UNIVERSITÉ
DE REIMS
CHAMPAGNE-ARDENNE

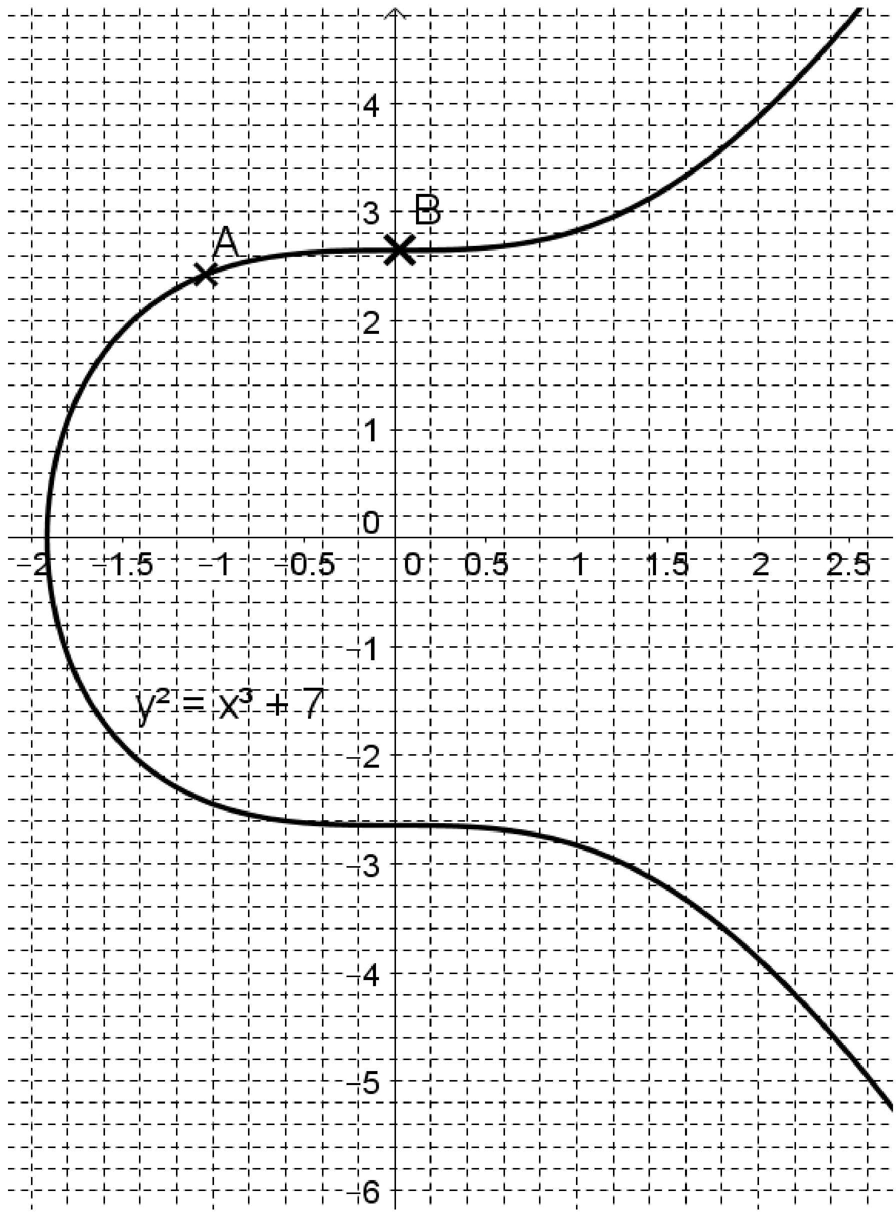
Licence Informatique
Info0603 – 2021/2022

Travaux dirigés et TP n° 8

Courbes Elliptiques et BlockChain

Dans tout le TP on n'utilisera que la courbe elliptique du bitcoin secp256k1 :$Y^3 + x^2 + 7$

Attention les représentations des blockchain présentes sur cette feuille sont toutes incomplètes et présente même des lacunes assimilables à des erreurs.



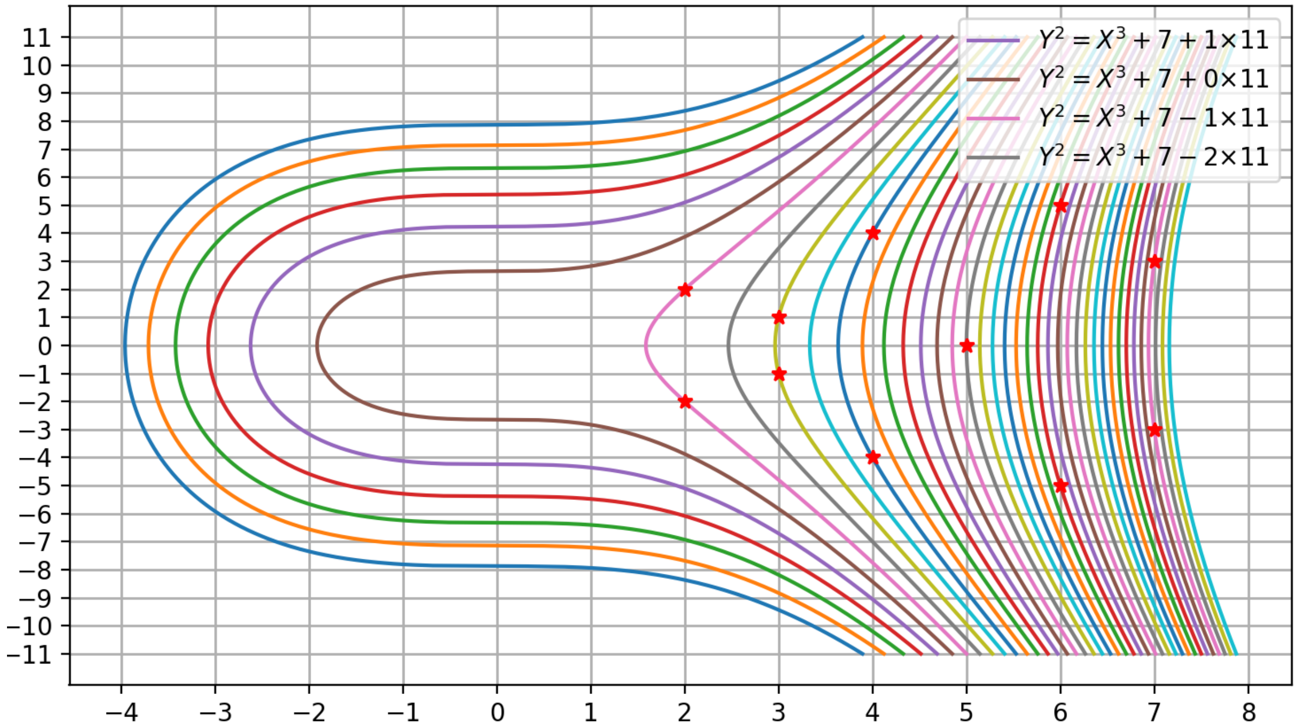
Exercice 1 (Premiers calculs sur les courbes elliptiques)

1. Calculer graphiquement A+B, 2.A, (A+B)-A.
2. Que peut-on dire de A+(-A) ?
3. Facultatif : Déterminer la formule des coordonnées de A+B et de 2.A
4. Calculer de nouveau les coordonnées des points précédents mais cette fois à partir des formules suivantes :
   Calcul de A+B : avec $\lambda = \frac{y_B - y_A}{x_B - x_A}$ $x = \lambda^2 - x_A - x_B$ et $y = \lambda x + y_A$ et $y = \lambda x + y_A$
   Calcul de 2.A : avec $\lambda = \frac{3.x_A^2}{2.y_A}$ $x = \lambda^2 - 2.x_A)$ et $y = \lambda x + yA$

Exercice 2 (Calculs sur F_{11})

1. Facultatif : Déterminer la formule des coordonnées de A+B et de 2.A

   Solution : *On a* $y_A^2 = x_A^3 + 7$,$y_B^2 = x_B^3 + 7$,$(AB) : Y = \lambda.(X - x_A) + y_A$ *avec* $\lambda = \frac{y_B - y_A}{x_B - x_A}$
   On a $M(x, y) \in (AB) \cap E$
   donc $y^2 - y_A^2 = x^3 - x_A^3$
   donc $(y - y_A).(y + y_A) = x^3 - x_A^3$
   donc $(\lambda.(x - x_A).(\lambda.(x - x_A) + 2.y_A) = (x - x_A).(x^2 + x.x_A + x_A^2)$
   et $(\lambda.(x - x_B).(\lambda.(x - x_B) + 2.y_B) = (x - x_B).(x^2 + x.x_B + x_B^2)$
   donc $\lambda^2.(x_B - x_A) + 2.\lambda(y_A - y_B) = x.(x_A - x_B) + (x_A^2 - x_B^2)$
   donc $x.(x_A - x_B) + (x_A - xB).(x_A + x_B) = \lambda^2.(x_B - x_A) + 2.\frac{(y_A - y_B)^2}{x_A - x_B)}$
   donc $x + (x_A + x_B) = -\lambda^2 + 2.\frac{(y_A - y_B)^2}{(x_A - x_B)^2}$
   donc le point d'intersection a pour coordonnées :$x = \lambda^2 - x_A - x_B$ *et* $y = \lambda(x - x_A) + y_A$
   donc $A + B : (\lambda^2 - x_A - x_B\ ; -\lambda(x - x_A) - y_A)$
   Le coefficient directeur de la tangente en A est donné par la dérivée de $(\sqrt{x^3 + 7})' = \frac{3.x^2}{2.\sqrt{x^3+7}} = \frac{3.x^2}{2y}$
   On a donc $M(x, y) \in TangenteenA \cap E$
   donc $y_A^2 = x_A^3 + 7$,$y^2 = x^3 + 7$,$y = \lambda.(x - x_A) + y_A$ *avec* $\lambda = \frac{3.x_A^2}{2.y_A}$
   $y^2 - y_A^2 = x^3 - x_A^3$
   $(y - y_A).(y + y_A) = x^3 - x_A^3$
   donc $(\lambda.(x - x_A).(\lambda.(x - x_A) + 2.y_A) = (x - x_A).(x^2 + x.x_A + x_A^2)$
   donc $(\lambda^2.(x - x_A) = (x^2 + x.x_A + x_A^2) - 2\lambda.y_A$
   donc $(\lambda^2.(x - x_A) = (x^2 + x.x_A + x_A^2) - 2\frac{3.x_A^2}{2.y_A}.y_A$
   donc $(\lambda^2.(x - x_A) = x^2 + x.x_A - 2x_A^2$
   donc $(\lambda^2.(x - x_A) = (x - x_A).(x + 2.x_A)$
   donc $x = \lambda^2 - 2.x_A)$ *et* $y = \lambda(x - x_A) +_y A$ *donc* $2.A : (\lambda^2 - 2.x_A; -\lambda(x - x_A) - y_A)$
2. Que peut-on dire,sur F_{11}, des points de coordonnées $A : (2, 2), B(3, 1), C(5, 3)$?
3. Calculer sur F_{11} A+B, 2.A,3.A, (A+B)-A.
4. Que peut-on dire de A+(-A) ?
5. Représenter vos résultats sur le graphique suivant



Solution : *C n'appartient pas à la courbe ! ElemtE07(2,2,11), ElemtE07(3,1,11) ElemtE07(4,4,11) ElemtE07(5,0,11) ElemtE07(6,5,11) ElemtE07(7,3,11)*

*»> ElemtE07(2,2,11)*2 == ElemtE07(5,0,11)*

*»> ElemtE07(2,2,11)+ElemtE07(3,1,11)== ElemtE07(7,3,11) »> ElemtE07(2,2,11)*3== ElemtE07(2,9,11)*

Exercice 3 (Classe ElemtE07)
Construire les méthodes de la classe des éléments de secp256k1 :$Y^3 + x^2 + 7$ dans F_p

```
class ElemtE07(object):
"Ensemble des solutions de Y²=X^3+7 dans Fp Courbe secp256k1"
def __init__(self,x,y=None,p=None):
Défini par deux ElmntZnZ mais seul le modulo de x est utilisé. Celui de y doit donc lui être égal.
Avec l'élément neutre ayant self.y='Inf'
ElemtE07(7,6,11) doit renvoyer une erreur
>>> ElemtE07(ElmtZnZ(7,11),ElmtZnZ(8,11))
ElemtE07(7,8,11)
>>> ElemtE07(1,"Inf",11)
ElemtE07(1,"INF",11)

def lDesElements(p=47):
>>> ElemtE07.lDesElements(5)
[ElemtE07(0,"INF",5), ElemtE07(2,0,5), ElemtE07(3,2,5), ElemtE07(3,3,5), ElemtE07(4,1,5), ElemtE07(4,4,5)]
>>> len(ElemtE07.lDesElements(11))
12
>>> ElemtE07(6,5,11) in (ElemtE07.lDesElements(11))
True
```

Une fonction de hash **injective** permettra de mettre des instances de ElemtE07 dans des ensembles mais aussi de programmer facilement des chiffreurs utilisant les courbes elliptiques.

Pour gagner en efficacité on pourra utiliser les ElmntZnZ qui comporte maintenant les fonctions estUnCarre et racineCarree.

```
def __hash__(self):
 On fera une fonction injective afin de l'utiliser également dans binCode

def ElemtE07DepuisHash(h,p):
>>> h=ElemtE07(6,5,11).__hash__()
>>> ElemtE07.ElemtE07DepuisHash(h,11)
ElemtE07(6,5,11)

def eDesElements(p=47,verbose=False):
>>> ElemtE07(8,3,17) in (ElemtE07.eDesElements(17))
True
```

Evidemment, les méthodes de calcul.

```
def __add__(self,other):
>>> ElemtE07(2,2,11)+ElemtE07(3,1,11)
ElemtE07(7,3,11)
>>> (ElemtE07(3,"INF",47)+ElemtE07(3,9,47))+ElemtE07(3,"INF",47)
ElemtE07(3,9,47)

def double(self):
>>> ElemtE07(2,2,11).double()
ElemtE07(5,0,11)

def lOrbite(self):
>>> ElemtE07(2,2,11).lOrbite()
[ElemtE07(2,2,11), ElemtE07(5,0,11), ElemtE07(2,9,11), ElemtE07(0,"INF",11)]

def __mul__(self,other):
>>> ElemtE07(6,5,11)*3
ElemtE07(5,0,11)
>>> ElemtE07(15,13,17)*0
ElemtE07(0,"INF",17)

def __rmul__(self,other):
>>> 2*ElmtZnZ(3,10)
ElmtZnZ(6,10)
2*(ElemtE07(3,"INF",47)+3*ElemtE07(3,9,47))+ElemtE07(3,"INF",47)
ElemtE07(43,32,47)

def __eq__(self,other):
>>> 3*ElemtE07(6,5,11)==ElemtE07(5,0,11) and ElemtE07(0,"Inf",47)==0
True
>>> ElemtE07(3,9,47)==ElemtE07(3,"Inf",47) or ElemtE07(3,"Inf",47)==ElemtE07(3,9,47)
False

def __neg__(self):
>>> -ElemtE07(7,3,11)== ElemtE07(7,8,11)

def __sub__(self,other):
>>> ElemtE07(3,10,11)-ElemtE07(7,3,11)==ElemtE07(4,7,11)
>>> ElemtE07(3,9,47)-ElemtE07(3,9,47)==0
```

Les méthodes de Groupe (Facultatif).

Ces méthodes se révèlent très lente dès que l'on passe à 16bits.

Proposer des suggestions pour accélérer tout l'ensemble.

```
def ordreCourbe(p=17):
>>> ElemtE07.ordreCourbe(11)== 12

def ordrePoint(self):
>>> ElemtE07(3,10,11).ordrePoint()== 3
>>> ElemtE07(7,3,11).ordrePoint()== 12

def estGenerateur(self):
>>> ElemtE07(7,3,11).estGenerateur()
True
>>> ElemtE07(3,10,11).estGenerateur()
False

def lDesElementsGenerateurs(p=47):
>>> ElemtE07.lDesElementsGenerateurs(11)
[ElemtE07(4,4,11), ElemtE07(4,7,11), ElemtE07(7,3,11), ElemtE07(7,8,11)]

def lDesElementsDOrdrePremier(p=47):
>>> ElemtE07.lDesElementsDOrdrePremier(11)
[ElemtE07(3,1,11), ElemtE07(3,10,11), ElemtE07(5,0,11)]

def elemtE07APartirDeX(x:ElmtZnZ):
Renvoie un point avec x ou une valeur proche de x comme abscisse
>>> ElemtE07.elemtE07APartirDeX(ElmtZnZ(2,11))
ElemtE07(2,2,11)

def randElemtE07(p):
 Renvoie un élément non nul au hasard

def randGenerateurE07(p=47):
 Renvoie un élément non nul au hasard
>>> ElemtE07.randGenerateurE07(47).estGenerateur()
True
```

Et enfin un graphique ! (Facultatif)

Exercice 4 (Codage à clé publique avec secp256k1 :$Y^3 + x^2 + 7$ **)**
Construisez la class CodeurE07 bassé sur secp256k1. Pourquoi une valeur par défaut d'un générateur est-elle proposée ? L'idée est de coder l'info dans l'abscisse de points de secp256k1. Quel problème cela pose-t-il ?

Solution : *Parce que les générateurs sont très durs à trouver !*

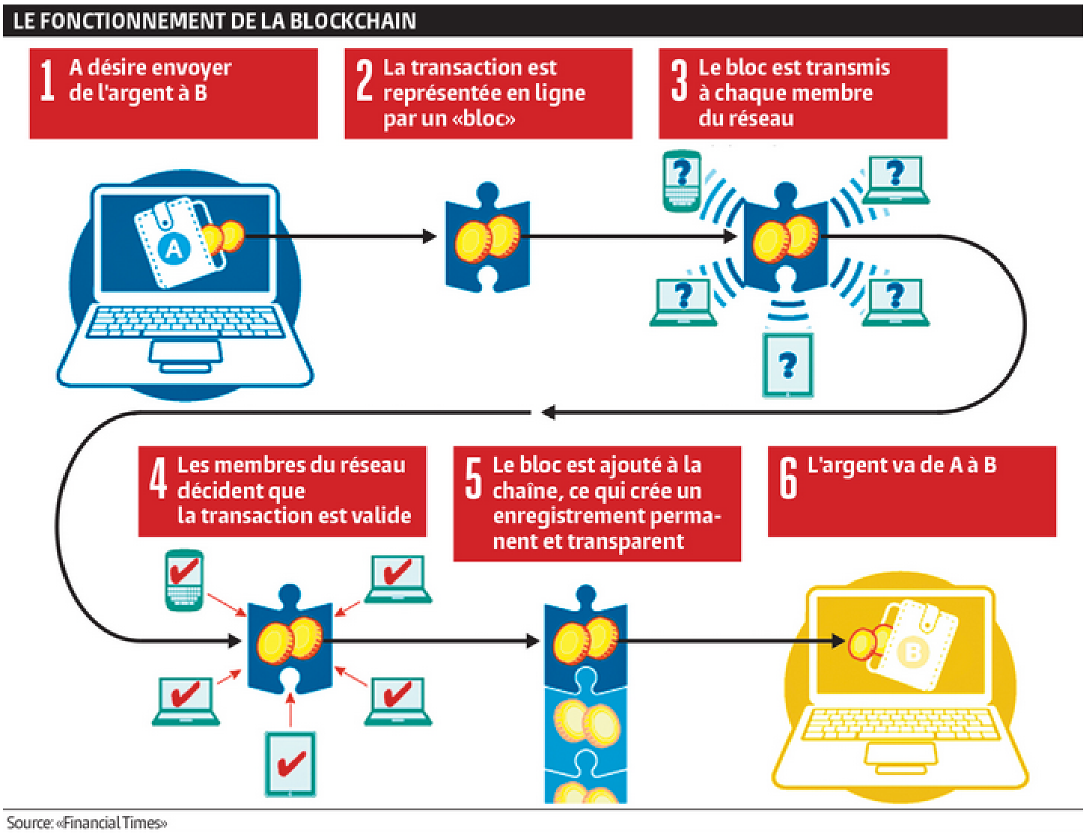
Il n'y a pas de points à chaque abscisses ! Il faudra donc se placer un peu à coté ! ! ! Le décalage maximal est faible mais il faudrait l'évaluer. Pour faire simple nous allons coder sur 2 octets chaque octet de monBin. On utilisera les méthodes :

```
M=ElemtE07.elemtE07APartirDeX(ElmtZnZ(m,self.p))

monBinC.ajouteMot40b(MP.__hash__())

h,pos=monBinC.lisMot(5,pos)

MP=ElemtE07.ElemtE07DepuisHash(h,self.p)
```

```
class CodeurE0765537(object):
"""Codeur à partir de la courbe elliptique sur F65537"""
def __init__(self,a,B,G=ElemtE07(47106,21934,65537),p =65537):
"""Dans les valeurs par défaut B=54321*G

def __str__(self):
return f"CodeurE0765537 avec la clé privé {self.a=} et sa clé publique {self.A=}, la clé publique d'un tie
def __repr__(self):

def binCode(self,monBinD:Binaire603)->Binaire603:

def binDecode(self,monBinC:Binaire603)->Binaire603:

#Exemple d'utilisation ;
mes=Texte603("Bonjour les amis !")
print(f"Message à coder :{mes}")
binc=ca.binCode(mes.toBinaire603())
print(f"Message codé avec la clé secrète de {a=} et la clé publique {B=} :{Texte603(binc)}")
bind=cb.binDecode(binc)
print(f"Message décodé avec la clé secrète de {b=} et la clé publique {A=} :{Texte603(bind)}")
```

Exercice 5 (Une vue plus réaliste de la BlockChain)
Directement sur ce graphique (https ://www.bitpanda.com/academy/fr/lecons/comment-fonctionne-une-blockchain/) corriger/compléter la description de la BlockChain.

Attention : **ce schéma ne vaut pas cours**.

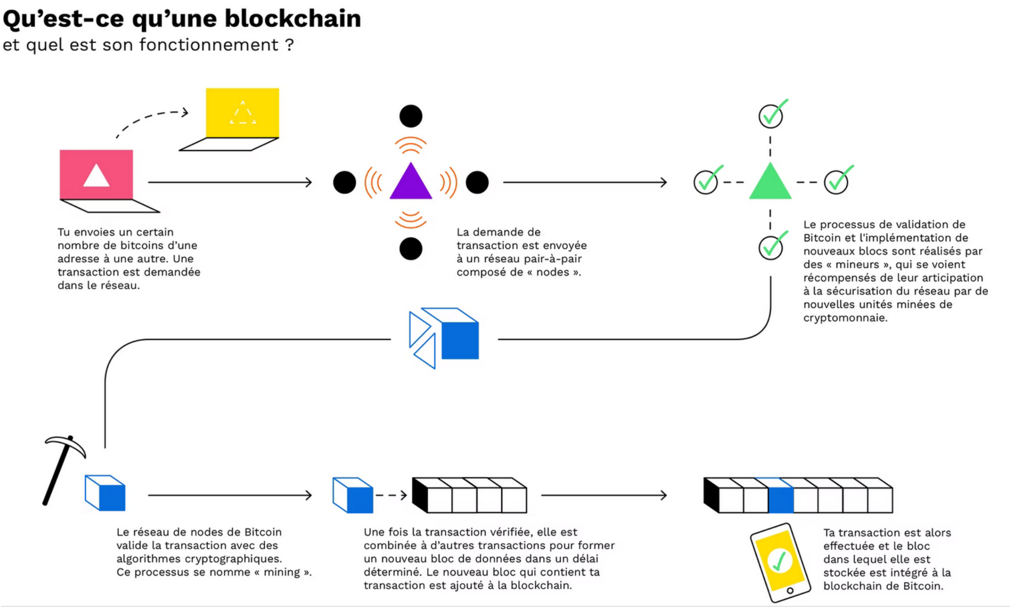

LE FONCTIONNEMENT DE LA BLOCKCHAIN
1 A désire envoyer de l'argent à B
2 La transaction est représentée en ligne par un «bloc»
3 Le bloc est transmis à chaque membre du réseau
4 Les membres du réseau décident que la transaction est valide
5 Le bloc est ajouté à la chaîne, ce qui crée un enregistrement perma-nent et transparent
6 L'argent va de A à B
A
B
Source: «Financial Times»

Exercice 6 (Programmation d'une BlockChain)
Construire les principales méthodes d'une blockChain en pseudoCode Python.

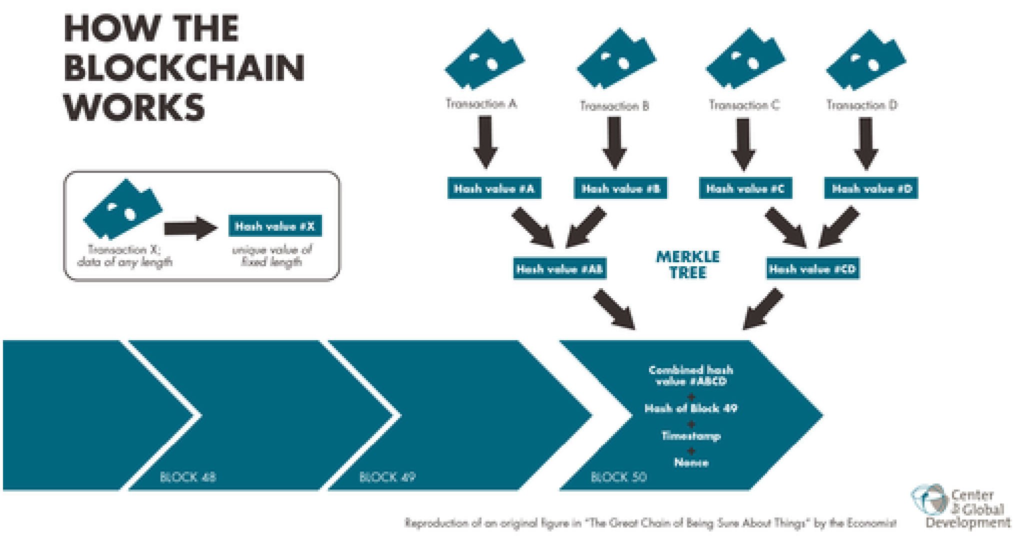
On utilisera des méthodes comme

- Broadcast pour envoyer des infos,
- for s in serveursVoisons : pour interroger les serveurs voisins
- La class BlockChain avec ces méthodes pour récupérer sa longueur ou ses derniers éléments
- Hasch256 et BinCodeE07

Attention : **ce schéma ne vaut pas cours.**



(https ://www.letemps.ch/economie/blockchain-promet-une-nouvelle-revolution)



(https ://www.clubic.com/antivirus-securite-informatique/cryptage-cryptographie/crypto-monnaie/article-842677-1-comment-fonctionnent-bitcoin-blockchain.html)

On écrira les méthodes comme :

- Ajoute réserve
- Construit BlockSuivant
- VerifieBlock (Appele à la réception d'un événement nouveau block trouvé)
- AjouteBlock