# BACCALAURÉAT GÉNÉRAL
## SESSION 2016/2017

# Série S
## Épreuve de Spécialité :

## INFORMATIQUE
## ET SCIENCES DU NUMÉRIQUE (ISN)

# DOSSIER-PROJET

Sujet:
**Les mathématiques dans le monde moderne : cryptage des données et apprentissage automatique**

Nom : DESMAREST
Prénom : Mathilde
Date de naissance : 23 Février 2000
Équipe : Hugo BOULIER

Équipe pédagogique :
Mr Chretien (lycée Queneau) – Mr Ramstein (lycée Queneau)

# Tables des matières



Le dossier ci-dessous est disponible sur le site suivant : http://desmarest.mathilde.free.fr/ Créé par nous même en Html, il reprend les thèmes majeurs du projet et permet de mieux visualiser les codes et leurs intérêts.
Les codes situés en annexe sont disponibles sur le Drive suivant :
https://drive.google.com/drive/folders/0BxXhMQ_yRH-JWkw5WlpqREptRW8

# Introduction

- La première et plus longue partie du projet traite de la cryptologie. Aujourd'hui utilisée pour les téléphones mobiles, les cartes bancaires, les passeports biométriques, les cartes de TV à péage ou même dans les clefs de démarrage de véhicules, elle fut longtemps confinée aux cercles militaires et diplomatiques avec une connotation de guerre. En effet, les exemples de textes sur la cryptologie citent majoritairement des batailles, des lieux de combats, des agents en missions, etc.

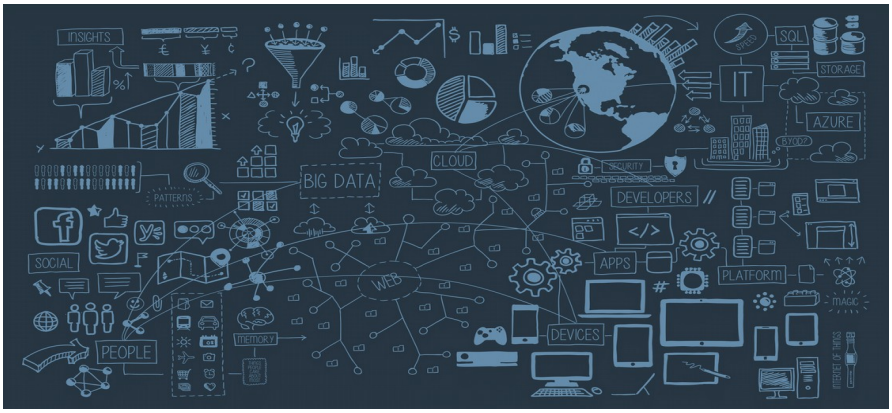
Les activités utilisant la cryptographie ne sont pas neutres, c'est à dire qu'il y aura toujours un impact ; l'industrie militaire avec les missiles nucléaires, le domaine bancaire avec la protection des transactions financières, l'industrie de la carte à puce avec le passeport biométrique, l'identification, le contrôle social.

Dans ce dossier projet, nous avons étudié trois formes de cryptologie ; la *cryptologie traditionnelle* qui ne consiste qu'au traitement du langage et plus particulièrement de l'écriture, la *cryptologie moderne* dont le but est de protéger le système de communications et la *cryptologie contemporaine* avec la révolution des clefs publiques qui permet de chiffrer, complétée par la clef privée qui permet le déchiffrement.

La cryptologie se base sur trois principes fondamentaux qui sont à respecter impérativement. Lorsque vous envoyez un message, vous voulez d'abord vous assurer que votre message ne sera lu que par votre destinataire: on parle donc de **confidentialité** de la cryptologie. Lorsque vous envoyez ce message, vous voulez vous assurer que le destinataire est bien la personne à laquelle vous pensez: on parle donc de **l'authentification** de la cryptologie. Le dernier principe concerne le respect de **l'intégration** du message. Ce principe permet au destinataire d'être certain que le message n'a pas été modifié au cours de l'envoi.

- La deuxième et plus courte partie traite de l'apprentissage automatique. L'apprentissage automatique est utilisé dans l'intelligence artificielle, la neuroscience, les statistiques et même en philosophie et en psychologie. D'après Wikipédia ; "*L'apprentissage automatique est utilisé pour doter des ordinateurs ou des machines de systèmes de : perception de leur environnement : vision, reconnaissance d'objets (visages, schémas, langages naturels, écriture, formes syntaxiques…) ; moteurs de recherche ; aide aux diagnostics, médical notamment, bio-informatique, chémoinformatique ; interfaces cerveau-machine ; détection de fraudes à la carte de crédit, analyse financière, dont analyse du marché boursier ; classification des séquences d'ADN ; jeu ; génie logiciel ; adaptation de sites Web ; locomotion de robots ; analyse prédictive en matière juridique et judiciaire…*".

Plus simplement en reprenant l'exemple d'Hugo : on donne à un ordinateur des photos de sapin et de bouleau en lui disant ce que c'est. Si on lui donne plus tard une photo de sapin, il saura reconnaître que c'est un sapin et non un bouleau !



# Étude du projet

## • Choix du sujet. Pourquoi ce projet ?

J’ai choisi ce projet, car en classe de Seconde, pendant l’année scolaire 2014-2015, j’ai eu la grande chance de pouvoir participer à un stage à l’UFR d’IEEA (Informatique, Électronique, Électrotechnique, Automatique) avec d’autres personnes de différents lycées. Lors de ce stage de 5 jours, j’ai pu découvrir le monde numérique grâce à mon maître de stage ; Eric Wegrzynowski qui est enseignant en informatique à l'Université Lille 1 et intéressé par la cryptologie. Ajoutons à cela que lors des TPE, j’ai étudié le lien entre le son et l’informatique, alors, dans la continuité de ces deux années, j’ai choisi le projet sur la cryptologie et le machine learning.

Le projet me paraissait très intéressant pour étudier les aspects du monde numérique et les méthodes de cryptage. De plus, voulant faire des études dans l’informatique et peut-être m’orienter vers la cybersécurité dans les forces armées, il me semblait inévitable de choisir ce projet par rapport aux autres proposés dans la liste en début d’année. Ce projet me permettait d’avoir une idée bien faite de l’aspect sécuritaire à différents niveaux des différents types de codes ; c’est à dire que certains codes sont faciles à décrypter tandis que d’autres, le sont nettement moins.

J’ai aussi choisi ce projet parce que, même sans s’en rendre compte, la cryptologie nous entoure continuellement. En effet, si l’on détient une carte bancaire, on souhaite être le seul à pouvoir l’utiliser et donc on souhaite protéger nos données bancaires : le cryptage bancaire se base donc sur la cryptologie.

## • Incorporation dans un grand thème

Le projet s'incorpore dans un contexte où l'utilisation des mathématiques dans le monde numérique est omniprésente, on utilise donc dans celui-ci des notions simples et fondamentales des mathématiques pour expliquer le monde numérique. La cryptologie est principalement utilisée pour sécuriser des données numériques, on peut prendre l’exemple des cartes bancaires citées plus haut : ces données numériques sont sécurisées par différents principes de cryptologie. Le machine learning quant à lui se base principalement sur des statistiques avec l'analyse qui peut concerner des graphes, arbres, ou courbes.

## • Cahier des charges

| | Mathilde | Hugo |
|---|---|---|
| Cryptage Affine | X | |
| Cryptage Vigenere | | X |
| Cryptage RSA | X | |
| Cryptage Enigma | | X |
| Partage Secret à 3 | X | |
| Partage Secret à 3 ou plus | | X |
| Cryptage Sac à Dos | X | X |
| Algo K-means 1D | X | |
| Algo K-means 2D | | X |
| Stéganographie | X | |

## • Objectifs à réaliser

Les objectifs à réaliser étaient d'inventer différents codes concrets, représentant les différentes conceptions de cryptage et d'apprentissage automatique des grandes figures du secret numérique, qui sont énoncées dans le cahier des charges, en utilisant différentes notions mathématiques plus ou moins complexes mais indispensables dans l'élaboration des codes. En effet, en passant du cryptage affine au cryptage RSA, en passant par le cryptage par la méthode du sac à dos, le niveau de sécurité en matière de données transmises est totalement différent. On pourra voir qu'en fonction des différents codes, la complexité devient de plus en plus importante. Il va de soi qu'un code de cryptage a toujours avec lui un code de décryptage (de déchiffrage), les objectifs étaient donc doubles : il fallait donc aussi créer le code de déchiffrement.

Les objectifs à réaliser correspondent aussi aux capacités suivantes demandées dans ce projet :

**Premièrement,** il fallait réussir à décrire et expliquer une situation, un système ou un programme c'est à dire de justifier dans une situation donnée, un codage numérique ou l'usage d'un format approprié, qu'un programme réalise l'action attendue. Il fallait aussi détailler le déroulement d'une communication numérique, le rôle des constituants d'un système numérique, le rôle des éléments constitutifs d'une page web, ce qu'effectue tout ou partie d'un programme ou de l'algorithme associé, l'enchaînement des événements qui réalisent la fonction attendue par un programme.

**Deuxièmement,** il fallait concevoir et réaliser une solution informatique en réponse à un problème, c'est à dire analyser un besoin dans un système d'information, le fonctionnement d'un algorithme… Structurer une formule logique, des données, une arborescence, une page web, une approche fonctionnelle en réponse à un besoin… Développer une interface logicielle ou une interface homme-machine, un algorithme, un programme, un document ou fichier numérique…

**Troisièmement**, il fallait réussir à collaborer efficacement au sein d'une équipe dans le cadre d'un projet et agir au sein d'une équipe dans des rôles bien définis, en interaction avec le professeur. Il fallait aussi maîtriser l'utilisation d'outils numériques collaboratifs du type ENT, groupe de travail, forums et rechercher et partager une information, une documentation, une explication.

**Quatrièmement**, il fallait faire un usage responsable des sciences du numérique en ayant conscience des problèmes sociétaux induits c'est à dire d'avoir conscience de l'impact du numérique dans la société notamment de la persistance de l'information numérique, de la non-rivalité des biens immatériels, du caractère supranational des réseaux, de l'importance des licences et du droit.

**Dernièrement**, il fallait savoir communiquer à l'écrit comme à l'oral en documentant un projet numérique pour en permettre la communication en cours de réalisation et à l'achèvement, tout en précisant le déroulement et la finalité du projet.

# Élaboration du projet

## • La répartition du travail ; planning

| Tâches | Nombre de séances (**+ travail à la maison**) | Nom(s) |
|---|---|---|
| Cryptage Affine | 1 séance | Mathilde D. |
| Cryptage Vigenere | 1 séance | Hugo B |
| Cryptage RSA | 3 séances + vacances | Mathilde D. et Hugo B. |
| Cryptage Enigma | 2 séances + vacances | Hugo B. |
| *Photomaton* | *2 séances* | *Mathilde D et Hugo B.* |
| Partage Secret à 3 | 2 séances | Mathilde D. |
| Partage Secret à 3 ou plus | 2 séances | Hugo B. |
| *Cryptage Sac à Dos* | *1 séance* | *Mathilde D. et Hugo B.* |
| Algo K-means 1D | 3 séances + vacances | Mathilde D. et Hugo B. |
| Algo K-means 2D | 2 séances + vacances | Hugo B. |
| Stéganographie | 2 séances | Mathilde D. et Hugo B. |
| *Site internet* | *Vacances* | *Mathilde D.* |

NB : Les lignes marquées en italique sont des tâches « en plus » du projet initial, ces tâches ont été réalisées en parallèle de celui-ci  mais peuvent s'y rattacher.

## • Les différentes phases du projet

La cryptologie est la « science du secret ». Elle regroupe la **cryptographie**, qui permet de **coder les messages**, et la **cryptanalyse**, qui permet de les **décoder**.

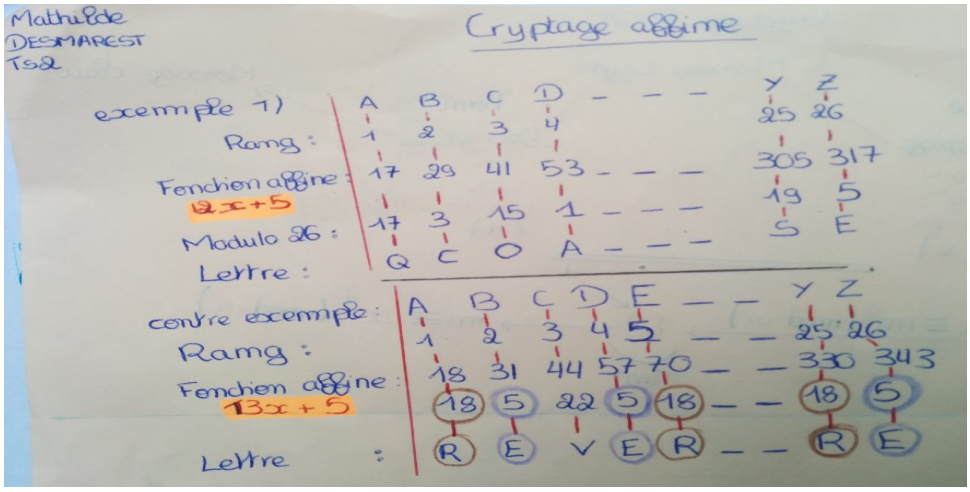
### ➢ Cryptage affine* (* voir annexes)

"*On possède enfin de César des lettres à Cicéron, et sa correspondance avec ses amis sur ses affaires domestiques. Il écrivait, pour les choses tout à fait secrètes, à travers des marques, c'est-à-dire un ordre arrangé de lettres de sorte qu'aucun mot ne pût être reconnu. Si on veut chercher et s'acharner jusqu'au bout, on change la quatrième lettre, c'est-à-dire un D à la place d'un A et pareillement pour toutes les autres.*" Suétone, La vie des douze Césars

Le chiffre affine est  donc une méthode de cryptologie basée sur un chiffrement par substitution mono-alphabétique, c'est-à-dire que la lettre d'origine n'est remplacée que par une unique autre lettre, contrairement au chiffre de Hill.

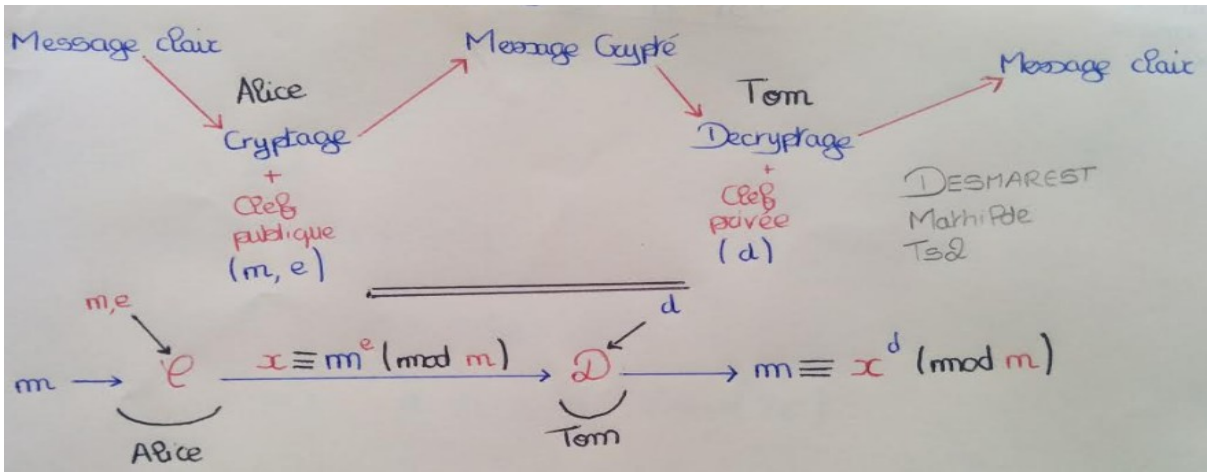
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |

La clef est sous forme **(ax;b)** et s'applique par **ax+b mod[26]**, x étant la lettre de départ. On cherche encore à pouvoir convertir à la fois des majuscules et des minuscules puisque les valeurs ASCII comportent des caractères spéciaux (pas des lettres). Les nombres premiers avec 26 sont interdits. On s'est intéressé aussi au décryptage.




## ➢ Cryptage RSA*

RSA est le sigle provenant des noms de ses inventeurs, qui sont respectivement Ron Rivest, Adi Shamir et Léonard Adleman. D'après le schéma du protocole du code RSA, nous avons différentes étapes à mettre en place pour le bon fonctionnement du principe. Le chiffrement RSA est asymétrique : il utilise une paire de clefs composée d'une **clé publique pour chiffrer** et d'une **clé privée pour déchiffrer** des données confidentielles.



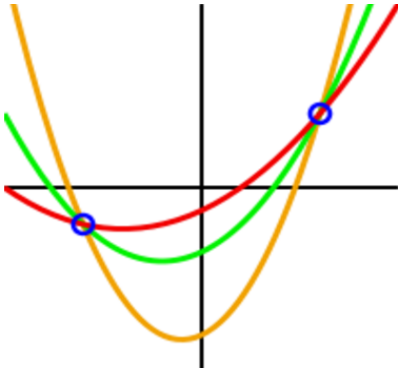

De manière très simpliste, nous avons deux personnages ; Alice et Tom. Alice souhaite transmettre à Tom un message de valeur ***m***, Tom transmet sa clef publique ***n,e*** sans précaution à Alice. Alice se sert donc de la clef publique de Tom pour crypter le message (ou la valeur) qu'elle souhaite lui faire parvenir. Elle lui transmet donc son message ayant reçu le calcul ***m^e (mod n).*** Tom est maintenant le seul à pouvoir décrypter, grâce à sa clef privée **d**, le message par le calcul ***x^d(mod n).***

On a cherché à justifier la force de RSA: quand on choisit deux nombres et que, soit on les multiplie entre eux, soit on les factorise entre eux, plus les nombres sont grands, plus cela devient difficile de les factoriser, ce qui donne sa complexité à RSA.

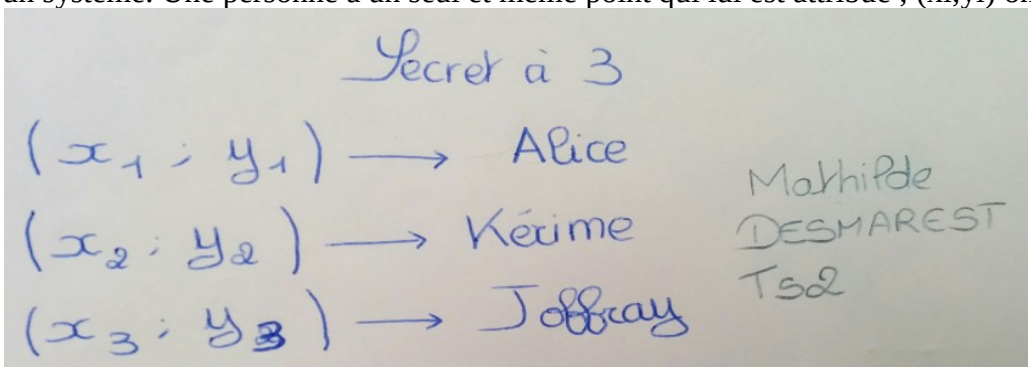
## ➢ Partage Secret à 3*

Le partage de clé secrète de Shamir est un algorithme de cryptographie. C'est une forme de partage de secret, où un secret est divisé en parties, donnant à chaque participant sa propre clé partagée. Certaines des parties, ou l'ensemble d'entre elles, sont nécessaires afin de reconstruire le secret.

Pour le secret à 3, on se base sur des fonctions et coordonnées mathématiques. Comme son nom l'indique, il nous faut donc 3 points donnés à 3 personnes pour retrouver **une fonction polynôme** (de forme $\mathbf{ax^2+bx+c}$): si on ne donne que 2 points on ne peut pas déduire la fonction passant par les points car on aurait une droite, il existerait donc une infinité de paraboles passant par les 2 points.



Il nous faut donc impérativement 3 personnes ce qui correspond donc à 3 points différents pour résoudre un système. Une personne a un seul et même point qui lui est attribué ; (xi;yi) on a donc:




Le secret se situe donc dans les variables, où chaque variable peut correspondre à un message déterminé à l'avance. C'est à dire :

- CHIFFRE → LETTRE // CHIFFRE → MOT // CHIFFRE →PHRASE etc.

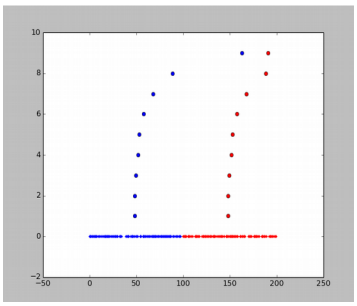
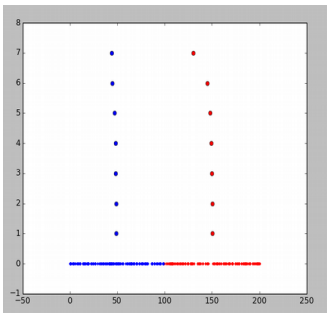
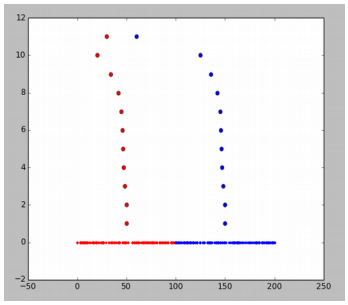
Le code qui permet de retrouver la fonction a été produit par Hugo Boulier, on le commentera en annexe.

## ➢ Algo K-means 1D*

"*L'apprentissage automatique fait référence au développement, à l'analyse et à l'implémentation de méthodes qui permettent à une machine (au sens large) d'évoluer grâce à un processus d'apprentissage, et ainsi de remplir des tâches qu'il est difficile ou impossible de remplir par des moyens algorithmiques plus classiques* ." Wikipédia

- On a plusieurs points générés aléatoirement sur une ligne. - On a deux centres eux aussi définis aléatoirement qui vont être créés sur la ligne, les points les plus proches des centres respectifs vont être regroupés par paquets appelés "cluster". - Pour créer le nouveau centre à

l’étape n+1 on fait la moyenne du placement des points. - Si à l’étape n finale on a la même position des points que l’étape n finale - 1 alors le programme est fini. - On obtient donc deux centres finaux obtenus par optimisation combinatoire.

## ➢ Stéganographie*

La stéganographie est l'art de la dissimulation : son objet est de faire passer inaperçu un message dans un autre message. Elle se distingue de la cryptographie, « art du secret », qui cherche à rendre un message inintelligible à autre que qui-de-droit.

« *Il prit une tablette double, en gratta la cire, puis écrivit sur le bois même les projets de Xerxès ; ensuite il recouvrit de cire son message : ainsi le porteur d'une tablette vierge ne risquait pas d'ennuis.* » Hérodote, Histoires, Livre VII, 239

« *Il fit raser la tête de son esclave le plus fidèle, lui tatoua son message sur le crâne et attendit que les cheveux eussent repoussé ; quand la chevelure fut redevenue normale, il fit partir l'esclave pour Milet.* » Hérodote, Histoires, paragraphe 35 livre V

Le code en annexe n’a pas été crée par les membres du groupe de ce projet, il nous a été fourni par Mr Chrétien. Nous avons donc cherché à comprendre le principe de la stéganographie : image dans une autre image et message dans une image, et aussi à comprendre les codes donnés par le professeur d’ISN.

Cette méthode consiste à modifier le bit de poids faible des pixels codant l'image (dernier chiffre de l’octet). Une image est un tableau constitué d'un ensemble de pixels. Pour chaque pixel, on code la couleur avec trois octets : un pour le rouge, un pour le vert, un pour le bleu.

Un petit tableau pour résumer l'altération du bit de poids faible:

On mais donne 3 pixels 1) 243, 87, 132
2) 255, 255, 255
3) 0, 0, 0

On cherche à cacher le 58 en hexadécimal
↳ soit 0010 0001 en binaire

Le changement de couleurs ne se voit pas.

| | Px1 R | Px2 V | Px3 B | Px1 R | Px2 V | Px3 B | Px1 R | Px2 V | Px3 B |
|---|---|---|---|---|---|---|---|---|---|
| Valeur décimale | 243 | 87 | 132 | 255 | 255 | 255 | 0 | 0 | 0 |
| Valeur binaire | 11110011 | 1010111 | 10000100 | 11111111 | 11111111 | 11111111 | 00000000 | 00000000 | 00000000 |
| Bit à cacher | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | ---------- |
| Couleur finale en binaire | 11110010 | 1010110 | 10000101 | 11111110 | 11111110 | 11111110 | 00000000 | 00000001 | ---------- |
| Couleur finale | 244 | 88 | 133 | 254 | 254 | 254 | 0 | 1 | 0 |

DESMAREST
Mathilde
TS2

## • Le plus du projet

### ➢ Sac à dos*

Le premier cryptosystème à clef publique, qui fut proposé par Ralph Merkle et Martin Hellman en 1978, est basé sur le problème du sac à dos . Il n'est plus utilisé actuellement puisque ce chiffre, ainsi que de nombreuses variantes, a été cassé au début des années 80 par Adi Shamir.

Aussi appelé le Knapsack ou le chiffre de Merkle-Hellman, le problème du sac à dos consiste à empiler des objets dans un sac, de manière à atteindre (si possible) un poids total fixé. Il existe deux types d'empilement : - un **empilement facile** avec une suite de nombres super croissante ou avec l'algorithme glouton, soluble en temps linéaire - un **empilement difficile** soluble en temps exponentiel qui doit tester toutes les solutions mathématiques possibles

Le chiffre de Merkle-Hellman se déroule en trois étapes.

<table>
<tr><td>La première consiste à créer la clef publique à partir de la super super-croissante avec la clef privée (N,M). Prenons la séquence super-croissante: {2,3, 6, 13, 27,52}<br>N=31 et M= 105.</td><td>La deuxième étape consiste à crypter le message binaire: 011000110101101110 (d'après le knapsack, 0 on ne prend pas la valeur, 1 on prend la valeur)</td></tr>
<tr><td>– 2*31 mod 105= 62<br>– 3*31 mod 105= 93<br>– 6*31 mod 105= 81<br>– 13*31 mod 105= 88<br>– 27*31 mod 105= 102<br>– 52*31 mod 105= 37<br>Le "knapsack" serait :{62, 93, 81, 88, 102,37}.</td><td>Message = 011000 110101 101110<br>- 011000 correspond à 93+81= 174<br>- 110101 correspond 62+93+88+37=280<br>-101110 correspond 62+81+88+102=333<br>Le message crypté est : {174,280,333}</td></tr>
<tr><td colspan="2">La dernière étape est le décryptage, le destinataire connaît: la clé privée et les valeurs de N et de M utilisées précédemment. On utilise le calcul inverse de N grâce à l'algorithme d'Euclide étendu</td></tr>
<tr><td colspan="2">Décryptage du message précédent:<br>– 174*61 mod 105=9= 3+6 ; qui correspond à 011000<br>– 280*61 mod 105=70= 2+3+13+52; qui correspond à 110101<br>– 333*61 mod 105=48= 2+6+13+27; qui correspond à 101110.</td></tr>
</table>

### ➢ PhotoMaton* (lien de présentation Prezi disponible sur le site dans la rubrique « autre »)

La transformation du cliché Photomaton est une description d'un type de mélange analogue à un cliché Photomaton qui, à partir d'une image, en fabrique quatre de plus petites dimensions, et ainsi de suite par itération. Cette transformation a été introduite en 1997 par Jean-Paul Delahaye qui est professeur d'informatique à l'université Lille 1 depuis 1988 et chercheur au sein du Laboratoire d'informatique fondamentale de Lille.

Protocole :

n est une puissance de 2 donc pair

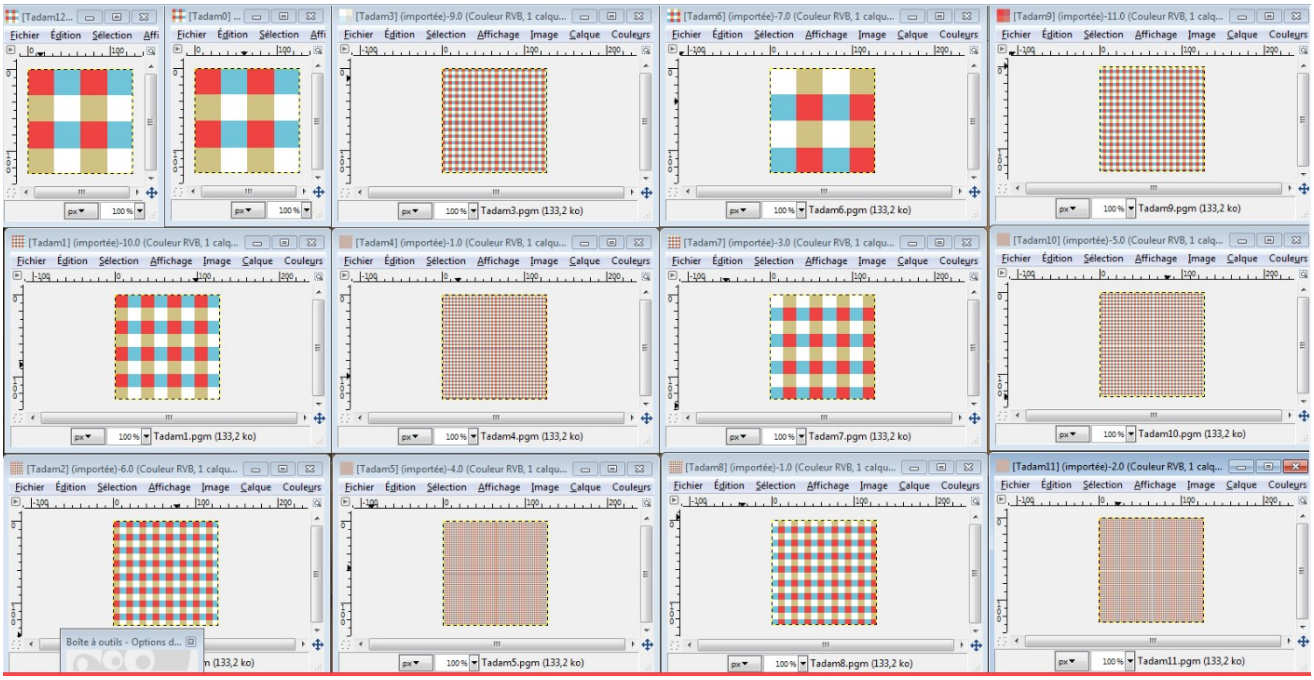
Les lignes impaires de la matrice initiale se retrouvent donc dans la moitié supérieure de la matrice photomaton, les lignes paires dans la moitié inférieure. De même les colonnes impaires se retrouvent dans la partie gauche, et les colonnes paires dans la partie droite.

Ainsi, si on appelle x un élément se situant à la ligne i et à la ligne j de la matrice initiale.

- *Si i est impair, x se retrouvera à la ligne i0 = (i + 1)/2.*
- *Si i est pair, x se retrouvera à la ligne i0 = (i + n)/2.*

On obtient des formules identiques pour exprimer la colonne où se retrouvera l'élément x.

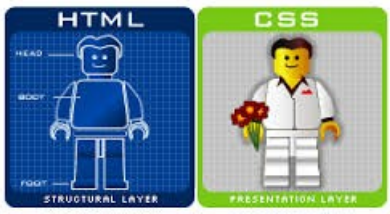
- *Si j est impair, x se retrouvera à la ligne j0 = (j + 1)/2.*
- *Si j est pair, x se retrouvera à la ligne j0 = (j + n)/2.*



## ➢ Site internet en HTML*

On peut considérer notre site internet de présentation comme un petit plus du projet. Ici, j'ai utilisé les connaissances acquises lors de l'apprentissage du HTML, j'ai aussi utilisé un serveur qui permet de nous montrer les codes sources et de les faire fonctionner. On remarque que pour les codes complexes faisant intervenir des variables ou des bibliothèques cela ne fonctionne pas mais ce n'est pas notre faute.

On remarquera aussi que le site http://hugo.boulier.free.fr/index.html, est le même . En effet, il m'a fait confiance et m'a fait parvenir ses codes d'accès à FileZilla (un serveur qui permet de mettre en ligne le site HTML) pour que je puisse déposer le site que j'avais fait et il n'a plus eu qu'à remplir ses parties. La mise en forme de ce site est obtenue grâce à un code CSS obtenu en ligne.



```
10
11 </p>
12
13 <body>
14 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
15 <div style="padding:4px; border:4px solid #e0e0e0;">
16 <div style="padding:3px; background-color:#e0e0e0;">
17 <strong><center>Cryptographie - Machine Learning</center></strong>
18 <strong><center>Hugo & Mathilde </center></strong>
19 </div>
20 </div>
21
22
23 <ul id="nav"><!--
24     --><li><a href="..\index.html">Accueil</a></li><!--
25     --><li><a href="..\classique\cryptageclass.html">Cryptages classiques</a></li><!--
26     --><li><a href="..\moderne\cryptagemod.html">Cryptages modernes</a></li><!--
27     --><li><a href="..\partage\secret3.html">Partage secret à 3 ou plus</a></li><!--
28     --><li><a href="..\algoK\kmeans.html">Algo Kmeans</a></li><!--
29     --><li><a href="..\stega\stega.html">Stéganographie</a></li><!--
30     --><li><a href="..\extensions\encore.html">Autres</a></li>
31 </ul>
32
33
34 <hr />
```

## • Les problèmes et les solutions envisagées

Le projet étant assez complexe dans sa globalité, je ne mentirai pas en disant que j'ai eu des difficultés lors de sa conception. Je reconnais avoir eu la chance de travailler avec un excellent binôme qui tout au long des mois de production était là pour m'aider en cas de besoin sur le codage. J'ai eu aussi des problèmes dans la compréhension des notions mathématiques à utiliser dans les codes, comme par exemple la notion avec **Euclide Étendu** dans le code RSA, qui est l'une des principales fonctions à utiliser. Toutefois, le travail en binôme m'a souvent permis de dépasser ces problèmes.

## Résultats et perspectives

### • État final du projet

A l'heure où ce dossier est rendu, l'état final du projet est plutôt satisfaisant. En effet, chaque tâche du cahier des charges a été traitée comme demandée, les consignes ont été respectées et les temps de production ont été adaptés pour chaque tâche. Des extensions ont mêmes été réalisées, elle peuvent se rattacher au projet d'une certaine façon : **Sac à Dos** autre méthode de cryptage, **PhotoMaton** qui permet d'avoir une image cachée en étape 5, qu'on peut décrypter en étape 12, et le **site HTML** qui est le principal support après le Drive et qui permet d'exposer toutes les recherches faites durant ce projet. Celui-ci sera utilisé pour présenter l'oral du projet le 29 mai 2017.

### • Amélioration et extensions possibles

Pour améliorer mon projet, je pense qu'il faudrait avoir plus de capacités au niveau de la programmation sous Python car plus le code devenait complexe plus les notions de programmations et de mathématiques devenaient indispensables. Si nous avions eu plus de temps, nous aurions pu aussi étudier l'intégrité et le chiffrement holomorphe, mais nous aurions pu aussi chercher à comprendre l'impact de la cryptographie sur la société puisque d'après Wikipédia "les domaines d'utilisations de la cryptographie sont très vastes et vont du domaine militaire, au commercial, en passant par la protection de la vie privée."

Comme extensions possibles, nous aurions pu prendre davantage contact avec Eric Wegrzynowski mais aussi Arnaud Bodin (enseignant en mathématiques à l'université Lille 1 qui fait des vidéos sur YouTube sur la chaîne Exo7). Il est devenu une référence pour les étudiants en licence et classe préparatoires. Cela nous aurait peut-être permis de corriger certaines erreurs peut-être cachées sous les différents codes.

### • Conclusion ; impressions personnelles

Si l'on part du principe que l'option ISN est une option d'initiation à l'informatique, le projet sur la cryptographie et le machine learning m'a semblé assez complexe. Cependant, il fût très intéressant à étudier, mais si les notions en Python avaient été mieux maîtrisées, je pense que la difficulté aurait été atténuée. Toutefois, j'ai beaucoup appris cette année, j'ai découvert d'autres horizons en me confrontant à un nouvel univers : celui de la programmation.

Je conseille aux prochains élèves de terminale en spécialité ISN de choisir ce projet sur la cryptographie et le machine learning car il me paraît être le plus intéressant de par sa complexité mais aussi parce qu' il y a beaucoup de documentations à ce sujet. De nombreux sites et livres traitent ce domaine, on peut donc s'en inspirer (mais pas recopier.). C'est donc avec beaucoup de recherches qu'on a réussi à réaliser ce projet qui m'était totalement inconnu avant le mois de janvier 2017.

J'ai aussi eu la chance d'avoir un bon binôme qui m'a aidée lorsque je rencontrais des difficultés et qui m'a réexpliqué les algorithmes qui m'étaient difficilement compréhensibles. Je remercie donc Hugo Boulier, Mr Chrétien, Mr Ramstein, Mr Bodin et Mr Wegrzynowski pour leur grande aide dans l'élaboration du projet.

Un grand merci surtout pour m'avoir permis de me projeter plus précisément dans mon avenir professionnel. Je sais maintenant que cette voie, même si elle me paraît encore difficile, m'intéresse énormément.

# Bibliographie et Webographie

**Bibliographie :**

- Mathématiques et Informatique : une nouvelle ère numérique, Bibliothèque Tangente, HS n°52, juillet 2014
- Programmation en Python pour les mathématiques, Alexandre Casamayou-Boucan / Pascal Chauvin / Guillaume Connan, édition Dunod, 2012

**Webographie :**

**Cryptographie :**

- Cours d'histoire sur la Cryptologie - Université Paris 8
- Thèse sur la cryptologie, Pierre-Alain Fouque, octobre 2001
- Cryptographie, Christophe Guyeux et Jean-François Couchot, 2013
- Cryptographie - Python
- Entre cryptage classique et cryptage moderne, Mathieu Cunche, 2011
- Différence de protection, François Cayre, 2009

**Cryptages classiques :**

- Méthodes de chiffrement, Andre Lovichi, 2015

**Cryptages modernes :**

- Echanges de clefs, Khan Academy, 2016

**Secret à 3 :**

- Partage Clef de Shamir – Wikipédia

**Algo K-means :**

- http://www.cril.univ-artois.fr/~koriche/Apprentissage2013-Partie1.pdf
- Utilisation de l'apprentissage automatique - Lipn Université Paris 13

**Stéganographie :**

- Stéganographie, Lycée Henri Darras de Liévin
- Stéganographie, Lycée des Flandres
- Art de la dissimulation, M. Lommelé, septembre 2016

# Annexes

- **Cryptage Affine**

CRYPTAGE

```
1
2  texte = input('Entrez votre texte :')   #on rentre le texte voulu
3  a = input('Entrer a:')                  #on insère la première variable de la formule ax+b
4  a = int(a)
5  b = input ('Entrer b:')                 #on insère la deuxième variable de la formule ax+b
6  b = int(b)
7  message = ''
8  nouveau = ''                            # on définit le nouveau message qui prendra une valeur
9  nombre = 0
10 lettre = ''
11
12
13 for i in range(len(texte)):   #Equivalent à la liste len(texte)
14     message = message + texte[i]
15     nombre = ord(message[i])        #on décompose le message lettre par lettre
16     if 97<=nombre<=122:          #Si la valeur de la lettre est compris entre 97 et 122 (ASCII minuscules)
17         nombre = a*nombre + b    #alors la valeur de la lettre recoit a*valeur +b
18         while 97>nombre:         #tant que la valeur est inférieure à 97
19             nombre += 26         #alors nombre = nombre + 26
20         while 122<nombre:        #tant que la valeur est supérieure à 122
21             nombre -= 26         #alors nombre = nombre- 26
22         lettre = chr(nombre)     #d'après le tableau ASCII on associe une lettre à sa valeur: chr()
23         nouveau = nouveau + lettre
24     elif 65<=nombre<=90:         #Sinon la valeur de la lettre est compris entre 65 et 90 (ASCII majuscules)
25         nombre = a * nombre + b #alors la valeur de la lettre reçoit a*valeur+b
26         while 65>nombre:         #tant que le nombre est inférieur à 65
27             nombre += 26         #on lui ajoute 26
28         while 90<nombre:         #tant que le nombre est supérieur à 90
29             nombre -= 26         #on lui retire 26
30         lettre = chr(nombre)       #d'après le tableau ASCII on associe une lettre à sa valeur: chr()
31         nouveau = nouveau + lettre
32     else:
33         nouveau = nouveau + message[i]
34
35 print(nouveau)
36
37 """ Problème si les facteurs choisis valent 13: les nombres ne doivent pas être premiers avec 26"""
```

```python
texte = input('Entrez votre texte :')   #on rentre le texte voulu
a = input('Entrer a:')                  #on insère la première variable de la formule ax+b
a = int(a)
b = input ('Entrer b:')                 #on insère la deuxième variable de la formule ax+b
b = int(b)
message = ''
nouveau = ''                            # on définit le nouveau message qui prendra une valeur
nombre = 0
lettre = ''
vr = 0

alphabet = ['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','R','Q','R','S','T','U','V','W','X','Y','Z','a','b','c','d','e','f','g','h','i','j','k','l
alphabet_code = []
inter = []
nombre_alphabet = 0

for i in range(len(alphabet)):
    nombre_alphabet = ord(alphabet[i])
    if 97<=nombre_alphabet<=122:
        nombre_alphabet = a*nombre_alphabet + b
        while 97>nombre_alphabet:
            nombre_alphabet += 26
        while 122<nombre_alphabet:
            nombre_alphabet -= 26
        lettre = chr(nombre_alphabet)
        inter += [lettre]
    elif 65<=nombre_alphabet<=90:
        nombre_alphabet = a * nombre_alphabet + b
        while 65>nombre_alphabet:
            nombre_alphabet += 26
        while 90<nombre_alphabet:
            nombre_alphabet -= 26
        lettre = chr(nombre_alphabet)
        inter += [lettre]
    alphabet_code += inter
    inter = []

for i in range(len(texte)):
    message = texte[i]
    for i in range(len(alphabet_code)):
        if message == alphabet_code[i]:
            nouveau += alphabet[i]
            vr = 1
    if vr == 0:
        nouveau += message
    vr = 0

print(nouveau)

```

- **Cryptage RSA**

```python

from tkinter import *

def pgcd(a, b) : #trouver le pgcd de deux nombres
   while a % b != 0 :
      a, b = b, a % b
   return b

def euclide_etendu(a, b):
    x = 1 ; xx = 0
    y = 0 ; yy = 1
    while b != 0:
        q = a // b
        a, b = b, a % b
        xx, x = x - q*xx, xx
        yy, y = y - q*yy, yy
    return(a,x,y)


def prepa(): #on donne p et q et ça calcule les coef phi, n, et vérifie si e est valable
    p = int(e1.get())
    q = int(e2.get())
    n = p * q
    phi= (p - 1) * (q - 1)
    e = int(e3.get())
    if pgcd(e,phi) != 1:
        return None
    c, d, dd = euclide_etendu(e,phi)
    d = d % phi
    print("La clef publique contient e =", e, ", et n =", n)
    print("La clef privee est d =", d)
    texte = 'La clef contient e = ' + str(e) + ', n = ' + str(n) + ' (public) et d = ' + str(d) + ' (privé)'
    global PREPA
    PREPA = Label(fenetre, text = texte, font = 'bold')
    PREPA.grid(row=3, column=3)

def codage(): #codage
    e = int(e3.get())
    n = int(e1.get()) * int(e2.get())
    m = int(e4.get())
    x = pow(int(m), e, n)
    texte = 'Votre chiffre codé est ' + str(x)
    global CODAGE
    CODAGE = Label(fenetre, text = texte, font='bold')
    CODAGE.grid(row = 4, column = 3)

def decodage(): #decodage
    p = int(e1.get())
    q = int(e2.get())
    e = int(e3.get())
    phi= (p - 1) * (q - 1)
    while pgcd(e,phi) != 1:
        e = int(input("Cette valeur n'est pas bonne, choisissez un autre e :"))
    x = int(e5.get())
    c, d, dd = euclide_etendu(e,phi)
    d = d % phi
    n = p * q
    m = x ** d % n
    texte = 'Le décodage donne ' + str(m)
    global DECODAGE
    DECODAGE = Label(fenetre, text = texte, font = 'bold')
    DECODAGE.grid(row = 5, column = 3)

"""Partie graphique du code"""
fenetre = Tk()
fenetre.title("Test RSA")
fenetre.geometry("")

Label(fenetre, text = "On va essayer de faire un truc avec RSA et Tkinter").grid(row = 0, columnspan = 2)

bouton=Button(fenetre, text="Fermer", command=fenetre.destroy)
Label(fenetre, text="Entrez p").grid(row=1)
Label(fenetre, text="Entrez q").grid(row=2)
Label(fenetre, text="Entrez e, tel qu'il soit premier avec (p - 1)(q - 1)").grid(row=3)

e1 = Entry(fenetre)
e2 = Entry(fenetre)
e3 = Entry(fenetre)
e4 = Entry(fenetre)
e5 = Entry(fenetre)
e1.grid(row=1, column=1)
e2.grid(row=2, column=1)
e3.grid(row=3, column=1)
e4.grid(row=4, column=1)
e5.grid(row=5, column=1)

Bouton_Prepa = Button(fenetre, text='1ère Étape', command = prepa).grid(row = 3, column = 2)
L4 = Label(fenetre, text = 'Entrez le chiffre à coder').grid(row=4, column = 0)
L5 = Label(fenetre, text = 'Entrez le chiffre à décoder').grid(row=5, column = 0)

Bouton_Codage = Button(fenetre, text='Codage', command = codage).grid(row = 4, column = 2)
Bouton_Decodage = Button(fenetre, text = 'Décodage', command = decodage).grid(row=5, column = 2)

bouton.grid(row = 10, column = 1)
fenetre.mainloop()
```

- **Partage secret à 3**

```python


import numpy as np

import random #on entre les variables a,b,c
a = int(input("Entrer la première variable de la fonction",))
b = int(input("Entrer la deuxième variable de la fonction",))
c = int(input("Entrer la troisième variable de la fonction",))

x1=random.randint(-100, 100) #x reçoit une valeur entre -100 et 100
x2=random.randint(-100, 100)
while x2==x1:    #on attribue à x2 une valeur différente de x1
    x2=random.randint(-100, 100)
x3=random.randint(-100, 100)
while x3==x1 or x3==x2 : #on attribue à x3 une valeur diférente de x2 ET de x1
    x3=random.randint(-100, 100)

def f(x):
    return a*x*x+b*x+c #définition de la fonction

y1=f(x1) #la personne 1 reçoit une valeur par la fonction où x est un chiffre aléatoire
y2=f(x2)
y3=f(x3)

X = np.array([[x1*x1,x1,1],[x2*x2,x2,1],[x3*x3,x3,1]]) #attribution
Y = np.array([y1,y2,y3]) #variable à retrouver
Coeff = np.linalg.solve(X, Y)  #calcul de la matrice

print ("La première personne reçoit le couple","(",x1,";",y1,")",sep='')
print ("La deuxième personne reçoit le couple","(",x2,";",y2,")",sep='')
print ("La troisième personne reçoit le couple","(",x3,";",y3,")",sep='')

```

CRYPTAGE

```python
from sympy import * #Les fonctions symbols pour faire du calcul littéral et simplify pour simplifier (légèrement) le polynôme résultat.
from matplotlib.pyplot import * #Pour tracer une jolie courbe à la fin.
from numpy import *

def Lagrange(Lx, Ly):
    X = symbols('X')
    if  len(Lx) != len(Ly): #Si on ne donne pas le même nombre d'ordonnées et d'abscisses
        print("Les listes font pas la même longueur, c'est génant non ?")
        return 1
    d = 0
    for k in range(len(Lx)):
        t = 1
        for j in range(len(Lx)):
            if j != k:
                t = t * ((X - Lx[j]) / (Lx[k] - Lx[j]))     # Calcul des polynômes de Lagrange
        d += t * Ly[k]          #Ajout des polynômes de Lagrange au polynôme principal
        d = simplify(d)         #Simplification (légère) d'écriture
    return d
Listex = []
Listey = []
x, y = 0, 0
compteur = 1

while x != '': #Demande des abscisses et ordonnées des points
    x = input('''Entrez une abscisse s'il vous plaît''')
    if x != '':
        Listex += [int(x)]
    if x == '':
        break
    y = input('''Entrez une ordonnée s'il vous plaît''')
    if y != '':
        Listey += [int(y)]
print(Lagrange(Listex, Listey))

def f(a):
    return Lagrange(Listex, Listey).evalf(subs={'X':a})

plot(Listex, Listey, 'o',)
axis([min(Listex) - 2, max(Listex) + 2, min(Listey) - 2, max(Listey) + 2])
plot([-100000, 100000], [0, 0], 'black')
plot([0, 0], [-100000, 100000], 'black')
title("Il faut s'imaginer la courbe avec ces points données :")

show()

```

DECRYPTAGE

- **Kmeans**

```python
from random import randint
from math import sqrt
import matplotlib.pyplot as plt

#Fonction qui retourne la moyenne des valeurs d'une liste
def moyenne(liste):
    somme = 0
    for i in liste:
        somme += i
    moyenne = somme/len(liste)
    return moyenne

#Choix des centres de départ (ici aléatoirement)
centre1 = randint(0,200)
centre2 = randint(0,200)

print(centre1, centre2)

#Création d'un dictionnaire qui recense les informations relatives aux 2 centres
#(ici on utilisera l'entrée "centreX" pour la liste des points associés à un centre et "hX" pour la liste des positions successives des centres)
dic = {}
dic["centre1"] = [centre1 + 0.01]
dic["centre2"] = [centre2 + 0.01]
dic["h1"] = [centre1]
dic["h2"] = [centre2]

#Création des variables
liste = []
for i in range(300):
    liste += [randint(0, 200)]

#Assignation des variables aux centres les plus proches, et mise à jour des places des centres.

while centre1 != moyenne(dic["centre1"]) and centre2 != moyenne(dic["centre2"]): #Tant que les centres en restent pas identiques 2 fois de suite
    centre1 = moyenne(dic["centre1"]) #On actualise les valeurs des centres
    centre2 = moyenne(dic["centre2"])
    dic["centre1"] = [] #On vide les listes de points associés à chaque centre
    dic["centre2"] = []
    for i in liste:
        if abs(i - centre1) < abs(i - centre2): #Calcul de la distance la plus courte entre le point I et le centre1 ou le centre2
            dic["centre1"] += [i] #Et on assigne le point I au centre dont il est le plus proche
        else:
```

```
            dic["centre2"] += [i]
    print(round(moyenne(dic["centre1"]), 2), round(moyenne(dic["centre2"]), 2))
    dic["h1"] += [moyenne(dic["centre1"])] #On ajoute la nouvelle position à la liste des positions des centres
    dic["h2"] += [moyenne(dic["centre2"])]

#Juste des arrondis ^^
a = round(moyenne(dic["centre1"]), 2)
b = round(moyenne(dic["centre2"]), 2)

print("Les positions des centres sont maintenant", a, "et", b, ". Il y a", len(dic["centre1"]), "termes associés au 1er centre et", len(dic['centre2']), "associés au 2nd centre")

#Test avec pyplot...

#Affichage dse points assignés au centre1 en rouge (d'où le r. pour afficher des points rouges)
abscisse = []
for i in range(len(dic["centre1"])):
    abscisse += [0]
plt.plot(dic["centre1"], abscisse, 'r.')

#Affichage dse points assignés au centre2 en bleu (d'où le b. pour afficher des points bleus)
abscisse = []
for i in range(len(dic["centre2"])):
    abscisse += [0]
plt.plot(dic["centre2"], abscisse, 'b.')

#Affichage de ronds rouges et bleus pour les positions successives des centres rouges et bleus
for i in range(len(dic["h1"])):
    plt.plot(dic["h1"][i], -i + len(dic["h1"]), 'ro' )
    plt.plot(dic["h2"][i], -i + len(dic["h1"]), 'bo' )

#Affichage de points blancs invisibles pour fixer la fenêtre. En gros les coordonnées de la fenetre seraient (0;0, 200;X)
#mais grace à ces points on le transforme en (-50;-10, 250;X) avec X le nombre de centres à afficher
plt.plot([-10],[-0.5], '.', color='1')
plt.plot([210],[len(dic["h1"]) + 0.5], '.', color='1')

plt.show()

```

- **Stéganographie**

```python
# https://euler.ac-versailles.fr/isn/Traitement_images_1_annexe_2.pdf
from PIL import Image
im1 = Image.open("seville.png")
L,H = im1.size
im2 = Image.open("renne.png")
im3 = Image.new("RGB",(L,H)) # image destination (image sténographiée, « vide » pour l'instant)
for y in range(H):
    for x in range(L):
        p1 = im1.getpixel((x,y)) # acquisition du pixel (x,y) de l'image 1
        r1 = p1[0]&240 # et mise à zéro des 4 bits de droite R,V, et B
        v1 = p1[1]&240
        b1 = p1[2]&240
        p2 = im2.getpixel((x,y)) # acquisition du pixel (x,y) de l'image 2
        r2 = p2[0]>>4 # et décalage de 4 bits vers la droite R,V, et B
        v2 = p2[1]>>4
        b2 = p2[2]>>4
        r = r1|r2 # insertion du pixel de l'image 2 dans celui de l'image 1
        v = v1|v2
        b = b1|b2
        im3.putpixel((x,y),(r,v,b)) # écriture de l'image sténographiée pixel par pixel
im3.save("seville_stegano.png")
im3.show()
```

```python
from PIL import Image  #on importe la bibliothèque qui traite les images
im = Image.open("message.png")  #on cherche l'image que nous allons utiliser

w,h=im.size                     #on récupère la taille de l'image grâce à PIL

r,g,b=im.split()                #On éclate l'image en trois (rouge vert bleu)

r=list(r.getdata())             #on transforme l'image en liste, c'est à dire la liste des pixels

#le message à  coder
c="Il faut blanchir les champignons! Ou un truc comme ça... dans le genre moisi, quoi"
#on note la longueur de la chaine et on la transforme en binaire
u=len(c)
v=bin(len(c))[2:].rjust(8,"0")
#on transforme la chaine en une liste de 0 et de 1
ascii=[bin(ord(x))[2:].rjust(8,"0") for x in c]
#transformation de la liste en chaine
a=''.join(ascii)
#on code la longueur de la liste dans les 8 premiers pixels rouges
for j in range(8):
    r[j]=2*int(r[j]//2)+int(v[j])

#on code la chaine dans les pixels suivants
for i in range(8*u):
    r[i+8]=2*int(r[i+8]//2)+int(a[i])
#on recrée l'image rouge
nr = Image.new("L",(16*p,16*q))
nr = Image.new("L",(w,h))
nr.putdata(r)
#fusion des trois nouvelles images
imgnew = Image.merge('RGB',(nr,g,b))
imgnew.save("messagecode.png")
```

```python
from PIL import Image
im = Image.open("messagecode.png")
r,g,b=im.split()
r=list(r.getdata())
#lecture de la longueur de la chaine
p=[str(x%2) for x in r[0:8]]
q="".join(p)
q=int(q,2)
#lecture du message
n=[str(x%2) for x in r[8:8*(q+1)]]

b="".join(n)
message=""

for k in range(0,q):
    l=b[8*k:8*k+8]
    message=message+chr(int(l,2))
print (message)
```

- **Sac à dos**

```python
def dec_to_bin(x):
    return int(bin(x)[2:])

def truc(message): #décimal en binaire en oubliant les espaces dans
#le message ET IL FAUT CODER CHAQUE CARACTERE SUR (ici) 10 BITS.
    count = 0
    texte = ''
    var = str(message) + ' '
    for i in range(len(var)):
        if (message + ' ')[i] != ' ':
            count += 1
        else:
            tr = ''
            tr = str(dec_to_bin(int(message[i-count:i])))
            while len(tr)<10:
                tr = '0' + tr
            texte += tr
            count = 0
    return texte

def decoupe(message, taille): #découpe par paquets
    NvMessage = ''
    for i in range(len(message)):
        NvMessage += message[i]
        if (i+1)%taille == 0:
            NvMessage += ' '
    return NvMessage

def decoupe2(message, taille): #découpe par paquets
    NvMessage = ''
    MsgListe = []
    for i in range(len(message)):
        NvMessage += message[i]
        if (i+1)%taille == 0:
            MsgListe += [NvMessage]
            NvMessage = ''
    return MsgListe

#on entre les variables a,b,c

a = int(input("1er chiffre de la séquence super-croissante: (1)",))
b = int(input("2eme chiffre de la séquence super-croissante: (2)",))
c = int(input("3eme chiffre de la séquence super-croissante: (4)",))
d = int(input("4eme chiffre de la séquence super-croissante: (7)",))
e = int(input("5eme chiffre de la séquence super-croissante: (12)",))
f = int(input("6eme chiffre de la séquence super-croissante: (20)",))
g = int(input("7eme chiffre de la séquence super-croissante: (33)",))
h = int(input("8eme chiffre de la séquence super-croissante (54):",))
i = int(input("9eme chiffre de la séquence super-croissante: (88)",))
j = int(input("10eme chiffre de la séquence super-croissante: (143)",))
N = int(input("Composante de la clef privée:",))
M = int(input("Composante de la clef privée:",))

#print(a,b,c,d,e,f)


liste = [a,b,c,d,e,f, g, h, i, j]
knapsack = []

for lettre in liste:
    knapsack += [lettre*N%M]

#print (knapsack)

message = input("Votre message à coder en décimal:",)

print(message)

message = truc(message)

messageliste = decoupe2(message, len(knapsack))

chiffre = 0
messagecode = []

for i in range(len(messageliste)):
    for j in range(len(messageliste[i])):
        if messageliste[i][j] == '1':
            chiffre += knapsack[j]
    messagecode += [chiffre]
    chiffre = 0

print(messagecode)
```

```python
def bin_to_dec(x):
    return int(str(x), 2)

def decoupe3(x):
    liste = []
    transit = ''
    for i in x:
        if i != ' ':
            transit += i
        else:
            if transit != '':
                liste += [transit]
                transit = ''
    liste += [transit]
    return liste

def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, x, y = egcd(b % a, a)
        return (g, y - (b // a) * x, x)

def inv_mod(b, n):
    g, x, _ = egcd(b, n)
    if g == 1:
        return x % n

def rang_liste(x, liste):
    for i in range(len(liste)):
        if liste[i] == x:
            return i

a = int(input("1er chiffre de la séquence super-croissante: (1)",))
b = int(input("2eme chiffre de la séquence super-croissante: (2)",))
c = int(input("3eme chiffre de la séquence super-croissante: (4)",))
d = int(input("4eme chiffre de la séquence super-croissante: (7)",))
e = int(input("5eme chiffre de la séquence super-croissante: (12)",))
f = int(input("6eme chiffre de la séquence super-croissante: (20)",))
g = int(input("7eme chiffre de la séquence super-croissante: (33)",))
h = int(input("8eme chiffre de la séquence super-croissante (54):",))
i = int(input("9eme chiffre de la séquence super-croissante: (88)",))
j = int(input("10eme chiffre de la séquence super-croissante: (143)",))
```

```python
N = int(input("Composante de la clef privée:",))
M = int(input("Composante de la clef privée:",))

liste = [j, i, h, g, f, e, d, c, b, a]
listt = [a, b, c, d, e, f, g, h, i, j]

messagecode = input("Entrez le message à décoder")
nouveaumessage = decoupe3(messagecode)
messag = []

for i in nouveaumessage:
    messag += [(int(i)*inv_mod(N, M))%M]

print(messag)

var = 0
war = []
xar = []
for i in range(len(messag)):
    var = messag[i]
    for j in liste:
        if var >= j:
            var = var - j
            war += [j]
    xar += war + ['STOP']
    war = []

yar = []
aar = ''

for i in liste:
    aar += '0'
dar = aar

for i in xar:
    if i == 'STOP':
        yar += [aar]
        aar = dar
    else:
        for j in listt:
            if i == j:
                aar = aar[:rang_liste(j, listt)] + '1' + aar[rang_liste(j, listt)+1:]
```

- **PhotoMaton**

```
import cv2;
img = cv2.imread('Tes

#Initialisation de to
img2 = img
a = img
b = img
c = img
d = img
rep = input('Nb de repetitions ?')

for F in range(int(rep)):

#Création des 4 nouvelles images
    for i in range(int(img.shape[0]/2)):
        for j in range(int(img.shape[1]/2)):
           a[i][j] = img[2*i][2*j]
           b[i][j] = img[2*i][2*j+1]
           c[i][j] = img[2*i+1][2*j]
           d[i][j] = img[2*i+1][2*j+1]

#Placement des ces images dans la nouvelle image
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if i < img.shape[0]/2:
                if j < img.shape[1]/2:
                    img2[i][j] = a[i][j] #En haut a gauche
                else:
                    img2[i][j] = c[i][j-((img.shape[1]+1)/2)] #En haut a droite
            else:
                if j < img.shape[1]/2:
                    img2[i][j] = b[i-((img.shape[0]+1)/2)][j] #En bat a gauche
                else:
                    img2[i][j] = d[i-((img.shape[0]+1)/2)][j-((img.shape[1]+1)/2)] #En bas a droite

#Création d'un document contenant l'image
    name = 'Tadam' + str(F)
    cv2.imwrite(name+'.pgm', img2)

#Initialisation de la nouvelle image en l'image à coder, pour recommencer si besoin est.
    img = img2

print(yar)

Dernier = []
for i in yar:
    Dernier += [bin_to_dec(i)]
print(Dernier)
```

- **Page HTML**

```
<strong><center>Cryptographie - Machine Learning</center></strong>
<strong><center>Hugo & Mathilde </center></strong>
</div>
</div>
<ul id="nav"><!--
    --><li><a href="..\index.html">Accueil</a></li><!--
    --><li><a href="..\classique\cryptageclass.html">Cryptages classiques</a></li><!--
    --><li><a href="..\moderne\cryptagemod.html">Cryptages modernes</a></li><!--
    --><li><a href="..\partage\secret3.html">Partage secret à 3 ou plus</a></li><!--
    --><li><a href="..\algoK\kmeans.html">Algo Kmeans</a></li><!--
    --><li><a href="..\stega\stega.html">Stéganographie</a></li><!--
    --><li><a href="..\extensions\encore.html">Autres</a></li>
</ul>
```

```
        <h2><li><b>Cryptage Secret à 3</b></li></h2>
</ul>

<p align="justify"> Le partage de clé secrète de Shamir est un algorithme
de cryptographie. C'est une forme de partage de secret, où un secret est
divisé en parties, donnant à chaque participant sa propre clé partagée, où
certaines des parties ou l'ensemble d'entre elles sont nécessaires afin de
reconstruire le secret. <br/>
Pour le secret à 3, on se base sur des fonctions et coordonnées
mathématiques. Comme son nom l'indique, il nous faut donc 3 points donnés
à 3 personnes pour retrouver une fonction polynôme (de forme ax²+bx+c): si
on ne donne que 2 points on ne peut pas déduire la fonction passant par
les points car on aurait une droite, il existerait donc une infinité de
parabole passant par les 2 points.
</p>
<center><td><img src="..\image\secret.PNG" width="200" height="200" alt=
"Courbe"></center></td>

<br/>

<p align="justify"> Il nous faut donc impérativement 3 personnes (ou plus
pour Hugo) pour résoudre un système, une personne à un seul et même point
qui lui est attribué: (xi;yi) on a donc: <br/>

<div style="text-indent: 50px;" >( x1 ; y1 ) → Personne 1<br/>
<div style="text-indent: 50px;" >( x2 ; y2 ) → Personne 2<br/>
<div style="text-indent: 50px;" >( x3 ; y3 ) → Personne 3</p>
```

```
    <ul>
        <li><b>Cryptage Enigma</b></li>
    </ul>

    <p align="justify"> La machine Enigma est une machine de codage qui a été utilisée par les forces alliées et les forces de l'axe pendant la Seconde Guerre Mondiale.
    Elle est basée sur une idée du hollandais Hugo Alexander Koch et cette idée a été reprise par l'allemand Arthur Scherbius, qui fabriqua le premier modèle
    commercialisable en 1923.<br/>

    Le principe de cette machine est assez simple : lorsqu'on appuyait sur une touche de son clavier, un courant électrique en partait, traversait 3 rotors (des roue
    trouées par lesquelles passait le courant), était renvoyé dans un autre trou du dernier rotor, refaisait le chemin à l'envers et ressortait derrière une petite lampe
    qui s'allumait et éclairait une lettre. Si ce n'était que cela, cela ressemblerait un une substitution classique, et il suffirait de connaître les positions des
    rotors pour déchiffrer le codage, mais les rotors tournent à chaque fois que l'on entre une lettre, ce qui fait que si l'on rentre plusieurs fois de suite la même
    lettre, elle sera codée différemment à chaque fois. Ce type de codage est dit symétrique car il décode et code de la même manière : </p>
    <ul>Si "message" est codé par "qwertyu", "qwertyu" sera codé par "message", ce qui rend le déchiffrage très facile si on sait quelle était la configuration de la
    machine de la personne qui a codé.
    </ul>


    <p align="justify">Le seul défaut technique de cette machine de codage était qu'une lettre n'est jamais codée par elle-même, ce qui permettait plus facilement de
    décoder les messages (un chiffreur allemand a un jour envoyé un message ne contenant que des T, quand les britanniques l'ont intercepté, ils ont vu un message sans
    aucun T, en ont déduit que ce message était composé uniquement de T et ont pu en déduire la position des rotors). Les autres défauts sont dus aux messages qui
    terminaient tous par HEIL HITLER ou qui contenaient souvent des mots comme KEINE BESONDERE EREIGNISSE, WETTER ou JUDE. </p>

    <br/>

    <iframe src="https://trinket.io/embed/python/1a0b875720" width="100%" height="400" frameborder="0" marginwidth="0" marginheight="0" allowfullscreen></iframe>
```