Abstract Syntax Notation One

Formaliser et transmettre l'information structurée

Nathanaël Cottin

contact@ncottin.net
http://www.ncottin.net

version 0.0.4 – 2010

Partie 1 – Présentation de l'ASN.1

▸ Afficher

Partie 2 – Marquage en ASN.1

▸ Afficher

Partie 3 – Encodages standards

▸ Afficher

Partie 4 – Sécurité de l'ASN.1

▸ Afficher

Partie I

Présentation de l'ASN.1

Concepts fondamentaux
Syntaxe de base

Nécessité de l'ASN.1

Comment échanger de l'information numérique ?

- Information numérique structurée
- Hétérogénéité

Emploi quotidien :

- Aéronautique
- Télécommunications
- Protocoles réseau et Internet
- Sécurité informatique

L'ASN.1 au cœur des standards

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

```
AlgorithmIdentifier ::= SEQUENCE {
  algorithm  OBJECT IDENTIFIER,
  parameters ANY DEFINED BY algorithm OPTIONAL
}
```

L'ASN.1 au cœur des standards

```
RSAPublicKey ::= SEQUENCE {
  modulus         INTEGER,   -- n
  publicExponent  INTEGER    -- e
}

RSAPrivateKey ::= SEQUENCE {
  version          Version,
  modulus          INTEGER, -- n
  publicExponent   INTEGER, -- e
  privateExponent  INTEGER, -- d
  prime1           INTEGER, -- p
  prime2           INTEGER, -- q
  (etc.)
}
```


Outils ASN.1 : 2002

- OSS Nokalva : C, C++, Java, C#
- Objective Systems : C, C++, Java, C#
- UniGone : Java, C#
- Baltimore : Java
- OpenSSL : C
- PowerASN : Java (C# à venir)
- ...

Histoire de l'ASN.1

- 1984 : recommandation ITU-T X.409
- 1984 : l'ISO réécrit X.409
- 1985 : l'ITU-T rejoint le travail de l'ISO relatif à l'ASN.1
- 1987 : ISO 8824 et ISO 8825
- 1989 : recommandations X.208 et X.209
- 1990 : mise à jour ISO 8824 et 8825
- 1994 : recommandation X.690
- 1995 : ISO/IEC 8825-1 et 8825-2

Depuis : mises à jour (XER, ...) et correctifs techniques

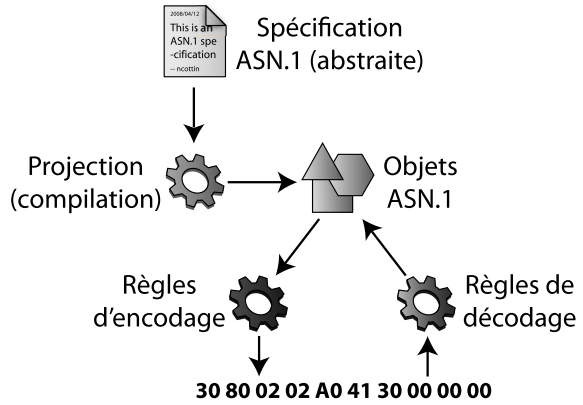
Définition de l'ASN.1

Composants

Standard international composé des éléments suivants :

- Notation formelle (syntaxe abstraite)
- Spécification de règles d'encodage :
  - Basées sur les octets
  - Basées sur les bits
  - Basées sur des caractères

Processus général



Syntaxes de transfert standards

- BER : Basic Encoding Rules
- CER : Canonical Encoding Rules
- DER : Distinguished Encoding Rules
- PER : Packed Encoding Rules
- XER (et dérivés) : XML Encoding Rules

Types simples

- `INTEGER`
- `ENUMERATED`
- `REAL`
- `BOOLEAN`
- `OCTET STRING` et `BIT STRING`
- `NumericString`, `VisibleString`, ...
- `UTCTime` et `GeneralizedTime`
- `NULL`
- `RELATIVE IDENTIFIER` et `OBJECT IDENTIFIER`
- ...

Types abstraits ASN.1 structurés

- `SEQUENCE` (`OF`)
- `SET` (`OF`)
- ...

Types transparents

- `CHOICE`
- `OpenType`

Contraintes sur les types simples

- Types numériques : intervalle de valeurs
- Types énumérés : valeurs acceptées
- Chaînes de caractères : taille, caractères autorisés
- Date : validité du format de date
- Identificateurs : valeurs entières qui les composent

Contraintes sur les types structurés

- Tailles minimum et maximum
- Type des éléments
- Présence requise ou optionnelle

Instanciation des types

```
instance ::= type valeur          -- Général
instance ::= <type>valeur</type> -- XML
```

Déclarations

```
Version ::= INTEGER
```

```
Chaine ::= UTF8String SIZE(1..50)
-- Chaîne non vide de moins de 51 caractères
```

```
Genre ::= ENUMERATED {
  masculin,
  feminin
}
```


Initialisations

```
vrai ::= BOOLEAN TRUE
```

```
une-valeur ::= INTEGER 1
```

```
Entier ::= INTEGER (0..10)
uneAutreValeur ::= Entier 2
```


Déclarations

```
Individu ::= SEQUENCE {
  version  Version DEFAULT v1(1),
  prenom   Chaine,
  nom      Chaine,
  age      INTEGER (0..130) OPTIONAL,
  genre    Genre DEFAULT feminin
}
```

```
Individus ::= SET OF Individu
```


Initialisations

```
ncottin Individu ::= {
  prenom "Nathanael",
  nom    "Cottin",
  -- 'age' est optionnel
  genre  masculin
}
```

```
groupe Individus ::= {
  {prenom "Pierre", nom "Dupont", age "30", genre
     masculin},
  {prenom "Chantal", nom "Durand"}
}
```

Déclarations

```
Temps ::= CHOICE {
  utc UTCTime,
  gen GeneralizedTime
}
```

```
Generique ::= SEQUENCE {
  id     OBJECT IDENTIFIER,
  valeur OpenType DEFINED BY id
}
```


Initialisations

```
avant Temps ::= {
  utc "0712242359Z"
}
```

```
blob Generique ::= {
  id     "1.2.3.4.5",
  valeur BOOLEAN TRUE
}
```


Version scolaire

```
ListeChainee1 ::= SEQUENCE {
  valeur  OpenType,
  suivant ElementSuivant
}

ElementSuivant ::= CHOICE {
  svt ListeChainee1,
  fin NULL
}
```


Première amélioration

```
ListeChainee2 ::= SEQUENCE {
  valeur  OpenType,
  suivant ListeChainee2 OPTIONAL
}
```

Seconde amélioration

Ou encore :

```
ListeChainee3 ::= SEQUENCE OF OpenType
```


Déclaration de modules

```
NomModule {OID} -- Identifiant optionnel
DEFINITIONS
IMPLICIT TAGS -- En-tete de module
  ::=
BEGIN
  -- Section IMPORTS
  -- Section EXPORTS
  -- Déclaration des types dérivés
END
```


Substitution de type

Avant :

```
numerique ::= INTEGER
```

Après :

```
numerique ::= CHOICE {
  entier INTEGER,
  reel   REAL
}
```


Extensibilité : spécification originelle

```
Individu ::= SEQUENCE {
  version Version DEFAULT v1(1),
  prenom  Chaine,
  nom     Chaine,
  age     INTEGER (0..130) OPTIONAL,
  genre   Genre DEFAULT feminin,
  ...
}
```

Extensibilité : première évolution

```
Individu ::= SEQUENCE {
  version  Version DEFAULT v1(1),
  prenom   Chaine,
  nom      Chaine,
  age      INTEGER (0..130) OPTIONAL,
  genre    Genre DEFAULT feminin,
  ...,
  [[2:adresse  Chaine,
      ville    Chaine]]
}
```


Extensibilité : seconde évolution

```
Individu ::= SEQUENCE {
  version  Version DEFAULT v1(1),
  prenom   Chaine,
  nom      Chaine,
  age      INTEGER (0..130) OPTIONAL,
  genre    Genre DEFAULT feminin,
  ...,
  [[2:adresse  Chaine,
      ville    Chaine]],
  [[3:email    GeneralString]]
}
```


Résumé

- Notation abstraite éprouvée
- Évolutivité des spécifications abstraites
- Interopérabilité
- Syntaxes de transfert normalisées et performantes
- Nombreuses implantations disponibles

Tend à être remplacé par XML

Limites d'expression

```
DeuxEntiers ::= SEQUENCE {
  valeur1 INTEGER OPTIONAL,
  valeur2 INTEGER OPTIONAL
}
```

Attention

La syntaxe ASN.1 permet des déclarations ambiguës !

Bibliographie

Documents normatifs :

- ITU-T X.680 (ISO/IEC 8824-1) : Abstract Syntax Notation One (ASN.1) : Specification of Basic Notation
- ITU-T X.681 (ISO/IEC 8824-2) : Abstract Syntax Notation One (ASN.1) : Information Object Specification
- ITU-T X.682 (ISO/IEC 8824-3) : Abstract Syntax Notation One (ASN.1) : Constraint Specification

Divers :

- Dubuisson O., *ASN.1 Communication entre systèmes hétérogènes*, Springer, ISBN 978-2-287-59670-4
- PowerASN, *Introduction to ASN.1*

Partie II

Marquage en ASN.1

Syntaxe évoluée
Lever les ambiguïtés
Améliorer les performances

Déclarations et initialisations

```
DeuxEntiers ::= SEQUENCE {
  valeur1     INTEGER OPTIONAL,
  valeur2 [0] INTEGER OPTIONAL
}
```

```
EntierPrive ::= [PRIVATE 12] INTEGER
valeur-avancee ::= EntierPrive 99
```

Équivalente à :

```
valeur-avancee ::= [PRIVATE 12] INTEGER 99
```


Gains du marquage

- Identification
- Simplification du décodage

Classes ASN.1

- UNIVERSAL
- CONTEXT
- PRIVATE
- APPLICATION

```
BOOLEAN         =  [UNIVERSAL 1]
INTEGER         =  [UNIVERSAL 2]
SEQUENCE (OF)   =  [UNIVERSAL 16]
SET (OF)        =  [UNIVERSAL 17]
```


Marquage implicite

```
DeuxEntiers ::= SEQUENCE {
  val1 [0]         IMPLICIT INTEGER OPTIONAL,
  val2 [PRIVATE 0] IMPLICIT INTEGER OPTIONAL
}
```

Définition

Se substitue à un type universel en remplaçant sa marque initiale

Marquage explicite

```
DeuxEntiers ::= SEQUENCE {
  val1 [0]         EXPLICIT INTEGER OPTIONAL,
  val2 [PRIVATE 0] EXPLICIT INTEGER OPTIONAL
}
```

Définition

Ajoute la nouvelle marque au marquage initial. L'encodage peut contenir plus d'informations que précédemment

Valeurs optionnelles – déclaration invalide

```
Ambigu ::= SEQUENCE {
  val1 INTEGER OPTIONAL,
  val2 INTEGER OPTIONAL
 }
```

Valeurs optionnelles – version corrigée

```
NonAmbigu ::= SEQUENCE {
  val1      INTEGER OPTIONAL,
  val2 [0] INTEGER OPTIONAL
}
```


Valeurs par défaut – déclaration invalide

```
Ambigu ::= SEQUENCE {
  version INTEGER DEFAULT v1(1),
  age     INTEGER (1..130) OPTIONAL,
  nom     Chaine,
  prenom  Chaine
}
```


Valeurs par défaut – version corrigée

```
NonAmbigu ::= SEQUENCE {
  version      INTEGER DEFAULT v1(1),
  age     [0] INTEGER (1..130) OPTIONAL,
  nom          Chaine,
  prenom       Chaine
}
```


Première ambiguïté de type – déclaration invalide

```
Ambigu ::= SEQUENCE {
  val1 INTEGER OPTIONAL,
  val2 INTEGER
 }
```

Première ambiguïté de type – version corrigée

```
NonAmbigu ::= SEQUENCE {
  val1 [0] INTEGER OPTIONAL,
  val2     INTEGER
}
```


Seconde ambiguïté de type – déclaration invalide

```
Ambigu ::= SET {
  val1 SEQUENCE OF INTEGER,
  val2 Individu
}
```


Seconde ambiguïté de type – version corrigée

```
NonAmbigu ::= SET {
  val1 [0] SEQUENCE OF INTEGER,
  val2     Individu
}
```


Ambiguïté de l'OpenType – déclaration invalide

```
Ambigu ::= CHOICE {
  val1 OpenType,
  val2 INTEGER
}
```

Ambiguïté de l'OpenType – version corrigée

```
NonAmbigu ::= CHOICE {
  val1 [0] OpenType,
  val2     INTEGER
}
```


Principe du marquage automatique

Ajout automatique de :

- Numéros incrémentaux
- `CONTEXT`
- `EXPLICIT`

Déclaration de module en ASN.1 97

```
ModuleNonAmbigu
DEFINITIONS
AUTOMATIC TAGS -- Marquage automatique
  ::=
BEGIN
  DeuxEntiers ::= SEQUENCE {
    valeur1 INTEGER OPTIONAL,
    valeur2 INTEGER OPTIONAL
  }
END
```


Déclaration manuelle équivalente

```
DeuxEntiers ::= SEQUENCE {
  valeur1 [0] EXPLICIT INTEGER OPTIONAL,
  valeur2 [1] EXPLICIT INTEGER OPTIONAL
}
```

Marquage d'alternatives

```
ModuleNonAmbigu
DEFINITIONS
AUTOMATIC TAGS -- Marquage automatique
  ::=
BEGIN
  UneAlternative ::= CHOICE {
    val1 BOOLEAN,
    val2 REAL
  }
END
```


Déclaration manuelle équivalente

```
UneAlternative ::= CHOICE {
  val1 [0] EXPLICIT BOOLEAN,
  val2 [1] EXPLICIT REAL
}
```


Critique du marquage

Avantages :

- Évite les ambiguïtés
- Améliore les performances

Inconvénients :

- Complique la notation
- Concepts peu évidents à appréhender

La version ASN.1 97 limite ces inconvénients

Marquage et encodage

Spécifications abstraites génériques

Théorie :

- Pas de syntaxe de transfert particulière visée
- Les syntaxes de transfert peuvent utiliser le marquage

Pratique :

- Connaître les syntaxes de transfert et le fonctionnement des décodeurs
- Spécifications ASN.1 97 gouvernent les processus de transfert
- Seule manière viable ?

Bibliographie

Documents normatifs :

- ITU-T X.680 (ISO/IEC 8824-1) : *Abstract Syntax Notation One (ASN.1) : Specification of Basic Notation*
- ITU-T X.681 (ISO/IEC 8824-2) : *Abstract Syntax Notation One (ASN.1) : Information Object Specification*
- ITU-T X.682 (ISO/IEC 8824-3) : *Abstract Syntax Notation One (ASN.1) : Constraint Specification*

Divers :

- Larmouth J., *ASN.1 Complete*, Morgan Kaufman Publishers, ISBN 0-12-233-435-3
- Dubuisson O., *ASN.1 Communication entre systèmes hétérogènes*, Springer-Verlag, ISBN 2-287-59670-4
- PowerASN, *ASN.1 Tagging Principles*

Partie III

Encodages standards

Syntaxes de transfert standards
TLV et XML

Types de syntaxes de tranfert

- Syntaxes basées sur les bits
- Syntaxes basées sur les octets
- Syntaxes basées sur les caractères

Principes d'encodage bit à bit

- Limiter les transferts d'information
- Spécification du protocole utilisé requise

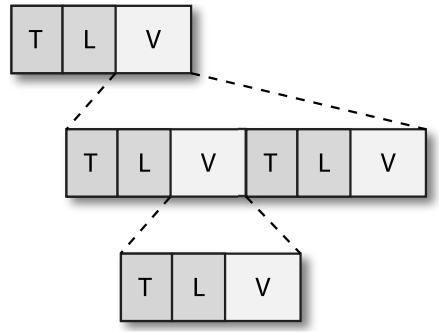
PER : Packed Encoding Rules

Deux variantes :

- Aligned PER
- Unaligned PER

Principes d'encodage TLV

T ype
L ongueur
V aleur



Remarque

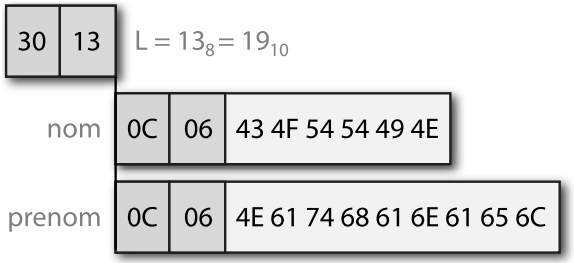
Le champ V peut être lui-même être encodé en TLV

Exemple complet

```
Individu ::= SEQUENCE {
  nom     UTF8String,
  prenom  UTF8String
}
```

```
ncottin ::= Individu {
  nom     "COTTIN",
  prenom  "Nathanael"
}
```

L'encodage TLV de l'instance *ncottin* est :



BER : Basic Encoding Rules

Définition

Historiquement les premières règles d'encodage ASN.1
Définissent des règles laxistes d'encodage TLV

⇒ Besoin de règles d'encodage plus strictes

CER : Canonical Encoding Rules

Définition

BER avec contraintes suivantes :

- Longueur encodée sous forme définie pour les types primitifs et sous forme indéfinie pour les types construits
- Booléen `VRAI` encodé en `11111111`
- Bits inutilisés d'un `BIT STRING` initialisés à `0`
- Les composants des `SET` sont encodés dans l'ordre ascendant de leur marquage
- Les composants des `SET OF` sont encodés dans l'ordre ascendant de leur encodage

DER : Distinguished Encoding Rules

Nées des contraintes imposées par les standards de sécurité et notamment ceux relatifs au format X.509 de l'IETF

Définition

CER avec contraintes suivantes :

- Longueur encodée sous forme définie uniquement
- `BIT STRING` encodé sous forme primitive uniquement

L'objectif est d'assurer un encodage unique

Principes d'encodage XML

- Représenter les données ASN.1 de manière lisible
- Réutiliser les codecs existants
- Bénéficier de la puissance d'XMLSchema

XER : XML Encoding Rules

- Ajout d'une section `ENCODING-CONTROL`
- Chaque élément ASN.1 est nommé
- Les types structurés peuvent être représentés par des listes
- Les `OCTET STRING` sont représentés en base64
- Les attributs XML ne sont pas supportés

E-XER : Extended XML Encoding Rules

- Nouvelles directives `ENCODING-CONTROL`
- Support des attributs XML
- Chaque liste simple est remplacée par une valeur unique en base64
- Meilleure intégration des espaces de nommage

Bibliographie

Documents normatifs :

- ITU-T X.690 (ISO/IEC 8825-1) : *ASN.1 Encoding Rules : Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*
- ITU-T X.691 (ISO/IEC 8825-2) : *ASN.1 Encoding Rules : Specification of Packed Encoding Rules (PER)*
- ITU-T X.693 (ISO/IEC 8825-4) : *ASN.1 Encoding Rules : Specification of XML Encoding Rules (XER)*

Divers :

- Bull R., *XML Encoding Rules (XER)*, 1999

Partie IV

Sécurité de l'ASN.1

Sécurité des encodages standards
Vulnérabilités des syntaxes de transfert
Recommandations d'implantation de décodeurs TLV et XML

Énoncé général du problème

```
Individu ::= SEQUENCE {
  nom    UTF8String,
  prenom UTF8String
}
```

```
ncottin ::= Individu {
  nom    "COTTIN",
  prenom "Nathanael"
}
```

L'encodage DER de l'instance *ncottin* est :

```
30  13
        0C  06  43 4F 54 54 49 4E
        0C  09  4E 61 74 68 61 6E 61 65 6C
```

⇒ Quel est donc le problème ?

Quelques vulnérabilités d'implantation

Dénis de service :

- Flux non finis
- Encodages tronqués
- Longueurs négatives

Injections de code :

- Débordement de tampon
- Dépassement de mémoire tampon
- Injection interne

Flux infinis

Énoncé de la vulnérabilité

Le décodeur reçoit un encodage BER de longueur indéfinie. L'émetteur envoie suffisament de données pour consommer les ressources allouées au décodeur et générer une erreur

Toujours spécifier une taille maximum et générer une erreur (attendue !) lorsque la taille effectivement reçue dépasse la taille maximum

Absence de fin de flux

Énoncé de la vulnérabilité

Le décodeur reçoit un encodage BER de longueur indéfinie. L'émetteur n'envoie jamais la marque de fin de flux et ne ferme pas le flux. Le décodeur attend indéfiniment

Toujours spécifier un délai d'attente maximum

Longueur négative

Énoncé de la vulnérabilité

Le décodeur reçoit un encodage comportant une longueur définie négative. Le décodeur reçoit cette longueur et procède à l'allocation d'une taille mémoire négative

Toujours vérifier la valeur de taille reçue

Longueur indécodable

Énoncé de la vulnérabilité

Le décodeur reçoit un encodage comportant une longueur définie positive exprimée sur un trop grand nombre d'octets. Soit le décodeur décode cette longueur mais l'interprète comme étant négative, soit il ne parvient pas à déterminer la longueur de la longueur

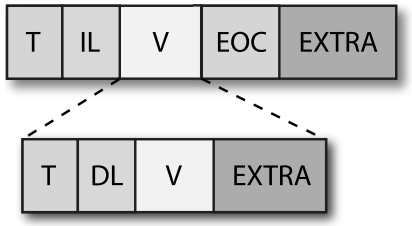
Toujours vérifier que la longueur de la longueur a été correctement décodée en vérifiant :

- Le nombre d'octets utilisés
- La valeur obtenue

Injections propres aux encodages TLV

Énoncé de la vulnérabilité

Du code malicieux est inséré à l'intérieur ou à la fin du flux TLV reçu sans que le décodeur s'en aperçoive. Cette attaque est valable quel que soit le format de la longueur



Solution

Il s'agit ici d'une mauvaise conception du décodeur qui doit :

- Éviter de faire appel à un décodage bufferisé et procéder à un décodage octet par octet
- Vérifier que la taille correspond effectivement au nombre d'octets décodés

Principales vulnérabilités XER

- Flux infinis
- Flux non terminés
- Ajout d'éléments
- Commentaires XML
- Failles de validation XMLSchema (ou DTD)
- Champs exprimés en base64
- Listes de valeurs

Bibliographie

Documents normatifs :

- ITU-T X.693 (ISO/IEC 8825-4) : *ASN.1 Encoding Rules : Specification of XML Encoding Rules (XER)*

Divers :

- Gallagher T., Jeffries B., Landauer L., *Chasser les failles de sécurité*, Microsoft Press, ISBN 978-2-10-050476-3
- PowerASN, *ASN.1 Security Issues*