Formato e conteúdo dos certificados X.509

Jeroen van de Graaf

18 de janeiro de 2006

Resumo

This note tries to sort out all the issues involved in defining the exact contents of X.509 certificate attributes, including the extensions.

1 Formato e conteúdo dos certificados X.509

Lembrete: historia X.509/PKIX; rfc 2459 X 3280

Lembrete: extendedKeyUsage, authorityInfoAccess

Lembrete: onde explicar OIDs

Lembrete: S= da Microsoft deve ser ST=

Lembrete: atualizar exemplos DNs?

1.1 Introdução

Como já explicado em secão ???, um certificado de identidade estabelece uma ligação entre uma chave pública e uma identidade. É um documento especial contendo uma chave pública, o nome do titular, e, opcionalmente, atributos (dados) adicionais que identificam o titular, assinado com a chave privada da AC emissora.

Podemos considerar a assinatura da AC como um **envelope** que protege o **conteúdo** do certificado (chave pública, o nome do titular, outros atributos) contra alterações. Veja o seguinte diagrama representando um certificado.

▷ chave pública
▷ nome titular
▷ *outros atributos*
▷ assinatura da AC

Este diagrama dá uma descrição esquemática que se aplica a qualquer certificado. O X.509 é um padrão que especifica exatamente o formato de um certificado: quais campos, como codificar cada campo, etc. Existem bastante motivos para criticar o padrão X.509, mas é o único padrão que ganhou aceitação mundialmente e, portanto, entender seu formato é muito interessante. Então, um certificado X.509 deve obedecer a um conjunto de regras e diretrizes, cujas mais importantes estão descritas neste apêndice.

2 Formato e conteúdo dos certificados X.509

2.1 Envelope e conteúdo de um certificado X.509

No conteúdo de um certificado X.509 podemos distinguir os *atributos básicos*, que todo certificado deve ter, e os atributos opcionais, normalmente chamadas de *extensões*, que um certificado pode ter. Nos apresentaremos os atributos básicos e extensões em seções distintos, a seguir.

Quanto a envelope, além do valor da assinatura, é indispensável saber qual algoritmo foi usado para cria-lo.

O seguinte diagrama atualiza o anterior:

- ▷ atributos básicos
- ▷ extensões
    - ▷ algoritmo da assinatura
    - ▷ valor da assinatura

2.2 Atributos básicos

Um certificado X.509 tem os seguintes atributos básicos:

version Este campo denote em qual versão de X.509 o certificado está codificado. Existem 3 versão de do padrão X.509 (v1, v2 e v3), mas na prática a versão usada é sempre v3.

serialNumber Uma AC deve atribuir um número serial diferente a cada certificado emitido, principalmente para seus próprios registros.

signature Indica qual algoritmo de assinatura é usado, usando um *object id*.

issuer É o nome do emissor certificado, usando um *Distinguished Name* (veja § **??**).

validity Contém dois campos: o primeiro indica a data e hora quando o certificado entre em vigor, o segundo a data e hora que ele expira. O fuso horário é o *Universal Time Coordinated* (antigamente chamado *Greenwich Mean Time*).

subject É o nome da entidade[1] a quem pertence o certificado, usando um *Distinguished Name* (veja § **??**). Pode ser uma pessoa, mas também um equipamento.

subjectPublicKeyInfo Este campo contém a chave privada, além do algoritmo usado.

issuerUniqueID Este atributo existe por motivos históricos, mas não é usado.

subjectUniqueID Este atributo existe por motivos históricos, mas não é usado.

2.3 O envelope

O envelope (a assinatura) protege o conteúdo do certificado, sendo os atributos obrigatórios (definidos na seção anterior) e as extensions (veja a Seção seguinte). O padrão X.509 define os seguintes atributos para a assinatura:

signatureAlgorithm Indica qual algoritmo de assinatura é usado, usando um *object id*, veja Seção **??**. Seu valor é igual ao valor do atributo `signature`, mas é repetido para impedir um determinado ataque à assinatura.

[1] A Microsoft traduz *subject* para *assunto*, o que é errado neste contexto. Solicitamos à Microsoft a tradução para *entidade*.

signatureValue A seqüência de bits que constitui a assinatura.

A seguinte tabela mostra os elementos discutidos até agora de um certificado X.509:

	nome atributo	**descrição atributo**	**exemplo**
conteúdo	`version`	versão do padrão X.509	v3
	`serialNumber`	número serial do certificado	27
	`signature`	algoritmo de assinatura	RSA with SHA-1
	`issuer`	emissor, sempre uma AC	
	`validity`	intervalo de validade	
	`subject`	proprietário do certificado	
	`subjectPublicKeyInfo`	chave pública	
	`issuerUniqueID`	não é usado	
	`subjectUniqueID`	não é usado	
	extensions	extensões, discutidos em § 2.4	
envelope	`signatureAlgorithm`	algoritmo de assinatura	RSA with SHA-1
	`signatureValue`	valor da assinatura	

2.4 Extensões do certificado X.509

Além dos atributos obrigatórios, a versão 3 de X.509 e a primeira versão que permite a inclusão de atributos opcionais, chamados extensões. Existem 14 extensões definidas no X.509/PKIX, mas desde do início a utilidade de algumas destas foi altamente criticada, e muitas vezes as fabricantes de programas simplesmente não implementaram a lógica toda necessária para interpretar estas extensões.

Por este motivo é importante saber que X.509 permite de cada extensão definir se ela é **crítica** (C) ou **não-crítica** (NC). Se certificado contém uma extensão critica, então uma aplicação que processa este certificado **deve** ser capaz de interpretar o significado desta extensão. Se ela não sabe interpretar, a aplicação deva recusar o certificado. No entanto, se uma extensão for marcada não-crítica, a aplicação pode interpretá-la ou ignorá-la.

Das 14 extensões definidas no X.509 e na RFC3280, apenas oito são relevantes, isto quer dizer, usadas para a maioria dos produtos. Eles estão listada em baixo. (O número x entre parênteses quer dizer que a extensão correspondente é definida na seção 4.2.1.x da RFC3280; isto facilita as referências cruzadas.)

Authority Key Identifier(1) Em teoria, uma AC poderia ter várias certificados distintos com o mesmo DN. Para facilitar a verificação da validade de um certificado o certificado filho pode incluir dados identificadores sobre o certificado pai, por exemplo o DN e o número serial.

Subject Key Identifier(2) Quando um usuário tem várias certificados com a mesma AC, determinar o certificado certo para verificar o caminho de certificação pode ser complicado. Esta extensão contém um campo `keyIdentifier` que pode conter o valor *hash* da chave pública. Esta extensão não deve ser marcada crítica, e ??? é usada pela ICP-Brasil.

Key Usage(3) Esta extensão é uma das mais importantes. Contém um conjunto de nove campos SIM/NÃO, definindo a aplicabilidade do certificado em determinados contextos. Infelizmente o significado de alguns destes campos é confuso, ainda mais são algumas combinações.

Os campos e as suas interpretações são:

`digitalSignature`	Este bit é marcado quando se trata de um certificado usado para verificar uma assinatura num documento ou num protocolo de autenticação. O bit não deve ser marcado nas situações descritas no `keyCertSign` e `cRLSign` a seguir.
`nonRepudiation`	Se o não-repúdio é importante (o que, por exemplo é o caso na verificação de assinaturas digitais, mas não num protocolo de autenticação) este bit deve ser marcado.
`keyEncipherment`	Este bit é marcado quando a chave privada correspondente pode ser usado para cifrar uma chave de sessão.
`dataEncipherment`	Este bit é marcado quando a chave privada correspondente pode ser usado para cifrar dados diferente que chaves criptográficas.
`keyAgreement`	Certificado é usado para o protocolo de troca de chaves de Diffie-Hellman.
`keyCertSign`	Se for marcado, o certificado é usado para verificar a validade de outros certificados. Ou seja, a chave privada correspondente foi usada para assinar certificados, portanto o dono do certificado é uma Autoridade Certificadora (Raiz ou Intermediária).
`cRLSign`	Se for marcado, o certificado é usado para verificar a assinatura numa LCR. Ou seja, a chave privada correspondente foi usada para assinar uma LCR (ou outro serviço de status), portanto o dono do certificado é uma Autoridade Certificadora (Raiz ou Intermediária) ou representa uma.
`encipherOnly`	Apenas a ser usado em combinação com `keyAgreement`, dizendo que a chave simétrica resultando do protocolo de troca de chave é usada para cifrar.
`decipherOnly`	Apenas a ser usado em combinação com `keyAgreement`, dizendo que a chave simétrica resultando do protocolo de troca de chave é usada para decifrar.

A seguinte tabela mostra quais bits devem ser marcada para várias aplicações.

Purpose of certificate	Required Key Usage bit
CA Signing	keyCertSign; cRLSign
SSL Client	digitalSignature
SSL Server	keyEncipherment
S/MIME Signing	digitalSignature
S/MIME Encryption	keyEncipherment
Certificate Signing	keyCertSign
Object Signing	digitalSignature
Diffie-Hellmann Key Exchange	keyAgreement; encipherOnly, decipherOnly

Certificate Policies(5) Esta extensão contém uma lista de links para políticas de certificados, usando OIDs, URLs ou texto.

Subject Alternative Name(7) Esta extensão contém um nome alternativo para o titular do certificado, e é explicada em Seção **??**.

Basic Constraints(10) Esta extensão consiste de dois campos: um campo booleano indicando se o dono do certificado é uma Autoridade Certificadora sim ou não, e um número positivo indicando o tamanho máximo do caminho de certificação de baixo do certificado. Esta extensão é crítica e é recomendável que ela conste em todos os certificados. (RFC3280 recomenda apenas para certificados de ACs, mas alguns produtos contem com a presença desta extensão também para certificados de entidades finais.)

Extended key usage (13) Esta extensão contém uma lista de aplicações para qual o certificado é usado, codificado com um *OID*. Caso a extensão seja marcada crítica, o certificado pode ser usado somente para a aplicação mencionada.

Valor	Aplicação
serverAuth	*Lembrete: explicar*
clientAuth	*Lembrete: explicar*
codeSigning	*Lembrete: explicar*
emailProtection	*Lembrete: explicar*
ipsecEndSystem	*Lembrete: explicar*
ipsecTunnel	*Lembrete: explicar*
ipsecUser	*Lembrete: explicar*
timeStamping	*Lembrete: explicar*

Lembrete: extendedKeyUsage msCodeInd, msEFS, msSGC, nsSGC

CRL Distribution Points(14) Contém um *link* para a Lista de Certificados Revogados mais recente.

As demais extensões, seis em número, são irrelevantes, já que ninguém as usa, e a grande maioria das aplicações não fazem um esforço para usá-las e interpretá-las. Trata-se das seguintes extensões: `Private Key Usage Period`(4), `Policy Mappings`(6), `Issuer Alternative Names`(8), `Subject Directory Attributes`(9), `Name Constraints`(11), `Policy Constraints`(12). O número entre parênteses se refere à subseção da seção 4.2.1 na RFC3280 em que a extensão correspondente é definida.

Ainda por cima é possível incluir extensões próprias, porém estas funcionarão apenas com as aplicações próprias; não pode se esperar que um outro produto conhece esta extensão e sabe interpretá-la corretamente.

2.5 Um resumo

A seguinte tabela resume as extensões recomendadas na RFC3280. Ela mostra nas linhas todas as extensões que estão incluídas na maioria dos produtos PKI. Nas colunas mostra os diferentes certificados presentes numa ICP: certificado raiz; certificado para uma AC Intermediária; entidade final – pessoa; e entidade final – equipamento.

#	extension	AC-Raiz	AC-Int	EF-pessoa	EF-equipamento
1	Authority Key Identifier	NC	NC	NC	NC
2	Subject Key Identifier	NC	NC	NC	NC
3	Key Usage	C	C	C	C
5	Certificate Policies	NC	NC	NC	NC
7	Subject Alternative Name	–	–	NC	NC
10	Basic Constraints	C	C	C	C
13	Extended key usage field	–	–	–	NC
14	CRL Distribution Points	NC	NC	NC	NC

3 Distinguished Names

3.1 A Teoria

O padrão X.509 na realidade faz parte de um outro padrão, X.500. Este é um projeto demasiadamente ambicioso com o objetivo de definir um estrutura mundial de diretórios. Segundo este padrão, cada entidade (pessoa, equipamento, tec) receberia um único nome hierárquico, chamado *Distinguished Name (DN)*. X.500 define os seguintes componentes para um DN:

Common Name (CN) Supostamente o nome de dia-a-dia da pessoa, por exemplo CN=Fulano de Tal.

Organizational Unit (OU) Parte da organização, por exemplo, OU=Vendas, or OU=Departamento de ciência de Computação. Uma DN poderia conter várias OUs, mas as implementações usam muitas vezes um limite fixo, por exemplo 3 OUs no máximo.

Organization (O) Empresa ou órgão, como O=Petrobras, O=UFMG, O=ICP-Brasil.

Locality (L) Normalmente o município: L=Brasília, L=Belo Horizonte

State (ST) Estado. Exemplo: ST=Minas Gerais

Country (C) Existe o padrão ISO 3166 com os códigos de cada país predefinido. Para Brasil é C=BR.

Email (E) Este componente oficialmente não faz parte de X.500, mas é muitas vezes usado em certificados. Veja a discussão.

Existem até outros componentes, como DC (domain component), que está pouco usado.

Os componentes CN, OU, O, L, ST têm um tamanho máximo de 64 caracteres. O algoritmo definido pelo ITU-T para estabelecer a igualdade de DNs é bastante genérico. Que, na comparação, não há distinção entre letras maiúsculas e minúsculas, e razoável, mas o padrão permite uma permutação na ordem dos componentes. Alguns fabricantes se recusam a implementar esta característica e simplesmente comparam os dois strings constituindo os DNs.

A ICP-Brasil impõe ainda mais restrições:

- Os caracteres devem obedecer ao Código NBR9611, e devem incluir:
  - os caracteres alfanuméricos [A-Za-z0-9]
  - o branco (espaço)
  - os caracteres especias `!"#$%&'()*+,-./:;=?@\`
- Acentos e a letra "ç"com cedilha não podem ser usados. Substitua-se um tal caracter pelo caracter não-acentuado ou pela letra "c".
- O CN da pessoa tem apenas letras maiúsculas, sem acentos ou cedilhas. Exemplo: o CN do José Gonçalves Guimarães é CN=JOSE GONCALVES GUIMARAES

3.2 A realidade

A realidade pode ser ilustrada através de duas perguntas: Você sabe qual é seu DN? E seu endereço email? As respostas mais prováveis são que você não tem um DN (talvez nunca ouviu falar disso antes), e que você tem um ou vários endereços email.

Os DNs foram inventadas na decada de (), mas mais uma vez a realidade não se comportou segundo os planos da ITU-T. Surgiu a internet, com seu próprio formato de endereços emails e endereços http.

A conseqüência disto é que hoje não há consenso sobre como usar os DNs num certificado. Quem abre alguns certificados com seu navegador [link] encontra um zoológico de variedades. Eis alguns exemplos reais:

Banco do Brasil – Issuer :
CN = Entidade Certificadora Banco do Brasil CA
OU = Class 2 CA - OnSite Individual Subscriber
OU = Terms of use at https://www.certisign.com.br/RPA (c)00
OU = VeriSign Trust Network
O = Banco do Brasil S.A.

Banco do Brasil – Subject :
E = jvdg@ufmg.br
CN = JEROEN ANTONIUS MARIA VAN DE GRAAF
OU = CPF - 00012345678901
OU = MCI - 123456789
OU = www.verisign.com/repository/CPS Incorp. by Ref.,LIAB.LTD(c)96
OU = Clientes
O = Banco do Brasil S.A.

ICP-Brasil Root :
CN = Autoridade Certificadora Raiz Brasileira
S = DF
L = Brasilia
OU = Instituto Nacional de Tecnologia da Informacao - ITI
O = ICP-Brasil
C = BR

e-CPF :
C=BR
O=ICP-Brasil
OU=Secretaria da Receita Federal-SRF
OU=SRF e-CPF
CN=Jose da Silva Valenca:67849823125

VeriSign Class 1 :
OU = Class 1 Public Primary Certification Authority
O = VeriSign, Inc.
C = US

VeriSign Class 1 :
CN = VeriSign Class 1 CA Individual Subscriber-Persona Not Validated
OU = www.verisign.com/repository/RPA Incorp. By Ref.,LIAB.LTD(c)98
OU = VeriSign Trust Network
O = VeriSign, Inc.

Ou seja, é uma confusão.

3.3 Aspectos de credibilidade e responsabilidade jurídica

Para ACs-Raiz é importante comunicar a credibilidade de seus certificados ao usuário, especificamente se o mesmo AC emite certificados distintos com graus de credibilidade diferentes, como a VeriSign. Note que o certificado dispõe de várias extensões potenciais que podem ser aproveitados para conter este tipo de informação. Porém, não se pode ter certeza que todas as aplicações suportam estas extensões e mostram as informações contidas nelas. Por este motivo, como se pode ver, várias ACs utilizam o componente OU do DN para incluir informações sobre a credibilidade do certificado.

3.4 Homônimos

Há um homônimo quando duas pessoas diferentes têm o mesmo nome. Em outras palavras: uma colisão de nome. Acontece com nomes freqüentes: José Carlos da Silva, Ana Paula Duarte. Segundo a doctrine X.500 todo mundo deve obter um nome único, portanto homônimos devem ser evitados. Existem várias estratégias para tal:

1. Inclua o endereço email no DN. Dependendo da aplicação e do público alvo da ICP isto pode ser uma boa solução, porque até pessoas com o mesmo nome deve ter endereços emails distintos, assim evitando problemas de colisão. Também há desvantagens, porém:
   - Em algumas comunidades de usuários, não cada pessoa tem endereço email. Por exemplo é o caso da população brasileira, para quem órgãos do governo contemplam emitir certificados.
   - Quando uma pessoa troca de endereço email, o seu DN muda, o certificado antigo deve ser revogado e um novo certificado deve ser emitido.
   - Incluir o endereço email poderia deixar o titular mais vulnerável a *spam*. Veja a seção sobre privacidade **??**.

- Numa comunidade com o mesmo espaço de endereço email (*domain name*), o problema da colisão é portanto transferido para este espaço. Quando se trata de um novo domínio, como se define o endereço email de duas pessoas com o mesmo nome?

2. Defina OUs adicionais até uma profundidade que colisões são muito improváveis. Esta deve ser a solução politicamente correto do X.500. Ela reduz a chance a uma colisão, mas não à elimina.

3. Criar um número de identificação único para cada pessoa, o usar um número existente. Esta é a estratégia adotada nos certificados do e-CPF, que evitam colis !oes incluindo o CPF do titular no CN.

4. Crie um número aleatório e aponha-o ao CN. As chances de colisão de números são baixas, as chances de colisão de nome **e** número são mínimas. O *DOE Science Grid* adotou esta solução; acrescentam uma seqüencia de caracteres alfa-numérico ([A-Z0-9]) de tamanho 5.

O grande leque de estratégias mostra que este problema recebeu muito atenção. Porém. nenhuma estratégia realmente atende: ou diminui as chances mas não resolve, ou junta um número esquisito ao nome da pessoa. Ninguem costa de assinar como "Fulano de Tal 12345". E apesar de ser um sacrilégio na doctrine X.500, o problema de homônimos **para uso em certificados** pode ser menor que parece. Pois qual é o problema de ter dois certificados com CN=José da Silva e os outros componentes do DN iguais também? Os certificados são diferentes já que a chave pública são diferentes, portanto o *hash*es dos certificados são diferentes também. Quem verifica o certificado vai querer saber de qual José da Silva se trata, informações que poderiam estar nos outros atributos do certificado (a extensão SubjectAltName foi criada por este motivo; veja a discussão a seguir) e não necessariamente no DN.

3.5 O endereço email

Como se pode verificar, o padrão X.500 foi desenvolvido como se email não existisse, e o componente "E="oficialmente não faz parte do X.500. Mas obviamente o endereço email é um elemento muito importante numa identidade na internet, portanto surgiram aplicações incluindo um componente Email no DN.

Esta estratégia foi retroativamente sancionada pelos autores da RFC2459 e repetida em RFC3280, dizendo que aplicações devem ser capaz de interpretar um tal tipo de DN (veja §4.2.1.6 da RFC3280). Porém, ao mesmo tempo eles dizem que novos certificados deveriam colocar um endereço email na extensão SubjectAltName.

3.6 A extensão SubjectAltName

Para atender a demanda de aplicações cujo espaço de endereços não se conforma a X.500 foi criada uma extensão especifica, a SubjectAltName. Ela pode conter um ou vários campos dos seguintes tipos: um endereço email convencional, um endereço X.400, um nome DNS, uma URI, um endereço IP, entre outros. Além disso, pode se incluir qualquer tipo de informação sobre o titular usando uma seqüência de bytes para codificar o que você quiser (a custo de ser incompatível com o resto do mundo).

Foi a abordagem seguida pela ICP-Brasil. Na sua Resolução ??? ela estipula que cada certificado emitido para uma pessoa física deve incluir três campos na extensão SubjectAltName, incluindo os seguintes dados:

"[...] nas primeiras 8 (oito)posições, a data de nascimento do titular, no formato ddmmaaaa; nas 11 (onze) posições subseqüentes, o Cadastro de Pessoa Física (CPF) do titular; nas 11 (onze) posições subseqüentes, o Número de Identificação Social-NIS (PIS, PASEP ou CI); nas 11 (onze) posições subseqüentes, o número do Registro Geral (RG) do titular; nas 6 (seis) posições subseqüentes, as siglas do órgão expedidor do RG e respectiva UF.

[...] nas 12 (doze)posições o número do Cadastro Específico do INSS (CEI) da pessoa física titular do certificado.

[...] nas primeiras 12 (doze) posições, o número de inscrição do Título de Eleitor; nas 3 (três) posições subseqüentes, a Zona Eleitoral; nas 4 (quatro) posições seguintes, a Seção; nas 22 (vinte e duas) posições subseqüentes, o município e a UF do Título de Eleitor."

Uma desvantagem desta abordagem é que a grande maioria das aplicações PKI (notadamente Internet Explorer e Mozilla) não visualizem os valores nestes campos, o que não permite a um usuário leigo verificar o conteúdo do seu próprio certificado. A última seção deste apêndice contém um exemplo de um certificado da ICP-Brasil usando uma ferramenta que mostra os campos da SubjectAltName corretamente.

4 A codificação binária de um certificado

4.1 ASN.1

Para especificar o formato exato de um certificado X.509 é usada a linguagem ASN.1. Esta linguagem foi criada pela ITU-T especificamente para o fim de especificar estrutura de dados, independente de qualquer linguagem de programação ou método existente; na realidade ASN.1 é um padrão em si.

Para inspeção casual, ASN.1 não é muito complicado. Veja como um certificado é especificado em §4.1 da [**?**].

```
Certificate  ::=  SEQUENCE  {
     tbsCertificate       TBSCertificate,
     signatureAlgorithm   AlgorithmIdentifier,
     signatureValue       BIT STRING  }

TBSCertificate  ::=  SEQUENCE  {
     version         [0]  EXPLICIT Version DEFAULT v1,
     serialNumber         CertificateSerialNumber,
     signature            AlgorithmIdentifier,
     issuer               Name,
     validity             Validity,
     subject              Name,
     subjectPublicKeyInfo SubjectPublicKeyInfo,
     issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     extensions      [3]  EXPLICIT Extensions OPTIONAL
                          -- If present, version MUST be v3
     }

Version  ::=  INTEGER  {  v1(0), v2(1), v3(2)  }

CertificateSerialNumber  ::=  INTEGER

Validity ::= SEQUENCE {
     notBefore      Time,
     notAfter       Time }

Time ::= CHOICE {
     utcTime        UTCTime,
     generalTime    GeneralizedTime }

UniqueIdentifier  ::=  BIT STRING

SubjectPublicKeyInfo  ::=  SEQUENCE  {
     algorithm            AlgorithmIdentifier,
     subjectPublicKey     BIT STRING  }
```

```
Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension

Extension  ::=  SEQUENCE  {
     extnID      OBJECT IDENTIFIER,
     critical    BOOLEAN DEFAULT FALSE,
     extnValue   OCTET STRING  }
```

Mas na realidade trata-se de uma linguagem bastante complexa. Veja por exemplo a especificação de `Subject Alt Name` embaixo. Felizmente no contexto deste apêndice não é preciso entrar em detalhes.

```
SubjectAltName ::= GeneralNames

GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName

GeneralName ::= CHOICE {
     otherName                       [0]     OtherName,
     rfc822Name                      [1]     IA5String,
     dNSName                         [2]     IA5String,
     x400Address                     [3]     ORAddress,
     directoryName                   [4]     Name,
     ediPartyName                    [5]     EDIPartyName,
     uniformResourceIdentifier       [6]     IA5String,
     iPAddress                       [7]     OCTET STRING,
     registeredID                    [8]     OBJECT IDENTIFIER }

OtherName ::= SEQUENCE {
     type-id    OBJECT IDENTIFIER,
     value      [0] EXPLICIT ANY DEFINED BY type-id }

EDIPartyName ::= SEQUENCE {
     nameAssigner            [0]     DirectoryString OPTIONAL,
     partyName               [1]     DirectoryString }
```

4.2 BER e DER

ASN.1 é uma linguagem de especificação alto nível, portanto é necessário definir como uma instância real de dados especificada em ASN.1, como, por exemplo, um certificado X.509, é codificada numa seqüência binária. Por este motivo existem os *Distinguished Encoding Rules (DER)*. Trata-se de uma extensão dos *Basic Encoding Rules*, que definem como um valores especificado em ASN.1 deve ser codificado numa seqüência binária. Infelizmente os BER permitem ambigüidades, que são resolvidas pelos DER.

Se você tenta abrir um certificado com um editor de texto e não consegue, é provavelmente porque o certificado está codificado em DER. Porém, existem programas que fazem o mapeamento inverso. Ou seja, que abrem e interpretam um arquivo binário codificado em DER, e mostram a estrutura de dados com seus valores correspondentes. Um tal programa é chamada um *ASN.1 parser*. Exemplos são: a opção `asn1parse` do OpenSSL, o program BERViewer para Windows, e o programa `dumpasn1.c` que deve ser compilado. Os grandes navegadores já têm um parser embutido, que é chamado quando você pede visualizar o conteúdo de um certificado. O *dump* no final deste apêndice foi feito com `dumpasn1`. [Talvez fica melhor no site do livro]

4.3 Object IDs

[Jvdg: Talvez um box seria melhor.]

4.4 Base64 e PEM

Mas suponha que você queira fazer um *backup* em papel do seu certificado, ou mandá-lo para alguém por fax ou email? (Mais comum é que um titular quer mandar uma requisição de certificado, também uma estrutura ASN.1, por email.) Como transformar um arquivo binário num arquivo legível?

Existe a codificação *base64* que faz exatamente isto. Base64 é capaz de transformar qualquer seqüência de inglesbytes numa seqüência de caracteres ASCII. Se você considere as letras maiúsculas e minúsculas `[A-Z]` e `[a-z]`, os algarismos `[0-9]` e os caracteres `"+"` e `"/"`, você tem um conjunto de $64 = 2^6$ símbolos. Portanto é possível definir um mapeamento entre o conjunto de todas as seqüências de 6 bits e estes símbolos, por exemplo $000000 \mapsto$ `A`, $000001 \mapsto$ `B`, ..., $111111 \mapsto$ `/`.

O seguinte diagrama mostra as primeiras e últimas linhas de um certificado codificada desta forma.


```
-----BEGIN CERTIFICATE-----
MIIZlzCCGH+gAwIBAgIBCDANBgkqhkiG9w0BAQUFADCBujELMAkGA1UEBhMCQlIx
CzAJBgNVBAgTAk1HMR8wHQYDVQQKExZFc3RhZG8gZGUgTWluYXMgR2VyYWlzMTsw
OQYDVQQLEzJJT01HIC0tIEltcHJlbnNhIE9maWNpYWwgZG8gRXN0YWRvIGRlIE1p

 ......
 ......

AxMaVEhJQUdPIE1BUkNPUyBQSU5UTyBTQU5UT1MxgakwgaYGA1UdERSBnlYxLjM7
TUctMTUxNDU2MjE7MDIvMDMvMTk5MDtTU1AvTUc7VEhJQUdPIE1BUkNPUyBQSU5U
TyBTQU5UT1M7TUFSQ09TIE1BUklOSE8gRE9TIFNBTlRPUztFTElBTkUgTUFSSUEg
-----END CERTIFICATE-----
```


Observe que a codificação em base 64 começa na segunda linha e termina na penúltima. A primeira e a última linha, com `BEGIN CERTIFICATE` e `END CERTIFICATE`, **não** fazem parte da base64, mas servem para delimitar o início e fim da codificação e para explicar o que significa aquela seqüência de símbolos quasi aleatórios. Esta convenção provêem do padrão PEM: em geral um arquivo de tipo PEM (às vezes com esta extensão) é um arquivo texto em base64 procedido e terminado com uma linha special com `BEGIN _OBJECT_` e `END _OBJECT_`.

5 Um exemplo de um certificado

```
   0 1267: SEQUENCE {
   4  987:   SEQUENCE {
   8    3:     [0] {
  10    1:       INTEGER 2
         :       }
  13   16:     INTEGER 3B 2D B0 8F 3E 68 78 91 C3 EC 6E 93 BB C5 04 C6
  31   13:     SEQUENCE {
  33    9:       OBJECT IDENTIFIER sha1withRSAEncryption (1 2 840 113549 1 1 5)
  44    0:       NULL
         :       }
  46  110:     SEQUENCE {
  48   11:       SET {
  50    9:         SEQUENCE {
  52    3:           OBJECT IDENTIFIER countryName (2 5 4 6)
  57    2:           PrintableString 'BR'
         :           }
         :         }
  61   19:       SET {
  63   17:         SEQUENCE {
  65    3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
  70   10:           PrintableString 'ICP-Brasil'
         :           }
         :         }
  82   44:       SET {
  84   42:         SEQUENCE {
  86    3:           OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
```

```
 91   35:           PrintableString 'Secretaria da Receita Federal - SRF'
        :           }
        :         }
128   28:       SET {
130   26:         SEQUENCE {
132    3:           OBJECT IDENTIFIER commonName (2 5 4 3)
137   19:           PrintableString 'AC CertiSign SRF V3'
        :           }
        :         }
        :       }
158   30:     SEQUENCE {
160   13:       UTCTime 12/04/2005 00:00:00 GMT
175   13:       UTCTime 11/04/2008 23:59:59 GMT
        :       }
190  131:     SEQUENCE {
193   19:       SET {
195   17:         SEQUENCE {
197    3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
202   10:           TeletexString 'ICP-Brasil'
        :           }
        :         }
214   18:       SET {
216   16:         SEQUENCE {
218    3:           OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
223    9:           TeletexString 'SRF e-CPF'
        :           }
        :         }
234   42:       SET {
236   40:         SEQUENCE {
238    3:           OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
243   33:           PrintableString 'Secretaria da Receita Federal-SRF'
        :           }
        :         }
278   11:       SET {
280    9:         SEQUENCE {
282    3:           OBJECT IDENTIFIER countryName (2 5 4 6)
287    2:           PrintableString 'BR'
        :           }
        :         }
291   31:       SET {
293   29:         SEQUENCE {
295    3:           OBJECT IDENTIFIER commonName (2 5 4 3)
300   22:           PrintableString 'ROGER TANG:08928296803'
        :           }
        :         }
        :       }
324  159:     SEQUENCE {
327   13:       SEQUENCE {
329    9:         OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
340    0:         NULL
        :         }
342  141:       BIT STRING, encapsulates {
346  137:         SEQUENCE {
349  129:           INTEGER
        :             00 CF 10 6B A1 A2 B7 CE 4F 98 5E 24 85 AA E7 FB
        :             B3 0C 3C 6E CD 13 6D 27 5E 57 B4 10 F9 44 81 B0
        :             31 4E C6 E0 2B DC 5D 99 92 0C ED F1 9C CB A6 6B
        :             16 43 07 C5 2F 39 D0 FB 34 D6 C7 7E 64 99 BA B8
        :             1B 66 97 53 16 1A 70 C4 0B 6C C9 CF 84 C8 02 8C
        :             F5 81 B7 69 0D DB E1 0F BF AA C7 C2 08 BC E4 B5
        :             7F DA A2 6A AC E3 CD A7 5F 02 92 1B B6 EC 3F F9
        :             5D 7C EC 4A AD DE 56 B7 6A 44 13 D8 30 9D 95 55
        :                     [ Another 1 bytes skipped ]
481    3:           INTEGER 65537
        :           }
        :         }
        :       }
486  505:     [3] {
490  501:       SEQUENCE {
494  162:         SEQUENCE {
497    3:           OBJECT IDENTIFIER subjectAltName (2 5 29 17)
502  154:           OCTET STRING, encapsulates {
505  151:             SEQUENCE {
```

```
 508   63:                 [0] {
 510    5:                   OBJECT IDENTIFIER '2 16 76 1 3 1'
 517   54:                   [0] {
 519   52:                     UTF8String
          :                     '05111970089282968030000000000000000019680940xSSP'
          :                     '  SP'
          :                     }
          :                   }
 573   21:                 [0] {
 575    5:                   OBJECT IDENTIFIER '2 16 76 1 3 6'
 582   12:                   [0] {
 584   10:                     UTF8String '1435042700'
          :                     }
          :                   }
 596   40:                 [0] {
 598    5:                   OBJECT IDENTIFIER '2 16 76 1 3 5'
 605   31:                   [0] {
 607   29:                     UTF8String '203928060175271004lSorocabaSP'
          :                     }
          :                   }
 638   19:                 [1] 'rtang@lsitec.org.br'
          :                 }
          :               }
          :             }
 659    9:           SEQUENCE {
 661    3:             OBJECT IDENTIFIER basicConstraints (2 5 29 19)
 666    2:             OCTET STRING, encapsulates {
 668    0:               SEQUENCE {}
          :               }
          :             }
 670   98:           SEQUENCE {
 672    3:             OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
 677   91:             OCTET STRING, encapsulates {
 679   89:               SEQUENCE {
 681   87:                 SEQUENCE {
 683   85:                   [0] {
 685   83:                     [0] {
 687   81:                       [6]
          :                     'http://icp-brasil.certisign.com.br/repositorio/l'
          :                     'cr/ACCertiSignSRFV3/LatestCRL.crl'
          :                       }
          :                     }
          :                   }
          :                 }
          :               }
          :             }
 770   31:           SEQUENCE {
 772    3:             OBJECT IDENTIFIER authorityKeyIdentifier (2 5 29 35)
 777   24:             OCTET STRING, encapsulates {
 779   22:               SEQUENCE {
 781   20:                 [0]
          :                   F6 9D 59 5D FE BF C5 72 CD DD CE C4 2E 66 1B 2E
          :                   EE 08 CF 76
          :                 }
          :               }
          :             }
 803   14:           SEQUENCE {
 805    3:             OBJECT IDENTIFIER keyUsage (2 5 29 15)
 810    1:             BOOLEAN TRUE
 813    4:             OCTET STRING, encapsulates {
 815    2:               BIT STRING 5 unused bits
          :                 '111'B
          :               }
          :             }
 819   85:           SEQUENCE {
 821    3:             OBJECT IDENTIFIER certificatePolicies (2 5 29 32)
 826   78:             OCTET STRING, encapsulates {
 828   76:               SEQUENCE {
 830   74:                 SEQUENCE {
 832    6:                   OBJECT IDENTIFIER '2 16 76 1 2 3 6'
 840   64:                   SEQUENCE {
 842   62:                     SEQUENCE {
 844    8:                       OBJECT IDENTIFIER cps (1 3 6 1 5 5 7 2 1)
```

```
 854   50:                     IA5String
         :                   'http://icp-brasil.certisign.com.br/repositorio/d'
         :                   'pc'
         :                     }
         :                   }
         :                 }
         :               }
         :             }
         :           }
 906   29:         SEQUENCE {
 908    3:           OBJECT IDENTIFIER extKeyUsage (2 5 29 37)
 913   22:           OCTET STRING, encapsulates {
 915   20:             SEQUENCE {
 917    8:               OBJECT IDENTIFIER emailProtection (1 3 6 1 5 5 7 3 4)
 927    8:               OBJECT IDENTIFIER clientAuth (1 3 6 1 5 5 7 3 2)
         :               }
         :             }
         :           }
 937   56:         SEQUENCE {
 939    8:           OBJECT IDENTIFIER authorityInfoAccess (1 3 6 1 5 5 7 1 1)
 949   44:           OCTET STRING, encapsulates {
 951   42:             SEQUENCE {
 953   40:               SEQUENCE {
 955    8:                 OBJECT IDENTIFIER ocsp (1 3 6 1 5 5 7 48 1)
 965   28:                 [6] 'http://ocsp.certisign.com.br'
         :                 }
         :               }
         :             }
         :           }
         :         }
         :       }
         :     }
 995   13:   SEQUENCE {
 997    9:     OBJECT IDENTIFIER sha1withRSAEncryption (1 2 840 113549 1 1 5)
1008    0:     NULL
         :     }
1010  257:   BIT STRING
         :     A7 17 BE B1 5A A5 26 D7 21 09 31 A7 2C 8C B2 30
         :     F9 97 BB 31 91 4C 0D 9E BC 25 13 13 28 41 81 4E
         :     6A F2 D1 BD 07 A3 5F 69 53 AD B8 54 E2 C9 D2 6F
         :     72 25 3A 28 D4 6B 70 48 28 9F 00 91 DD D3 CD FD
         :     CC 9A 69 4B 04 E4 C4 C6 8C 21 3A 14 8E 73 B9 5A
         :     DB 5C 72 D4 15 DC 80 EF 8D 78 28 0F 10 AA 91 09
         :     14 3B A1 89 E2 38 E0 A7 86 95 C4 A1 60 D1 FF C2
         :     88 27 0C E0 A7 E5 8B 55 6A B0 1C E4 51 24 05 9C
         :             [ Another 128 bytes skipped ]
         :   }
```

Lembrete: comentar o exemplo