M25 : Sécurité des Réseaux

Jean-Marc THIRIET
jean-marc.thiriet@univ-grenoble-alpes.fr
http://www.gipsa-lab.grenoble-inp.fr/
~jean-marc.thiriet/lpro/rims_fr.html
……./lpro/cnms_en.html

Jean-Marc THIRIET
jean-marc.thiriet@univ-grenoble-alpes.fr

M25-1. Détection et correction des erreurs

Jean-Marc THIRIET
jean-marc.thiriet@univ-grenoble-alpes.fr

1.1 Détection d'erreurs

1.2 Parité verticale et longitudinale

1.3 Code cyclique ou polynomiaux (arithmétique polynomiale)

1.4 Les checksums

1.5 Conclusion

Objectifs de ce chapitre

- Revoir les concepts de codage et détection d'erreurs
- Premier niveau de détection de violation d'intégrité
- Réviser le binaire et l'hexadécimal
- Réviser le concept de division modulo (reste de la division entière) très utilisé en cryptographie…

1.1 Détection et correction d'erreurs

- Transmissions de trames
  - S'assurer que le récepteur a reçu correctement les informations
  - Contrôle de **l'intégrité** des données reçues
- Causes des erreurs de transmission
  - Aspects physiques
    - Bruit thermique
    - Perturbations électromagnétiques
    - Distance entre deux éléments (sans fil, mobilité)
  - Caractéristiques des protocoles
    - CSMA/CD
    - Mobilité (réseaux sans fil, téléphonie mobile, objets mobiles)

Type of méthodes

- BER (**Bit error rate**)
  - Erreurs de bit uniques
  - Groupes de bits en erreur
    (Erreurs en rafale / **burst errors**)
- BER : probabilité P qu'un bit unique soit corrompu dans un temps fini
- BER de $P=10^{-3}$ signifie que, en moyenne, 1 bit parmi 10^3 => corrompu
- Pour une chaîne de N bits => $1-(1-P)^N$
  - Avec $P=10^{-3}$ et N=10 =>la probabilité pour la chaîne d'être corrompue => 10^{-2}

Choix de la méthode

- Basée sur
  - La tailles des blocs (paquets, trames)
  - Probabilité d'erreurs
- Ex :
  - Bloc de 1250 octets avec $P=10^{-3}$
    - $1-(1-P)^N = 1-(1-10^{-3})^{1250*8}$= 100 % erreurs !!
  - Bloc de 125 octets avec $P=10^{-3}$
    - $1-(1-P)^N = 1-(1-10^{-3})^{125*8}$= 63 % erreurs ! (plus qu'1 bloc parmi 2 !)
- La longueur du bloc (trame) est trop grande
- La taille maximale d'un paquet à transmettre dépend de la qualité de la communication (taux d'erreur)

Distance de Hamming

- Entre deux mots de même longueur, nombre de positions ayant des symboles différents
  - Ex : 1 0 1 1 1 0 1
    1 **1** **0** 1 1 **1** 1

  Ici la distance de Hamming vaut 3
  - si la distance entre deux mots du code est d, d erreurs peuvent transformer un mot en l'autre
- D'une manière générale, pour détecter "d" erreurs, il faut un code avec une distance de "d+1"

Les codes détecteurs

Exemple : contrôle de parité

parité paire : 10110111

bit parité

parité impaire : 10110110

- Parité verticale
- Parité longitudinale:

	0	1	0	0	0	1	1	**0**
	1	0	1	1	1	0	0	**1**
	1	1	0	1	0	1	0	**1**
LRC	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**

(parité impaire)

	0	1	0	0	0	1	1	**0**
	1	0	1	1	**0**	0	0	**1**
	1	1	0	1	0	1	0	**1**
LRC	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**

Erreur de parité dans une ligne et dans une colonne.

Exercice à rendre 1 (1/2), Ex 1

- Pour lundi 19 octobre
- 1.1
- 10110
- 11011
- 11001
- Ajouter des bits de parité verticalement et horizontalement, en parité paire
- 1.2 Un récepteur reçoit ce message en parité paire
- 111001
- 101110
- 001111
- 001000
- Y a-t-il des erreurs de transmission? Si oui, est-il possible de corriger les erreurs de transmission? Quel est le message correct ? Vous pouvez commenter ce résultat..

Exercice à rendre 1 (1/2), Ex 1 CORRECTION partie 1

- exercice 1.1
- 101101
- 110110
- 110011
- **101000**
- Donnez la parité (parité paire) horizontalement et verticalement

- Un récepteur reçoit en parité paire le message ci-dessous (naturellement le récepteur n'a aucune information concernant le message réel qui a été transmis par l'émetteur)
- 111001 **correct**
- 101110 **correct**
- 001111 **correct**
- 001000 **Incorrect**
- **Il y une erreur dans la dernière ligne (dernier paquet de 6 bits), mais à ce stade nous ne savons pas où est l'erreur…**

Exercice à rendre 1 (1/2), Ex 1 CORRECTION partie 2

- **Vérification des colonnes**
- **clcccc => Il y a une erreur dans la seconde colonne (c=correct ; I=Incorrect)**
- Y a-t-il des erreurs de transmission? Si oui, est-il possible de corriger les erreurs de transmission ?
- **Si nous combinons les informations connues verticalement et horizontalement, nous pouvons localiser quel bit est erroné.**
- 111001
- 101110
- 001111
- 0**0**1000
- **Puisque ce bit n'est pas bon, nous allons l'inverser, donc remplacer ce 0 par un 1. Nous venons de réaliser une correction, qui ne nécessite pas de réémission de la part de l'émetteur**
- Quel est le message correct ?
- **Le message correct initialement transmis est donc :**
- 111001
- 101110
- 001111
- 001000
- Faire un commentaire complémentaire.
- **Nous pouvons noter dans cet exemple que le bit erroné n'était pas un bit de donnée, mais un bit de parité. Cette constatation nous permet de prendre conscience du fait que les mécanismes de sécurité s'appliquent naturellement à tous types de bits de la même manière : les bits de données et les bits de contrôle. … Nous pouvons également noter qu'il n'y avait en fait aucune erreur de transmission dans les bits de données dans cet exemple…**

1.3 L'arithmétique polynomiale modulo 2

- Un polynôme représente une suites de bits:

  110001 ------- > x^5+x^4+1

- Addition et soustraction **sans retenue**: OU EXCLUSIF

	1	0	0	1	1	0	1	1	$x^7 + x^4 + x^3 + x + 1$
+	1	1	0	0	1	0	1	0	$+ x^7 + x^6 + x^3 + x$
=	0	1	0	1	0	0	0	1	$= x^6 + x^4 + 1$

  - **Les codes cycliques: CRC (Cyclic Redundandy Check)**

- **EMETTEUR**
  - **Données (message à transmettre) : suite de bits représentée par M(x)**
  - **L'émetteur divise x^r.M(x) par G(x) avec G(x) de degré r (r + 1 bits), polynôme générateur**
  - **x^r.M(x) = G(x).Q(x) + R(x), le degré du reste R(x) sera au maximum de r-1 (r bits)**
  - **On transmet la trame T(x) = x^r.M(x) + R(x)**
- **RECEPTEUR**
  - **Le récepteur divise T(x) par G(x) et le reste doit être 0**

Polynômes normalisés

- **CRC12 = $x^{12} + x^{11} + x^3 + x^2 + x + 1$**
- **CRC16 = $x^{16} + x^{15} + x^2 + 1$**
- **CRC-CCITT = $x^{16} + x^{12} + x^5 + 1$ (recommandation V41 utilisé dans HDLC)**
- **CRC-32 (Ethernet): = $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + X + 1$**

Exemple de calcul de CRC par l'émetteur

- M : 1101011011
- Polynôme générateur G : 10011,
- Quelle est la taille du reste
- Montrer en faisant la division polynomiale que le reste est ????
- Le message transmis sera donc :
  - 1101011011????

Exemple de calcul de CRC par l'émetteur

- M : 1101011011
- Polynôme générateur G : 10011, donc de degré 4 => le reste sera au plus de degré 3 (un de moins que G) donc une taille de 4 bits
- Montrer en faisant la division polynomiale que le reste est 1110
- Le message transmis sera donc :
  - 11010110111110

Que fait le récepteur ?

- Il reçoit 1101011011**1110** et divise par le polynôme générateur 10011

  ⇒le reste obtenu doit être nul,

  ⇒si ce n'est pas le cas, chaque bit à 1 dans le reste traduit un bit erroné

  - ! A condition de respecter la règle ! Si le reste contient n bits, je peux au mieux détecter et corriger n erreurs, ici 4 !

- Ex précédent : Je reçois 1101001011**1110** avec G(x)=10011 => Peut-on détecter l'erreur ?

Exercice à rendre 1 (2/2), pour lundi 19 octobre 2020

- Exercices 1 et 2 et 6, page 5 dans le poly

- **Ex M25-1.4** On désire transmettre le mot “AB”. Chaque caractère est codé en ASCII sur 7 bits et un 8ème bit de parité est ajouté à droite, en parité paire. Précisez quel contenu binaire doit être transmis.
- Nous allons ensuite calculer le CRC de cette suite binaire. Calculer le CRC émis en utilisant comme polynôme générateur $x^8 + x^3 + 1$

Exercice à rendre 1 (2/2), CORRECTION Ex.1 p. 4 Partie 1

- Dans le code ASCII, le mot “INT” est codé par les 3 caractères de 7 bits (le code ASCII est donné en hexadecimal):
- I → 49 **=> 1001001**
- N → 4E **=> 1001110**
- T → 54 **=> 1010100**
- a) Donner le message transmis, en ajoutant aux caractères une parité paire pour le VRC et une pour le LRC. Nous considérons que le bit de parité est à droite.
- **10010011**
- **10011100**
- **10101001**
- **10100110**

Exercice à rendre 1 (2/2), CORRECTION Ex.1 p. 4 Partie 2

- b) Même question avec une parité impaire
- **10010010**
- **10011101**
- **10101000**
- **01011000**

- **! Lorsque l'on compare les parités paires et impaires, tous les bits sont inversés excepté le bit en bas à droite (à l'intersection entre les parités verticales et horizontales) !**

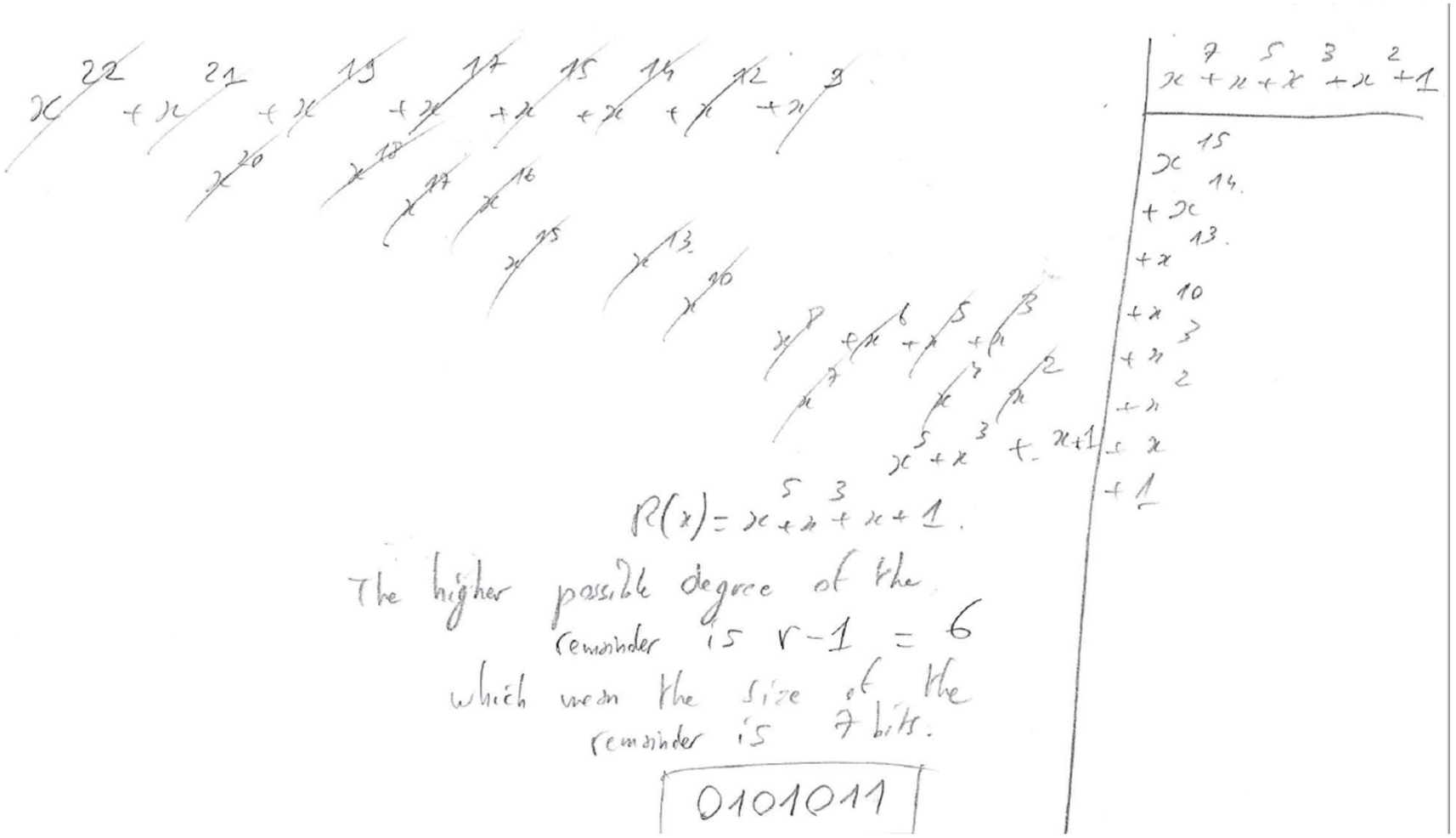
Exercice à rendre 1 (2/2), CORRECTION Ex.2 p. 4 Partie 1

- Pour transmettre le message 11010101 10100100, on souhaite calculer le CRC en utilisant le polynôme générateur équivalent à la suite binaire 10101101.
- Calculer le CRC, précisez quelle est la taille du CRC et pourquoi.

Exercice à rendre 1 (2/2), CORRECTION Ex.2 p. 4 Partie 2

- $M(x) = x^{15} + x^{14} + x^{12} + x^{10} + x^8 + x^7 + x^5 + x^2$
- $G(x) = x^7 + x^5 + x^3 + x^2 + 1$, degré 7
- Donc $x^r.M(x) = x^{22} + x^{21} + x^{19} + x^{17} + x^{15} + x^{14} + x^{12} + x^9$
- Nous devons réaliser la division de $x^r.M(x)$ par $G(x)$

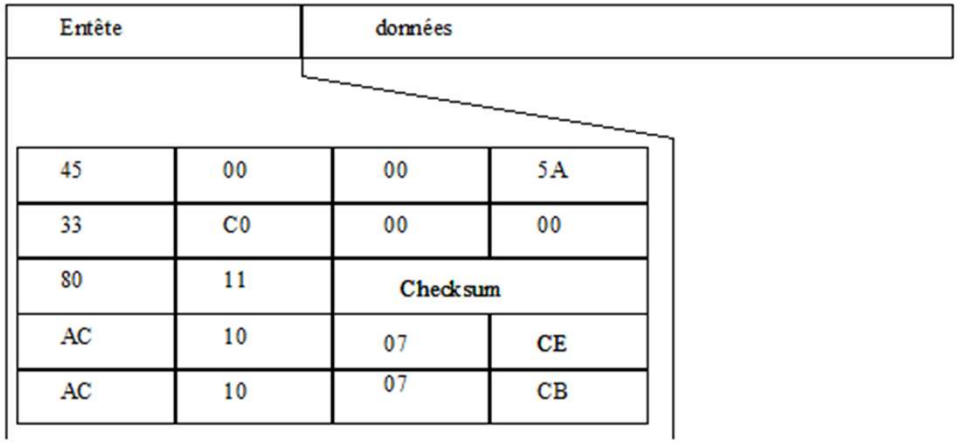
Exercice à rendre 1 (2/2), CORRECTION Ex.2 p. 4 Partie 3



Exercice à rendre 1 (2/2), CORRECTION Ex.6 p. 4 Partie 1

- Exercise 6, page 4 in the textbook

Ex M25-1.6 Un paquet IP est constitué d'un entête de 20 octets et de donnée selon la structure suivante :



45	00	00	5A
33	C0	00	00
80	11	Checksum	
AC	10	07	CE
AC	10	07	CB

Les octets 11 et 12 correspondent au checksum de l'entête.
Les octets 13 à 16 représentent en hexadécimal l'adresse IP de la machine émettrice.
Les octets 17 à 20 représentent en hexadécimal l'adresse IP de la machine destinataire

a) Calculer le checksum sur 16 bites de l'ensemble de l'en-tête (les 18 octets),
b) Déterminer les adresses IP source et destination en notation décimale

Exercice à rendre 1 (2/2), CORRECTION Ex.6 p. 4 Partie 2

- Calculer le checksum
- 4500 + 005A + 33C0 + 0000 + 8011 + AC10 + 07CE + AC10 + 07CB
- 455A + 33C0 + 8011 + B3DE + B3DB
- 791A+133EF + B3DB
- 1AD09 + B3DB = 260E4

- Ensuite nous ajoutons la retenue : 60E4 + 2 = 60E6

- Ensuite nous devons inverser les bits, nous pouvons le faire directement en hexadécimal (0 est l'inverse de F, 1 de E, 2 de D …, 7 de 8) => cf l'avant-dernière planche de cette présentation
- Nous pouvons également utiliser la forme binaire
- 60E6 = 0110 0000 1110 0110
- L'inverse bit à bit est 1001 1111 0001 1001 = 9F19
- Donc le Cheksum vaut 9F19

Exercice à rendre 1 (2/2), CORRECTION Ex.6 p. 4 Partie 3

- b) Déterminer les adresses IP source et destination en notation décimale
- Adresse IP source AC1007CE =>
  - ACh corresponds to 172d
  - 10h corresponds to 16d
  - 07h corresponds to 7d
  - CEh corresponds to 206d
- Donc l'IP source est 172.16.7.206
- De la même manière nous pouvons determiner l'IP destination
- AC1007CB will corresponds to 172.16.7.203

1.4 Le Checksum

- RFC 791:
  - *The checksum field is the 16-bit one's complement of the one's complement sum of all 16-bit words in the header.*
- Les en-têtes IP et TCP utilise une méthode facile à implémenter : CHECKSUM (*somme de contrôle*).
- D'après la RFC 1071 qui définit les méthodes de calcul dans l'environnement IP
  - Prenez un mot de 16-bits
  - Calculez la *somme complément à 1* (! Somme « normale » à laquelle la retenue est directement ajoutée au résultat !)

  - Puis, déterminer le *complément à 1* du résultat (! Inversion de tous les bits du résultat!)
  - Finalement ce résultat doit être placé dans le champ checksum

Les Checksum

- Pour vérifier le bon déroulement de la transmission, la somme en complément à 1 est calculée sur le même flux d'octets y compris le checksum. Si le résultat est 0000 après l'inversion finale, cela signifie qu'il n'y a pas d'erreurs !
- Remarque :
  - Le **complément à 1** est le nombre obtenu par inversion de tous les bits
  - !! La **somme en complément à 1** est une somme dans laquelle la retenue est elle-même ajoutée au résultat !!
- Ex : Calcul de 0xE + 0x3 sur 4 bits en somme complément à 1
  - Calcul normal 0xE + 0x3 = 0x11
  - Calcul en somme complément à 1 sur 4 bits de 0xE + 0x3 = 0x2 = $(0010)_2$
- Ex : Le complément à 1 de ce résultat vaut $(1101)_2$ soit 0xD

Les Checksum

- Exemple :
- Soit le paquet : 01 00 F2 03 F4 F5 F6 F7 00 00 (00 00 est le champ checksum)
- Formons les mots 16 bits :
- 0100 F203 F4F5 F6F7
- Calculons la somme :
- 0100 + F203 + F4F5 + F6F7

Les Checksum

- Exemple :
- Soit le paquet : 01 00 F2 03 F4 F5 F6 F7 00 00 (00 00 est le champ checksum)
- Formons les mots 16 bits :
- 0100 F203 F4F5 F6F7
- Calculons la somme :
- 0100 + F203 + F4F5 + F6F7 = 0002 DEEF (Cette somme est stockée dans un mot 32 bits)
- Additionnons la retenue pour obtenir la somme en complément à 1 :
- DEEF + 002 = DEF1

Les Checksum

- Exemple :
- Soit le paquet : 01 00 F2 03 F4 F5 F6 F7 00 00 (00 00 est le champ checksum)
- Formons les mots 16 bits :
- 0100 F203 F4F5 F6F7
- Calculons la somme :
- 0100 + F203 + F4F5 + F6F7 = 0002 DEEF (Cette somme est stockée dans un mot 32 bits)
- Additionnons la retenue pour obtenir la somme en complément à 1 :
- DEEF + 002 = DEF1
- Le checksum est le complément à 1 du résultat :
- ~DEF1 = 210E
- Le paquet envoyé inclut le checksum :
- 01 00 F2 03 F4 F5 F6 F7 21 0E
- A la réception :
- 0100+F203+ F4F5+ F6F7 + 210E =

Les Checksum

- Exemple :
- Soit le paquet : 01 00 F2 03 F4 F5 F6 F7 00 00 (00 00 est le champ checksum)
- Formons les mots 16 bits :
- 0100 F203 F4F5 F6F7
- Calculons la somme :
- 0100 + F203 + F4F5 + F6F7 = 0002 DEEF (Cette somme est stockée dans un mot 32 bits)
- Additionnons la retenue pour obtenir la somme en complément à 1 :
- DEEF + 002 = DEF1
- Le checksum est le complément à 1 du résultat :
- ~DEF1 = 210E
- Le paquet envoyé inclut le checksum :
- 01 00 F2 03 F4 F5 F6 F7 21 0E
- A la réception :
- 0100+F203+ F4F5+ F6F7 + 210E = 0002 FFFD
- FFFD + 0002 = FFFF
- Inversion FFFF => 0000
- Ce qui est correct !

1.5 Conclusion

- BER
- Taille des blocs (trames)
- La Parité est meilleure pour détecter des erreurs aléatoirement distribuées
- CRC est meilleur pour les erreurs en rafale
- On estime que 999 erreurs sur 1000 sont corrigées grâce au CRC
  - Si le BER sur le medium physique est de 10^{-6}, il passe à 10^{-9} grâce au CRC

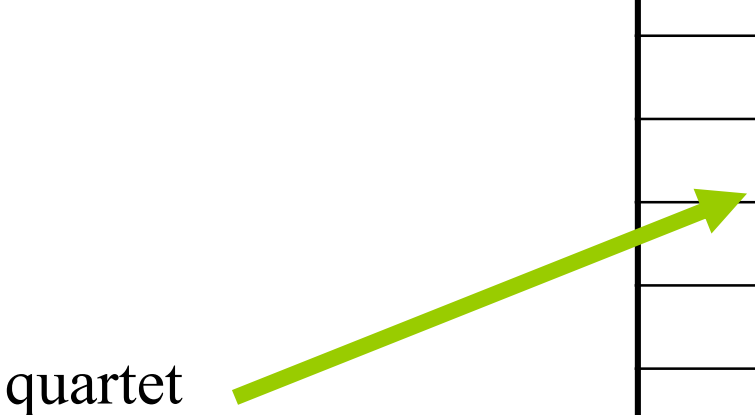
Conversion rapide décimal/binaire/hexadecimal

- Décimal => binaire
  - 125

– 2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
– 128	64	32	16	8	4	2	1
– 0	1	1	1	1	1	0	1

Conversion rapide décimal/binaire /hexadecimal

Base 2	Base 16	Base 10
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15



Références

- Transmissions et réseaux, S. Lohier & D. Présent, Dunod, Paris, 2003.