

Apéndice A

Definición de las políticas de PKI

Este apéndice incluye la especificación ASN.1 [105] de los elementos de información que componen el sistema de políticas presentado en el capítulo 3. Parte de las definiciones están basadas en estructuras ya definidas en el estándar X.509 [106].

A.1 Estructura general de la política

Se compone de un número de serie, una fecha de emisión, una fecha de próxima emisión y un conjunto de elementos de política. A su vez, cada elemento de política incluye un campo que especifica el conjunto de entidades afectadas por dicho elemento (la especificación se realiza utilizando la estructura GeneralName, la cual abarca los Distinguished Names X.500), y uno o más valores relacionados con el parámetro que está siendo controlado.

```
PKIPolicy ::= SEQUENCE {
    serialNumber    INTEGER,
    thisUpdate      Time,
    nextUpdate      Time OPTIONAL,
    elements        SEQUENCE OF PolicyElement
}

PolicyElement ::= SEQUENCE {
    entities        SEQUENCE OF GeneralName OPTIONAL,
    rule            Rule
}

Rule ::= SEQUENCE {
    ruleID          RULE.&id({RuleSet}), --Identificador de la regla
    ruleValue       OCTET STRING        --Codificación DER de la regla
}
```

```
RuleSet RULE ::= {...}
```

A.2 Reglas de la política

KeyType. Tipo de clave que contendrá el certificado.

```
KeyType RULE ::= SEQUENCE OF OBJECT IDENTIFIER --DSA,RSA
```

RSALength. Longitud máxima y mínima que debe tener la clave RSA contenida en la solicitud.

```
RSALength RULE ::= SEQUENCE {
  minLength    INTEGER,
  maxLength    INTEGER
}
```

DSALength. Longitud máxima y mínima que debe tener la clave DSA contenida en la solicitud.

```
DSALength RULE ::= SEQUENCE {
  minLength    INTEGER,
  maxLength    INTEGER
}
```

AlternativeSubject. Especifica el tipo de nombres alternativos que pueden emplearse y los criterios que deben seguir.

```
AlternativeSubject RULE ::= SEQUENCE OF AltFilteredName
```

```
AltFilteredName ::= SEQUENCE {
  mandatory    BOOLEAN, --Indica si es obligatorio u opcional
  name         OBJECT IDENTIFIER, --Tipo de nombre
  filter       OCTET STRING OPTIONAL --Filtro a cumplir
}
```

UniqueIdentifier. Indica si es obligatoria la utilización de un campo de identificador único de usuario.

```
UniqueIdentifier RULE ::= BOOLEAN
```

CertNetscape. Extensiones de tipo Netscape que puede contener el certificado a generar.

`CertNetscape` RULE ::= SEQUENCE OF NetscapeCertType

KeyUsage. Usos que se le puede dar a la clave a certificar.

`KeyExtUsage` RULE ::= SEQUENCE OF KeyUsage

ValidityDates. Contiene información acerca del periodo de validez que tendrá el certificado a generar.

```
ValidityDates RULE ::= SEQUENCE {
    validityCA      INTEGER,  --Validez en caso de ser CA
    validityUser    INTEGER    --Validez en caso contrario
}
```

RenewalValidity. Contiene tres clases de información relacionada con la renovación de certificados. La primera de ella es el periodo a partir del cual se puede solicitar la renovación del certificado, es decir, el número mínimo de días que deben faltar para que el certificado caduque. El segundo tipo de información es el relativo al nuevo periodo de validez del certificado, es decir, el número de días por el cual será renovado. El tercero hace referencia al periodo máximo en días durante el cual una clave puede ser renovada.

```
RenewalValidity RULE ::= SEQUENCE {
    threshold      INTEGER,
    newInterval    INTEGER,
    maxInterval    INTEGER
}
```

CRLIssuance. Indica si la emisión de CRLs debe realizarse tras la notificación de una revocación, de forma periódica cada cierto número de días, o de ambas formas

```
CRLIssuance RULE ::= SEQUENCE {
    immediate      BOOLEAN DEFAULT TRUE,
    periodically   BOOLEAN DEFAULT TRUE,
    interval       INTEGER
}
```


Apéndice B

Estructuras de datos del protocolo AMBAR

Este apéndice contiene la especificación completa del protocolo AMBAR. La notación empleada está basada en el estándar de representación de datos XDR [143]. Las diferentes secciones están estructuradas en función de los distintos módulos del marco AMBAR, y contienen tanto la estructura de datos de los distintos mensajes AMBAR como los detalles relativos a la generación de los valores criptográficos.

B.1 Transport Convergence

TransportData

```
struct {
    ContentType type;
    uint32      length;
    opaque      data[AMBARMessage.length];
} AMBARMessage;
```

ContentType

```
enum {
    session_management(1), request_management(2),
    authorization_results_management(3), error_management(4),
    data_stream_management(5)
} ContentType;
```

Ciphertext

```

struct {
    select (CipherSpec.cipherType) {
        case null: GenericPlain;
        default:  GenericBlockCipher;
    } data;
} Ciphertext

```

GenericPlain

```

struct {
    opaque content[Ciphertext.length];
} GenericPlain;

```

GenericBlockCipher

```

cbc-block-ciphered struct {
    opaque content[];
    opaque MAC[HMAC_SIZE];
} GenericBlockCipher;

```

B.2 Error Management

ErrorLevel

```

enum {
    warning(1), fatal(2)
} ErrorLevel;

```

ErrorDescription

```

enum {
    invalid_signature(1), unknown_message(2), illegal_parameter(3),
    out_of_sequence(4), close_notify(5), sm_failure(6),
    bad_certificate(7), bad_credential(8), bad_calculation(9),
    no_common_suite(10), unspecified_error(11), ciphertext_error(12)
} ErrorDescription;

```

ErrorMessage

```
struct {
    ErrorLevel      level;
    ErrorDescription description;
    uint16          errorNumber;
} ErrorMessage;
```

B.3 Authorization Results Management

ResultType

```
enum {
    negative_notification(1), positive_notification(2), resource(3)
} ResultType;
```

AuthorizationResult

```
struct {
    ResultType  type;
    uint16      transaction_number;
    select (type) {
        case negative_notification: NegativeNotificationContent;
        case positive_notification: PositiveNotificationContent;
        case resource: ResourceContent;
    } content;
} AuthorizationResult;
```

NegativeNotificationContent

```
struct {
    uint8  notification_number;
    opaque notification<0..2^8>;
    bool   partial;
} NegativeNotificationContent;
```

PositiveNotificationContent

```
struct {
    opaque notification<0..2^8>;
} PositiveNotificationContent;
```

ResourceContent

```
struct {
    opaque resource<uint32>;
} ResourceContent;
```

B.4 Request Management

AuthorizationDataType

```
enum {
    request_and_asserts(1), asserts(2), policy(3), calculation(4)
} AuthorizationDatatype;
```

AuthorizationData

```
struct {
    AuthorizationDataType type;
    uint16 transaction_number;
    uint16 transaction_step;
    select (type) {
        requests_and_asserts: RequestAndAssertsContent;
        asserts:              AssertsContent;
        policy:                PolicyContent;
        calculation:          CalculationContent;
    } content;
} AuthorizationData;
```

AssertItem

```
struct {
    opaque assert<0..216>;
} AssertItem;
```

StreamFlag

```
enum {
    switch_stream (1), unrelated_stream (2)
} StreamFlag;
```

RequestAndAssertsContent

```
struct {
    StreamFlag  sflag;
    opaque      request<0..216>;
    AssertItem  asserts<0..216>;
} RequestAndAssertsContent;
```

AssertsContent

```
struct {
    AssertItem asserts<0..216>;
} AssertsContent;
```

PolicyContent

```
struct {
    opaque policy<0..216>;
} PolicyContent;
```

CalculationContent

```
struct {
    opaque calculation<0..216>;
} CalculationContent;
```

B.5 Data Stream Management

DSMMessage

```
struct {
    uint16  stream_id;
    uint32  sequence_number;
    opaque  stream<0..216>;
} DSMMessage;
```

B.6 Session Management

SMTtype

```
enum {
    client_init(1), server_init(2), pk_value(3), activate_crypto(4),
    init_session(5)
} SMTtype;
```

SMMessage

```
struct {
    SMTtype type;
    select (type) {
        case client_init:      ClientInit;
        case server_init:     ServerInit;
        case pk_value:        PKValue;
        case activate_crypto:  ActivateCrypto;
        case init_session:    InitSession;
    } content;
} SMMessage;
```

ClientInit

```
struct {
    ProtocolVersion version;
    Random          nonce;
    AssertType      atype;
    Category        category;
    SymmetricCipher ciphers<3..2^8>;
    IdentityType    itype;
    AssertsDistribution distribution;
} ClientInit;
```

AssertType

```
enum {
    spki(1), keynote(2), ac(3)
} AssertType;
```

SymmetricCipher

```
struct {
    SymmetricAlgorithm algorithm;
    uint8               length;
} SymmetricCipher;
```

SymmetricAlgorithm

```
enum {
    idea(1), des(2), triple_des(3), cast(4), aes(5), null(255)
} SymmetricAlgorithm;
```

IdentityType

```
enum {
    x509(1), pgp(2), sdsi(3), rsa_public_key(4)
} IdentityType;
```

AssertsDistribution

```
enum {
    push_calculation(1), push_asserts(2), push_both(3), pull(4)
} AssertsDistribution;
```

ServerInit

```
struct {
    ProtocolVersion version;
    Random          nonce;
    AssertType      atype;
    SessionID       session_id;
    Category        category;
    SymmetricCipher cipher;
    IdentityType    itype;
    AssertsDistribution distribution;
} ServerInit;
```

Certificate

```
opaque CodedCert<1..216>;

struct {
    CodedCert certificate_list<1..216>;
} Certificate;
```

RSAParams

```
struct {
    opaque rsa_modulus<1..216>;
    opaque rsa_exponent<1..216>;
} RSAParams;
```

PKValue

```
struct {
    IdentityType pktype;
    select (pktype) {
        case rsa: RSAParams;
        default: Certificate;
    } value;
} PKValue;
```

Signature

```
private-key-encrypted struct {
    opaque md5_hash[16];
    opaque sha_hash[20];
} Signature;
```

PreMasterSecret

```
struct {
    opaque random[PREMASTER_LENGTH];
} PreMasterSecret;
```

EncryptedPreMasterSecret

```
struct {
    public-key-encrypted PreMasterSecret pre_master_secret;
} EncryptedPreMasterSecret;
```

ActivateCrypto

```
struct {
    EncryptedPreMasterSecret material;
    Signature                pk_verification;
} ActivateCrypto;
```

Identifier

```
enum {
    principal(0x50524E43), guard(0x4752444E)
} Identifier;
```

InitSession

```
struct {
    opaque md5_hash[16];
    opaque sha_hash[20];
} InitSession;
```

B.7 Valores criptográficos

Esta sección muestra cómo se realiza el cálculo de los valores criptográficos utilizados tanto por el módulo SM como por el nivel TC.

B.7.1 Valores relacionados con el mensaje ActivateCrypto

$$\text{md5_hash} = \text{MD5}(\text{ClientInit.nonce} + \text{MasterSecret} + \text{ServerInit.nonce})$$

$$\text{sha_hash} = \text{SHA1}(\text{ClientInit.nonce} + \text{MasterSecret} + \text{ServerInit.nonce})$$

$$\begin{aligned} \text{MasterSecret} = & \text{MD5}(\text{PreMasterSecret} + \text{SHA1}(\text{ClientInit.nonce})) + \\ & \text{MD5}(\text{ServerInit.nonce} + \text{SHA1}(\text{PreMasterSecret})) \end{aligned}$$

B.7.2 Valores relacionados con el mensaje InitSession

```
md5_hash = MD5(MasterSecret + Identifier + SM_Messages)
```

```
sha_hash = SHA1(MasterSecret + Identifier + SM_Messages)
```

B.7.3 Derivación de claves simétricas a partir del MasterSecret

```
clientKey = SHA1(MasterSecret + "AMBAR" + ClientInit.nonce) +  
            SHA1(MasterSecret + "AMBAR" + ServerInit.nonce)
```

```
serverKey = SHA1(MasterSecret + "AMBAR" + MD5(ServerInit.nonce)) +  
            SHA1(MasterSecret + "AMBAR" + MD5(ClientInit.nonce))
```

```
MACKey = MD5(ServerInit.nonce + SHA1(MasterSecret) +  
            ClientInit.nonce + "AMBAR")
```

```
IV = MD5(ClientInit.nonce + ServerInit.nonce + MACKey)
```

Apéndice C

Elementos de información de DCMS

Este apéndice incluye la especificación en BNF de los elementos de información correspondientes al sistema de gestión distribuida de credenciales (DCMS). Parte de las estructuras que se han empleado, aquellas que contienen dos asteriscos, están definidas en el documento de especificación de SPKI [68]

C.1 Naming Management System

C.1.1 Solicitudes NMS

```
<request-nms>:: "(" "cert-request" <issuer-name> <subject-req>
                <valid>? ")" ;

<issuer-name>:: "(" "issuer" "(" "name" <principal> <id> ")" ")" ;

<principal>:: <*pub-key*> | <*hash-of-key*> ;

<id>:: <byte-string> ;

<subject-req>:: "(" "subject" <subj-req> ")" ;

<subj-req>:: <principal> | <name> ;

<name>:: "(" "name" <principal> <id> ")" ;

<valid>:: "(" "valid" <valid-basic> ")" ;

<valid-basic>:: <not-before>? <not-after>? ;

<not-after>:: "(" "not-after" <date> ")" ;
```

```
<not-before>:: "(" "not-before" <date> ")" ;
```

```
<date>:: <byte-string> ;
```

C.1.2 Políticas de nombramiento

```
<acl-nms>:: "(" "acl" <acl-nms-entry>* ")" ;
```

```
<acl-nms-entry>:: "(" "entry" <subject-acl>? <deleg>? <tag-nms>
    <valid>? ")" ;
```

```
<subject-acl>:: "(" "subject" <subj-acl> ")" ;
```

```
<subj-acl>:: <principal> | <name> | <name-prefix> | <set-subj>;
```

```
<name-prefix>:: "(" "name" <principal> <prefix> ")" ;
```

```
<prefix>:: "(" "*" "prefix" <id> ")" ;
```

```
<set-subj>:: "(" "*" "set" <subj-acl>* ")" ;
```

```
<deleg>:: "(" "propagate" ")" ;
```

```
<tag-nms>:: "(" "cert-request" <issuer-acl> <subject-acl>
    <valid>? ")" ;
```

```
<issuer-acl>:: "(" "issuer" <iss-acl> ")" ;
```

```
<iss-acl>:: <name> | <name-prefix>;
```

C.2 Authorization Management System

C.2.1 Solicitudes AMS

```
<request-ams>:: "(" "cert-request" <issuer> <subject-req> <deleg>?
    <*tag*> <valid>? ")" ;
```

```
<issuer>:: "(" "issuer" <principal> ")" ;
```

C.2.2 Políticas de autorización

```
<acl-ams>:: "(" "acl" <acl-ams-entry>* ")" ;
```

```
<acl-ams-entry>:: "(" "entry" <subject-acl>? <deleg>? <tag-ams>  
    <valid>? ")" ;
```

```
<tag-ams>:: "(" "cert-request" <issuer> <subject-acl> <deleg>?  
    <*tag*> <valid>? ")" ;
```

C.3 Reduction Management System

C.3.1 Solicitudes RMS

```
<request-rms>:: "(" "sequence" <chain-reduction> <*cert*>* ")"
```

```
<chain-reduction>:: "(" "chain-reduction" <issuer> <subject-principal>  
    <*tag*> ")" ;
```

```
<subject-principal>:: "(" "subject" <principal> ")" ;
```

C.3.2 Políticas de reducción

```
<acl-rms>:: "(" "acl" <acl-rms-entry>* ")" ;
```

```
<acl-rms-entry>:: "(" "entry" <subject-principal> <deleg> <*tag*>  
    <valid>? ")" ;
```