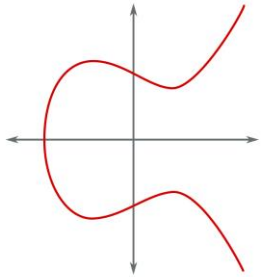
Partie « cryptographie sur courbes elliptiques » (2h)

Alexandre Guitton

Introduction brève

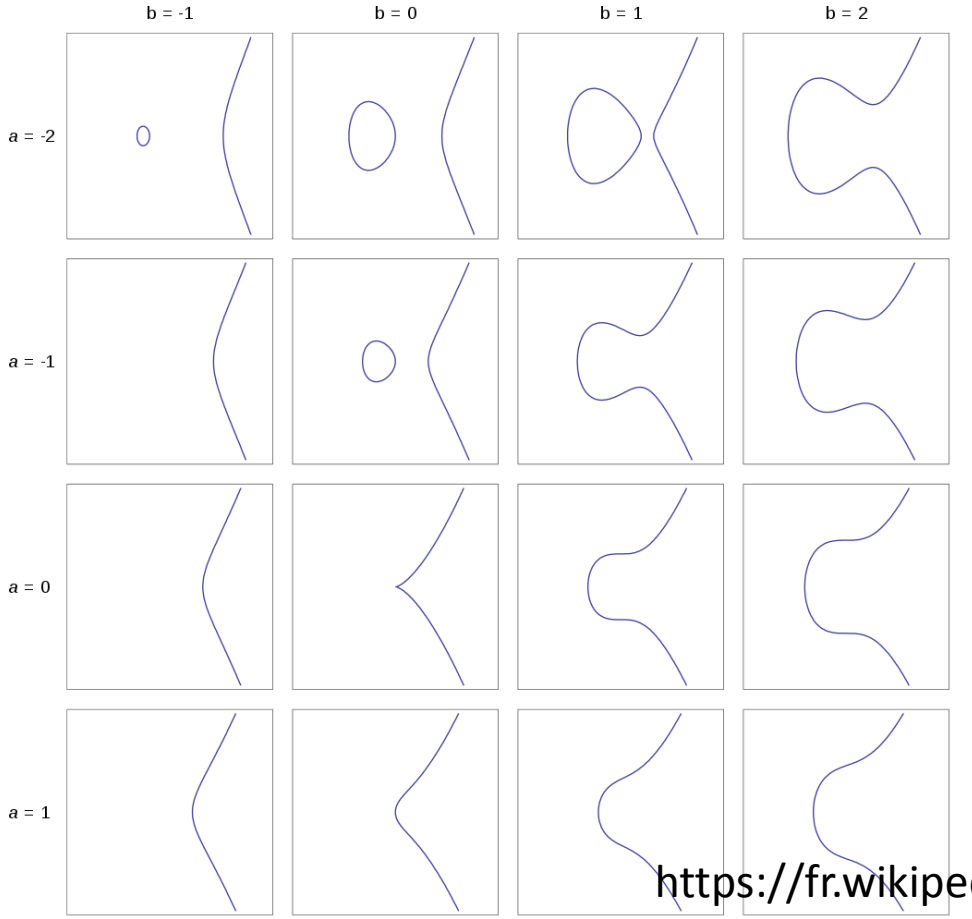
- Les courbes elliptiques sont des courbes ayant des propriétés mathématiques intéressantes



$$y^2 = x^3 + ax + b$$

- Les courbes elliptiques permettent de concevoir des protocoles cryptographiques asymétriques efficaces
  - Avec des clés plus petites que les protocoles cryptographiques asymétriques qui se basent sur l'arithmétique modulaire (notamment : RSA, Diffie-Hellman)

Plusieurs exemples de courbes



https://fr.wikipedia.org/wiki/Courbe_elliptique

Rappel sur les protocoles asymétriques

- Dans un protocole cryptographique asymétrique, il y a deux clés : une clé publique et une clé privée
- Confidentialité (ex : RSA)
  - La clé publique sert à chiffrer, la clé privée sert à déchiffrer
  - RSA étant lent, il sert davantage pour la signature (intégrité) que pour le chiffrement
- Intégrité (ex : RSA et fonction de hachage)
  - La clé privée sert à signer, la clé publique sert à vérifier la signature
- Echange de secret (ex : Diffie-Hellman)
  - Les deux partenaires échangent des informations chiffrées avec la clé publique, et calculent un secret partagé en utilisant leur clé privée

Efficacité de RSA

- En 2021, la taille des clés RSA recommandée est d'environ 2048 bits
  - Pour information, le record de factorisation (au 28/02/2020) était de 829 bits, soit un nombre à factoriser de 250 chiffres décimaux

    https://en.wikipedia.org/wiki/RSA_Factoring_Challenge
- Cela signifie que toutes les opérations qui utilisent des clés RSA (ex : chiffrement, déchiffrement, signature, etc.) se font sur 2048 bits
  - C'est long… surtout pour les ordinateurs ayant peu de ressources (ex : téléphones portables, montres connectées, objets de l'IoT, etc.)
- Pourquoi faut-il des clés aussi longues ?
- Comment concevoir un protocole plus efficace que RSA ?

Pourquoi faut-il des clés aussi longues dans RSA ? (1/2)

- Une clé RSA valide est un produit de deux nombres premiers => certains nombres ne sont pas des clés valides
  - Par exemple, 2021 est une clé RSA valide (2021=43*47)
  - Mais, 2022 n'est pas une clé RSA valide (2022=2*3*337)
- Donc, le nombre de clés RSA valides est inférieur à 2^n, avec n la taille de la clé RSA
  - Par exemple, si la clé RSA faisait 8 bits, il n'y aurait que 158 clés valides sur les 2^8=256 entiers. Un algorithme de cryptanalyse n'aurait qu'à tester ces 158 clés pour trouver la bonne.
  - Si la clé RSA faisait 16 bits, il n'y aurait qu'à tester 31262 clés (sur les 65536 nombres)

Pourquoi faut-il des clés aussi longues dans RSA ? (2/2)

- De plus, l'algorithme actuel le plus rapide pour factoriser un nombre n est assez efficace
  - Il s'appelle le crible généralisé du corps des nombres
  - Sa complexité est $\exp\left(\left(\sqrt[3]{\frac{64}{9}}+o(1)\right)(\ln n)^{\frac{1}{3}}(\ln\ln n)^{\frac{2}{3}}\right) = L_n\left[\frac{1}{3}, \sqrt[3]{\frac{64}{9}}\right]$

    https://en.wikipedia.org/wiki/General_number_field_sieve

  - Pour une clé de 1024 bits, il faut 2^{86} opérations, donc une sécurité de 86 bits
  - Pour une clé de 2048 bits, il faut 2^{116} opérations, donc une sécurité de 116 bits
- (Remarque : dans les algorithmes de chiffrement symétriques comme DES ou AES, une clé de 128 bits fournit 128 bits de sécurité)

Vers un protocole plus efficace que RSA

- L’attaque principale de RSA est liée à la factorisation des entiers => il faut trouver un nouveau protocole qui ne se base pas sur la factorisation des nombres entiers
- Et on doit trouver une fonction qui est :
  - Facile à calculer dans un sens (c’était l’exponentiation modulaire pour RSA)
  - Difficile à calculer dans l’autre sens (c’était le logarithme discret pour RSA)
  - Qui possède une faille (c’était la connaissance de la clé privée pour RSA, qui permet une énorme simplification du calcul du logarithme discret)

Les courbes elliptiques

- Les courbes elliptiques sont des fonctions mathématiques particulières
  - Elles correspondent à des équations de la forme : $y^2 = x^3 + ax + b$
  - … avec quelques contraintes supplémentaires (ex : il faut que $4a^3+27b^2 <> 0$)
- La cryptographie sur courbes elliptiques se base sur le fait :
  - Que l'opération de multiplication (voir plus loin) est facile à calculer
  - Que l'opération inverse (appelée « logarithme discrets sur les courbes elliptiques ») est très difficile à calculer
  - Que l'opération inverse possède une faille que l'on peut exploiter si l'on possède certaines informations

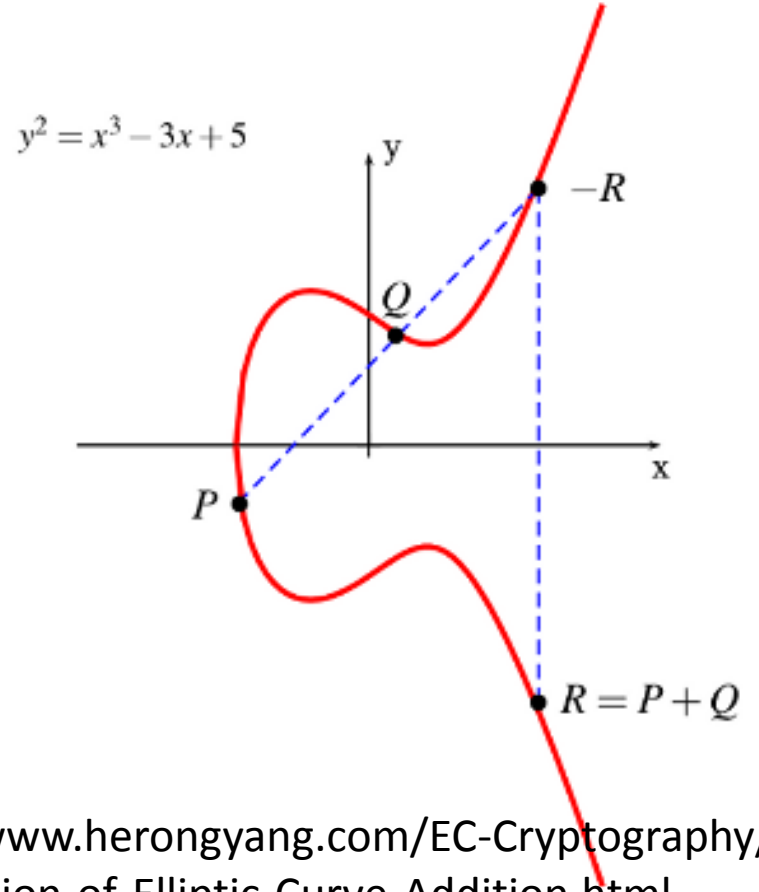
Préalable : les groupes cycliques

- Les courbes elliptiques se basent sur la notion mathématique de groupe cyclique (cette notion est d'ailleurs très présente dans RSA aussi)
- En algèbre, un groupe cyclique est un groupe (G, +) qui est fini et monogène
  - La loi « + » doit être interne (pour tous a et b dans G, a+b est dans G)
  - La loi « + » est associative (pour tous a, b et c, (a+b)+c = a+(b+c))
  - La loi « + » possède un élément neutre noté e (pour tout a, a+e = a) ; on le notera O
  - La loi « + » possède un symétrique (pour tout a, il existe b tel que a+b=e) ; on le notera –a
  - Le nombre d'éléments de G est fini
  - G possède un élément g tel que tout élément du groupe peut s'écrire n.g
- Un (sous-)groupe cyclique G est donc égal à {0.g, 1.g, 2.g, 3.g, 4.g, ...}

Points d'une courbe elliptique

- Une courbe elliptique est (presque) l'ensemble des points satisfaisant l'équation suivante $y^2 = x^3 + ax + b$
  - On ajoute un point (que l'on peut imaginer à l'infini), noté O
- On va ensuite définir deux opérations :
  - L'addition de deux points P et Q, notée « P+Q »
  - La multiplication d'un point P par un entier k, notée « k.P »

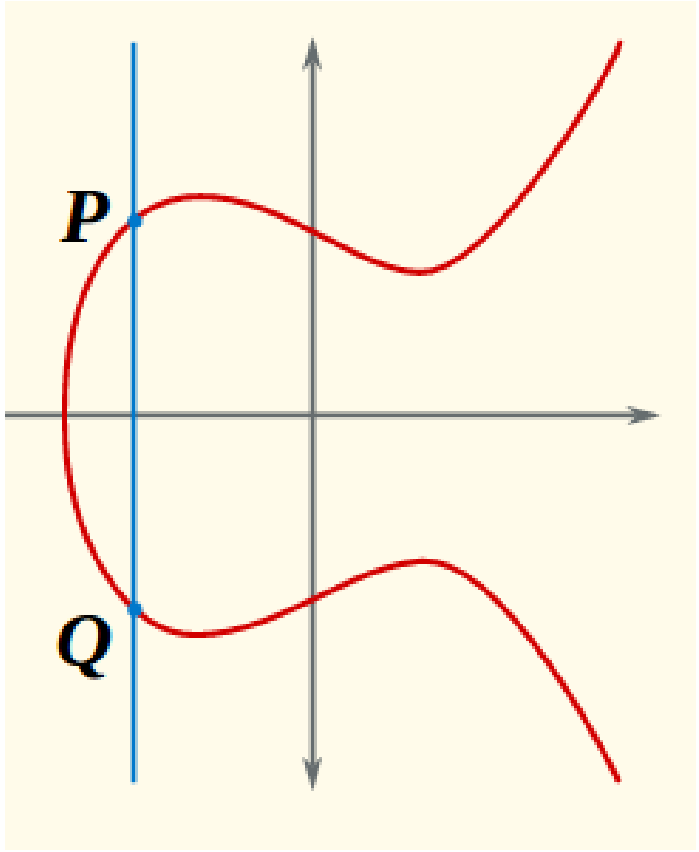
Addition de deux points P et Q (1/3)



http://www.herongyang.com/EC-Cryptography/Algebraic-Description-of-Elliptic-Curve-Addition.html

- Approche géométrique
  - On trace la droite (PQ)
  - On cherche la troisième intersection avec la courbe
  - On calcule le symétrique par rapport à l'axe des abscisses
  - Ce point R est égal à P+Q

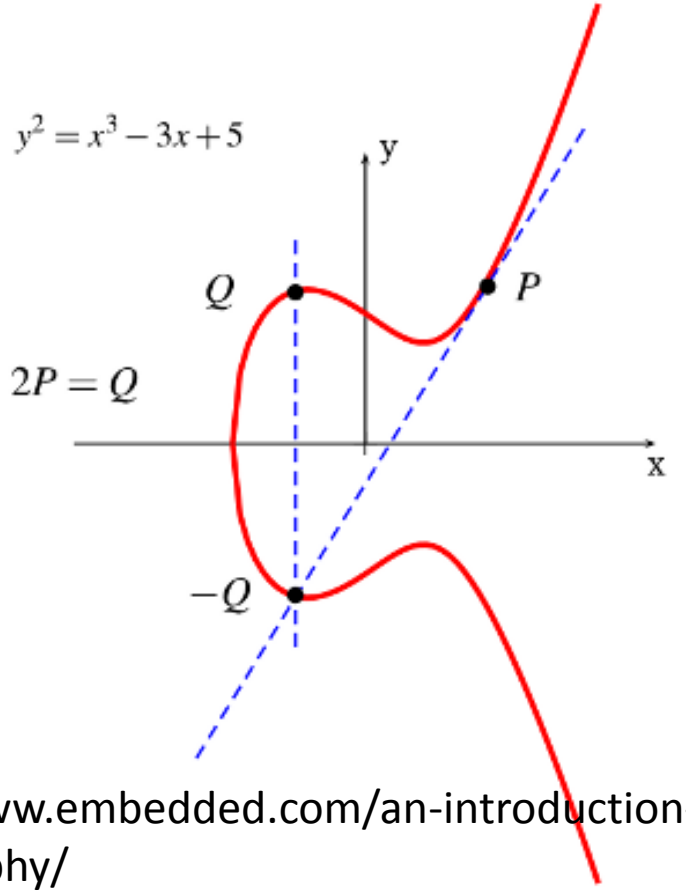
Addition de deux points P et Q (2/3)



https://electriccoin.co/blog/snark-explain7/

- Approche géométrique
  - Si la droite (PQ) est parallèle à l'axe des ordonnées, on considère que la troisième intersection est le point à l'infini O
  - Le symétrique de O est O
  - Donc R=O
  - Remarque : dans ce cas, Q est le symétrique de P par rapport à l'axe des abscisses, donc on note Q=-P. On a ainsi P+(-P)=O, ce qui est cohérent avec notre interprétation du O comme élément neutre

Addition de deux points P et Q (3/3)



https://www.embedded.com/an-introduction-to-elliptic-curve-cryptography/

- Approche géométrique
  - Pour ajouter P à P, on trace la tangente à la courbe en P, et on considère que cette tangente touche 2 fois la courbe
  - On cherche la troisième intersection, et on calcule le symétrique
  - On a donc calculé P+P
  - On note P+P=2.P

Multiplication d’un point P par un entier k

- On cherche à calculer Q=k.P
  - On développe : Q=P+P+P+…+P (k fois)
  - On calcule P+P=2.P, puis (2.P)+P=3.P, puis (3.P)+P.=4.P, etc., jusqu’à obtenir k.P
- Remarque :
  - On peut aller plus rapidement en utilisant la multiplication rapide
  - Exemple : pour calculer 9.P, on peut calculer :
    - 2.P => coûte une addition
    - Puis 4.P=2.P+2.P => coûte une addition
    - Puis 8.P=4.P+4.P => coûte une addition
    - Puis 9.P=8.P+P => coûte une addition

Signatures

- Elliptic Curve Digital Signature Algorithm (ECDSA)
- Génération des clés :
  - Alice choisit une courbe elliptique (a, b, p, P, n)
  - Alice choisit un entier aléatoire s entre 1 et n-1
  - La clé privée est s
  - La clé publique est Q=s.P
- Signature d'un message M par Alice :
  - Alice choisit un nombre aléatoire k entre 1 et n-1, et calcule le point k.P=(i,j)
  - Alice envoie (i [n], k^{-1}(H(M)+s.i [n]))
- Vérification par Bob :
  - Bob calcule le point (H(M).y^{-1} [n]).P+(x.y^{-1} [n]).Q=(i,j), et vérifie que x=i [n]

Echange de clés

- Elliptic Curve Diffie-Hellman (ECDH)
- Alice et Bob s'accordent sur une courbe elliptique (a, b, p, P, n)
- Alice et Bob ont chacun leur clé privée, et la clé publique de l'autre
  - Ces clés peuvent être créées pour l'occasion
- La clé partagée est : $priv_{Alice}.priv_{Bob}.P$
  - Alice la calcule avec $priv_{Alice}.pub_{Bob}$
  - Bob la calcule avec $priv_{Bob}.pub_{Alice}$

Complication…

- Malheureusement, ce n'est pas du tout pratique d'utiliser des courbes elliptiques définies sur des réels…
  - Et la méthode géométrique n'est pas très précise…
  - On va donc utiliser des courbes elliptiques définies sur des entiers
  - Et on va aussi se limiter à des entiers codés avec un nombre de bits fixe, donc on va travailler en arithmétique modulaire
  - L'équation devient : $y^2 = x^3 + ax + b \quad [p]$
- On va utiliser une approche analytique (plus précise)

Méthode analytique pour l'addition de deux points (1/3)

- Entrée :
  - Les valeurs de a, b et p dans : $y^2 = x^3 + ax + b \quad [p]$
  - Hypothèse : a et b sont tels que : $4a^3 + 27b^2 \neq 0$
  - Les coordonnées du point générateur P sont connues
- Exemple : p=17, a=2, b=2, P=(5,1), n=19 est le nombre de points du groupe cyclique

Méthode analytique pour l'addition de deux points (2/3)

- Addition de A et B, avec A et B différents de O et A<>B
  - On calcule $s = (y_A - y_B).((x_A - x_B)^{-1})$ [p]
  - On a $x_C = s^2 - (x_A + x_B)$ [p]
  - On a $y_C = s(x_A - x_C) - y_A$ [p]
- Addition de A et B, avec A=B
  - On calcule $s = (3x_A^2 + a).((2y_A)^{-1})$ [p]
  - On a $x_C = s^2 - 2x_A$ [p]
  - On a $y_C = s(x_A - x_C) - y_A$ [p]
- Remarque : si on essaye de calculer 0^{-1} [p], le résultat est le point O

Méthode analytique pour l'addition de deux points (3/3)

- Exemple 1 : Calcul de 2.P, avec p=17, a=2, b=2, P=(5,1)
  - On doit calculer P+P
  - $s_1 = 3x_P^2+a$ [p] = 3.5^2+2 = 77 = 9 [17]
  - $s_2 = (2y_P)^{-1}$ [p] = $(2.1)^{-1} = 2^{-1}$ [17], et 2.9 = 18 = 1 [17] donc $s_2 = 9$
  - $s = s_1.s_2$ [p] = 9.9 = 81 = 13 [17]
  - $x_{2.P} = s^2-2x_P$ [p] = 13^2-2.5 = 169-10 = 159 = 6 [17]
  - $y_{2.P} = s(x_P-x_{2.P})-y_P$ [p] = 13.(5-6)-1 = -14 = 3 [17]
- Exemple 2 : Calcul de 3.P
  - On doit calculer P+2.P
  - $s_1 = y_P-y_{2.P}$ [p] = 1-3 = -2 = 15 [17]
  - $s_2 = (x_P-x_{2.P})^{-1}$ [p] = $(5-6)^{-1} = (-1)^{-1} = 16^{-1}$ [17], et 16.16 = 256 = 1 [17] donc s_2=16
  - $s = s_1.s_2$ [p] = 15.16 = 240 = 2 [17]
  - $x_{3.P} = s^2-(x_P+x_{2.P})$ [p] = 2^2-(5+6) = 4-11 = -7 = 10 [17]
  - $y_{3.P} = s(x_P-x_{3.P})-y_P$ [p] = 2.(5-10)-1 = -11 = 6 [17]

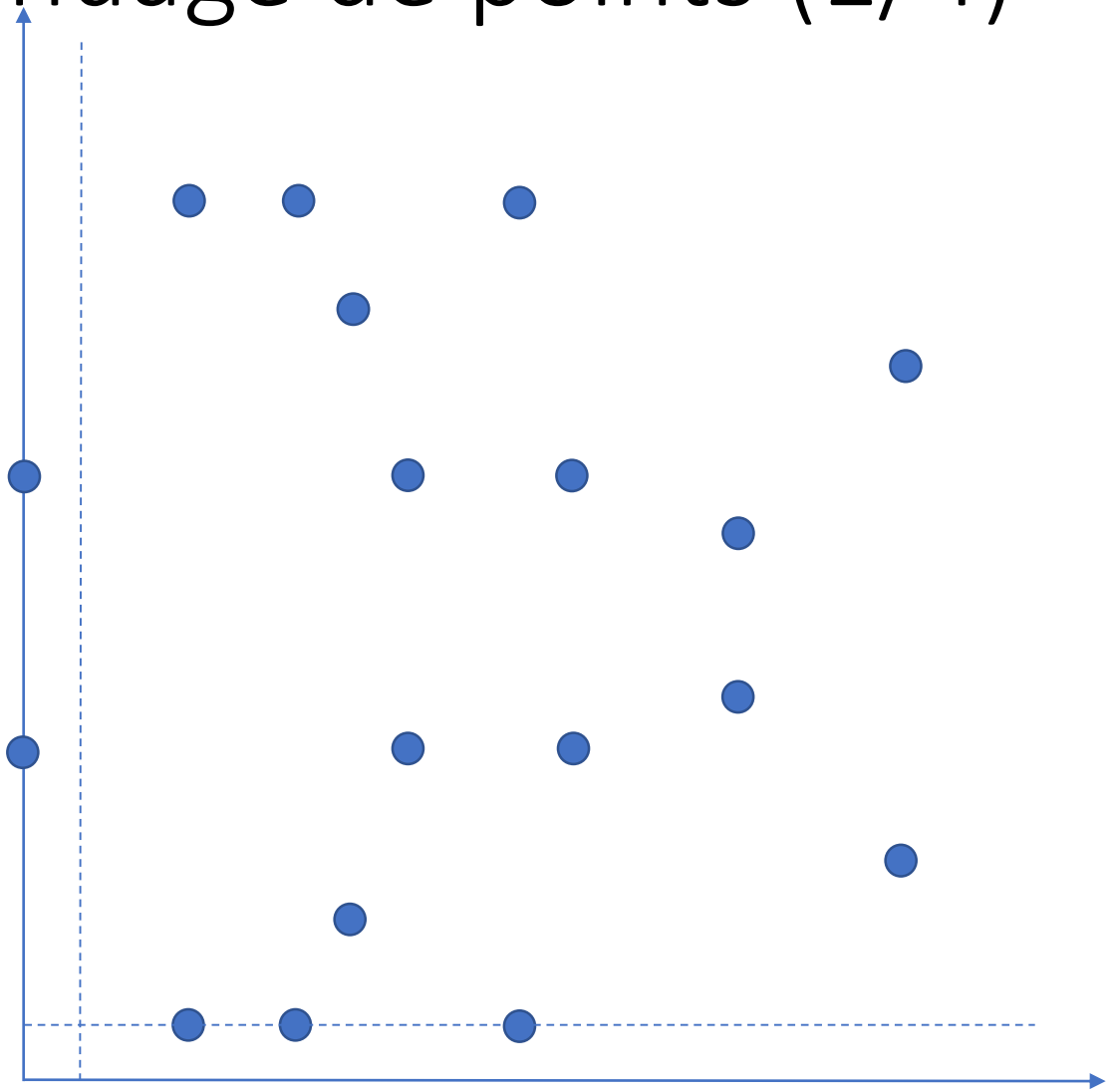
Rappel : calcul de l'inverse en arithmétique modulaire (1/2)

- Algorithme d'Euclide étendu, de complexité O(log(min(a,b)))
  - Entrée : a et b
  - Sortie : (r, u, v) tels que a*u+b*v=r, avec r=PGCD(a,b)
  - (r, u, v, r', u', v') ← (a, 1, 0, b, 0, 1)
  - Tantque r'<>0 faire
    - q ← r/r'
    - (r, u, v, r', u', v') ← (r', u', v', r-q*r', u-q*u', v-q*v')
  - Retourner (r, u, v)

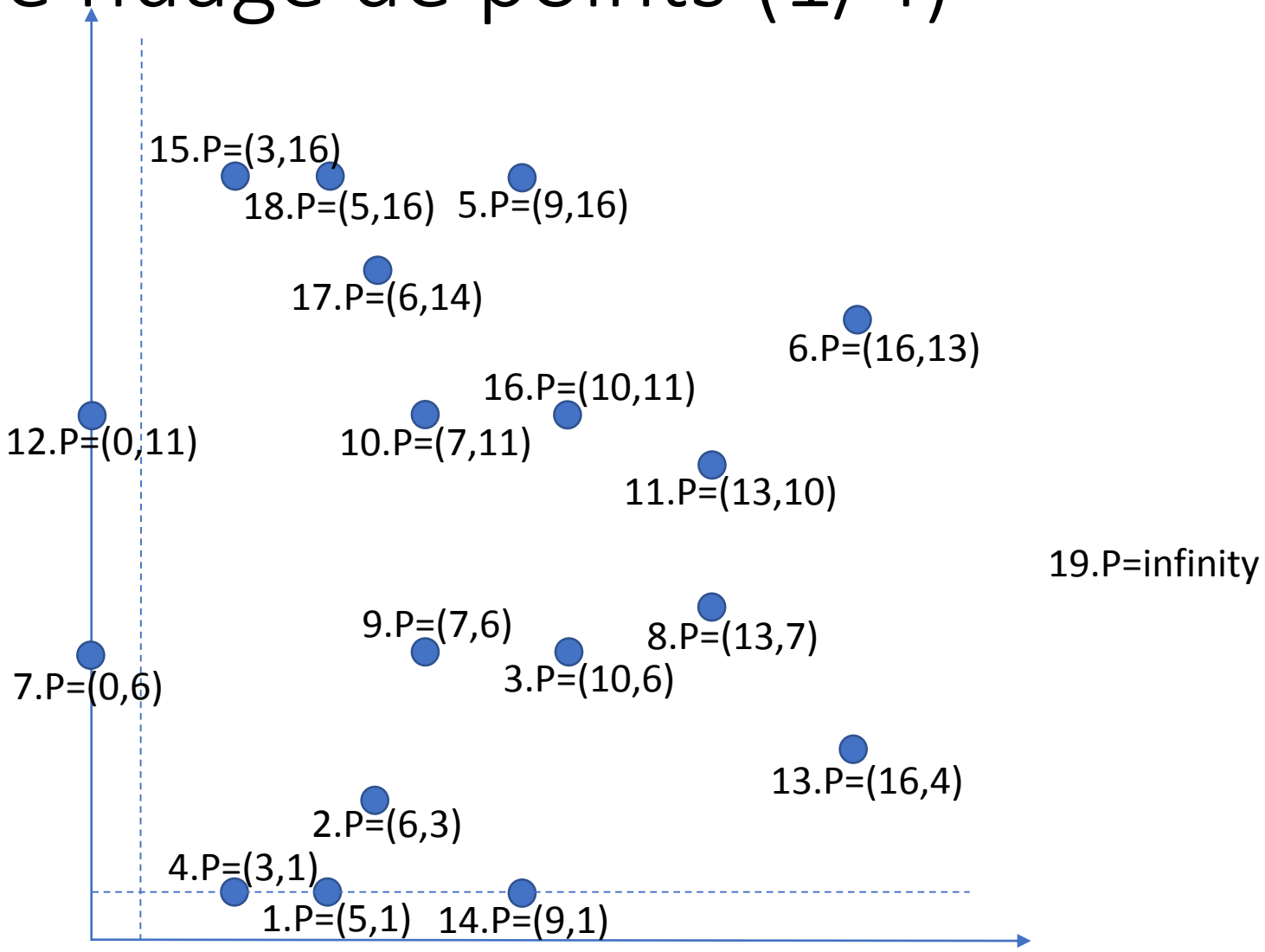
Rappel : calcul de l'inverse en arithmétique modulaire (2/2)

- Application au cas de l'inverse de a modulo p
  - On cherche b tel que a*b+p*k=r, avec r=PGCD(a,p)=1
- Exemple : on cherche l'inverse de a=2 modulo p=17
  - (r, u, v, r', u', v') ← (a, 1, 0, p, 0, 1) = (2, 1, 0, 17, 0, 1)
  - q ← r/r' = 2/17 = 0
  - (r, u, v, r', u', v') ← (17, 0, 1, 2-0*17, 1-0*0, 0-0*1) = (17, 0, 1, 2, 1, 0)
  - q ← r/r' = 17/2=8
  - (r, u, v, r', u', v') ← (2, 0, 1, 17-8*2, 0-8*1, 1-8*0) = (2, 0, 1, 1, -8, 1)
  - q ← r/r' = 2/1 = 2
  - (r, u, v, r', u', v') ← (1, -8, 1, 2-2*1, 0-2*(-8), 1-2*1) = (1, -8, 1, 0, 16, -1)
  - r' vaut 0 donc la valeur recherchée est u = -8 [17] = 9 [17]

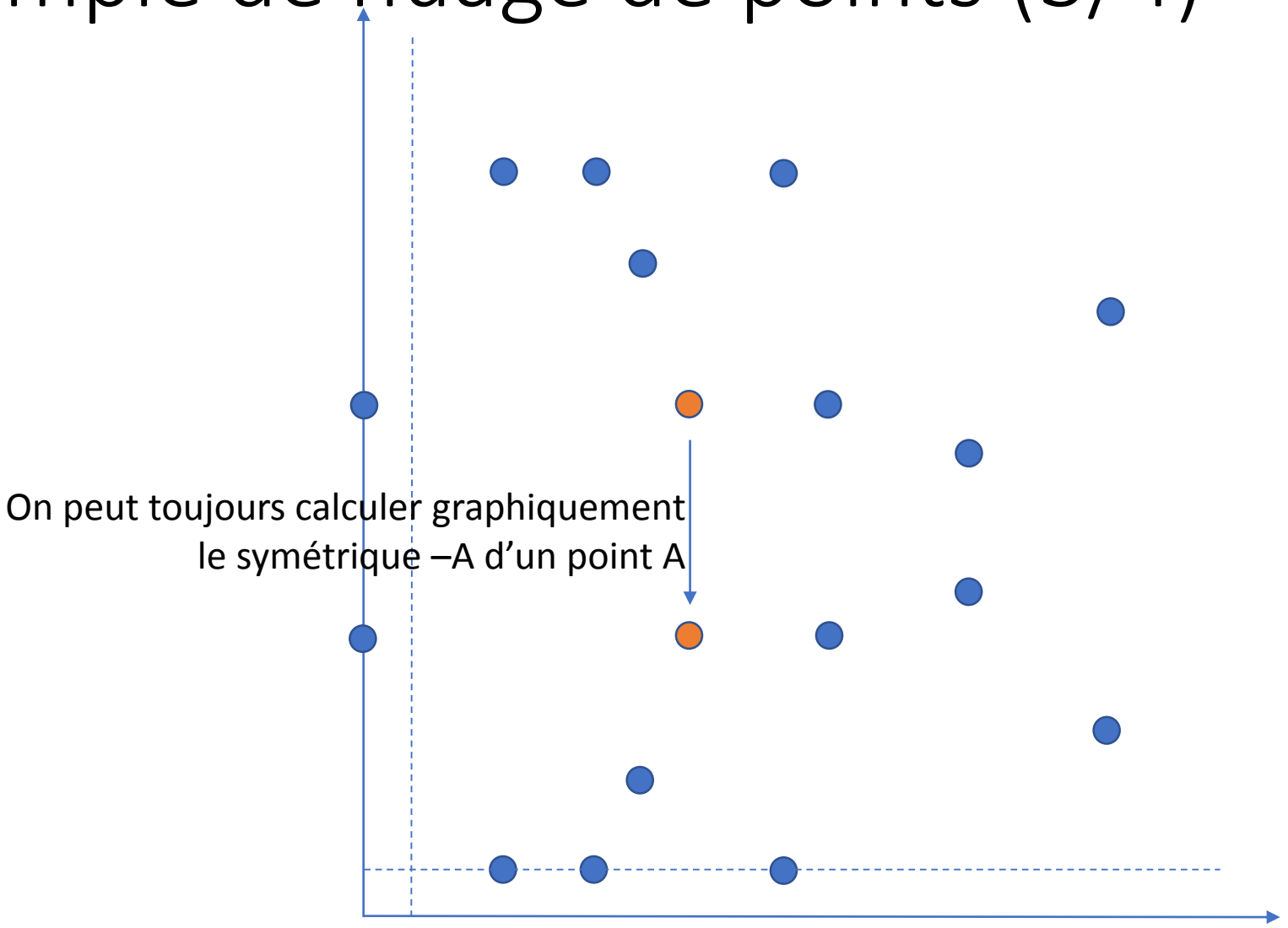
Exemple de nuage de points (1/4)



Exemple de nuage de points (1/4)



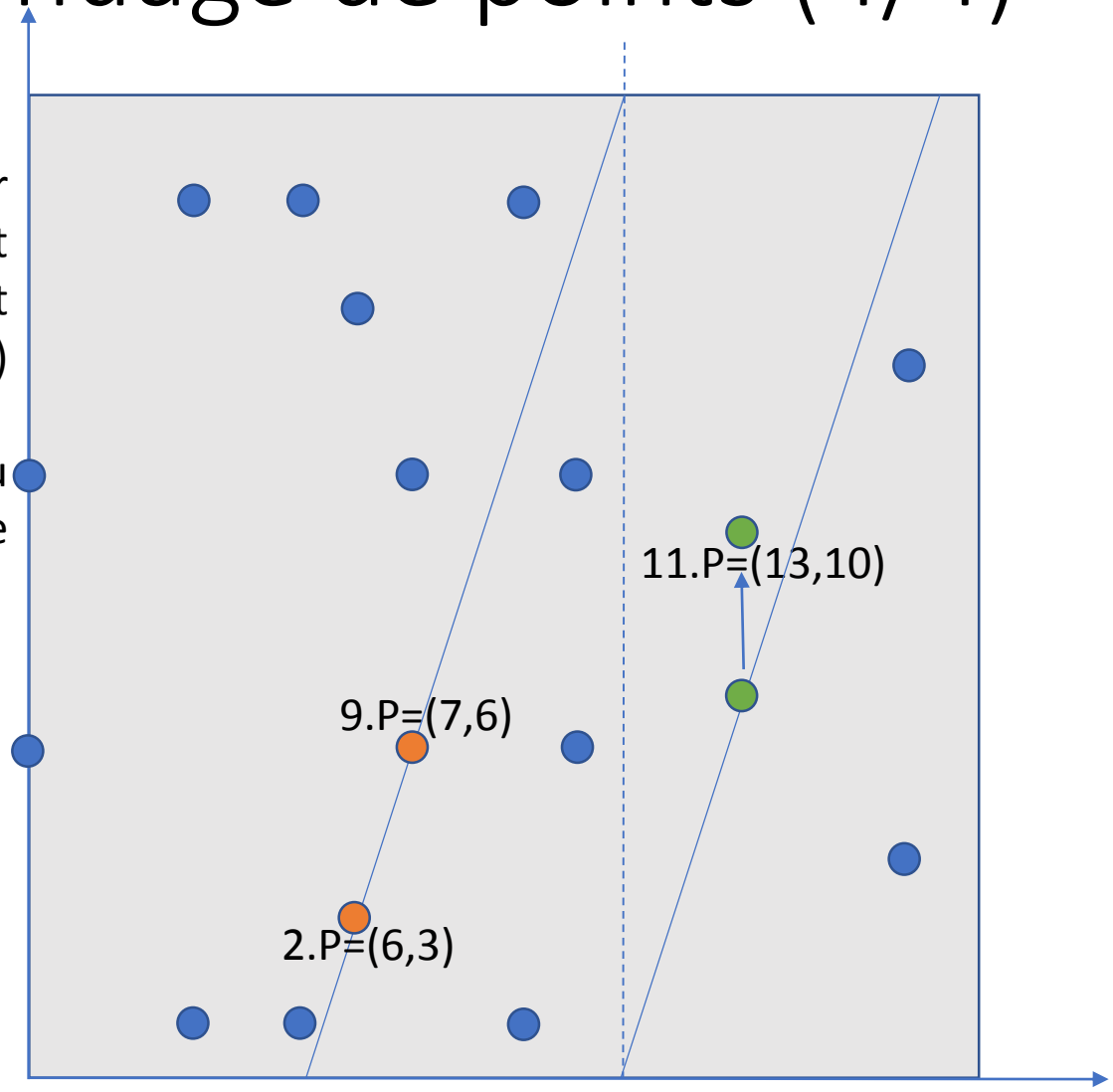
Exemple de nuage de points (3/4)



Exemple de nuage de points (4/4)

On peut toujours calculer graphiquement le point A+B, mais il faut « *wrapper* » la droite (AB)

Cette méthode reste peu précise



Attaques sur les courbes elliptiques

- Les seules attaques connues sur les courbes elliptiques sont les attaques sur les groupes cycliques
  - Attaque *baby step giant step* [Shanks, 1971] : complexité O(sqrt(n)), avec n l'ordre du groupe
  - Attaque rho de Pollard [Pollard, 1978] : complexité O(sqrt(n)), avec n l'ordre du groupe
- Pour atteindre la même sécurité avec une courbe elliptique qu'avec RSA, on peut prendre une taille de clé plus petite
  - Pour atteindre 80 bits de sécurité, il faut un groupe cyclique d'ordre n, avec n sur 160 bits (la complexité pour cryptanalyser étant de $O(\text{sqrt}(2^{160}))=O(2^{80})$, au lieu de 1024 bits dans RSA)

Attaque *baby step giant step* (1/2)

- On connaît P et Q tels que Q=k.P, et on cherche k
- On pose m=ceil(sqrt(n))
- On calcule i.P pour tout 0<=i<m
- On calcule Q-j.m.P pour tout 0<=j<m
- Dès qu'on trouve une correspondance entre les i.P et les Q-j.m.P, on déduit que k=i+j.m
- La complexité est O(sqrt(n)) en nombre d'opérations sur la courbe elliptique (c'est-à-dire, sans la recherche de la correspondance)

Attaque *baby step giant step* (2/2)

- Exemple :
  - On connaît P=(5,1) et Q=(7,6), et on cherche k (qui vaut 9)
  - On pose m=ceil(sqrt(n))=ceil(sqrt(19))=5
  - On a 0.P = infinity, 1.P=(5,1), 2.P=(6,3), 3.P=(10,6), 4.P=(3,1)
  - On a Q-0.m.P=Q=(7,6)
    Q-1.m.P=Q-5.P=Q+14.P=4.P=(3,1)
    Q-2.m.P=Q-10.P=Q+9.P=(5,16)
    Q-3.m.P=Q-15.P=(16,4)
    Q-4.m.P=Q-20.P=Q-P=(13,7)
  - La correspondance est entre i=4 (car 4.P=(3,1)) et j=1 (car Q-1.m.P=(3,1)), on calcule k=4+1*5=9, ce qui est bien la valeur recherchée

Comment choisir les paramètres d'une courbe elliptique ?

- Des courbes pré-calculées existent
  - Leurs paramètres a, b et p sont calculés pour que le groupe cyclique ait un ordre important
  - Ex : Curve25519 : a=486662, b=1, p=2^252+277(…)493, fournissant 128 bits de sécurité
  - Ex : NIST P-256 : a=-3, b=410(…)291, p=2^256-2^224+2^192+2^96-1, fournissant 128 bits de sécurité
  - Pour calculer l'ordre du sous-groupe cyclique généré, on utilise d'abord l'algorithme de Schoof, qui donne l'ordre N de la courbe elliptique. Ensuite, on peut trouver assez facilement des sous-groupes cycliques qui ont pour ordre un diviseur de N, ainsi que le point générateur associé.
  - Les paramètres sont choisis de manière à éviter des attaques connues