICIT Extensions OPTIONAL}
SiblingNode: : = SEQUENCE {
left                BOOL,
digestContent       OCTET STRING }
CRTRoot: : = SEQUENCE {
```

```
    crtRootInfo          CRTRootInfo,
    signatureAlgorithm   AlgorithmIdentifier,
    signatureValue       BIT STRING,
    certs                [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL}
    CRTRootInfo::= SEQUENCE {
    signatureAlgorithm   AlgorithmIdentifier,
    crtIssuer            Name,
    caName               Name,
    thisUpdate           GeneralizedTime,
    nextUpdate           [0] EXPLICIT GeneralizedTime OPTIONAL,
    rootDigestContent    OCTET STRING,
    crtExtensions        [1] EXPLICIT Extensions OPTIONAL }
```

应答消息中包含状态值,如果应答消息的 responseStatus 状态值不是 Successful,则应答消息中不应包括 responseBytes 结构。当处理出错时,返回相应的错误代码,malformedRequest 表示收到的请求消息格式不符合规范;internalError 表示响应器内部错误;如果服务存在,只是短暂地不能处理请求,则会返回 tryLater,以向用户指示可以稍后再试;sigRequired 表示响应器要求请求消息被签名;如果响应器执行访问控制,而当前的请求者不是授权用户,则会返回 Unauthorized。

响应消息 ResponseBytes 结构中包含的信息有:进行验证的杂凑算法 hashAlgorithm、叶子节点、验证路径上的节点链以及签名的根节点。三类节点分别用 LeafNode, SiblingNode 和 CRTRoot 表示。

叶子节点 LeafNode 中包括下界证书信息和上界证书信息,它们都由结构 BoundCert 表示。BoundCert 中包括表示证书序列号的 certSerialNumber,证书撤销时间的 revocationTime,可选的撤销原因 revocationReason 和扩展域。

SiblingNode 的定义比较简单,一个 BOOL 域 Left 用来表示该节点是其父节点的左儿子(值为 True 时)还是右儿子(值为 Flase 时),使用此值是由于杂凑函数对连接操作没有交换性,验证中计算杂凑值必须知道两个节点的先后顺序。digestContent 表示此内部节点的值。

CRTRoot 含有根节点的信息结构 CRTRootInfo 和对此结构签名的相关信息,如签名算法 signatureAlgorithm、签名值 signatureValue 以及辅助请求者验证签名的可选证书链 Certs。CRTRootInfo 中包含的信息有:签名算法 signatureAlgorithm,此算法必须同 CRTRoot 中的签名算法一致;CRT 的签发者 CrtIssuer;本 CRT 所对应的 CA 名 caName,caName 应是请求证书的签发者;同 CRL 以及 OCSP 中意义相同的 thisUpdate 和 nextUpdate 域,thisUpdate 表示该 CRT 被签发的时间,nextUpdate 表示新的 CRT 可得的时间;rootDigestContent 是根节点对应的杂凑值;还有可选的扩展域 crtExtensions。

3.3 说明

采用 ASN.1 定义消息格式,使该协议具有很好的通用性,可以用于任何以二叉树 CRT 作为证书撤销机制的 PKI 系统。用户在验证过程中,验证根节点签名和计算根节点的杂凑值可以同时进行。最后再将计算出的杂凑值同 CRTRootInfo 中的 rootDigestContent 相比较,加快验证速度。

应用程序在验证过程中,还应注意其他一些问题,如除验证根节点签名外,还需判断是否信任 CRT 签发者,CA 名是否对应,当前时间是否在 CRT 有效期内:nextUpdate 必须晚于当前系统时间,而 thisUpdate 必须早于当前系统时间。

协议只是定义了消息的格式,并没有指定消息的传输机制,实现时可以灵活选择。

4 CRT-PCS 响应器的实现

CRT-PCS 响应器的功能是接收并处理客户端 DER 编码的 CRT-PCS 请求消息,并将 DER 编码的 CRT-PCS 应答消息返回给客户端。在编程实现时,实际操作处理的是请求消息与应答消息的内部数据结构表示,请求消息 CRTRequest 对应于结构 struct CRTRequest_st{ },而应答消息 CRTResponse 对应于结构 struct CRTResponse_st{ }。称将消息由 DER 编码格式转换成内部表示的解码过程为 D2I(DER to Internal),相反的编码过程为 I2D(Internal to DER)。响应器的处理流程如图 3 所示。

图 3 响应器处理流程图

ASN.1 DER 编解码涉及众多原子数据类型和复杂的结构类型,对其完整的实现是非常琐碎而庞大的工作。所以 D2I 和 I2D 过程是 CRT-PCS 响应器实现的一个技术难点(此问题不只存在于本文的 CRT-PCS 响应器实现中,对于使用 ASN.1 定义消息格式的工程项目都会遇到)。

当前已经有 ASN.1 DER 编解码相关的工具和库,在安全领域使用最广的是 OpenSSL[8]。它在实现密码库的同时也提供了对 SSL 和 X.509 相关标准的支持,其中包括常用的基本安全数据类型(如第 3 节中提到的 Certificate, GeneralName, AlgorithmIdentifier 等导入类型) D2I 和 I2D 过程,基于此,在实现本协议时可以将上述类型视为基本类型。有效地利用 OpenSSL 能够方便地实现 ASN.1 定义的安全协议消息的 DER 编解码,下面将以本协议为例进行介绍。

4.1 OpenSSL 对 ASN.1 DER 编解码的支持

OpenSSL 为每种 ASN.1 的基本类型都定义了内部结构,如 INTEGER 对应于 ASN1_INTEGER, OCTET STRING 对应于 ASN1_OCTET_STRING。每种类型都有相应的空间分配和释放函数,DER 编码和解码函数,其中编码函数名以 I2D 开头,解码函数以 D2I 开头。如 ASN1_INTEGER 对应的四个函数为

```
ASN1_INTEGER*  ASN1_INTEGER_new(void);
void ASN1_INTEGER_free(ASN1_INTEGER*);
int i2d_ASN1_INTEGER(ASN1_INTEGER*, unsigned char**);
ASN1_INTEGER* d2i_ASN1_INTEGER(ASN1_INTEGER**, unsigned char**, long);
```

OpenSSL 还提供了大量的宏方便使用者实现自己的 DER 编解码函数,本文将在实现 CRT-PCS 请求协议的消息编解码中用到它们。

4.2 CRT-PCS 请求协议消息的 DER 编解码

实现时,消息的 ASN.1 定义中的每一个 SEQUENCE 类型都有相应的内部结构,每一个结构都实现了空间分配/释放函数,DER 编码/解码函数。下面给出 CRTResponse 和 ResponseBytes 的内部结构定义,其中 CRTResponse_st{ }中使用了 ResponseBytes_st{ },而 ResponseBytes_st{ }中使用了 LeafNode, SiblingNode, CRTRoot 三种自定义结构类型和一种 OpenSSL 已实现类型 X509_ALGOR。因为实现的相似性,在此略去了 Leaf-

Node 等结构的定义。下面仅以 CRTResponse 为例进行说明:

```
typedef struct CRTResponse_st{
ASN1_INTEGER      * responseStatus;
ResponseBytes     * responseBytes;
} CRTResponse;
typedef struct    ResponseBytes_st{
X509_ALGOR        * hashAlgorithm;
LeafNode          * leafNode;
STACK_OF( SiblingNode)    * siblingList;
CRTRoot           * root;
} ResponseBytes;
```

代码中以 M_ASN1 开头的标志符是 OpenSSL, 为方便实现 DER 编解码函数而提供的宏。编写分配空间函数常用的宏为 M_ASN1_New_Malloc, M_ASN1_New, M_ASN1_New_Error, 分别用于主结构空间分配、结构子域空间分配以及出错处理。

下面是 CRTResponse 的空间分配函数:

```
CRTResponse * CRTResponse_new( void) {
CRTResponse * ret = NULL;
ASN1_CTX c;
M_ASN1_New_Malloc( ret, CRTResponse) ;
/* 为结构 CRTResponse 的域 responseStatus 分配空间 * /
M_ASN1_New( ret - > responseStatus, ASN1_INTEGER_new) ;
/* responseBytes 在 ASN. 1 定义中是可选( OPTIONAL) 的, 故此处不为其分配空间 * /
ret - > responseBytes = NULL;
return ( ret) ;
M_ASN1_New_Error( CRTPCS_F_CRTResponse_NEW) ;
}
```

CRTResponse 的空间释放函数原型为 void CRTResponse_free(CRTResponse *) , 它首先释放子域 responseStatus 和 responseBytes, 然后释放自身的空间, 在此省略了它的代码。编写编码函数时, 首先调用宏 M_ASN1_I2D_vars 声明函数内部变量; 然后调用 M_ASN1_I2D_len 类的宏计算编码各个子域所需的空间; 若整个结构是 SEQUENCE 类型, 调用宏 M_ASN1_I2D_seq_total 处理 SEQUENCE 结构头部(Identifier octets + Length octets) ; 最后调用 M_ASN1_I2D_put 类的宏编码子域, 并调用宏 M_ASN1_I2D_finish 设置返回值。

CRTResponse 的编码函数如下, 参数中 a 为待编码的结构, pp 是指向存放编码串结果的指针的指针。若 pp 为 NULL, 不实际编码, 但返回编码所需的空间; 若 pp 不为 NULL, 将编码结果写入* pp 指向的空间, 更新* pp 指向编码串末尾的下一个位置。失败返回 0。

```
int i2d_CRTResponse( CRTResponse * a, unsigned char * * pp) {
int v0;
M_ASN1_I2D_vars( a) ;
M_ASN1_I2D_len( a - > responseStatus, i2d_ASN1_INTEGER) ;
/* 处理简单类型 ASN1_INTEGER * /
/* 处理可选的 EXPLICIT[ 0] 标记类型, 并将 responseBytes 的编码长度存入 v0, i2d_ResponeByte 是 ResponseBytes 的编码函数 * /
M_ASN1_I2D_len_EXP_opt( a - > responseBytes, i2d_ResponseBytes, 0, v0) ;
M_ASN1_I2D_seq_total( ) ;
M_ASN1_I2D_put( a - > responseStatus, i2d_ASN1_INTEGER) ;
/* 编码子域 responseStatus * /
M_ASN1_I2D_put_EXP_opt( a - > responseBytes, i2d_ResponseBytes, 0, v0) ; /* 编码子域 responseBytes* /
M_ASN1_I2D_finish( ) ;
}
```

编写解码函数时, 首先调用宏 M_ASN1_D2I_vars 和 M_ASN1_D2I_Init 声明并初始化函数内部变量; 如果整个结构是 SEQUENCE 类型, 调用宏 M_ASN1_D2I_start_sequence 处理结构的头部; 之后通过调用 M_ASN1_D2I_get 类的宏分别解码各个子域; 最后调用 M_ASN1_D2I_Finish 结束处理。

CRTResponse 的解码函数如下: a 是指向目标结构的指针的指针, pp 是指向待解码的 DER 串的指针的指针, Length 是串* pp 的长度。若 a 或* a 为 NULL, 会新创建结构 CRTResponse, 若* a 不为 NULL, 则解码结果放入* a 中。成功时, * pp会更新为* pp + Length。

```
CRTResponse * d2i_CRTResponse( CRTResponse * * a, unsigned char * * pp, long length) {
M_ASN1_D2I_vars( a, CRTResponse * , CRTResponse_new) ;
M_ASN1_D2I_Init( ) ;
/* 处理 SEQUENCE 对应的头( Identifier octets + Length octets) * /
M_ASN1_D2I_start_sequence( ) ;
M_ASN1_D2I_get( ret- > responseStatus, d2i_ASN1_INTEGER) ;
/* 解码子域 responseStatus * /
/* 处理可选的 EXPLICIT[ 0] 标记类型的子域 responseBytes, d2i_ResponseBytes 是 ResponseBytes 的解码函数 * /
if ( ( c. slen ! = 0) && ( M_ASN1_next = = ( V_ASN1_CONTEXT_SPECIFIC | V_ASN1_CONSRUCTED | 0) ) ) {
M_ASN1_D2I_get_EXP_opt( ret- > responseBytes, d2i_ResponseBytes, 0) ;
}
else {
if ( ret- > responseBytes ! = NULL) {
ResponseBytes_free( ret- > responseBytes) ;
ret- > responseBytes = NULL;
}
}
/* 检查长度、DER 编码的 EOS 字节等, 结束处理 * /
M_ASN1_D2I_Finish( a, CRTResponse_free, CRTPCS_F_D2I_CRTResponse) ;
}
```

5　结论

本文提出了 CRT-PCS 请求协议, 以解决实际应用中 CRT 证书状态证据请求的通用性问题, 对 CRT 的推广和标准化有一定的参考意义。同时给出了一种 CRT 系统的逻辑结构模型。通过介绍 CRT-PCS 响应器的实现, 说明了使用 OpenSSL 进行 ASN. 1 DER 编解码的方法, 这一方法也可用于其他使用 ASN. 1 定义的安全协议实现中。

参考文献:

[1]　erkovits S, Chokani S, Geiter J A, *et al*. Public Key Infrastructure Study: Final Report. MITRE Corporation for NIST[EB/ OL] . http: / / csrc. nist. gov/ pki/ documents/ mitre. ps, 1994-04.

[2]　R Housley, W Ford, W Polk, *et al*. Internet X. 509 Public Key Infrastructure, Certificate and CRL Profile[S] . RFC 2459, 1999.

[3]　P Kocher. On Certificate Revocation and Validation[C] . Proc. of Financial Cryptography, 1998. 172-177.

[4]　P Kocher. A Quick Introduction to Certificate Revocation Trees[EB/ OL] . http: / / www. valicert. com/ resources/ body. html, 1998.

[5]　Burton S, Kaliski Jr. A Layman s Guide to a Subset of ASN. 1, BER, and DER. An RSA Laboratories Technical Note[EB/ OL] . ftp: / / ftp. rsasecurity. com/ pub/ pkcs/ ps/ layman. ps, 1993-11-01.

[6]　Petra Wohlmacher. Digital Certificates: A Survey of Revocation Methods[C] . ACM Multimedia Workshops, 2000. 111-114.

[7]　M Myers, R Ankney, A Malpani, *et al*. 509 Internet Public Key Infrastructure, Online Certificate Status Protocol-OCSP [S] . RFC 2560, 1999.

[8]　http: / / www. openssl. org[EB/ OL] .

作者简介:

薛源(1979-) , 女, 黑龙江齐齐哈尔人, 硕士研究生, 主要研究领域为网络与信息安全理论与技术; 郭建锋(1977-) , 男, 山西太原人, 硕士研究生, 主要研究领域为信息安全; 倪惜珍(1947-) , 女, 研究员, 主要领域为网络与信息安全理论与技术。