الجمهوريـة الجزائـريـة الديمقراطيـة الشعبيـة
République Algérienne Démocratique et Populaire
وزارة التعليـم العالـي والبحـث العلمـي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

N° Réf :……..

Centre Universitaire de Mila

Institut des Sciences et de la Technologie **Département Mathématiques et Informatique**

Mémoire préparé En vue de l'obtention du diplôme de licence

En Filière : Informatique générale

Thème

Sécurité des données par le système cryptographique à clé mixte : PGP

Préparé par : - Hammouda yaaQoub
- Bouchemal zakaria
- Belbedroune farid

Encadré par: Mme deffas zineb M.A.A

Année universitaire :2013/2014

REMERCIMENTS

Je tiens tout d'abord à remercier Dieu le tout puissant et miséricordieux, qui m'a donné la force et la patience d'accomplir ce modeste travail.

Un remerciement particulier à mon encadreur Mme DEFFAS ZINEB pour sa présence, son aide et surtout pour ses précieux conseils qui m'ont beaucoup assisté pour l'accomplissement de ce projet.

Je tiens à exprimer mes sincères remerciements à tout le personnel de l'institut de sciences et de la technologie du centre universitaire de Mila surtout les enseignants qui m'ont enseigné durant des années d'étude.

Mes remerciements à mes très chers parents, frères, sœurs, amis qui m'ont encouragé, soutenu durant tout mon parcours.

Un remerciement spécial à mon amie « walide »

Je remercie les étudiants de la promotion 2013/2014 pour avoir été liés et unis tout au long de cette année et tous ceux qui ont collaborés de prés ou de loin à l'élaboration de ce travail. Qu'ils acceptent mes humbles remerciements.

Enfin, que toutes les personnes qui ont contribué à l'élaboration de cette mémoire, trouvent ici l'expression de ma connaissance.

Dédicace

Je dédie ce travail en tout premiers lieu à mon dieu ALLAH le tout puissant et miséricordieux qui m'a donné la force, la volonté et le courage pour accomplir ce modeste projet.

À Ma chére Mére « haçina » pour ses sacrifices depuis Qu'elle m'a mis au monde , Mon Pére « el Sadek », qui ont le droit de recevoir mes chaleureux remerciements pour le courage et le sacrifice qu'ils ont consentis pendant la durée de mes études, J'espère qu'ils trouveront dans ce travail toute l'expression de ma reconnaissance.

À mes frères et sœur « asma , meriem ,khawla ,ishak , yousef mehdi ,ossama ». « tant lindaa , zahwa, nassima ,atika, efatiha,atika ,fahima .

À mes grands parents « mohammed, zineb , horia » et Ma petite bébé rana et nibrasse abd rahman .

À toute ma famille « hammouda et baali » toute Oncles et tantes.

À tous mes chers amis « baggio ,abd rahman, bilal , daoud et tout les amis de cafétéria kolombo, » , et mes collègues de l'Université « mohammed , zaki ,faride , imi , madiha , chocho ,moks, rokiya ,joujou, amina² ,aicha ,…… » , et à tous ce qui m'ont enseigne toute au long de mon parcoure

A mon encadreur Mem DEFFAS ZINEB.

[***Hammouda yaaQoub***]

Dédicace

Je dédie ce travail en tout premiers lieu à mon dieu ALLAH le tout puissant et miséricordieux qui m'a donné la force, la volonté et le courage pour accomplir ce modeste projet.

A mes très chers parents(Razika et Djamel), qui ont le droit de recevoir mes chaleureux remerciements pour le courage et le sacrifice qu'ils ont consentis pendant la durée de mes études, J'espère qu'ils trouveront dans ce travail toute l'expression de ma reconnaissance.

À mes frères et sœur(youcef,soumia,amel,moufida).

À Ma tonte (nadjwa).

À Ma cousine (Nardjes) .

À mes grands mère(alhadja massouda).

À toute ma famille(Bouchemalet Makroud)

À tous mes chers amis(walid,mehdi,roukho,hamza,yaakoub,farid,taha ,salah,mouhamed,etles amis de cafétéria Alzaim) et mes collègues de l'Université (bilel,sif,2,abdou,farid,yaakoub,walid,abd alssalem,abd albastet) à tous ce qui m'ont enseigne toute au long de mon parcoure

A mon encadreur DEFFAS ZINEB.

Bouchemal zakaria

Dédicace

Je dédie ce travail en tout premiers lieu à mon dieu ALLAH le tout puissant et miséricordieux qui m'a donné la force, la volonté et le courage pour accomplir ce modeste projet.

À Ma chére Mére « Zahiya » pour ses sacrifices depuis Qu'elle m'a mis au monde, Mon Pére « Hocine », qui ont le droit de recevoir mes chaleureux remerciements pour le courage et le sacrifice qu'ils ont consentis pendant la durée de mes études, J'espère qu'ils trouveront dans ce travail toute l'expression de ma reconnaissance.

À mes frères et sœur « Nadir, Charaf, Houssam, Iman ».

À ma grand mére « Yamina ».

À toute ma famille « Belbedroune et Bensikhaled et bougassa » toute Oncles et tantes.

À tous mes chers amis « Khaled ,Faysal, Charaf , Abdou,Ramzi,Fares,Abdeljebbar,Kacem ,Aicha » , et mes collègues de l'Université « Sayf , zaki , chouchou ,moks, Toufik, Bilal, Abdou ,…… » , et à tous ce qui m'ont enseigne toute au long de mon parcoure

A mon encadreur Mem DEFFAS ZINEB.

Belbedroune Farid

Résumé

Le développement rapide des réseaux mondiaux et les immenses possibilités offertes par les transactions électroniques posent aujourd'hui de manière cruciale le problème de protection des informations échangées sur les réseaux informatiques mondiaux, d'où l'utilisation des techniques cryptographiques. Le cryptage et le décryptage sont devenus de plus en plus indispensables comme moyen de confidentialités et de protection de l'information contre tout système d'espionnage. L'objectif de notre travail est l'étude des différentes classes d'algorithmes cryptographiques ainsi que la réalisation d'un système cryptographique à clé mixte PGP (la clé IDEA utilisé pour crypter et décrypter le texte et les clés RSA pour crypter et décrypter la clé IDEA).

Table des matières

chapitre 1 : Etat de l'art sur la cryptographie

chapitre 2 : Systéme PGP

chpiter 3 : Implémentation

Table des Figure :

chapitre 2: Systéme PGP .

Chapitre 3 : Implémentation

Liste des Tableaux

Introduction générale

Introduction générale

Depuis les temps historiques les plus reculés, les hommes ont utilisé diverses méthodes pour coder des messages et des textes et rendre ceux-ci inintelligibles à des lecteurs indésirables.

Aujourd'hui, le champ d'application de la cryptologie s'est élargi et a trouvé un regain d'actualité avec les problèmes posés par la sécurité des transactions bancaires, des transmissions de fichiers et de bases de données sous forme électronique.

La cryptographie (sécurité des données) est l'ensemble des processus de verrouillage visant à protéger l'accès à certaines données afin de les rendre incompréhensible aux personnes non autorisées, autrement dit garantir la confidentialité, l'intégrité de ces informations, ainsi que leur imputabilité. L'émetteur d'une information doit être certain de l'identité du destinataire et inversement.

La cryptographie est une science très ancienne, elle est un moyen de sauvegarde le caractère confidentiel des informations. Elle ne protégé pas les communications en tant que telles mais plutôt leur contenu. Plusieurs algorithmes ont été proposé pour le cryptage, ces algorithmes sont classifié selon différents critères, par exemple en peut les classifier selon l'apparition (classiques et modernes), ou selon la nature de la clé (publique ou secrète).

Puisque cette science est très large, nous essayons de faire une implémentation l'algorithme PGP (Pretty Good Privacy).

Afin d'atteindre notre objectif nous divisons notre travail en 3 chapitres:

Le premier chapitre présente un état de l'art sur la cryptographie, et un panorama sur des algorithmes cryptographique existent avec des exemples. Et termine par quelques concepts sur la cryptanalyse et les différentes techniques utilisent dans cette opération.

Le deuxième chapitre explique en détaille l'algorithme que nous avons choisi.

Le dernier chapitre commence par la description du langage et l'environnement de développement choisi pour l'implémentation. Puis présente l'interface graphique de notre application suivie par des exemples. [12]

chapiter 1:

Etat de l'art sur la cryptographie

Chapitre1 : Etat de l'art sur la cryptographie

1. Introduction :

Dans ce chapitre, nous présentons un aperçu global sur la notion de la cryptographie. Nous commençons par l'historique, puis, nous donnons quelques définitions de la cryptographie et leurs fonctions. Par la suite, nous présentons une classification des algorithmes cryptographique, et enfin nous terminons ce chapitre par la cryptanalyse et les différentes attaques utiliser.

2. Historique de la cryptographie :

Avant de décrire la cryptographie en soi, passons en revue son histoire, son apparition et son évolution au fil du temps ; les aspects techniques brièvement présentés dans cet élément. La science de chiffrement n'est pas une science moderne mais l'histoire de cryptage est plus long est classé sur trois grand étapes :

2.1. L'âge artisanal (part des origines) :

La cryptographie serait apparue pour la première fois il y a 4000 ans, au bord du Nil un scribe aurait tracé d'une façon particulière des hiéroglyphes sur la tombe de son maître. Le but n'était néanmoins pas de rendre le texte illisible, mais de le rendre plus solennel. Mais le cryptage développé avec Jules César utilisait, semble-t-il un mécanisme de confidentialité rudimentaire, où chaque Lettre d'un message était remplacée par celle située trois positions plus loin dans l'alphabet. La méthode se généralise en opérant une permutation quelconque de l'alphabet, et prend le nom de substitution. Une autre méthode, dite de transposition, change l'ordre des lettres; elle a été mise en ouvré au Moyen Age notamment par un dispositif appelé "grille de Cardan". De façon générale, jusqu'au début du vingtième siècle, la cryptographie était affaire de substitutions et de transpositions. On opérait d'ailleurs fréquemment des substitutions non seulement sur des lettres mais sur des mots, en s'aidant d'une sorte de dictionnaire à double entrée, nommé code ou répertoire.

2.2. L'âge technique :

De nombreux dispositifs mécaniques furent mis en place afin de faciliter le chiffrement. La plupart de ces dispositifs se basaient sur des rotors (des disques où sont imprimées les lettres de l'alphabet).

C'est le cas de la machine Enigma, destinée initialement aux civils (inventée en 1919 par Arthur Scherbius et commercialisée en 1923), mais très vite utilisée par l'armée allemande à partir des années 30. La plupart des communications allemandes seront chiffrées via la machine Enigma, d'où l'intérêt pour les alliés de pouvoir déchiffrer leurs messages.

La machine Enigma effectue une substitution qui change à chaque lettre, son fonctionnement est basé sur un assemblage d'un tableau de connexion (qui ne fait que permuter quelques lettres, afin de rendre la cryptanalyse plus difficile), de plusieurs rotors qui effectuent les permutations des lettres, et d'un réflecteur qui renvoie le courant sur le panneau lumineux où la lettre chiffrée s'allume. À chaque lettre tapée, le premier rotor avance d'une position ; lorsque le premier rotor a fait un tour complet, c'est le second qui tourne, et ainsi de suite. Grâce à la combinaison de ces dispositifs, on peut obtenir plus de 1017 clés possibles pour une machine à 5 rotors.

2.3. L'âge paradoxal :

Après la guerre 40-45, il faudra attendre une trentaine d'années avant de nouvelles avancées dans le domaine de la cryptologie.

En 1971, un cryptographe d'IBM, Horst Feistel met au point un algorithme de chiffrement par bloc, nommé Lucifer qui possède de nombreuses variantes. Deux ans plus tard, la NSA modifie Lucifer pour sortir le Data Encryptions Standard, DES, en 1977 ; il fut encore amélioré par la suite et longuement utilisé et reste utilisé encore aujourd'hui (bien qu'il soit moins répandu), notamment avec le Triple DES, qui consiste simplement à opérer trois DES consécutifs avec deux ou trois clés. La même année, le chiffrement à clé publique (ou asymétrique) est présenté pour la première fois dans une publication de W. Diffue et M. Hellman. Ce concept est mis en pratique avec le chiffrement RSA, inventé par Ron Rivest, Adi Shamir et Leonard Adleman et présenté en 1978 dans une publication. Grâce aux algorithmes à clé publique, le problème de distribution des clés est résolu via l'utilisation de deux clés : une pour chiffrer, rendue publique, et une pour déchiffrer, gardée secrète par la personne censée déchiffrer le message.

3. Définition de la cryptographie :

La cryptographie est la science qui utilise les mathématiques pour le cryptage et le décryptage de données.

Elle vous permet ainsi de stocker des informations confidentielles ou de les transmettre sur des réseaux non sécurisés (tels que l'Internet), afin qu'aucune personne autre que le destinataire ne puisse les lire.

Alors que la cryptographie consiste à sécuriser les données, la cryptanalyse est l'étude des informations cryptées, afin d'en découvrir le secret. La cryptanalyse

Classique implique une combinaison intéressante de raisonnement analytique, d'application d'outils mathématiques, de recherche de modèle, de patience, de détermination et de chance. Ces cryptanalyses sont également appelés des pirates.

La cryptologie englobe la cryptographie et la cryptanalyse.

La cryptographie est l'ensemble des techniques qui permet de rendre un message inintelligible. L'action de coder le message initial (plain text) en un message inintelligible, appelé « cryptogramme » (chiper texte), se nomme « chiffrement » (ou cryptage). L'opération inverse est le « déchiffrement » (ou décryptage).

La cryptographie est un moyen de sauvegarder le caractère confidentiel des informations. Elle ne protège pas les communications en tant que telles mais plutôt leur contenu pour protéger l'accès à certaines données afin de les rendre non autorisées incompréhensible aux personnes. [2]

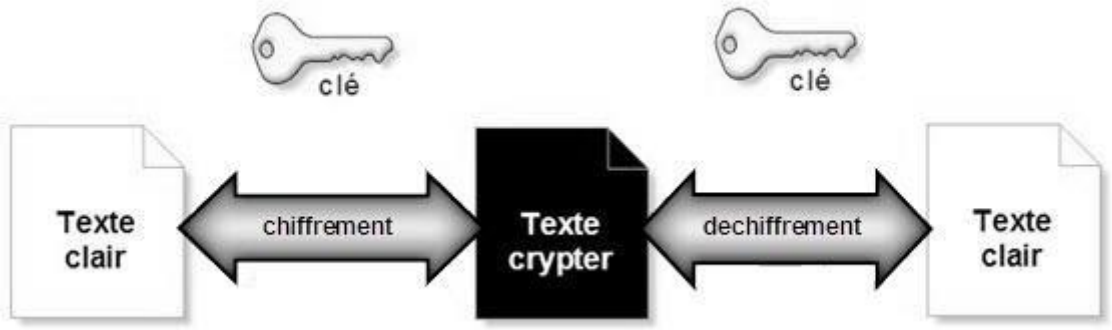


Figure 1:schéma générale de cryptographie

3.1. Les fonctions de la cryptographie :

La cryptographie est traditionnellement utilisée pour dissimuler des messages aux yeux de certains utilisateurs. Cette utilisation a aujourd'hui un intérêt d'autant plus grand que les communications via internet circulent dans des infrastructures dont on ne peut garantir la confidentialité. Désormais, la cryptographie sert non seulement à préserver la confidentialité des données mais aussi à garantir leur intégrité et leur authenticité.

3.1.1. L'authenticité :

L'authentification consiste à assurer l'identité d'un utilisateur, c'est-à-dire de garantir à chacun des correspondants que son partenaire est bien celui qu'il croit être. Un contrôle d'accès peut permettre (par exemple par le moyen d'un mot de passe qui devra être crypté) l'accès à des ressources uniquement aux personnes autorisées.

3.1.2. Non-répudiation :

La non-répudiation de l'information est la garantie qu'aucun des correspondants ne pourra nier la transaction.

Mécanisme pour enregistrer un acte ou un engagement d'une personne ou d'une entité de telle sorte que celle-ci ne puisse pas nier avoir accompli cet acte ou pris cet engagement.

3.1.3. L'intégrité :

Vérifier l'intégrité des données consiste à déterminer si les données n'ont pas été altérées durant la communication (de manière fortuite ou intentionnelle). Garantit que le contenu d'une communication ou d'un fichier n'a pas été modifié. Par exemple, on peut souhaiter vérifier qu'aucun Changement du contenu d'un disque dur n'a eu lieu : des produits commerciaux, mettant en jeu des méthodes cryptologiques, sont disponibles à cet effet.

3.1.4. La confidentialité :

La confidentialité consiste à rendre l'information inintelligible à d'autres personnes que les seuls acteurs de la transaction.

Garantit que le contenu d'une communication ou d'un fichier n'est pas accessible aux tiers. Des services de confidentialité sont offerts dans de nombreux contextes

• en téléphonie mobile, pour protéger les communications dans la partie "aérienne" ;

• en télévision à péage pour réserver la réception des données aux abonnés ;

• dans les navigateurs, par l'intermédiaire du protocole SSL (Secure Socket Layer), dont l'activation est souvent indiquée par un cadenas fermé représenté En bas de la fenêtre.

3.2. A quoi sert la cryptographie ?

L'application la plus évidente de la cryptographie est la protection de la confidentialité d'une information, qu'elle soit stockée localement sur une machine, ou transmise sur un réseau.

Le besoin de confidentialité n'est pas l'apanage des militaires ou de certain gros industriels.

Tous les individus, toutes les organisations ont, à des degrés divers, un tel besoin:

• Confidentialité des transactions bancaires.

• Protection de secrets industriels ou commerciaux.

3.2.1. Mécanismes de la cryptographie :

Un algorithme de cryptographie ou un chiffrement est une fonction mathématique utilisée lors du processus de cryptage et de décryptage. Cet algorithme est associé à une clé

(un mot, un nombre ou une phrase), afin de crypter le texte en clair. Avec des clés différentes, le résultat du cryptage variera également.

La sécurité des données cryptées repose entièrement sur deux éléments : l'invulnérabilité de l'algorithme de cryptographie et la confidentialité de la clé. Un système de cryptographie est constitué d'un algorithme de cryptographie, ainsi que de toutes les clés et tous les protocoles nécessaires à son fonctionnement.

3.3. Les méthodes de cryptographie actuelle :

3.3.1. Le chiffrement actuel :

Le chiffrement est l'action de transformer une information claire, compréhensible de tout le monde, en une information chiffrée, incompréhensible. Le chiffrement est toujours associé au déchiffrement, l'action inverse. Pour ce faire, le chiffrement est opéré avec clé publique ou avec à clé privée.

3.3.1.1 .Systèmes à clé privée :

Les systèmes de cryptage à clé privé, appelé aussi système de cryptage symétrique, sont Utilisés depuis déjà plusieurs siècles. C'est l'approche la plus authentique du chiffrement de donnée et mathématiquement la moins problématique , Dans ce type de système, la clé servant à chiffrer les données peut être facilement déterminée (à l'aide d'ordinateurs) si l'on connaît la clé servant à déchiffrer. Il est aussi facile de trouver la clé de déchiffrement en sachant la clé de chiffrement. Dans la plupart des systèmes symétriques, la clé de chiffrement est la même que la clé de déchiffrement , Autres termes anglais utilisés : Single-key , one-key, private-key, conventional encryption.

Il y a deux catégories de systèmes à clé privée : les chiffrements par blocs et les chiffrements de flux.

- **Chiffrements par blocs :**

Dans ce type de chiffrement, il y a une séparation du texte clair en blocs d'une longueur fixe selon un alphabet, et un algorithme chiffre un bloc à la fois, une grandeur pertinente de la clé définie une bonne sécurité, car il faut considérer une recherche approfondie. Les clés très longues sont plus coûteuses en travail à cause notamment de leur génération, de leur transmission, de leur espace mémoire et de la difficulté de s'en rappeler (mots de passe),La taille des blocs a un impact sur la sécurité et sur la complexité : les blocs de grandes dimensions sont plus sécuritaires mais sont plus lourds à implémenter. Pour avoir un chiffrement et un déchiffrement unique, il faut que la fonction de chiffrement

soit une fonction un-à-un , Les chiffrements par blocs sont aussi utilisés dans des systèmes à clé publique.

❖ **Transformation :**

- **Chiffrement par substitution :**

Les substitutions consistent à remplacer des symboles ou des groupes de symboles par d'autres symboles ou groupes de symboles dans le but de créer de la confusion, le chiffrement par substitution offre un très grand choix de clés différentes. Par exemple, pour un alphabet de 26 lettres et considérant que l'alphabet des messages clairs et des messages chiffrés est le même, il y a 26! clés possibles, soit 291 461 126 605 635 584 000000, Cependant, à l'aide d'une analyse statistique des symboles du message chiffré, il est possible de casser les algorithmes de substitution. La faiblesse est que chaque symbole d'un message clair est chiffré de la même façon : un symbole donné est toujours remplacé par le même symbole. Il suffit de chercher le symbole le plus fréquent dans le message chiffré et le remplacé par le symbole le plus fréquent dans le langage du texte clair. Par exemple le "e" est le symbole le plus utilisé dans les textes français et anglais.

- **Chiffrement par transposition :**

Les transpositions consistent à mélanger les symboles ou les groupes de symboles d'un message clair suivant des règles prédéfinies pour créer de la diffusion. Ces règles sont déterminées par la clé de chiffrement. Une suite de transpositions forment une permutation. On appelle "transposition avec expansion" une transposition dans laquelle les mêmes caractères sont utilisés plusieurs fois. Une analyse statistique sur les chiffrements par transposition n'est pas utile. Seul l'ordre des symboles est différent mais les symboles restent les mêmes : les symboles les plus fréquents dans le message clair resteront les plus fréquents dans le message chiffré.

Néanmoins, avec leur puissance, les ordinateurs n'éprouvent aucune difficulté à solutionner ce type de chiffrement.

- **Chiffrement par produit : la combinaison des deux :**

Le chiffrement par substitution ou par transposition ne fournit pas un haut niveau de sécurité, mais en combinant ces deux transformations, on peut obtenir un chiffrement robuste. La plupart des algorithmes de cryptage par clés symétriques utilisent le chiffrement pare produit. [3]

Un « round » est complété lorsque les deux transformations ont été faites une fois (substitution et tronsposition).

La substitution dans un round est destinée à rendre complexe la relation entre la clé et le message chiffré alors que la transposition mélange et étend les redondances du message clair dans le message chiffré. Exemples détaillés d'algorithmes importants. Chaque algorithme est décrit en trois sections principales : l'introduction, l'algorithme (expansion de la clé s'il y a lieu, chiffrement et déchiffrement) et les considérations. Blowfish, DES , IDEA , RC2, RC5, RC6 , Rijndael , TripleDES .

- **Chiffrements de flux :**

Les algorithmes de chiffrement de flux (streamciphers) peuvent être définis comme étant des algorithmes de chiffrement par blocs, où le bloc a une dimension unitaire (1 bit, 1 octet, etc.) ou relativement petite. Leurs avantages principaux viennent du fait que la transformation (méthode de chiffrement) peut être changée à chaque symbole du texte clair et du fait qu'ils soient extrêmement rapides. De plus, ils sont utiles dans un environnement où les erreurs sont fréquentes car ils ont l'avantage de ne pas propager les erreurs (diffusion). Ils sont aussi utilisés lorsque l'information ne peut être traitée qu'avec de petites quantités de symboles à la fois, comme par exemple si l'équipement n'a pas de mémoire physique ou une mémoire tampon très limitée.

Ils appliquent de simples transformations selon un keystream utilisé. Lekeystream est une séquence de bits utilisée en tant que clé qui peut être générée aléatoirement, ou par un algorithme qui les génère aléatoirement (keystreamgenerator). Avec un keystream choisi aléatoirement et utilisé qu'une seule fois, le texte chiffré est excessivement sécuritaire. En fait, les chiffrements de flux sont une approximation des propriétés théoriques de l'algorithme one time pad, appelé aussi chiffrement Vernam (pour son auteur).
La génération du keystream peut être indépendant du texte clair et du texte chiffré, appelée chiffrement de flux synchrone (synchronousstreamcipher), ou elle peut être dépendante (self-synchronizingstreamcipher).

Les chiffrements de flux les plus répandus sont synchrones ,Les chiffrements par blocs peuvent être utilisés en chiffrements de flux en les utilisant avec certains modes d'opération, par exemple les modes CFB et OFB, À noter qu'il n'y a aucun standard parmi les chiffrements de flux . [4]

- **Linear Feedback Shift Registre :**

Le Linear Feedback Shift Register (LFSR) est un mécanisme très souvent utilisé dans les chiffrements symétriques de flux. Il génère des séquences de bits pseudo-aléatoires. La série de bits (le registre, de l'anglais register) est initialisée par un vecteur d'initialisation qui est la plupart du temps la clé du chiffrement.

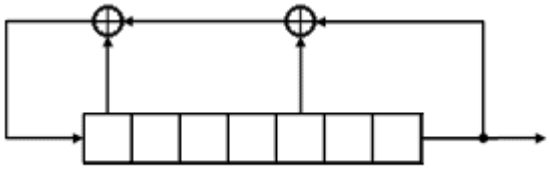


Figure 2: Mécanisme Linear Feedback Shift Register .

Le comportement du vecteur "registre" est défini par rapport à un compteur : à chaque itération de la boucle, le contenu du registre est décalé vers la droite d'une position et l'opération de l'OÙ-Exclusif est appliquée sur un sous-ensemble de bits (choisi selon l'algorithme), dont le résultat est placé à l'extrême gauche du registre. À la fin de chaque itération, un bit de sortie est généralement gardé pour former le régistre transformé résultant, Les LFSR sont très rapides et faciles d'implémentation autant dans des logiciels que sur du matériel. Cependant, utilisés seuls sans algorithme de chiffrement plus sécuritaire, ils sont vulnérables aux attaques avec la puissance des ordinateurs actuels.

- **Algorithmes importants :**

Les chiffrements de flux sont beaucoup moins nombreux que les chiffrements par blocs. Cependant, leur popularité est croissante avec la quantité toujours grandissante d'informations circulant sur les réseaux comme Internet et étant traitée par les logiciels. C'est dans ce domaine des logiciels que les streamciphers ont toute leur importance.

Seulement deux ont été relevés : RC4 et SEAL.

3.3.1.2. Systèmes à clé publique :

Le chiffrement à clé publique, aussi appelé le chiffrement asymétrique, implique une paire de clés : une clé publique et une clé privée. La clé publique est publiée dans des annuaires et ainsi connue de tous alors que la clé privée n'est connue que de la personne à qui la paire appartient.

Figure 3: schéma de système a clé publique.

Pour envoyer un texte chiffré à une personne, il faut utiliser la clé publique de cette personne et chiffrer le texte.

Une fois reçu, le destinataire utilise sa clé privée correspondante pour retrouver le message clair. Les systèmes à clé publique les plus utilisés sont RSA et PGP, Contrairement au chiffrement à clé privée, le chiffrement asymétrique requiert beaucoup d'opérations et donc n'est pas recommandé pour de grande quantité d'information.

Autre point négatif : il peut y avoir un problème d'authentification d'un destinataire. En effet, il n'y a aucune certitude à savoir qui est le véritable propriétaire d'une clé publique. Ainsi un faussaire peut tenter d'entraîner une personne à chiffrer un message avec sa clé publique, alors que cette personne croit que la clé publique appartient à une autre personne. Une société devra être mise en place pour assurer la vérification des clés publiques avec leur véritable propriétaire. [5]

- **Signatures numériques :**

Les signatures numériques (traduites parfois à tort "digitales") sont fondamentales au niveau de l'authentification, de l'identification d'entité, de l'autorisation et de la non-répudiation. Le but est de fournir des moyens à une entité de pouvoir lier son identité à une information.

Son fonctionnement général est l'inverse du système à clé publique et implique aussi la paire de clés publique/privée. Une personne voulant assurer le destinataire qu'il est bel et bien la bonne personne chiffrera un message avec sa clé privée et le destinataire déchiffrera le message chiffré avec la clé publique correspondante de l'expéditeur. Habituellement, une fonction de hachage est utilisée pour créer une empreinte du message et la transformation à l'aide de la clé privée est appliquée sur l'empreinte.

➢ **Fonctionnement :**

- L'expéditeur calcule l'empreinte de son message à l'aide d'une fonction de hachage.
- L'expéditeur chiffre l'empreinte avec sa clé privée.

- L'expéditeur chiffre l'empreinte chiffrée avec le texte clair à l'aide de la clé publique du destinataire.
- L'expéditeur envoie le message chiffré au destinataire.
- Le destinataire déchiffre le message avec sa clé privée.
- Le destinataire déchiffre l'empreinte avec la clé publique de l'expéditeur.
- Le destinataire calcule l'empreinte du texte clair à l'aide de la même fonction de hachage que l'expéditeur.
- Le destinataire compare les deux empreintes.

Les systèmes de chiffrement à clé publique peuvent habituellement aussi servir à générer des signatures numériques. Néanmoins, le standard américain est le DSS, lequel spécifie trois algorithmes : le DSA (Digital Signature Algorithm), RSA et ECDSA (EllipticCurves Digital signature algorithem). Les algorithmes de signatures numériques ne sont jamais utilisés pour le chiffrement de données. [10]

- **Factorisation :**

les systèmes fonctionnant avec la difficulté de factorisation sont les plus populaires. En fait, ils sont fondés sur la difficulté de factoriser des grands nombres qui sont le produit de deux grands nombres premiers. Multiplier deux grands nombres premiers est une fonction à sens unique; il est facile de multiplier deux nombres pour obtenir un produit, mais difficile de factoriser ce produit et de retrouver les deux grands nombres premiers.

4. La cryptanalyse :

4.1. Définition :

La cryptanalyse est la science qui consiste à tenter de déchiffrer un message ayant été chiffré sans posséder la clé de chiffrement. Le processus par lequel on tente de comprendre un message en particulier est appelé une attaque.

Une attaque est souvent caractérisée par les données qu'elle nécessite :

Attaque sur texte chiffré seul (ciphertext-only en anglais) : le cryptanalyste possède des exemplaires chiffrés des messages, il peut faire des hypothèses sur les messages originaux qu'il ne possède pas. La cryptanalyse est plus ardue de par le manque d'informations à disposition.

Attaque à texte clair connu (known-plaintext attack en anglais) : le cryptanalyste possède des messages ou des parties de messages en clair ainsi que les versions chiffrées. La cryptanalyse linéaire fait partie de cette catégorie.

Attaque à texte clair choisi (chosen-plaintext attack en anglais) : le cryptanalyste possède des messages en clair, il peut créer les versions chiffrées de ces messages avec l'algorithme que l'on peut dès lors considérer comme une boîte noire. La cryptanalyse différentielle est un exemple d'attaque à texte clair choisi.

Attaque à texte chiffré choisi (chosen-ciphertext attack en anglais) : le cryptanalyste possède des messages chiffrés et demande la version en clair de certains de ces messages pour mener l'attaque.

4.2. Familles d'attaques cryptanalytiques :

Il existe plusieurs familles d'attaques cryptanalytiques, les plus connues étant l'analyse fréquentielle, la cryptanalyse différentielle et la cryptanalyse linéaire.

4.2.1.L'analyse fréquentielle :

L'analyse fréquentielle, découvert au ixe siècle par Al-Kindi, examine les répétitions des lettres du message chiffré afin de trouver la clé. Elle est inefficace contre les chiffrements modernes tels que IDEA, RSA. Elle est principalement utilisée contre les chiffrements mono-alphabétiques qui substituent chaque lettre par une autre et qui présentent un biais statistique.

4.2.2. L'indice de coïncidence :

L'indice de coïncidence, inventé en 1920 par William F. Friedman, permet de calculer la probabilité de répétitions des lettres du message chiffré. Il est souvent couplé avec l'analyse fréquentielle. Cela permet de savoir le type de chiffrement d'un message (chiffrement mono-alphabétique ou poly-alphabétique) ainsi que la longueur probable de la clé. [8]

4.2.3. L'attaque par mot probable :

L'attaque par mot probable consiste à supposer l'existence d'un mot probable dans le message chiffré. Il est donc possible d'en déduire la clé du message si le mot choisi est correct. Ce type d'attaque a été mené contre la machine Enigma durant la Seconde Guerre mondiale.

4.2.4. L'attaque par dictionnaire :

L'attaque par dictionnaire consiste à tester tous les mots d'une liste comme mot clé. Elle est souvent couplée à l'attaque par force brute.

4.2.5. L'attaque par force brute :

L'attaque par force brute consiste à tester toutes les solutions possibles de mots de passe ou de clés. C'est le seul moyen de récupérer la clé dans les algorithmes les plus modernes et encore inviolés comme AES. Il est peu utilisé pour des mots de passe possédant un très grand

nombre de caractères car le temps nécessaire devient alors trop important. De même plusieurs brevets rendent cette méthode inefficace, comme celui de Bell ou d'IBM.

4.2.6. Attaque par paradoxe des anniversaires :

Le paradoxe des anniversaires est un résultat probabiliste qui est utilisé dans les attaques contre les fonctions de hachage. Ce paradoxe permet de donner une borne supérieure de résistance aux collisions d'une telle fonction. Cette limite est de l'ordre de la racine de la taille de la sortie, ce qui signifie que, pour un algorithme comme MD5 (empreinte sur 128 bits), trouver une collision quelconque avec 50 % de chance nécessite 264 hachages d'entrées distinctes

4.3.Cryptanalyse moderne :

Dès les années 70 apparaissent les méthodes de chiffrement modernes par blocs comme DES. Il sera passablement étudié et attaqué ce qui mènera à des attaques majeures dans le monde de la cryptographie. Les méthodes présentées ci-dessous ne sont pas vraiment génériques et des modifications sont nécessaires pour attaquer un type de chiffrement donné.

Souvent, on ne s'attaque pas à une version complète de l'algorithme de chiffrement mais une variante avec moins de tours (dans le cas des schémas de type Feistel ou les fonctions de hachage). Cette analyse préliminaire, si elle permet de déceler des vulnérabilités, laisse entrevoir une attaque sur l'algorithme complet. [8]

4.3.1. Cryptanalyse linéaire :

La cryptanalyse linéaire, due à Mitsuru Matsui, consiste à faire une approximation linéaire de la structure interne de la méthode de chiffrement. Elle remonte à 1993 et s'avère être l'attaque la plus efficace sur DES. Les algorithmes plus récents sont insensibles à cette attaque.

4.3.2. Cryptanalyse différentielle :

La cryptanalyse différentielle est une analyse statistique des changements dans la structure de la méthode de chiffrement après avoir légèrement modifié les entrées. Avec un très grand nombre de perturbations, il est possible d'extraire la clé. Cette attaque date de 1990 (présentée à la conférence Crypto 90). Elle est due à Eli Biham et Adi Shamir. Toutefois, on sait maintenant que les concepteurs de DES connaissaient une variante de cette attaque nommée attaque-T. Les algorithmes récents (AES, IDEA, etc.) sont conçus pour résister à ce type d'attaque. Les attaques différentielles sont aussi possibles sur les fonctions de hachage,

moyennant des modifications dans la conduite de l'attaque. Une telle attaque a été menée contre MD5.

4.3.3. Cryptanalyse différentielle-linéaire :

Introduite par Martin Hellman et Langford en 1994, la cryptanalyse différentielle-linéaire combine les deux principes. L'attaque différentielle produit une approximation linéaire de l'algorithme. Avec cette attaque, Hellman et Langford ont pu attaquer un DES de 8 rondes avec seulement 512 textes en clair et quelques secondes sur un PC de l'époque. Cette méthode a également été employée pour trouver des clés faibles dans IDEA. Ce type de cryptanalyse a été améliorée par Eli Biham en 2002.

4.3.4. Cryptanalyse χ^2 :

La cryptanalyse χ^2, concept dû à Serge Vaudenay, permet d'obtenir des résultats similaires à des attaques linéaires ou différentielles. L'analyse statistique associée permet de s'affranchir des défauts de ces dernières en évitant d'avoir à connaître le fonctionnement exact du chiffrement.

4.3.5. Cryptanalyse quadratique :

La cryptanalyse quadratique est une invention récente de Nicolas Courtois et Josef Pieprzyk. Cette attaque (nommée attaque XSL) vise en particulier AES et les autres chiffrements basés sur Rijndael. L'attaque XSL est le sujet de beaucoup de controverses quant à sa véritable efficacité de par sa nature heuristique. Elle consiste à résoudre un système d'équations de très grande taille.

4.3.6. Cryptanalyse moduleo n :

Suggérée par Bruce Schneier, David Wagner et John Kelsey en 1999, cette technique consiste à exploiter les différences de fonctionnement (selon une congruence variable) des algorithmes qui utilisent des rotations binaires.

4.4. propriétés analysées :

Certaines propriétés observées dans les algorithmes de chiffrement ne mènent pas forcément à des attaques mais permettent de déceler des faiblesses dans la conception, problèmes qui peuvent en cacher d'autres plus importants

4.4.1. Les clés fiables :

Certains algorithmes sont susceptibles d'avoir des clés dites faibles. Si une telle clé est utilisée pour chiffrer un message une première fois et que l'on rechiffre le résultat, toujours avec la même clé, alors on obtient le message en clair.

4.4.2. Biais statistique :

On peut chercher si la structure de chiffrement produit des biais statistiques. En général, un algorithme de chiffrement est censé produire un résultat proche d'un générateur de nombres aléatoires uniformément distribués, de manière à donner le moins d'information possible et maximiser l'entropie. Si un biais est observé (par exemple, on observe plus de bits à 1 que de bits à 0) alors des analyses supplémentaires peuvent parfois permettre de concevoir une attaque. Citons entre autres des attaques sur RC6 dont les permutations s'écartent sensiblement des caractéristiques normalement observées dans les générateurs de nombres pseudo-aléatoires. [9]

4.5. Attaques :

4.5.1. L'attaque à texte chiffré seulement (ciphertext-only attack) :

Le cryptanalyste dispose du texte chiffré de plusieurs messages, tous ayant été chiffrés avec le même algorithme. La tâche du cryptanalyste est de retrouver le plus grand nombre de messages clairs possibles, ou mieux encore de retrouver la ou les clefs qui ont été utilisées, ce qui permettrait de déchiffrer d'autres messages chiffrés avec ces mêmes clé.

4.5.2. L'attaque à texte clair connu (known-plaintext attack) :

Le cryptanalyste a non seulement accès aux textes chiffrés de plusieurs messages, mais aussi aux textes clairs correspondants. La tâche est de retrouver la ou les clefs qui ont été utilisées pour chiffrer ces messages ou un algorithme qui permet de déchiffrer d'autres messages chiffrés avec ces mêmes clefs.

4.5.3. L'attaque à texte clair choisi (chosen-plaintext attack) :

Le cryptanalyste a non seulement accès aux textes chiffrés et aux textes clairs correspondants, mais de plus il peut choisir les textes en clair. Cette attaque est plus efficace que l'attaque à texte clair connu, car le cryptanalyste peut choisir des textes en clair spécifiques qui donneront plus d'informations sur la clef.

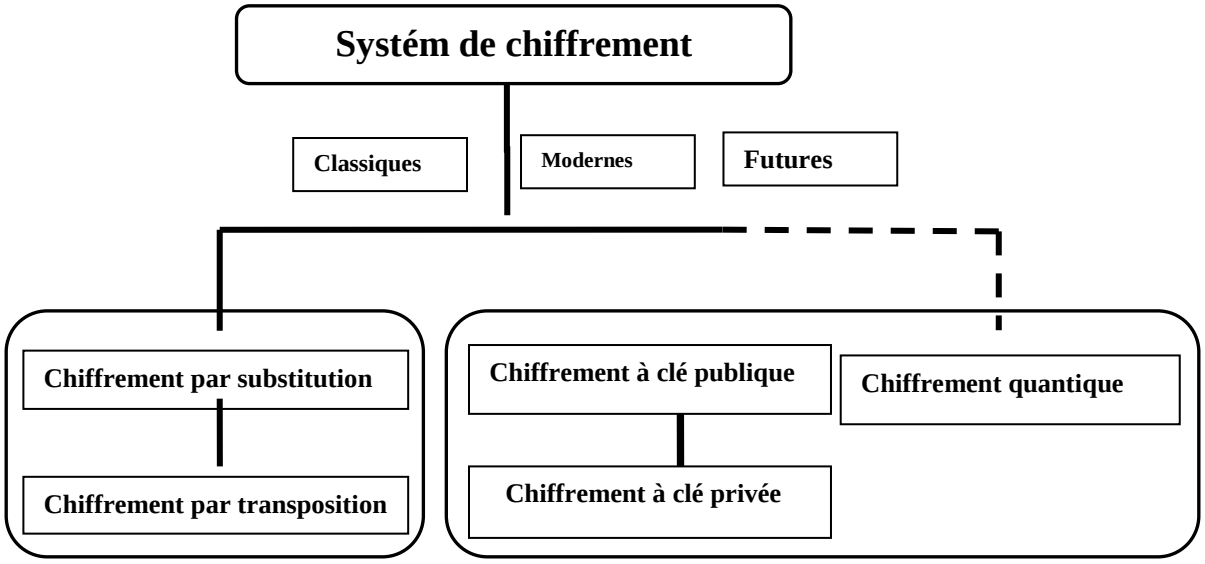
4.5.4. L'attaque à texte chiffré choisi (chosen-ciphertext attack) :

Le cryptanalyste peut choisir différents textes chiffrés à déchiffrer. Les textes déchiffrés lui sont alors fournis. Par exemple, le cryptanalyste a un dispositif qui ne peut être désassemblé et qui fait du déchiffrement automatique. Sa tâche est de retrouver la clef.

5. Classification des algorithmes cryptographique :

Comme nous venons de le voir dans la partie précédente de nombreuses méthodes de chiffrement différentes ont été imaginées pour se protéger de la curiosité et de la malveillance

de ses ennemis depuis de nombreux siècles. On peut classer ces méthodes en trois grandes classes, comme nous le montre le schéma qui suit



Opérable.

En développement.

Figure 4 : principale Classification cryptographique.

La cryptographie classique décrit la période avant les ordinateurs. Elle traite des systèmes reposant sur les lettres et les caractères d'une langue naturelle (allemand, anglais, français, etc...). Les principaux outils utilisés remplacent des caractères par des autres et les transposent dans des ordres différents. Les meilleurs systèmes (de cette classe d'algorithmes) répètent ces deux opérations de base plusieurs fois. Cela suppose que les procédures (de chiffrement ou déchiffrement) soient gardées **secrètes** ; et sans cela comme nous l'avons déjà dit le système est complètement inefficace (n'importe qui peut déchiffrer le message codé). On appelle généralement cette classe de méthodes : le chiffrement à usage **restreint**.

Les méthodes utilisées de nos jours sont plus complexes, cependant la philosophie reste la même. La différence fondamentale est que les méthodes modernes (les algorithmes, puisque l'on utilise maintenant des ordinateurs) manipulent directement des bits (liés à l'implantation sur les machines) contrairement aux anciennes méthodes qui opéraient sur des caractères alphabétiques. Ce n'est donc qu'un changement de taille (ou de représentation), puisque l'on utilise plus que deux éléments au lieu des 26 lettres de l'alphabet. La plupart des

bons systèmes de cette catégorie combinent toujours des substitutions et des transpositions, et les règles sont connues de tous, c'est pourquoi on appelle cette classe : le chiffrement à usage **général**. La sécurité de ces méthodes reposent maintenant sur un nouveau concept clé : les **clés** (pour faire un mauvais jeu de mot).

Comme on l'a déjà énoncé les moyens de chiffrement évoluent tous les jours, c'est pourquoi il est possible que les standards d'aujourd'hui ne soient plus les standards de demain. En montrant les principaux axes de recherches en cours actuellement, on peut présumer de ce que sera le futur demain, ou du moins imaginer quelles seront les améliorations des systèmes déjà en place. [9]

5.1 Le chiffrement classique :

Il existe des centaines de façon de chiffrer des données représentées par l'alphabet classique, tout en gardant les opérations réalisées secrètes. Ici on ne va pas présenter toutes ces méthodes, mais plutôt les concepts mathématiques (connus depuis très longtemps) qui sont à la source de celles-ci. On va ainsi voir que finalement il n'y en a pas tant que l'on pouvait le penser, et surtout qu'elles sont extrêmement **simples**.

5.1.1. Substitution :

La substitution consiste effectuer des dérivations pour que chaque caractère du message chiffré soit différent des caractères du message en clair. Le destinataire légitime du message applique la dérivée inverse au texte chiffré pour recouvrer le message initial. La complexité des systèmes à substitutions dépend de trois facteurs :

- la composition spécifique de l'alphabet utilisé pour chiffrer ou pour communiquer,
- le nombre d'alphabets utilisés dans le cryptogramme,
- la manière spécifique dont ils sont utilisés.

On distingue couramment quatre types de substitutions différentes :

- **Substitution simple ou substitution mono alphabétique :**

Chaque caractère du texte en clair est remplacé par un caractère correspondant dans le texte chiffré. Les exemples les plus célèbres sont les algorithmes deCésar, Rot13, et bien évidemment le code morse. Ils sont encore utilisés aujourd'hui pour cacher le sens de certains messages (par exemple la solution de certains jeux dans des journaux), mais bien sûr elles sont très peu sûrs. En effet avec ce principe, les lettres les plus fréquentes dans le texte en clair restent les plus fréquentes dans le texte chiffré, il ne cache donc pas les fréquences d'apparition des caractères. C'est une faiblesse importante puisque des techniques statistiques

peuvent être utilisés pour associer aux lettres les plus fréquentes, une lettre probable et en appliquant une technique sémantique récursive, les algorithmes à base de substitutions **mono alphabétiques** sont facilement cassés par les spécialistes.

Exemple : texte en clair = « NONJE NE SUIS PAS FOU »

Texte chiffré **(avec 5 division)** = « ABAWR ARFHV FCNFS BH »

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
N O P Q R S T U V W X Y Z A B C D E F G H I J K L M

Figure 5: Exemple du texte chiffré substitution mono alphabétique.

- **Substitution homophonique :**

comme pour le principe précédent, sauf qu'à un caractère du texte en clair on fait correspondre plusieurs caractères dans le texte chiffré. Par exemple, " A " peut correspondre à 5, 13, 25 ou 56 ; " B " 7, 19, 31, ou 42 ; etc. Ce procédé est plus sûr , mais aussi craqué par les cryptanalystes ou des espions expérimentés.

Exemple : texte en clair = « CHANGEONS LES MENTALITES FRANCAISES »

texte chiffré = «  »

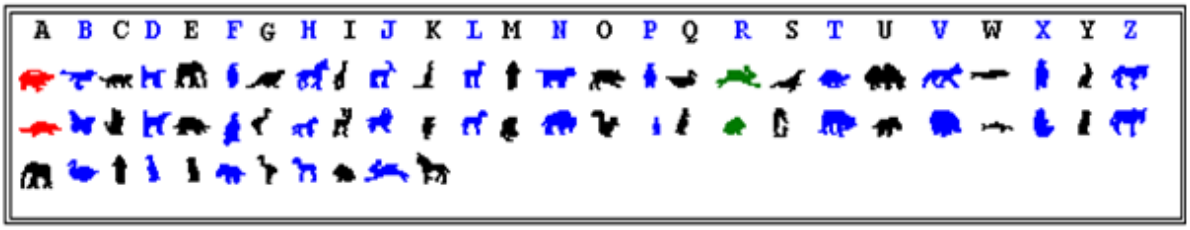


Figure 6: dictionnaire de Substitution homophonique.

- **Substitution poly alphabétique :**

le principe consiste à remplacer chaque lettre du message en clair par une nouvelle lettre prise dans ou plusieurs alphabets aléatoires associés. Par exemple, on pourra utiliser n substitutions monoalphabétiques ; celle qui est utilisée dépend de la position du caractère à

chiffrer dans le texte en clair. On choisit une clé qui sert d'entrée dans la grille polyalphabétique incluant autant de symboles qu'il y a de lettres différentes à chiffrer. Chaque caractère de la clé désigne une lettre particulière dans la grille de codage. Pour coder un caractère, on doit lire le caractère correspondant du texte en clair en utilisant la grille polyalphabétique et le mot clé associé dans l'ordre séquentiel (on répète la clé si la longueur de celle-ci est inférieure à celle du texte de départ). L'exemple le plus célèbre est l'algorithme de VIGENERE et de BEAUFORT. L'illustration la plus simple qui corresponde à ce principe est l'utilisation d'une fonction à base de ou exclusif (XOR)

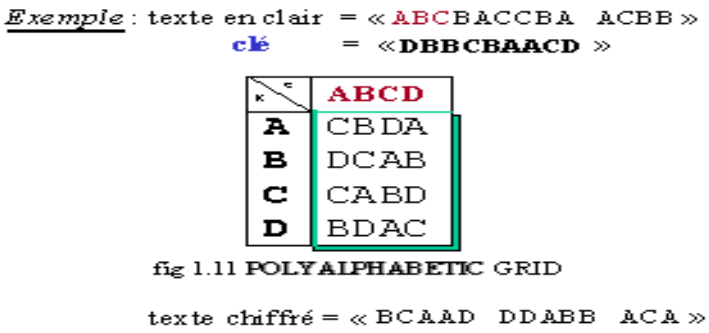


Figure 7: Substitution polyalphabétique Grid.

- **Substitution par polygrammes :**

les caractères du texte en clair sont chiffrés par blocs. Par exemple, " ABA " peut être chiffré par " RTQ " tandis que " ABB " est chiffré par " SLL ". Les exemples les plus célèbres sont les algorithmes de PLAYFAIR et de HILL inventés en 1854 et utilisés pendant le première guerre mondiale par les anglais.

5.1.2. Transposition :

Avec le principe de la transposition toutes les lettres du message sont présentes, mais dans un ordre différent. Il utilise le principe mathématique des **permutations**. Plusieurs types différents de transpositions existent :

- **Transposition simple par colonnes :**

on écrit le message horizontalement dans une matrice prédéfinie, et on trouve le texte à chiffrer en lisant la grille verticalement (cf. la figure ci-dessous). Le destinataire légal pour décrypter le message réalise le procédé inverse. L'algorithme allemand ADFGFX est fondé sur ce principe et fut utilisé pendant la première guerre mondiale. Il fut cassé par une jeune étudiante française. [11]

- **Transposition complexe par colonnes :**

un mot clé secret (avec uniquement des caractères différents) est utilisé pour dériver une séquence de chiffres commençant à 1 et finissant au nombre de lettres composant le mot clé. Cette séquence est obtenue en numérotant les lettres du mot clé en partant de la gauche vers la droite et en donnant l'ordre d'apparition dans l'alphabet. Une fois que le séquence de transposition est obtenue, on chiffre en écrivant d'abord le message par lignes dans un rectangle (comme le dessin ci-dessous le montre), puis on lit le texte par colonnes en suivant l'ordre déterminé par la séquence.

- **Transposition par carré polybique :**

un mot clé secret est utilisé pour construire un alphabet dans un tableau. Les coordonnées des lignes et des colonnes correspondant au lettres du texte à chiffrer sont utilisés pour transcrire le message en chiffres. Avec ce procédé chaque lettre du texte en clair est représenté par deux chiffres écrit verticalement. Ces deux coordonnées sont ensuite transposées en les recombinant par deux sur la ligne ainsi obtenue.

Il est important de faire remarquer que les transpositions sont plus contraignantes que les substitutions, car elles ont besoin de plus de mémoire et ne fonctionnent que sur des messages à chiffrer d'une longueur limitée ; c'est pourquoi elles sont moins utilisées dans les algorithmes bien que pourtant un peu plus sûres que les substitutions

5.2. Le chiffrement moderne :

Les algorithmes de chiffrement contemporains sont peu sûrs en général ; le cassage du programme FANTASIA pendant la seconde guerre mondiale reposant à la fois sur des transpositions et des substitutions combinées atteste de la vulnérabilité de ces techniques.

Le chiffrement moderne utilise la puissance des ordinateurs modernes. Comme les données traitées par les ordinateurs sont uniquement sous forme numériques (bits), les procédés de substitutions et de transpositions sont toujours utilisées mais maintenant seulement sur deux éléments primaires (0 et 1). On constate donc que les idées et principes vues précédemment reste d'actualité, c'est pourquoi il était à notre avis intéressant de les mettre en valeur tout de suite, ce qui est généralement jamais fait à notre regret.

Ce changement de dimension rend plus sûr les techniques de chiffrement actuelles. Certaines sont incassables, ou du moins prendraient des millions d'années avec la puissance actuelle de nos meilleurs super-calculateurs. D'autre part il fait que maintenant les algorithmes ne sont plus cachés mais au contraire sont connus de tous. Les clés sont leur sécurité. [1]

Comme nous l'avons déjà rapidement vu on distingue deux types d'algorithmes à clés : les systèmes de chiffrement symétriques et les systèmes asymétriques.

5.2.1. Chiffrement symétrique :

Les systèmes symétriques sont synonymes de systèmes à clés secrètes. Une même clé est utilisé pour le chiffrement et le déchiffrement, d'ou l'obligation que celle-ci reste confidentielle, sous peine de rendre le système inéfficient.

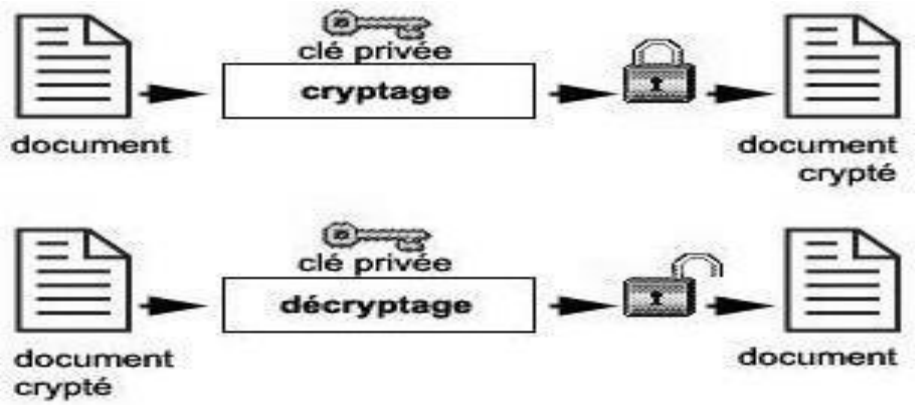


Figure 8 :La cryptographie symétrique.

- **Théorie :**

L'émetteur (Alice) et le destinataire (Bob) doivent se mettre d'accord préalablement sur la clé (k) à utiliser, pour ceci ils ne doivent pas utiliser le réseau de communication standard qui est susceptible d'être espionné (par Oscar). Chaque fois qu'Alice veux transmettre un message (m) à Bob, elle utilise sa clé secrète pour chiffrer (c=Ek(m)), et elle envoie le résultat de ce chiffrement par l'intermédiaire du même canal. Bob utilise à son tour la même clé secrète et le même algorithme public pour déchiffrer le message codé qu'il a reçu.

Les problèmes de cette technologie sont les suivants :

- si la clé secrète est compromise (volée, extorquée, piratée, ...) par un opposant, alors ce dernier pourra déchiffrer tous les messages encodés avec celle-ci. Oscar peut même se faire passer pour Alice ou Bob.

- les clés doivent être distribuées secrètement : c'est très difficile à l'échelle planétaire (se rencontrer, utiliser un messager sûr, etc...).

- si une clé différente est utilisée pour chaque paire différentes d'utilisateurs du réseau, le nombre total des clés augmente très rapidement en fonction du nombre total d'utilisateurs.

- **The Data Encryption Standard :**

Le D.E.S. est un standard mondial depuis plus de 15 ans. Il a été développé en 1976 par IBM pour le N.B.S. (National Bureau of Standards) avec pour objectif de fournir un nouveau standard de fait faisant à limiter la prolifération d'algorithmes différents qui ne pouvaient pas communiquer entre eux. Bien qu'il montre des signes de vieillesse, il a remarquablement bien résisté à des années de cryptanalyse et il est toujours sûr contre tous les adversaires excepté peut-être les plus puissants. Il est devenu le système de chiffrement le plus utilisé dans le monde. Depuis son adoption, le D.E.S. a été réévalué environ tous les cinq ans, et sa plus récente version date de Janvier 1994. Ce standard devrait cependant bientôt se terminer, puisqu'il n'a été renouvelé que jusqu'en 1998.

C'est un algorithme à clé secrète, il chiffre un bloc de texte clair de 64 bits en utilisant une clé de 56 bits, pour obtenir un bloc de texte chiffré de 64 bits. Il utilise les deux grandes lois de Shannon : diffusion (en utilisant des permutations) et confusion (en utilisant des substitutions) de bits pour casser la fréquence d'apparition des lettres dans le texte en clair, et compliquer le lien entre le fichier encodé et la clé secrète utilisée. Il repose sur 16 itérations imbriquées, et avec une clé suffisamment longue et pas trop simple (pas une succession de 1 par exemple), sa résistance aux différentes attaques possibles est bonne.

Il reste très utilisé dans le domaine commercial et des banques, et il est implanté dans de nombreuses cartes de crédits dédiés (smart cards, système électronique de communication). Son principal avantage est d'offrir une vitesse de chiffrement et déchiffrement élevée. [8]

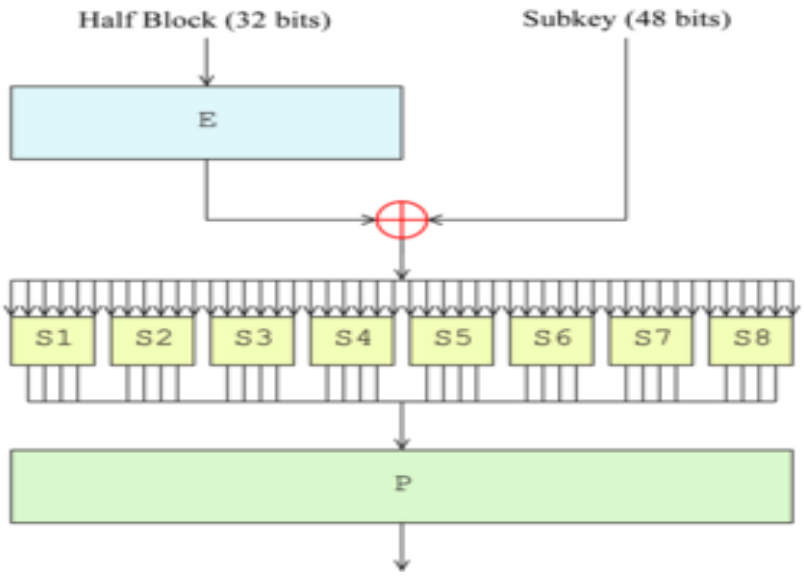


Figure 9 : schéma de DES.

- **IDEA (International Data Encryption Algorithm) :**

Développé à Zurich en Suisse par Xuejia Lai et James Massey en 1992, le chiffrement par blocs IDEA (International Data EncryptionAlgorithm) utilise des blocs de 64 bits et est contrôlé par une clé de 128 bits. Malgré le fait que IDEA n'est pas un chiffrement Feistel, le déchiffrement est effectué avec la même fonction que celle utilisée dans le chiffrement. L'algorithme a innové dans son domaine en utilisant des opérations de trois groupes algébriques différents.

Le processus de chiffrement est le même que pour le déchiffrement, à moins d'une utilisation de différentes sous-clés, ce qui est rare. En gros, le processus se résume à huit étapes de chiffrement identiques, les rounds, suivies d'une transformation au bloc de sortie. Contrairement au DES et à Blowfish, IDEA n'utilise aucune S-Box.L'algorithme peut être utilisé avec les modes d'opération ECB, CBC, CFB et OFB. Sa vitesse est environ la même que le DES.

5.2.2. Chiffrement asymétrique :

Ces algorithmes sont aussi synonymes d'algorithmes à clés publiques. Une clé différente est utilisée à la fois pour chiffrer et déchiffrer, et il est impossible de générer une clé à partir de l'autre. Il a été inventé en 1975 par deux ingénieurs en électronique : WhitfieldDiffe et Martin Hellman de l'Université de Stanford.

- **Théorie :**

Une des difficultés principales de la méthode ci-dessus est que chaque couple potentiel d'utilisateurs doit posséder sa propre clé secrète, et se l'échanger par un moyen sécurisé avant leur premier échange d'informations, ce qui peut s'avérer difficile à réaliser dans la pratique. Le but d'un système à clé publique est de résoudre ce problème. La clé publique est généralement publiée dans un répertoire. L'avantage est donc qu'Alice peut envoyer un message à Bob sans communication privée préalable (elle choisit sa clé privée, et la clé publique de Bob). Bob est la seule personne à pouvoir déchiffrer le message en appliquant sa clé secrète et personnelle, et la clé publique d'Alice. On dit généralement que chaque clé déverrouille le code produit par l'autre. Une remarque intéressante à faire est qu'avec ce système, même Alice qui a chiffré un message pour Bob, ne pourra déchiffrer le message ainsi codé. C'est un des systèmes les plus évolués que l'on peut actuellement trouver.

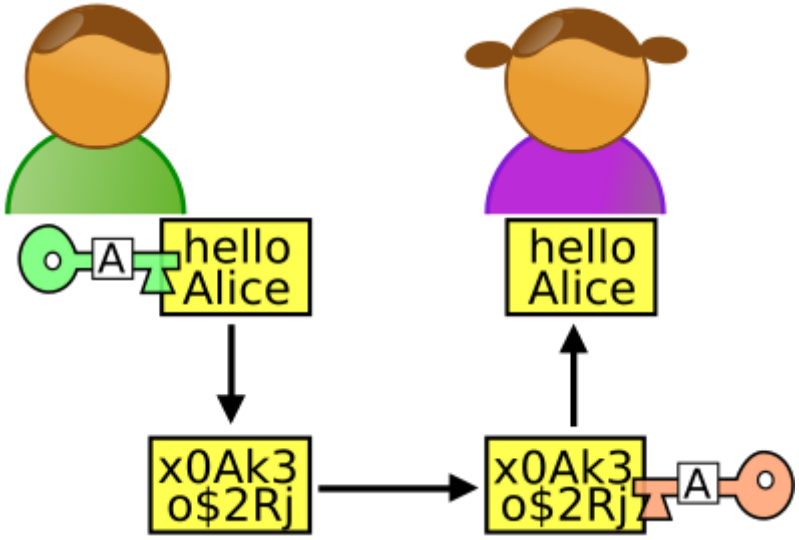


Figure 10: Chiffrement asymétrique.

- **R.S.A :**

Cet algorithme a été inventé par Rivest R., Shamir A., et Adlemann L. du M.I.T. (Massachussets Institute of Technology). C'est l'algorithme à clé publique le plus commode qui existe. Comme pour le D.E.S. sa sécurité repose sur l'utilisation de clés suffisamment longue (512 bits n'est pas assez, 768 est modérément sûr, et 1024 bits est une bonne clé). C'est la difficulté que l'on a à factoriser les entiers premiers (le problème des logarithmes discrets est souvent considérés comme insurmontable) qui font que l'on ne peut que difficilement casser cet algorithme. Cependant de larges avancées en matière de factorisation des entiers larges, ou une augmentation considérable de la puissance de nos super-calculateurs rendront RSA très vulnérable.

RSA est aujourd'hui utilisé dans une large variété de produits (téléphones, réseaux Ethernet, etc...) , de logiciels de différentes marques (Microsoft, Apple, Novell, Sun), dans des industries et enfin dans les télécommunications.

5.2.3. Chiffrement mixte et authentification :

Les algorithmes à clé publique sont assez lents. La méthode généralement utilisée pour envoyer un message, est de tirer au hasard une clé secrète, chiffrer le message avec un algorithme à clé privée en utilisant cette clé, puis chiffrer cette clé aléatoire elle-même avec la clé publique du destinataire. Ceci permet d'avoir la sécurité des systèmes à clé publique, avec la performance des systèmes à clé privée. Il existe un logiciel qui effectue toutes ces opérations et de manière transparente, et qui, de plus, est gratuit et téléchargeable à partir de

dizaines de sites de par le monde : le célèbre PGP de Phil Zimmermann. (Il y a d'autres logiciels aussi performants, mais PGP est sûrement le plus connu.). Correctement utilisé, il est sûr, même contre les meilleurs cryptanalystes du monde (c'est d'ailleurs pour cette raison que son utilisation est formellement interdite en France).

L'authentification permet de prouver son identité à travers le réseau. Il utilise généralement la technique des signatures électroniques. L'envoyeur joint une signature électronique à son message. Des explications intéressantes sur le principe de l'authentification peuvent être trouvées sur le site suivant.

5.3. Chiffrement du futur :

Tous les systèmes étudiés précédemment prenaient pour acquis que les communications numériques pouvaient être toujours espionnées d'une façon passive (c'est à dire sans détecter une modification éventuelle de l'intégrité des données échangées), ou enregistrées par un tiers pour un usage futur, même si ce dernier ne peut en comprendre le sens.

L'enregistrement d'une communication chiffrée incompréhensible peut servir à quelqu'un qui espérerait découvrir à une date ultérieure la clé secrète ou l'algorithme lui même dans le cas du chiffrement restreint, car peut-être que celui-ci après avoir accumulé suffisamment de textes chiffrés pourra plus facilement mener à terme sa cryptanalyse, ou bien par simple corruption ou espionnage découvrira la clé secrète, et sera alors en mesure de décoder tous les messages secrets accumulés.

La cryptographie quantique est née au début des années 70. Elle repose sur le principe d'incertitude d'Heisenberg, selon lequel la mesure d'un système quantique perturbe ce système. Une oreille indiscrète sur un canal de transmission quantique engendre des perturbations inévitables qui alertent les utilisateurs légitimes. Ainsi, il est possible de distribuer une clef secrète aléatoire à deux utilisateurs qui ne partagent initialement aucun secret, de façon sécurisée contre des espions même de puissance de calcul infinie. Une fois cette clef secrète établie, elle peut être utilisée avec un système cryptographique classique. La cryptographie quantique ne nécessite aucune hypothèse comme "P et NP sont distincts", ou "factoriser est difficile". On obtient ainsi des preuves de sécurité reposant uniquement sur la correction des principes quantiques. Pour plus de détails concernant les principes techniques de la cryptographie quantique, nous vous recommandons la consultation de cette page particulièrement bien faite.

Les systèmes quantiques sont toujours à un stade expérimental, cependant depuis 1992, ils ont quitté le stade de la Science-fiction depuis que Benett et Brassard, deux chercheurs américains ont construit un prototype fonctionnant sur une courte distance (un peu moins d'un kilomètre). Cette approche souffre aujourd'hui du désavantage que les transmissions quantiques sont très faibles et sont difficilement amplifiables par la route, et que la polarisation des photons posent encore des problèmes en raison de l'imperfection de l'appareil lui-même.

Seul le futur nous dira si cette nouvelle approche, un peu plus compliquée puisque reposant directement sur la physique quantique, remplacera les systèmes utilisés actuellement. Cependant, pour l'instant les réponses aux problèmes posés restent incertaines, ce qui permet encore de beaux jours aux DES, RSA et autres standards du chiffrement moderne.

5. Conclusion

Dans ce chapitre, nous avons tenté de dresser un état de l'art sur la cryptographie, et son importance dans la protection des informations pendant le transfert. Ensuite, Nous avons abordé la classification des algorithmes cryptographique avec des exemples sur chaque classe d'algorithme .Enfin, nous avons vu quelques concepts sur la cryptanalyse et les différentes techniques utilisent dans cette opération.

Le chapitre suivant est consacré à la présentation détaillé le systéme PGP que nous avons choisi.

chapitre 2 :

SYSTEM PGP

Chapitre2 : Système PGP

1. Introduction :

Après l'étude des différentes classes des algorithmes cryptographique dans le chapitre précédent. Nous prenons dans ce chapitre système PGP (Pretty Good Privacy) en détail, qui utilise deux algorithmes pour crypter les données, l'algorithme symétrique IDEA et asymétrique RSA.

Dans ce chapitre on réalise le principe de PGP et on étudie les deux algorithmes IDEA et RSA en détail puis on termine avec les avantages et les inconvénients.

2. Histoire de PGP :

Le PGP (pretty good Privacy) est un logiciel de cryptographie hybride développé par Phil Zimmermann. Il est disponible gratuitement auprès du grand public depuis 1991, plus seulement à l'armée

Phil Zimmermann est devenu la bête noire des autorités américaines qui lui reprochent d'avoir mis un outil aussi puissant à la portée de tous.

Philip Zimmermann, son développeur, a mis PGP en téléchargement libre en 1991. Violant de façon subtile les restrictions à l'exportation pour les produits cryptographiques (il avait été placé sur un site web américain d'où il était possible de le télécharger depuis n'importe où), PGP a été très mal accueilli par le gouvernement américain, qui a ouvert une enquête en 1993 abandonnée en 1996, sans donner de raison.

À l'époque où GnuPG, un logiciel libre compatible (car utilisant le même format OpenPGP), n'était pas encore très utilisé, PGP avait la réputation d'être le logiciel gratuit de cryptographie asymétrique le plus sûr au monde (en fait, PGP n'est pas un logiciel libre mais un « logiciel semi-libre »). Son code source étant ouvert (bien qu'il ne soit pas un logiciel libre), et étant toujours soutenu par son auteur Philip Zimmerman, il possède encore la confiance d'un grand nombre d'utilisateurs (en particulier envers la présence éventuelle de « portes dérobées .

Zimmermann souligne qu'il a développé PGP dans un souci de droit à la vie privée et de progrès démocratique : « PGP donne aux gens le pouvoir de prendre en main leur intimité. Il y a un besoin social croissant pour cela. C'est pourquoi je l'ai créé.

3. Le principe de PGP :

PGP est un système de cryptographie hybride, utilisant une combinaison des fonctionnalités de la cryptographie à clé publique et de la cryptographie symétrique.

Chapitre2 : Système PGP

Lorsqu'un utilisateur chiffre un texte avec PGP, les données sont d'abord compressées. Cette compression des données permet de réduire le temps de transmission par tout moyen de communication, d'économiser l'espace disque et, surtout, de renforcer la sécurité cryptographique.

La plupart des cryptanalystes exploitent les modèles trouvés dans le texte en clair pour casser le chiffrement. La compression réduit ces modèles dans le texte en clair, améliorant par conséquent considérablement la résistance à la cryptanalyse.

Ensuite, l'opération de chiffrement se fait principalement en deux étapes :

-PGP crée une clé secrète IDEA de manière aléatoire, et chiffre les données avec cette clé

-PGP crypte la clé secrète IDEA et la transmet au moyen de la clé RSA publique du destinataire.

L'opération de décryptage se fait également en deux étapes :

-PGP déchiffre la clé secrète IDEA au moyen de la clé RSA privée.

-PGP déchiffre les données avec la clé secrète IDEA précédemment obtenue.

Cette méthode de chiffrement associe la facilité d'utilisation du cryptage de clef publique à la vitesse du cryptage conventionnel. Le chiffrement conventionnel est environ 1000 fois plus rapide que les algorithmes de chiffrement à clé publique. Le chiffrement à clé publique résoud le problème de la distribution des clés. Utilisées conjointement, ces deux méthodes améliorent la performance et la gestion des clefs, sans pour autant compromettre la sécurité.

4. Algorithme RSA :

4.1. Un peu d'Histoire :

Le système de cryptage RSA a été inventé en 1978 par Ron Rivest (Cryptologue américain), Adi Shamir (Expert en cryptanalyse israélien) et Len Adleman (Chercheur en informatique et biologie moléculaire américain), d'où le sigle RSA. Ces trois auteurs avaient décidé de travailler ensemble sur le fait que le nouveau système de codage révolutionnaire de W.Diffie et M.Hellman dénommé : "Système à clé publique"(Cryptage Symétrique) était une impossibilité logique (qu'il présentait des failles). Même si ils n'ont pas réussi à l'époque à le prouvé, ils mirent en place un nouveau système de cryptage qui supplanta très vite celui de Diffie et Hellman, le RSA (le premier cryptage dit « asymétrique »).

1978 Invention du système de cryptage RSA .

1983 Le RSA est breveté par le MIT aux Etats-Unis .

21 septembre 2000 Le brevet expire et le RSA tombe dans le domaine publique .

4. 2. Le RSA aujourd'hui :

Les RSA est aujourd'hui un système universel servant dans une multitude d'applications. Au fil des années, il a supplanté tous ses concurrents, soit parce que l'on y a trouvé des points faibles, soit parce que ils n'ont pas assez été étudié pour prouver leurs efficacité face au RSA. La technologie du RSA était protéger par un brevet jusqu'en septembre 2000, aujourd'hui il est dans le domaine publique, elle a été commercialisée par plus de 350 entreprise et dans plus de 300 millions de programmes vous l'utiliser sans aucun doute tous les jours sans le savoir. On l'utilise notamment dans les transactions sécurisées sur internet, il est aussi implanter dans de nombreux standards informatique comme le Society for Worldwide Interbank Financial Télécommunication Standard, le French Financial Industry's ETEBAC-5 ou encore le X9.44 draft Standard for the U.S. Banking Industry. Le RSA est aussi implanter dans les systèmes d'exploitation de Sun, Apple, Microsoft et Nobel. Egalement intégré dans les cartes Ethernet, les Cartes à puce Bancaires, mais aussi dans grand nombre d'institutions gouvernementales, militaires et universitaires. Bref, le RSA est partout.

4.3. Principe du RSA :

4.3.1. Présentation général :

Le principe du RSA est relativement simple. Utilisons deux utilisateur classique, Alice et Bob, avec Bob qui veut envoyer un message à Alice mais en utilisant le RSA.

Alice va générer deux clés :

Une clé publique qu'elle diffusera aux personnes voulant lui parler. Cette clé sert à crypter et uniquement crypter les messages, comme nous le verront plu tard on ne peut pas décrypter les messages avec la clé publique.

Une clé privée qu'Alice gardera bien cacher des autres utilisateurs. Cette clé sert à décrypter tous les messages qui ont été crypté avec sa clé publique.

Alice envoie donc sa clé publique a Bob pour qu'il puisse lui envoyer en message crypter. Puis Alice récupère le message crypté de Bob et le décrypter à l'aide de sa clé Privé.

Illustration : Message de Bob vers Alice.

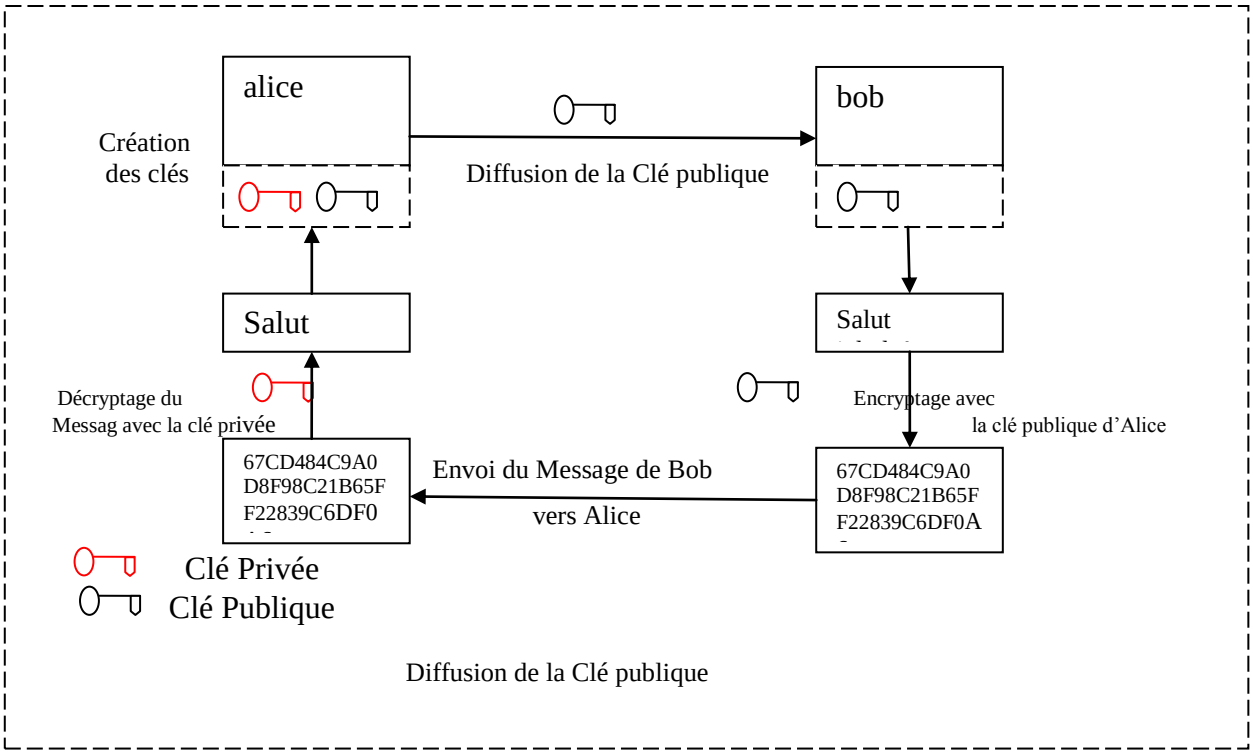


Figure 11: principe du RSA est relativement simple.

4.3.2. Génération des Clés :

Maintenant que le principe est compris passons maintenant à l'aspect mathématique du RSA, comment génère-t-on les clés, comment crypter le message et comment le décrypter.

Soit m le message en clair (non crypté).

Soit c le message encrypté.

Soit (e,n) le couple qui constitue la clé publique.

Soit (d,n) le couple qui constitue la clé privé.

- On choisit deux grands nombres premiers p et q du même ordre de grandeur.
- On calcule n : n= p*q
- On calcul e tel que e n'ai aucun facteur commun avec (p-1)(q-1).

Pour cela on va utiliser l'algorithme de Bachet - Bézout.

(Conféré : Compléments Mathématiques).

Apres avoir calculé e, on a la clé publique (e,n).

- On calcule d tel que ed mod (p-1)(q-1) = 1

(L'opération modulo (mod) est décrite dans les Compléments Mathématiques). [12]

Apres avoir calculer d, on a la clé privé (d,n).

Maintenant que nous avons nos couples de clés (publique et privé) nous pouvons encrypter nos message et les décrypter.

Pour cela on effectue les opérations suivantes :

Pour encrypter le message : c = m^e mod n

Pour décrypter le message : m = c^d mod n

- **Exemple :**

On choisit d'illustrer l'algorithme sous forme d'un exemple simple :

On prend deux nombres premier : p = 47 et q = 71

Bien sûr en réaliser ces nombres est beaucoup plus grand, 1024 chiffres dans notre réalisation en java.

On calcul n = p*q = 47 * 71 = 3337

(p-1)(q-1) = 3220

e = 79 (démonstration avec le Théorème de Bezout)

On calcul d = = 1019

On vérifie que e*d = 80501 = 1 mod 3220

Prenons un message préalablement transformé en nombre garce par exemple à la table ASCII

m = 6882326879666683

On découpe m : m1=688 m2=232 m3=687 m4=966 m5=668 m6=3

On calcul chaque bloc : cx=mxe mod n (Grace a au couple (e,n), clé publique)

On trouve c1=1570 c2=2756 c3=2091 c4=2276 c5=2423 c6=158

Le message crypter est donc : c = 15702756209122762423158

On retrouve m en calculant mx=cxd mod n (Grace a au couple (d,n), clé privé).

4.3.3. Compléments Mathématiques :

4.3.3.1. Algorithme d'Euclide :

Avant d'aborder le théorème de Bachet - Bézout il faut déjà avoir compris le l'algorithme d'Euclide, nous nous proposons donc de l'étudier rapidement ici.

L'algorithme d'Euclide sert à calculer le PGCD (Plus Grand Commun Diviseur) mais aussi à calculer les différents coefficients dans la formule de Bezout.

Le calcul du PGCD de a et b, deux nombres entiers naturels utilise la division euclidienne de a par b, tel que a=b*q+r avec r<b. Tout diviseur de et b divise r=a-b.q. Par réciproque, tout

diviseur commun de b et r divise aussi a=b.q+r. Le calcul du PGCD de a et b se ramène au calcul de b et r. En réitérant, le dernier reste non nul est le PGCD recherché.

- **Exemple :**

Prenons deux nombres : 383 et 127

383 > 127 donc : 383 = 127 *3 + 2

127 > 2 donc 127 = 2*63 + 1

Donc PGCD(383 ; 127) = 1

On rappelle que si le PGCD de deux nombre est égale à 1, alors ils sont premiers entre eux.

4.3.3.2. Théorème de Bachet - Bézout : *[Retour à Génération des Clés.]*

Le Théorème de Bachet - Bezout est très important dans le RSA car il permet de calculer e facilement. Il prouve l'existence d'une solution a*u + b*v = PGCD(a,b)

Pour le RSA on cherche à calculer e tel qu'il n'est aucun facteur commun avec (p-1)(q-1), c'est à dire que le PGCD est égale à 1 (premier entre eux).

- **Exemple :**

Reprenons l'exemple précédent avec 383 et 127.

On cherche à calculer u et v tel que 383*u + 127*v = 1

(a) 1 = 127 - 2 * 63 (Démontré précédemment avec Euclide)

(b) 2 = 383 - 127 * 3

On remplace (b) dans (a) :

1 = 127 - (383 - 127 * 3) * 63 (on développe)

1 = 127 - 383*63 + 127*189

1 = 127 (1 + 189) + 383 * (-63)

Donc on a bien 383*u + 127*v = 1 avec u= -63 et v= 190

Pour le RSA, on prendra e=v=190 car u<0.

4.3.3.3. L'opération Modulo (mod) : *[Retour à Génération des Clés.]*

L'opération Modulo est une fonction fondamentale dans l'arithmétique modulaire en mathématique utilisé par exemple dans le RSA également très utilisé en informatique car une très grande partie des calculs réalisés en informatique sont des calculs d'arithmétique modulaire.

L’opération Modulo est tout simplement le reste d’une division euclidienne. Par exemple, en divisant 5 par 2 l’on obtient 5= 2*2+1 ceux qui correspond à 5 = 1[2].

4.4. Réalisation d'un programme utilisant le RSA :

4.4.1 Présentation du programme :

4.4.1.1. But de notre programme :

Le but de notre programme est de faire une communication entre deux utilisateurs, avec chaque message crypté avec l'algorithme RSA, comme le montre le schéma ci-dessous.

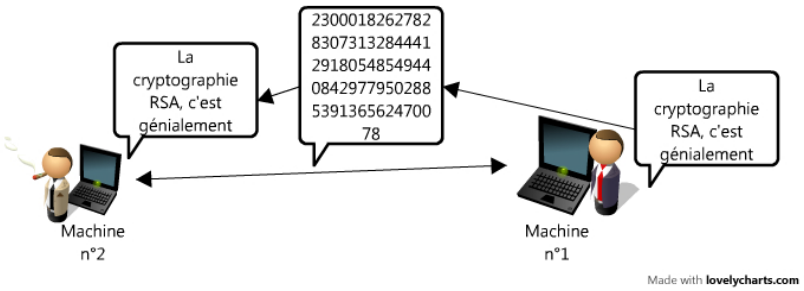


Figure 12: Schéma But de notre programme.

La machine n°1 envoie un message à la machine n°2, cette dernière le reçois comme-ci rien ne s'était passé alors qu'en réalité sur le transfert du message, celui-ci a été crypté puis restitué normalement.

Notre programme devait répondre à plusieurs critères :

- Crypté les messages à envoyer et décrypté les messages que l'on recevait
- Permettre d'afficher les messages cryptés, en console
- Le programme doit savoir gérer les caractères non connus comme &#%$£@...
- Avoir une interface graphique simple et efficace

4.4.1.2. Choix du protocole de communication :

Notre première problématique durant le développement était le choix du protocole de transfert entre l'UDP et le TCP. Effectivement, même si l'UDP offre un transfert plus rapide, celui-ci reste moins sécurisé et nos paquets peuvent se perdre. De plus le TCP établie une connexion puis échange les paquets qui non aucun chance de se perdre et/ou d'altération. Cela ressemble bien au principe d'application du RSA en général. Pour économiser des ressources de calculs, le RSA est utilisé pour établir la communication et puis un cryptage symétrique pour l'échange de paquet.

Notre programme en TCP fonctionne sur le principe d'un client-serveur. Le serveur se lance est attend une demande de connexion sur un port précis. Le client va donc demander une connexion au serveur et attendre une réponse de celui-ci. En fin de compte, une fois la connexion établis entre les 2 clients, le principe du RSA va s'appliquer, l'échange des clés va

alors se faire et ensuite les 2 machines (le serveur et le client) vont pouvoir s'échanger des messages cryptés sur le réseau.+

4.4.2. Schéma du programme :

Créer un schéma pour un programme nous permet de définir facilement les fonctions à mettre ne place dans notre programme.

Le fonctionnement de notre programme vous a été présenté, mais comment notre programme traite-t-il les informations ?

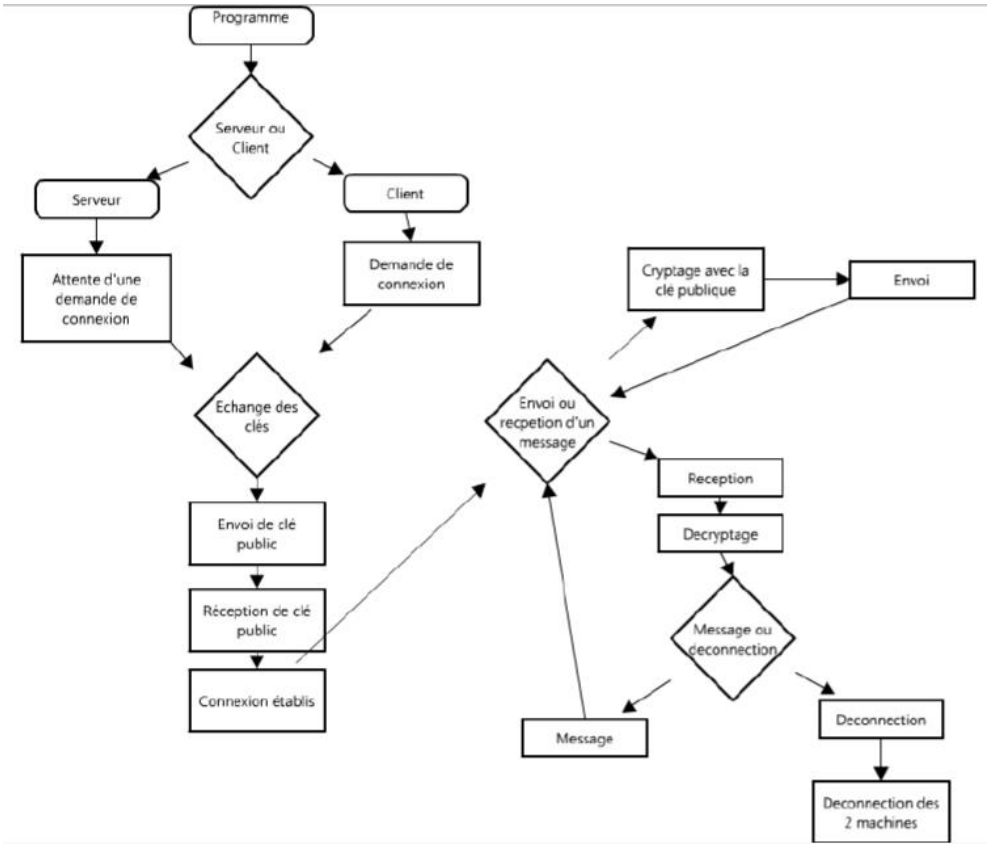


Figure 13:: Schéma du programme.

4.4.3. Réalisation en Java :

Nous avons choisi de programmer notre programme en Java car d'une part c'est le langage que nous connaissons le mieux et d'autre part, cela permet à notre programme de tourner sur tous les types de machines (Linux, Windows, IOS, …).

En java, il existe déjà des classes pour implémenter la cryptographie RSA dans un programme mais nous avons décidé de créer nos propres classes. Disponible en annexe .

4.4.3.1. Notre table ASCII :

Le problème dans notre programme c'est que nous devons traiter des chiffres en RSA mais échanger des messages en caractères, pour cela nous avons dû coder notre propre table ASCII, une variable de l'ASCII connu.

Lorsqu'on envoi notre chaine de caractères, nous devons découper la chaine pour chaque caractères ensuite chaque caractère est converti en entier avec notre table ASCII. Une fois que chaque caractère est codé en entier, l'on ajoute à la suite les nombres de chaque caractère.

Ainsi une fois notre message convertit en entier, l'on pourra appliquer différents calculs dessus, notamment ceux nécessaires pour l'algorithme RSA.

Nous avons donc vu ce que fait la classe ASCII lors de l'envoi d'un message. Mais que l'on reçoit le message, que fait-on ? D'abord, c'est la classe RSA qui va nous retourner le long entier, qui correspond à notre chaine de caractères en ASCII, ensuite nous allons découper cet entier en groupe de 2 chiffres et transforme chaque groupe de 2 en caractères.

4.4.3.2. Notre classe RSA :

Nous avons ensuite codé une classe qui va nous permettre de coder et décoder nos messages, la classe RSA.java. Cette dernière créer les clés publiques et privés et gère aussi le cryptage et décryptages des messages.

Or un autre problème rencontrée en java, est que de travailler sur de grand nombre entier est impossible avec la classe int, ni même avec la classe long.

Nous avons donc découvert la classe BigInteger et une fonction de celle-ci vraiment très intéressant qui est « probablePrime » qui permet de générer un BigInteger très grand PROBABLEMENT premier.

Création d'un entier probablement premier d'une taille définie et aléatoire :

```
BigInteger p = BigInteger.probablePrime(tailleCle, new Random());//Nombre Premier de
taille 512 p
```

On test si p et q sont premiers entre eux.

```
while ((!this.e.gcd(w).toString().equals("1")) || (this.e.compareTo(w) != -1)) { // e premier avec w
et e < w
e = new BigInteger(tailleCle + 1, new Random());
```

De plus la classe BigInteger intègre toutes les fonctions de calculs nécessaires pour créer l'algorithme RSA.

Par exemple pour le cryptage des données :

```
public static BigInteger cryptage2(BigInteger message, BigInteger e2, BigInteger n2) {
return message.modPow(e2, n2); // On crypte notre message : message(Puissance : clé public)
modulo(n = p*q)
}
```

Une fois que l'on a ce qu'il faut pour transformer une chaine de caractère en chiffre et chiffrer nos messages, il ne nous restait plus qu'à utiliser cela dans un programme graphique.

4.5. La sécurité du RSA :

- Attaque par Factorisation :

L'attaque par factorisation consiste à essayer de factoriser le module N et donc comme objectif de retrouve p et q. Avec cela, l'on pourra calculer d pour déchiffrer les messages.

La société RSA Security qui a été créée pour vendre le cryptosystèmes RSA a créé en 1991 un concours de factorisation. Les concours ont été arrêté en 2007 mais certain chercheurs essaient encore de factoriser des clés de toutes tailles.

En janvier 2010 une clé RSA de 768 bits à était cracker par factorisation, cela a coûter 2 ans et demi de recherche et de calculs, 5To de données traitées, des calculs distribués sur plus de 1700 coeurs, environ 1020 opérations.

5. Algorithme IDEA :

5.1. Le chiffrement IDEA :

• assure la sécurité de haut niveau ne se fonde pas sur le maintien de l'algorithme secret, mais plutôt sur l'ignorance de la clé secrète

• est entièrement spécifiée et facile à comprendre.

• est disponible pour tout le monde.

• est approprié pour une utilisation dans une large gamme d'applications.

• peut être mis en œuvre sur le plan économique dans les composants électroniques (VLSI Chip).

• peut être utilisé efficacement.

• peuvent être exportés dans le monde entier.

• est protégés par un brevet pour empêcher la fraude et le piratage.

5.2. Description de l'IDEA :

Le bloc de chiffrement IDEA fonctionne avec texte en clair de 64 bits et chiffrement des blocs de texte et est contrôlé par une clé de 128 bits . L' innovation fondamentale dans la conception de cet algorithme est l'utilisation des opérations algébriques à partir de trois groupes différents . Les boîtes de substitution et les tables de référence associés utilisés dans

les algorithmes de chiffrement par blocs disponibles à ce jour ont été complètement évité . La structure de l' algorithme a été choisi de telle sorte que , à l'exception que les différents sous-blocs de clé sont utilisés, le processus de chiffrement est identique au processus de décryptage.

5.3. Génération de la clé :

Le bloc de texte en clair de 64 bits est divisée en quatre sous-blocs de 16 bits , puisque toutes les opérations algébriques utilisés dans le procédé de cryptage fonctionnent sur des nombres de 16 bits . Un autre procédé produit pour chacun des tours de chiffrement , six 16 - bit clés sous-blocs de la clé de 128 bits . Depuis quatre autres 16 bits clé - sous- blocs sont nécessaires pour la transformation de la production ultérieure , un total de 52 (= 8 x 6 + 4) différents sous- blocs de 16 bits à être produite à partir de la clé de 128 bits .

Les sous- blocs de clé utilisées pour le chiffrement et le déchiffrement dans les tours individuels sont montrés dans le tableau 1 .

5.3.1. Le chiffrement des clés de sous-blocs :

Tour 1	$Z_1^{(1)}\ Z_2^{(1)}\ Z_3^{(1)}\ Z_4^{(1)}\ Z_5^{(1)}\ Z_6^{(1)}$
Tour 2	$Z_1^{(2)}\ Z_2^{(2)}\ Z_3^{(2)}\ Z_4^{(2)}\ Z_5^{(2)}\ Z_6^{(2)}$
Tour 3	$Z_1^{(3)}\ Z_2^{(3)}\ Z_3^{(3)}\ Z_4^{(3)}\ Z_5^{(3)}\ Z_6^{(3)}$
tour 4	$Z_1^{(4)}\ Z_2^{(4)}\ Z_3^{(4)}\ Z_4^{(4)}\ Z_5^{(4)}\ Z_6^{(4)}$
Tour 5	$Z_1^{(5)}\ Z_2^{(5)}\ Z_3^{(5)}\ Z_4^{(5)}\ Z_5^{(5)}\ Z_6^{(5)}$
Tour 6	$Z_1^{(6)}\ Z_2^{(6)}\ Z_3^{(6)}\ Z_4^{(6)}\ Z_5^{(6)}\ Z_6^{(6)}$
Tour 7	$Z_1^{(7)}\ Z_2^{(7)}\ Z_3^{(7)}\ Z_4^{(7)}\ Z_5^{(7)}\ Z_6^{(7)}$
Tour 8	$Z_1^{(8)}\ Z_2^{(8)}\ Z_3^{(8)}\ Z_4^{(8)}\ Z_5^{(8)}\ Z_6^{(8)}$
transformée de sortie	$Z_1^{(9)}\ Z_2^{(9)}\ Z_3^{(9)}\ Z_4^{(9)}$

Tableau 1 : inverse lors du chiffrement sous-blocs.

Les 52 à 16 bits des sous-blocs de clé qui sont générés à partir de la clé de 128 bits sont produites comme suit:

• Premièrement, la clé de 128 bits est divisée en huit sous-blocs de 16 bits qui sont alors directement utilisables comme les premiers huit sous-blocs de clé.

• La clé de 128 bits est ensuite décalée cycliquement vers la gauche en 25 positions, après quoi le bloc de 128 bits qui en résulte est à nouveau divisé en huit sous-blocs de 16 bits pour être directement utilisé comme ici huit sous-blocs de clé.

• La procédure de décalage cyclique décrit ci-dessus est répété jusqu'à ce que tous les 52 à 16 bits principaux sous-blocs nécessaires ont été générés.

4.3.2. Décryptage des clés sous-blocs :

Tour 1	$Z_1^{(1)}\ Z_2^{(1)}\ Z_3^{(1)}\ Z_4^{(1)}\ Z_5^{(1)}\ Z_6^{(1)}$
Tour 2	$Z_1^{(2)}\ Z_2^{(2)}\ Z_3^{(2)}\ Z_4^{(2)}\ Z_5^{(2)}\ Z_6^{(2)}$
Tour 3	$Z_1^{(3)}\ Z_2^{(3)}\ Z_3^{(3)}\ Z_4^{(3)}\ Z_5^{(3)}\ Z_6^{(3)}$
tour 4	$Z_1^{(4)}\ Z_2^{(4)}\ Z_3^{(4)}\ Z_4^{(4)}\ Z_5^{(4)}\ Z_6^{(4)}$
Tour 5	$Z_1^{(5)}\ Z_2^{(5)}\ Z_3^{(5)}\ Z_4^{(5)}\ Z_5^{(5)}\ Z_6^{(5)}$
Tour 6	$Z_1^{(6)}\ Z_2^{(6)}\ Z_3^{(6)}\ Z_4^{(6)}\ Z_5^{(6)}\ Z_6^{(6)}$
Tour 7	$Z_1^{(7)}\ Z_2^{(7)}\ Z_3^{(7)}\ Z_4^{(7)}\ Z_5^{(7)}\ Z_6^{(7)}$
Tour 8	$Z_1^{(8)}\ Z_2^{(8)}\ Z_3^{(8)}\ Z_4^{(8)}\ Z_5^{(8)}\ Z_6^{(8)}$
transformée de sortie	$Z_1^{(9)}\ Z_2^{(9)}\ Z_3^{(9)}\ Z_4^{(9)}$

Tableau .2 : Inverse lors du déchiffrement sous-blocs.

Le processus de calcul utilisée pour le décryptage du texte chiffré est essentiellement le même que celui utilisé pour le chiffrement de texte en clair . La seule différence avec le cryptage est que lors du déchiffrement , différents 16 bits principaux sous-blocs sont générés .

Plus précisément, chacune des 52 à 16 bits des sous- blocs de clé utilisée pour le déchiffrement est l'inverse de la sous-bloc de clé utilisée pour le chiffrement à l'égard de l'opération de groupe algébrique appliquée. De plus, les principaux sous-blocs doivent être utilisés dans l'ordre inverse lors du déchiffrement, afin d'inverser le processus de chiffrement , comme indiqué dans le tableau 2 .

5.4. Chiffrement :

La représentation fonctionnelle du procédé de chiffrement est représenté sur la **figure.3.** L'opération s'effectue en huit étapes de cryptage identiques (appelées tours de chiffrement), suivie d'une transformation de sortie. La structure de la première série est représentée en détail.

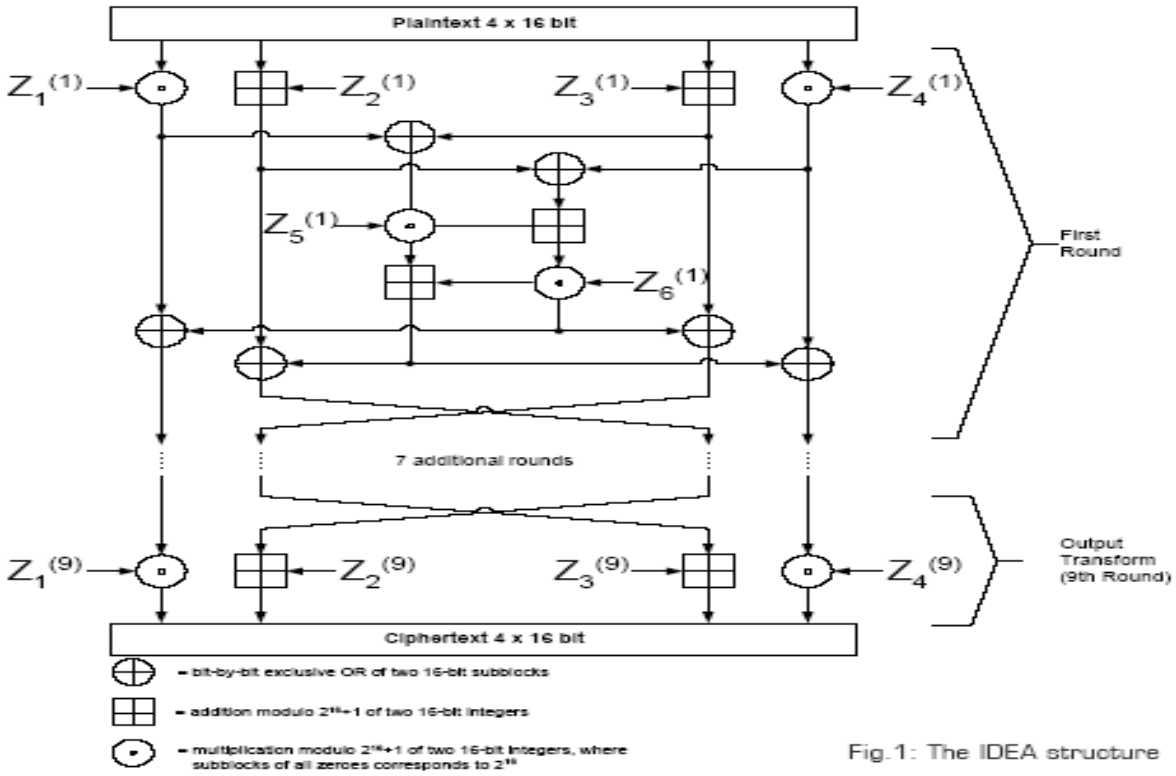


Figure 14: Schéma de Chiffrement IDEA .

Dans le premier tour de chiffrement , les premières quatre principaux sous-blocs de 16 bits sont combinées avec deux des 16 bits des blocs de texte en clair en utilisant addition modulo 216 , et avec les deux autres blocs de texte en clair en utilisant une multiplication modulo 216 + 1 . Les résultats sont ensuite traitement ultérieur comme le montre la Figure 1, dans lequel plus de deux clés à 16 bits des sous-blocs entrent dans le calcul et le troisième opérateur de groupe algébrique , le bit à bit OU exclusif , est utilisé . A la fin de la première ronde cryptage quatre valeurs 16 bits sont produits qui sont utilisés comme entrée pour le deuxième tour de cryptage dans un ordre partiellement modifié. Le processus décrit ci-dessus pour la première ronde est répété dans chacun des cryptage ultérieur 7 tours en utilisant différentes clés de 16 bits sous-blocs pour chaque combinaison . Au cours de la transformation de sortie ultérieure, les quatre valeurs de 16 bits produites à la fin du huitième cycle de chiffrement sont combinés avec les quatre premiers des 52 sous- blocs de clé en

utilisant addition modulo 216 et de la multiplication modulo 216 + 1 dernière pour former les quatre 16 résultant bits des blocs de texte chiffré.

5.5. Modes de fonctionnement :

IDEA supporte tous les modes de fonctionnement tel que décrit par le NIST dans sa publication FIPS 81 . D'un bloc de chiffrement crypte et décrypte clair en blocs de taille fixe bits (surtout 64 et 128 bits) . Pour en clair dépasser cette taille fixe , l' approche la plus simple consiste à partitionner le texte en clair en blocs de longueur égale et chiffrer séparément. Cette méthode est appelée Electronic Code Book mode (BCE) . Cependant, le code de livre électronique n'est pas un bon système pour utiliser de petites tailles de bloc (par exemple, inférieure à 40 bits) et des modes de chiffrement identiques .

Comme la BCE a des inconvénients dans la plupart des applications, d'autres méthodes nommées modes ont été créés . Ils sont Cipher Block Chaining (CBC) , Cipher Feedback (CFB) et Output Feedback modes (OFB) . [10]

5.6. Clés faibles pour IDEA :

Selon le rapport de Daemon [6] , de grandes classes de clés faibles ont été trouvés pour le bloc algorithme de chiffrement IDEA . IDEA a une clé de 128 bits et chiffre des blocs de 64 bits. Pour une classe de 223 touches IDEA présente un facteur linéaire. Pour une certaine classe de 235 touches le chiffre a une caractéristique globale avec une probabilité de 1. Pour une autre classe de 251 touches seulement deux cryptages et la résolution d'un ensemble de 16 équations booléennes non linéaires avec 12 variables est suffisante pour tester si la clé utilisée appartient à cette classe . Si c'est le cas, en particulier sa valeur peut être calculée de manière efficace. Il est montré que le problème de la faiblesse des clés peut être éliminé en modifiant légèrement le calendrier clé de l'IDEA .

Dans [4] , deux nouvelles attaques sur un nombre réduit de tours de IDEA sont présentés : attaque différentielle tronquée et attaque différentielle linéaire . L'attaque différentielle tronquée trouve la clé secrète de 3,5 tours de IDEA dans plus de 86 % de tous les cas en utilisant un nombre estimé de 256 textes clairs choisis et une charge de travail d'environ 267 cryptages de 3,5 tours de IDEA . Avec 240 textes clairs choisis l'attaque fonctionne à 1 % de toutes les clés . L'attaque différentielle linéaire trouve la clé secrète de 3 tours d'IDEA . Il doit à la plupart des 229 paires choisies de texte en clair et une charge de travail d'environ 244 cryptages avec 3 tours d'IDEA .

5.6.1. Applications :

Aujourd'hui, il ya des centaines de solutions de sécurité basées IDEA- disponibles dans de nombreux domaines du marché , allant des services financiers , et de la radiodiffusion au gouvernement. IDEA est le nom d'un algorithme de chiffrement par bloc éprouvée , sûre et universellement applicable , ce qui permet une protection efficace des données transmises et stockées contre l'accès non autorisé par des tiers . Les critères fondamentaux pour le développement de l'IDEA ont des exigences de sécurité les plus élevés ainsi que le matériel et les logiciels mise en œuvre facile pour une exécution rapide .

L'algorithme IDEA peut facilement être intégré dans n'importe quel logiciel de cryptage . Le cryptage de données peut être utilisé pour protéger la transmission de données et le stockage. Les domaines typiques sont :

- Audio et les données vidéo pour la télévision par câble, télévision à péage, la vidéoconférence, l'apprentissage à distance , la télévision de l'entreprise , la Voie IP
- Données financières et commerciales sensibles
- E-mail via les réseaux publics
- Liaisons de transmission par modém , routeur ou lien ATM , la technologie GSM
- Les cartes à puce.

6. Avantages et Inconvénients :

Ces deux méthodes de cryptage associent la facilité d'utilisation du cryptage à clé publique à la vitesse du cryptage conventionnel.

Le cryptage conventionnel est environ 1000 fois plus rapide que le cryptage à clé publique.

De plus, le cryptage à clé publique résout non seulement le problème de la distribution des clefs, mais également de la transmission des données.

Utilisées conjointement, ces deux méthodes améliorent la performance et la distribution des clé, sans pour autant compromettre la sécurité.

7. Conclusion :

Au niveau de ce chapitre on a vus le système PGP en détail (l'historique, le principe, les avantages et les inconvénients).On a aussi étudié les deux algorithmes RSA et IDEA avec les détails (le principe, présentation, Le chiffrement), puis a la fin Les fonctionnalités de PGP . Dans le chapitre suivant on réalise l'implémentation de notre programme . [5]

chapitre 3 :

Implémentation

Chapitre 3 : Implémentation

1. Introduction :

Pour réaliser tous les applications qui nous avons vus dans le chapitre2, on avait besoin d'utiliser un langage simple a manipuler, qu'il soit un langage orienté objet, ce langage permet d'effectues toutes les taches d'un langage de haut niveau (bureautique, graphique, multimédia, base de donné, environnement de développement, etc.).

Le langage que nous avons choisis est le langage java.

2. Pour quoi on utilise java ?

Nous avons choisi Java comme langage de programmation pour réaliser notre système pour les raisons suivantes:

Maitriser un nouveau langage de programmation, je n'ai pas maîtrisé auparavant.

comprendre bien le concept orienté objet.

les exceptions qui distingue ce langage par rapport les autres langages comme la facilité (leur syntaxe est une extension de langage C)…etc.

Je pense que c'est le meilleur langage de programmation pour réaliser cette application.

2.1. Présentation générale :

Java a été créée en 1991 par Sun Microsystems afin d'offrir un langage de programmation adapté au contrôle d'appareils électroniques, dans le cadre du projet Green. Le but était de pouvoir contrôler en quelque sorte les appareils du futur, de manière à ce qu'ils deviennent interactifs et puissent communiquer entre eux. Au départ, les chercheurs de ce projet créèrent un prototype du nom de Star7, une sorte de télécommande capable de communiquer avec ses semblables. Ceux-ci désirèrent alors créer un système d'exploitation adapté à cette télécommande en C++, qui était de loin le langage de programmation orienté objet le plus utilisé en cette période. Cependant, les contraintes imposées par ce langage irrita l'un des membres de ce projet, James Gosling, et ce dernier finit par concevoir un nouveau langage plus adapté au contrôle de cet appareil, qu'il nomma Oak. Cependant, ce nom étant déjà utilisé dans le monde de l'informatique, il fut finalement convenu de lui donner le nom de Java.

Comme Java était initialement développé pour contrôler des appareils électroniques et non des ordinateurs, il présente l'avantage de produire des programmes peu encombrants (donc rapidement téléchargeables depuis une page web), fiables, performants et faciles à porter vers d'autres appareils (un programme Java fonctionne aussi bien sous Windows, Mac OS, Linux ou tout autre système d'exploitation, et ce sans aucune modification de la part du

développeur). De plus, Java est un langage sûr, qui empêche tout dégât sur une machine distante ayant lancé une applet (application prévue pour s'exécuter depuis une page web). Il faut toutefois préciser que ce n'est plus le cas lorsqu'une application autonome est générée: à ce moment, le programme sera autorisé à lire et écrire librement sur le disque de l'utilisateur, et à gérer tous les évènements que les autres langages sont en mesure d'effectuer.

2.2. Caractéristiques du langage java :

Java possède un certain nombre de caractéristiques qui ont largement contribué à son énorme succès :

- **Java est orienté objet :**

Comme la plupart des langages récents, Java est orienté objet. Chaque fichier source contient la définition d'une ou plusieurs classes qui sont utilisées les unes avec les autres pour former une application. Java n'est pas complètement objet car il définit des types primitifs (entier, caractère, flottant, booléen,...).

- **Java est simple :**

le choix de ses auteurs a été d'abandonner des éléments mal compris ou mal exploités des autres langages tels que la notion de pointeurs (pour éviter les incidents en manipulant directement la mémoire), l'héritage multiple et la surcharge des opérateurs, ...

- **Java est fortement typée :**

Toutes les variables sont typées et il n'existe pas de conversion automatique qui risquerait une perte de données. Si une telle conversion doit être réalisée, le développeur doit obligatoirement utiliser un caste ou une méthode statique fournie en standard pour la réaliser.

- **Java assure la gestion de la mémoire :**

L'allocation de la mémoire pour un objet est automatique à sa création et Java récupère automatiquement la mémoire inutilisée grâce au garbage collecter qui restitue les zones de mémoire laissées libres suite à la destruction des objets.

- **Java est sûre :**

La sécurité fait partie intégrante du système d'exécution et du compilateur. Un programme Java planté ne menace pas le système d'exploitation. Il ne peut pas y avoir d'accès direct à la mémoire. L'accès au disque dur est réglementé dans une applet.

- **Java est économe :**

Le pseudo code a une taille relativement petite car les bibliothèques de classes requises ne sont liées qu'à l'exécution.

- **Java est multitâche :**

Il permet l'utilisation de threads qui sont des unités d'exécution isolées. La JVM, elle même, utilise plusieurs threads.

3. Pour quoi NetBeans ?

On a distingué l'environnement de développement NetBeans parce que il est faciles à comprendre et à utiliser, très riche, et disponible gratuitement, pour compiler et exécuter les programmes Java, les environnements de développement intégrés peuvent nécessiter des ressources importantes et être lourds à utiliser pour de petits programmes. Comme solution intermédiaire, vous disposez des éditeurs de texte appelant le compilateur Java et exécutant les programmes Java. Lorsque vous aurez maîtrisé les techniques présentées.

4. Présentation général de NetBeans :

4.1. Définition :

NetBeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Développent 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).

Conçu en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Développement Kit JDK est requis pour les développements en Java.

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 greffons

4.2. Historique :

En 1997, NetBeans naît de Xelfi, un projet d'étudiants dirigé par la Faculté de mathématiques et de physique de l'Université Charles de Prague. Plus tard, une société se forme autour du projet et édite des versions commerciales de l'EDI NetBeans, jusqu'à ce qu'il soit acheté par Sun en 1999. Sun place le projet sous double licence CDDL et GPL v2 en juin de l'année suivante.

4.3. Caractéristiques :

✓ Interopérabilité

Il est possible d'importer des projets Eclipse vers NetBeans avec NetBeans IDE 4.1 ou supérieur, il suffit d'installer 'Eclipse Projet Importer'.

✓ Modularité

-NetBeans possède un module de facilitation et de hiérarchie classloader.

-Un module dépendance influence runtime classpath.

-Un module peut tourner sur d'autre module.

✓ Coopération de modules

-Un module possède une fenêtre, un autre une action.

-Windows peut avoir des associés contextes.

-Sélection fenêtre définit le contexte mondial.

-Eléments de l'interface utilisateur mise à jour en fonction de son classloader.

-Le menu, la barre d'outils des éléments obtenus fusionnés par le cadre de NetBeans.

-Windows Layout.

✓ Il intègre

-Ruby Support.

-Swing GUI développement.

4.4. Avantages de NetBeans :

NetBeans fournit un environnement riche pour le règlement de problèmes et l'optimisation des applications.

- Intégration de :

-Swing.

-La plut part des outils java.

-Windows System.

-Docking cadre extensions autour de Swing.

-Module Système.

- Mise à jour automatique.
- Exécuter des morceaux de code bien spécifiques à un moment donné (en utilisant les points d'arrêt comme délimiteurs).
- Suspendre l'exécution lorsqu'une condition spécifiée est rencontrée (comme par exemple quand un itérateur atteint une certaine valeur).

- Suspendre l'exécution lors d'une exception, soit à la ligne de code qui a provoqué l'exception, ou dans l'exception elle-même.
- Pister la valeur d'une variable ou d'une expression.
- Pister l'objet référencé par une variable (fixed watched).
- Fixer le code à la volée et continuer la session de débogage.
- Suspendre les threads individuellement ou collectivement.
- Revenir au début d'une méthode appelée précédemment (pop a call) dans le call stack actuelle.

Exécuter de multiples sessions de débogage en même temps. Par exemple, on pourrait recourir à cette capacité pour déboguer une application client/serveur.

5. Description de l'application :

5.1. Les classes :

Est contient deux classes :

- **IDEA,COR1,COR2:** les trois classes sont pour crypter et décrypter le texte.
- **RSA:** cette classe est pour la génération de l'ensemble clé publique/clé privé et pour crypter et décrypter les clés .

5.2 . L'interfaces graphiques :

Notre application contient une interface graphique simple facile pour l'utilisation représentée par la figure suivant :

Figure 15: Interface graphique principale

Chapitre 3 : Implémentation

Figure 16: Boutons raccourcis

1- Bouton text : pour tapez un nouveau texte dans le champ texte.

2- Bouton image: pour choisi l'image que nous allons crypter/décrypter.

3- Bouton crypter: pour entrez la clé de cryptage afin de crypter le texte .

4- Bouton decrypter: pour entrez la clé afin de décryptage le texte .

5- Bouton a propos: information sur l'application.

Exemple d'application sur un texte:

a)Le chiffrement :

1- Saisir le texte clair :

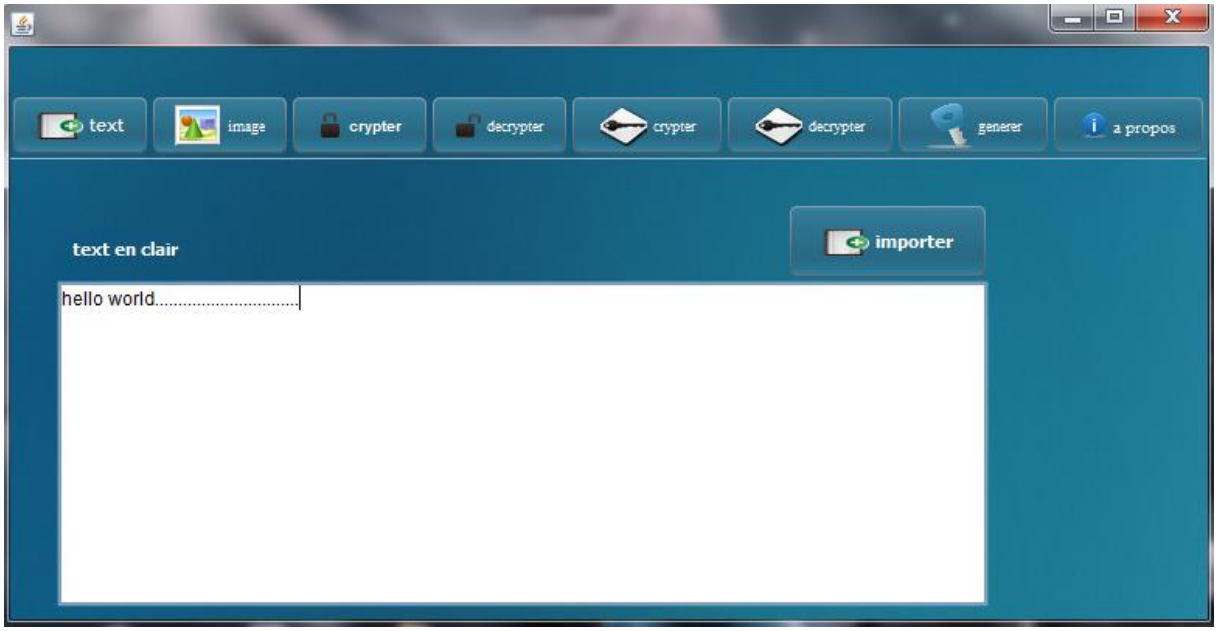


Figure 17: Texte Clair

2-Cliqué sur le bouton crypter.

3-Entrez la clé de chiffrement (16 caractères).

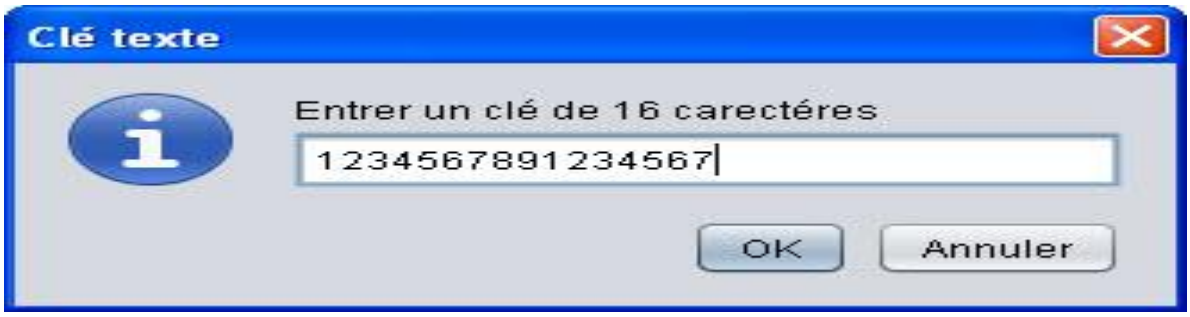


Figure 18:Le message pour.

entrer la clé dans le champ

Le résultat de cryptage sera affiché dans le même champ de saisir

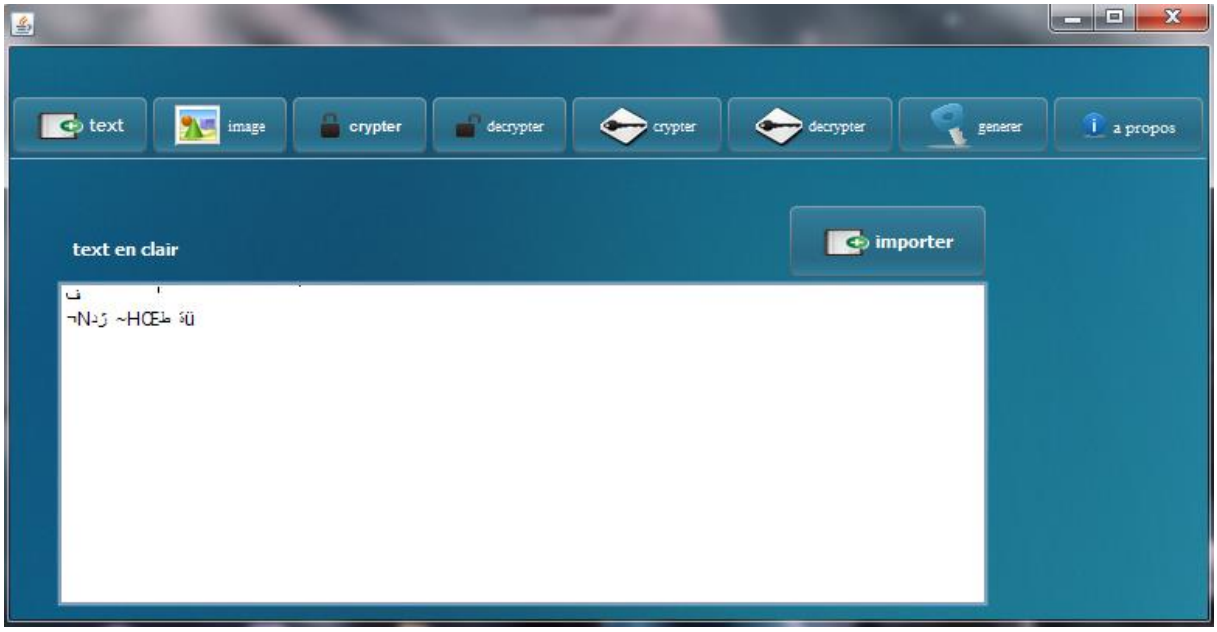
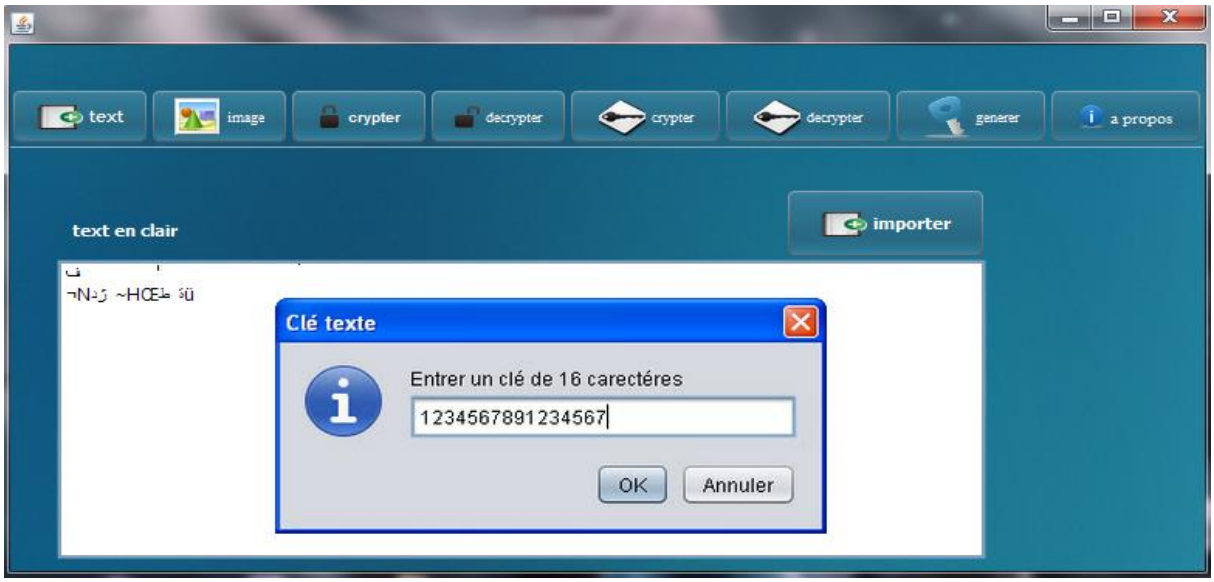


Figure 19:Le résultat de cryptage.

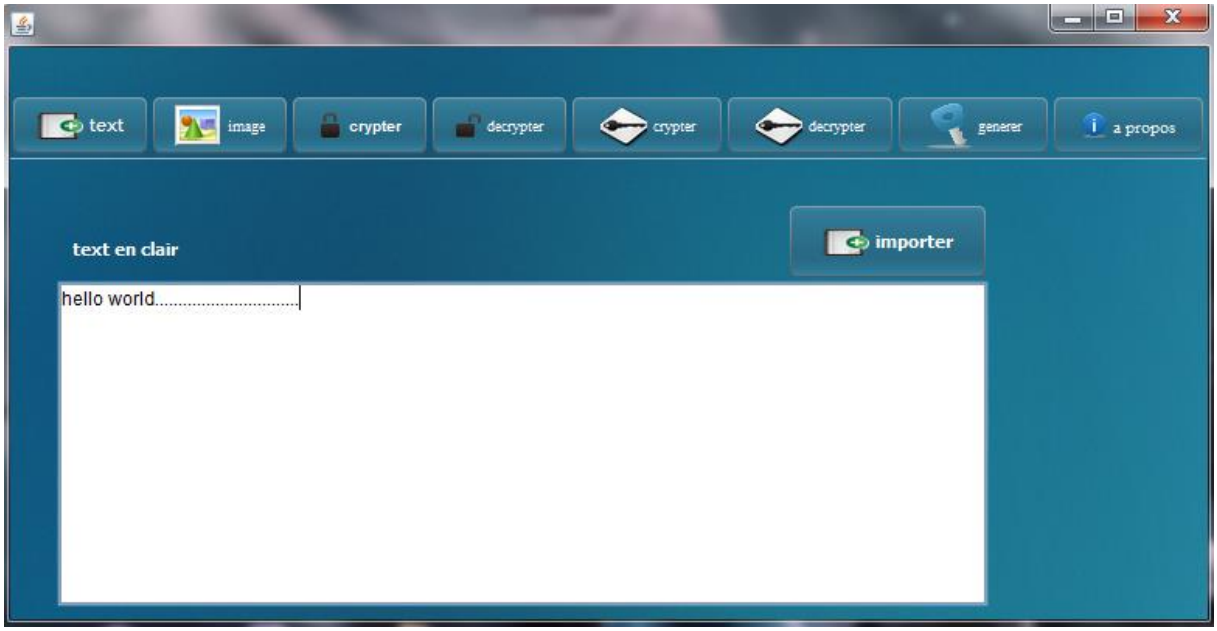
b)Le déchiffrement

1-Pour le décryptage nous suivons les mêmes étapes précédentes mais nous choisissons le bouton décrypter au lieu de crypter.

Chapitre 3 : Implémentation



Le résultat de décryptage sera affiché dans le même champ de saisir.



Exemple d'application sur la clé :

a)Le cryptage

1-Entrez la clé publique dans le champ clé publique.

2-Entrez la clé en clair dans le champ clé.

3-Choisis le bouton ok.

4-Le résultat s'affiche dans le champ clé chiffré.

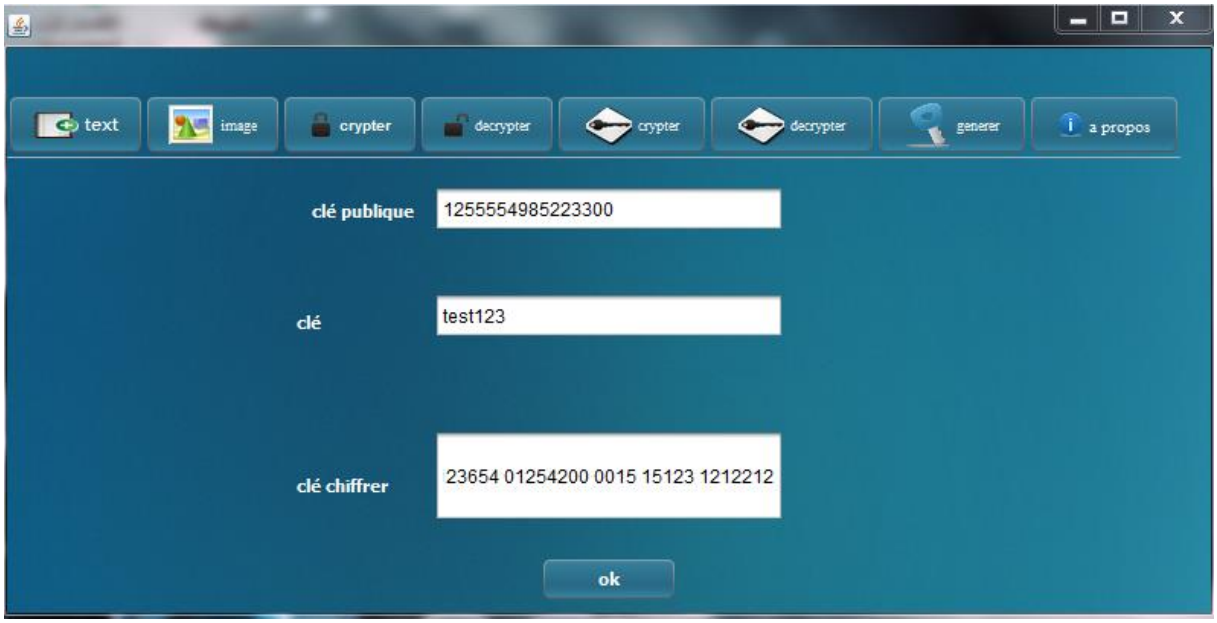


Figure 20: le cryptage de la clé.

b)Le décryptage

1-Entrez la clé privé dans le champ clé privé.

2-Entrez la clé chiffrer dans le champ clé chiffrer.

3-Choisis le bouton ok.

4-Le résultat s'affiche dans le champ password.

Chapitre 3 : Implémentation

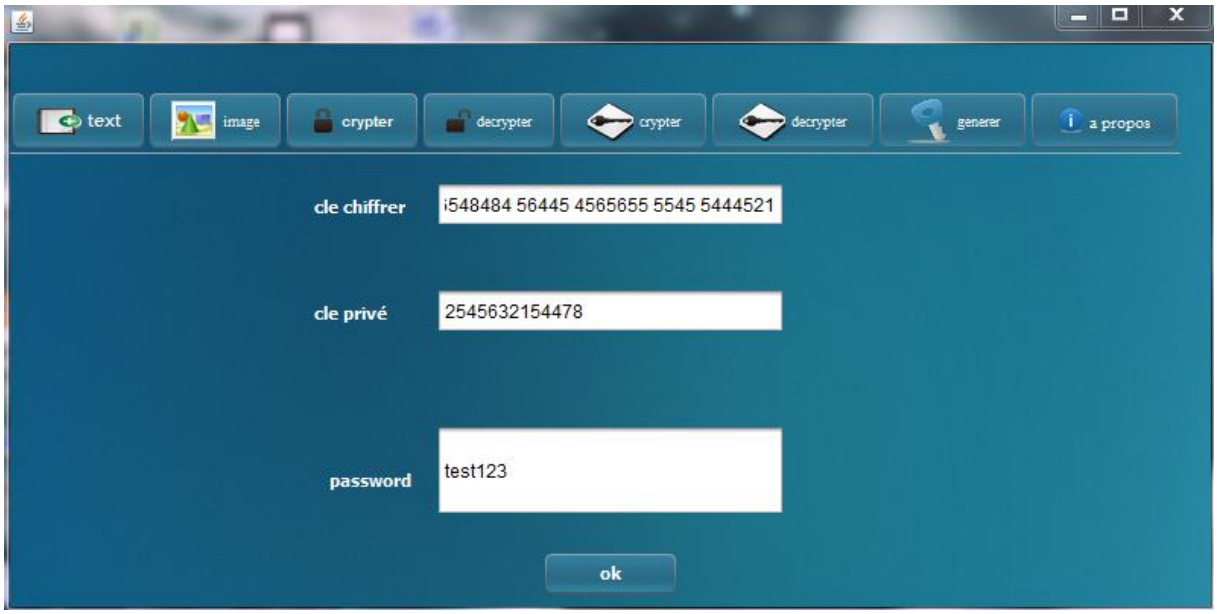


Figure 21:le décryptage de la clé

Exemple d'application sur la génération des clés :

1-Choisis le bouton générer pour choisir la taille des clés.

2-Choisis le bouton calculer pour générer les clés.

La clé publique s'affiche dans le champ clé publique.

La clé privé s'affiche dans le champ clé privé .

Figure 22:la génération des clés.

6. Conclusion

Ce dernier chapitre était consacré pour la réalisation de PGP. Nous avons commencé par la description du langage et l'environnement de développement choisi. Puis nous avons présenté l'interface graphique de notre application suivie par des exemples de cryptage et décryptage d'un texte, cryptage et décryptage de clé et la génération des clés.

Conclusion générale

La cryptographie est une science très vaste, elle a un lien direct avec la sécurité des données et la communication. Aujourd'hui il y a plusieurs d'informations qui doivent rester secrètes ou confidentielles, par exemple les informations échanger par les banques, ou un mot de passe ne doivent pas divulgues et personne ne doit pouvoir les déduire. C'est pourquoi ce genre d'information est crypté. Le cryptage est donc, nécessaire pour que les données soient non intelligibles sauf à l'auditoire voulu..

Dans notre travail, nous avons élaboré un état de l'art sur tous les concepts en relation avec la cryptographie à partir des notions fondamentales jusqu'aux mécanismes avancées, suivi par un panorama sur les différentes classes d'algorithmes cryptographiques existantes., et un aperçu global sur la cryptanalyse,

Enfin, nous avons présenté le PGP, suivi par une description des classes d'objets utilisées ainsi que l'interface graphique de l'application et les résultats obtenus.

Comme perspectives relatives au domaine que nous avons traité nous souhaitons étendre notre système pour :

- Le chiffrement et le déchiffrement des images.

Bibliographie

http://www.bibmath.net/crypto/moderne/aes.php3

http://www.cryptologie.com/W eb_Ter

[1] http://www.jscoron.fr/thesis/node2.html

[2] http://www.nku.edu/~christensen/simplified%20IDEA%20algorithm.pdf

[3]http://www.java2s.com/Code/Java/Security/SimpleRSApublickeyencryptionalgorithmimplementation.htm

[4] http://strepoetlo.tuxfamily.org/repo/Stegano_crypto/RSA.pdf

[5] http://fr.wikipedia.org/wiki/International_Data_Encryption_Algorithm

[6] http://stackOverflow.com

[7] http://PGP.org

[8] http://www.RSA.com/rsalabs/node.asp?id=2254

[9]http://boraxemu.wordpress.com/2012/07/13/le-cryptage-rsa/

[10]http://fr.openclassrooms.com/informatique/cours/l-algorithme-rsa/crypter-et-decrypter#r-477234

[11]http://www.bibmath.net/crypto/moderne/aes.php3

[12]http://www.cryptologie.com/