|  |  |
| --- | --- |
|  | **SIGLE DU COURS** |
| Titre du cours |
| ***Série A, B, C ou D, etc., s’il y a lieu*** |

**Travail noté**

Titre du travail (Pondération)

|  |
| --- |
| ■ Remplissez soigneusement cette feuille d’identité.  ■ Rédigez votre travail, en commençant à la page suivante.  ■ Sauvegardez votre travail de cette façon : SIGLEDUCOURS\_TN1\_VOTRENOM.  ■ Utilisez le *Dépôt des travaux* pour acheminer votre travail à votre professeur ou son délégué. <http://www.teluq.ca/mateluq/> |

Feuille d’identité

Nom       Prénom

Numéro d’étudiant       Trimestre

Adresse

      Code postal

Téléphone Domicile       Travail

Cellulaire

Courriel

Nom du professeur ou son délégué

**Réservé à l’usage du professeur ou son délégué**

Date de réception       Date de retour

Note

Date d’envoi

Projet de fin d’études - Implémentation et comparaison des algorithmes de chiffrage asymétrique

Le présent projet de fin d’études vise à montrer l’efficacité et l’efficience de l’implémentation des trois algorithmes de chiffrement et de déchiffrement asymétriques nommés RSA (Ron Rivest, Adi Shamir et Leonard Adleman), El-Gamal et ECC (Elliptic Curve Cryptography), afin de fournir et offrir des solutions à certains problèmes que surviennent en général, lorsque l’on implémente, en un langage de programmation donné (C, C#, C++, Java, Python etc.), ces trois algorithmes déjà mentionnés ci-dessus.

Afin de satisfaire à cet objectif général, global et primaire, Gonzalo Alfredo Romero Francia a décidé de concevoir la planification de la réalisation de toutes les tâches de toutes les étapes de réalisation du projet qu’il a choisi de réaliser, dans le cadre de la réalisation de son cours INF 1430, ce projet est le projet nommé « Implémentation et comparaison des algorithmes de chiffrage asymétrique ».

Afin de réaliser le projet nommé ci-dessus avec efficacité et efficience, et sans subir aucun retard ni aucun problème de réalisation de ce projet qui pourrait compromettre la réussite de la réalisation de ce projet, Gonzalo Alfredo Romero Francia a conçu, avec une énorme facilité, la planification de la réalisation de son projet nommé « Implémentation et comparaison des algorithmes de chiffrage asymétrique » suivante :

* Phase 1: faire des recherches sur Internet, sur les algorithmes de chiffrement asymétrique RSA, El-Gamal et ECC.
* Phase 2: me familiariser avec la théorie et les notions qu'utilisent ces trois algorithmes de chiffrement asymétrique, sous le plan purement théorique (lecture de pages web et visionnement de vidéos).
* Phase 3: me familiariser avec la théorie et les notions qu'utilisent ces trois algorithmes de chiffrement asymétrique, sous le plan purement mathématique et informatique (modélisation des formules et équations utilisées par ces trois algorithmes de chiffrement asymétrique en utilisant le logiciel Maple 2024, que je vais acheter prochainement).
* Phase 4: commencer à implémenter ces trois algorithmes de chiffrement asymétrique, tout en implémentant avec C++, chacune des formules et équations que ces trois algorithmes utilisent, tant pour chiffrer que pour déchiffrer des messages variés.
* Phase 5: Terminer la programmation de l'implémentation des algorithmes RSA, El-Gamal et ECC, pour rendre les 3 programmes C++ en version finale.
* Phase 6: réaliser des mesures de performance à ces trois algorithmes de chiffrement asymétrique, en utilisant plusieurs critères qui seront déterminées par moi prochainement.
* Phase 7: rédiger le document Word nécessaire pour montrer le résultat de l'implémentation de ces 3 algorithmes, ainsi que les résultats obtenus.

Gonzalo Alfredo Romero Francia, l’auteur du présent document, va fournir des informations et des explications sur la forme de réalisation des tâches associées à chacune des phases de la planification de son projet de fin d’études déjà montré ci-dessus. Il a commencé avec la réalisation de toutes les tâches de la phase 1, qui sont les suivantes :

1a. Réalisation de recherches d’informations générales sur la théorie associée à l’algorithme RSA, sans mettre de l’accent sur théorie mathématique associée à cet algorithme.

1b. Réalisation de recherches d’informations générales sur la théorie associée à l’algorithme RSA, tout en mettant l’accent sur théorie mathématique associée à cet algorithme.

1c. Réalisation de recherches de pages web montrant comment réaliser, de manière générale et pas très détaillée, à l’aide d’exemples simples de calcul associés au chiffrement et déchiffrement RSA, les calculs mathématiques permettant d’obtenir les nombres premiers qui permettent de créer les clés publiques et privées utilisées par l’algorithme RSA.

1d. Réalisation de recherches de pages web montrant comment réaliser, de manière générale et pas très détaillée, à l’aide d’exemples complexes de calcul associés au chiffrement et déchiffrement RSA, les calculs mathématiques permettant d’obtenir les nombres premiers qui permettent de créer les clés publiques et privées utilisées par l’algorithme RSA.

1e. Réalisation d’un exercice théorique associé à la création des nombres premiers utilisés pour créer les clés publiques et privées, à la création de clés publiques et privées, et à la réalisation du processus de chiffrement et de déchiffrement de messages clairs et pleins quelconques, pour l’algorithme RSA.

1f. Réalisation de recherches d’informations générales sur la théorie associée à l’algorithme El-Gamal, sans mettre de l’accent sur théorie mathématique associée à cet algorithme.

1g. Réalisation de recherches d’informations générales sur la théorie associée à l’algorithme El-Gamal, tout en mettant l’accent sur théorie mathématique associée à cet algorithme.

1h. Réalisation de recherches de pages web montrant comment réaliser, de manière générale et pas très détaillée, à l’aide d’exemples simples de calcul associés au chiffrement et déchiffrement El-Gamal, les calculs mathématiques permettant d’obtenir les nombres premiers qui permettent de créer les clés publiques et privées utilisées par l’algorithme El-Gamal.

1i. Réalisation de recherches de pages web montrant comment réaliser, de manière générale et pas très détaillée, à l’aide d’exemples complexes de calcul associés au chiffrement et déchiffrement El-Gamal, les calculs mathématiques permettant d’obtenir les nombres premiers qui permettent de créer les clés publiques et privées utilisées par l’algorithme El-Gamal.

1j. Réalisation d’un exercice théorique associé à la création des nombres premiers utilisés pour créer les clés publiques et privées, à la création de clés publiques et privées, et à la réalisation du processus de chiffrement et de déchiffrement de messages clairs et pleins quelconques, pour l’algorithme El-Gamal.

1k. Réalisation de recherches d’informations générales sur la théorie associée à l’algorithme ECC, sans mettre de l’accent sur théorie mathématique associée à cet algorithme.

1l. Réalisation de recherches d’informations générales sur la théorie associée à l’algorithme ECC, tout en mettant l’accent sur théorie mathématique associée à cet algorithme.

1m. Réalisation de recherches de pages web montrant comment réaliser, de manière générale et pas très détaillée, à l’aide d’exemples simples de calcul associés au chiffrement et déchiffrement El-Gamal, les calculs mathématiques permettant d’obtenir les nombres premiers qui permettent de créer les clés publiques et privées utilisées par l’algorithme ECC.

1n. Réalisation de recherches de pages web montrant comment réaliser, de manière générale et pas très détaillée, à l’aide d’exemples complexes de calcul associés au chiffrement et déchiffrement El-Gamal, les calculs mathématiques permettant d’obtenir les nombres premiers qui permettent de créer les clés publiques et privées utilisées par l’algorithme ECC.

1o. Réalisation d’un exercice théorique associé à la création des nombres premiers utilisés pour créer les clés publiques et privées, à la création de clés publiques et privées, et à la réalisation du processus de chiffrement et de déchiffrement de messages clairs et pleins quelconques, pour l’algorithme ECC.

Afin de réaliser toutes les tâches montrées ci-dessus, Gonzalo Alfredo Romero Francia a réalisé des tas de recherches d’informations sur les algorithmes RSA, El-Gamal et ECC, et il a trouvé des tas de pages web lui enseignant tout sur les théories mathématiques associées à ces trois algorithmes de chiffrement et de déchiffrement asymétrique, le voici la bibliographie montrant toutes les pages web qu’il a trouvés sur Internet, et qu’il a utilisés, pour étudier en profondeur toutes les informations qu’il a trouvées sur les algorithmes RSA, El-Gamal et ECC :

Les algorithmes cryptographiques asymétriques, <https://slideplayer.fr/slide/3260205/>, ISIMA 2014.

La cryptographie , <https://www.mcours.net/cours/pdf/hassbg/hassbgli837.pdf>

La Cryptographie à base des Courbes Elliptiques : Étude Comparative, <http://dspace.univ-jijel.dz:8080/xmlui/bitstream/handle/123456789/12836/Inf.ILM.08-22.pdf?sequence=1>

Professeur Taoufik Serraj, Notes de Cours de Cryptographie, <https://espace-fpn.ump.ma/ftp/etudiants/Cours%20PRT%2020-21/CryptoSMA6%20-%20SERRAJ%20Taoufik.pdf>

Wikipédia, Algorithme d'Euclide étendu,

https://fr.wikipedia.org/wiki/Algorithme\_d%27Euclide\_%C3%A9tendu

Wikipédia, Chiffrement RSA, https://fr.wikipedia.org/wiki/Chiffrement\_RSA

Wikipédia, Inverse modulaire, <https://fr.wikipedia.org/wiki/Inverse_modulaire>

|  |  |
| --- | --- |
|  | Cryptographie moderne > Chiffrement par bloc à clé publique RSA > Démonstration 2/2, |

https://www.hds.utc.fr/~wschon/sr06/crypto/RSA4\_2.htm

Exponentiation rapide, <https://www.labri.fr/perso/betrema/deug/poly/exp-rapide.html>

Cryptographie, https://www.labri.fr/perso/betrema/deug/poly/crypto.html

RSA, A. Dragut, Univ. Aix-Marseille Cours de cryptographie Chapitre III, https://pageperso.lis- lab.fr/andreea.dragut/enseignementCLAA/CryptoChap3RSA2012.pdf

Cryptographie par RSA, <https://www.irif.fr/~emiquey/TP/tp5.pdf>

Introduction à la cryptographie - TP 1, RSA,

https://people.irisa.fr/Jeremy.Metairie/teaching/intro\_crypto/rsa.pdf

Cryptographie asymétrique et courbes elliptiques, Culture Math, https://culturemath.ens.fr/thematiques/lycee/cryptographie-asymetrique-et-courbes-elliptiques

RSA, <https://ataraxy.info/Crypto?Chapitres/RSA>

Cryptographie à clef publique Cours 2, https://lvzl.fr/teaching/2020-21/docs/CP-slides-2.pdf

Introduction au chiffrement asymétrique ElGamal,

https://lab.yvesago.net/2015/11/Introduction-au-chiffrement-asymetrique-ElGamal.html

Andreea Dragut (dragut@univmed.fr) Cours de cryptographie Chapitre IV, ELGamal, https://pageperso.lis-lab.fr/andreea.dragut/enseignementCLAA/CryptoChap4ElGamal2012.pdf

Tbowan, ElGamal encryption, <https://www.arsouyes.org/blog/2018/01_ElGamal/>

Anca Nitulescu, Ecole Normale Supérieure, Paris,

<https://www.di.ens.fr/~nitulesc/files/crypto4.pdf>

Le système d'ElGamal, <https://igm.univ-mlv.fr/~jyt/Crypto/5/elgamal.html>

Cécile Pierrot, Chargée de Recherche INRIA Nancy, Introduction `a la Cryptographie 6. Cryptographie asymétrique (2), https://members.loria.fr/CPierrot/papers/Telecom\_cours\_06.pdf

LES TECHNIQUES DE CHIFFREMENT ET LEUR ÉVOLUTION Étude du chiffrement El Gamal, https://www-complexnetworks.lip6.fr/~tarissan/enum/2022/Chiffrement-2022-Guyot.pdf

Cryptosystème de ElGamal, De Wikipedia, l'encyclopédie libre, https://www.wikiwand.com/fr/Cryptosyst%C3%A8me\_de\_ElGamal

Hakima Ould-Slimane, École de technologie supérieure (ÉTS) Département de génie électrique ,Cryptographie à clé publique, https://cours.etsmtl.ca/mgr850/documents/cours/Hiver2014/MGR850\_H14\_Cours-06\_cryptoAsym.pdf

Propriétés du chiffrement ElGamal, https://lab.yvesago.net/2015/12/Proprietes-du-chiffrement-ElGamal.html

[Documentation Crypto M1 MIC 2020-2021](https://webusers.imj-prg.fr/~pascal.molin/cours/crypto/index.html), https://webusers.imj-prg.fr/~pascal.molin/cours/crypto/cours\_logdiscret.html#:~:text=Calcul%20d'indice,g%3D11%20g%3D11.

Wikipédia, Logarithme discret, https://fr.wikipedia.org/wiki/Logarithme\_discret

Chiffre RSA, [Cryptographie](https://www.dcode.fr/liste-outils#cryptography), [Cryptographie Moderne](https://www.dcode.fr/liste-outils#modern_cryptography), [Chiffre RSA](https://www.dcode.fr/chiffre-rsa), <https://www.dcode.fr/chiffre-rsa>

Algorithme d'Euclide Étendu**,** [Mathématiques](https://www.dcode.fr/liste-outils#mathematics), [Arithmétique](https://www.dcode.fr/liste-outils#arithmetics), [Algorithme d'Euclide Étendu](file:///C:\Users\Gonzalo%20Alfredo\Downloads\Algorithme%20d'Euclide%20Étendu), <https://www.dcode.fr/euclide-etendu>

Décomposition en Nombres Premiers, [Mathématiques](https://www.dcode.fr/liste-outils#mathematics), [Arithmétique](https://www.dcode.fr/liste-outils#arithmetics), [Décomposition en Nombres Premiers](https://www.dcode.fr/decomposition-nombres-premiers), <https://www.dcode.fr/decomposition-nombres-premiers>

WIKIMATH, [EL GAMAL](https://cours-info.iut-bm.univ-fcomte.fr/pmwiki/pmwiki.php/Crypto/ElGamal), LE CHIFFREMENT D'EL GAMAL, https://cours-info.iut-bm.univ-fcomte.fr/pmwiki/pmwiki.php/Crypto/ElGamal

Judicaël Courant Lycée La Martinière-Monplaisir, Lyon Conférence Algorithmique et Programmation CIRM Le 6 mai 2016, Application de la complexité en cryptographie, <https://www.cirm-math.fr/ProgWeebly/Renc1446/Courant.pdf>

[SYNETIS](https://www.synetis.com/), [SÉCURITÉ](https://www.synetis.com/category/securite/), [NOTIONS DE CRYPTOLOGIE ET ALGORITHME DE CHIFFREMENT](https://www.synetis.com/notion-de-cryptologie-et-algorithme-de-chiffrement/), <https://www.synetis.com/notion-de-cryptologie-et-algorithme-de-chiffrement/>

Cryptosystème de ElGamal, <https://www.slideshare.net/AyoubSIAHMED/cryptosystme-de-elgamal>

Gabriel Chênevert, Département d’Informatique et Mathématiques Appliquées, ISEN Lille, Association mathématique du Québec, Conférence plénière Dimension algorithmique et chiffrement post-quantique, <https://www.amq.math.ca/wp-content/uploads/bulletin/vol59/no3/05-V2-bull.pdf>

Keeper Security, Qu’est-ce que la cryptographie à courbe elliptique ?,

<https://www.keepersecurity.com/blog/fr/2023/06/07/what-is-elliptic-curve-cryptography/#:~:text=La%20cryptographie%20%C3%A0%20courbe%20elliptique%20(ECC)%20est%20une%20forme%20de,signatures%20num%C3%A9riques%20et%20le%20chiffrement>, 7 juin 2023

Wikipédia, Cryptographie sur les courbes elliptiques,

<https://fr.wikipedia.org/wiki/Cryptographie_sur_les_courbes_elliptiques>

Thibaut Probst, Découvrir les courbes elliptiques en cryptographie, <https://thibautprobst.fr/posts/elliptic-curves/>, 7 mai 2023

Wikipédia, Addition dans le cadre de la théorie des Courbes Elliptiques, https://fr.wikipedia.org/wiki/Addition#Propri.C3.A9t.C3.A9s

Wikipédia, Modular multiplicative inverse,

https://en.wikipedia.org/wiki/Modular\_multiplicative\_inverse

Thibault Jouannic, [Un peu de crypto avec les courbes elliptiques](https://www.miximum.fr/blog/cryptographie-courbes-elliptiques-ecdsa/),

<https://www.miximum.fr/blog/cryptographie-courbes-elliptiques-ecdsa/>, 17 juin 2014

Marie Lesty, Courbes elliptiques : un nouveau virage pour le chiffrement, <https://www.01net.com/actualites/courbes-elliptiques-un-nouveau-virage-pour-le-chiffrement-143980.html>, 26 mars 2001

Simon Blake-Wilson, Daniel R. L. Brown, Certicom Corp, P. Lambert, Cosine Communications, Groupe de travail Réseau, Traduction Claude Brière de L'Isle, Utilisation des algorithmes de cryptographie de courbe elliptique (ECC) dans la syntaxe de message cryptographique (CMS), <http://abcdrfc.free.fr/rfc-f/rfc3278.html#_Toc218773833>, avril 2002

[Tania Martin](https://www.smalsresearch.be/author/martin/), [Smals Research](https://www.smalsresearch.be/), Elliptic Curve Cryptography for dummies 1: introduction, <https://www.smalsresearch.be/elliptic-curve-cryptography-tutoriel1/>, 25 février 2015.

nina.az, Cryptographie sur les courbes elliptiques, https://www.wikidata.fr-fr.nina.az/Cryptographie\_sur\_les\_courbes\_elliptiques.html

[Hervé LEHNING, agrégé de mathématiques](https://blogs.futura-sciences.com/lehning/), [CRYPTOLOGIE](https://blogs.futura-sciences.com/lehning/category/sciences/cryptologie/), [MATHÉMATIQUES](https://blogs.futura-sciences.com/lehning/category/maths-2/)

COMMENT PEUT-ON CHIFFRER AVEC UNE COURBE ?, <https://blogs.futura-sciences.com/lehning/2019/01/11/comment-peut-on-chiffrer-avec-une-courbe/>, 11 janvier 2019

Valentine A**.**, Chercheure chez Gate.io (page web), Cryptographie à courbe elliptique, <https://www.gate.io/fr/blog_detail/782/elliptic-curve-cryptography>.

Tania Martin, Consultante Recherche chez Smals Research, Elliptic Curve Cryptography for dummies 2: en pratique pour la cryptographie, <https://www.smalsresearch.be/elliptic-curve-cryptography-tutoriel2/>, 12 août 2015

Abdelouahid BEN TAMOU, hamza noursaid, Cryptography, Elliptic Curve Cryptography, Rapport sur les cryptosystèmes utilisant les courbes elliptiques en Tifinagh, https://www.academia.edu/11513925/Rapport\_sur\_les\_cryptosyst%C3%A8mes\_utilisant\_les\_courbes\_elliptiques\_en\_Ti\_finagh

INTChain French Community, Cryptographie sur les Courbes Elliptiques, <https://medium.com/int-chain-fr/cryptographie-sur-les-courbes-elliptiques-74878e5905ca>, 30 juin 2019

Engue Gillier, Ingénieur Sécurité chez Facebook,

[Sécurisez vos données avec la cryptographie, Utilisez le chiffrement asymétrique, https://openclassrooms.com/fr/courses/1757741-securisez-vos-donnees-avec-la-cryptographie/6031872-utilisez-le-chiffrement-asymetrique, 3 janvier 2024](file://C:\\Users\\Gonzalo Alfredo\\Downloads\\Sécurisez vos données avec la cryptographie, Utilisez le chiffrement asymétrique, https:\\openclassrooms.com\\fr\\courses\\1757741-securisez-vos-donnees-avec-la-cryptographie\\6031872-utilisez-le-chiffrement-asymetrique, 3 janvier 2024)

AVI Networks, Elliptic Curve Cryptography, Elliptic Curve Cryptography Definition, <https://avinetworks.com/glossary/elliptic-curve-cryptography/>, année 2024

Cédric Murdica. Physical security of elliptic curve cryptography. Cryptography and Security [cs.CR]. Télécom ParisTech, 2014. English. ffNNT : 2014ENST0008ff. fftel-01179584

D. Sravana Kumar, CH. Suneetha, A. ChandrasekhAR, International Journal of Distributed and Parallel Systems (IJDPS) Vol.3, No.1, January 2012 DOI : 10.5121/ijdps.2012.3125 301 ENCRYPTION OF DATA USING ELLIPTIC CURVE OVER FINITE FIELDS, <https://arxiv.org/ftp/arxiv/papers/1202/1202.1895.pdf>, Janvier 2012

n software, IPWorks Encrypt 2020 C++ Edition, ECC Class,

<https://cdn.nsoftware.com/help/IEF/cpp/ECC.htm>

Crypto Stack Exchange Forum, Cryptography, [521-bit ECC keys are the same strength as RSA 15,360-bit keys](https://crypto.stackexchange.com/questions/79944/521-bit-ecc-keys-are-the-same-strength-as-rsa-15-360-bit-keys), <https://crypto.stackexchange.com/questions/79944/521-bit-ecc-keys-are-the-same-strength-as-rsa-15-360-bit-keys>, 2021

Renaud Dumont, Université de Liège Faculté des Sciences Appliquées, Cryptographie et Sécurité informatique INFO0045-2, Notes de cours provisoires 2009 – 2010, https://doc.lagout.org/security/Crypto/2010\_cours\_crypto.pdf

Awouzouba Esso-Essinam, Cours cryptographie, Gran guru Computador, Magister Informatica, <https://www.slideshare.net/KarlAwouzouba/cours-cryptographie>, 1 décembre 2017

[Alegsaonline.com](https://fr.alegsaonline.com/) - [Effet avalanche](https://fr.alegsaonline.com/art/7672) - Leandro Alegsa - 2021-05-09 21:13:44 - url: https://fr.alegsaonline.com/art/7672

Geeks for Geeks page web (<https://www.geeksforgeeks.org>), Avalanche Effect in Cryptography,

<https://www.geeksforgeeks.org/avalanche-effect-in-cryptography/>,14 Mars 2022

Freeman Law page web (<https://freemanlaw.com/>), WHAT YOU NEED TO KNOW ABOUT THE “AVALANCHE” EFFECT, THE CRITERIA OF THE STRICT AVALANCHE EFFECT, <https://freemanlaw.com/what-you-need-to-know-about-the-avalanche-effect/>, année 2022

[Satish Kumar](https://www.tutorialspoint.com/authors/satish-kumar-166980035458), [Computer Network](https://www.tutorialspoint.com/articles/category/Computer-Network), [Digital Signature](https://www.tutorialspoint.com/articles/category/digital-signature), [Cryptography](https://www.tutorialspoint.com/articles/category/cryptography), Avalanche Effect in Cryptography, <https://www.tutorialspoint.com/avalanche-effect-in-cryptography>, 6 Février 2023

Affan Malik, page web educative.io (<https://www.educative.io/>), What is the avalanche effect?, <https://www.educative.io/answers/what-is-the-avalanche-effect>, année 2024

Avec toutes les informations sur les algorithmes de chiffrement et de déchiffrement asymétrique RSA, El-Gamal et ECC, fournies par les pages web trouvés par l’auteur du présent document, et qui sont montrés dans la bibliographie montrée ci-dessus, il a compris en profondeur, jour après jour, la théorie associée à ces trois algorithmes.

De plus, et afin de finir par comprendre, maîtriser et devenir plus qu’habile avec la compréhension de toutes les théories mathématiques associées aux algorithmes de chiffrement et de déchiffrement asymétrique RSA, El-Gamal et ECC, il a utilisé toutes les références bibliographiques fournies par le professeur tuteur de Gonzalo Alfredo Romero Francia, Habib Louafi, ces références bibliographiques sont les suivantes :

1. **Comprendre et implémenter RSA**

1.1 <https://www.askpython.com/python/examples/rsa-algorithm-in-python>

1.2 <https://www.geeksforgeeks.org/rsa-algorithm-cryptography/>

1.3.<https://www.teach.cs.toronto.edu/~csc110y/fall/notes/08-cryptography/05-rsa-cryptosystem-implementation.html>

1. **RSA théorie:**

2.1<https://www.math.ucdavis.edu/~anne/SQ2014/thematic_tutorials/numtheory_rsa.html#:~:text=The%20Rivest%2C%20Shamir%2C%20Adleman%20(,our%20private%20keys%20to%20ourselves>.

2.2 <https://www.cantorsparadise.com/the-rsa-cryptosystem-explained-4396612d763a>

1. Arithmétique modulaire, inverse multiplicatif et autres:

3.1<https://www.jsums.edu/nmeghanathan/files/2015/05/CSC439-Sp2013-1-Number-Theory-and-RSA-Public-Key-Encryption.pdf>

**4. RSA implémentation :**

4.1 [RSA Algorithm: Theory and Implementation in Python - AskPython](https://www.askpython.com/python/examples/rsa-algorithm-in-python)

4.2 [RSA Algorithm in Cryptography - GeeksforGeeks](https://www.geeksforgeeks.org/rsa-algorithm-cryptography/)

4.3 [8.5 Implementing RSA in Python (toronto.edu)](https://www.teach.cs.toronto.edu/~csc110y/fall/notes/08-cryptography/05-rsa-cryptosystem-implementation.html)

1. **EL-GAMAL**

**5.1 Théorie:**

5.1.1 <https://mathstats.uncg.edu/sites/pauli/112/HTML/secelgamal.html>

5.1.2<https://pycryptodome.readthedocs.io/en/latest/src/public_key/elgamal.html#encryption-algorithm>

**5.2 Implémentation:**

5.2.1 <https://www.geeksforgeeks.org/elgamal-encryption-algorithm/>

5.2.2 <https://medium.com/@mghanbari.maxa/the-elgamal-encryption-algorithm-d59a1d9616f>

1. **ECC:**

**6.1 Théorie:**

6.1.1 <https://math.uchicago.edu/~may/REU2020/REUPapers/Shevchuk.pdf>

6.1.2 <https://mschmidt34.math.gatech.edu/files/ecc-summary.pdf>

6.1.3 <https://github.com/nakov/Practical-Cryptography-for-Developers-Book/blob/master/asymmetric-key-ciphers/elliptic-curve-cryptography-ecc.md>

6.1.4 <https://cryptobook.nakov.com/asymmetric-key-ciphers/ecc-encryption-decryption>

**6.2 Implémentation:**

6.2.1 <https://pycryptodome.readthedocs.io/en/latest/src/public_key/ecc.html>

6.2.2 <https://www.youtube.com/watch?v=qW9vYPLzCns>

Une fois qu’il a réussi à obtenir et à organiser toutes les pages web dont les références bibliographiques sont montrées ci-haut, l’auteur du présent document a procédé à passer à la phase 2 de son projet de fin d’études de son cours INF 1430, qui est la phase de familiarisation avec la théorie et les notions qu'utilisent ces trois algorithmes de chiffrement asymétrique, sous le plan purement théorique, plus précisément, réaliser la lecture de pages web et le visionnement de vidéos trouvés à la phase 1, alors Il a continué avec la réalisation de toutes les tâches de la phase 2, qui sont les suivantes :

2a. Lire attentivement chacune des pages web trouvées sur Internet à la phase 1, en lien avec l’algorithme asymétrique RSA.

2b. Visionner attentivement chacune des pages web hébergeant des vidéos trouvées sur Internet à la phase 1, en lien avec l’algorithme asymétrique RSA.

2c. Surligner les informations les plus importantes et les faits saillants en lien avec l’algorithme asymétrique RSA.

2d. Relire attentivement et en profondeur chacune des pages web et chacune des pages web hébergeant des vidéos trouvées sur Internet, et assimiler plus en profondeur toutes les théories et notions sur l’algorithme asymétrique RSA.

2e. Lire attentivement chacune des pages web trouvées sur Internet à la phase 1, en lien avec l’algorithme asymétrique El-Gamal.

2f. Visionner attentivement chacune des pages web hébergeant des vidéos trouvées sur Internet à la phase 1, en lien avec l’algorithme asymétrique El-Gamal.

2g. Surligner les informations les plus importantes et les faits saillants en lien avec l’algorithme asymétrique El-Gamal.

2h. Relire attentivement et en profondeur chacune des pages web et chacune des pages web hébergeant des vidéos trouvées sur Internet, et assimiler plus en profondeur toutes les théories et notions sur l’algorithme asymétrique El-Gamal.

2i. Lire attentivement chacune des pages web trouvées sur Internet à la phase 1, en lien avec l’algorithme asymétrique ECC.

2j. Visionner attentivement chacune des pages web hébergeant des vidéos trouvées sur Internet à la phase 1, en lien avec l’algorithme asymétrique ECC.

2k. Surligner les informations les plus importantes et les faits saillants en lien avec l’algorithme asymétrique ECC.

2l. Relire attentivement et en profondeur chacune des pages web et chacune des pages web hébergeant des vidéos trouvées sur Internet, et assimiler plus en profondeur toutes les théories et notions sur l’algorithme asymétrique ECC.

Une fois qu’il a réussi à lire tous les textes et une fois qu’il a réussi à visionner tous les vidéos de toutes les pages web dont les références bibliographiques sont montrées à la phase 1, l’auteur du présent document a procédé à passer à l’étape 3 de son projet de fin d’études de son cours INF 1430, qui consiste à se familiariser avec la théorie et les notions qu'utilisent les trois algorithmes de chiffrement asymétrique RSA, El-Gamal et ECC, sous le plan purement mathématique et informatique, plus exactement, la modélisation des formules et équations utilisées par ces trois algorithmes de chiffrement asymétrique en utilisant le logiciel Maple 2024, qui a été acheté dernièrement, alors Il a continué avec la réalisation de toutes les tâches de la phase 3, qui sont les suivantes :

3a. Repérage des pages web type document texte, et les pages web type vidéo, provenant des références de la bibliographie créé à la phase 1, afin de prendre des exemples simples et complexes de calculs mathématiques associés au processus de calcul des nombres premiers générant les clés publiques et privées associées à l’algorithme asymétrique RSA.

3b. Repérage des pages web type document texte, et les pages web type vidéo, provenant des références de la bibliographie créé à la phase 1, afin de prendre des exemples simples et complexes de calculs mathématiques associées aux processus d’échange de clés publiques et privées, de chiffrement et de déchiffrement de messages clairs pleins, de l’algorithme asymétrique RSA.

3c. Réalisation des calculs mathématiques, avec luxe de détails, associés au processus de calcul des nombres premiers générant les clés publiques et privées associées à l’algorithme asymétrique RSA, en utilisant papier, crayon et calculatrice TI Nspire CX CAS 2 et/ou le logiciel Maple 2023 (23.2).

3d. Réalisation des calculs mathématiques, avec luxe de détails, associées aux processus d’échange de clés publiques et privées, de chiffrement et de déchiffrement de messages clairs pleins, de l’algorithme asymétrique RSA, en utilisant papier, crayon et calculatrice TI Nspire CX CAS 2 et/ou le logiciel Maple 2023 (23.2).

3e. Repérage des pages web type document texte, et les pages web type vidéo, provenant des références de la bibliographie créé à la phase 1, afin de prendre des exemples simples et complexes de calculs mathématiques associés au processus de calcul des nombres premiers générant les clés publiques et privées associées à l’algorithme asymétrique El-Gamal.

3f. Repérage des pages web type document texte, et les pages web type vidéo, provenant des références de la bibliographie créé à la phase 1, afin de prendre des exemples simples et complexes de calculs mathématiques associées aux processus d’échange de clés publiques et privées, de chiffrement et de déchiffrement de messages clairs pleins, de l’algorithme asymétrique El-Gamal.

3g. Réalisation des calculs mathématiques, avec luxe de détails, associés au processus de calcul des nombres premiers générant les clés publiques et privées associées à l’algorithme asymétrique El-Gamal, en utilisant papier, crayon et calculatrice TI Nspire CX CAS 2 et/ou le logiciel Maple 2023 (23.2).

3h. Réalisation des calculs mathématiques, avec luxe de détails, associées aux processus d’échange de clés publiques et privées, de chiffrement et de déchiffrement de messages clairs pleins, de l’algorithme asymétrique El-Gamal, en utilisant papier, crayon et calculatrice TI Nspire CX CAS 2 et/ou le logiciel Maple 2023 (23.2).

3i. Repérage des pages web type document texte, et les pages web type vidéo, provenant des références de la bibliographie créé à la phase 1, afin de prendre des exemples simples et complexes de calculs mathématiques associés au processus de calcul des nombres premiers générant les clés publiques et privées associées à l’algorithme asymétrique ECC.

3j. Repérage des pages web type document texte, et les pages web type vidéo, provenant des références de la bibliographie créé à la phase 1, afin de prendre des exemples simples et complexes de calculs mathématiques associées aux processus d’échange de clés publiques et privées, de chiffrement et de déchiffrement de messages clairs pleins, de l’algorithme asymétrique ECC.

3k. Réalisation des calculs mathématiques, avec luxe de détails, associés au processus de calcul des nombres premiers générant les clés publiques et privées associées à l’algorithme asymétrique ECC, en utilisant papier, crayon et calculatrice TI Nspire CX CAS 2 et/ou le logiciel Maple 2023 (23.2).

3l. Réalisation des calculs mathématiques, avec luxe de détails, associées aux processus d’échange de clés publiques et privées, de chiffrement et de déchiffrement de messages clairs pleins, de l’algorithme asymétrique ECC, en utilisant papier, crayon et calculatrice TI Nspire CX CAS 2 et/ou le logiciel Maple 2023 (23.2).

Cependant, et afin de mieux comprendre comment fonctionnent les algorithmes asymétriques RSA, El-Gamal et ECC. Il a décidé de se familiariser en profondeur avec les théories mathématiques impliquées et associées à tous les processus concernant ces trois algorithmes de chiffrement et de déchiffrement asymétriques, qui sont : la génération de nombres premiers permettant de créer des clés publiques et privées, la création de ces deux types de clés, le chiffrement et le déchiffrement de messages pleins.

Afin de réussir à se familiariser en profondeur avec toutes les théories mathématiques associées aux algorithmes asymétriques RSA, El-Gamal et ECC, il a essayé, au tout début du cours INF 1430, de réaliser lui-même les calculs mathématiques qui sont utilisées pour réaliser, mathématiquement et à la main et avec seulement une calculatrice TI Nspire CX CAS 2, tous les processus concernant ces trois algorithmes de chiffrement et de déchiffrement asymétriques, qui sont : la génération de nombres premiers permettant de créer des clés publiques et privées, la création de ces deux types de clés, le chiffrement et le déchiffrement de messages pleins.

Cependant, quand il n’a pas réussi à réaliser tous ces processus associés aux algorithmes RSA, El-Gamal et ECC, il s’est vite rendu compte que la tâche mathématique de génération de nombres premiers aléatoires permettant de créer des clés publiques et privées, est très complexe mathématiquement, car ce processus génère, crée et manipule des nombres entiers premiers gigantesques, tellement qu’il est devenu impossible, pour l’auteur du présent document, de réaliser seulement cette tâche de génération de ces nombres premiers aléatoires.

Alors, et afin de comprendre comment fonctionnent les algorithmes asymétriques RSA, El-Gamal et ECC, pour les processus de génération de nombres premiers permettant de créer des clés publiques et privées, celui de la création de ces deux types de clés, celui du chiffrement et celui du déchiffrement de messages pleins, il a décidé d’acheter un logiciel de calcul mathématique très complexe, performant, efficace et facile à utiliser nommé Maple 2023.

Alors, en utilisant ce logiciel, il a réussi à réaliser des exercices complets d’implémentation théorique et mathématique des algorithmes RSA, El-Gamal et ECC, et cela sans problème, car il a déjà utilisé le logiciel Maple auparavant, il a commencé à utiliser le logiciel Maple depuis l’année 2003, alors il a déjà maîtrisé la programmation avec Maple, tant les versions antérieures que la dernière version de ce logiciel, qui est la version 23.2.

Maintenant qu’il a réussi de se familiariser avec la théorie associé aux algorithmes de chiffrement et de déchiffrement asymétriques RSA, El-Gamal et ECC, il a procédé à passer à la phase 3 de son projet de fin d’études, qui est la phase de familiarisation avec la théorie et les notions qu'utilisent ces trois algorithmes de chiffrement asymétrique, sous le plan purement mathématique et informatique (modélisation des formules et équations utilisées par ces trois algorithmes de chiffrement asymétrique en utilisant le logiciel Maple 2024, que je vais acheter prochainement), alors Il a continué avec la réalisation de toutes les tâches de la phase 4, qui sont les suivantes :

4a. Faire des recherches d’informations permettant de trouver des pages web montrant, avec luxe de détails, comment implémenter les calculs mathématiques associés aux processus de génération de nombres premiers aléatoires permettant de générer les clés publiques et privées, au processus de création des clés publiques et privées, aux processus d’authentification de clés publiques et privées et/ou d’échange de ces clés, et aux processus de chiffrement et déchiffrement, ces processus étant associés à l’algorithme asymétrique RSA.

4b. Utiliser les références bibliographiques fournies par le professeur tuteur du cours INF 1430, qui est la personne qui encadre Gonzalo Alfredo Romero Francia, l’auteur du présent document, à réaliser son projet de fin d’études nommé «  Implémentation et comparaison des algorithmes de chiffrage asymétrique », afin de mieux comprendre comment réaliser l’implémentation des processus de génération de nombres premiers aléatoires permettant de générer les clés publiques et privées, des processus de création des clés publiques et privées, des processus d’authentification de clés publiques et privées et/ou d’échange de ces clés, et des processus de chiffrement et déchiffrement, ces processus étant associés à l’algorithme asymétrique RSA.

4c. Faire un copier-coller aux codes python trouvés sur Internet, implémentant les algorithmes de chiffrement et de déchiffrement asymétriques RSA, El-Gamal et ECC, et faire un copier-coller aux codes python fournis par le professeur tuteur du cours INF 1430 de Gonzalo Alfredo Romero Francia, afin de regarder le fonctionnement global et sommaire du code python de chacune de ces trois algorithmes de chiffrement et de déchiffrement asymétriques.

4d. Déterminer, parmi tous les codes python trouvés sur Internet et parmi ceux fournis par le professeur tuteur du cours INF 1430 de Gonzalo Alfredo Romero Francia, les trois codes python de base, qui serviront comme base à la programmation et/ou modification/amélioration de ces trois codes python de base.

4e.Reprendre tous les exercices mathématiques associés aux processus de génération de nombres aléatoires permettant la création des clés publiques et privées, aux processus de création des clés publiques et privées, aux processus d’authentification et/ou échange de ces clés publiques et privées, et aux processus de chiffrement et de déchiffrement de messages clairs et pleins, associées aux algorithmes asymétriques RSA, El-Gamal et ECC, trouvés par les références bibliographiques trouvées à la phase 1 du projet de fin d’études de Gonzalo Alfredo Romero Francia, et lues, analysées, comprises, apprises, assimilées et familiarisées avec, lors de la réalisation des phases 2 et 3 de son projet de fin d’études, afin de réaliser, avec les codes python choisie comme les codes python de base, tous les processus déjà mentionnés, en utilisant le langage de programmation python.

4f. Lire attentivement chaque ligne de code python, de chacun des codes python de base des algorithmes asymétriques RSA, El-Gamal et ECC, choisis à l’étape 4d, afin de comprendre plus en profondeur, comment ces codes python de base ont été codés, et afin de comprendre comment ces codes python ont réussi à implémenter efficacement chacun de ces algorithmes de chiffrement et de déchiffrement asymétriques, tout en compilant et en exécutant chacun de ces trois codes python de base.

4g. Faire des centaines des tests de fonctionnement de chacun de ces trois codes python implémentant les algorithmes de chiffrement et de déchiffrement asymétriques RSA, El-Gamal et ECC, afin de voir le fonctionnement global et sommaire que fournissent ces trois codes python de base, ainsi que pour voir et analyser en profondeur la sortie finale que donnent ces trois codes python de base, implémentant ces algorithmes de chiffrement et de déchiffrement asymétriques.

4h. Réaliser quelques petites modifications à quelques lignes de code python associés aux trois codes python de base, implémentant les algorithmes de chiffrement et de déchiffrement asymétriques RSA, El-Gamal et ECC, ces lignes de code sont associées au processus de génération des nombres premiers utilisés pour créer les clés publiques et privées de ces trois algorithmes asymétriques, afin de voir le fonctionnement global et sommaire que fournissent ces trois codes python de base, ainsi que pour voir et analyser en profondeur la sortie finale que donnent ces trois codes python de base, implémentant ces algorithmes de chiffrement et de déchiffrement asymétriques.

4i. Faire des copier-coller aux variables stockant les nombres premiers générés par les algorithmes de chiffrement et de déchiffrement asymétriques RSA, El-Gamal et ECC, permettant de créer les clés publiques et privées associées aux algorithmes de chiffrement et de déchiffrement asymétriques RSA, El-Gamal et ECC, afin de déterminer la taille en nombre de bits, du nombre premier qui a généré ces clés publiques et privées associées à ces algorithmes de chiffrement et de déchiffrement asymétriques.

4j. Faire des copier-coller aux caractères des messages clairs et pleins utilisés, pour tester le fonctionnement des trois codes python de base modifiés un peu à l’étape 4h, implémentant les algorithmes de chiffrement et de déchiffrement asymétriques RSA, El-Gamal et ECC, ces lignes de code sont associées au processus de génération des nombres premiers utilisés pour créer les clés publiques et privées de ces trois algorithmes asymétriques, afin de déterminer la taille en nombre de bits, de ces messages clairs et pleins.

4k. Réaliser des tests de fonctionnalité sur les trois codes python modifiés, associés aux trois codes python de base, implémentant les algorithmes de chiffrement et de déchiffrement asymétriques RSA, El-Gamal et ECC, dont les lignes de code python ont été modifiées à l’étape 4h, afin de regarder attentivement la sortie fournie par ces trois codes python modifiés, dans le but de réaliser avec une forte attention et de manière analytique, les étapes 4i à 4j, afin de vérifier le bon fonctionnement de ces trois codes python modifiés.

4l. Réaliser quelques modifications plus que significatives à des blocs de lignes de code python associés aux trois codes python de base, implémentant les algorithmes de chiffrement et de déchiffrement asymétriques RSA, El-Gamal et ECC, ces lignes de code sont associées au processus de génération des nombres premiers utilisés pour créer les clés publiques et privées de ces trois algorithmes asymétriques, afin de voir le fonctionnement global et sommaire que fournissent ces trois codes python de base, ainsi que pour voir et analyser en profondeur la sortie finale que donnent ces trois codes python de base, implémentant ces algorithmes de chiffrement et de déchiffrement asymétriques.

4m. Faire des copier-coller aux variables stockant les nombres premiers générés par les algorithmes de chiffrement et de déchiffrement asymétriques RSA, El-Gamal et ECC, permettant de créer les clés publiques et privées associées aux algorithmes de chiffrement et de déchiffrement asymétriques RSA, El-Gamal et ECC, afin de déterminer la taille en nombre de bits, du nombre premier qui a généré ces clés publiques et privées associées à ces algorithmes de chiffrement et de déchiffrement asymétriques.

4n. Faire des copier-coller aux caractères des messages clairs et pleins utilisés, pour tester le fonctionnement des trois codes python de base modifiés un peu à l’étape 4l, implémentant les algorithmes de chiffrement et de déchiffrement asymétriques RSA, El-Gamal et ECC, ces lignes de code sont associées au processus de génération des nombres premiers utilisés pour créer les clés publiques et privées de ces trois algorithmes asymétriques, afin de déterminer la taille en nombre de bits, de ces messages clairs et pleins.

4o. Réaliser des tests de fonctionnalité sur les trois codes python modifiés, associés aux trois codes python de base, implémentant les algorithmes de chiffrement et de déchiffrement asymétriques RSA, El-Gamal et ECC, dont les lignes de code python ont été modifiées à l’étape 4l, afin de regarder attentivement la sortie fournie par ces trois codes python modifiés, dans le but de réaliser avec une forte attention et de manière analytique, les étapes 4i à 4j, afin de vérifier le bon fonctionnement de ces trois codes python modifiés.

Après avoir réalisé toutes les étapes montrées ci-dessus, Gonzalo Alfredo Romero Francia a obtenu une infinité d’informations en lien avec tous les calculs mathématiques associés aux algorithmes de chiffrement et de déchiffrement asymétrique RSA, El-Gamal et ECC, ainsi qu’il a obtenu une infinité d’informations en lien avec l’implémentation, en langage de programmation python, de ces trois algorithmes de chiffrement et déchiffrement asymétriques, notamment en lien avec tous les calculs mathématiques associés aux processus de génération de nombres premier aléatoires permettant de créer des clés publiques et privées, au processus de création de clés publiques et privées, ainsi qu’au processus de chiffrement et de déchiffrement de messages clairs et pleins. Les voici ces informations que l’auteur du présent a trouvées sur Internet, en lien avec les points déjà abordés ci-dessus :

Tout d’abord, on commence avec la détermination des critères d’évaluation de la performance des trois codes python associés aux algorithmes de chiffrement et de déchiffrement asymétrique RSA, El-Gamal et ECC, afin de déterminer l’efficacité de compilation et de fonctionnement de ces trois codes python implémentant ces trois algorithmes de chiffrement et de déchiffrement asymétrique. Les voici ces 9 critères d’évaluation du fonctionnement de ces trois codes python déjà décrites ci-dessus :

1. Temps d'exécution.
2. Utilisation de la mémoire.
3. Taille des clés générées.
4. Complexité algorithmique.
5. Gestion des grandes quantités de données.
6. Comparaison des tailles de clés.
7. Comparaison des temps d'exécution avec différentes tailles de clés.
8. Comparaison de la taille des messages clairs utilisés, pour bien sécuriser les messages cryptés.
9. L'effet avalanche.

Le voici une courte explication sur comment chacun de ces critères d’évaluation du fonctionnement de ces trois codes python déjà décrites ci-dessus, va évaluer ces trois codes python, en termes ou critères d’évaluation différents :

1. **Temps d'exécution :** Ce critère d’évaluation mesure la durée nécessaire à l'exécution du code Python. Un temps d'exécution plus court est généralement préférable car il indique une meilleure efficacité du code. Cependant, cela dépend du contexte d'utilisation et des exigences de performance spécifiques.
2. **Utilisation de la mémoire :** Ce critère d’évaluation mesure la quantité de mémoire (RAM) utilisée par le code Python pendant son exécution. Une utilisation de mémoire plus faible est généralement préférable car elle permet d'optimiser les ressources système et peut améliorer les performances globales du système.
3. **Taille des clés générées :** Pour les algorithmes de cryptographie, notamment RSA, ce critère d’évaluation fait référence à la longueur des clés générées. Une clé plus longue est généralement considérée comme plus sécurisée car elle rend plus difficile pour les attaquants de casser le cryptage en force brute. Cependant, une clé plus longue peut également entraîner des temps de calcul plus longs et une utilisation de mémoire accrue.
4. **Complexité algorithmique :** Ce critère d’évaluation fait référence à la mesure de la complexité d'un algorithme, généralement en termes de temps ou d'espace. Il peut s'agir de la complexité temporelle (temps d'exécution en fonction de la taille de l'entrée) ou de la complexité spatiale (utilisation de la mémoire en fonction de la taille de l'entrée). Une complexité algorithmique plus faible est généralement préférable car elle indique une meilleure efficacité de l'algorithme.
5. **Gestion des grandes quantités de données :** Ce critère d’évaluation évalue la capacité du code Python à gérer efficacement de grandes quantités de données en termes de temps d'exécution et d'utilisation de la mémoire. Un bon code doit être capable de gérer efficacement les données sans sacrifier les performances ou la stabilité du système.
6. **Comparaison des tailles de clés :** Ce critère d’évaluation compare les tailles des clés générées par différents algorithmes de cryptographie. Il peut être utilisé pour évaluer la sécurité relative des différents algorithmes en fonction de la longueur de la clé.
7. **Comparaison des temps d'exécution avec différentes tailles de clés :** Ce critère d’évaluation compare les temps d'exécution des algorithmes de cryptographie avec des tailles de clés différentes. Cela permet de déterminer l'impact de la longueur de la clé sur les performances de l'algorithme.
8. **Comparaison de la taille des messages clairs utilisés :** Ce critère d’évaluation compare la taille des messages en texte brut (non cryptés) utilisés en entrée pour le chiffrement. Cela peut être important pour évaluer l'efficacité du cryptage pour différents types de données et pour déterminer les limites de taille acceptables pour les messages clairs.
9. **L'effet avalanche :** En cryptographie, l'effet avalanche fait référence à la propriété d'un algorithme de cryptage où un petit changement dans l'entrée (par exemple, un seul bit modifié dans le message d'entrée) entraîne un changement significatif dans la sortie (par exemple, de nombreux bits modifiés dans le message crypté). Cela garantit que de petites modifications dans les données d'entrée conduisent à des résultats radicalement différents, ce qui est essentiel pour la sécurité des algorithmes de cryptage.

Afin de bien réaliser la programmation de chacun des critères d’évaluation de la performance informatique des trois codes python modifiés implémentant les algorithmes de chiffrement et de déchiffrement RSA, El-Gamal et ECC, Gonzalo Alfredo Romero Francia a réussi à trouver, sur Internet, des informations en lien avec la réalisation de plusieurs types de tests de fonctionnement logique et informatique de codes écrits en langages de programmation divers, comme le langage de programmation python, les voici ces informations en lien avec la réalisation de tests de fonctionnement de codes informatiques, qui ont permis à l’auteur du présent document de réussir à réaliser des tests de fonctionnement logique et informatique à ces trois codes python, implémentant les algorithmes asymétriques RSA, El-Gamal et ECC :

1. **Temps d'exécution :**

Mesurer le temps nécessaire pour effectuer le chiffrement et le déchiffrement pour chaque algorithme.

1. **Utilisation de la mémoire :**

Mesurer l'utilisation de la mémoire pendant l'exécution de vos codes.

Utiliser des outils informatiques et logiciels pour le langage python, tels que memory profiler pour obtenir des informations détaillées.

1. **Taille des clés générées :**

Mesurer la taille des clés générées par chaque algorithme.

1. **Complexité algorithmique :**

Analyser la complexité algorithmique (notation big-O ou Omicron) pour le chiffrement et le déchiffrement de chaque algorithme asymétrique (RSA, El-Gamal et ECC).

1. **Optimisations Python :**

Explorer et implémenter des optimisations spécifiques à Python pour améliorer les performances.

1. **Gestion des grandes quantités de données :**

Évaluer la performance lors du traitement de grandes quantités de données.

1. **Comparaison des tailles de clés :**

Comparer la taille des clés générées par chaque algorithme pour une même longueur de clé (par exemple, 1024 bits, 2048 bits etc).

1. **Comparaison des temps d'exécution avec différentes tailles de clés :**

Mesurer le temps d'exécution avec différentes tailles de clés pour évaluer l'impact de la taille des clés sur les performances.

1. **Stabilité et fiabilité :**

Vérifier la stabilité et la fiabilité de vos codes sur plusieurs exécutions.

1. **Tests unitaires :**

Mettez en place des tests unitaires pour s'assurer de la fiabilité des différentes parties de votre code.

1. **Maintenance et lisibilité du code :**

Évaluez la facilité de maintenance et de compréhension du code source.

1. **Documentation :**

S’assurer que le code est bien documenté pour faciliter sa compréhension et sa maintenance.

1. **Évolutivité :**

Évaluer la capacité des trois codes python (RSA, El-Gamal et ECC) à gérer des charges de travail plus importantes.

1. **Répétabilité des Tests :**

S’assurer que les tests logiciels et informatiques sont reproductibles et fournissent des résultats cohérents.

En réalisant tous ces types de test de fonctionnalité logique et informatiques, Gonzalo Alfredo Romero Francia a finir par apprendre, comprendre, assimiler et maîtriser toutes les théories et notions mathématiques en lien avec las algorithmes de chiffrement et de déchiffrement asymétrique RSA, El-Gamal et ECC.

Nonobstant, pendant la réalisation toutes les étapes de la phase 4, Gonzalo Alfredo Romero Francia a obtenu, appris, compris, assimilé et maîtrisé les informations suivantes, qui sont des faits saillants en lien avec la génération de clés publiques et privées, ainsi que sur l'encryptage et le décryptage de messages clairs, en utilisant les algorithmes asymétriques RSA, El-Gamal et ECC (Version hybride AES-GMC):

1. **Pour que le processus de création/génération de clés publiques et privées, et celui de l'encryptage et de décryptage de messages clairs, soit efficace avec l'algorithme RSA et El-Gamal, il faut satisfaire les points suivants:**

1.1 La taille du message clair, en nombre de caractères et le nombre de bits total qu'occupe ce message, soit suffisamment énorme pour provoquer la génération d'une clé publique gigantesque, dont le nombre de type exposant qui l'a généré (valeur de la clé publique appliquée au logarithme base 2) soit d'au moins 2048 bits.

1.2 Le message clair doit être suffisamment long pour générer un nombre premier exposant d'au moins 2048 bits, qui va générer une clé privée de taille énorme, qui garantira la sécurité de ce message clair.

Exemple:

Pour le message suivant : "This is my very first plain message, that must be first encrypted and then decrypted by the assymetric El-Gamal algorithm!", les algorithmes RSA et El-Gamal ont généré un nombre premier gigantesque, dont le nombre de type exposant a une valeur de 293 bits, ce qui est insuffisant pour garantir une sécurité optimale aux processus d'encryptage et de décryptage de message clairs.

1.3 Pour obtenir des seuils de sécurité plus significatifs, il faut utiliser un message encore plus long que le message "This is my very first plain message, that must be first encrypted and then decrypted by the assymetric El-Gamal algorithm!", qui va générer un nombre premier colossal, dont le nombre de type exposant qui l'a généré ait une valeur d'au moins 2048 bits.

Ceci ayant été dit, et dans le cas de l'exemple montré ci-dessus, il faudrait fournir en total, 2048/293=6.98976=7 lignes de même longueur que le message "This is my very first plain message, that must be first encrypted and then decrypted by the assymetric El-Gamal algorithm!".

Cela veut dire qu'il faut fournir 6 autres lignes de message clair de la même longueur que le message montré ci-dessus, afin d'implémenter efficacement l'algorithme RSA, avec un message clair 7 fois plus long que le message "This is my very first plain message, that must be first encrypted and then decrypted by the assymetric El-Gamal algorithm!"!

**2. Points saillants en lien avec l’algorithme de chiffrement et de déchiffrement asymétrique RSA :**

2.1 La taille en bits du message encrypté dépend de deux facteurs qui sont les suivants :

* La valeur de la clé privée et le nombre de chiffres que contient cette clé.
* La valeur du message clair et plein et le nombre de lettres que contient ce message.

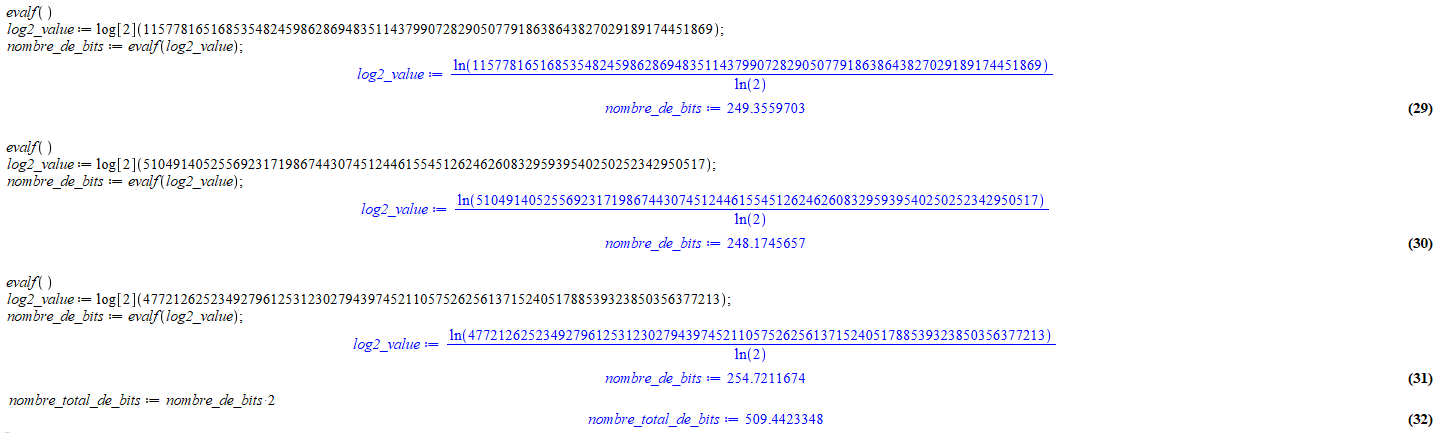
2.2 La taille en bits du bloc de données doit être à peu près égale à la taille en bits des clés publique et privée . Pour que cette affirmation soit vraie, il faut que la clé privée générée par l’algorithme RSA soit un nombre à 4 chiffres.

Par exemple : Pour encrypter un message plein et clair de 512 bits, et obtenir un message crypté de 512 bits, il faut générer une clé privée d’au plus 512 bits.

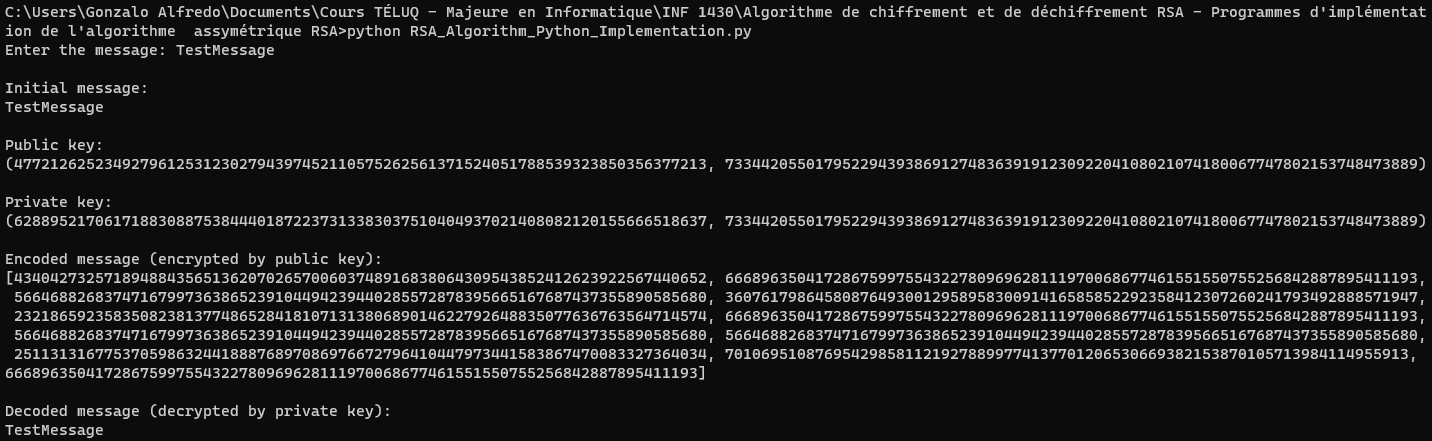
Le voici la démonstration de la validité du point saillant 2.2 :







Et le voici la sortie du programme python codé et modifié par Gonzalo Alfredo Romero Francia, pour l’algorithme RSA :



En lisant attentivement toutes le figures montrées aux pages 20 et 21 du présent document, on remarque les faits suivants :

Les clés publiques et privées crées par l’algorithme RSA sont représentés sous forme de pair de nombres, alors cet algorithme crée efficacement deux clés : une clé publique représenté sous la forme de la paire de nombres entiers premiers (e,n) et une clé privée représenté sous la forme de la paire de nombres entiers premiers (d,n), chacune ayant une taille de valeur très variable, car l’algorithme RSA utilise des fonctions mathématiques qui génèrent des nombres premiers de taille colossale différents, de valeurs différentes.

Dans le cas de l’exemple de calcul montré à la page 21 du présent document, la taille en nombre de bits, du nombre entier premier e, est de 255 bits (254.72), alors chacune de ces clés publique et privée ont une taille totale de 255\*2=510 bits, **taille qui se rapproche trop de la valeur théorique de la taille en bits de ces clés publique et privée,** qui est calculé a 256\*2= 512 bits.

Les deux points nommés ci-dessus permettent de conclure que l’algorithme RSA, dans le cas de l’exemple de calcul montré à la page 21 du présent document, cet algorithme va générer des nombres entiers et premiers de taille colossale, de taille en nombre de bits de 255 bits, et cela pour chaque caractère du message clair et plein nommé « TestMessage » qui contient 11 caractères.

Alors, et en guise de conclusion, dans le cas de l’exemple de calcul montré à la page 21 du présent document, cet algorithme va générer des nombres entiers et premiers de taille colossale, de taille en nombre de bits de 255 bits, et cela pour chaque caractère du message clair et plein nommé « TestMessage » qui contient 11 caractères. Cela veut dire que le message encrypté aura une taille colossale et gigantesque, dont sa taille est la taille de 11 nombres entiers et premiers de taille colossale et gigantesque de 255 bits, alors cela donne : 255bits\*11=2805 bits pour représenter et afficher le message encrypté associé au message « TestMessage »

**3. Points saillants en lien avec l’algorithme de chiffrement et de déchiffrement asymétrique El-Gamal :**

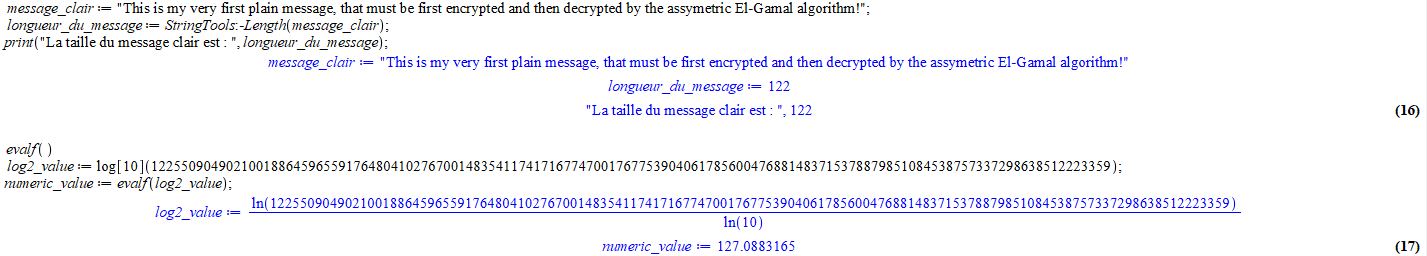
3.1 La taille en bits du message encrypté dépend de deux facteurs qui sont les suivants :

* La valeur de la clé privée et le nombre de chiffres que contient cette clé.
* La valeur du message clair et plein et le nombre de lettres que contient ce message.

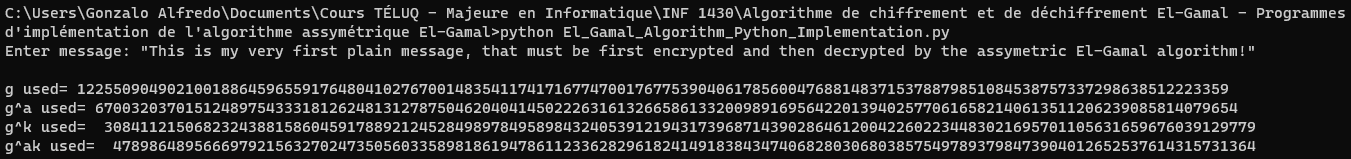
3.2 La taille en bits du bloc de données doit être supérieure à la taille en bits des clés publique et privée, **ou au moins à peu près de la même taille de ces clés**. Pour que cette affirmation soit vraie, il faut que la clé privée générée par l’algorithme El-Gamal soit un nombre gigantesque dont le nombre de bits qui le génère soit supérieure au nombre de caractères du message clair et plein, **ou à peu près égale au nombre de caractères de ce message.**

Par exemple : Pour encrypter un message plein et clair de 122 caractères (8\*122=976 bits), il faut générer un nombre gigantesque, dont le nombre de bits pour le générer est au moins 128 bits ), la taille de ce nombre gigantesque est de 128\*8=1024 bits.

Le voici la démonstration de la validité du point saillant 3.2 :



Et le voici la sortie du programme python codé et modifié par Gonzalo Alfredo Romero Francia, pour l’algorithme RSA :



En regardant attentivement aux figures montrés ci-dessus, on peut rapidement s’apercevoir que l’affirmation de Gonzalo Alfredo Romero Francia, associée au point saillant 3.2 déjà expliqué à la page 22 du présent document, est très correcte, car l’exemple de compilation de son code python implémentant l’algorithme El-Gamal, appliqué à cet exemple, donne comme sorties des nombres entiers premiers gigantesques et colossales permettant de générer le clés publiques et privées montrés ci-dessus, des valeurs gigantesques dont leur taille réelle et expérimentale est très proche de la taille théorique calculée pour l’exemple déjà nommé ci-haut.

Plus exactement, cet exemple consiste en encrypter et décrypter le message « This is my very first plain message, that must be first encrypted and then decrypted by the assymetric El-Gamal algorithm! », un message texte contenant 122 caractères avec les espaces, ce qui est affiché par le logiciel Maple 2023 (23.2). Le nombre entier et premier généré par l’algorithme El-Gamal, pour réaliser le processus de création des clés publique et privée que fournit cet algorithme, est un nombre gigantesque et colossale dont sa valeur est le nombre entier et premier suivant :

12255090490210018864596559176480410276700148354117417167747001767753904061785600476881483715378879851084538757337298638512223359

Ce nombre a été généré par un nombre entier et premier gigantesque dont la taille est, calculée en théorie par un nombre aléatoire gigantesque d’une résolution de 128 bits, alors ce nombre entier premier aléatoire possède une valeur aléatoire si gigantesque, qu’il nécessite une représentation de 128 bits pour afficher ce nombre. Maintenant, la sortie que donne la code python implémentant l’Algorithme El-Gamal de l’auteur du présent document a donné, comme nombre aléatoire gigantesque, permettant de créer les clés publique et privée montrées dans la deuxième figure, de haut en bas, montrée à page 23 du présent document, le nombre entier premier déjà affiché à la même page.

Afin de calculer la résolution en nombre de bits que possède la représentation du nombre entier et premier 12255090490210018864596559176480410276700148354117417167747001767753904061785600476881483715378879851084538757337298638512223359, Gonzalo Alfredo Romero Francia a réussi a coder une fonction Maple qui lui donne la résolution, en nombre de bits, que donne ce nombre entier et premier gigantesque, ce nombre est affiché par la première figure montré, de haut en bas, à la page 23 du présent document, alors le code python implémentant l’algorithme El-Gamal, codé par l’auteur du présent document, a donné une résolution expérimentale de 127.0883≈127 bits, une valeur de résolution en bits **très proche** de la valeur théorique calculé pour la résolution du nombre entier et premier aléatoire permettant de créer les clés publique et privée déjà montré à la première figure montré de la page 23 du présent document, de haut en bas, ces deux clés sont les suivantes :

g^a créé :

6700320370151248975433318126248131278750462040414502226316132665861332009891695642201394025770616582140613511206239085814079654

g^k créé :

30841121506823243881586045917889212452849897849589843240539121943173968714390286461200422602234483021695701105631659676039129779

Alors, et pour conclure, le point saillant découvert, appris, compris, assimilé, maîtrisé et démontré aux pages 23 et 24 du présent document, a été correctement démontré, alors, en ce qui a trait l’algorithme El-Gamal, la taille des clés publiques et privées, en nombre de bits, est directement associé à la taille, en nombre de bits du nombre entier et premier aléatoire qui les génère, mais n’a nécessairement pas une association directe avec la taille, en nombre de bits, du message ou bloc de données qui est utilisé pour encrypter et décrypter ce message, avec l’algorithme de chiffrement et de déchiffrement asymétrique El-Gamal.

**4. Points saillants en lien avec l’algorithme de chiffrement et de déchiffrement asymétrique ECC :**

Avec l’algorithme de chiffrement et de déchiffrement asymétrique ECC, c’est normal d’obtenir deux clés publiques (cyphertext pubKey) différents, car avec cet algorithme de chiffrement et de déchiffrement asymétrique, il y a deux processus différents qui sont les suivants :

* Création/génération de clés publiques et privées.
* Encryptage et décryptage de messages clairs et pleins.

Afin de démontrer la validité de cette affirmation définie ci-dessus, Gonzalo Alfredo Romero Francia a trouvé, sur Internet, une page web montrant comment réaliser une implémentation simple et efficace de l’algorithme ECC, tout en utilisant l’algorithme de chiffrement et de déchiffrement symétrique AES-GCM (Advanced Encryption Standard-Galois/Counter Mode), cette page web a l’adresse URL suivante :

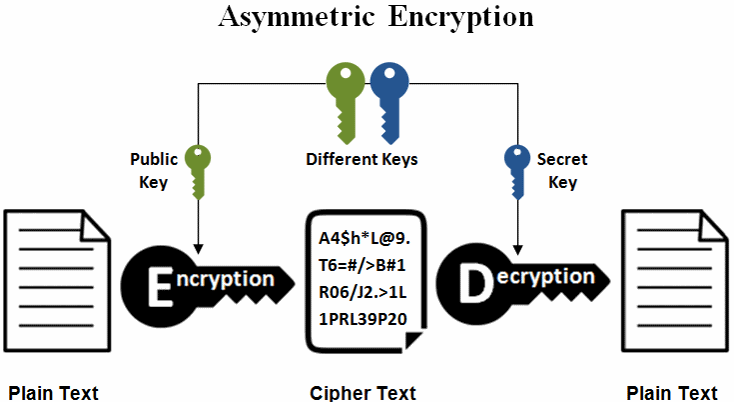
<https://github.com/nakov/Practical-Cryptography-for-Developers-Book/blob/master/asymmetric-key-ciphers/ecc-encryption-decryption.md>

Le voici le contenu de cette page web, qui est écrit en anglais (citation tel que trouvé):

**ECC Encryption / Decryption**

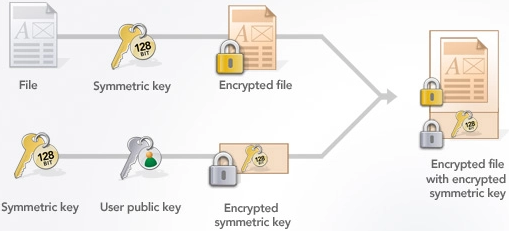
In this section we shall explain how to implement **elliptic-curve based public-key encryption / decryption** (asymmetric encryption scheme based on ECC). This is **non-trivial** and usually involves a design of hybrid encryption scheme, involving ECC cryptography, ECDH key exchange and symmetric encryption algorithm.

Assume we have a ECC **private-public key pair**. We want to encrypt and decrypt data using these keys. By definition, **asymmetric encryption** works as follows: if we **encrypt data by a public key**, we will be able to **decrypt** the ciphertext later by the corresponding **private key**:

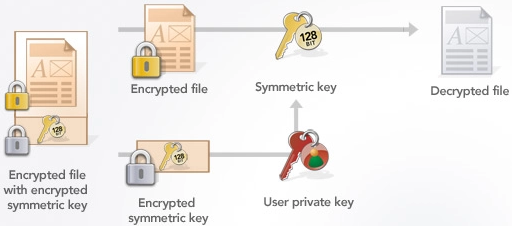


The above process can be directly applied for the **RSA** cryptosystem, but not for the **ECC**. The elliptic curve cryptography (ECC) **does not directly provide encryption** method. Instead, we can design a **hybrid encryption scheme** by using the **ECDH** (Elliptic Curve Diffie–Hellman) key exchange scheme to derive a **shared secret key** for symmetric data encryption and decryption.

This is how most **hybrid encryption schemes** works (the encryption process):



This is how most **hybrid encryption schemes** works (the decryption process):



Let's get into details how to design and implement an **ECC-based hybrid encryption scheme**.

**ECC-Based Secret Key Derivation (using ECDH)**

Assume we have a **cryptographic elliptic curve** over finite field, along with its generator point **G**. We can use the following two functions to calculate a **shared a secret key** for **encryption** and **decryption** (derived from the ECDH scheme):

* **calculateEncryptionKey**(pubKey) --> (sharedECCKey, ciphertextPubKey)
  1. Generate **ciphertextPrivKey** = *new****random****private key*.
  2. Calculate **ciphertextPubKey** = ciphertextPrivKey \* G.
  3. Calculate the ECDH shared secret: **sharedECCKey** = pubKey \* ciphertextPrivKey.
  4. Return both the **sharedECCKey** + **ciphertextPubKey**. Use the **sharedECCKey** for symmetric encryption. Use the randomly generated **ciphertextPubKey** to calculate the decryption key later.
* **calculateDecryptionKey**(privKey, ciphertextPubKey) --> sharedECCKey
  1. Calculate the ECDH shared secret: **sharedECCKey** = ciphertextPubKey \* privKey.
  2. Return the **sharedECCKey** and use it for the decryption.

The above calculations use the same math, like the **ECDH** algorithm (see the [previous section](https://github.com/nakov/Practical-Cryptography-for-Developers-Book/blob/master/asymmetric-key-ciphers/ecdh-key-exchange.md)). Recall that EC points have the following property:

* (***a*** \* **G**) \* ***b*** = (***b*** \* **G**) \* ***a***

Now, assume that ***a*** = privKey, ***a*** \* **G** = pubKey, ***b*** = ciphertextPrivKey, ***b*** \* **G** = ciphertextPubKey.

The above equation takes the following form:

* pubKey \* ciphertextPrivKey = ciphertextPubKey \* privKey = **sharedECCKey**

This is what exactly the above two functions calculate, directly following the **ECDH key agreement** scheme. In the hybrid encryption schemes the encapsulated **ciphertextPubKey** is also known as "**ephemeral key**", because it is used temporary, to derive the symmetric encryption key, using the ECDH key agreement scheme.

**ECC-Based Secret Key Derivation - Example in Python**

The below Python code uses the tinyec library to generate a **ECC private-public key pair** for the message recipient (based on the brainpoolP256r1 curve) and then derive a **secret shared key** (for encryption) and ephemeral **ciphertext public key** (for ECDH) from the recipient's **public key** and later derive the same **secret shared key** (for decryption) from the recipient's **private key** and the generated earlier ephemeral **ciphertext public key**:

from tinyec import registry

import secrets

curve = registry.get\_curve('brainpoolP256r1')

def compress\_point(point):

return hex(point.x) + hex(point.y % 2)[2:]

def ecc\_calc\_encryption\_keys(pubKey):

ciphertextPrivKey = secrets.randbelow(curve.field.n)

ciphertextPubKey = ciphertextPrivKey \* curve.g

sharedECCKey = pubKey \* ciphertextPrivKey

return (sharedECCKey, ciphertextPubKey)

def ecc\_calc\_decryption\_key(privKey, ciphertextPubKey):

sharedECCKey = ciphertextPubKey \* privKey

return sharedECCKey

privKey = secrets.randbelow(curve.field.n)

pubKey = privKey \* curve.g

print("private key:", hex(privKey))

print("public key:", compress\_point(pubKey))

(encryptKey, ciphertextPubKey) = ecc\_calc\_encryption\_keys(pubKey)

print("ciphertext pubKey:", compress\_point(ciphertextPubKey))

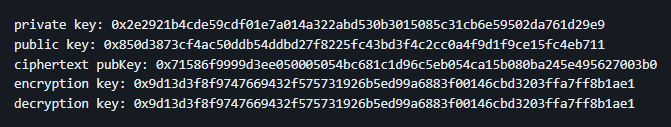
print("encryption key:", compress\_point(encryptKey))

decryptKey = ecc\_calc\_decryption\_key(privKey, ciphertextPubKey)

print("decryption key:", compress\_point(decryptKey))

Run the above code example: <https://repl.it/@nakov/ECC-based-secret-key-derivation-in-Python>.

The code is pretty simple and demonstrates that we can generate a pair { **secret key** + **ciphertext public key** } from given EC **public key** and later we can recover the same **secret key** from the pair { **ciphertext public key** + **private key** }. The above code produces output like this:



It is clear from the above output that the **encryption key** (derived from the public key) and the **decryption key** (derived from the corresponding private key) **are the same**. This is due to the above discussed property of the ECC: pubKey \* ciphertextPrivKey = ciphertextPubKey \* privKey. These keys will be used for data encryption and decryption in an integrated encryption scheme. The above output will be different if you run the code (due to the randomness used to generate ciphertextPrivKey, but the encryption and decryption keys will always be the same (the ECDH shared secret).

The above demonstrated mechanism for generating a shared ephemeral secret key, based on a ECC key pair, is an example of **KEM** (key encapsulation mechanism), based on the ECC and ECDH.

**ECC-Based Hybrid Encryption / Decryption - Example in Python**

Once we have the **secret key**, we can use it for **symmetric data encryption**, using a symmetric encryption scheme like AES-GCM or ChaCha20-Poly1305. Let's implement a fully-functional **asymmetric ECC encryption and decryption** hybrid scheme. It will be based on the brainpoolP256r1 curve and the **AES-256-GCM** authenticated symmetric cipher.

We shall use the tinyec and pycryptodome Python libraries respectively for ECC calculations and for the AES cipher:

pip install tinyec

pip install pycryptodome

Let's examine this full **ECC + AES hybrid encryption** example:

from tinyec import registry

from Crypto.Cipher import AES

import hashlib, secrets, binascii

def encrypt\_AES\_GCM(msg, secretKey):

aesCipher = AES.new(secretKey, AES.MODE\_GCM)

ciphertext, authTag = aesCipher.encrypt\_and\_digest(msg)

return (ciphertext, aesCipher.nonce, authTag)

def decrypt\_AES\_GCM(ciphertext, nonce, authTag, secretKey):

aesCipher = AES.new(secretKey, AES.MODE\_GCM, nonce)

plaintext = aesCipher.decrypt\_and\_verify(ciphertext, authTag)

return plaintext

def ecc\_point\_to\_256\_bit\_key(point):

sha = hashlib.sha256(int.to\_bytes(point.x, 32, 'big'))

sha.update(int.to\_bytes(point.y, 32, 'big'))

return sha.digest()

curve = registry.get\_curve('brainpoolP256r1')

def encrypt\_ECC(msg, pubKey):

ciphertextPrivKey = secrets.randbelow(curve.field.n)

sharedECCKey = ciphertextPrivKey \* pubKey

secretKey = ecc\_point\_to\_256\_bit\_key(sharedECCKey)

ciphertext, nonce, authTag = encrypt\_AES\_GCM(msg, secretKey)

ciphertextPubKey = ciphertextPrivKey \* curve.g

return (ciphertext, nonce, authTag, ciphertextPubKey)

def decrypt\_ECC(encryptedMsg, privKey):

(ciphertext, nonce, authTag, ciphertextPubKey) = encryptedMsg

sharedECCKey = privKey \* ciphertextPubKey

secretKey = ecc\_point\_to\_256\_bit\_key(sharedECCKey)

plaintext = decrypt\_AES\_GCM(ciphertext, nonce, authTag, secretKey)

return plaintext

msg = b'Text to be encrypted by ECC public key and ' \

b'decrypted by its corresponding ECC private key'

print("original msg:", msg)

privKey = secrets.randbelow(curve.field.n)

pubKey = privKey \* curve.g

encryptedMsg = encrypt\_ECC(msg, pubKey)

encryptedMsgObj = {

'ciphertext': binascii.hexlify(encryptedMsg[0]),

'nonce': binascii.hexlify(encryptedMsg[1]),

'authTag': binascii.hexlify(encryptedMsg[2]),

'ciphertextPubKey': hex(encryptedMsg[3].x) + hex(encryptedMsg[3].y % 2)[2:]

}

print("encrypted msg:", encryptedMsgObj)

decryptedMsg = decrypt\_ECC(encryptedMsg, privKey)

print("decrypted msg:", decryptedMsg)

The above example starts from generating an ECC public + private **key pair** for the message recipient: pubKey + privKey, using the tinyec library. These keys will be used to **encrypt** the message msg through the hybrid encryption scheme (asymmetric ECC + symmetric AES) and to **decrypt** is later back to its original form.

Next, we **encrypt** msg by using the pubKey and we obtain as a result the following set of output: { ciphertext, nonce, authTag, ciphertextPubKey }. The ciphertext is obtained by the symmetric AES-GCM encryption, along with the nonce (random AES initialization vector) and authTag (the MAC code of the encrypted text, obtained by the GCM block mode). Additionally, we obtain a randomly generated ephemeral public key ciphertextPubKey, which will be encapsulated in the encrypted message and will be used to recover the AES symmetric key during the decryption (using the ECDH key agreement scheme, as it was show before).

To **decrypt** the encrypted message, we use the data produced during the encryption { ciphertext, nonce, authTag, ciphertextPubKey }, along with the decryption privateKey. The result is the decrypted plaintext message. We use authenticated encryption (GCM block mode), so if the decryption key or some other parameter is incorrect, the decryption will fail with an **exception**.

Internally, the encrypt\_ECC(msg, pubKey) function first generates an ephemeral **ECC key-pair** for the ciphertext and calculates the symmetric encryption shared ECC key sharedECCKey = ciphertextPrivKey \* pubKey. This key is an EC point, so it is then transformed to **256-bit AES secret key** (integer) though hashing the point's x and y coordinates. Finally, the **AES-256-GCM** cipher (from pycryptodome) **encrypts** the message by the 256-bit shared secret key secretKey and produces as **output** ciphertext + nonce + authTag.

The decrypt\_ECC(encryptedMsg{ciphertext, nonce, authTag, ciphertextPubKey}, privKey) function internally first calculates the symmetric encryption shared ECC key sharedECCKey = privKey \* ciphertextPubKey. It is an EC point, so it should be first transformed to **256-bit AES secret key** though hashing the point's x and y coordinates. Then the **AES-256-GCM cipher** is used to **decrypt** the ciphertext + nonce + authTag by the 256-bit shared secret key secretKey. The produced output is the original plaintext message (or an exception in case of incorrect decryption key or unmatching authTag).

The output from the above code looks like this:

original msg: b'Text to be encrypted by ECC public key and decrypted by its corresponding ECC private key'

encrypted msg: {'ciphertext': b'b5953b3082fcefdbde91dd3c03cf83dde0822c19be6ae906a634db65115295e7cbcd7a1a492d69ba5be91990c70d8df9dc84360cf554f155ef81ce1f0ad44bd9fdabbc5f960517089262b3390e61b37610012bee4e6bcae335', 'nonce': b'9d55f4b5c87fff773d0457f3b23a953e', 'authTag': b'5c9d339778925aa4e44f43252a28681d', 'ciphertextPubKey': '0x21dbc985b625f2a42d0f86fc234b49b55477928bae73dfac73bafd9bed50abe70'}

decrypted msg: b'Text to be encrypted by ECC public key and decrypted by its corresponding ECC private key'

Lorsque Gonzalo Alfredo Romero Francia a trouvé le contenu de cette page web, montré par les pages 25 à 30 du présent document, il a décidé de le prendre comme code python de base, pour coder son propre code python lui permettant d’implémenter efficacement l’algorithme ECC, d’offrir à l’utilisateur de spécifier le type de courbe elliptique (EC) qui sera utilisé pour encrypter et décrypter des messages clairs et pleins.

Cependant, avant même de commencer à coder ce code python, satisfaisant à toutes les demandes du projet de fin d’études nommé «  Implémentation et comparaison des algorithmes de chiffrage asymétrique », il a fallu avant qu’il apprenne et comprenne toute la théorie mathématique associée à l’algorithme de chiffrement et de déchiffrement de messages clairs et pleins ECC,

//Continuer lundi prochain!

**4. Points saillants en lien avec le critère d’évaluation de la performance des codes python associés aux algorithmes asymétriques RSA, El-Gamal et ECC, nommé l’effet avalanche :**

En ce qui a trait le critère d’évaluation de la performance des codes python associés aux algorithmes asymétriques RSA, El-Gamal et ECC, nommé l’effet avalanche, il existe deux types de ce critère d’évaluation, formalisés à partir de la définition du concept de l'effet avalanche, et sont les suivantes :

4.1 Le critère d'avalanche strict (SAC) : est le type de critère d’avalanche qui est satisfait si chaque bit de sortie change avec une probabilité de 50 % chaque fois qu'un seul bit d'entrée est complété.

4.2 Le critère d'indépendance des bits (BIC) : est le type de critère d’avalanche qui stipule que si un seul bit d'entrée s'inverse pour tous les x, y et z, les bits de sortie y et z doivent changer indépendamment.

Le point 4.2 fournit des explications supplémentaires plus détaillées sur la définition du critère d'indépendance des bits (BIC), et sont les suivantes :

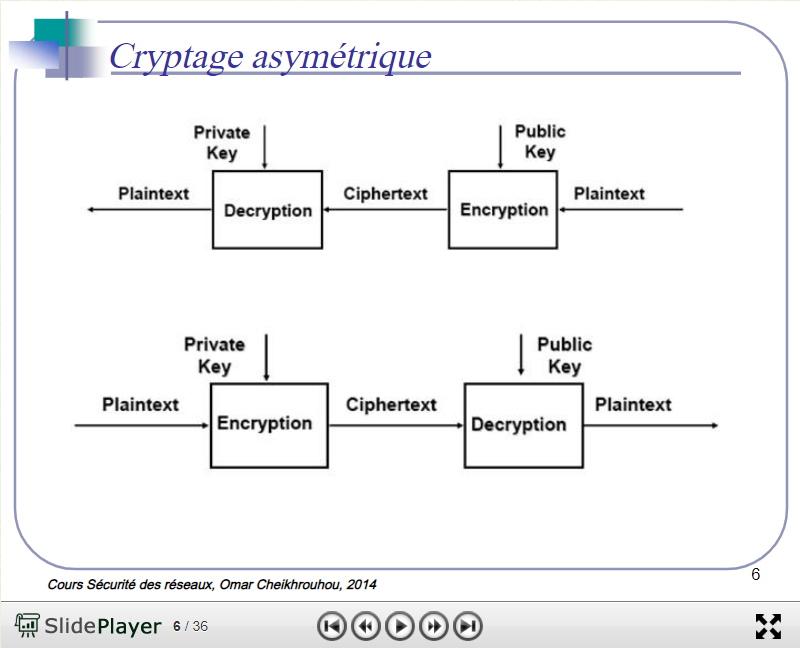
* La valeur de la variable x change de manière indépendante.
* La valeur de la variable y change de manière indépendante, sans produire une relation mathématique avec la valeur de la variable x ou avec le changement indépendant de la valeur de cette variable.
* La valeur de la variable z change de manière indépendante, sans produire une relation mathématique avec la valeur de la variable y ou avec le changement indépendant de la valeur de cette variable.

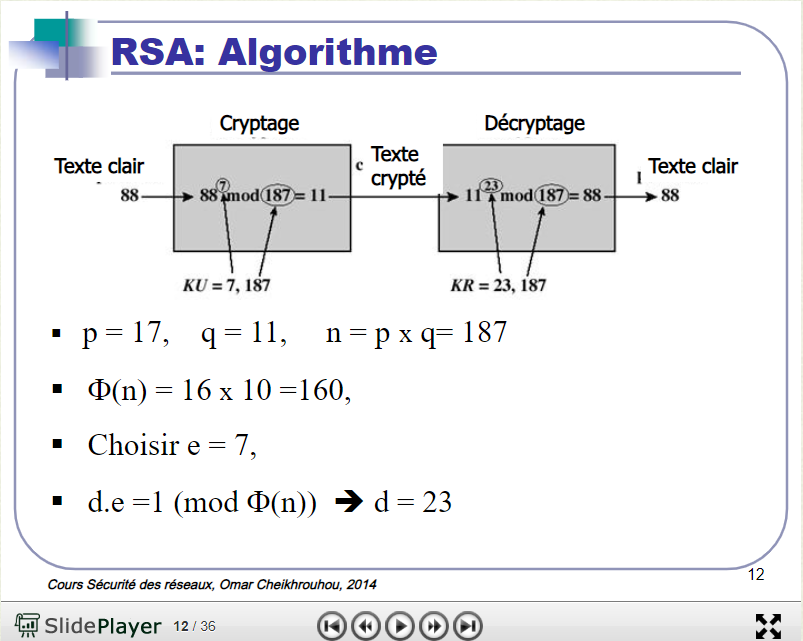
En réalisant beaucoup d’investigations sur la définition du critère d’évaluation des codes python associés aux algorithmes asymétriques RSA, El-Gamal et ECC, nommé l’effet avalanche, Gonzalo Alfredo Romero Francia s’est aperçu et a appris et assimilé que tant le critère d'avalanche strict (SAC) que le critère d'indépendance des bits (BIC) s’appliquent tant aux messages clairs et pleins qu’aux clés publiques et privées, ainsi qu’aux clés intermédiaires créées lors du processus d’authentification et d’échange de clés.

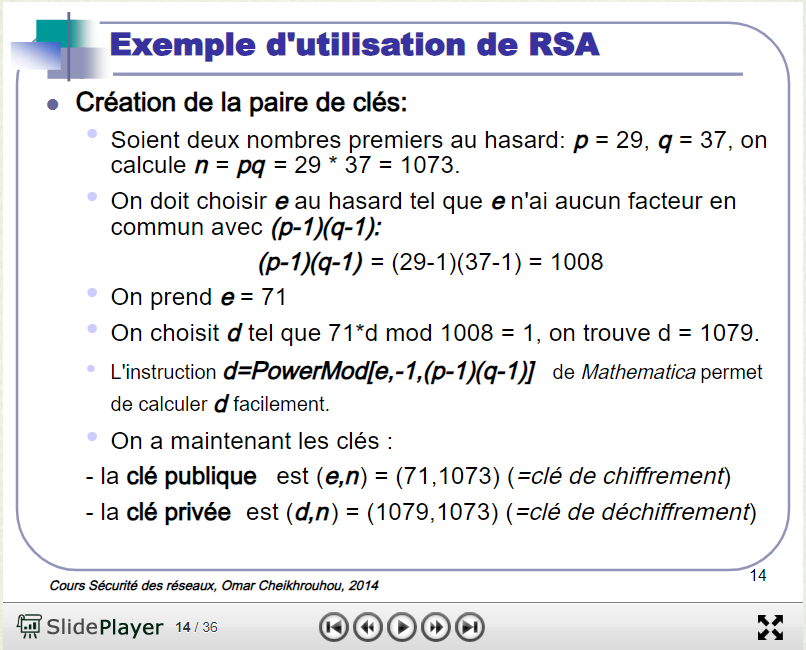
Maintenant, afin de montrer qu’il a bien appris, compris, assimilé et maîtrisé toutes les théories et notions mathématiques en lien avec las algorithmes de chiffrement et de déchiffrement asymétrique RSA, El-Gamal et ECC, l’auteur du présent document a décidé d’utiliser le logiciel Maple 2023 (23.2), afin de montrer les calculs qu’il a faits, en lien avec tous les calculs mathématiques associés aux processus de génération de nombres aléatoire permettant de créer des clés publiques et privées, au processus de création des clés publiques et privées, et aux processus de chiffrement et de déchiffrement asymétrique de messages clairs et pleins, et cela pour chaque algorithme de chiffrement et de déchiffrement asymétrique RSA, El-Gamal et ECC, les voici tous ces calculs effectués pour ces trois algorithmes asymétriques :

**Calculs faits avec Maple 2023, associés à l’algorithme asymétrique RSA :**

Afin de montrer le fait que l’auteur du présent document a réussi à apprendre, à comprendre, à assimiler et à maîtriser toutes les notions et concepts mathématiques que l’algorithme RSA utilise, pour chiffrer et déchiffrer des messages clairs et pleins, il a décidé de prendre un bon exemple de calcul simple, qu’il a trouvé sur Internet, de tous les processus associés à cet algorithme asymétrique, qui sont la génération des nombres premiers aléatoires permettant de créer les clés publiques et privées, la création de ces deux types de clés, ainsi que le chiffrement et le déchiffrement de messages clairs et pleins (numériques, alphabétiques ou alpha-numériques). Le voici des captures d’écran des diapositives montrant en détail cet exemple :

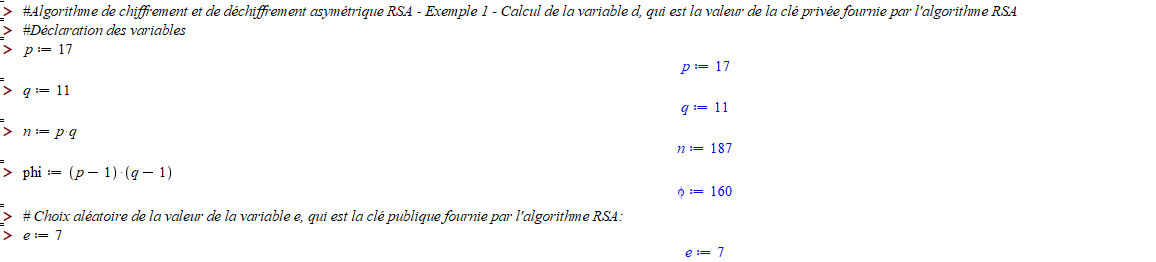




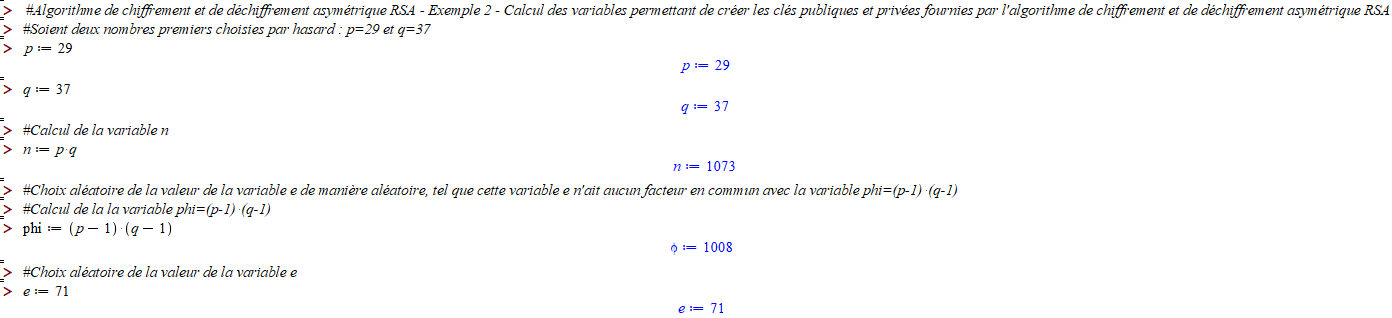




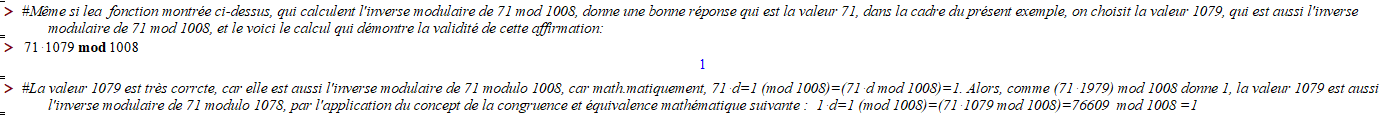
Les voici tous les calculs associés à l’exemple montré ci-dessus, calculés en utilisant le logiciel Maple 2023 (23.2) :

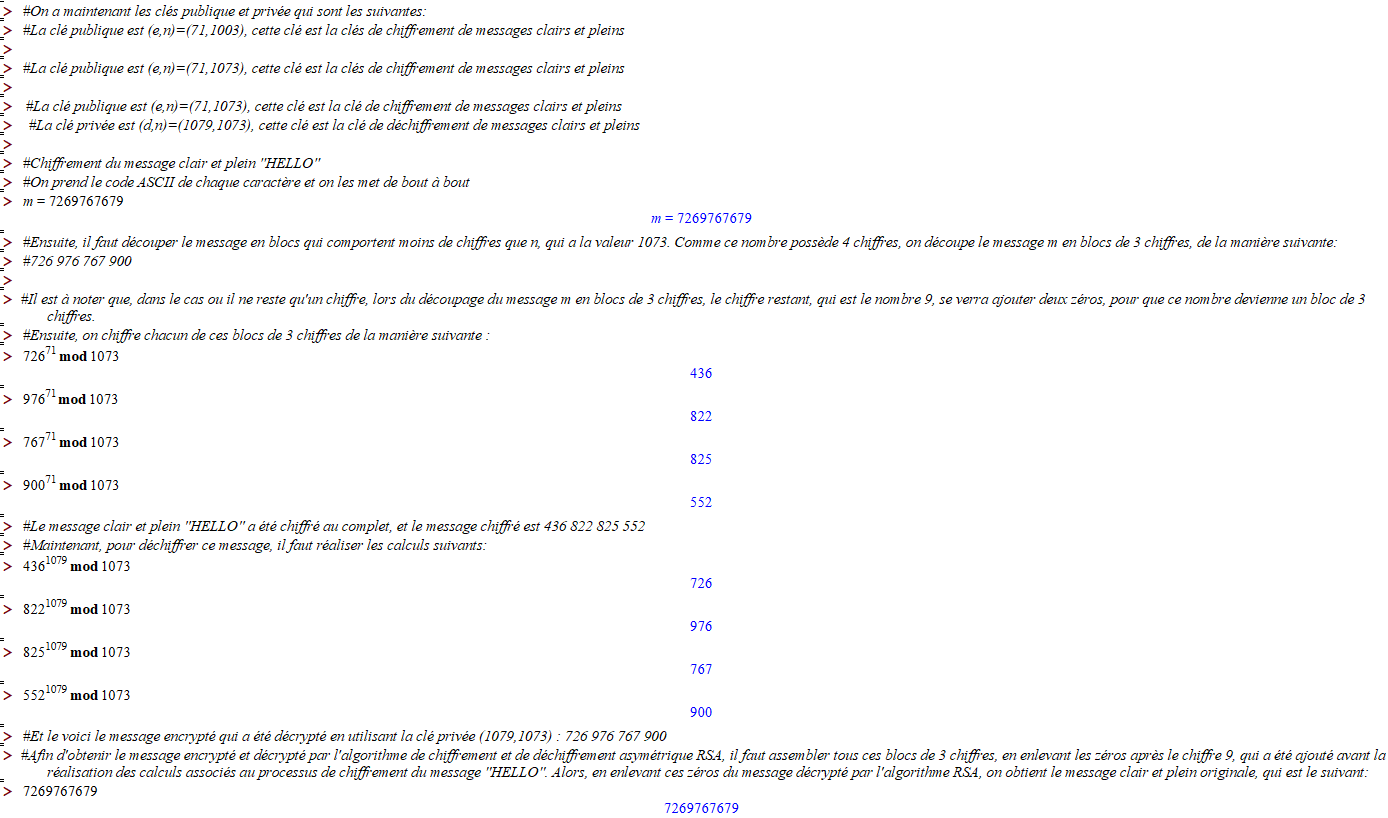










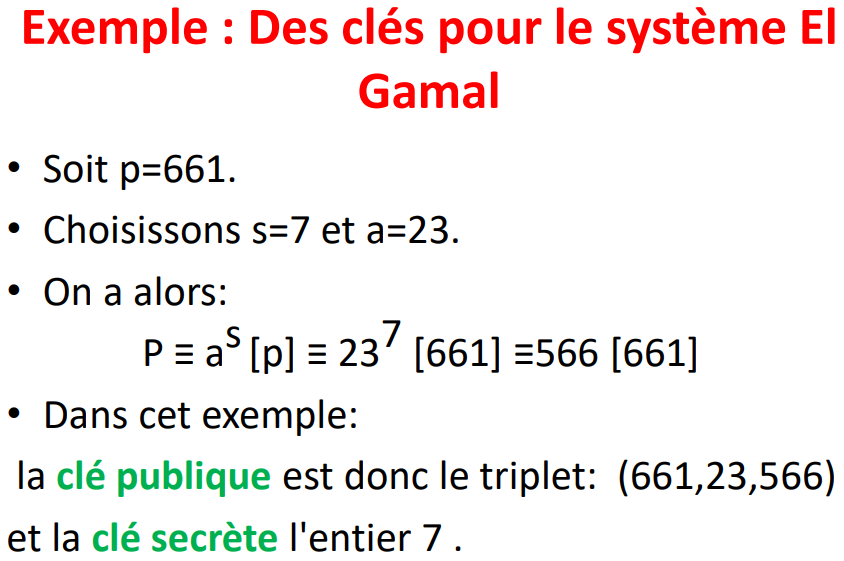


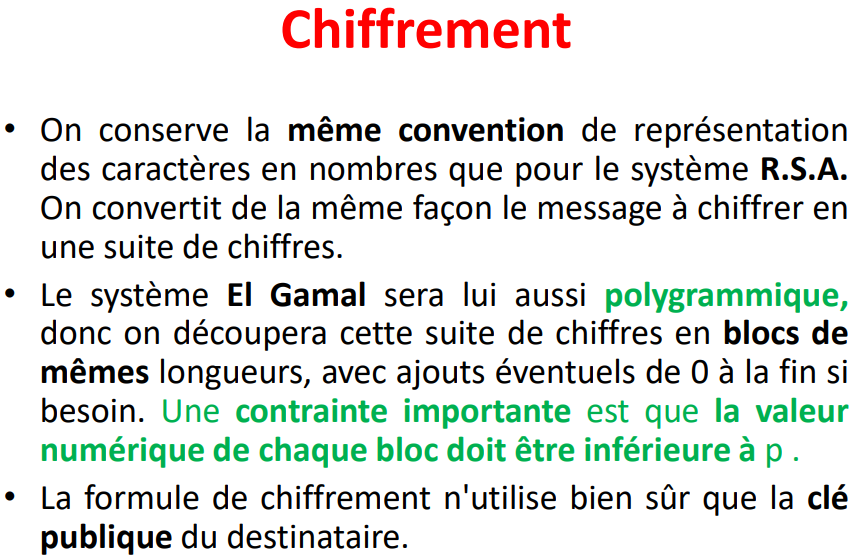
Et les voici les calculs mathématiques réalisés, nécessaires pour réaliser tous les processus associés à l’algorithme de chiffrement et de déchiffrement RSA, notamment le processus de génération des nombres entiers premiers utilisés pour créer les clés publiques et privées, le processus de création de clés publique et privé, et le processus de chiffrement et déchiffrement de messages clairs pleins, en utilisant des calculs de type modulaire avec la valeur des variables e,d et n, qui sont la clé publique, la clé privée et le nombre entier premier utilisé pour encrypter et décrypter des messages clairs pleins. Les calculs montrés dans les pages 21 à 27 du présent document montrent clairement et catégoriquement que Gonzalo Alfredo Romero Francia a bien réussi à apprendre, à comprendre, à assimiler et à maîtriser toutes les notions mathématiques associées à l’algorithme de chiffrement et de déchiffrement asymétrique RSA.

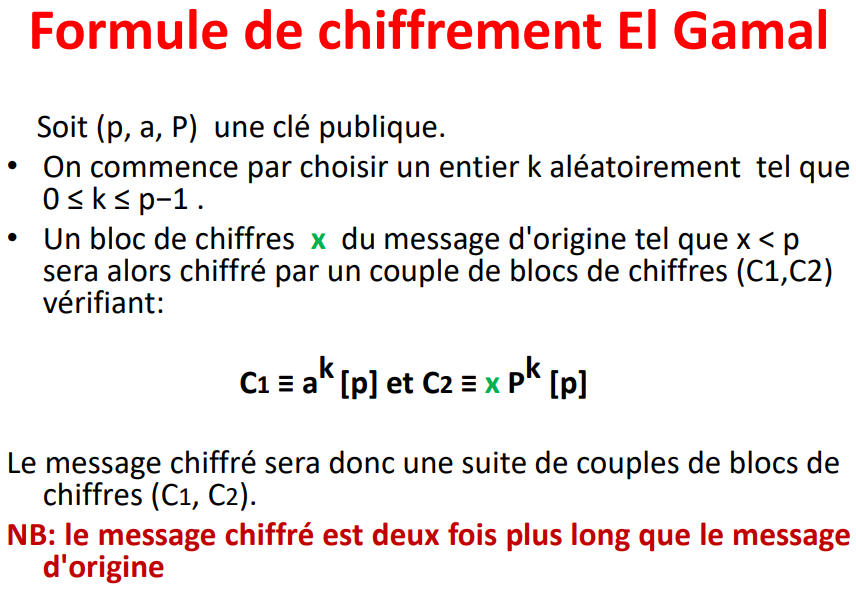
Cependant, il est très clair que l’implémentation de cet algorithme de chiffrement et de déchiffrement asymétrique, en langage de programmation Python, va générer des nombres entiers premiers aléatoires gigantesques et colossales qui vont créer des clés publiques et privées aussi de valeurs gigantesques et colossales, et la même chose se produit avec les messages encryptés et décryptés. Alors pour conclure, l’auteur du présent document a décidé de montrer simplement qu’il maîtrise très bien tous les calculs mathématiques nécessaires à la réalisation de tous les processus associés à l’algorithme asymétrique RSA, qui sont le processus de génération des nombres entiers premiers utilisés pour créer les clés publiques et privées, le processus de création de clés publique et privé, et le processus de chiffrement et déchiffrement de messages clairs pleins, en utilisant des calculs de type modulaire avec la valeur des variables e,d et n.

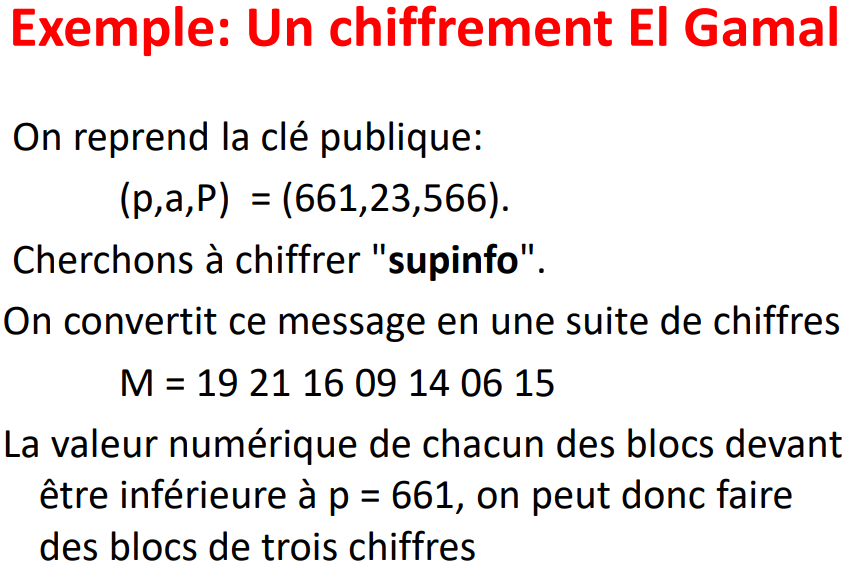
**Calculs faits avec Maple 2023, associés à l’algorithme asymétrique El-Gamal :**

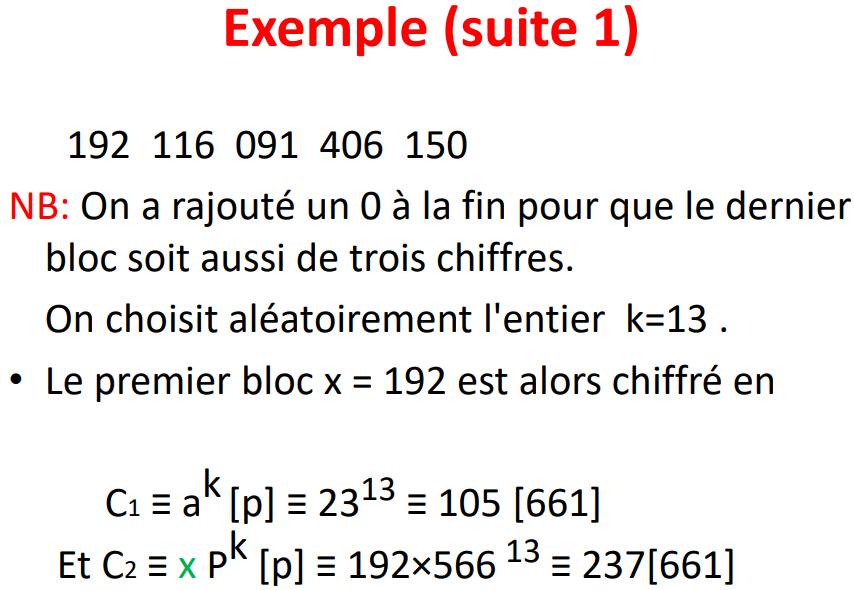
Afin de montrer le fait que l’auteur du présent document a réussi à apprendre, à comprendre, à assimiler et à maîtriser toutes les notions et concepts mathématiques que l’algorithme El-Gamal utilise, pour chiffrer et déchiffrer des messages clairs et pleins, il a décidé de prendre un bon exemple de calcul simple, qu’il a trouvé sur Internet, de tous les processus associés à cet algorithme asymétrique, qui sont la génération des nombres premiers aléatoires permettant de créer les clés publiques et privées, la création de ces deux types de clés, ainsi que le chiffrement et le déchiffrement de messages clairs et pleins (numériques, alphabétiques ou alpha-numériques). Le voici des captures d’écran des diapositives montrant en détail cet exemple :

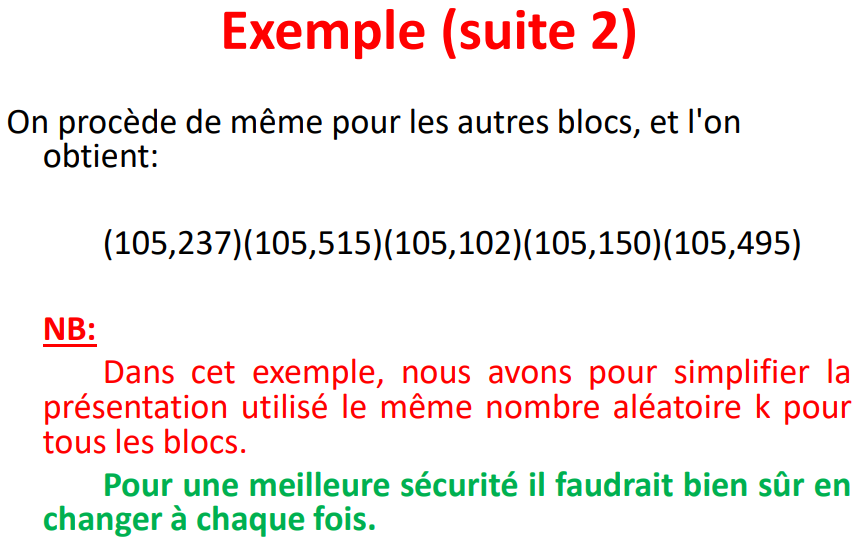


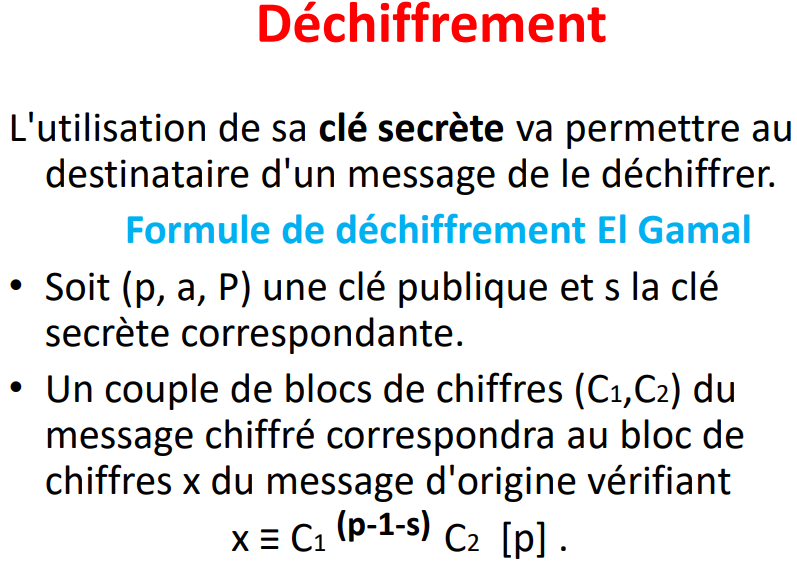


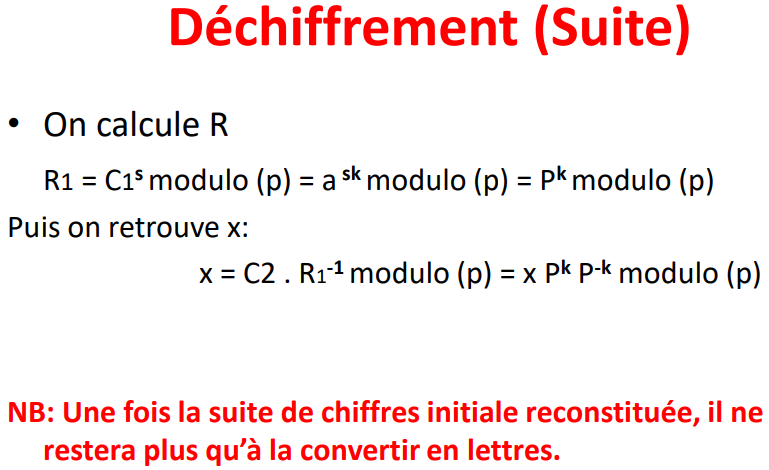


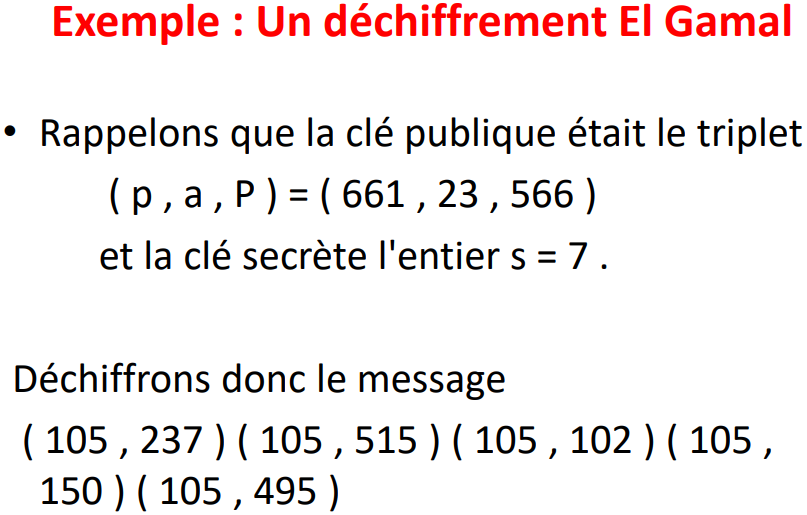


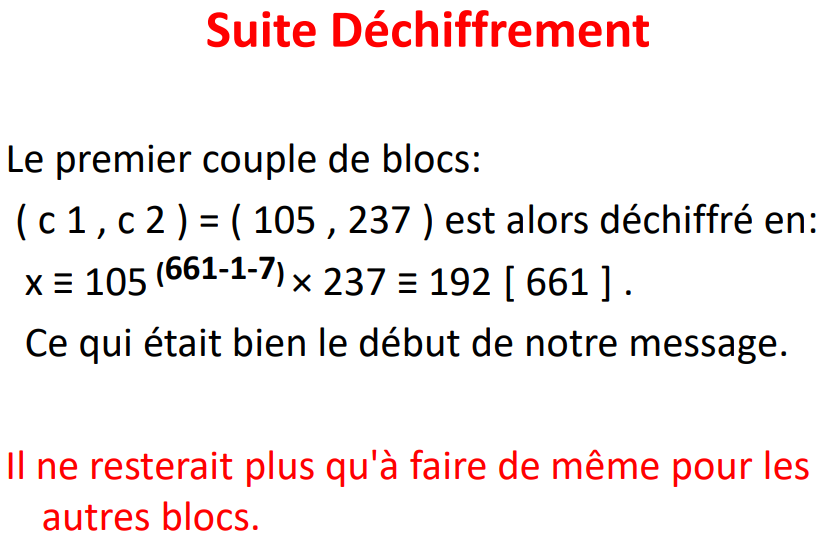


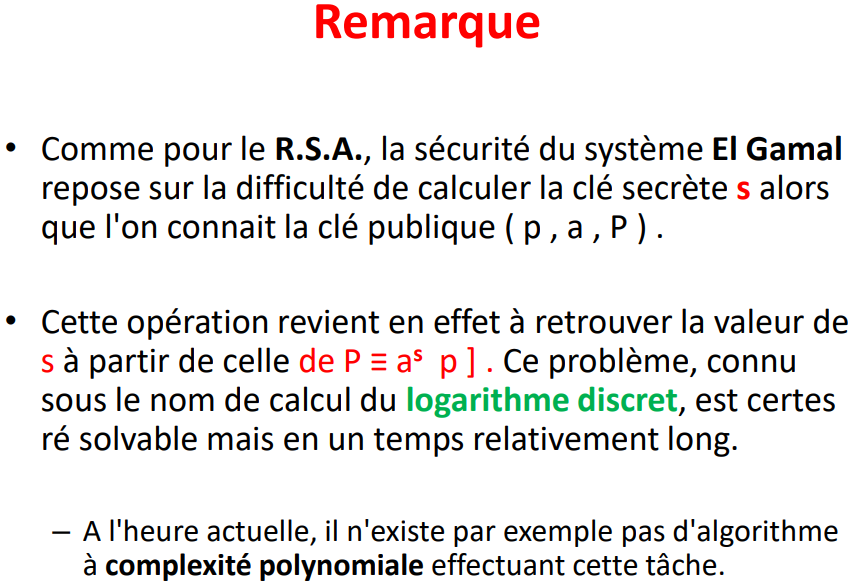












Les voici tous les calculs associés à l’exemple montré ci-dessus, calculés en utilisant le logiciel Maple 2023 (23.2) :

