**Rapport de projet M1**

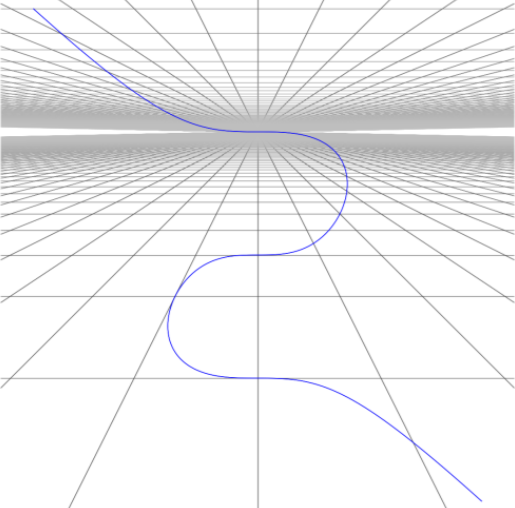
**Projet n°40 : Application pédagogique sur les courbes elliptiques**

Intervenant : Gabriel CHENEVERT.

Etudiants : François BERNARD

Simon CORDIER

Fabien DELATTRE



**Sommaire**

* Présentation de la mission
* Les courbes elliptiques et la cryptographie
* Interface graphique
* Calculs sur les courbes
* Difficultés rencontrées
* Conclusion

**Présentation de la mission**

Nous avions pour objectif de créer une application pédagogique illustrant les calculs sur les courbes elliptiques et leur utilité dans le monde de la cryptographie.

Cette application devait être pédagogique, c’est-à-dire pouvoir permettre d’utiliser facilement les courbes elliptiques et montrer la vitesse d’exécution des différentes opérations que l’on peut faire sur celles-ci.

Ces courbes n’étant pas très intuitives, nous devions produire quelque chose de compréhensible pour des personnes dont ce n’est pas le domaine et faire en sorte qu’elles puissent les utiliser.

**Les courbes elliptiques et la cryptographie**

La cryptographie sur les courbes elliptiques (ECC : Elliptic Curve Cryptography en anglais) est apparue en 1985, proposée par Victor S. Miller et Neal Koblitz en 1987.

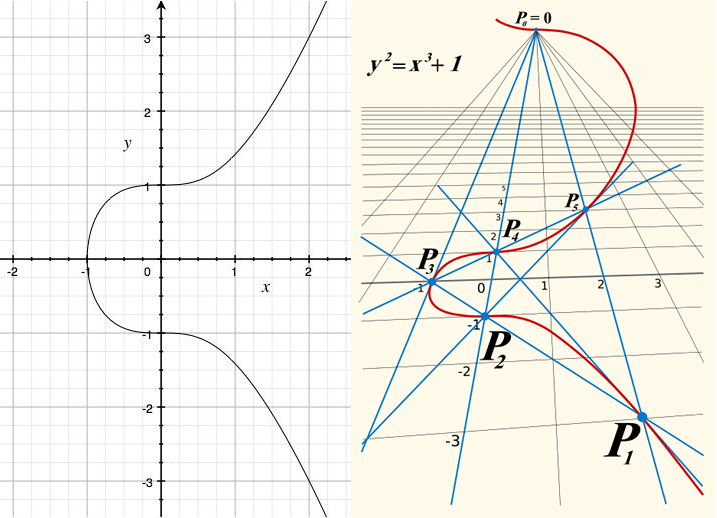
Elle a été proposée comme une amélioration du protocole d’échange de clés de Diffie-Hellmann, apportant une rapidité d’exécution 20% supérieure. Ces échanges sur canaux non sécurisés sont également appelés chiffrement asymétrique.

Pour différencier rapidement la nouvelle méthode, il a été vérifié qu’une clé de 200 bits pour des calculs sur les courbes elliptiques est plus sûre qu’une clé de 1024 bits pour le chiffrement RSA. La rapidité n’est donc pas comparable.

La cryptographie sur ces courbes a déjà été brevetée de nombreuses fois et est aujourd’hui surement déjà plus utilisée que la méthode RSA. Résoudre son problème revient à résoudre le problème du logarithme discret, certes contournable, mais qui demande énormément de temps avec les technologies actuelles.

Une courbe elliptique possède une équation du type . Elle possède une formule d’addition de 2 points appartenant à la courbe, ainsi qu’un point neutre étant le point à l’infini noté O.

Pour deux points P et Q sur la courbe, si l’on trace la droite (PQ), on coupe la courbe en un troisième point. La symétrie de ce troisième point par rapport à l’axe des abscisses est le résultat P+Q. Dans le cas où P et Q ont la même abscisse, alors le troisième point est le point à l’infini (point O).



L’utilisation d’un modulo dans la formule d’une courbe elliptique () rend la résolution du problème bien plus difficile car il s’agit alors de résoudre le logarithme discret.

Certains algorithmes comme les attaques par force brute permette de résoudre ce problème mais lorsque que l’on prend de très grands entiers, la résolution peut alors prendre plusieurs années.

L’utilisation des courbes elliptiques s’effectue comme suit :

Alice et Bob choisissent tous les deux un entier suffisamment grand et qu’ils gardent **secrets**.

Ils se mettent d’accord et peuvent donner publiquement les coefficients a et b ainsi que le module p de la courbe. De même, ils prennent en commun un point de la courbe P, calculent chacun de leur côté le résultat de et et se les échangent.

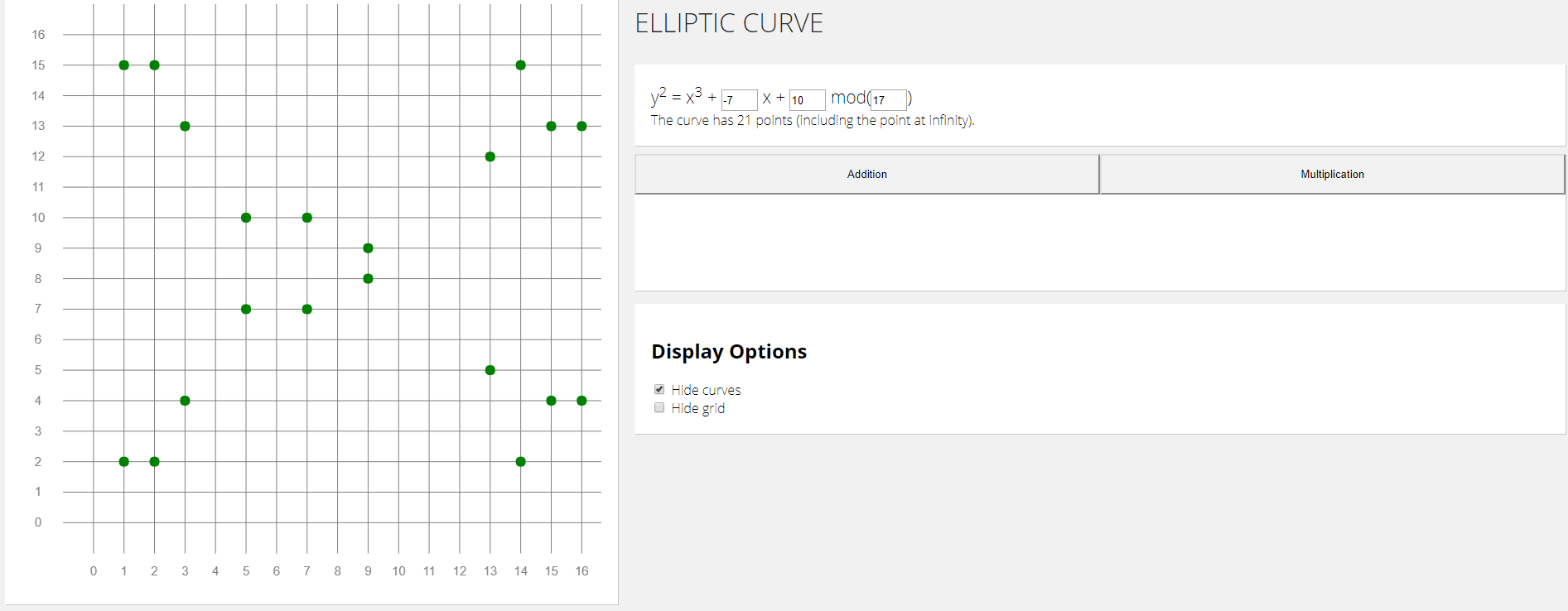
Une fois tout ceci réuni, si le but était un **échange de clés**, ils peuvent tous les deux calculer qui deviendra leur clé secrète commune.

Pour **envoyer un message chiffré**, celui qui veut le transmettre (disons Alice) choisit un entier n qu’il garde secret, puis envoie le résultat de n.P et à l’autre. La personne en face (Bob) peut alors calculer (avec n.P) et le soustraire à pour récupérer le message M caché.

Le seul moyen envisageable de trouver M est de trouver , ce qui demande un temps incalculable.

Si les entiers sont choisis secrètement et rigoureusement, cette méthode de chiffrement devient invulnérable.

**Interface graphique**



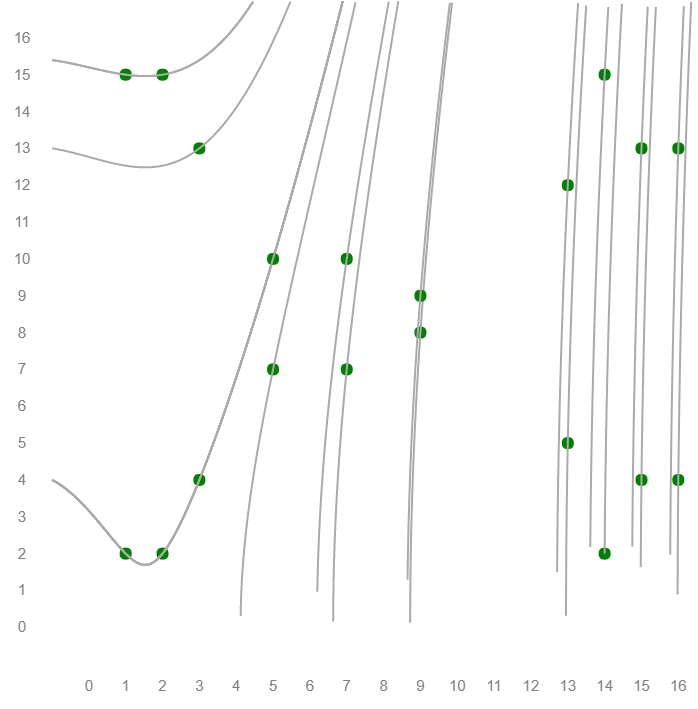
*INTERACTION AVEC LA PAGE HTML*

Entrer la fonction :



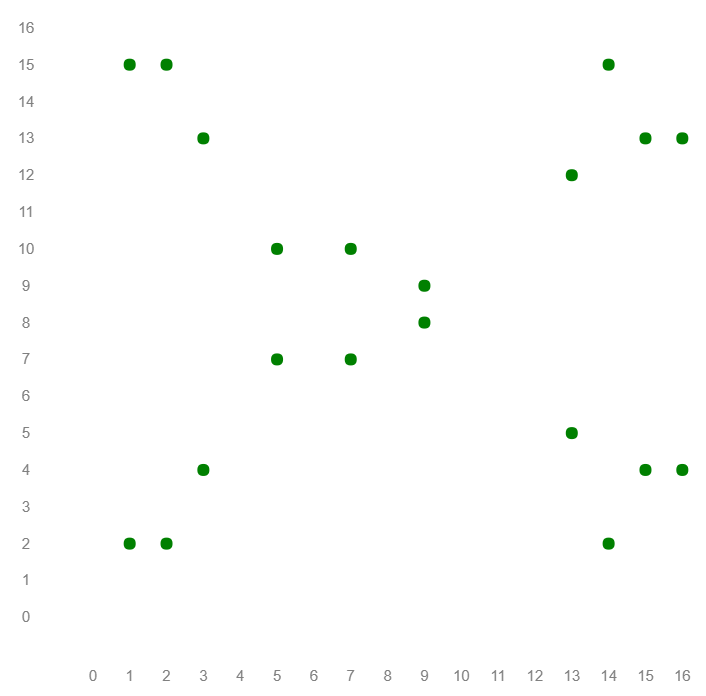
Vous pouvez choisir la valeur du coefficient de X (a), de la constante (b) et le module (m). (Exemple ici : a = -7 ; b=10 ; m=17)

Option d’affichage :



Cela permet :

-De cacher (ou non) la grille.



-De cacher (ou non) la courbe.

*LISTE DES FICHIERS EN LIEN AVEC L’INTERFACE GRAPHIQUE :*

Le fichier HTML : **modulo.html**

Le fichier CSS : **Style.css**

Les fichiers JS :

-**Canvas.js** : Contient la classe Canvas, il permet d’obtenir la grille.

-**Point.js** : Contient la classe Point, il permet de créer des points.

-**Menu.js** : Différencier le menu Addition et Multiplication.

-**Elliptic.js**:

constructor(a,b,m,canvas) : Créer une courbe elliptique ;

drawModulo() : Tracer la courbe elliptique ;

click(point) : Sélectionner un point sur la courbe elliptique  ;

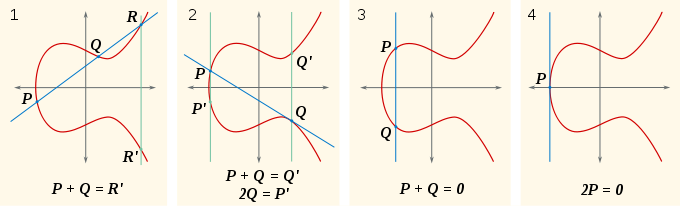
drawLine() : Tracer la droite passant par les deux points sélectionnés ;

animationLine(that, start, end, progress) : complément de la fonction drawLine() , anime le tracé de la courbe.

**Calculs sur les courbes**

Nous allons maintenant voir ce qui se cache derrière cette interface graphique. D’abord, il faut savoir que nous avons choisi d’utiliser les coordonnées homogènes pour nos points, dans le but de pouvoir simplement définir le point à l’infini (point O) ayant pour coordonnées (0, 1, 0).

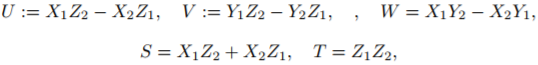
Plusieurs types d’opérations ont été implémentées : l’addition, la multiplication et la division. Plusieurs algorithmes ont été utilisés pour la multiplication et la division, afin de pouvoir comparer leur temps d’exécution. Voyons tout ceci plus en détail.



**Addition**

L’addition est l’opération la plus rapide sur ces courbes. En prenant deux points P et Q vérifiant l’équation de la courbe, on peut obtenir un troisième point de la courbe en traçant la droite (PQ). L’opposé de ce troisième point par rapport à l’axe des abscisses est le résultat de notre addition. De plus, un point P additionné au point à l’infini est égal à ce même point P car O est le neutre de l’addition.

Nous l’avons implémenté comme dit ci-dessus en regardant d’abord si un des deux points est le point O puis à l’aide des formules suivantes :





On obtient ainsi les coordonnées homogènes du résultat.

**Multiplication**

La multiplication sur les courbes elliptiques consiste à multiplier un point de départ P un nombre n de fois, c’est-à-dire l’additionner par lui-même plusieurs fois afin d’obtenir un point final appelé nP. On trouve 3 différentes manières pour effectuer ce calcul :

* Les additions successives

Il s’agit de la méthode la plus lente car nous nous contentons d’appeler la fonction ‘addition’ de deux points plusieurs fois, avec en paramètres P et P. Cette méthode s’exécute en n opérations ce qui n’est pas très optimisé.

* Exponentiation rapide en base 2

Se basant sur le doublement du point, cette technique est largement plus rapide que la précédente car elle a une complexité en log(n). Il s’agit de découper le facteur n en puissance de 2 afin de l’écrire en binaire. On parcourt le nombre binaire et on effectue certaines additions en fonction de la valeur du bit parcouru. A la fin, on obtient les coordonnées du point résultant nP.

* Exponentiation rapide en base 3

Nous n’avons pas implémenté cette méthode car elle ne représentait pas une priorité pour nous et nous avions déjà l’exponentiation rapide en base 2.

**Division**

A l’inverse de la multiplication ou l’on multiplie un point P par un entier n, la division a pour but de retrouver n en lui donnant le point n\*P et le point P. Nous appelons généralement ces points ‘endPoint’ et ‘startPoint’.

Ici, nous trouvons deux différentes méthodes : la méthode naïve (par force brute), et la méthode du Rho de Pollard.

* La méthode naïve

Cet algorithme repose sur le principe de la force brute, c’est-à-dire que l’on essaie pour une valeur de n qui s’incrément de 1 en 1 de faire la multiplication du ‘startPoint’ par n et voir si l’on obtient le ‘endPoint’.

Ceci est vraiment très longue dans le cas ou les nombres entiers sont assez grands mais est très intuitive.

* Le Rho de Pollard

En 1975, John M. Pollard a conçu un algorithme de décomposition en produits de facteurs premiers dans le domaine modulaire, qui a trouvé une variante dans la recherche de collision, ce que nous utilisons pour la division sur les courbes elliptiques.

En partitionnant le domaine de la courbe (le modulo) en 3 parties quasi égales, il est possible d’utiliser une fonction qui nous permettra d’écrire un certain point comme une combinaison linéaire du 'startPoint’ et du ‘endPoint’. Dans le domaine modulaire, les points de la courbes forment un cycle, on va donc avancer à deux vitesses différentes : lorsqu’on regardera le point , le point aura déjà avancé deux fois plus vite. Une fois les deux points égaux sur leurs trois coordonnées, on a trouvé une collision et par un calcul sur leurs coefficients a et b nous pourrons trouver, sauf en cas d’échec, la valeur de n.

La complexité en mémoire de cet algorithme est son gros avantage : O(1) car rien n’est stocké. En vitesse nous sommes aux alentours de O(√n). Cet algorithme est de nos jours le plus rapide que l’on connaisse permettant de résoudre ce problème.

**Difficultés rencontrées**

Au cours de ce projet, quelques difficultés ont été rencontrées. D’abord, le JavaScript était un nouveau langage pour François et moi, nous avons dû nous familiariser avec celui-ci ce qui nous a couté beaucoup de temps. Toutefois, il a de nombreuses ressemblances avec le Java ce qui nous permettait de suivre largement.

Ensuite est venu le problème des ressources sur les courbes elliptiques. Comme ce sujet n’est pas si vieux et pas non plus si facile à utiliser, les archives et exemples sont peu présents et il était difficile de trouver de quoi nous aider à coder certaines fonctions, notamment dans les coordonnées homogènes. Heureusement nous avons reçu beaucoup d’aide à ce niveau de la part de notre professeur que l’on remercie.

Enfin, en guise de recommandation, nous avons été plutôt déçus par l’attribution des projets qui ne correspondent pas forcément à nos attentes et pour certains qui ne correspondent pas au domaine choisi en M1. Peut-être serait-il préférable de fournir un nombre de sujets relatifs au nombre d’étudiants dans la majeure en question ?

**Conclusion**

Hum hum