

Chapitre 4

Implémentation et résultats expérimentaux

Sommaire :

2. Outil de programmation.....	38
3. Utilisation de java pour chiffrement des images.....	39
4. Structure générale d'application.....	42
5. Résultats expérimentaux.....	42
5.1. Opérations de chiffrement/déchiffrement d'image	43
5.2. Analyse d'histogrammes	44
5.3. Chiffrement des clés.....	46

1. Introduction

Pour évaluer la performance des algorithmes de chiffrement d'image qui sont présenté dans le précédent chapitre, nous allons implémenter l'algorithme de chiffrement hybride avec le langage Java. Nous utilisons l'environnement de développement NetBeans, cet environnement et ce langage sont enrichis par des packages des domaines de cryptographie et le traitement d'images.

2. Outil de programmation

2.1. Java: est un langage de programmation dans la tradition de C et C++, par conséquent, si vous avez une expérience avec C ou C++, vous vous retrouverez en terrain connu souvent que vous apprenez les différentes fonctionnalités de Java.

Cependant, Java se distingue des autres langages de programmation dans quelques significatif, Les sections suivantes décrivent les différences les plus importantes:

❖ **L'indépendance de plate-forme:** Une des principales raisons pour Java est si populaire, c'est son indépendance de plate-forme, ce qui signifie simplement que les programmes Java peuvent être exécutés sur différents types d'ordinateurs, un programme Java s'exécute sur n'importe quel ordinateur avec un environnement d'exécution Java.

ü **Orienté objet:** Java est intrinsèquement orienté objet, ce qui signifie que les programmes Java sont fabriqués à partir d'éléments de programmation appelé objets, tout simplement un objet est une entité de programmation qui représente soit un objet du monde réel ou un concept abstrait.

ü **L'API Java (Application Programming Interface):** Le langage Java lui-même est très simple. cependant, java est livré avec une bibliothèque de classes qui fournissent des fonctions d'utilité couramment utilisés que la plupart des programmes, Java ne peuvent pas faire sans, cette bibliothèque de classes, appelé l'API Java, est autant une partie de Java comme langage lui-même. en fait le véritable défi d'apprendre à utiliser Java n'est pas l'apprentissage de la langue, il apprend l'API. Le langage Java n'a que 48 mots-clés, mais l'API Java possède plusieurs milliers de classes, avec des dizaines de milliers de méthodes que vous pouvez utiliser dans vos programmes. [36]

2.2. NetBeans: est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and Distribution License). en plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML, il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

NetBeans constitue par ailleurs une plate forme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE NetBeans s'appuie sur cette plate forme. L'ide NetBeans s'enrichit à l'aide de plugins. [37]

3. Utilisation de java pour chiffrement des images

Pour savoir comment crypter et décrypter d'une image numérique en Java, il faut connaître la manipulation de l'image en ce langage. On cite ci-dessous quelques bibliothèques et packages utilisées dans notre travail:

3.1. Manipulation des images en java : Pour travailler avec les images sous Java 2D, il faut connaître deux classes importantes de l'API :

ü **java.awt.Image :** c'est la super classe fournie dans la JDK depuis sa version 1.0. C'est une classe abstraite qui permet de représenter les images sous forme d'un rectangle de pixels. La restriction de cette classe est qu'elle ne permet pas d'accéder aux pixels. Elle est donc inadaptée pour le traitement d'images.

Ü **java.awt.image.BufferedImage** : Ajoutée à la JDK depuis sa version 2, elle hérite de la classe `Image` et implémente ses interfaces pour permettre d'examiner l'intérieur des images chargées et travailler directement sur ses données. On peut donc à partir d'un objet `BufferedImage`, récupérer les couleurs des pixels et changer ces couleurs. Comme son nom l'indique, `BufferedImage` est une image tampon. Cette classe gère l'image dans la mémoire et fournit des méthodes pour l'interprétation des pixels. Si on veut faire du traitement d'images, il faut donc travailler avec des objets `BufferedImage`. [38]

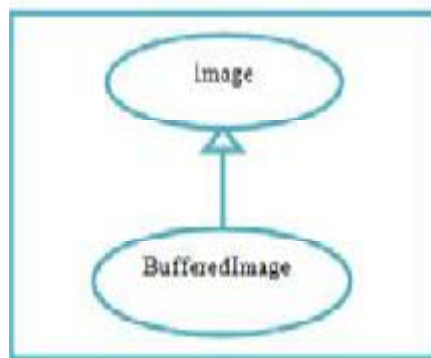


Figure 4.1. Architecture de l'API de Java2D. [38]

3.2. Cryptographie en java

La plate-forme Java insiste fortement sur la sécurité, y compris la sécurité de la langue, de la cryptographie, infrastructure à clé publique et l'authentification, la communication sécurisée et le contrôle d'accès.

Le JCA(Java Cryptography Architecture) est une pièce majeure de la plate-forme, et qui contient une architecture de " provider" et un ensemble d'API pour les signatures numériques, résumés de message (hashes), certificats et la validation du certificat, chiffrement (bloc / flux de chiffrement symétrique / asymétrique), la génération de clés et de gestion, et de la génération de nombre aléatoire sécurisé, pour n'en nommer que quelques-uns, ces API permettent aux développeurs d'intégrer facilement la sécurité dans leur code d'application.[39]

L'API Java Security est un ensemble de packages qui sont utilisés pour l'écriture de programmes sécurisés en Java, en particulier, les classes et interfaces dans les paquets suivants font partie de l'API de sécurité:

Ü **org.bouncycastle:** Le package crypto de Bouncy Castle est une implémentation d'algorithmes cryptographiques pour la plateforme Java, les algorithmes du package BC sont conformes au Framework JCE, et est un package libre et open source, il est distribué sous la License MIT X Consortium, il ressemble à la librairie C open ssl qui est conforme aux différents standards en vigueur.

L'API BC existe aussi en version C# et fournit les mêmes fonctionnalités que celle du Java. [40]

Ü **javax.crypto:** Beaucoup de classes prévues dans ce package sont basées sur le fournisseur, la classe se définit une interface de programmation d'applications qui peuvent écrire. Les implémentations eux-mêmes peuvent alors être écrites par les fournisseurs et tiers indépendants en toute transparence selon les besoins, c'est pourquoi les développeurs d'applications peuvent profiter de n'importe quel nombre d'implémentations de fournisseur à base sans avoir à ajouter ou à réécrire le code.

Soutien à cryptage inclut symétriques, asymétriques, bloc, et les chiffrements de flux, ce package soutient également les flux sécurisés et les objets scellés.

Ü **java.security:** La technologie de sécurité Java comprend un grand ensemble d'API, outils, et des implémentations d'algorithmes de sécurité couramment utilisés, mécanismes et protocoles, L'API de sécurité de Java couvre un large éventail de domaines, notamment la cryptographie, infrastructure à clé publique, et sécurise la communication, l'authentification et le contrôle d'accès. La technologie de sécurité Java fournit au développeur un cadre de sécurité complète pour les applications d'écriture, et fournit également à l'utilisateur ou à l'administrateur avec un ensemble d'outils pour gérer les applications en toute sécurité. [41]

4. Structure générale d'application

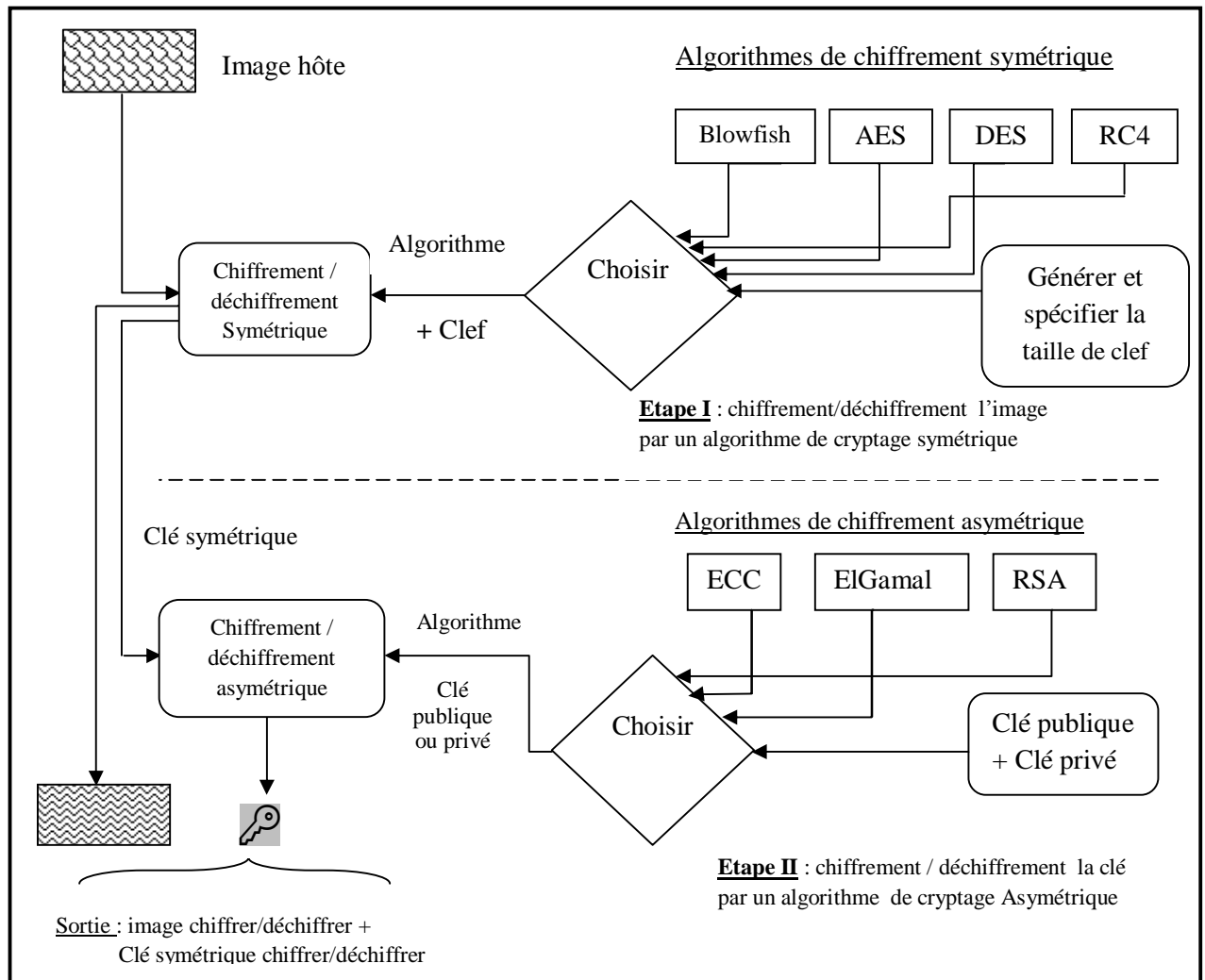


Figure 4.2 schéma générale d'une application de sécurisé d'images.

5. Résultats expérimentaux

Afin de mener une étude comparative pour plusieurs algorithmes de chiffrement, nous choisirons quelques algorithmes de chiffrement symétrique importants. Pour le chiffrement d'image (AES, DES, RC4, Blowfish), et aussi, nous choisirons des algorithmes de chiffrement asymétrique pour le chiffrement de clé secret (RSA, Elgamal, ECC).

Nous avons réalisé un programme sous l'environnement Windows7 et sur la plateforme Java.

Le tableau ci-dessous présente différentes tailles des images testées :

Tailles	100 Ko	134 Ko	217 Ko	290 Ko	405 Ko	486 Ko	650 Ko
Pixels	256*256	350*350	550*550	700*700	900*900	1020*1020	1300*1300

Tableau 4.1 Images de différentes tailles (Ko) et différentes Pixels.

La machines sur laquelle ces algorithmes ont été testés a les caractéristiques suivantes :

- Ø CPU: Intel(R) Core(TM) i3 M380 @ 2.53GHz 2.53 GHz.
- Ø RAM : 4 GB DDR3 Memory.
- Ø Type du système : système d'exploitation 32 bit.

Les résultats obtenus lors de l'exécution de ce programme ont donné deux parties (opération sur les images / sur les clés) :

5.1. Opérations de chiffrement/déchiffrement d'image : Les résultats d'opérations de chiffrement ont donné sur le tableau suivant :

	Temps et vitesse de cryptage (chiffrement des images) en milliseconde								
Taille en Ko	100 Ko	134 Ko	217 Ko	290 Ko	405 Ko	486 Ko	650 Ko	Durée totale	Vitesse Ko/mi-sec
AES (128 bits)	15	19	32	44	72	95	151	428	5.33
DES (56 bit)	63	103	215	347	575	738	1248	3289	0.70
RC4 (120 bit)	10	15	17	24	43	55	87	251	9.09
Twofish (120 bit)	17	27	32	52	86	110	177	501	4.55

Tableau 4.2 Temps de chiffrement des images (ms) pour des tailles différentes (ko).

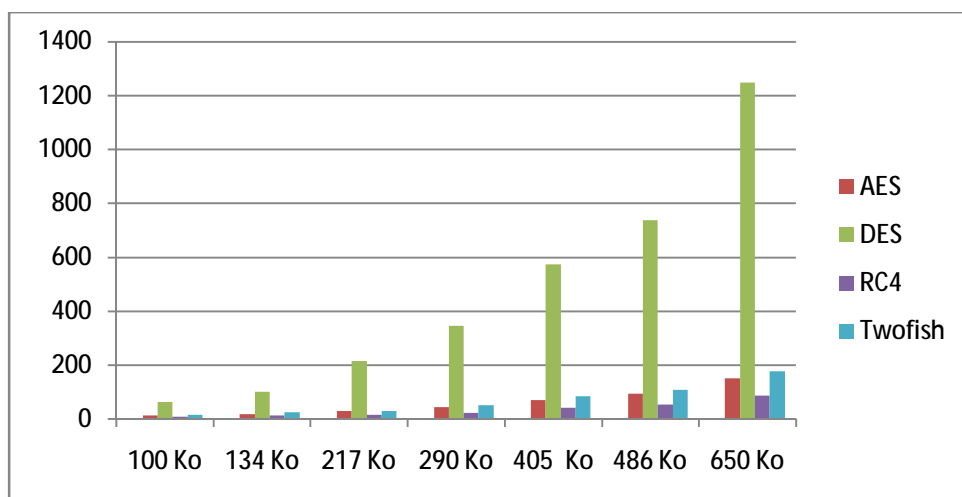


Figure 4.3 Graphe du temps de chiffrement des images. (Ms)

✓ Conclusion tirées de ce tableau

ü Le temps de chiffrement de l'image est proportionnel à sa taille.

ü Quelle que soit la taille de l'image, les algorithmes classés par rapidité de chiffrement décroissante sont comme suit : RC4, AES, Twofish, DES.

Les opérations de déchiffrement ont donné le tableau suivant :

	Temps et vitesse de décryptage (déchiffrement des images) en milliseconde								
Taille en Ko	100 Ko	134 Ko	217 Ko	290 Ko	405 Ko	486 Ko	650 Ko	Durée totale	Vitesse Ko/mi-sec
AES (128 bits)	14	21	27	44	76	97	157	436	5.23
DES (56 bit)	67	97	209	347	561	719	1182	3182	0.72
RC4 (120 bit)	9	13	24	26	40	51	83	246	9.28
Twofish (120 bit)	20	24	34	57	94	118	193	540	4.22

Tableau 4.3 Temps de déchiffrement des images (ms) pour des tailles différentes (ko).

✓ Conclusion tirées de ce tableau

ü Le temps de chiffrement d'une image de taille quelconque est presque égal au temps de son déchiffrement par le même algorithme et même si la taille de l'image augmente, la différence est en générale de l'ordre de quelques millisecondes.

ü Pour les petites tailles, la différence est presque nulle.

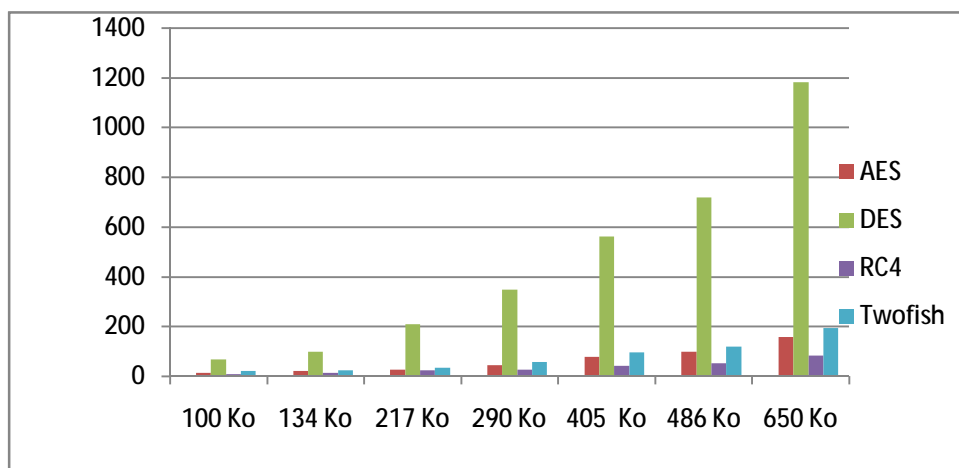

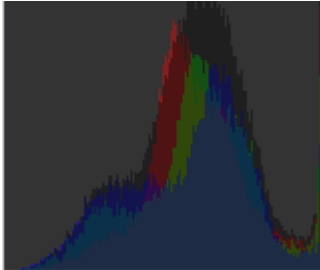
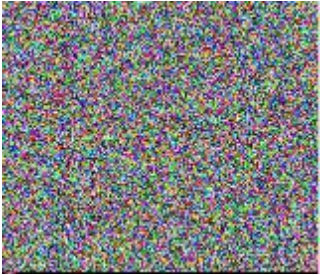
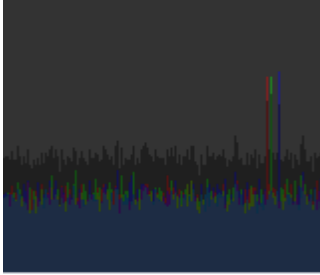

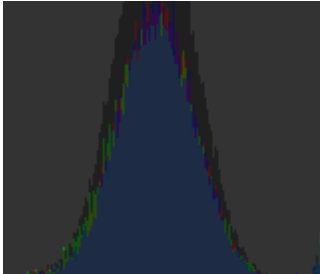

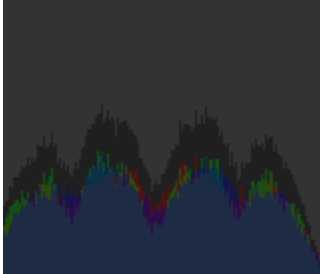


Figure 4.4 Graphe du temps de déchiffrement des images. (ms).

5.2. Analyse d'histogrammes : Un histogramme d'images montre la manière de distribution des pixels dans une image en traçant le nombre de pixel correspondant à chaque intensité de couleur.

A titre d'exemple, l'image (a) de la Tableau 4.4 (image l'*Emir Abdel Kader*) de taille (240/197 px) et ses images chiffrées par les algorithmes de chiffrement proposé sont montrées avec leur histogrammes dans la tableau suivant :

 <p>(a)</p>	 <p>(b)</p>
 <p>(c)</p>	 <p>(d)</p>
 <p>(e)</p>	 <p>(f)</p>
 <p>(g)</p>	 <p>(h)</p>

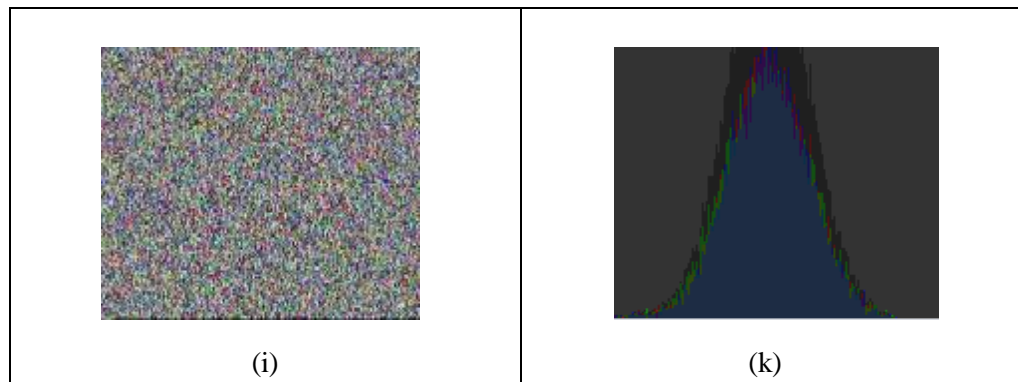


Tableau 4.4 Images avec histogramme

(a) images hôtes, avec histogramme (b), (c) image crypté par: AES [128 bit] avec histogramme (d), (e) image crypté par : DES [56 bit] avec histogramme (f), (g) image crypté par RC4 [120 bit] Avec histogramme (h), (i) image crypté par : Twofish [120 bit] avec histogramme (k).

On peut bien voir que les histogrammes des images chiffrées (pour chaque algorithme) très différents de celui de l'image originale, Les algorithmes de chiffrement utilisé fait en sorte que la dépendance des propriétés statistiques de l'image chiffrée et de l'image originale soit quasi aléatoire. Ceci rend la cryptanalyse de plus en plus difficile.

5.3.Chiffrement des clés

Dans cette section, nous avons cryptée la clé symétrique en utilisant des algorithmes de chiffrement asymétrique différents tel que (RSA, EL GAMAL, ECC).

Pour faire le comparatif de la vitesse de cryptage et le décryptage, nous avons chiffré trois clés de différentes tailles (128, 512, 1024 bit).

🚦 Opérations de chiffrement

	Temps et vitesse de cryptage (chiffrement des clés) en milliseconde				
Taille en bit Algorithme	128 bits	512 bits	1024 bits	Durée totale	Vitesse bit/mi-sec
RSA (512 bits)	7	15	30	52	32
EL GAMAL(512)	4	14	25	43	38.70
ECC (192bit)	0.7	0.8	1	2.5	665.6

Tableau 4.5 Temps de chiffrement des clés (ms) pour des tailles différentes (bit).

✓ Conclusion tirées de ce tableau

- ü RSA et EL GAMEL sont très lents en chiffrement comparativement aux autres algorithmes. On peut considérer ceci comme un inconvénient. Mais sont des algorithmes robuste, leurs clef privée est difficile à casser.
- ü ECC est le meilleur algorithme de chiffrement pour la vitesse.

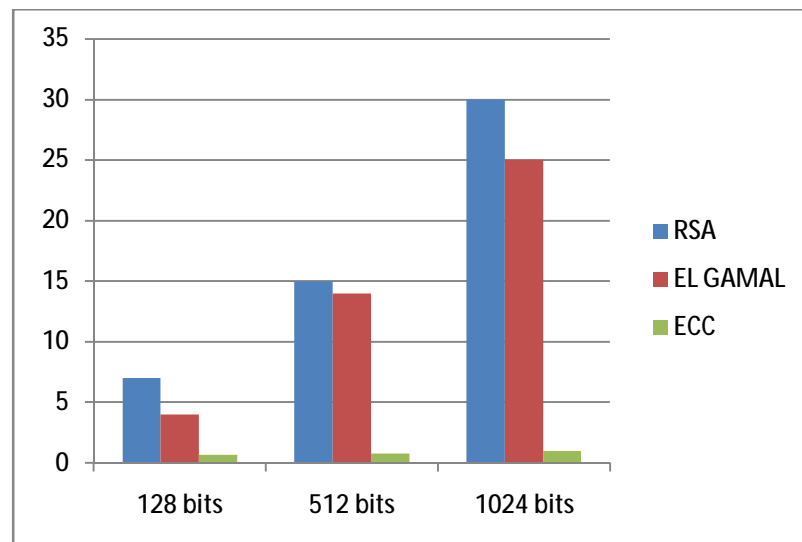


Figure 4.5 Graphe du temps de chiffrement des clés (ms).

✚ Opérations de déchiffrement

Taille en bit Algorithme	Temps et vitesse de décryptage (déchiffrement des clés) en milliseconde.				
	128 bits	512 bits	1024 bits	Durée totale	Vitesse bit/mi-sec
RSA (512 bits)	8	16	29	53	31.40
EL Gamal (512 bits)	3	8	15	26	64
ECC (192bit)	0.65	0.76	1.07	2.48	671

Tableau 4.6 Temps de déchiffrement des clés (ms) pour des tailles différentes (bit)

✓ Conclusion tirées de ce tableau

- ü Le temps de chiffrement de clé est proportionnel à sa taille.
- ü les algorithmes classés par rapidité de déchiffrement décroissante sont comme suit : ECC, EL GAMAL, RSA.

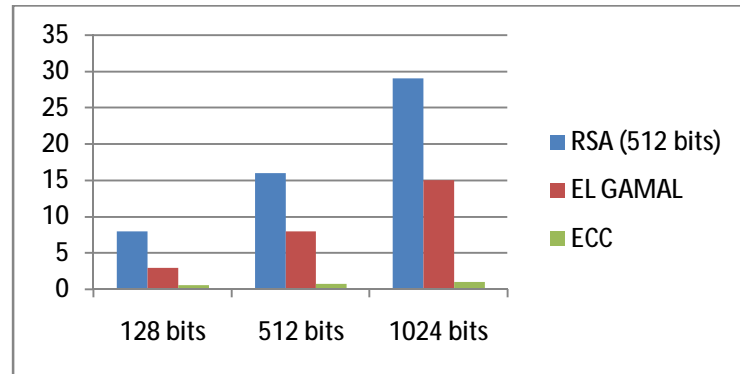


Figure 4.6 Graphe du temps de déchiffrement des clés (ms).

6. Conclusion

Nous avons présenté dans ce chapitre comment implémenter les algorithmes hybride par langages java, nous avons vu aussi pourquoi choisi le langage Java dans ce projet, et quels sont les avantages de ce langage pour le traitement/ Chiffrement des images. A partir de nos résultats expérimentaux, on a fait une étude comparative entre les algorithmes implémentés sur la base de la performance de chiffrement /déchiffrement.