couche 6/OSI : PRESENTATION

1. Rôle

Permettre à des applications de communiquer en échangeant des données **structurées** dans le cadre d'un dialogue ordonné.

Son rôle est identique à celui du langage dans la communication entre deux personnes: pour se comprendre elles doivent utiliser la même langue c'est à dire une grammaire et un vocabulaire communs.

Ce vocabulaire est constitué par les objets que la couche Présentation va manipuler.

La grammaire correspond aux fonctions s'appliquant aux objets définis par le vocabulaire.

Dans le cadre de l'architecture OSI, la présentation s'interesse à la *syntaxe* **des données échangées.**

L'application, elle, se charge de la sémantique des données.

La difficulté de définir **des structures de données universelles** ou au moins couvrant un grand nombre d'applications à conduit à créer une **syntaxe de Transfert : X409** (Syntaxe abstraite) permettant de décrire n'importe quel type de données aussi bien au niveau Présentation qu'au niveau Application.

Une phase de **négociation** permet aux deux entités communiquantes de choisir la syntaxe qui sera utilisée par la suite. **La représentation en ligne du choix de cette syntaxe et du transfert des données conformémént à cette syntaxe constitue le protocole de présentation.**

2. Service requis.

Pour créer un dialogue ordonné la couche présentation utilise le **service Session**. Dans ce cadre elle fait transiter

Des commandes issues de l'applications en traitant les paramètres qui sont de son ressort.

3. Service fourni.

Il est décrit dans le projet de norme ISO/DIS 8822.

Transfert d'unités de données de protocole Application (APDU) entre entités d'Application. Pour coder les APDU les entités d'Application utilise une **syntaxe abstraite** commune. Les données à transférer sont passées à la couche Présentation dans les paramètres de données utilisateur des primitives de Présentation.

Transformation des données codées selon une syntaxe abstraite depuis ou vers une **syntaxe de Transfert**, acceptable mutuellement par les entité de Présentation, qui assure l'intégité des données à transmettre.

L'encryptage et la compression de données sont caractéristiques de syntaxes de transfert particuliéres.

Négociation d'une **syntaxe de transfert** utilisable. De cette négociation résulte l'association d'une syntaxe abstraite et d'une syntaxe de transfert compatible; cette association constitue un **contexte de présentation.** Vu du service utilisateur de la Présentation (Application) ce contexte de présentation représente un usage spécifique d'une syntaxe abstraite.

4. Eléments du service Présentation.

4.1. Contexte de présentation.

4.1.1 Définition d'un contexte de présentation.

Le service de Présentation fournit des facilités pour la **définition d'un contexte de présentation** adapté aux besoins du transfert d'information. Parfois il est nécessaire de créer plusieurs contextes de présentation pour satisfaire à tous les besoins d'une connection.

Pour définir ces contextes on dispose de deux services élémentaires :

P-CONNECT
P-ALTER-CONTEXT

Ce second service permet aussi de supprimer les contextes devenus inutiles.

Tout contexte de présentation nouvellement créé est ajouté à l'ensemble des contetxtes disponibles pour utilisation immédiate par n'importe quelle connexion.

Lorsque l'ensemble des contextes définis est vide, il doit être possible de transférer les données utilisateur. Pour cela on utilise un contexte par défaut toujours présent. Ce contexte par défaut est aussi requis pour le transfert de données express dans le service
P-EXPEDITED-DATA.

4.1.2 Gestion de l'ensemble des contextes de présentation.

Ce service est fourni en option par le protocole de présentation ISO/DIS8822. Il met en oeuvre deux unités fonctionnelles :

Gestion de contexte et Restairation de contexte.

L'unité fonctionnelle Gestion de contexte est mise en oeuvre à la reception d'une primitive P-ALTER- CONTEXT. Pour être sûr que les contextes de sdeux entités paires sont identiques, ce service est confirmé.

Des précautions doivent être prises lors de modification d'un contexte en cas de Resynchronisation, d'Interruption ou de Reprise d'Activité. En particulier, si la gestion d'Activité est utilisée on doit posséder le Jeton " Synchro Majeure/Activité" pour exécuter la primitive P-ALTER-CONTEXT.

L'unité fonctionnelle Restauration du contexte permet de rétablir un contexte en cours d'utilisation à un Point de Synchronisation en cas de Resynchronisation ou de Reprise d'Activité. Les références des différents contextes utilisés successivement doivent donc être mémorisées. En cas de fin ou d'abandon d'Activité, l'ensemble de contextes en cours est abandonné et un ensemble de contextes hors activité est rétabli. Sur une reprise d'activité l'ensemble des contextes lié à l'activité est rétabli.

4.2 Services élémentaires et primitives

4.2.1 Etablissement de connexion

Ce service permet :

de selectionner les unités fonctionnelles à utiliser
d'établir un ensemble de contextes initial

de définir les caractéristiques du Service Session à utiliser
d'établir la syntaxe abstraite utilisée dans le contexte par défaut.

4.2.2 Terminaison de connexion

Terminaison ordonnée et **non destructive** de la connexion de présentation
ou
Terminaison destructive de la connexion

4.2.3 Gestion de contexte

Définition des contextes et identification locale.
Supression des contextes

4.2.3 Transfert d'information

Données normales gérées par jeton
Données normales sans contrôle
Information de capacité
Données express

4.2.4 Gestion du dialogue

Cette facilité permet de mettre en oeuvre les fonctions de Gestion des jetons, de Synchronisation, de Resynchronisation et de gestion d'Activité. Elle est en correspondance directe avec le service Session et se contente de lui retransmettre les requètes de l'Application en leur ajoutant les paramètres qu'elle détient.

4.2.5 Primitives

Etablissement
P-CONNECT confirmée
Terminaison
P-RELEASE confirmée
P-U-ABORT non confirmée utilisateur
P-P-ABORT non confirmée fournisseur
Gestion de contexte
P-ALTER-CONTEXT confirmée
Transfert de données
P-TYPED-DATA non confirmée pas de jeton
P-DATA non confirmée soumis à jeton
P-EXPEDITED-DATA non confirmée
P-CAPABILITY-DATA confirmée

Gestion du dialogue

P-U-EXCEPTION-REPORT	non confirmée
P-P-EXCEPTION-REPORT	initiée par fournisseur
P-TOKEN-GIVE	non confirmée
P-TOKEN-PLEASE	non confirmée
P-CONTROL-GIVE	non confirmée
P-SYNC-MINOR	confirmation optionnelle
P-SYNC-MAJOR	confirmée
P-RESYNCHRONISE	confirmée
P-ACTIVITY-START	non confirmée
P-ACTIVITY-RESUME	non confirmée
P-ACTIVITY-END	confirmée
P-ACTIVITY-INTERRUPT	confirmée
P-ACTIVITY-DISCARD	confirmée

A titre indicatif les paramètres de la primitive P-CONNECT-request sont indiqués ci-dessous :

Adresse présentation appelante
Adresse présentation appelée
Contextes multiples
Si paramètre absent un seul contexte dans l'ensemble des contextes.
Liste de définition du contexte de Présentation
pour chaque contexte : identificateur et nom de la syntaxe abstraite correspondante.
Nom du contexte par défaut
Qualité de service
Besoins de Présentation
Besoins de Session
Numéro de série initial de point de synchronisation
Assignation initiale des jetons
Identificateur de connexion de Session
Données utilisateur

4.3 Introduction à la compression de données

La compréssion de données est une fonction qui est du ressort de la couche présentation. Toutefois, les quelques élements ci-dessous ne correspondent pas à un projet de norme OSI mais fournissent quelques idées sur le sujet.

La compréssion de données permet de réduire la quantité de données utilisateur à transmettre **sans diminuer la quantité d'information du message**.

On distingue trois niveaux de compression de données.

Compression de premier niveau :

Elle consiste à éliminer du message les caractères non significatifs par exemple des caractères "espace" en fin d'articles de taille fixe

Compression de second niveau :

Elle consiste à coder de manière spécifique et condensée les séquences de caractères identiques, par exemple chaines de * ou de - ou d'espaces. Dans un texte codé en alphbet numèro 5, on peut utiliser un carctère spécial indiquant la répétion suivi du facteur de répétion et du catactère répété.

Cette compression peut aussi porter sur des chaines de bits comportant beaucoup plus de "0" que de "1" (ou l'inverse). On peut alors ne transmettre que le nombre de "0" séparant deux "1" consécutifs avec une numération adéquate..

Compression de troisième niveau :

Elle utilise les propriétés statistiques du message à transmettre et un codage à code de longueur variable adapté à la nature de l'information. Cette compression est fondée sur la redondance des messages qu'ils portent un message "littéraire" ou des données physiques (par exemple un électrocardiogramme).

La fréquence des symboles est mesurée à l'avance et un code optimal calculé pour représenté le message. Ce code ou son identificateur est transmis au collecteur de données.

En transmission, étant donné une probabilité d'erreur de transmission parfois non négligeable on peut utiliser des variantes des codes d'Huffmann , par exemple des codes d'Huffamn-Shannon-Fano autosynchronisants permettant de retrouver un décodage cohérent même après une erreur de transmission résiduelle.

4.4 Introduction à la cryptographie.

La protection des données peut se faire à différents niveaux; la protection la plus efficace se fait de bout en bout et s'effectue au niveau Présentation. Les notions ci-dessous ne sont pas liées aux projets de normes OSI en ce domaine.

4.4.1. Menaces . Types de protection

On peut classifier les menaces selon : l'endroit menacé, noeud ou maille du réseau ou le type d'intrusion : passive ou active.

Les intrusions passives consistent à se brancher en parallèle sur la chaine de transmission et à enregister les données qui circulent.

Lors d'une intrusion active, l'intrus non seulement écoute les communications, mais tente aussi de modifier les échanges en créant,modifiant,supprimant ou retardant tout ou partie des messages.

Les mailles d'un réseau semblent plus fragiles que les noeuds. Cependant les intrusions à un noeud d'un réseau peuvent être beaucoup plus graves .

La protection peut donc porter sur les liaisons de données, elle se fait alors au niveau physique, ou de bout en bout pour protéger toute la chaine de communication.

Pour se protéger contre les intrusions passives un **chiffrement** des données est suffisant.

Contre les intrusions actives il est nécessaire de contrôler l'accès aux données ou la source des messages transmis. On procéde alors à une **authentification de signature.**

remarque : Le chiffrement d'un message consiste à transformer un message clair en un message chiffré (secret) à l'aide d'un code. La récupération de ce message à partir du code et du message secret est faite par une opération de déchiffrement. Le décryptage du message secret permet de retrouver le message clair à partir du seul message secret (sans connaitre le code).

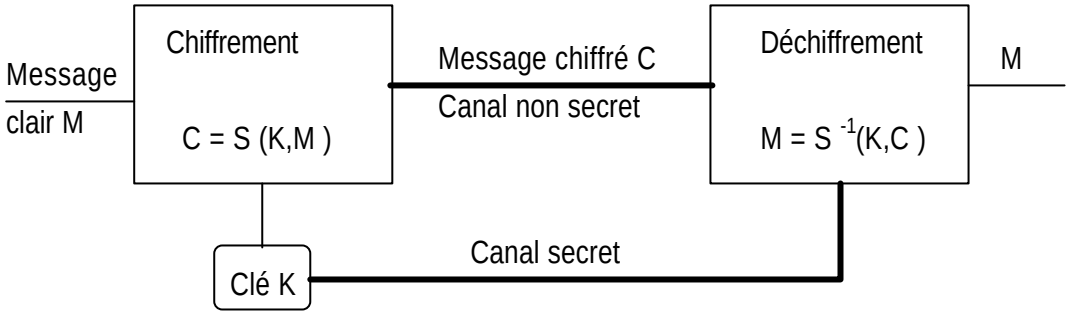
4.4.2 Méthodes de cryptages

Un système de cryptage peut être sûr à cause du coût ou de la durée des calculs nécessaires pour le décoder ou inconditionnellement sûr s'il peut resister à toute analyse du message crypté, même si l'on connait le message clair correspondant.

Le chiffrement peut se faire au vol, sur des blocs de taille indéfinie ou au moins très longs bloc par bloc . La taille des blocs doit être suffisante pour interdire un décryptage en un temps raisonnable par simple analyse de toutes les combinaisons. On travaillera souvent sur des blocs de 64 bits.

Cryptosystèmes à clé unique (privée)

Dans ces systèmes les messages sont cryptés avant d'être transmis sur un canal non secret et la clé de cryptage est transmise sur un canal secret.



La fonction de cryptage S est inversible telle que :

$$S^{-1}(K,C) = S^{-1}(S(K,M) = M$$

La fragilité d'un tel système réside dans le partage de la clé entre source et collecteur.

Les méthodes classiques utilisent généralement une combinaison de cryptage par transposition et de cryptage par substitution . Dans le premier cas les symboles sont permutés et souvent regroupés en mots de taille fixe. Dans le second les symboles sont remplacés par d'autres symboles selon la clé de codage (cf. code de César ...). Ces méthodes sont adaptées à un codage manuel ou par automates rigides électroniques ou électromécaniques.

Exemples :

Le système *DES : Data encryption standard* développé par IBM et adopté par l'administration américaine suit ces principes.

Le cryptage porte sur des blocs d'information de 64 bits et utilise une clé à 56 bits (plus 8 bits de controle) . Entre une transposition initiale et une transposition finale on réalise 16 itérations d'une fonction mélant transposition et subtitution selon l'algorithme suivant : si L_i et R_i désignent les demi-blocs gauche et droit à l'itération i, on calcule

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} + f(R_{i-1},K_i) \text{ modulo } 2$$

où $K_i = KS(i,clé)$

avec KS : fonction "Key Schedule"

Toute la puissance du codage réside en fait dans cette **fonction non-linéaire** qui est réalisée par une table.

Pour cela le champ Ri-1 est étendu de 32 à 48 par duplication de la moitié de ses bits. La clé Ki est ajoutée, puis le champ obtenu est réduit grâce à une table (publique) qui fait

correspondre des champs de 4 bits aux 8 champs de 6 bits obtenus ci-dessus. Cette table est très facile à cabler.

Ce système travaille avec des blocs de 64 bits mais en fait sur des demi-mots. Il est simple à mettre en oeuvre sur des ordinateurs ayant des entiers codés sur 32 bits mais peu rapide.

Cet algorithme est donc beaucoup plus facile à implanter dans un circuit intégré qu'à programmer. Ces circuits existent mais sont soumis à des restrictions de diffusion (interdiction d'exportation, autorisation administrative pour l'achat auprès des fabricants nationaux).

Le déchiffrement utilise le même algorithme.

Ce système présente $2^{56} \# 7{,}2\ 10^{16}$ clés possibles. Sa distance d'unicité n'est que de 17,5. Théoriquement pour casser ce code il suffit de 18 caractères du texte clair et crypté mais un temps de calcul très long.

Les **seuls codes réellement indéchiffrables** sont les codes de VERNAM qui consistent à ajouter un séquence aléatoire au texte clair.

Si cette **clé aléatoire est utilisée** *une seule fois* le code est impossible à traduire.
La clé est retrouvée immédiatement (par un simple ou exclusif) si on possède simultanément les tests clair et chiffré. Ceci implique donc que la clé soit à usage unique et renouvelée pour chaque chiffrement.

On peut aussi utiliser des **codes cycliques** avec des algorithmes de multiplication et de division. Un ou plusieurs polynomes générateurs constituent la clé de cryptage. Un polynome de degré élevé permet de travailler sur des blocs très longs. L'encryptage par multiplication de polynome ajoute N bits aux message clair; l'encryptage par division supprime 16 bits. On pourra donc garder sa taille au message en appliquant successivement les deux algorithmes. On réalise ainsi une substitution sur des blocs très très longs (de plusieurs milliers de bits). Des transpositions peuvent être ajoutées avant ou après les opérations de multiplication ou de division.

Cryptosystèmes à clé publique

Pour éviter les dangers introduits par le partage des clés de cryptage on utilise deux clés différentes pour le chiffrement et le déchiffrement. La clé de chiffrement est publique; la clé de déchiffrement reste secrète et n'appartient qu'au collecteur du message.

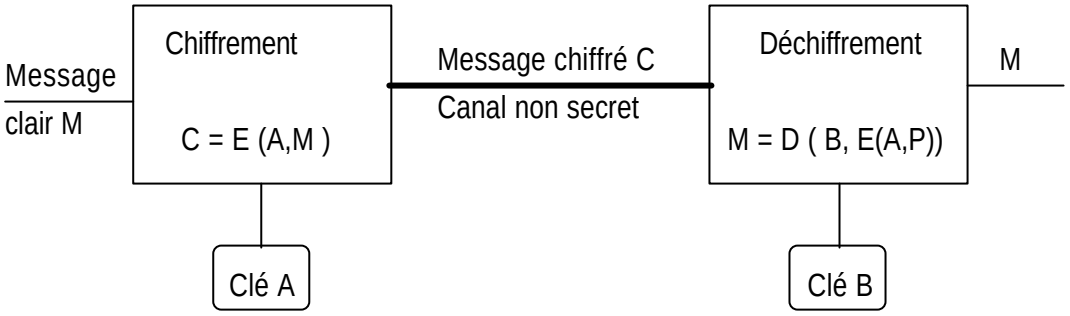
Le principe de ce système est que, d'une part un intrus est incapable de retrouver un message clair même s'il connait la clé publique et le message codé, et que d'autre part, la source du message est incapable de retrouver la clé de déchiffrement même connaissant la clé publique, le message clair et le message codé (décryptage en un temps excessivement long, par balyage

systèmatique par exemple). Seul le détenteur de la clé privée peut restituer toutes les composantes du système.

Pour un sens de transfert, on part d'une clé K et d'un couple d'algorithmes E(K,M) et D(K,C) . E est l'inverse de D . Les conditions suivantes doivent être satisfaites :

- pour tout K , il est possible de trouver un couple d'inverses E(K) et D(K) à partir de K. E et D sont (facilement) calculables

- il est impossible de trouver D à partir de E à cause de la durée ou du coût des calculs



A et B sont determinés par le collecteur à partird'une clé de base K.

L'algorithme E et la clé A sont publics. Ils sont fournis par le collecteur du message à tous ses correspondants. Ils sont calculés en fonction de D et dela clé secrète K.

Exemple 1 : système Rivest - Shamir - Adleman (RSA)

Ce système est fondé sur la décomposition en facteurs premiers de grands nombres. Sa puissance tient au fait que pour déterminer la clé secrète D, un intrus doit pouvoir factoriser un nombre n supérieur à 10^{200}. D'après Rivest cette tâche nécessiterait um million d'années avec le meilleur algorithme connu sur une machine traitant un million d'instructions par seconde.

Pour établir les clés publique et secrète on part d'un nombre aléatoire E pris dans un intervalle défini ci-dessous et de deux (ou plusieurs) nombres **premiers** distincts p et q très grands ($> 10^{100}$) .

La clé publique est constituée du couple (E,n) avec $n = \prod_j p_j$

Soit $\Phi(n)$ l'indicateur d'Euler de n .

$\Phi(n)$ est le nombre d'entiers premiers avec n qui n'entrent pas dans sa factorisation .
Par exemple
$14 = 2 * 7$ $\Phi(14) = 6$ (à savoir 1,3,5,9,11,13)

Si n n'a que des facteurs p_j $\Phi(n) = \prod_j p_j - 1$

Ainsi $10 = 2 * 5$ et $\Phi(10) = 1 * 4 = 4$

$9 = 1$ modulo 4

2^9 modulo 10 = 2^1 modulo 10 = 2

comme on peut le vérifier :

2^9 modulo 10 = 512 modulo 10 = 2

L'**encryptage** utilise la clé E, nombre aléatoire compris entre 3 et $\Phi(n)$. Il est réalisé de la manière suivante :

Le message clair est codé en une séquence de chiffres (non nécessairement binaires) et cette séquence segmentée en blocs B_i tels que

$0 \leq B_i < n$.

Pour chaque bloc B_i on réalise la substitution :

$$C_i = B_i^E \text{ modulo } n .$$

C_i est transmis sur le canal non secret.

Pour le **déchiffrement** on détermine la clé secrète D telle que :

$$E * D \text{ modulo } \Phi(n) = 1 .$$

Le calcul de D est identique à celui de $z = w^{-10}$ dans le système précédent.

Remarque : E et n sont publics et servent à calculer la clé secrète D par l'intermédiaire de $\Phi(n)$. Il convient donc de choisir n suffisamment complexe pour qu'il ne soit pas possible de trouver $\Phi(n)$ et en déduire D.

Le message clair est donné par le calcul de

$B_i = C_i^D$ modulo n

Justification :

$C_i = B_i^E$ modulo n

$C_i^D = B_i^{E\,D}$ modulo n = B_i^1 modulo n = B_i

Applications :

L'exemple ci-dessous est donné à titre indicatif et ne correspond pas à un choix de clés judicieux. En effet le terme n est le produit de facteurs trop petits; ceci permet de calculer (trop) simplement la clé de décryptage connaissant E et n.

Calcul des clés :

p = 31 q = 47 soit n = 1457

$\Phi(n) = 30 * 46 = 1380$

clé publique :

$E = 889 \in [3,1380]$

clé secrète :

$D = E^{-1}$ modulo 1380 = 1009

soit 889 * 1009 modulo 1380 = 1

bloc numérique à coder :

B = 234

chiffrement :

$C = B^E$ modulo n = 234^{889} modulo 1457 = 892

déchiffrement :

$B = C^D$ modulo n = 892^{1009} modulo 1457 = 234

autre exemple :

si p_j = 2,3,11,17

n = 2 * 3 * 11 * 17 = 1122

$\Phi(n) = 1 * 2 * 10 * 16 = 320$

si E = 47

$D = E^{-1}$ modulo $\Phi(n) = 143$

alors 234^{47} modulo 1122 = 174

et 174^{143} modulo 1122 = 234

Exemple 2 : système Merckle-Diffie-Hellman

Le message clair est segmenté en blocs de taille n correspondant à la longueur des clés A ou B. n doit être assez grand pour que connaissant uniquement la clé publique, on ne puisse retrouver le message original à partir du message crypté qu'en un temps très très long.

Soit $B = [b_1, b_2, \ldots, b_n]$ la clé *privée* du destinataire
$A = [a_1, a_2, \ldots, a_n]$ la clé *publique* de la source
$M = [x_1, x_2, \ldots, x_n]$ un bloc du message clair

Les coefficients a_i et b_i sont des entiers naturels et les variables x_i les bits du message clair

Les coefficients b_i sont choisis de manière aléatoire avec la condition ci-dessous nécessaire pour déchiffrer simplement C. On impose aux coefficients b_i la propriété:

$$b_i > \sum_{j=0}^{i-1} b_j$$

La clé de chiffrement A, fournie aux correspondants, est calculée à partir de deux entiers naturels w et m premiers entre eux et de B . w et m sont secrets .

On a $a_i = b_i * w$ modulo m

Pour déchiffrer le message C on doit aussi calculer une "gâche" $z = w^{-1}$ telle que z * w modulo m = 1.

Le message crypté est $C = AM = a_1x_1 + a_2x_2 + \ldots + a_nx_n$
C'est un entier naturel.

Au déchiffrement on calcule :

Bx = C * z modulo m = AM * z modulo m
puis on utilise l'algorithme d'empilement suivant :
pour i = n jusqu'à 1

si $b_i > C - \sum_{j=i+1}^{n} x_j b_j$ alors $x_i = O$ sinon $x_i = 1$.

Rq : la somme est supposée nulle si i = n

Justification :

$$Ax = \sum_{i=1}^{n} a_i x_i$$

$$Axz \text{ modulo } m = \sum_{i=1}^{n} a_i x_i z \text{ modulo } m$$

$$= \sum_{i=1}^{n} (a_i z) x_i \text{ modulo } m$$

$$= \sum_{i=1}^{n} b_i x_i$$

$$= Bx$$

Il suffit donc de calculer Axz pour retomber sur un algorithme simple de décodage.

Application simple :

On suppose que le message est décomposé en blocs de 8 bits (beaucoup trop courts en réalité ...)

Clé de déchiffrement secrète :

w = 889 m = 1457 B = [3,7,12,23,47,95,189,377]

Calcul de la "gâche" secrète :

z = 1398 tel que 1 = 889 * 1398 modulo 1457

Calcul de la clé publique :

$a_i = 889 * b_i$

soit A = [1210,395,469,49,987,1406,466,43]

Premier bloc du message à coder :

"M" soit 01001101

Premier bloc du message chiffré :

C = 395 + 987 + 1406 + 43 = 2831

Bloc déchiffré :

Bx = 2831 * 1398 modulo 1457 = 526

Décodage par empilement

soit $x_i = 0$ si $b_i > Bx - \sum_{j=i+1}^{n} x_j b_j$

377 > 526	x_i	= 1
189 > 526 - 377 = 149		= 0
95 > 149		= 1
47 > 149 - 95 = 56		= 1
23 > 56 - 47 = 7		= 0
12 > 7		= 0

$7 \leq 7 = 1$
$3 > 7 - 7 = 0$

On retrouve bien le bloc original

Couche 6-7/OSI : SYNTAXE ABSTRAITE ASN.1

La syntaxe ASN.1 a été définie dans le cadre de la couche **7/OSI** pour fournir une représentation commune standard aux PDU spécifiques des diverses applications . Dans ce cadre elle est utilisée comme syntaxe abstraite .

Elle peut aussi être utilisée dans le cadre de la couche **6/OSI** (Présentation) pour donner une description commune des informations à transmettre tenant compte de différents types de terminaux (alphabet n°5 , vidéotex, télétex, facsimilé,etc.) ou des structures de fichiers .
Elle est alors utilisée comme syntaxe de transfert.

L'OSI a repris et légérement étendu sous le nom de syntaxe ASN.1: Abstract Syntax Numero 1, standards OSI IS/8824 et 8825, la Recommendation X409 du CCITT, créée en 1984 pour coder les PDU de l'Application de messagerie interpersonnelle. Dans le livre bleu (1990) cette recommendation est devenue X208 et X209 ..... L'ancienne version doit être désignée par X409-84.

1. STRUCTURE des PDU

Toutes les PDU sont codées sous forme de **structures d'items imbriqués** , le champ valeur de chaque item pouvant être lui-même un item. Tous ces **items** ont une structure de type **T.L.V.: Type,Longueur,Valeur.**

T: Identificateur codé sur 1 ou plusieurs octets
L: Longueur codée su 1 ou de 2 à 127 octets
V:Champ valeur pouvant atteindre une longueur de 2 puissance 1008 octets.

Ce champ peut contenir uniquement des données utilisateur ou être un autre item.

1.1. Identificateur T :

Champ	Nature	Code
bits 8-7	classe universel	00
	application	01
	contexte	10
	privé	11
bit 6	forme **primitif**	0
	constructeur	1
bits 5 à 1	code valeur	0 à 11110
	extension	11111
octets suivants	si extension valeur	0 à 254
	extension	255

Dans la classe "universel" la syntaxe ASN.1 fourni des types de base '**prédéfinis**" et des types construits "**définis**" souvent rencontrés. La liste ci-dessous, créée pour les applications de messagerie interpersonnels n'est pas limitative et sera complétée au fur et à mesure de la spécification de nouvelles applications.

Le type "étiqueté" sert à créer des types spécifiques à une application ou une partie d'application (contexte).

Tous les codes sont donnés en hexadécimal. Les types "chaîne" peuvent être **primitifs** (codes ci-dessous) ou **constructeurs** (bit 6 à 1; ex: 03 -> 23). Un type constructeur correspond à un type dont le champ "valeur" contient un ou plusieurs autres types. Les types séquence, ensemble, choix sont constructeurs

(exemple : séquence -> code ID = 10h , code réel =30h)

types prédéfinis:

code	types prédéfinis	commentaires
00		Avec longueur 0 fin de contenu
01	booléen	vrai (FF) ou faux (00)
02	entier	Codé binaire en complément à 2
03	chaîne binaire	
04	chaîne d'octets	
05	vide	longueur 00
06	identificateur d'objet	
07	descripteur d'objet	
08	externe	
09	réel	
10	énuméré	
	étiqueté	**type créé** : explicite ou **implicite** code : valeur fixée à la création
	choix	parmi plusieurs possibilités; lié ou non lié code : type choisi + valeur
	quelconque	permet de coder n'importe quelle valeur explicitée par ailleurs ....

code	types définis	commentaires
16	Séquence	suite **ordonnée** d'éléments
17	Ensemble	ensemble **non ordonné** de membres
18	Chaîne numérique	
19	Chaîne imprimable	
20	Chaîne Télétex	
21	Chaîne Vidéotex	
22	Chaîne AI5 (T100)	
23	heure généralisée	date et heure légale
24	heure UTC	temps universel ; terminé par Z
25	Chaîne de caractères graphiques	
26	Chaîne de caractères visibles (T61)	Terminaux Télétex
27	Chaînes de caractères générale	

Nota : d'autres codes de type universel peuvent encore être ajoutés.

1.2. Longueur

Elle peut coder un champ valeur de longueur indéfinie, courte ou longue.

* indéfinie : code 80h; caractère de fin d'item : EOC= 00

* courte :- code 0 à 7Fh; champ valeur de 0 à 127 octets
* longue - code 81h à FFh
1er octet = longueur de la longueur - 1 (0 à 127 octets)
octet suivant = longueur du champ valeur

1.3. Contenu

Le champ V peut être **primitif ou constructeur** . Dans ce cas il possède une structure d'items imbriqués.

2. CREATION DE TYPES

2.1. macro

Il est possible de définir une **notation non standard** au moyen d'une **macro** . Cette notation est spécifiée dans le corps de macro qui est précédé par "begin" et suivi de "end". Chaque nom de macro est écrit en majuscules.

Exemple (d'autres structures sont permises):

ATTRIBUT MACRO ::=
BEGIN
TYPE NOTATION ::= "LIST"<Type::= SEQUENCE OF Chaîne-IA5>/vide<Type::= Chaîne-IA5>
VALUE NOTATION ::= valeur(VALUE Type)
END

Utilisation:

destinataire ATTRIBUT LIST ::={"NOM = J.Dupont","AGE=42"}
ou
destinataire ATTRIBUT ::= "NOM = J.Dupont"

2.2. module

Un **module** sert à regrouper des définitions apparentées de types, valeurs et macros ,par exemple, celles d'une spécification particulière de protocole. Un corps de module est aussi encadré par "BEGIN-END".

Chaque nom de module commence par une majuscule.
Exemple:

```
Couleur DEFINITION ::=
BEGIN
Couleur-primaire::=INTEGER{rouge(0),jaune(1),bleu(2)}
Couleur-Par-Défaut Couleur-Primaire ::= jaune
END
```

3. EXEMPLES :

3.1. Séquence, ensemble, choix, étiquété, quelconque

```
Justificatifs ::= SEQUENCE {

nom-utilisateur Chaîne-IA5,
mot-Passe Chaîne-IA5,
numéro-Compte INTEGER }

Attributs-Fichier ::= SET {          (ensemble)
propriétaire [0] IMPLICIT Nom-Utilisateur,
taille-Contenu-En-Octets [1] IMPLICIT INTEGER,
[2] IMPLICIT Contrôle_Accès.}

Identificateur-Fichier ::= CHOICE {
nom_Relatif  [0] IMPLICIT Chaîne-IA5,
nom_Absolu   [1] IMPLICIT Chaîne-IA5,
numéro_Série [2] IMPLICIT INTEGER }

Nom_Fichier ::= [APPLICATION 8] IMPLICIT SEQUENCE {
nom_Répertoire Chaîne-IA5,
nom_Fichier Chaîne-IA5 }

Contenu_Fichier ::= ANY      (quelconque)
```

3.2. "Message"

L'exemple ci-dessous, tiré de la Recommandation ASN.1, illustre l'utilisation de cette syntaxe. Il donne successivement une description informelle de l'enregistrement à représenter, une description formelle de sa structure puis de la valeur de cet enregistrement , enfin une représentation "commentée" de la valeur de cet enregistrement après codage dans la syntaxe ASN.1.

Exercice: Codage ASN.1d'une structure d'accès

Dans les spécifications FTAM est défini un champ "Context" de type CONTEXT-SPECIFIC. Le troisième champ défini de ce type (valeur 2) est décrit par:

identificateur A2 ([CONTEXT-SPECIFIC 2])
longueur 03
contenu
identificateur 02 ([UNIVERSAL 2] -> INTEGER)
longueur 01
contenu 06

On veut coder le type "AccesStructureType" de valeur

{ structureType hierarchical,
maxDepth 16 }

par une structure universelle "ensemble" (SET champ code 17 ou 11_H) contenant deux champs "Context", pour structureType et maxDepth, dont les valeurs de type sont 0 et 1.

Hierarchical sera codé par un entier de valeur 2.

Donner la description ASN.1 de cette partie de FPDU et la chaîne d'octets effectivement transmise (codée en héxadécimal)

Réponse :

les TLV pour structureType et maxDepth sont de type constructeur "Context"
maxDepth sera codé sur un TLV de type INTEGER de valeur 16
le paramètre hierachical sera aussi codé par un INTEGER de valeur 2.

On place dans un SET deux champs CONTEXT-SPECIFIC construits sur le modèle ci-dessus, soit :

identificateur 31 ([UNIVERSAL 11H -> SET) 0 0 1 10001
longueur 0A
contenu
identificateur A0 ([CONTEXT-SPECIFIC 0] -> StructureType)
longueur 03
contenu
identificateur 02 ([UNIVERSAL 2] -> INTEGER)
longueur 01
contenu 02 (hierarchical = 2)
identificateur A1 ([CONTEXT-SPECIFIC 1] -> maxDepth)
longueur 03
contenu
identificateur 02 ([UNIVERSAL 2] -> INTEGER)
longueur 01

contenu 10 (16)

SOIT 31 0A A0 03 02 01 02 A1 03 02 01 10