



*École nationale supérieure d'informatique et d'analyse  
des systèmes*

*PROJET Cryptographie*

---

# Cryptographie Homomorphe

---

Réaliser par :

**GUEZRI AMINE**

**BESRI AYMANE**

**Encadrant : Professeur K. Abdelmoumen**

# Résumé

La cryptographie à base de réseaux Euclidiens est aujourd'hui un domaine scientifique en pleine expansion. Son attractivité est plurielle : les opérations sont élémentaires, sa complexité asymptotique quasi-optimale, elle résiste aux ordinateurs quantiques (contrairement aux algorithmes actuels qui seront obsolètes dès l'existence de calculateurs quantiques suffisamment performants), mais surtout rend possible de nouvelles applications qui pourraient permettre à terme d'obtenir un Cloud complètement sécurisé et respectueux de la vie privée. Exemple frappant : vous pourriez effectuer une requête sur un moteur de recherche sans que celui-ci ait connaissance de ce que vous avez recherché.

Considéré comme le Saint Graal de la cryptographie , l'existence d'un tel chiffrement n'a été prouvée possible qu'en 2009 et utilise les réseaux Euclidiens. Il s'agit du **chiffrement complètement homomorphe**.

Dans ce projet, on va étudier des algorithmes de cryptage (partiellement et complètement homomorphe), ensuite on va implémenter les deux algorithmes à savoir **RSA, Paillier** tout en faisant une étude comparative.

**Mots-clés :** Chiffrement homomorphe, Chiffrement dans le cloud, Réseaux euclidiens, Cryptage, RSA, Paillier ...

# Abstract

Cryptography based on Euclidean networks is today a rapidly expanding field of science. Its attractiveness is plural: the operations are elementary, its asymptotic complexity is almost optimal, it resists quantum computers (unlike current algorithms that will be obsolete from the existence of quantum computers sufficiently powerful), but especially makes possible new applications that could ultimately enable a completely secure and privacy-friendly cloud. Striking example: you could query a search engine without it being aware of what you searched for.

Considered the Holy Grail of cryptography, the existence of such encryption has been proven possible only in 2009 and uses the Euclidean networks. This is completely homomorphic encryption.

In this project, we will study encryption algorithms (partially and completely homomorphic), then we will implement the three algorithms namely RSA, Paillier while doing a comparative study.

**Keywords:** Homomorphic Encryption, Cloud Encryption, Euclidean Networks, Encryption, RSA,

# Liste des figures

**Figure 1 : Tableau récapitulatif**

**Figure 2 : Schéma démonstratif**

**Figure 3 : Interface de l'application**

**Figure 4 : Interface RSA – Opération multiplication – Clé 256 bit**

**Figure 5 : Interface RSA – Opération multiplication – Clé 512 bit**

**Figure 6 : Interface RSA – Opération addition**

**Figure 7 : Interface Paillier – Opération addition - clé 512**

**Figure 8: Interface Paillier – Opération addition - clé 1024**

# **Table de matière**

## **Chapitre 1 : Contexte générale du projet**

Problématique

Objectif de projet

La démarche suivie

## **Chapitre 2 : Le Chiffrement**

Introduction

1. Introduction

2. L'homomorphisme

3. Chiffrement partiellement homomorphe

4. Chiffrement complètement homomorphe

Conclusion

## **Chapitre 3 : État de l'art sur les Algorithmes de chiffrement**

Introduction

1. Cryptosystème RSA

1.1. Généralités

1.2. Propriétés homomorphiques

1.3. Exemple

## **2. Cryptosystème Paillier**

### **2.1. Généralités**

### **2.2. Propriétés homomorphiques**

### **2.3. Exemple**

## **Conclusion**

## **Chapitre 4 : Réalisation**

### **1. Paramètre de comparaison des cryptosystèmes**

### **2. Implémentation des cryptosystèmes**

### **3. Interface de l'application**

### **4. Évaluation des cryptosystèmes**

## **Conclusion et perspectives**

## **Bibliographie**

# Introduction générale

La cryptographie, dont on attend en général qu'elle assure la confidentialité et l'authenticité des communications, subit actuellement une mutation significative, qui se traduit par des changements dans les paradigmes mêmes qui la fondent. Jusqu'à présent, la construction des primitives cryptographiques était pensée de point à point, entre deux entités (Alice et Bob), qui souhaitent échanger des données de façon sécurisée.

Cette cryptographie classique est très mal adaptée aux nouveaux usages, liés en particulier au développement du cloud. Le cloud permet de stocker des quantités de données extrêmement importantes, dans des fermes de serveurs sur lesquelles les usagers n'ont en réalité aucune garantie de sécurité. Parmi les fonctionnalités cryptographiques qui seraient nécessaires dans un environnement de ce type, on peut citer l'accès anonyme aux données, les calculs sécurisés sur celles-ci, et le contrôle de leur accès. Ce dernier point est crucial pour maîtriser la diffusion d'informations personnelles. Les politiques d'accès des données personnelles stockées sur le cloud peuvent être à la fois très sensibles (notamment pour les applications médicales), évoluées et dynamiques (comme par exemple sur les réseaux sociaux). Actuellement, les services de gestion de données de type cloud ne mettent en œuvre aucun protocole cryptographique adapté.

Pour concevoir des systèmes cryptographiques avancés permettant d'obtenir ces fonctionnalités avancées, un objet mathématique semble particulièrement prometteur : il s'agit des réseaux euclidiens. Ce sont des grilles de points régulièrement espacés dans l'espace euclidien  $\mathbb{R}^n$ . Ils étaient déjà implicitement utilisés dans les cryptosystèmes de type sac-à-dos, ainsi qu'en cryptanalyse. Mais leur utilisation cryptographique a connu un renouveau récent, grâce aux travaux d'Ajtai, puis de Regev. En plein essor, ils sont aujourd'hui considérés comme l'alternative la plus crédible à la cryptographie traditionnelle, reposant sur des variantes du problème de la factorisation de grands entiers et du problème du logarithme discret dans des groupes algébriques. Les réseaux euclidiens offrent une grande flexibilité pour la conception de cryptosystèmes : ils sont au cœur du premier protocole de chiffrement complètement homomorphe, qui permet d'effectuer n'importe quel calcul (efficace) sur des messages, en ne manipulant que leurs chiffrés.

## **Chapitre 1 : Contexte générale du projet**

### **Problématique :**

Devenu une notion inévitable ces dernières années, le cloud computing est passé d'un concept brumeux à la stratégie ICT «du futur» vers laquelle toute organisation va tôt ou tard se tourner. La raison majeure de cet engouement est la facilité d'accès à un parc de ressources informatiques au potentiel presque illimité, tout cela avec un effort minimal de gestion. Une organisation peut ainsi louer les ressources partagées d'un service cloud, devenant alors locataire (appelé tenant en anglais) d'infrastructure plutôt que propriétaire. Le partage de telles infrastructures est malheureusement le talon d'Achille des services cloud. En effet, des cyber-attaques ont montré la possibilité d'exploiter la proximité de partage des tenants d'un même service cloud. Les problèmes de sécurité, en particulier de confidentialité et intégrité des données, sont donc la préoccupation majeure des utilisateurs du cloud car leurs données se retrouvent gérées hors du cadre de leur gouvernance. Dans le contexte gouvernemental, de sécurité sociale et soins de santé, ces problèmes sont d'autant plus accrus car ils concernent potentiellement les données sensibles des citoyens et entreprises. En 2014, l'étude «Cloud Security Guidance» de la section Recherche de Smals avait brièvement étudié diverses techniques cryptographiques intervenant dans la sécurisation des données dans le cloud. L'objectif de cette présente publication est de détailler davantage les mécanismes cryptographiques connus et existants qui s'appliquent aux environnements cloud pour protéger leurs deux applications les plus connues: le stockage et le traitement sur les données.

### **Objectif de projet :**

L'objectif de notre projet est de choisir le cryptosystème le mieux adapté à nos besoins en terme de (temps d'exécution, propriété homomorphique et sécurité)

### **La démarche suivie :**

Nous avons divisé notre projet en deux grandes parties : Une partie théorique et une autre pratique.



Partie Théorique :

1. Comment manipuler nos informations dans un Cloud en toute sécurité ?
2. Est-ce que la notion d'homomorphisme est supporté par tout les cryptosystèmes ?
3. Parfois on a besoin d'un algorithme rapide, et parfois sécurisé. Nous allons alors étudier un certain nombres de cryptosystèmes pour dégager les points forts de chacun et pour une association optimale Besoin/Algo.

Partie Pratique :

1. Implémentation de trois cryptosystèmes.
2. Réalisation d'une interface graphique englobant tous ces cryptosystèmes.

## **Chapitre 2 : Le Chiffrement**

### **Introduction**

Pour contrer la menace que représentent les pirates du net, qu'ils le fassent par jeu ou pour des raisons commerciales, politiques ou terroristes, une solution évidente apparaît : le chiffrement des données. En effet, rendre les données ou textes illisibles pour des tiers en les remplaçant par d'autres caractères au moyen d'un ou d'une série de moyens techniques contrôlés par l'auteur et mis à disposition du lecteur semble le moyen idéal de se protéger face à ces pirates. Lorsque les informations restent en local, c'est-à-dire qu'elles restent uniquement sur une seule machine (et ne voyagent donc pas sur le réseau), le chiffrement peut sembler superflu en raison des protections multiples habituelles des accès au système local mais lorsque les données sortent sur la toile même temporairement, le chiffrement prend tout son sens. Le chiffrement ou cryptage est un procédé de cryptographie grâce auquel on souhaite rendre la compréhension d'un document impossible à toute personne qui n'a pas la clé de (de)chiffrement. Ce principe est généralement lié au principe d'accès conditionnel.

L'objet de ce présent chapitre est, de présenter des solutions de chiffrement efficaces et plus particulièrement dans le cas de transfert de données ou de requêtes sur internet appelées : le cryptage homomorphique.

Prenant le cas de transfert d'instructions à un Cloud qui est un lieu virtuel de stockage de données sur le Net comme exemple illustrant cette méthode de cryptage :

Lorsque l'on place des données dans un Cloud, elles doivent pouvoir être manipulées. Pour cela, il faut que l'utilisateur émette des requêtes. La logique de requête traditionnelle à un Cloud nécessite plusieurs étapes:

#### **1. Envoyer la requête au serveur Cloud pour qu'il l'exécute:**

- a.** Chiffrer la requête sur l'ordinateur du client (l'émetteur de la requête).
- b.** Ensuite l'envoyer au serveur qui contient la base de données (le Cloud) au travers du réseau.

c. Faire exécuter la requête par le serveur entre les étapes de déchiffrement de la requête et rechiffrement du résultat.

## **2. Récupérer les données chiffrées :**

a. Le serveur envoie les données chiffrées correspondantes à la requête.

b. Les données chiffrées sont déchiffrées sur la machine du client pour permettre leurs utilisations.

Le problème majeur qui se pose est de trouver le/les moyens pour chiffrer la requête et faire en sorte qu'elle soit utilisable par le serveur. De plus, il apparaît déjà que l'exécution, dans le cas d'un système de chiffrement traditionnel, se fait sans protection dans le Cloud lui-même car la requête est en effet déchiffrée puis exécutée en clair donc sans codage. Ce qui représente évidemment un danger en terme de protection face au propriétaire physique du Cloud ou toute autre personne ayant réussi à en prendre le contrôle de manière licite ou illicite. Il existe, à ce jour, plusieurs solutions très différentes. Afin de les juger à leur juste valeur, certaines classifications seront faites.

## **1. L'homomorphisme**

Le cryptage homomorphe est une forme de cryptage qui permet d'effectuer certains types de calculs sur un texte chiffré et de générer un résultat chiffré qui, une fois déchiffré, correspond au résultat d'opérations effectuées sur le texte en clair.

Par exemple, un chiffrement capable de faire:

$$\text{Dec}[\text{Enc}(1) + \text{Enc}(2)] = 3$$

Appliquée au stockage des données chiffrées placées sur le Cloud, la définition de cette méthode de cryptage semble être une bonne solution permettant de traiter, au sein même du Cloud, la requête sans devoir décrypter les données au niveau du serveur.

Pour exprimer ce type de chiffrement à travers une analogie concrète, prenons l'exemple d'un gérant d'une bijouterie.

Ce gérant aimerait que ses employés assemblent des pierres et des matériaux précieux pour en faire des bijoux finis mais il est très inquiet par rapport aux vols. Il résout le problème en construisant des boîtes pourvues de gants, boîtes dans lesquelles il stocke les pierres et matériaux précieux et dont il est le seul à posséder la clé. En utilisant les gants, un employé peut manipuler les pierres ou matériaux à l'intérieur des boîtes.

L'employé peut aussi introduire de nouveaux matériaux sans que rien ne puisse en sortir. Une fois le travail terminé, le gérant peut venir récupérer le travail terminé en utilisant sa clé.

Bien sûr, cette analogie n'est pas entièrement correcte car dans le cas d'un chiffrement totalement homomorphique, seul le produit final importe.

Plusieurs vieux systèmes de chiffrement connus tels que RSA (1977 ), Benaloh (1994 ou Paillier (1999 ) sont partiellement homomorphiques. En effet, ils permettent le calcul homomorphique d'une seule opération (l'addition ou la multiplication) sur des textes en clair . Un système de cryptographie qui supporte à la fois l'addition et la multiplication est appelé "cryptage entièrement homomorphique" (ou Fully Homomorphic Encryption en anglais, ) et est beaucoup plus puissant. En outre, l'utilisation d'un tel système permet à un processus ou à un algorithme d'être "homomorphisé".

### Définition 1:

Pour l'exprimer autrement, l'essence même de l'encryptions homomorphique est simple:

Soit un texte original : **( N1, N2, ....., Nt)**

Et sa version chiffrée : **A( N1, N2, ....., Nt)**

Un chiffrement homomorphique de ce texte sera tout chiffrement qui permet à n'importe qui (pas seulement le détenteur des clés de cryptage) de produire un texte chiffré qui encode **f ( N1, N2, ....., Nt)** Pour toute fonction **f** .

C'est-à-dire que n'importe qui sera capable d'effectuer une permutation, à partir de

**A( N1, N2, ....., Nt)    comme suit    A( f ( N1, N2, ....., Nt) )**

## **Définition 2:**

Un système de chiffrement homomorphe est un cryptosystème permettant de faire des calculs sur les données chiffrées. Formellement, si **C1** (respectivement **C2**) est un chiffré de **m1** (respectivement **m2**) il existe deux opérations et telles que + et \*

$$\text{Dec}(C1 + C2) = \text{Dec}(C1) * \text{Dec}(C2) = m1 * m2$$

## **2. Chiffrement partiellement homomorphe**

Un cryptosystème de chiffrement est simplement homomorphique ou (partiellement homomorphe) s'ils ne supportent qu'une seule opération : addition ou bien multiplication, par exemple on peut dire que RSA est partiellement homomorphe pour la loi multiplication

## **3. Chiffrement complètement homomorphe**

Un système de chiffrement complètement homomorphe n'est rien d'autre qu'un système de chiffrement homomorphe où toute fonction peut être évaluée sur les données chiffrées. Comme toute fonction peut être exprimée comme un polynôme et qu'un polynôme consiste en une série d'additions et de multiplications, un système de déchiffrement sera complètement homomorphe dès lors qu'il permettra d'évaluer un nombre arbitraire d'additions et de multiplication.

## **Conclusion**

Dans ce chapitre, on a vu la définition du chiffrement partiellement et complètement homomorphe. On a vu leurs utilités dans l'environnement de virtualisation Cloud.

Par la suite, on va implémenter deux cryptosystèmes ( l'un additivement homomorphe, l'autre est homomorphe pour les deux opérations addition et multiplication).

## Chapitre 3 : État de l'art sur les Algorithmes de chiffrement

### Introduction

Le développement de stockage en nuage est plates-formes informatiques permet aux utilisateurs d'externaliser le stockage et les calculs sur leurs données, et permet aux entreprises de se décharger de la tâche du maintien de centres de données. Cependant, les inquiétudes sur la perte de la vie privée et d'affaires valeur des données privées est une écrasante obstacle à l'adoption des services cloud par les consommateurs et les entreprises. Une excellente façon d'apaiser ces vie privée préoccupations est de stocker toutes les données dans le nuage crypté, et effectuer des calculs sur des données chiffrées. À cette fin, nous besoin d'un système de

cryptage qui permet le calcul significatif sur les données chiffrées. à savoir un chiffrement homomorphique schème qui permet d'effectuer certains types de calculs sur un texte chiffré.

Dans ce chapitre, nous décrivons certains systèmes de chiffrement homomorphes qui ont créé un intérêt considérable chez les chercheurs dans le domaine de la cryptographie. RSA , Goldwasser\_Micali , Paillier et ses variantes sont bien connues pour leur efficacité et le niveau élevé de sécurité pour qu'ils fournissent homomorphique le cryptage. Nous ne discutons pas leurs considérations mathématiques en détail, mais résumons leurs paramètres et propriétés importante.

### 1. Cryptosystème RSA

Le chiffrement RSA (nommé par les initiales de ses trois inventeurs) est un algorithme de cryptographie asymétrique, très utilisé dans lecommerce électronique, et plus généralement pour échanger des données confidentielles sur Internet. Cet algorithme a été décrit en 1977 par Ronald Rivest, Adi Shamir et Leonard Adleman.

## 1.1. Généralités

### Génération de clés :

1. Choisir  $p$  et  $q$ , deux nombres premiers distincts ;
2. calculer leur produit  $n = pq$ , appelé *module de chiffrement* ;
3. calculer  $\varphi(n) = (p - 1)(q - 1)$  (c'est la valeur de l'indicatrice d'Euler en  $n$ ) ;
4. choisir un entier naturel  $e$  premier avec  $\varphi(n)$  et strictement inférieur à  $\varphi(n)$ , appelé *exposant de chiffrement* ;
5. calculer l'entier naturel  $d$ , inverse de  $e$  modulo  $\varphi(n)$ , et strictement inférieur à  $\varphi(n)$ , appelé *exposant de déchiffrement* ;  $d$  peut se calculer efficacement par l'algorithme d'Euclide étendu.

### Chiffrement :

Si  $M$  est un entier naturel strictement inférieur à  $n$  représentant un message, alors le message chiffré sera représenté par

$$C \equiv M^e \pmod{n},$$

l'entier naturel  $C$  étant choisi strictement inférieur à  $n$ .

### Déchiffrement :

Pour déchiffrer  $C$ , on utilise  $d$ , l'inverse de  $e$  modulo  $(p - 1)(q - 1)$ , et l'on retrouve le message clair  $M$  par

$$M \equiv C^d \pmod{n}.$$

## 1.2. Propriétés homomorphiques

RSA est homomorphique multiplicatif.  $D(E(m_1) * E(m_2)) \bmod N = m_1 * m_2 \bmod N$

### 1.3. Exemple

Un exemple avec de petits nombres premiers (en pratique il faut de très grands nombres premiers)

1. on choisit deux nombres premiers  $p = 61$ ,  $q = 53$
2. leur produit  $N = 61 \times 53 = 3233$
3.  $\phi(n) = (61 - 1) \times (53 - 1) = 60 \times 52 = 3120$
4. on choisit  $e = 17$  (premier avec 3120)  $\text{pgcd}(e, \phi(N)) = 1$ .
5.  $d = 2753$  tel que  $e.d = 17 \times 2753 \equiv 1 \pmod{3120}$ .

La clé publique est  $(N, e) = (3233, 17)$ , et clé privée  $(N, d) = (3233, 2753)$ .

On a  $m=65$

Pour chiffrer le message on calcule :

$$C = m^e \pmod{N}$$

$$C = 65^{17} \pmod{3233} = 2790$$

Pour déchiffrer le cryptogramme  $C$  on calcule :

$$m = C^d \pmod{N}$$

$$m = 2790^{2753} \pmod{3233} = 65$$

## 2. Cryptosystème Paillier

Le cryptosystème de Goldwasser\_Micali (GM) est un algorithme asymétrique de cryptographie à clé publique, développé par Shafi Goldwasser et Silvio Micali en 1982. Fait notoire, GM le premier cryptosystème à chiffrement probabiliste qui est prouvablement sûr avec des hypothèses cryptographiques standards. Toutefois, il n'est pas efficace : les textes chiffrés peuvent être des centaines de fois plus longs que les textes d'origine.



## 2.1 Généralités

### Génération de clés :

1. Choisir deux **nombre premiers** de grande taille, indépendants et aléatoires :  $p$  et  $q$  ;
2. Calculer la clef publique  $\mathbf{pk} \triangleq N = p \cdot q$  (un module **RSA**) et la clé privée  $\mathbf{sk} \triangleq \varphi(N) = (p - 1) \cdot (q - 1)$  .

### Chiffrement :

Soit  $m$  un message à chiffrer avec  $0 \leq m < N$  . Soit  $r$  , un entier aléatoire tel que  $0 < r < N$  (appelé l'aléa). Le chiffré est alors :

$$c = (1 + N)^m \cdot r^N \mod N^2$$

### Déchiffrement :

Pour retrouver le texte clair  $m$ , on commence par remarquer que :

$$c \equiv r^N \mod N,$$

et

$$\begin{aligned} \frac{c}{r^N} &= (1 + N)^m \mod N^2 \\ &= 1 + m \cdot N + N^2 + N^2 \cdot \left( \sum_{i=2}^m \binom{m}{i} m^i N^{m-i-2} \right) \mod N^2 \\ &= 1 + m \cdot N \mod N^2. \end{aligned}$$

On obtient ainsi :

$$r = c^{N^{-1} \mod \varphi(N)} \mod N.$$

D'où :

$$m = \frac{(c \cdot r^{-N} \mod N^2) - 1}{N}.$$

## 2.2. Propriétés homomorphiques

Le cryptosystème Paillier est homomorphique additif et multiplicatif.

$$D(E(m_1) * E(m_2)) \bmod N^2 = m_1 + m_2 \bmod N$$

ET

$$D(E(m_1)^{E(m_2)}) \bmod N^2 = m_1 * m_2 \bmod N \text{ ( de meme si on inverse } m_1 \text{ et } m_2)$$

## 2.3. Exemple

Un exemple de schéma de chiffrement Paillier avec petite paramètres sont indiquées comme suit.

$$p = 7; \quad q = 11 \text{ Alors } N = p * q = 7 * 11 = 77$$

$$\lambda = \lambda(N) = \text{ppcm}(p-1, q-1) = 30$$

On prend  $g = 5652$  tel que  $\text{pgcd}((g^\lambda \bmod N^2) - 1)/N, N) = 1$

Pour chiffrer un message  $m = 42$

où  $m \in \mathbb{Z}_N$  choisissez un hasard

$$r = 23$$

où  $r$  est un entier non nul et  $r \in \mathbb{Z}_N$  . et on calcule :

$$C = g^m r^N \bmod N^2 = 4624 \bmod 5929 = 4624$$

Pour déchiffrer le cryptogramme  $C$  on calcule :

$$\text{On a } l(u) = (u - 1)/N$$

$$\text{On calcul } k = l(g^\lambda \bmod N^2) = 51^{77}$$

$$u = k^{-1} \bmod N = 51^{-1} \bmod 77 = 74$$

$$\text{On trouve alors } m = l(c^{\lambda} \bmod N^2) \cdot u \bmod N = l(4852) \cdot 74 \bmod 77 = 42$$

## Synthèse :

Algorithme	Clé			Chiffrement	Déchiffrement	Homomorphique		
	clé publique	Clé privée	génération des clés			Additif	Multiplicatif	OU exclusif
Paillier	$(g, N)$	$(p, q)$	$N = p * q$ $\lambda(N) = \text{ppcm}(p-1, q-1)$ $g =$ $\text{pgcd}\left(\frac{(g^{\lambda} \bmod N^2) - 1}{N}, N\right) = 1$	$m \in \mathbb{Z}_N$ $r \in \mathbb{Z}_N$ $C = g^m r^N \bmod N^2$	$l(u) = (u - 1)/N$ $k = l(g^{\lambda} \bmod N^2)$ $u = k^{-1} \bmod N$ $m = l(c^{\lambda} \bmod N^2) \cdot u \bmod N$	✓	✓	✗
RSA	$(N, e)$	$(p, q, d, \phi(N))$	$N = p * q$ $\phi(N) = (p - 1)(q - 1)$ $0 < e$ $< \phi(N), \text{pgcd}(e, \phi(N)) = 1$ $d = e^{-1} \bmod \phi(N)$	$m \in \mathbb{Z}_N$ $C = m^e \bmod N$	$m = C^d \bmod N$	✗	✓	✗

« Figure 1 : Tableau récapitulatif »

## Conclusion

Dans ce chapitre, nous avons étudié les deux algorithmes (RSA et Paillier). Nous avons détaillés les étapes qui constituent la procédure avec laquelle fonctionnent ces algorithmes. Nous avons donnés des exemple pour bien comprendre le fonctionnement de ces cryptosystèmes. Enfin, nous avons dressés un tableau récapitulatif des différents cryptosystèmes.

## Chapitre 4 : Réalisation

Ce chapitre décrit la phase de l'implémentation des deux cryptosystèmes RSA et paillier. Tout d'abord, on va définir les paramètres qu'on va prendre en compte lors de la comparaison entre les deux cryptosystèmes. Ensuite, nous allons les implémenter pour en finir faire une évaluation.

### 1. Paramètre de comparaison des cryptosystèmes

Les paramètres de comparaison sont :

- Le temps d'exécution.
- La taille du texte chiffré
- Les opérations homomorphique.

Nous citons aussi le matériel utilisé pour l'implémentation de ce projet :

**Processeur : Intel(R) Core(TM) i7 – 3520M CPU @ 2,90 GHz 2,90 GHz**

**RAM : 8,00 Go**

Le langage utilisé est : Python

L'interface est implémenter à l'aide de la bibliothèque : Tkinter

### 2. Implémentation des cryptosystèmes

Dans cette partie on va implémenter deux cryptosystèmes qui peuvent être utilisés au niveau du cloud vu leurs propriétés homomorphique.

Voilà un schéma qui explique l'idée de ce projet de manière détaillée :

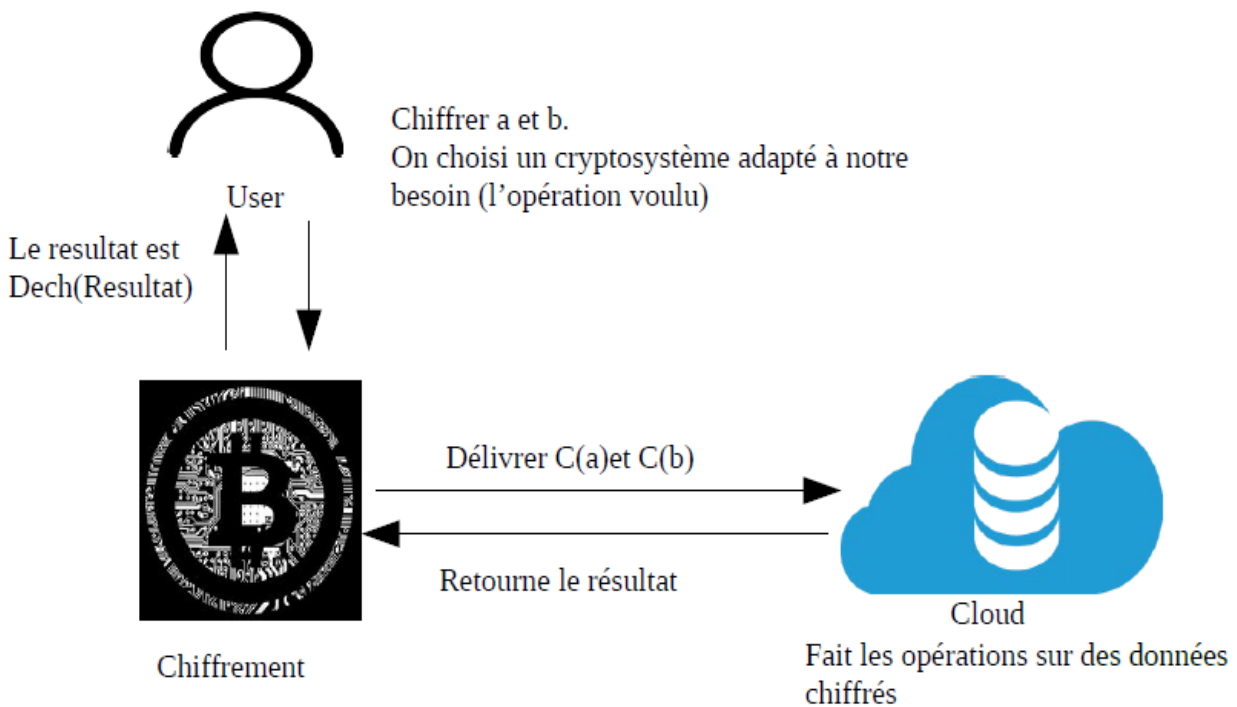
Un utilisateur veut faire l'opération  $2 + 5$  mais il n'a pas les moyens pour le faire. Ça peut être dû à une insuffisance au niveau de puissance de calcul ...

L'utilisateur est obligé alors de délivrer ces informations à une entité qui a les moyens nécessaires : Le Cloud.

Le problème n'est pas résolu puisque les informations sont d'une importance ultime, et notre utilisateur ne veut pas que ces informations soient vues par d'autres entités.

Voilà la solution :

$a = 2$  et  $b = 5$



« Figure 2 : Schéma démonstratif »

### 3. Interface de l'application

Dans cette partie, on va décrire les composantes de notre application.

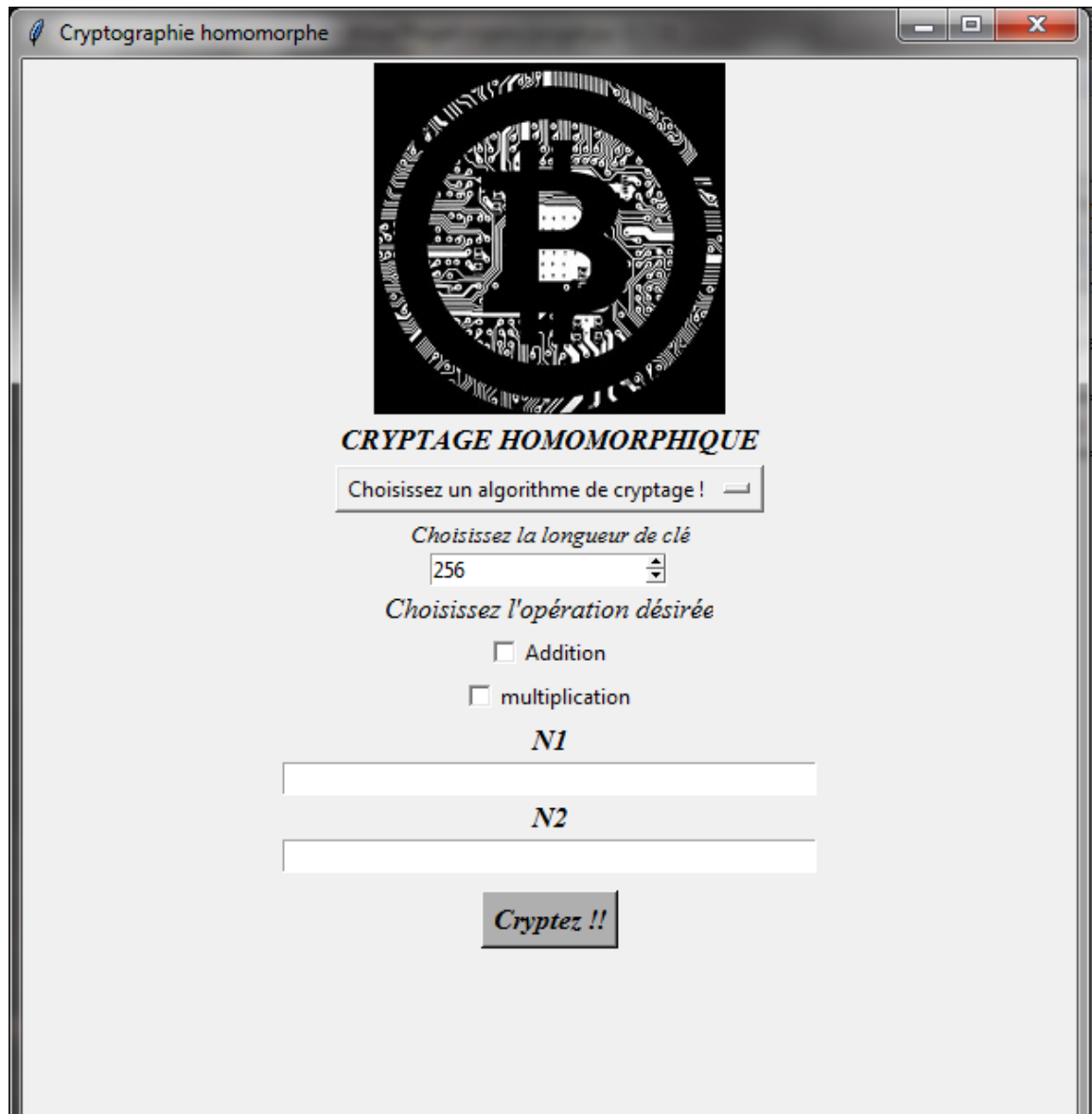
- + *Choice box 1* : Comportant les cryptosystèmes RSA et pailler
- + *Choice box 2* : Comportant la taille des clés (256,512 ou 1024)
- + *Des choix* : Addition ou multiplication.
- + *Deux TestField* : Pour saisir les deux nombres pour faire l'opération

+ Un bouton 'Cryptez' : Pour crypter les deux nombres, en utilisant le cryptosystème choisi dans Choice box 1.

Notre application crypte les deux nombres  $N1$  et  $N2$ . Nous aurons alors  $C1$  et  $C2$ .

Nous faisons l'opération sur  $C1$  et  $C2$  (C'est le cloud qui est supposé faire cette opération)

Le résultat est  $C3$ . On déchiffre  $C3$ , et on vérifie si le résultat est correcte.




« Figure 3: Interface de l'application »

## 4. Évaluation des Cryptosystèmes

### Cryptosystème RSA :

Cryptographie homomorphe



**CRYPTAGE HOMOMORPHIQUE**

RSA

Choisissez la longueur de clé

256

Choisissez l'opération désirée

☐ Addition

☒ multiplication

**N1**

2

**N2**

5

**Cryptez !!**

Le chiffré de N1 est:

52264034481725472425243279283904976869735546171541738612295250497516937220585

Le chiffré de N2 est:

34172963222550503981779781926160784652466857084120071332441015911670196423536

Le produit des deux nombres est :

16724939293400612114222871507791877398056748991502335920679502420209856316092

**Decryptage ...**

10

TEMPS D'EXECUTION

0.0760042667388916

« Figure 4: Interface RSA – Opération multiplication – Clé 256 bit »



## CRYPTAGE HOMOMORPHIQUE

RSA

Choisissez la longueur de clé

512

Choisissez l'opération désirée

☐ Addition

☒ multiplication

$N1$

5

$N2$

4

**Cryptez !!**

Le chiffré de  $N1$  est:

57695762351517177971564427261123052615451599367786688091865036513533476918596774865030

Le chiffré de  $N2$  est:

58473670923595351603387505652951582078732598966741768466590096630532138387377898340719

Le produit des deux nombres est :

5017368121036122374012235306651938712906166453907927026762717807739600767948993828080

**Decryptage ...**

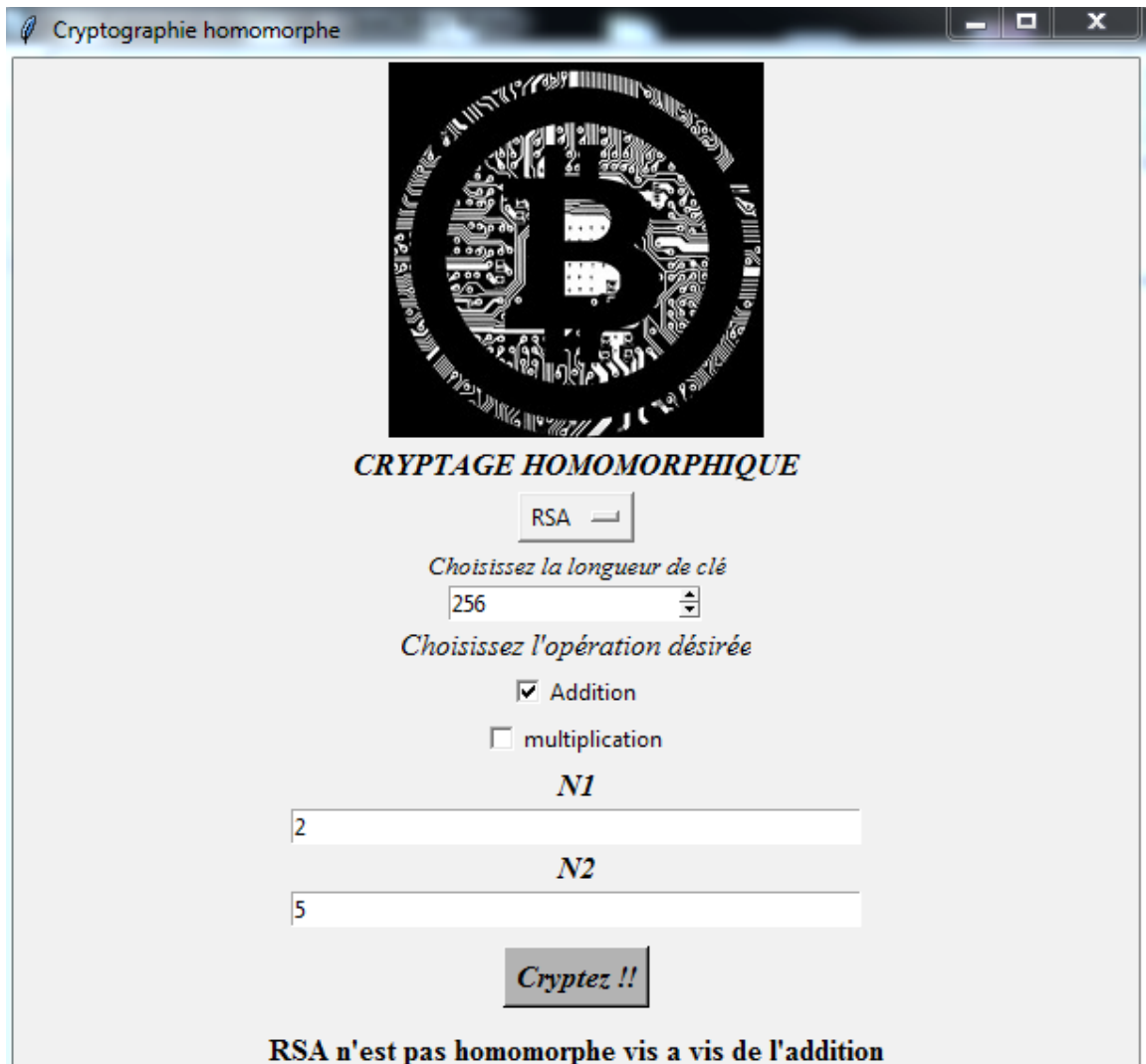
10

TEMPS D'EXECUTION

0.29401683807373047

« Figure 4: Interface RSA – Opération multiplication – Clé 512 bit »





« Figure 5: Interface RSA – Opération addition »

Comme on a vu dans la partie théorique, le cryptosystème RSA n'est pas additivement homomorphe. Le cryptage de  $N1$  et  $N2$  normalement devra se faire sans problème. Mais On a pas vu l'intérêt de crypter les deux nombres sans faire l'opération voulu : Parce qu'en premier lieu le but du projet et le chiffrement homomorphe.

## Cryptosystème Paillier:

Cryptographie homomorphe



**CRYPTAGE HOMOMORPHIQUE**

Paillier

Choisissez la longueur de clé

512

Choisissez l'opération désirée

☒ Addition

☐ multiplication

**N1**

2

**N2**

5

**Cryptez !!**

Le chiffré de N1 est:

57048960807882639546933400028637912277068780813085386042350283524726463185361526883029168273769889056

Le chiffré de N2 est:

7369045731476269113227873955638706445582346947056521750113905731866511048913393071437263517782496213

La somme des deux nombres est :

3497946213240998315486668810130351910779506830634413334557293324666735259941910255115305683212018515

**Decryptage ...**

7

TEMPS D'EXECUTION

0.3480198383331299

« Figure 6: Interface Paillier – Opération addition - clé 512»



## CRYPTAGE HOMOMORPHIQUE

Paillier

Choisissez la longueur de clé

1024

Choisissez l'opération désirée

☒ Addition

☐ multiplication

$N1$

2

$N2$

5

Cryptez !!

Le chiffré de  $N1$  est:

17527448747590246111059336104106528991771230835999831519499178075

Le chiffré de  $N2$  est:

24681438820530861456164207688025842639528475859333354671153925134

La somme des deux nombres est :

32410050829131053344844280878424994240555663313212486810890847676

Decryptage ...

7

TEMPS D'EXECUTION

1.6692028045654297

« Figure 7: Interface Paillier – Opération addition - clé 1024»

## Synthèse :

Pour une clé de chiffrement de 256 bits.

	RSA	Paillier
Temps de exécution (ms)	16	32
Taille du texte chiffré(caractère)	154	308

Pour une clé de chiffrement de 512 bits.

	RSA	Paillier
Temps de exécution (ms)	31	234
Taille du texte chiffré(caractère)	308	616

Pour une clé de chiffrement de 1024 bits.

	RSA	Paillier
Temps de exécution (ms)	156	1700
Taille du texte chiffré(caractère)	612	1233

## Conclusion

Le temps d'exécution de RSA est inférieur au temps d'exécution de Paillier. Mais La taille du texte chiffré au niveau de Paillier est plus grande.

*D'où l'axiome suivant :*

Un client X veut utiliser le chiffrement homomorphe. Il est devant deux méthodes : RSA ou Paillier.

« X opte pour une sécurité optimale → Choisir Paillier »

« X opte pour un temps d'exécution optimale → Choisir RSA »

## **Conclusion et perspectives**

## **Bibliographie**