UNIVERSITE D'ABOMEYCALAVI
(BENIN)

THE ABDUS SALAM
INTERNATIONAL CENTRE FOR
THEORETICAL PHYSICS (ITALY)

INSTITUT DE MATHEMATIQUES ET DE SCIENCES PHYSIQUES

Centre d'Excellence Africain en Sciences Mathématiques, Informatique et Applications (CEA-SMIA)

PROJET SCIENTIFIQUE
POUR L'OBTENTION DU DIPLÔME DE LICENCE SPÉCIALE DES CLASSES PRÉPARATOIRES

Option : Mathématiques

Thème

Algorithmes d'Euclide et de Bézout avec Initiation aux chiffrements

Présenté par :

OLADÉ JUSTE LUC OGODJA
ogodjajusteluc@gmail.com

Superviseurs

PROFESSEUR GUY DEGLA
gdegla@imsp-uac.org

DOCTEUR JAPHET ODJOUMANI
japhet.odjoumani@imsp-uac.org

Dédicace

À nos chers parents et à nos amis,
Nous dédions ce travail.
Juste OGODJA & Romain HOUNSOUNON

Remerciements

Aux familles HOUNSOUNON et OGODJA, en reconnaissance de votre amour inconditionnel et de votre soutien constant qui ont été les fondements de notre passion pour les mathématiques et l'informatique, nous exprimons notre profonde gratitude.

À nos professeurs des classes préparatoires de l'IMSP, en particulier au Directeur de l'IMSP Monsieur Carlos OGOUYANDJOU, au Directeur-Adjoint de l'IMSP Monsieur Vincent MONWANOU, au Coordonnateur du Projet d'Excellence Africain en Sciences Mathématiques, Informatique et Applications Monsieur Joël TOSSA, et à nos encadreurs Monsieur Guy DEGLA et Monsieur Japhet ODJOUMANI, nous adressons nos plus sincères remerciements pour votre expertise, vos conseils avisés et votre accompagnement bienveillant, qui ont été essentiels à notre réussite.

À nos proches, amis et personnes anonymes ayant contribué de près ou de loin à ce travail, nous vous sommes reconnaissants. Cette expérience restera inoubliable et nous servira dans notre quête de savoir et d'excellence.

Merci infiniment.

Juste OGODJA & Romain HOUNSOUNON

Table des matières

Introduction

1. Introduction

Cette étude portera sur l'algorithme d'Euclide, l'algorithme de Bézout et leur utilisation dans le domaine de la cryptographie, plus précisément dans le chiffrement. Il explorera les fondements mathématiques nécessaires à la compréhension de ces algorithmes et présentera des exemples concrets d'applications cryptographiques.

L'objectif de ce projet est de fournir une analyse approfondie de ces algorithmes et d'initier le lecteur aux concepts de base des chiffrements.

1.1 Présentation

Depuis des temps immémoriaux, les êtres humains ont cherché à protéger leurs secrets et leurs communications sensibles. C'est dans cette quête de confidentialité que l'art mystérieux du chiffrement a vu le jour. À travers les siècles, de nombreuses méthodes de chiffrement ont été développées, mais deux algorithmes en particulier ont résisté à l'épreuve du temps et se sont avérés d'une importance cruciale : ***l'algorithme d'Euclide et l'algorithme de Bézout.***

Dérivé des travaux du mathématicien grec antique **Euclide**, l'***algorithme d'Euclide*** a été l'un des premiers algorithmes jamais conçus. Son objectif initial était de trouver le plus grand commun diviseur (PGCD) de deux entiers, mais son utilité va bien au-delà de cette simple tâche. Grâce à son élégance et à son efficacité, l'***algorithme d'Euclide*** est rapidement devenu l'un des outils préférés des cryptographes.

En parallèle, l'***algorithme de Bézout***, attribué au mathématicien français **Étienne Bézout**, a été développé pour résoudre des équations diophantiennes, qui sont des équations où les solutions sont restreintes aux entiers. Cependant, les propriétés de l'***algorithme de Bézout*** se sont avérées particulièrement utiles dans le contexte du *chiffrement*, car il permet de déterminer les *coefficients de Bézout*, qui jouent un rôle essentiel dans le célèbre ***chiffrement RSA***(Rivest-Shamir-Adleman).

Ces deux algorithmes, l'***algorithme d'Euclide*** et l'***algorithme de Bézout***, sont étroitement liés dans le domaine du chiffrement. Leur compréhension nous ouvre les portes vers des méthodes de codage et de décodage sophistiquées, qui protègent non seulement les informations confidentielles, mais qui jouent également un rôle crucial dans la sécurité des transactions en ligne, des communications gouvernementales et bien d'autres domaines de la vie moderne.

Dans cette initiation aux chiffrements, nous explorerons en profondeur les principes sous-jacents de l'algorithme d'Euclide et de l'algorithme de Bézout, et nous découvrirons comment ces joyaux mathématiques continuent de façonner le paysage complexe de la cryptographie contemporaine. Préparez-vous à plonger dans le monde fascinant des chiffres,

des diviseurs et des équations, pour comprendre comment ces algorithmes ingénieux permettent de préserver la confidentialité des informations les plus vitales de notre ère numérique.

1.2 Objectifs

Les objectifs de ce projet sur les algorithmes de Bézout et d'Euclide avec initiation aux chiffrements incluent :

- Comprendre les concepts fondamentaux de la cryptographie ;
- Étudier les algorithmes de Bézout et d'Euclide ;
- Analyser quelque(s) méthode(s) de chiffrement basées sur les algorithmes de Bézout et d'Euclide ;
- Mettre en œuvre des exemples de chiffrement basés sur les algorithmes de Bézout et d'Euclide ;
- Évaluer la complexité des algorithmes de Bézout et d'Euclide.

2. Fondements mathématiques

Cette section présentera les concepts mathématiques essentiels pour la compréhension des algorithmes d'Euclide et de Bézout et de certains chiffrements, elle abordera les notions de congruence, de nombres premiers, etc. Ces concepts serviront de base pour l'analyse ultérieure des algorithmes et la présentation des chiffrements.

Introduction à la théorie des nombres

La théorie des nombres est une branche passionnante des mathématiques qui se concentre sur l'étude des propriétés et des comportements des nombres entiers. Elle explore des concepts fondamentaux tels que les nombres premiers, les congruences et les motifs dans les suites numériques. Cette discipline joue un rôle essentiel dans de nombreux domaines, y compris la cryptographie et les problèmes mathématiques non résolus, suscitant ainsi un intérêt et une fascination continus pour ses énigmes et ses découvertes potentielles.

2.1 Divisibilité

Définition 2.1 Soient a et b deux entiers.
On dit que b divise a, s'il existe $q \in \mathbb{Z}$ tel que $a = qb$.

Exemple

- 1 divise tout entier a. En effet : $a = a \cdot 1$;
- Tout entier b divise 0. En effet : $0 = 0 \cdot b$.

Remarques

1. 0 ne divise que 0 ;
2. Les seuls diviseurs de 1 sont 1 et -1 ;
3. Si b divise a, on dit aussi a est un multiple de b.

2.2 Divisibilité dans $\mathbb{Z}$

Théorème 2.1 Soient a, b deux entiers.
Si $b \neq 0$, alors il existe un couple $(q,r) \in \mathbb{Z}^2$ tels que $a = qb + r$ et $0 \leq r < |b|$.

Preuve

Pour montrer l'existence on a deux cas :

1. Si $b > 0$, soit r le plus petit élément positif ou nul de $a - b\mathbb{Z}$,
   $a - b\mathbb{Z} = \{..., a+2b, a+b, a, a-b, a-2b, ...\}$. On a $r = a - bq$ pour un certain $q \in \mathbb{Z}$ et $r \geq 0$. Pour montrer que $r < |b| = b$, on suppose $b \leq r$, alors on aura $0 \leq r - b = a - (q+1)b \in a - b\mathbb{Z}$, donc $r \leq r - b$(car r est le plus petit), ce qui est impossible car $b > 0$.
2. Si $b < 0$, alors $-b > 0$ et d'après le cas précédent, il existe (q,r) tel que $a = q(-b) + r$ avec $0 \leq r < -b$, ce qui revient à dire que $a = (-q)b + r$ avec $0 \leq r < |b| = -b$

Dans les deux cas : $\exists\ (q,r) \in \mathbb{Z}^2$ tel que $a = qb + r$ et $0 \leq r < |b|$

2.3 Sous groupes de $(\mathbb{Z}, +)$

Définition 2.2 Un sous groupe de ($\mathbb{Z}$, +) est un sous ensemble non vide de $\mathbb{Z}$ stable par sommation et par passage au symétrique. C'est-à-dire :
Un sous ensemble H de $\mathbb{Z}$ est un sous groupe de ($\mathbb{Z}$, +) , si, et seulement si

$$H \neq \emptyset \text{ et } \forall\, x, y \in H : (x + y \in H \text{ et } -x \in H).$$

Exemple

On peut facilement vérifier que les ensembles $a\mathbb{Z}$ sont des sous groupes de $(\mathbb{Z}, +)$.
Le théorème suivant montre que tous les sous groupes de $(\mathbb{Z}, +)$ sont de la forme $a\mathbb{Z}$.

Théorème 2.2 Soit H un sous groupe de $(\mathbb{Z}, +)$.
Il existe un unique entier naturel a, tel que $H = a\mathbb{Z}$.

Preuve

1er cas : Si $H = \{0\}$, alors $H = 0\mathbb{Z}$ et 0 est l'unique entier a vérifiant $a\mathbb{Z} = \{0\}$.

2ème cas : Si $H \neq \{0\}$, Soit a le plus petit élément strictement positif de H.
Comme H est un sous groupe de $(\mathbb{Z}, +)$, alors $as = \underbrace{a + a + ... + a}_{\text{s termes}} \in H$ d'où $a\mathbb{Z} \subset H$.

Inversement ; pour tout $x \in H$, il existe $(q,r) \in \mathbb{Z}^2$ tel que $r = x - qa$, avec $0 \leq r < a$.
Comme H est un sous groupe et $x \in H$, $qa \in H$, alors $x - qa \in H$ et le fait que $0 \leq r < a$, implique $r = 0$ (car a est le plus petit entier naturel appartenant à H).
Par conséquent $x = aq$ et $H \subset a\mathbb{Z}$
Les deux inclusions obtenues donnent l'égalité $H = a\mathbb{Z}$. Si $a\mathbb{Z} = a'\mathbb{Z}$, alors $a = a't$ et $a' = at'$ donc a divise a' et a' divise a , d'où l'égalite $a = a'$ et l'unicité de a.

Proposition 2.1 Soient a et b deux entiers.
1. a divise b, si et seulement, si $b\mathbb{Z} \subset a\mathbb{Z}$.
2. $a = \pm b$, si et seulement si $b\mathbb{Z} = a\mathbb{Z}$.

2.4 Notions de pgcd et de ppcm

Soient $a_1, a_2, ..., a_p$ des entiers. On rappelle que

$$\bigcap_{i=1}^{p} a_i\mathbb{Z} = a_1\mathbb{Z} \cap a_2\mathbb{Z} \cap ... \cap a_p\mathbb{Z} \text{ et } \sum_{i=1}^{p} a_i\mathbb{Z} = a_1\mathbb{Z} + a_2\mathbb{Z} + ... + a_p\mathbb{Z}$$

Théorème 2.3 Soient $a_1, a_2, ..., a_p$ des entiers.
1. Il existe un entier naturel d vérifiant $\sum_{i=1}^{p} a_i\mathbb{Z} = \boldsymbol{d}\mathbb{Z}$;
2. Il existe un entier naturel unique m vérifiant $\bigcap_{i=1}^{p} a_i\mathbb{Z} = \boldsymbol{m}\mathbb{Z}$.

d s'appelle le **plus grand commun diviseur** de la famille $a_1, a_2, ..., a_p$ et il est noté $\boldsymbol{pgcd}(a_1, a_2, ..., a_p)$;

m s'appelle le **plus petit commun multiple** de la famille $a_1, a_2, ..., a_p$ et il est noté $\boldsymbol{ppcm}(a_1, a_2, ..., a_p)$.

Preuve

1. Montrons que $\sum_{i=1}^{p} a_i\mathbb{Z}$ est un sous groupe de $(\mathbb{Z}, +)$. En effet : Si $x, y \in \sum_{i=1}^{p} a_i\mathbb{Z}$, alors $x = a_1x_1 + a_2x_2 + ... + a_px_p$ et y = $a_1y_1 + a_2y_2 + ... + a_py_p$, d'où $x + y = a_1(x_1 + y_1) + a_2(x_2 + y_2) + ... + a_p(x_p + y_p) \in \sum_{i=1}^{p} a_i\mathbb{Z}$ et $-y = a_1(-y_1) + a_2(-y_2) + ... + a_p(-y_p) \in \sum_{i=1}^{p} a_i\mathbb{Z}$.
   Donc $\sum_{i=1}^{p} a_i\mathbb{Z}$ est un sous groupe de $(\mathbb{Z}, +)$ et par application du **théorème 2.2** on conclut l'existence et l'unicité de l'entier naturel $\boldsymbol{d}$ vérifiant $\sum_{i=1}^{p} a_i\mathbb{Z} = \boldsymbol{d}\mathbb{Z}$.
2. Montrons que $\bigcap_{i=1}^{p} a_i\mathbb{Z}$ est un sous groupe de $(\mathbb{Z}, +)$. En effet : Si $x, y \in \bigcap_{i=1}^{p} a_i\mathbb{Z}$, alors $\forall\, i \in \{1, 2, ..., p\}$ on a $x \in a_i\mathbb{Z}$ et $y \in a_i\mathbb{Z}$ d'où $x + y \in a_i\mathbb{Z}$ et $-y \in a_i\mathbb{Z}$.
   Donc $\bigcap_{i=1}^{p} a_i\mathbb{Z}$ est un sous groupe de $(\mathbb{Z}, +)$ et par application du **théorème 2.2** on conclut l'existence et l'unicité de l'entier naturel $\boldsymbol{m}$ vérifiant $\bigcap_{i=1}^{p} a_i\mathbb{Z} = \boldsymbol{m}\mathbb{Z}$.

Exemple :

$4\mathbb{Z} = \{..., -12, -8, -4, 0, 4, 8, 12, ...\}$ et $6\mathbb{Z} = \{..., -12, -6, 0, 6, 12, ...\}$ alors $4\mathbb{Z} \cap 6\mathbb{Z} = 12\mathbb{Z}$ donc $\boldsymbol{ppcm}(4, 6) = 12$.

2.5 Caractérisation du pgcd et du ppcm

Théorème 2.4 Soient $a_1, a_2, ..., a_p$ des entiers.

1. L'entier naturel $\boldsymbol{d}$ est le $\boldsymbol{pgcd}(a_1, a_2, ..., a_p)$, si et seulement, si
   i. $\forall\, k \in \{1, 2, ..., p\}$ $\boldsymbol{d}$ divise a_k ;
   ii. $(\forall\, k \in \{1, 2, ..., p\}$ $\boldsymbol{d'}$ divise $a_k) \Rightarrow \boldsymbol{d'}$ divise $\boldsymbol{d}$.
2. L'entier naturel $\boldsymbol{m}$ est le $\boldsymbol{ppcm}(a_1, a_2, ..., a_p)$, si et seulement, si
   i. $\forall\, k \in \{1, 2, ..., p\}$ a_k divise $\boldsymbol{m}$;
   ii. $(\forall\, k \in \{1, 2, ..., p\}$ a_k divise $\boldsymbol{m}) \Rightarrow \boldsymbol{m}$ divise $\boldsymbol{m'}$.

Preuve

1. i. Si $d = pgcd(a_1, a_2, ..., a_p)$, pour tout k on a $a_k\mathbb{Z} \subset \sum_{k=1}^{p} a_k\mathbb{Z} = d\mathbb{Z}$
   donc d divise a_k, pour tout k.

   ii. Si d' divise tous les a_k, alors $a_k\mathbb{Z} \subset d'\mathbb{Z}$ pour tout k , d'où $d'\mathbb{Z} \supset \sum_{k=1}^{p} a_k\mathbb{Z} = d\mathbb{Z}$
   et d' divise d.

Inversement, soit d un entier naturel vérifiant i. et ii.
D'après i. , d divise tous les a_k donc $a_k\mathbb{Z} \subset d\mathbb{Z}$ et $\sum_{k=1}^{p} a_k\mathbb{Z} \subset d\mathbb{Z}$, d'où d divise $pgcd(a_1, a_2, ..., a_p)$.
En utilisant ii. et le fait que, $\forall\, k \in \{1, 2, ..., p\}$: $pgcd(a_1, a_2, ..., a_p)$ divise a_k, on conclut que $pgcd(a_1, a_2, ..., a_p)$ divise d , d'où l'égalité $pgcd(a_1, a_2, ..., a_p) = d$.

L'assertion 2. se démontre de la même façon que l. ∎

2.6 Propriétés du pgcd et du ppcm

Théorème 2.5 Soient $a_1, a_2, ..., a_p$ des entiers.

1. Le $\boldsymbol{pgcd}$ et le $\boldsymbol{ppcm}$ ne changent pas en permutant les a_i ;

2. $\boldsymbol{pgcd}(\pm a_1, \pm a_2, ..., \pm a_p) = \boldsymbol{pgcd}(a_1, a_2, ..., a_p)$
   et $\boldsymbol{ppcm}(\pm a_1, \pm a_2, ..., \pm a_p) = \boldsymbol{ppcm}(a_1, a_2, ..., a_p)$;

3. Pour tout $c \in \mathbb{Z}$ on a $\boldsymbol{pgcd}(ca_1, ca_2, ..., ca_p) = |c|\boldsymbol{pgcd}(a_1, a_2, ..., a_p)$
   et $\boldsymbol{ppcm}(ca_1, ca_2, ..., ca_p) = |c|\boldsymbol{ppcm}(a_1, a_2, ..., a_p)$;

4. Pour tout $q \in \{1, 2, ..., p-1\}$, on a :
   $\boldsymbol{pgcd}(a_1, a_2, ..., a_p) = \boldsymbol{pgcd}(\boldsymbol{pgcd}(a_1, a_2, ..., a_q)\ , \boldsymbol{pgcd}(a_{q+1}, a_{q+2}, ..., a_p))$
   et $\boldsymbol{ppcm}(a_1, a_2, ..., a_p) = \boldsymbol{ppcm}(\boldsymbol{ppcm}(a_1, a_2, ..., a_q)\ , \boldsymbol{ppcm}(a_{q+1}, a_{q+2}, ..., a_p))$.
   (C'est-à-dire $\boldsymbol{pgcd}$ et $\boldsymbol{ppcm}$ sont associatifs)

Preuve

La preuve est donnée seulement pour le *pgcd* car elle est analogue pour le *ppcm*.

1. $\sum_{i=1}^{p} a_i\mathbb{Z}$ ne change pas en permutant les a_i d'où le résultat.
2. Pour tout i , on a $\pm a_i\mathbb{Z} = a_i\mathbb{Z}$, donc $\sum_{i=1}^{p} a_i\mathbb{Z}$ ne change pas en remplaçant les a_i par $\pm a_i$, d'où le résultat.
3. Pour tout i , on a $ca_i\mathbb{Z} = \pm ca_i\mathbb{Z} = |c|a_i\mathbb{Z}$, donc $ca_i\mathbb{Z} = |c|a_i\mathbb{Z}$,d'où le résultat.
4. $\sum_{i=1}^{p} a_i\mathbb{Z} = \sum_{i=1}^{q} a_i\mathbb{Z} + \sum_{i=q+1}^{p} a_i\mathbb{Z}$, donc le *pgcd* est associatif.

Exemple

$$\begin{aligned} pgcd(-2,4,-7) &= pgcd(2,4,7) = pgcd(pgcd(2,4),pgcd(7)) \\ &= pgcd(2pgcd(1,2),7) = pgcd(2,7) = 1. \end{aligned}$$

2.7 Nombres premiers entre eux

Définition 2.3 Soit $(a_1,a_2,...,a_p) \in \mathbb{Z}^p$.

1. On dit que $a_1,a_2,...,a_p$ sont **premiers entre eux** si $\boldsymbol{pgcd}(a_1,a_2,...,a_p) = 1$;
2. On dit que $a_1,a_2,...,a_p$ sont **deux à deux premiers entre eux** si $\boldsymbol{pgcd}(a_i,a_j) = 1$ pour tout $i \neq j$ (i, $j \in \{1,2,...,p\}$).

2.8 Identité de Bezout

Théorème 2.6 Soient a, b des entiers. Il existe u**,** $v \in \mathbb{Z}$ tels que

$$au + bv = \mathbf{pgcd}(a, b).$$

— Les entiers u, v sont des ***coefficients de Bézout***.
— Ils s'obtiennent en « remontant » l'algorithme d'Euclide.

2.9 Théorème de Bezout

Corollaire 2.1 Les entiers $a_1,a_2,...,a_p$ sont **premiers entre eux** si, et seulement, s'il existe $(u_1,u_2,...,u_p) \in \mathbb{Z}^p$ tel que $\sum_{i=1}^{p} a_iu_i = 1$.

Preuve

On a pgcd(a_1, a_2, ..., a_p) = 1 donc $\sum_{i=1}^{p} a_i\mathbb{Z} = 1\mathbb{Z}$, alors il existe $(u_1,u_2,...,u_p) \in \mathbb{Z}^p$ tel que $\sum_{i=1}^{p} a_iu_i = 1$.

Inversement, si $\sum_{i=1}^{p} a_i u_i = 1$, alors tout diviseur commun des a_i divise 1,
donc $pgcd(a_1, a_2, ..., a_p)$ divise 1, alors il est égal a 1 .

2.10 Théorème de Gauss

Théorème 2.7 Soient a, b, c des entiers.
Si a ***divise*** bc et, si a et b sont ***premiers entre eux***, alors a ***divise*** c.

Preuve

a et b premiers entre eux alors il existe u et v tels que $au + bv = 1$ en multipliant par $c = auc + bvc$ donc a divise c car a divise $auc + bvc$.

2.11 Résolution d'équations linéaires dans $\mathbb{Z}$

Soient a, b des entiers non nuls et c un entier quelconque et soit l'équation

$$ax + by = c \qquad \textbf{(1)}$$

d'inconnues x et y.

Soit $\boldsymbol{d} = \boldsymbol{pgcd}(a, b)$;

1. Si $\boldsymbol{d}$ ne divise pas $\boldsymbol{c}$, alors l'équation ***(1)*** n'a pas de solution ;

2. Si $\boldsymbol{d}$ divise $\boldsymbol{c}$ l'équation ***(1)*** est équivalente à l'équation $a'x + b'y = c' \qquad$ ***(1')***,
   où $a' = \frac{a}{d}$, $b' = \frac{b}{d}$ et $c' = \frac{c}{d}$.

On a $\boldsymbol{pgcd}(a', b') = 1$, alors il existe u' et v' dans $\mathbb{Z}$ tels que $a'u' + b'v' = 1$,
donc $a'u'c' + b'v'c' = c'$ et le couple $(x_0, y_0) = (u'c', v'c')$ est une solution particulière de l'équation ***(1)***.

Pour trouver toutes les solutions de ***(1)***, il suffit de remarquer que
$a'x + b'y = c' = a'x_0 + b'y_0$, d'où $a'(x - x_0) = -b'(y - y_0)$ et par application du **théorème de Gauss**, on conclut du fait que $pgcd(a', b') = 1$, que $x - x_0 = b'q$, $q \in \mathbb{Z}$, ce qui entraine $y - y_0 = -a'q$.

Inversement, il est clair que les couples trouvés $(x, y) = (b'q + x_0, -a'q + y_0)$, $q \in \mathbb{Z}$, vérifient l'équation ***(1)***.

Exemple 1

L'équation $4x+6y=7$ n'admet pas de solutions, car $pgcd(4,\,6)$ ne divise pas 7.

Exemple 2

Soit l'équation $4x+6y=8$. $pgcd(4,\,6)$ = 2 et l'équation donnée est équivalente à $2x+3y=4$ qui admet $(x_0,\,y_0)$ = (-1, 2) comme solution particulière,
ainsi $2x+3y$ = 2 (-1) + 3 (2), alors $2(x+1)=-3(y-2)$, donc

$$\begin{cases} x+1=3q & q\in\mathbb{Z} \\ y-2=-2q \end{cases} \tag{2.1}$$

et les couples (x,y) = $(3q-1,-2q+2)$, $q\in\mathbb{Z}$ sont toutes les solutions de l'équation donnée.

2.12 Congruences

Définition 2.4 Pour chaque $n\in\mathbb{N}$, soit la relation $\mathscr{R}_n$ définit sur $\mathbb{Z}$ par :

$$x\mathscr{R}_n y \text{ si, et seulement, si } n \text{ divise } y-x \ \blacksquare$$

1. La relation $\mathscr{R}_n$ est une ***relation d'équivalence***, appelée ***relation de congruence modulo*** n.

   $x\mathscr{R}_n y$ est note $x\equiv y[n]$ et se lit :" x est congru y modulo n".

2. L'ensemble des classes d'équivalences $\bar{x}$ de cette relation est note $\mathbb{Z}/n\mathbb{Z}$

$$\bar{x}=\{x+nq : q\in\mathbb{Z}\} \ \blacksquare$$

Preuve

Il est facile de monter que $\mathscr{R}_n$ est une relation d'equivalence.

Théorème 2.8 La relation de congruence modulo n est compatible avec les lois $+$ et $\times$.
C'est-à-dire : (x $\equiv y[n]$ et $x'\equiv y'[n]$) $\Rightarrow$ $(x+x'\equiv y+y'[n]$ et $xx'\equiv yy'[n])$ ∎

Preuve

on a :

$$\begin{cases} x\equiv y[n], & alors\ \ n\ \ divise\ \ (y-x)\ \ et\ \ (y'-x') \\ x'\equiv y'[n] \end{cases}$$

et comme

$$\begin{cases} (y+y')-(x+x')=(y-x)+(y'-x') \\ yy'-xx'=y(y'-x')+x'(y-x) \end{cases}$$

alors n divise $(y+y')-(x+x')$ et $yy'-xx'$
et par conséquent

$$\begin{cases} x+x'\equiv y+y'[n] \\ xx'\equiv yy'[n] \end{cases}$$

Exemple

Déterminons les entiers naturels n tels que 7 divise $2^n - 1$. Ce qui revient à trouver n tel que $2^n \equiv 1$ [7] . On a :

$2^0 \equiv 1[7]$
$2^1 \equiv 2[7]$
$2^2 \equiv 4[7]$
$2^3 \equiv 1[7]$

En utilisant la compatibilité de la congruence et sa transitivité, on peut montrer que la suite des restes de la division de 2^n par 7 est périodique, de période égale à 3,
donc $2^n \equiv 1$ [7] ,si et seulement si, $n = 3p$, $p \in \mathbb{N}$

2.13 Nombres premiers

Définition 2.5 On appelle nombre premier tout entier $p > 1$ dont les seuls diviseurs positifs sont 1 et p.

Exemple

- 2 est un nombre premier (et c'est le seul nombre premier pair)
- 1 n'est pas un nombre premier.
- -3 n'est pas un nombre premier.

Proposition 2.2 Soient p un nombre premier et $a, a_1, a_2, \ldots, a_q$ des entiers, alors :

1. p est premier avec a ou p divise a ;
2. Si p divise $\prod_{i=1}^{p} a_i$ alors p divise au moins l'un des a_i ∎

Preuve

1. Les seules diviseurs positifs de p sont 1 et p, alors ou bien $pgcd(a, p) = 1$, donc p est premier avec a, ou bien $pgcd(a, p) = p$ et p divise a.

2. Supposons que p ne divise aucun a_i , alors d'après l) , il est premier avec tous les a_i, et d'après, p est premier avec $\prod_{i=1}^{p} a_i$, ce qui est absurde.

Théorème 2.9 Tout entier $n > 1$ possède au moins un diviseur premier.

Preuve

L'Ensemble $\mathscr{D}_n$ des diviseurs de n, qui sont plus grands que 1 n'est pas vide (car $n \in \mathscr{D}_n$), alors il possède un plus petit élément p. Cet élément p est premier. En effet : Soit d un diviseur de p tel que $d > 1$, alors $d \in \mathscr{D}_n$, donc $d \geq p$ (car $p = \min \mathscr{D}_n$), d'où l'égalité $d = p$, alors p est premier.

2.14 Décomposition en facteurs premiers

Théorème 2.10 Pour tout entier $n > 1$, il existe, de façon unique, des nombres premiers $p_1 < p_2 < ... < p_r$ et des nombres entiers naturels $\alpha_1, \alpha_2, ..., \alpha_r$ tels que n = $\prod_{i=1}^{r} p_i^{\alpha_i}$ ∎

Preuve

Supposons qu'il y a des entiers $n > 1$ qui ne s'ecrivent pas sous forme de produit de facteurs premiers et soit n_0 le plus petit de ces entiers. n_0 n'est pas premier (Sinon il s'ecrira comme produit d'un seul facteur) , alors $n_0 = n_1 n_2$ tels que $n_1 > 1$, $n_2 > 1$ et $n_1 < n_0$, $n_2 < n_0$, donc n_1 et n_2 s'ecrivent comme produit de facteurs premiers, et par suite leur produit n_0 s'ecrit comme produit de facteurs premiers, ce qui est absurde.
D'où l'existence de la décomposition.

Pour l'unicite, supposons n = $\prod_{i=1}^{r} p_i^{\alpha_i} = \prod_{j=1}^{s} q_j^{\beta_j}$. On a chaque p_i divise $\prod_{j=1}^{s} q_j^{\beta_j}$, alors p_i divise au moins l'un des q_j , alors $p_i = q_j$.
Par conséquent $\{p_1, p_2, ..., p_r\} \subset \{q_1, q_2, ..., q_s\}$. De la même façon, on montre l'autre inclusion, d'où l'égalité $\{p_1, p_2, ..., p_r\} = \{q_1, q_2, ..., q_s\}$.

Alors $r = s$ et $p_i = q_i$, car $(p_i)_i$ et $(q_j)_j$ sont strictement ordonnés.

Pour montrer que $\alpha_i = \beta_i$, supposons $\alpha_i < \beta_i$ et éliminons p_i du premier membre de l'égalité $\prod_{i=1}^{r} p_i^{\alpha_i} = \prod_{i=1}^{s} q_i^{\beta_i}$, alors $\prod_{\substack{k=1 \\ k \neq i}}^{r} p_k^{\alpha_k} = q_i^{\beta_i - \alpha_i} \prod_{\substack{k=1 \\ k \neq i}}^{r} q_k^{\beta_k}$ et p_i divise le deuxième membre sans diviser le premier, ce qui est impossible. De la même façon, on montre qu'il est impossible d'avoir $\alpha_i > \beta_i$, d'où l'égalité $\alpha_i = \beta_i$ ∎

2.15 Théorème des chinois

Théorème 2.11 Soit m et n des entiers ***premiers entre eux***.
Alors quelque soit a et b entiers il existe des ***solutions simultanées*** de $x \equiv a \ (mod\ m)$ et $x \equiv b \ (mod\ n)$, et cette solution x est ***unique modulo*** mn.

2.15.0.1 Unicité

Soit x une solution simultanée des deux congruences, et soit y un deuxième entier. Alors y est aussi une solution des deux congruences si et seulement si on a $x \equiv y \ (mod$ m) et x $\equiv y \ (mod\ n)$. Alors $m|x-y$ et $n|x-y$, ce qui équivaut à ce que $ppcm(m,\ n)\ |x-y$ ou $x \equiv y \ (mod\ ppcm(m,n))$. Mais comme m et n sont premiers entre eux, on a $ppcm(m,n) = mn$.
Donc y est aussi une solution des deux congruences si et seulement si $x \equiv y \ (mod\ mn)$.

2.15.0.2 Existence

On cherche une relation de Bezout $mu+nv=1$. Alors on a $nv = 1-mu$ et $mu = 1-nv$. Il s'ensuit qu'on a

$$\begin{array}{llll} nv \equiv 1 & (mod & m), & nv \equiv 0 \quad (mod \quad n), \\ mu \equiv 0 & (mod & m), & mu \equiv 1 \quad (mod \quad n). \end{array}$$

Il s'ensuit que si on prend $x = anv + bmu$ on a bien $x \equiv a*1+b*0 \equiv a \ (mod\ m)$ et $x \equiv a*0+b*1 \equiv b \ (mod\ n)$.

Quand m ***et*** n ***ne sont pas premiers entre eux, on a le théorème suivant.***

Théorème 2.12 Soit m et n entiers naturels, et $\boldsymbol{d = pgcd}(m,n)$.
Alors il existe une ***solution simultanée*** x de $x \equiv a \ (mod\ m)$ et $x \equiv b \ (mod\ n)$ si et seulement si on a $a \equiv b \ (mod\ d)$. La solution x est ***unique modulo ppcm***(m,n) ∎

Existence

Soit $y = x-a$. On cherche y vérifiant $y \equiv 0 \ (mod\ m)$ et $y \equiv b-a \ (mod\ n)$.
Par le lemme de Bezout il existe u et v avec $mu+nv=d$. On a donc $mu \equiv 0 \ (mod\ m)$ et $mu \equiv d \ (mod\ n)$. Comme on a $a \equiv b \ (mod\ d)$, le nombre $\frac{b-a}{d}$ est entier. On prend $y = mu\frac{b-a}{d}$ et donc $x = a+mu\frac{b-a}{d}$ comme solutions.

Unicité

Similaire au cas où m et n sont premiers entre eux.

2.16 Petit théorème de Fermat

Théorème 2.13 Si p est un nombre premier et $a \in \mathbb{Z}$ alors $a^p \equiv a \ (mod\ \ p)$ ∎

Corollaire 2.2 Si p ne divise a alors $a^{p-1} \equiv 1 \ (mod\ \ p)$ ∎

Lemme

p divise C_p^k pour $1 \leq k \leq p-1$ c'est à dire $C_p^k \equiv 0\ (mod\ \text{p})$ ∎

Preuve

$C_p^k = \frac{p!}{k!(p-k)!}$ donc $p! = k!(p-k)!C_p^k$. Ainsi $p|k!(p-k)!C_p^k$.
Or $1 \leq k \leq p-1$ alors p ne divise pas $k!$ (sinon p divise l'un des facteurs de $k!$ mais il sont tous $< p$). De même p ne divise pas $(p-k)!$, donc par le lemme d'Euclide p divise C_p^k ∎

Preuve du théorème

Nous le montrons par récurrence pour les $a \geq 0$.

- Si $a = 0$ alors $0 \equiv 0\ (mod\ p)$.
- Fixons $a \geq 0$ et supposons que $a^p \equiv a\ (mod\ p)$. Calculons $(a+1)^p$ à l'aide de la formule du binôme de Newton :

$$(a+1)^p = a^p + a^{p-1}C_p^{p-1} + a^{p-2}C_p^{p-2} + ... + C_p^1 + 1$$

Réduisons maintenant modulo p :

$$\begin{aligned}(a+1)^p &\equiv a^p + a^{p-1}C_p^{p-1} + a^{p-2}C_p^{p-2} + ... + C_p^1 + 1 \quad (mod \quad p)\\ &\equiv a^p + 1 \quad (mod \quad p)\\ &\equiv a + 1 \quad (mod \quad p)\end{aligned}$$ ∎

- Par le principe de récurrence nous avons démontré le petit théorème de Fermat pour tout $a \geq 0$. Il n'est pas dur d'en déduire le cas des $a \leq 0$.

2.17 Petit théorème de Fermat généralisé

Définition 2.6 Soit un entier $n \geq 1$.
On appelle **fonction d'Euler** de n l'entier $\varphi(n)$ défini par :

$$\varphi(n) = card\{k \in \mathbb{N} : 0 < k < n \ \ et \ \ pgcd(k,n) = 1\}.$$

Exemple

- $\varphi(1) = \varphi(2) = 1$;
- $\varphi(3) = \varphi(4) = \varphi(6) = 2$;
- $\varphi(5) = 4$;
- $\varphi(8) = 4$ car les quatre nombres 1, 3, 5 et 7 sont premiers avec 8.

Quelques propriétés de la fonction d'Euler

Voici quelques propriétés permettant de le calculer :

- Si p est premier et $\alpha \geq 1$,

$$\varphi(p) = p-1 \text{ et } \varphi(p^\alpha) = p^\alpha - p^{\alpha-1}$$

- Si $pgcd(n,m) = 1$, alors

$$\varphi(nm) = \varphi(n)\varphi(m)$$

- En particulier, les propriétés précédentes permettent de calculer l'indicateur d'Euler $\varphi(n)$ connaissant la décomposition en facteurs premiers de n. Ainsi on a :

$$\begin{aligned}\varphi(n) &= \varphi\left(\prod_{i=1}^{r} p_i^{\alpha_i}\right)\\ &= \varphi\left(p_1^{\alpha_1}\right)...\varphi\left(p_r^{\alpha_r}\right)\\ &= \left(p_1^{\alpha_1} - p_1^{\alpha_1-1}\right) \times ... \times \left(p_r^{\alpha_r} - p_r^{\alpha_r-1}\right)\\ &= n\left(1-\frac{1}{p_1}\right)...\left(1-\frac{1}{p_r}\right)\end{aligned}$$

Exemple

- $\varphi(31) = 30$;
- $\varphi(81) = 2 \cdot 3^3 = 54$;
- $\varphi(60) = \varphi(2^2) \cdot \varphi(3) \cdot \varphi(5) = 2 \cdot 2 \cdot 4 = 16,$;
- $\varphi(1.000.000.000) = \varphi(2^9) \cdot \varphi(5^9) = 2^8 \cdot (4 \cdot 5^8) = 400.000.000.$

Théorème 2.14 Soit un entier $n \geq 2$. Pour tout $a \in \mathbb{Z}$ tel que $pgcd(a,n) = 1$ alors

$$a^{\varphi(n)} \equiv 1 \ (mod\ n),$$

où φ est la **fonction d'Euler**.

Corollaire 2.3 Soient p et q deux nombres premiers distincts et soit $\boldsymbol{n = pq}$.
Pour tout $a \in \mathbb{Z}$ tel que $\boldsymbol{pgcd}(a,n) = 1$ alors :

$$a^{(p-1)(q-1)} \equiv 1 \ \ (mod\ \ n)$$

L'hypothèse $pgcd(a,n) = 1$ équivaut ici à ce que a ne soit divisible ni par p, ni par q.
Par exemple pour $p = 5$, $q = 7$, $n = 35$ et $\varphi(n) = 4 * 6 = 24$.
Alors pour $a = 1,2,3,4,6,8,9,11,12,13,16,17,18,...$ on a bien $a^{24} \equiv 1$ $(mod\ 35)$.

Preuve

Notons $c = a^{(p-1)(q-1)}$. Calculons c modulo p :

$$c \equiv a^{(p-1)(q-1)} \equiv (a^{(p-1)})^{q-1} \equiv 1^{q-1} \equiv 1 \equiv (mod\ \ p)$$

où l'on appliquer le petit théorème de Fermat : $a^{p-1} \equiv 1 (mod\ \ p)$, car p ne divise pas a.

Calculons ce même c mais cette fois modulo q :

$$c \equiv a^{(p-1)(q-1)} \equiv (a^{(q-1)})^{p-1} \equiv 1^{p-1} \equiv 1 \equiv (mod\ \ q)$$

où l'on appliquer le petit théorème de Fermat : $a^{q-1} \equiv 1\ (mod\ q)$, car q ne divise pas a.
Conclusion partielle : $c \equiv 1\ \ (mod\ \ p)\ \ et\ \ c \equiv 1 (mod\ \ q)$.

Nous allons en déduire que $c \equiv 1\ \ (mod\ \ pq)$.

Comme $c \equiv 1\ \ (mod\ \ p)$ alors il existe $\alpha \in \mathbb{Z}$ tel que $c = 1 + \alpha p$;
Comme $c \equiv 1\ \ (mod\ \ q)$ alors il existe $\beta \in \mathbb{Z}$ tel que $c = 1 + \beta q$. Donc $c - 1 = \alpha p = \beta q$.
De l'égalité $\alpha p = \beta q$, on tire que $p|\beta q$.

Comme p et q sont premiers entre eux (car ce sont des nombres premiers distincts) alors par le lemme de Gauss on en déduit que $p|\beta$. Il existe donc $\beta' \in \mathbb{Z}$ tel que $\beta = \beta' p$.

Ainsi $c = 1 + \beta q = 1 + \beta' pq$. Ce qui fait que $c \equiv 1\ \ (mod\ \ pq)$, c'est exactement dire

$$a^{(p-1)(q-1)} \equiv 1\ \ (mod\ \ n)\ \blacksquare$$

2.18 Inverse modulo n

Définition 2.7 Soit $a \in \mathbb{Z}$.
On dit que $x \in \mathbb{Z}$ est un ***inverse de*** a ***modulo*** n si $ax \equiv 1\ (mod\ n)$.

Proposition 2.3
- a admet ***un inverse modulo*** n si et seulement si a et n sont ***premiers entre eux*** ;
- Si $au + nv = 1$ alors u est un ***inverse de*** a ***modulo*** n .

En d'autres termes, trouver un inverse de a ***modulo*** n ***revient à calculer les coefficients de Bézout associés à la paire*** (a,n)***.***

Preuve

La preuve est essentiellement une reformulation du théorème de Bézout :

$$\begin{aligned} pgcd(a,n) = 1 &\Leftrightarrow \exists\ \ u,v \in \mathbb{Z} \quad au + nv = 1 \\ &\Leftrightarrow \exists\ \ u \in \mathbb{Z} \quad au \equiv 1\ \ (mod\ \ n)\ \blacksquare \end{aligned}$$

2.19 L'exponentiation rapide

Nous aurons besoin de calculer rapidement des puissances modulo n.
Pour cela il existe une méthode beaucoup plus efficace que de calculer d'abord a^k
puis de le réduire modulo n. Il faut garder à l'esprit que les entiers que l'on va manipuler ont des dizaines voire des centaines de chiffres.
Voyons la technique sur l'exemple de $5^{11}\ (mod\ 14)$. L'idée est de seulement calculer $5, 5^2, 5^4, 5^8, \ldots$ et de réduire modulo n à chaque fois. Pour cela on remarque que
11 = 8 + 2 + 1 donc

$$5^{11} = 5^8 \times 5^2 \times 5^1\ \blacksquare$$

Calculons donc les 5^{2^i} $(mod\ 14)$:

$$\begin{aligned}
5 &\equiv 5 \ \ (mod\ \ 14)\\
5^2 &\equiv 25 \equiv 11 \ \ (mod\ \ 14)\\
5^4 &\equiv 5^2 \times 5^2 \equiv 11 \times 11 \equiv 121 \equiv 9 \ \ (mod\ \ 14)\\
5^8 &\equiv 5^4 \times 5^4 \equiv 9 \times 9 \equiv 81 \equiv 11 \ \ (mod\ \ 14)\ \blacksquare
\end{aligned}$$

à chaque étape est effectuée une multiplication modulaire. Conséquence :

$$\begin{aligned}
5^{11} &\equiv 5^4 \times 5^2 \times 5^1 \equiv 11 \times 11 \times 5\\
&\equiv 11 \times 55 \equiv 11 \times 13 \equiv 143 \equiv 3 \ \ (mod\ \ 14)\ \blacksquare
\end{aligned}$$

Nous obtenons donc un calcul de 5^{11} $(mod\ 14)$ en 5 opérations au lieu de 10 si on avait fait $5 \times 5 \times 5 \cdots$

II Algorithmes d'Euclide et de Bézout

3. Algorithme d'Euclide

Cette partie aborderas l'algorithme d'Euclide, un moyen efficace de trouver le plus grand commun diviseur de deux nombres puis illustreras son fonctionnement avec des exemples concrets, et enfin présenteras sa complexité.

3.1 Propositions

Proposition 3.1 Soit $a \geq b \geq 1$ deux entiers. Alors les diviseurs communs à a et b sont exactement les diviseurs communs à $a-b$ et b. En particulier,

$$\boldsymbol{pgcd}(a,b) = \boldsymbol{pgcd}(a-b,b).$$

Preuve

Supposons que k divise à la fois a et b.
Cela signifie que l'on peut trouver des entiers u et v tels que $a = ku$ et $b = kv$.
En particulier, $a-b = ku-kv = k(u-v)$, donc ce nombre est également divisible par k.
Ainsi, si un nombre divise à la fois a et b, il divise à la fois $a-b$ et b.

Réciproquement, si k divise à la fois b et $a-b$, on peut trouver des entiers v et w tels que $a = kv$ et $a-b = kw$.
On a alors $b = b+(a-b) = kv+kw = k(u+w)$, donc ce b est également divisible par k.
Ainsi, si un nombre divise à la fois $a-b$ et b, il divise à la fois a et b et on a démontré la première partie du résultat.

La deuxième partie s'en déduit immédiatement : on vient de voir que les diviseurs communs à a et b étaient exactement les mêmes que les diviseurs communs à $a-b$ et b.
En particulier, le plus grand de ces diviseurs communs doit être le même, c'est-à-dire

$$pgcd(a,b) = pgcd(a-b,b) \blacksquare$$

Exemples

Voici quelques exemples concrets d'utilisation de la proposition précédente :

1. **Exemple 1** : Calculer le $pgcd$ de 225 et 60
   *Le PGCD de 225 et 60 est **15**.*
   Déroulement

$$\begin{aligned} pgcd(225,60) &= pgcd(225-60,60) = pgcd(165,60) \\ &= pgcd(165-60,60) = pgcd(105,60) \\ &= pgcd(105-60,60) = pgcd(45,60) = pgcd(60,45) \\ &= pgcd(60-45,45) = pgcd(15,45) \\ &= 15, \end{aligned}$$

2. **Exemple 2** : Calculer le $pgcd$ de 168 et 300
   *Le PGCD de 168 et 300 est **12**.*
   Déroulement

$$\begin{aligned} pgcd(300,168) &= pgcd(168,132) \\ &= pgcd(132,36) \\ &= pgcd(96,36) \\ &= pgcd(60,36) \\ &= pgcd(36,24) \\ &= pgcd(24,12) \\ &= 12. \end{aligned}$$

Remarquons que l'on a arrêté le calcul à $pgcd(15, 45) = 15$ parce que l'on a noté que 15 était un diviseur de 45, mais que l'on aurait pu continuer à appliquer la méthode pour obtenir successivement $pgcd(45, 15) = pgcd(30, 15) = pgcd(15, 15) = 15$ sans devoir faire preuve d'aucune astuce.

Proposition 3.2 Soit a et b deux entiers, avec $a > b$.
Si on note q et r le quotient et le reste obtenus en posant la division de a par b (c'est-à-dire que $a = bq + r$, avec $r < b$), on a l'égalité

$$\textbf{\textit{pgcd}}(a, b) = \textbf{\textit{pgcd}}(b, r).$$

Preuve

- Soit d un diviseur de a et de b. Alors d divise $a - bq = r$;
- Soit d un diviseur de b et de r. Alors d divise aussi $bq + r = a$.

Les diviseurs de a et de b sont exactement les mêmes que les diviseurs de b et r. ∎

L'algorithme d'Euclide consiste alors à calculer le PGCD de deux nombres en appliquant cette proposition de façon répétée.

3.2 Présentation de l'algorithme d'Euclide

L'**algorithme d'Euclide** est un algorithme classique et fondamental en mathématiques, utilisé pour trouver le plus grand commun diviseur (PGCD) de deux entiers. Il porte le nom du mathématicien grec **Euclide**, qui l'a décrit pour la première fois dans son ouvrage "Les Éléments" vers 300 avant notre ère. Voici la description de l'algorithme d'Euclide :

1. Débutez avec deux entiers positifs a et b pour lesquels vous souhaitez trouver le PGCD ;
2. Effectuez la division de a par b et notez le reste r ;
3. Si $r = 0$, alors $b = pgcd(a,b)$. Dans ce cas, arrêtez l'algorithme ;
4. Si $r \neq 0$, remplacez a par b et b par r ;
5. Répétez les étapes 2 à 4 jusqu'à ce que le reste $r = 0$;
6. Lorsque r est égal à zéro, le PGCD de a et b est la valeur actuelle de b. ∎

L'**algorithme d'Euclide** est efficace pour calculer le PGCD de deux entiers, même pour de grands nombres, car il réduit rapidement les valeurs à considérer lors des divisions successives. Il est également utilisé comme base pour d'autres algorithmes plus avancés en mathématiques et en cryptographie.

3.3 Implémentation

Voici une implémentation simple de l'algorithme d'Euclide en Python :

```
1        def gcd(a, b):
2        while b != 0:
3        a, b = b, a % b
4        return a
```

Dans cette implémentation, la fonction gcd(a,b) prend en entrée deux entiers a et b. Cet algorithme itère en utilisant la division euclidienne jusqu'à ce que le reste soit nul, et le dernier quotient non nul est le PGCD des deux nombres. Finalement, il renvoie PGCD "a" des deux entiers.

3.4 Exemples concrets d'utilisation

Voici quelques exemples concrets d'utilisation de l'algorithme d'Euclide :

1. **Exemple 1** : Trouver le PGCD de 48 et 18 :
   Le PGCD de 48 et 18 est ***6***.
   Déroulement

$$48 = 2 * 18 + 12$$
$$18 = 12 * 1 + 6$$
$$12 = 6 * 2 + 0 \ ∎$$

   L'algorithme s'arrête lorsque le reste r est égal à zéro.

2. **Exemple 2** : Trouver le PGCD de 1071 et 462.
   *Le PGCD de 1071 et 462 est **21**.*
   Déroulement

$$1071 = 462 * 2 + 147$$
$$462 = 147 * 3 + 21$$
$$147 = 21 * 7 + 0 \blacksquare$$

   L'algorithme s'arrête lorsque le reste r est égal à zéro.

3.5 Complexité

La ***complexité*** de l'**algorithme d'Euclide** pour trouver le plus grand commun diviseur (PGCD) de deux nombres dépend principalement de la taille des nombres en entrée. On le désigne généralement comme un algorithme efficace et rapide pour des entiers de taille raisonnable.

La ***complexité*** de l'**algorithme d'Euclide** est généralement exprimée en ***termes d'itérations***. Supposons que les deux nombres en entrée soient a et b, où $a > b$. L'algorithme effectuera au plus **O(log b)** itérations. Cela signifie que ***la complexité de l'algorithme d'Euclide est logarithmique par rapport à la plus petite des deux valeurs***.

Dans le pire des cas, lorsque les nombres en entrée sont proches de valeurs extrêmement grandes, la complexité peut être légèrement supérieure. Cependant, l'algorithme d'Euclide reste très efficace et pratique pour calculer le PGCD de deux nombres. Sa simplicité et sa rapidité en font l'un des choix préférés pour de nombreuses applications mathématiques et informatiques.

4. Algorithme de Bézout

Cette section présentera l'algorithme de Bézout, utilisé pour trouver les coefficients de Bézout et le PGCD de deux entiers puis illustreras son fonctionnement avec des exemples concrets, et enfin présenteras sa complexité.

4.1 Théorème

Théorème 4.1 Soient a, b des entiers. Il existe m, $n \in \mathbb{Z}$ tels que

$$am + bn = \boldsymbol{pgcd}(a,b) \blacksquare$$

Preuve

Soit G l'ensemble formé par les entiers naturels strictement positifs de la forme $ma+nb$ où m et n sont des entiers relatifs.
G est une partie de $\mathbb{N}$ non vide : on vérifie facilement que $|a| \in G$.

G admet donc un plus petit élément d tel que $d = au + bv$;

- $D = pgcd(a,b)$ divise a et b donc D divise $au+bv = d$ et donc $\boldsymbol{D \leq d}$;
- Montrons que d divise a ;

  Divisons a par d, on a alors $a = dq + r$ avec $0 \leq r < d$.
  On isole le reste et on remplace d par $au+bv$:
  $r = a - dq = a - auq - bvq = a(1-uq) + b(-vq)$ donc $r = 0$. En effet si $r \neq 0$ alors $r \in G$, or $r < d$ et d est le plus petit élément de G, cela est absurde.
  $r = 0$ donc d divise a. En faisant le même raisonnement, on montrerait que d divise aussi b.
  d divise a et b donc $\boldsymbol{d \leq D}$;

- ***Conclusion*** : $\boldsymbol{d \leq D}$ et $\boldsymbol{D \leq d}$ donc $\boldsymbol{D = d}$. $\blacksquare$

Corollaire 4.1 Soit k un entier divisant à la fois a et b. Alors k divise $\boldsymbol{pgcd}(a,b)$.

4.2 Présentation de l'algorithme de Bézout

L'***algorithme de Bézout*** est un algorithme utilisé pour calculer le plus grand diviseur commun (*PGCD*) de deux entiers et pour trouver les coefficients de Bézout qui satisfont l'***identité de Bézout***.
L'***identité de Bézout***, stipule que pour deux entiers a et b, il existe des entiers x et y tels que :

$$ax + by = PGCD(a,b) \quad \blacksquare$$

Voici les étapes de l'algorithme de Bézout :
Étant donnés deux entiers positifs a et b, l'algorithme de Bézout procède comme suit :

a. **Étape 1** : Initialisation des variables.
   - Initialisez les variables a_0, b_0, a_1, b_1, a_2, b_2 avec les valeurs a, b, 1, 0, 0, 1 respectivement.

b. **Étape 2** : Boucle de calcul.
   - Tant que b n'est pas égal à zéro, continuez les étapes suivantes :
   - $q = a$ div b (division entière de a par b) ;
   - $(a, b) = (b, a \ mod \ b)$ (a prend la valeur de b, b prend la valeur de a modulo b) ;
   - $(a_2, b_2) = (a_0 - q * a_1, b_0 - q * b_1)$;
   - $(a_0, b_0) = (a_1, b_1)$;
   - $(a_1, b_1) = (a_2, b_2)$ ■

   À la fin de cette boucle, b deviendra égal à zéro, et a sera égal au PGCD des entiers d'origine a et b.

c. **Étape 3** : Résultat.
   - Les coefficients de Bézout seront les valeurs finales de a_0 et b_0. Ils fournissent une combinaison linéaire du PGCD de a et b : PGCD$(a,b) = a_0 * a + b_0 * b$.

4.3 Implémentation

Voici une implémentation simple de l'algorithme de Bézout en Python :

```
1          def bezout(a, b):
2
3          if b == 0:
4          return 1, 0, a
5
6          x_prev, x = 1, 0
7          y_prev, y = 0, 1
8
9          while b != 0:
10         q = a // b
11         a, b = b, a % b
12         x, x_prev = x_prev - q * x, x
13         y, y_prev = y_prev - q * y, y
14
15         return x_prev, y_prev, a
```

Dans cette implémentation, la fonction bezout(a,b) prend en entrée deux entiers a et b. Elle utilise l'algorithme de Bézout pour trouver les coefficients de Bézout x et y tels que $ax+by = gcd(a,b)$, où $gcd(a,b)$ représente le PGCD de a et b.
L'algorithme fonctionne en itérant tant que b n'est pas égal à zéro. À chaque itération, il met à jour les valeurs de a, b, x et y en utilisant les formules de l'algorithme de Bézout. Finalement, il renvoie les coefficients de Bézout x et y, ainsi que le PGCD "a" des deux entiers.

4.4 Exemples concrets d'utilisation

Voici quelques exemples concrets d'utilisation de l'algorithme de Bézout :

a. **Exemple 1** : Trouver les coefficients de Bézout et le PGCD de 35 et 21.
*Les coefficients de Bézout sont **2** et **-1**, et le PGCD de 35 et 21 est **7**.*
Déroulement
- *Appliquer l'algorithme d'Euclide pour trouver le PGCD.*

  $35 = 21*1+14$

  $21 = 14*1+7$

  $14 = 7*2+0$ ∎

- *Remonter les équations en utilisant le quotient précédent pour exprimer le reste.*

  $7 = 21-14*1$

  $14 = 35-21*1$ ∎

- *Simplifier les équations pour obtenir les coefficients de Bézout.*

  $7 = 21-(35-21*1)*1$

  $= 21*2-35$ ∎

b. **Exemple 2** : Trouver les coefficients de Bézout et le PGCD de 48 et 18.
*Les coefficients de Bézout sont **3** et **-1**, et le PGCD de 48 et 18 est **6**.*
Déroulement
- *Appliquer l'algorithme d'Euclide pour trouver le PGCD* :

  $48 = 18*2+12$

  $18 = 12*1+6$

  $12 = 6*2+0$ ∎

- *Remonter les équations en utilisant le quotient précédent pour exprimer le reste* :

  $6 = 18 - 12 * 1$

  $12 = 48 - 18 * 2$ ∎

- *Simplifier les équations pour obtenir les coefficients de Bézout* :

  $6 = 18 - (48 - 18 * 2) * 1$

  $= 18 * 3 - 48$ ∎

4.5 Complexité

La ***complexité*** de l'**algorithme de Bézout** dépend de la taille des entrées, c'est-à-dire des nombres entiers donnés.

Pour deux entiers a et b, la ***complexité*** de l'**algorithme de Bézout** est généralement exprimée en termes du nombre de divisions euclidiennes effectuées pour obtenir le PGCD. Dans le pire des cas, elle est proportionnelle au logarithme du plus petit des deux entiers donnés, c'est-à-dire $\min(log(a), log(b))$.
Cela signifie que l'algorithme a ***une complexité logarithmique***.

Plus précisément, la ***complexité*** de l'**algorithme de Bézout** est **O(log(min(a, b)))**. Cela implique que le temps d'exécution de l'algorithme augmente logarithmiquement avec la taille des nombres a et b.

Il convient de noter que la complexité de l'algorithme de Bézout ne prend pas en compte les opérations arithmétiques de base telles que les multiplications et les additions, car celles-ci sont généralement considérées comme étant de complexité constante par rapport à la taille des entiers donnés.

En résumé, l'**algorithme de Bézout** a une ***complexité logarithmique*** et est considéré comme efficace pour calculer le PGCD et les coefficients de Bézout de deux entiers. Cependant, il est important de prendre en compte d'autres facteurs, tels que la taille des entiers et les opérations arithmétiques utilisées, lors de l'évaluation de la performance globale de l'algorithme.

III Initiation Aux Chiffrements

5. Introduction

Cette partie introduira les notions de base en cryptographie puis en chiffrement et présentera également quelques exemples concrets de chiffrements tels que le chiffrement affine et le chiffrement RSA tout en expliquant leurs fonctionnement et en mettant l'accent sur la manière dont les algorithmes d'Euclide et de Bézout sont utilisés pour garantir la sécurité des données.

5.1 Introduction aux notions de base en cryptographie

La ***cryptographie*** est l'étude des techniques et des méthodes utilisées pour sécuriser les communications, protéger les informations sensibles et garantir la confidentialité, l'authenticité et l'intégrité des données. Les notions de base en cryptographie sont essentielles pour comprendre les mécanismes utilisés dans les systèmes de sécurité modernes. Voici une introduction à quelques-unes de ces notions :

a. **Chiffrement** : Le chiffrement est le processus de conversion d'un message en clair (texte original) en un message chiffré (texte codé), rendant ainsi le message illisible pour les personnes non autorisées. Le chiffrement repose sur l'utilisation d'un
algorithme et d'une clé. Il existe deux types de chiffrement couramment utilisés : le *chiffrement symétrique* et le *chiffrement asymétrique*.

b. **Chiffrement symétrique** : Le chiffrement symétrique, également appelé chiffrement à clé secrète, utilise une seule et même clé pour chiffrer et déchiffrer les données. C'est un processus plus rapide que le chiffrement asymétrique car les opérations de chiffrement et de déchiffrement sont simples et similaires.

- ***Clé*** : Dans le chiffrement symétrique, la même clé est utilisée pour chiffrer et déchiffrer les données. Cette clé doit rester confidentielle, car si elle est compromise, les données chiffrées peuvent être facilement déchiffrées.

- ***Opérations de chiffrement et de déchiffrement*** : Pour chiffrer un message, la clé est utilisée dans un algorithme de chiffrement pour transformer le texte en clair en un texte chiffré. Pour déchiffrer le message, la même clé est utilisée dans un algorithme de déchiffrement pour retrouver le texte en clair original à partir du texte chiffré.

Exemple d'algorithme de chiffrement symétrique populaire :

AES (Advanced Encryption Standard).

c. **Chiffrement asymétrique** : Le chiffrement asymétrique, également connu sous le nom de chiffrement à clé publique, utilise une paire de clés distinctes : une *clé publique* et une *clé privée*. Les clés sont mathématiquement liées, mais il est pratiquement impossible de déduire la clé privée à partir de la clé publique. Cette approche permet des fonctionnalités supplémentaires telles que la confidentialité, l'authentification et la non-répudiation.

   - ***Clés publiques et privées*** : Dans le chiffrement asymétrique, chaque utilisateur possède une paire de clés : une clé publique qu'il partage avec les autres et une clé privée qu'il garde secrète.

   - ***Opérations de chiffrement et de déchiffrement*** : Pour chiffrer un message, la clé publique du destinataire est utilisée. Une fois chiffré, seul le destinataire possédant la clé privée correspondante peut déchiffrer le message.

   Exemple d'algorithme de chiffrement asymétrique populaire :
   RSA (Rivest-Shamir-Adleman).

d. **Hachage** : Le hachage est une fonction qui transforme une quantité de données en une empreinte numérique unique, appelée "hash" ou "condensat". Les fonctions de hachage sont utilisées pour vérifier l'intégrité des données et garantir qu'elles n'ont pas été modifiées. Un bon algorithme de hachage doit être résistant à la collision, c'est-à-dire qu'il doit être difficile de trouver deux messages différents ayant le même condensat. MD5 et SHA-256 sont des exemples d'algorithmes de hachage couramment utilisés.

e. **Signature numérique** : Une signature numérique est un mécanisme permettant de garantir l'authenticité, l'intégrité et la non-répudiation d'un message. Elle utilise une combinaison de chiffrement asymétrique et de hachage. L'émetteur signe le message en utilisant sa clé privée, et le destinataire peut vérifier l'authenticité de la signature à l'aide de la clé publique de l'émetteur. Si la signature est valide, cela prouve que le message n'a pas été modifié et que l'émetteur est bien celui qu'il prétend être.

Ces notions de base en cryptographie sont utilisées dans de nombreux domaines, tels que les communications sécurisées sur Internet, le stockage de données sensibles et les protocoles de sécurité des systèmes d'exploitation. La cryptographie moderne continue d'évoluer pour faire face aux défis de sécurité croissants et pour protéger nos informations dans un monde numérique en constante évolution.

5.2 Introduction aux notions de base en chiffrement

Le ***chiffrement*** est un processus utilisé pour sécuriser les informations en les rendant inintelligibles à toute personne non autorisée. Il s'agit d'une technique largement utilisée dans le domaine de la sécurité de l'information et de la confidentialité des données.
Voici une introduction aux notions de base du chiffrement :

a. **Message en clair** : Il s'agit du texte ou des données d'origine que l'on souhaite protéger. Le message en clair est généralement compréhensible par les utilisateurs autorisés.

b. **Algorithme de chiffrement** : Il s'agit d'une série d'opérations mathématiques ou logiques qui transforment le message en clair en une forme illisible, appelée message chiffré ou cryptogramme. Les algorithmes de chiffrement peuvent être symétriques (utilisant une seule clé pour chiffrer et déchiffrer) ou asymétriques (utilisant une paire de clés, une pour chiffrer et une autre pour déchiffrer).

c. **Clé de chiffrement** : Une clé est une valeur secrète utilisée par l'algorithme de chiffrement pour transformer le message en clair en message chiffré. La clé peut être une séquence de bits ou une chaîne de caractères. Dans le chiffrement symétrique, la même clé est utilisée pour chiffrer et déchiffrer les données. Dans le chiffrement asymétrique, une clé est utilisée pour le chiffrement et une autre clé correspondante est utilisée pour le déchiffrement.

d. **Message chiffré** : Il s'agit du résultat du chiffrement du message en clair à l'aide de l'algorithme de chiffrement et de la clé de chiffrement. Le message chiffré est généralement illisible et ne peut pas être compris sans utiliser la clé de déchiffrement appropriée.

e. **Déchiffrement** : Il s'agit du processus inverse du chiffrement, où le message chiffré est converti en message en clair à l'aide de l'algorithme de chiffrement et de la clé de déchiffrement appropriée.

Le chiffrement joue un rôle essentiel dans de nombreux domaines, tels que la sécurité des communications, le stockage sécurisé des données, la protection des informations sensibles et la confidentialité des transactions en ligne. En utilisant des algorithmes de chiffrement robustes et des pratiques de gestion des clés adéquates, il est possible de protéger efficacement les informations confidentielles contre les accès non autorisés.

6. Chiffrements

6.1 Chiffrement affine

Le chiffrement affine est une méthode de chiffrement classique qui effectue une transformation linéaire sur chaque lettre d'un message pour le chiffrer. Il utilise une fonction mathématique affine de la forme :

$$C = (a * L + b) \mod m$$

où *C est le* ***caractère chiffré****, L est le* ***caractère d'origine (non chiffré)****, a et b sont des* ***coefficients de la fonction affine****, et m est la* ***taille de l'alphabet utilisé***. "***mod***" signifie que le résultat est obtenu en prenant le reste de la division par m .

Pour chiffrer un message à l'aide du chiffrement affine, ***chaque lettre du message est remplacée par une autre lettre*** selon la formule ci-dessus. Le coefficient a est ***choisi de manière à être premier avec*** m (c'est-à-dire que a et m n'ont pas de facteur commun autre que 1) afin d'assurer l'unicité du déchiffrement.

Le coefficient b est ***un décalage ajouté après la multiplication par*** a, ce qui introduit une composante de substitution simple dans le chiffrement.
Le résultat est ensuite réduit modulo m pour obtenir une lettre chiffrée dans l'alphabet.

Pour déchiffrer un message chiffré avec le chiffrement affine, on utilise la formule inverse :

$$L = (a^{-1} * (C - b)) \mod m$$

où a^{-1} est l'inverse multiplicatif de a modulo m, c'est-à-dire un nombre x tel que $(a * x) \mod m = 1$.
Il convient de noter que le chiffrement affine est relativement faible du point de vue de la sécurité, car il est vulnérable à des attaques par force brute et à des attaques statistiques, notamment lorsque la taille de l'alphabet utilisé est petite.

Fonctionnement

6.1.1 Clé de Chiffrement

La clé est (a,b) tel que :

- a,$b \in [0..25]$;
- PGCD(a,26) = 1 ∎

Comment choisir les nombres a et b ?

- Chaque entier $n \in [0..25]$ soit associé une unique image $n' = a*n+b \ mod\ 26$;
- Cela ne sera pas le cas pour tous les couples (a,b) ∎

Exemple

- Si (a,b) = (4,1) ;
- Pour $n = 0$: $n' = 4*0+1 \ mod\ 26$ = 1 ;
- Pour $n = 13$: $n' = 4*13+1 \ mod\ 26$ = 1 ;
- 0 et 13 ont la même image par la transformation affine ∎

Remarque

- Les valeurs de n' seront toutes différentes si et seulement si a est premier avec 26 ;
- b peut être librement choisi ;
- Cela laisse ainsi un choix pour a un des nombres : $1,3,5,7,9,11,15,17,19,21,23,25$. Dans ce cas le nombre de clés possible est $12*26 = 312$ ∎

6.1.2 Chiffrement

$$C = (a*L+b) \ mod\ 26$$

- L : représente la position de la lettre du texte clair,
- C : le résultat du chiffrement de la lettre L,
- (a,b) : représente la clé de chiffrement ; ∎

Exemple à la main

Le message à chiffrer est "**HELLO**" avec les clés a = 5 et b = 8.
Le pgcd(5,26) = 1, la clé est donc valide pour le chiffrement Affine.
Table de correspondance : A=0, B=1, C=2, ..., Z=25.

- **Message à chiffrer** : "HELLO"
- **Le message chiffré** : "LCTTK".
- **Déroulement** :
  - H (7) devient : (5 * 7 + 8) mod 26 = 11 (L)
  - E (4) devient : (5 * 4 + 8) mod 26 = 2 (C)
  - L (11) devient : (5 * 11 + 8) mod 26 = 19 (T)
  - L (11) devient : (5 * 11 + 8) mod 26 = 19 (T)
  - O (14) devient : (5 * 14 + 8) mod 26 = 10 (K)

Le message chiffré est bien "**LCTTK**".

6.1.3 Déchiffrement

$$L = a^{-1} * (C - b) \ mod \ 26$$

- C : le résultat du chiffrement de la lettre L,
- L : représente la position de la lettre du texte clair,
- a^{-1} : l'inverse modulaire de a dans [0..25], tel que :
  $a^{-1} * a \ mod \ 26 = 1$ ∎

Exemple à la main

Le message à déchiffrer est "**LCTTK**" avec les clés a = 5 et b = 8.
a^{-1} pour a=5 est 21 (car 5 * 21 mod 26 = 1) ∎

- **Message à déchiffrer** : "LCTTK"
- **Le message Déchiffré** : "HELLO".
- **Déroulement** :
  — L (11) devient : (21 * (11 - 8)) mod 26 = 7 (H)
  — C (2) devient : (21 * (2 - 8)) mod 26 = 4 (E)
  — T (19) devient : (21 * (19 - 8)) mod 26 = 11 (L)
  — T (19) devient : (21 * (19 - 8)) mod 26 = 11 (L)
  — K (10) devient : (21 * (10 - 8)) mod 26 = 14 (O)

  Le message déchiffré est bien "**HELLO**".

6.1.4 Implémentation

Voici une implémentation simple du chiffrement affine en Python :

```
        def gcd(a, b):
        while b != 0:
        a, b = b, a % b
        return a

        def mod_inverse(a, m):
        for x in range(1, m):
        if (a * x) % m == 1:
        return x
        return None

        def affine_encrypt(text, a, b):
        encrypted_text = ""
        for char in text:
        if char.isalpha():
        if char.isupper():
        encrypted_char = chr((a * (ord(char) - 65) + b) %
            ↪ 26 + 65)
        else:
        encrypted_char = chr((a * (ord(char) - 97) + b) %
            ↪ 26 + 97)
        encrypted_text += encrypted_char
        else:
        encrypted_text += char
        return encrypted_text

        def affine_decrypt(encrypted_text, a, b):
        decrypted_text = ""
        a_inverse = mod_inverse(a, 26)
        for char in encrypted_text:
        if char.isalpha():
        if char.isupper():
        decrypted_char = chr((a_inverse * (ord(char) - 65
            ↪ - b)) % 26 + 65)
        else:
        decrypted_char = chr((a_inverse * (ord(char) - 97
            ↪ - b)) % 26 + 97)
        decrypted_text += decrypted_char
        else:
        decrypted_text += char
        return decrypted_text

        # Exemple d'utilisation :
        message = "Messieurs, la rencontre aura lieu au
            ↪ Sacre Coeur, au lever du jour. Soyez
            ↪ ponctuels."
        a = 5
        b = 8
```

```
43
44          print("Message clair:", message)
45
46          encrypted_message = affine_encrypt(message, a, b)
47          print("Message chiffre:", encrypted_message)
48
49          decrypted_message = affine_decrypt(
               ↪ encrypted_message, a, b)
50          print("Message dechiffre:", decrypted_message)
```

Cette implémentation utilise un alphabet de 26 lettres (de 'a' à 'z' en minuscules) et ne chiffre que les lettres alphabétiques, en conservant les autres caractères tels quels. Il est possible d'ajuster l'alphabet et la logique de chiffrement en fonction de vos besoins. Il faudra s'assurer d'utiliser des valeurs appropriées pour les coefficients a et b afin d'obtenir un chiffrement valide.

6.1.5 Test avec un ordinateur

Clé : (a, b) = (5, 8)
Message clair: Messieurs, la rencontre aura lieu au Centre Pompidou, au lever du jour. Soyez ponctuels.
Message chiffré: Qcuuwcepu, li pcvsavzpc iepi lwce ie Scvzpc Faqfwxae, ie lcjcp xe baep. Uaycd favszeclu.
Message déchiffré: Messieurs, la rencontre aura lieu au Centre Pompidou, au lever du jour. Soyez ponctuels.

- **Clé** : (a, b) = (5, 8)
- **Message clair** : Message clair : Messieurs, la rencontre aura lieu au Centre Pompidou, au lever du jour. Soyez ponctuels.
- **Message chiffré** : Qcuuwcepu, li pcvsavzpc iepi lwce ie Scvzpc Faqfwxae, ie lcjcp xe baep. Uaycd favszeclu.
- **Message déchiffré** : Messieurs, la rencontre aura lieu au Centre Pompidou, au lever du jour. Soyez ponctuels.

6.2 Chiffrement RSA

Le ***chiffrement RSA*** est l'un des algorithmes de chiffrement asymétrique les plus utilisés en cryptographie moderne. Il tire son nom des initiales de ses inventeurs : **Rivest**, **Shamir** et **Adleman**. Le ***chiffrement RSA*** est basé sur le concept mathématique de la factorisation des nombres premiers.

Voici un aperçu du fonctionnement du chiffrement RSA :

a. Génération des clés :
- Choix de deux nombres premiers distincts p et q ;
- Calcul du produit $n = p \times q$, qui représente le module de chiffrement ;
- Calcul de la fonction d'Euler $\varphi(n) = (p-1) \times (q-1)$, qui donne le nombre d'entiers positifs inférieurs à n et premiers avec n ;
- Choix d'un entier e, tel que $1 < e < \varphi(n)$, et qui est premier avec $\varphi(n)$. e est la **clé publique d'encryption** ;
- Calcul de l'entier d, tel que $d * e \equiv 1 \pmod{\varphi(n)}$, où d est la **clé privée de déchiffrement**.

b. **Chiffrement** :
- Le message à chiffrer est divisé en blocs de taille appropriée ;
- Chaque bloc du message est converti en un nombre entier représentant le texte en clair ;
- Pour chiffrer chaque bloc, la formule de chiffrement est utilisée : $x \equiv m^e \ (mod \ \ n)$, où x est **le texte chiffré** et m est le **texte en clair**.

c. **Déchiffrement** :
- Pour déchiffrer le texte chiffré x, la formule de déchiffrement est utilisée : $m \equiv x^d (mod \ \ n)$, où m est le texte en clair ;
- Le texte en clair m est converti en blocs de données pour reconstituer le message original.

Fonctionnement et Exemple

Pour crypter un message on commence par le transformer en un ou plusieurs nombres. Les processus de chiffrement et déchiffrement font appel à plusieurs notions :

- On choisit deux nombres premiers p et q que l'on garde secrets et on pose $n = p \times q$. Le principe étant que même connaissant n il est très difficile de retrouver p et q (qui sont des nombres ayant des centaines de chiffres) ;

- La **clé secrète** et la **clé publique** se calculent à l'aide de l'**algorithme d'Euclide** et des **coefficients de Bézout** ;

- Les calculs de cryptage se feront **modulo** n ;

- Le déchiffrement fonctionne grâce à une **variante du petit théorème de Fermat**.

Dans cette section, ***c'est Bruno qui veut envoyer un message secret à Alice***. La processus se décompose ainsi :

a. Alice prépare une clé publique et une clé privée,
b. Bruno utilise la clé publique d'Alice pour crypter son message,
c. Alice reçoit le message crypté et le déchiffre grâce à sa clé privée.

6.2.1 Calcul de la clé publique et de la clé privée

6.2.1.1 Choix de deux nombres premiers

Alice effectue, une fois pour toute, les opérations suivantes (en secret) :
- Elle choisit deux nombres premiers distincts p et q (dans la pratique ce sont de très grand nombres, jusqu'à des centaines de chiffres),
- Elle calcule $n = p \times q$,
- Elle calcule $\varphi(n) = (p-1) \times (q-1)$.

Exemple

- $p = 5$ et $q = 17$;
- $n = p \times q = 85$;
- $\varphi(n) = (p-1) \times (q-1) = 64$ ∎

Vous noterez que le calcul de $\varphi(n)$ n'est possible que si la décomposition de n sous la forme $p \times q$ est connue. D'où le caractère secret de $\varphi(n)$ même si n est connu de tous.

6.2.1.2 Choix d'un exposant et calcul de son inverse

Alice continue :
- elle choisit un exposant e tel que $pgcd(e, \varphi(n)) = 1$,
- elle calcule l'inverse d de e modulo $\varphi(n)$: $d \times e \equiv 1 \ (mod\ \varphi(n))$. Ce calcul se fait par l'algorithme d'Euclide étendu.

Exemple

- Alice choisit par exemple $e = 5$ et on a bien $pgcd(e, \varphi(n)) = pgcd(5, 64) = 1$;
- Alice applique l'algorithme d'Euclide étendu pour calculer les coefficients de Bézout correspondant à $pgcd(e, \varphi(n)) = 1$. Elle trouve $5 \times 13 + 64 \times (-1) = 1$. Donc $5 \times 13 \equiv 1 \ (mod\ 64)$ et l'inverse de e modulo $\varphi(n)$ est $d = 13$ ∎

6.2.1.3 Clé publique

La clé publique d'Alice est constituée des deux nombres : n *et* e.
Et comme son nom l'indique Alice communique sa clé publique au monde entier.

Exemple

$n = 85$ et $e = 5$ ∎

6.2.1.4 Clé privée

Alice garde pour elle sa clé privée : d.
Alice détruit en secret p, q et $\varphi(n)$ qui ne sont plus utiles. Elle conserve secrètement sa clé privée.

Exemple

$d = 13$ ∎

6.2.2 Chiffrement du message

Bruno veut envoyer un message secret à Alice. Il se débrouille pour que son message soit un entier (quitte à découper son texte en bloc et à transformer chaque bloc en un entier).

6.2.2.1 Message

Le message est un entier m, tel que $0 \leq m < n$.

Exemple

Bruno veut envoyer le message $m = 10$.

6.2.2.2 Message chiffré

Bruno récupère la clé publique d'Alice : n et e avec laquelle il calcule, à l'aide de l'algorithme d'exponentiation rapide, le message chiffré :

$$x \equiv m^e \pmod{n}$$

Il transmet ce message x à Alice.

Exemple

$m = 10$, $n = 85$ et $e = 5$ donc

$$x \equiv m^e \pmod{n} \equiv 10^5 \pmod{85}$$ ∎

On peut ici faire les calculs à la main :

$$10^2 \equiv 100 \equiv 15 (mod\ \ 85)$$
$$10^4 \equiv (10^2)^2 \equiv 15^2 \equiv 225 \equiv 55 (mod\ \ 85)$$
$$x \equiv 10^5 \equiv 10^4 \times 10 \equiv 55 \times 10 \equiv 550 \equiv 40 (mod\ \ 85)$$

Le message chiffré est donc $x = 40$.

6.2.3 Déchiffrement du message

Alice reçoit le message x chiffré par Bruno, elle le décrypte à l'aide de sa clé privée d, par l'opération :

$$m \equiv x^d (mod \ \ n)$$

qui utilise également l'algorithme d'exponentiation rapide.

Nous allons prouver dans le lemme 1, que par cette opération Alice retrouve bien le message original m de Bruno.

Exemple

$c = 40$, $d = 13$, $n = 85$ donc $x^d \equiv (40)^{13}$ $(mod\ 85)$.
Calculons à la main $40^{13} \equiv (mod\ 85)$ on note que 13 = 8 + 4 + 1, donc $40^{13} = 40^8 \times 40^4 \times 40$.

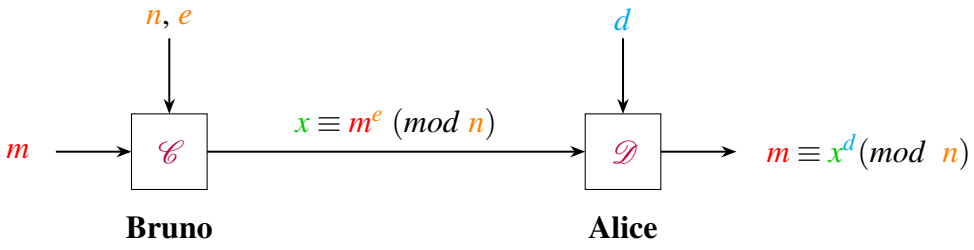
$$40^2 \equiv 1600 \equiv 70\ (mod\ 85)$$
$$40^4 \equiv (40^2)^2 \equiv 70^2 \equiv 4900 \equiv 55\ (mod\ 85)$$
$$40^8 \equiv (40^4)^2 \equiv 55^2 \equiv 3025 \equiv 50\ (mod\ 85)$$

Donc

$$x^d \equiv 40^{13} \equiv 40^8 \times 40^4 \times 40 \equiv 50 \times 55 \times 40 \equiv 10 (mod \ \ 85)$$

qui est bien le message m de Bruno.

6.2.4 Schéma



Clés d'Alice :

- publique : n, e
- privée : d

6.2.5 Lemme de déchiffrement

Le principe de déchiffrement repose sur le petit théorème de Fermat généralisé.

6.2.5.1 Lemme 1

Soit d l'inverse de e modulo $\varphi(n)$.

$$\text{Si } x \equiv m^e \pmod{n} \text{ alors } m \equiv x^d \pmod{n} \blacksquare$$

Ce lemme prouve bien que le message original m de Bruno, chiffré par clé publique d'Alice (e, n) en le message x , peut-être retrouvé par Alice à l'aide de sa clé secrète d.

Preuve

- Que d soit l'inverse de e modulo $\varphi(n)$ signifie $d*e \equiv 1 \pmod{\varphi(n)}$.
  Autrement dit, il existe $k \in \mathbb{Z}$ tel que $d*e = 1+k*\varphi(n)$.
- On rappelle que par le petit théorème de Fermat généralisé : lorsque m et n sont premiers entre eux

$$m^{\varphi(n)} \equiv m^{(p-1)(q-1)} \equiv 1 \pmod{n} \blacksquare$$

- **Premier cas** $pgcd(m,n) = 1$. Notons $c \equiv m^e \pmod{\text{n}}$ et calculons x^d :

$$x^d \equiv (m^e)^d \equiv m^{ed} \equiv m^{1+k\varphi(n)} \equiv m*m^{k\varphi(n)} \equiv m*(m^{\varphi(n)})^k \equiv m*(1)^k \equiv m \pmod{n} \blacksquare$$

- **Deuxième cas** $pgcd(m,n) \neq 1$.
  Comme n est le produit des deux nombres premiers p et q et que m est strictement plus petit que n alors si m et n ne sont pas premiers entre eux cela implique que p divise m ou bien q divise m (mais pas les deux en même temps).

  Faisons l'hypothèse $pgcd(m,n) = p$ et $pgcd(m,q) = 1$(le cas $pgcd(m,n) = q$ et $pgcd(m,p) = 1$ se traiterait de la même manière).

  Étudions $(m^e)^d$ à la fois modulo p et modulo q à l'image de ce que nous avons fait dans la preuve du théorème de Fermat généralisé.
  — modulo p : $m \equiv 0 \pmod{p}$ et $(m^e)^d \equiv 0 \pmod{\text{p}}$ donc $(m^e)^d \equiv m \pmod{p}$,
  — modulo q : $(m^e)^d \equiv m \times (m^{\varphi(n)})^k \equiv m \times (m^{q-1})^{(p-1)k} \equiv m \pmod{q}$.
  Comme p et q sont deux nombres premiers distincts, ils sont premiers entre eux et on peut écrire comme dans la preuve du petit théorème de Fermat généralisé que

$$(m^e)^d \equiv m \pmod{n} \blacksquare$$

6.2.6 Implémentation

Voici une implémentation simple du chiffrement et du déchiffrement RSA en Python :

```
import random

def gcd(a, b):
while b != 0:
a, b = b, a % b
return a

def mod_inverse(a, m):
for x in range(1, m):
if (a * x) % m == 1:
return x
return None

def is_prime(num):
if num <= 1:
return False
for i in range(2, int(num**0.5) + 1):
if num % i == 0:
return False
return True

def generate_prime(bits):
while True:
num = random.getrandbits(bits)
if is_prime(num):
return num

def generate_keypair(bits):
p = generate_prime(bits)
q = generate_prime(bits)
n = p * q
phi_n = (p - 1) * (q - 1)

e = 2
while gcd(e, phi_n) != 1:
e += 1

d = mod_inverse(e, phi_n)
return ((e, n), (d, n))

def rsa_encrypt(message, public_key):
e, n = public_key
encrypted_message = [pow(ord(char), e, n) for char
    ↪  in message]
return encrypted_message

```

```
46          def rsa_decrypt(encrypted_message, private_key):
47          d, n = private_key
48          decrypted_message = [chr(pow(char, d, n)) for char
               ↪  in encrypted_message]
49          return ''.join(decrypted_message)
50
51          # Exemple d'utilisation :
52          message = "Jonathan, Le soleil se l ve   06h03,
               ↪ arrive   l'heure dans notre int r t
               ↪ tous."
53          bits = 10
54
55          public_key, private_key = generate_keypair(bits)
56          print("Cl  publique:", public_key)
57          print("Cl  privee:", private_key)
58
59          print("Message clair:", message)
60
61          encrypted_message = rsa_encrypt(message,
               ↪ public_key)
62          print("Message chiffre:", encrypted_message)
63
64          decrypted_message = rsa_decrypt(encrypted_message,
               ↪  private_key)
65          print("Message d chiffre:", decrypted_message)
```

6.2.7 Test avec un ordinateur

```
Clé publique: (5, 73357)
Clé privée: (58061, 73357)
Message clair: Jonathan, Le soleil se lève à 06h03, arrive à l'heure dans notre intèrêt à tous.
Message chiffré: [30731, 38509, 10792, 23123, 60407, 50503, 23123, 10792, 9688, 30283, 14028, 23040, 30283, 36116, 38509, 2038
2, 23040, 18051, 20382, 30283, 36116, 23040, 30283, 20382, 25742, 23406, 23040, 30283, 15515, 30283, 35107, 23561, 50503, 3510
7, 27280, 9688, 30283, 23123, 60677, 60677, 18051, 23406, 23040, 30283, 15515, 30283, 20382, 68446, 50503, 23040, 53696, 60677,
23040, 30283, 47117, 23123, 10792, 36116, 30283, 10792, 38509, 60407, 60677, 23040, 30283, 18051, 10792, 60407, 25742, 60677, 3
1061, 60407, 30283, 15515, 30283, 60407, 38509, 53696, 36116, 49877]
Message déchiffré: Jonathan, Le soleil se lève à 06h03, arrive à l'heure dans notre intèrêt à tous.
```

- **Clé publique** : (5, 73357)
- **Clé privée** : (58061, 73357)
- **Message clair** : Jonathan, Le soleil se lève à 06h03, arrive à l'heure dans notre intèrêt à tous.
- **Message chiffré** : [30731, 38509, 10792, 23123, 60407, 50503, 23123, 10792, 9688, 30283, 14028, 23040, 30283, 36116, 38509, 20382, 23040, 18051, 20382, 30283, 36116, 23040, 30283, 20382, 25742, 23406, 23040, 30283, 15515, 30283, 35107, 23561, 50503, 35107, 27280, 9688, 30283, 23123, 60677, 60677, 18051, 23406, 23040, 30283, 15515, 30283, 20382, 68446, 50503, 23040, 53696, 60677, 23040, 30283, 47117, 23123, 10792, 36116, 30283, 10792, 38509, 60407, 60677, 23040, 30283, 18051, 10792, 60407, 25742, 60677, 31061, 60407, 30283, 15515, 30283, 60407, 38509, 53696, 36116, 49877]
- **Message déchiffré** : Jonathan, Le soleil se lève à 06h03, arrive à l'heure dans notre intérêt à tous.

6.3 Applications

6.3.1 Applications du Chiffrement affine

Voici quelques domaines de la vie où le chiffrement affine peut être appliqué :

a. **Éducation et jeux éducatifs** : Le chiffrement affine peut être utilisé dans les programmes éducatifs pour enseigner les concepts de cryptographie et de mathématiques. Les étudiants peuvent créer des messages secrets pour que leurs pairs les déchiffrent à l'aide de clés appropriées.

b. **Jeux de société** : Il est souvent utilisé dans les jeux de société pour ajouter un élément de cryptographie et de mystère. Les énigmes cryptées peuvent être résolues en utilisant cette méthode, ce qui ajoute un défi supplémentaire au jeu.

c. **Loisirs et divertissement** : Il est parfois utilisé dans des puzzles, des magazines ou des livres de loisirs pour cacher des messages ou des réponses aux énigmes.

d. **Cadeaux personnalisés** : Dans un contexte non critique, le chiffrement affine peut être utilisé pour créer des cadeaux personnalisés tels que des cartes de vœux, des messages cryptés sur des cadeaux, etc.

e. **Codes secrets pour enfants** : Les enfants peuvent utiliser le chiffrement affine pour créer des codes secrets pour communiquer entre eux et garder leurs messages privés.

Cependant, il est important de noter que le chiffrement affine n'est pas considéré comme sécurisé pour une utilisation sérieuse en cryptographie, car il est vulnérable à des attaques par force brute et à d'autres techniques cryptanalytiques. Dans des applications où la sécurité est primordiale, des algorithmes de chiffrement plus robustes et éprouvés tels que AES (Advanced Encryption Standard) sont utilisés.

6.3.2 Applications du Chiffrement RSA

Le chiffrement RSA est utilisé dans de nombreux domaines de la vie pour sécuriser les communications et protéger les données sensibles. Voici quelques exemples d'utilisation du chiffrement RSA :

a. **Sécurité des communications en ligne :** Le chiffrement RSA est largement utilisé pour sécuriser les communications en ligne, telles que les transactions bancaires en ligne, les achats sur les sites de commerce électronique, l'accès aux comptes utilisateurs sur les réseaux sociaux, etc. Lorsque vous accédez à un site sécurisé (indiqué par "https ://" dans l'URL), RSA est souvent utilisé pour chiffrer les données transmises entre votre navigateur et le serveur du site.

b. **Sécurité des e-mails :** Il est également utilisé pour sécuriser les e-mails. Lorsque vous envoyez un e-mail chiffré, l'algorithme RSA est utilisé pour chiffrer le contenu du message de manière à ce que seul le destinataire légitime puisse le déchiffrer.

c. **Authentification et signatures numériques :** Il est utilisé pour générer des paires de clés publiques et privées, où la clé privée est gardée secrète tandis que la clé publique est partagée avec le monde. Cette paire de clés est utilisée pour des mécanismes d'authentification et de signatures numériques. Par exemple, les certificats SSL/TLS qui sécurisent les sites Web utilisent des signatures numériques basées sur le chiffrement RSA.

d. **Gestion des identités et accès :** Dans les systèmes de gestion des identités et des accès, le chiffrement RSA peut être utilisé pour créer des jetons d'authentification sécurisés et des clés d'accès pour les utilisateurs, garantissant que seules les personnes autorisées peuvent accéder à certaines ressources.

e. **Systèmes de paiement sécurisé :** Le chiffrement RSA est utilisé dans les systèmes de paiement sécurisé pour protéger les informations sensibles lors des transactions électroniques, telles que les paiements par carte de crédit ou de débit en ligne.

f. **Sécurité des appareils IoT (Internet des objets) :** Dans l'Internet des objets, RSA peut être utilisé pour sécuriser les communications entre les appareils et les serveurs, empêchant ainsi les attaquants d'intercepter ou de manipuler les données échangées.

g. **Cryptographie de bout en bout dans les applications de messagerie :** Certaines applications de messagerie utilisent le chiffrement RSA pour assurer la confidentialité des conversations entre les utilisateurs en utilisant le chiffrement de bout en bout. Cela signifie que même le fournisseur de service de messagerie ne peut pas accéder au contenu des messages.

IV Conclusion

7. Conclusion

L'étude des algorithmes d'Euclide et de Bézout revêt une importance capitale dans le domaine de la cryptographie et de la sécurité informatique. Ces deux algorithmes fondamentaux permettent de résoudre efficacement des problèmes liés aux mathématiques discrètes, notamment pour trouver le plus grand commun diviseur de deux entiers et pour déterminer les coefficients de Bézout pour des équations diophantiennes.
Grâce à l'utilisation de ses algorithmes, les cryptographes ont pu
développer des méthodes de chiffrement puissantes, telles que le chiffrement RSA. Ces techniques sont essentielles pour garantir la confidentialité des données dans un monde de plus en plus numérique et interconnecté.
L'étude des algorithmes d'Euclide et de Bézout offre également une initiation précieuse aux concepts mathématiques sous-jacents, tels que les congruences, les arithmétiques modulaires et la théorie des nombres. Cette connaissance approfondie permet de mieux comprendre les mécanismes de cryptographie moderne et offre des possibilités d'innovation dans le domaine de la sécurité informatique.
Ainsi, les algorithmes d'Euclide et de Bézout sont des piliers fondamentaux de la cryptographie et des mathématiques discrètes. Leur compréhension ouvre la voie à un monde sécurisé et confidentiel où les échanges d'informations peuvent s'effectuer en toute tranquillité. Continuer à explorer et à perfectionner ces concepts est essentiel pour relever les défis futurs de la sécurité numérique et préserver la confidentialité des communications dans notre société de plus en plus connectée.

Bibliographie

[1] BODIN A., BOUTIN B. et ROMO P., ***Exo 7***, *Algèbre*, Cours de Mathématiques Première Année, Chap. Arithmétique, pages 45-58, 2016.

[2] BODIN A. et RECHE F., ***Exo 7***, *Algorithmes et informatique* chap. Cryptographie.

[3] ODJOUMANI J., Note de cours d'Algebre commutative, L3-Maths IMSP, 2023.

[4] ARNAULT F., *Theorie des nombres et cryptographie*, Université de Limoges, Cours de D.E.A, 2002 et ses références.

[5] STAMP M. et LOW R., *Applied Cryptanalysis Breaking Ciphers*, San Jose State University, San Jose, CA et ses références.

[6] PRINTEMS J. *Arithmétique modulaire, nombres rationnels et cryptographie*, Module libre sciences « Mathématiques Expérimentales » Licence 2, Université de Paris 12, 2008–09 et ses références.

[7] JAVET J., *Quelques éléments de Cryptographie SageMath et Python (3OC$_{math}$)*, 2021.

[8] Sites Web :

- ***Exo 7*** : http://exo7.emath.fr/un.html
- ***Techno-science*** : https://www.techno-science.net/glossaire-definition/Indicatrice-d-Euler.html
- ***Bibmath*** : https://www.bibmath.net/dico/index.php?action=affiche&quoi=./i/indicateureuler.html
- ***Congruences Côte d'Azur University*** : https://math.univ-cotedazur.fr/~walter/L1_Arith/cours3.pdf