République **A**lgérienne **D**émocratique et **P**opulaire
Ministère De L'**E**nseignement **S**upérieur Et De La **R**echerche **S**cientifique
Université **M**ouloud **M**ammeri De **T**izi-**O**uzou

Faculté De **G**énie **E**lectrique Et D'**i**nformatique
Département D'informatique

Mémoire De Fin D'études De MASTER ACADEMIQUE

Domaine : **Mathématique et Informatique**
Filière : **Informatique**
Spécialité : **Conduite de Projets Informatiques**

Thème

Simulation de quelques attaques sur le cryptosystème RSA

Présenté devant le jury composé de :

Président : Mr HAMMACHE.A
Examinateur : Mr RADJA.H
Encadreur : Mme HADAOUI .R

Présenté par :

KERNOUF Yamina
ZERROUKI Cylia

Année universitaire : 2019/2020

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

Table des acronymes

CIA: Central Intelligence Agency

Protocole AAA : Authentication, Authorization and Accounting protocol

PIN: Personal Identification Number

DAR: Disque ARchive

USB: Universal Serial Bus

DIT: Data In Transit

DES: Data Encryption Standard

AES: Advanced Encryption Standard

NIST: National Institute of Standards and Technology

RSA: Rivest.R, Shamir.A, Adleman.L

ECC: Elliptic Curve Cryptography

ANSI: American National Standards Institute

IEEE: Institute of Electrical and Electronics Engineers

ISO: International Organization for Standardization

ECDSA: Elliptic Curve Digital Signature Algorithm

MD: Message Digest

SHA: Secure Hash Algorithm

RIPEMD: RACE Integrity Primitives Evaluation Message Digest

IA: Intelligence Artificielle

CRT: Chinese Remainder Theorem

CPU : Central Processing Unit

GHz : Gigahertz

Introduction générale

Introduction générale

L'évolution rapide des réseaux informatiques, privés ou publics, engendre un volume toujours plus important de données sauvegardées et transmises, générant ainsi de nouveaux besoins en matière de sécurité. Dans un monde où l'entreprise dépend de plus en plus de son système informatique, la sécurité est donc devenue une préoccupation primordiale.

Dans ce contexte, la cryptographie étant une science de chiffrement préservant le texte en clair de l'indiscrétion des utilisateurs malintentionnés. Bien utilisée elle assure tous les objectifs de la sécurité, à savoir la confidentialité des données, l'authentification, la disponibilité, l'intégrité et la non répudiation.

Etant l'algorithme de cryptographie le plus utilisé, le cryptosystème RSA a révélé quelques failles après des études cryptanalytiques, ces dernières démontrent qu'il faut implémenter RSA avec beaucoup de précautions et cela en choisissant des clés de chiffrement de grandes tailles et en évitant de chiffrer de petits blocs.

L'objectif de ce travail est d'étudier la cryptanalyse de RSA. Il est divisé en 3 chapitres et une annexe.

Le chapitre 1, présente les principes et notions de la sécurité et de la cryptographie, quelques algorithmes cryptographiques et la cryptanalyse en général.

Le chapitre 2, présente le fonctionnement de l'algorithme RSA, le principe de son chiffrement, du déchiffrement et de la signature, ainsi que la cryptanalyse de RSA et quelques attaques.

Le chapitre 3, présente la conception et la simulation de deux attaques sur le RSA à savoir l'attaque par factorisation et l'attaque de Wiener, explique et expose l'application réalisée dans ce sens, en montrant ses différentes fonctionnalités et les scénarios possibles.

L'annexe, comporte le code de notre application avec des explications détaillées.

Enfin nous terminerons par une conclusion générale.

Chapitre 1

Sécurité informatique et cryptographie

1.1. Introduction

Dans ce chapitre nous introduirons les principes et notions de la sécurité informatique, nous parlerons ensuite de la cryptographie qui permet d'atteindre les objectifs de la sécurité, nous évoquerons par la suite les algorithmes cryptographiques les plus connus, la signature numérique, la fonction de hachage et l'autorité de certification et nous terminerons par donner un aperçu sur la cryptanalyse, ses catégories et ses familles d'attaques.

1.2. Principes et notions de sécurité

1.2.1. Définition de la sécurité informatique

La sécurité informatique est l'ensemble des moyens techniques, organisationnels, juridiques et humains nécessaires et mis en place pour conserver, rétablir, et garantir la sécurité de l'information et des systèmes d'informations. [1]

1.2.2. Objectifs de la sécurité

La notion de sécurité fait référence à la propriété d'un système, d'un service ou d'une entité. Elle s'exprime par les objectifs de sécurité suivant :

➢ La disponibilité

Est la propriété qui permet de garantir l'accès aux ressources, une ressource peut être un serveur, un réseau, une donnée,…etc. mais il ne suffit pas qu'une ressource soit disponible, elle doit être utilisable avec des temps de réponse acceptables, il faut aussi garantir la continuité de service, c'est-à-dire un service doit être assuré avec un minimum d'interruption
La disponibilité est obtenue par exemple par une certaine redondance ou duplication des ressources. [2]

➢ L'intégrité

Elle est liée au fait que des ressources ou services n'ont pas été altérés (détruits ou modifiés) que ce soit d'une façon accidentelle ou d'une façon intentionnelles, donc il est indispensable de se protéger contre la modification des données lors de leur stockage, traitement ou transfert. [3]

➢ La confidentialité

C'est la propriété qui garantit que les informations transmises ne sont compréhensibles que par les entités autorisées, il y a deux actions complémentaires permettant d'assurer la confidentialité des données :

- Limiter et contrôler l'accès aux données afin que seules les personnes autorisées puissent les lire.
- Transformer les données par des techniques de chiffrement pour qu'elles deviennent inintelligibles aux personnes qui n'ont pas les moyens de les déchiffrer.

Le chiffrement des données (ou cryptage) contribue à assurer la confidentialité et à augmenter la sécurité des données lors de leur transmission ou de leur stockage. [3]

➢ L'authentification

C'est une propriété qui consiste à vérifier l'identité d'une entité avant de lui donner l'accès à une ressource, cette entité devra prouver son identité par un mot de passe, empreinte biométrique,…etc. Tous les mécanismes de contrôle d'accès logique aux ressources informatiques nécessitent l'identification et l'authentification (pas d'accès anonyme aux ressources). [3]

➢ La non-répudiation

Est le fait de ne pouvoir nier qu'un événement a eu lieu (un évènement peut être une action ou une transaction), par exemple, la non-répudiation permet d'avoir une preuve qu'un utilisateur a envoyé (ou reçu) un message particulier, ainsi l'utilisateur ne peut nier cet envoi (ou réception). [3]

1.2.3. Schémas de sécurité

Pour atteindre les objectifs de la sécurité, plusieurs schémas de sécurité ont vu le jour. Les plus connus sont :

➢ Le triangle CIA (1987)

Le triangle CIA est utilisé en tant que base par les autres modèles, il représente les trois grands axes de la sécurité qui sont: [4]

- Confidentialité.
- Intégrité.
- Disponibilité.

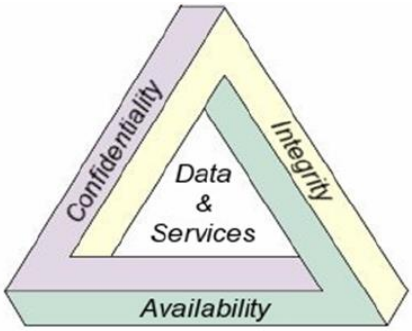

Figure 1. Triangle de CIA (1987).

➢ Le contrôle d'accès, protocole AAA

Le contrôle d'accès se fait en quatre étapes : [4]

1. Identification : qui êtes-vous ?
2. L'authentification : prouvez-le ! visant à prouver l'identité d'un individu qui peut se faire de plusieurs manières :

   - Ce que vous savez (mot de passe, code PIN,… etc.).
   - Ce que vous avez (carte magnétique, lecteur de carte,…etc.).
   - Ce que vous êtes (empreintes digitales, réseau rétinien,…etc.).

L'authentification forte résulte de la combinaison de 2 de ces facteurs.

3. Autorisation : avez-vous les droits requis ?
4. Accounting: permet la collecte d'informations sur l'utilisation des données (durée de connexion, adresse IP de l'utilisateur…).

Ces étapes (les deux premières étapes fusionnées) forment le protocole AAA.

➢ Le pentagone de confiance

Défini par Piscitello en 2006, il précise la notion d'accès à un système. Indépendamment des notions définies dans le triangle CIA, ce modèle précise la confiance que peut/doit avoir l'utilisateur en présence d'un système informatisé.

Figure 2. Pentagone de confiance.

Dans ce pentagone de confiance en plus des notions vues précédemment (authentification, disponibilité, autorisation et intégrité (Authenticité)) on y retrouve une nouvelle notion qui est l'admissibilité, cette notion nous permet de savoir si la machine cible est fiable. [4]

➢ L'hexagone de Parker

Ce modèle, initié par Donn Parker, ajoute la nuance d'utilité (une information chiffrée pour laquelle on a perdu la clé de déchiffrement n'est plus d'aucune utilité, bien que l'utilisateur y ait accès, que cette information soit confidentielle, disponible et intègre). [4]

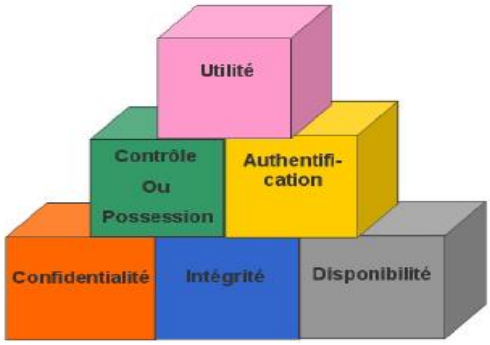

Figure 3. Hexagone de Parker.

➢ Le cube de Mc Cumber

On y retrouve les trois piliers de la sécurité (CIA), mais deux autres dimensions apparaissent: [4]

- **L'état des données :** stockage, transmission, traitement
  - Stockage : données inactives appelées également DAR, telles que celles stockées en mémoire, sur bande magnétique, sur disque ou sur clé USB.
  - Transmission : appelée également données en transit (DIT).
  - Traitement : opérations sur des données afin d'atteindre un objectif souhaité.
- **Les méthodes :** ce sont les principes et les règles à sélectionner pour atteindre le niveau de sécurité souhaité (politiques et pratiques, facteurs humains et technologies).
  - Politiques et pratiques : opérations, directives de gestion.
  - Facteurs humains (personnel) : veiller à ce que les utilisateurs de système soient informés de leurs rôles et de leurs responsabilités en matière de protection du système et soient capables de respecter les normes.
  - Technologies : solutions logicielles et matérielles utilisées pour assurer les objectifs de sécurité (exemples : antivirus, pare-feu, cryptosystème, systèmes de détection d'intrusion,…etc.).

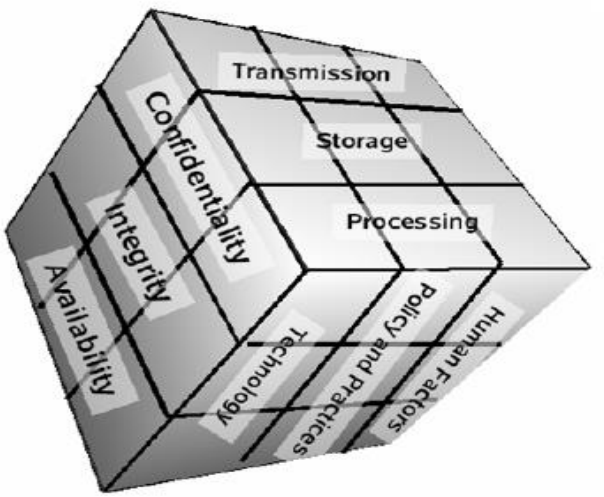

Figure 4. Cube de Mc Cumber.

1.3. Cryptographie

1.3.1. Concepts de base

- **Cryptologie:** La cryptologie est la science qui étudie les aspects scientifiques des techniques de la cryptographie et la cryptanalyse. [5]
- **Stéganographie :** en grec « l'écriture couverte » cache (dissimule) un secret dans un support anodin, par exemple des images ou un texte. [6]
- **Cryptographie:** Le mot cryptographie est un terme générique désignant l'ensemble des techniques permettant de rendre un message inintelligible (incompréhensible). [7]
- **Chiffrement:** Le chiffrement des données consiste à convertir des données afin que seules les personnes pourvues d'une clé secrète ou d'un mot de passe soient en mesure de les lire. On utilise une clé de chiffrement pour chiffrer les données et une clé de déchiffrement pour les déchiffrer. Alors qu'elles sont initialement en texte brut (plaintext) les données chiffrées sont en texte chiffré (ciphertext). [8]
- **Déchiffrement:** Est le processus inverse du chiffrement, consiste à retrouver le texte original du message chiffré dont on possède la clé de déchiffrement. A ne pas confondre avec le décryptement. [9]
- **Cryptanalyse:** Elle consiste à casser des fonctions cryptographiques existantes dans le but de démontrer leur sécurité. [10]
- **Décryptage:** consiste à retrouver le texte original à partir du message chiffré sans posséder la clé de déchiffrement. [11]

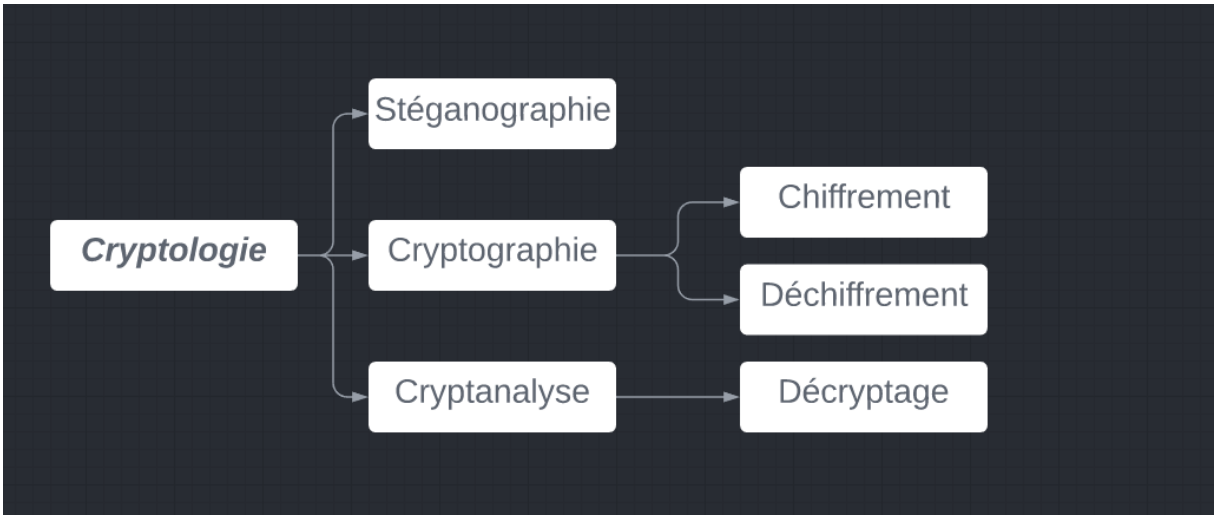

Figure 5. Organigramme de la cryptologie.

1.3.2. Définition de la cryptographie

La cryptographie est une discipline très ancienne. Étymologiquement, le mot cryptographie vient du grec «kryptos» qui signifie cacher, et «graphein» signifiant écrire. Ainsi la cryptographie est l'étude des différentes méthodes et techniques mathématiques reliées aux aspects de sécurité de l'information, pour assurer la confidentialité et l'authenticité des messages, elle permet de stocker des informations sensibles ou de les transmettre à travers des réseaux non sûrs (comme Internet) de telle sorte qu'elles ne peuvent être lues par personne à l'exception du destinataire convenu.

Afin de protéger un message (texte en clair) on lui applique une transformation qui le rend incompréhensible, c'est le chiffrement, et pour retrouver le message d'origine, le destinataire applique une autre transformation, qui est le déchiffrement. [12]

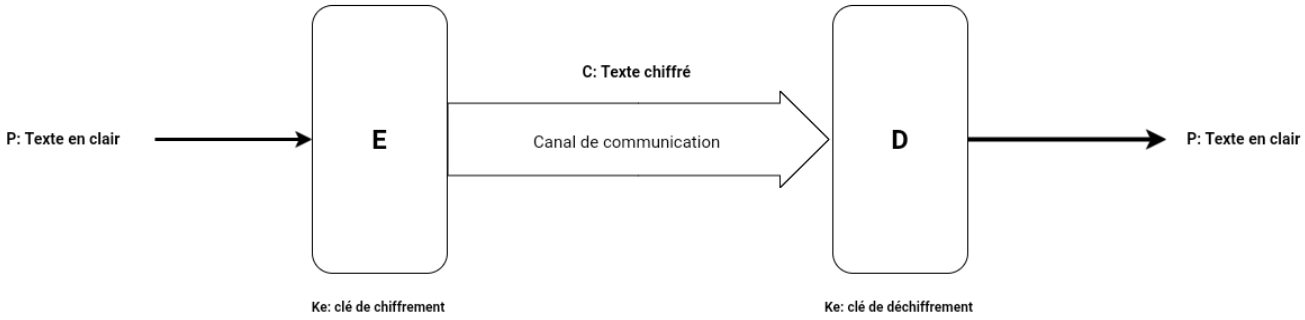

Figure 6. Principe de la cryptographie.

1.3.3. Apport de la cryptographie à la sécurité

La sécurité informatique est un ensemble de mesures prises pour protéger un système informatique contre d'éventuelles attaques. Différentes techniques et méthodes ont été développées pour mettre en œuvre la stratégie de sécurité, la cryptographie est l'une de ces méthodes.

On vise aussi à satisfaire les objectifs de la sécurité à savoir, la confidentialité, la disponibilité, l'intégrité, l'authentification et la non-répudiation via la cryptographie. [3]

1.3.4. Classes de la cryptographie

On distingue deux grandes classes de cryptographies : la cryptographie classique et la cryptographie moderne [13].

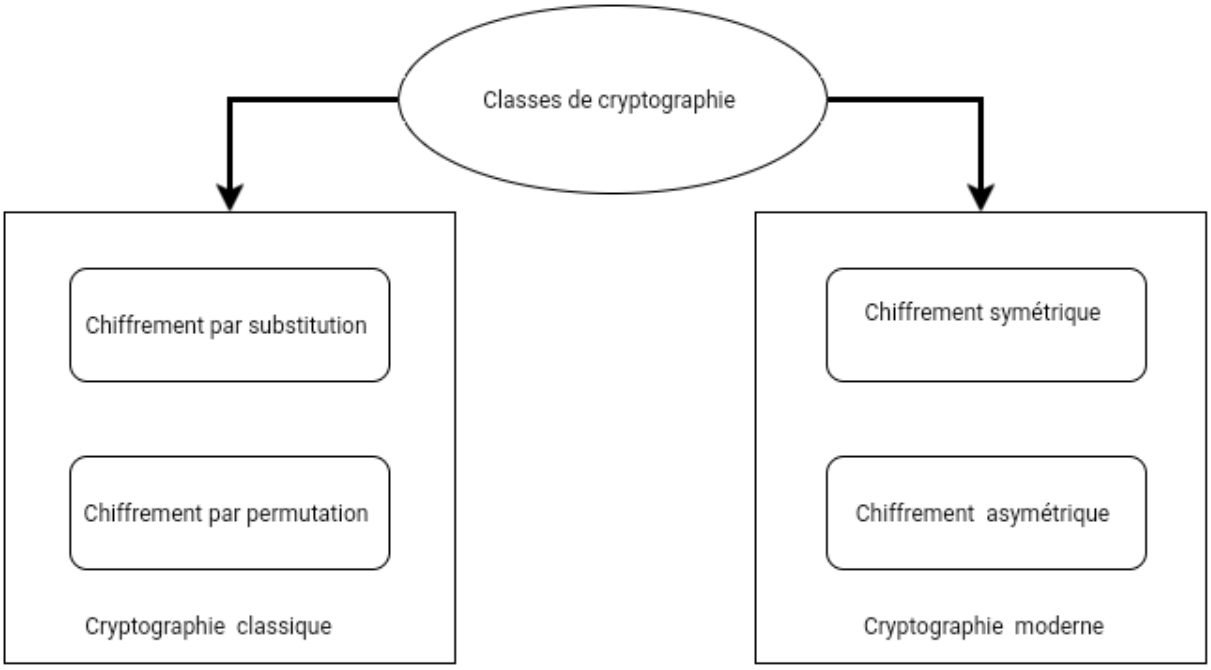

Figure 7. Les deux classes de la cryptographie.

1.3.4.1. Cryptographie classique

La cryptographie classique repose sur deux principes essentiels, la substitution et la permutation. [13]

➢ Chiffrement par substitution

Chaque symbole ou ensemble de symboles est remplacé par un autre symbole ou un ensemble de symboles pour obtenir le texte chiffré. On distingue trois types de substitution qui sont :

• Substitution mono-alphabétique

Elle consiste à remplacer chaque symbole du texte en clair, par un autre symbole pour obtenir le texte chiffré. Exemples :

- **Code de César:** consiste à décaler les lettres de l'alphabet d'un nombre K qui représente la clé.
- **Chiffrement mono-alphabétique aléatoire:** permet une substitution aléatoire de chaque lettre par une autre, la clé est formée de 26 caractères.

• Substitution poly-alphabétique

Elle consiste à utiliser une suite de chiffres mono-alphabétiques réutilisée périodiquement. Un même symbole peut être remplacé par plusieurs symboles. Exemple :

- **Chiffrement de Vigenère:** C'est un chiffrement symétrique de substitution poly-alphabétique et une amélioration du code de César, il utilise un mot de passe dont chaque lettre indique le décalage alphabétique à appliquer sur chaque lettre du texte en clair.

• Substitution de poly-grammes

Consiste à substituer un groupe de **n** symboles dans le texte en clair par un autre groupe de **n** symboles. Exemples :

- **Chiffrement de Playfair:** on dispose 25 lettres de l'alphabet (toutes sauf le W. Pour la variante anglaise on garde le W et on fusionne le I et le J) dans une grille 5x5 ce qui donne la clé. Pour former les grilles de chiffrement on utilise un **mot clé secret** pour créer un alphabet désordonné, avec lequel on remplit la grille ligne par ligne. Les autres lettres de l'alphabet sont ajoutées dans l'ordre dans la grille pour la compléter.

- **Chiffrement de Hill:** consiste à :
  - Remplacer chaque lettre par son ordre dans l'alphabet, A=0, B=1…..Z=25.
  - Regrouper les nombres ainsi obtenus en bloc de **m.**
  - Pour chaque bloc de **m** nombres à coder, calculer le texte codé en effectuant des combinaisons linéaires avec une clé K de la forme d'une matrice carrée d'ordre **m.**
  - Le déchiffrement peut être effectué en utilisant la matrice inverse $\mathbf{K^{-1}}$ dans Z26.

➢ Chiffrement par permutation

C'est une technique dans laquelle le texte en clair est réécrit comme une séquence de lignes, puis réordonnée comme une séquence de colonnes. L'ordre des colonnes ainsi que leur

nombre constitue la clé de cryptage. La position de chaque symbole du texte en clair est échangée ou permutée avec la position d'un autre symbole pour obtenir le texte chiffré.

1.3.4.2. Cryptographie moderne

La cryptographie moderne se scinde en deux parties nettement différenciées : [13]

- La cryptographie utilisant la même clé pour le chiffrement et le déchiffrement ou encore appelée Symétrique.

- La cryptographie utilisant une clé publique pour le chiffrement et une clé privée pour le déchiffrement ou encore appelée Asymétrique.

➢ Chiffrement symétrique

Les algorithmes de chiffrement symétriques (ou à clé privée) sont ceux pour lesquels l'émetteur et le destinataire partagent une même clé, autrement dit, les clés de chiffrement et de déchiffrement sont identiques. L'emploi de ce genre d'algorithme lors d'une communication nécessite donc l'échange préalable de la clé entre les deux interlocuteurs à travers un canal sécurisé ou au moyen d'autres techniques cryptographiques.
L'avantage principal de ce mode de chiffrement est sa rapidité. Cependant un paramètre essentiel pour la sécurité d'un système à clé privée est la taille de la clé. En effet, il est possible de retrouver la clé en menant une attaque dite l'attaque exhaustive, cette attaque consiste simplement à énumérer toutes les clés possibles du système et à essayer d'utiliser chacune d'entre elles pour décrypter un message chiffré. Si la taille de la clé est de k bits, le nombre de tentatives d'attaque exhaustive en vue de décrypter le message chiffré est égal à 2^k. Donc, pour pénaliser une telle attaque, il faut que la taille de la clé soit suffisamment grande.
Il existe d'autres types d'attaques sur les systèmes de chiffrement à clé privée dont la plupart consistent à exploiter certaines structures particulières de l'algorithme ou certaines caractéristiques statistiques dans la distribution des couples de textes clairs-chiffrés. Les plus connues sont la cryptanalyse différentielle, inventée par *Biham* et *Shamir* en 1991, et la cryptanalyse linéaire du Japonais Matsui en 1993. [14]

Figure 8. Chiffrement à clé privée.

Le chiffrement à clé privée se subdivise en deux grandes catégories : Les cryptosystèmes par flots et les cryptosystèmes par blocs.

- **Cryptosystèmes par flots**

Les chiffrements par flot (on parle aussi de chiffrements en continu) agissent directement sur chaque bit du texte, on chiffre un bit/caractère à la fois, la structure d'un chiffrement par flots repose sur un générateur de clé qui produit une séquence de clés $K_1 \dots K_i$.

Le principal avantage des chiffrements par flot est qu'ils permettent de chiffrer et déchiffrer un message en continu, sans avoir besoin de connaitre tout le message. Ceci justifie leur utilisation dans les domaines où il faut du chiffrement et du déchiffrement en temps réel et simultané (communications téléphoniques, Bluetooth,…etc.).

- **Cryptosystèmes par blocs**

L'opération de chiffrement s'effectue sur des blocs de texte clair. L'idée générale d'un chiffrement par bloc est la suivante :

1. Remplacer les caractères par un code binaire.
2. Découper la chaine en blocs de même longueur.
3. Chiffrer un bloc en l'additionnant bit par bit à une clé.
4. Recommencer éventuellement un certain nombre de fois l'opération 3.
5. Passer au bloc suivant et retourner au point 3 jusqu'à ce que tout le message soit chiffré.

- **Avantages du chiffrement à clé privée**

  - Rapidité.
  - Ces algorithmes utilisent des clés de taille relativement faible et peu de ressources.

- **Inconvénients du chiffrement à clé privée**

  - Problème de distribution de clé (l'émetteur et le récepteur doivent se mettre d'accord à l'avance sur la clé à utiliser).
  - Problème de gestion de clés (Chaque entité doit disposer d'autant de clés secrètes qu'elle a d'interlocuteurs).

➢ Chiffrement asymétrique

Le concept du chiffrement à clé publique, qui est un autre nom du chiffrement asymétrique, a été présenté pour la première fois par Whitfield Diffie et Martin Hellman dans leur article en 1976, puis publié quelques mois plus tard sans pouvoir donner un exemple d'un système à clé publique. Il a fallu attendre 1978 où la version académique du premier cryptosystème à clé publique a fait apparition.
Dans le chiffrement à clé publique chaque interlocuteur dispose d'un couple de clés, une clé publique connue par tous et une clé privée gardée secrète. L'émetteur (la source) chiffre le message en utilisant la clé publique du destinataire, et le destinataire déchiffre le message en utilisant sa clé privée qu'il est le seul à connaitre.

La notion primordiale sur laquelle repose le chiffrement à clé publique est celle de fonction à sens unique avec trappe. Sachons qu'une fonction est appelée à sens unique si elle est aisément calculée, mais difficile à inverser, et dite à trappe, si le calcul de l'inverse devient facile dès que l'on possède une information supplémentaire qui est la trappe. L'utilité de l'utilisation d'une telle fonction, dans le domaine de la cryptographie, réside dans le fait de rendre difficile la détermination du message en clair à partir du message chiffré sans connaitre la clé privée.
Cependant, la définition de ces fonctions particulières n'est pas aussi facile puisqu'elle s'appuie généralement sur des problèmes mathématiques réputés difficiles tel que le problème de la factorisation de grands nombres entiers. Mais, si on arrive un jour à simplifier la résolution d'un de ces problèmes, l'algorithme correspondant, finira par être abandonné. [15]

Figure 9. Chiffrement à clé publique.

- **Avantages du chiffrement à clé publique**
  - Pas de problèmes de gestion de clés.
  - Chaque interlocuteur ne garde secret que sa clé privée.
- **Inconvénients du chiffrement à clé publique**
  - Charge de calcul (les algorithmes asymétriques sont particulièrement lents à comparer avec ceux symétriques et demandent un temps de traitement processeur important).
  - Ils nécessitent des clés plus longues (1024 bits, 2048 bits).

1.4. Algorithmes connus en cryptographie

1.4.1. DES

Data Encryption Standard (DES) est un algorithme symétrique par bloc, dans lequel le message en clair est décomposé en blocs de 64 bits, chaque bloc entre d'un côté de l'algorithme et un bloc crypté de 64 bits sort de l'autre côté, ceci en utilisant une clé de 56 bits, cette clé est extraite d'une clé saisie de 64 bits (8 octets) dans laquelle on ignore un bit de chaque octet.
L'algorithme repose sur un certain nombre de combinaisons, des opérations de substitutions et de permutation. Mais, il s'avère que la clé 56 bits est insuffisante pour résister aux attaques de

force brute, beaucoup de travaux de cryptanalyse se sont intéressés au DES et ont abouti à des méthodes permettant de le casser en présence de centaines de configuration de clés, pour y remédier à cela et améliorer la robustesse du DES, d'autres formes sont utilisés comme le Triple DES qui répète trois fois le DES. [16]

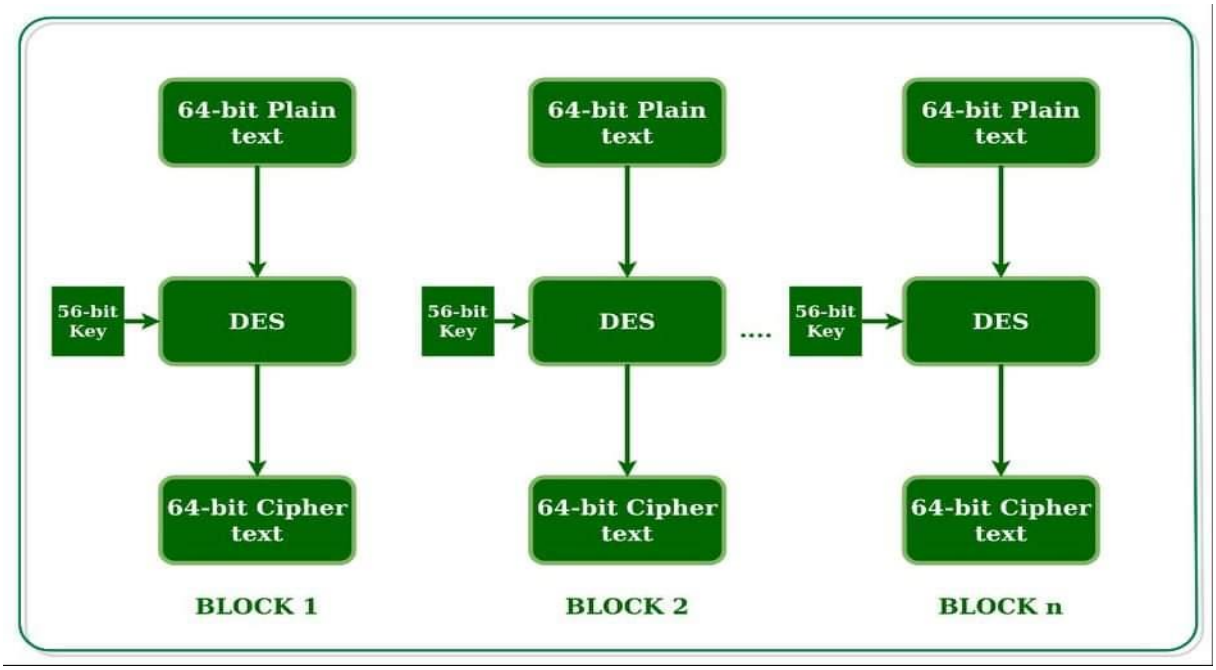

Figure 10. Schémas descripteur du DES.

1.4.2. AES

AES pour Advanced Encryption Standard. En 1997, le NIST lance un appel d'offre pour remplacer DES. Quinze solutions ont été proposées parmi lesquelles, nous citons : Mars, Rc6, Rijndael, Serpent et Twofish. En octobre 2000, Rijndoael (du nom de ses deux concepteurs : Joan Daemen et Vincent Rijmen) a été choisi comme solution standard par le NIST, nommée également AES. Cet algorithme utilise des blocs de 128 bits et des clés variante entre 128 et 256 bits. Il est basé sur plusieurs opérations : permutations, rotations des lignes de la matrice M, transformations linaires basées sur la multiplication binaire de la matrice M avec des polynômes et le OU exclusif. Ces opérations sont répétées plusieurs fois en fonction de la clé de chiffrement. Pour des clés 128, 192 ou 256 bits, AES nécessite respectivement 10, 12 ou 14 itérations.

En juin 2003, le gouvernement américain a déclaré que l'algorithme AES, avec ses différentes clés est largement suffisant pour assurer la confidentialité des données secrètes. Cependant, la cryptanalyse a pu "casser" AES en utilisant des attaques par canal auxiliaire et par texte clair. [17]

1.4.3. RSA

Parmi les algorithmes de chiffrement à clé publique, RSA est l'algorithme le plus connu, il tire son nom des noms de ses trois inventeurs : R. Rivest, A. Shamir, et L. Adleman. Inventé en 1976, il est le premier protocole de chiffrement à clé publique. Il fournit une sécurité élevée et il est basé sur une suite d'opérations arithmétiques de forme exponentielle modulaire (*modulo N*) sur de grands nombres premiers. RSA est aujourd'hui un système universel servant dans une multitude d'applications. On l'utilise dans les transactions sécurisées sur internet, il est aussi implanté dans de nombreux standards informatique comme le Society for Worldwide Interbank Financial Télécommunication Standard ou encore le X9.44 draft Standard for the U.S Banking Industry. Il tire sa sécurité du problème de factorisation et son principe est comme suit : on possède une paire de clés, l'une publique (N, e) et l'autre privée(N, d). Pour générer ces clés il faut : [18]

- Choisir deux grands nombres premiers p et q et calculer $N = p \times q$.
- calculer $\varphi(N) = (p-1) \times (q-1)$
- choisir e tel que 1<e<$\varphi(N)$ et pgcd(e, $\varphi(N)$)=1.
- calculer d tel que: e×d mod $\varphi(N)$=1 (1<d<$\varphi(N)$)).

La clé publique est formée par (N, e) et la clé privée par (N, d).

Dans RSA, les messages sont représentés par des blocs d'entiers naturels strictement inférieurs à N. Il est utilisé pour assurer la confidentialité, l'intégrité des données, l'authentification de l'origine de données et la non répudiation.

1.4.4. Courbe elliptique

La cryptographie sur courbes elliptiques, ou *Elliptic Curve Cryptography* (ECC) en anglais, c'est une approche de chiffrement à clé publique, introduite par N.Koblitz et Victor S. Miller en 1985. Ces deux chercheurs ont vu le potentiel offert par la structure mathématique des courbes elliptiques pour l'adapter à la cryptographie. Standardisé à la fin des années 1990 par un certain nombre d'organisations (ANSI X9.63, IEEE P1363, ISO 15946, NIST SP 800-56), elle a commencé à être acceptée pour commercialisation. De nos jours, l'ECC est principalement utilisée dans les environnements à faible ressource tels que les réseaux sans fil ad-hoc et les réseaux mobiles, car elle ne nécessite pas de longues clés pour assurer un haut niveau de sécurité, contrairement aux autres algorithmes à clés publiques qui nécessitent des clés suffisamment grandes pour assurer un niveau de sécurité élevée, (par exemple, pour un même niveau de sécurité, RSA utilise une clé de 1024 bits, alors que l'ECC utilise une clé de160 bits). Elle est utilisée pour: [30]

- L'échange de clé.
- Chiffrer et transmettre les données.
- Signature digitale ECDSA (Elliptic Curve Digital Signature Algorithm).

La courbe elliptique est de la forme :

$$Y^2 = X^3 + aX + b$$

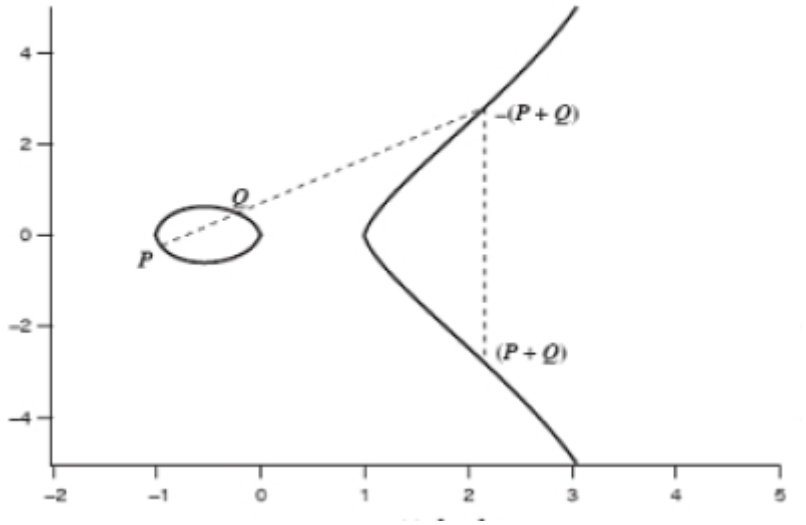

Figure 11. Exemple de EC (avec $Y^2 = X^3 - X$).

Les courbes elliptiques permettent de remplacer les calculs sur des entiers, par des calculs dans les groupes associés à une courbe elliptique .Ce qui fait la différence des algorithmes basés sur les entiers comme RSA par rapport aux algorithmes de chiffrement à base de courbes elliptiques est que ces algorithmes tirent leur sécurité du logarithme discret sur le groupe de la courbe elliptique, et non un problème sur les entiers. Ces groupes sont plus difficiles à manipuler, ils peuvent différer beaucoup les uns des autres si on change les paramètres. Ainsi, résoudre un problème de logarithme discret sur une courbe elliptique est réputé être un problème plus difficile que le problème similaire dans les entiers modulo N. La théorie des fonctions elliptiques est complexe, et encore relativement récente, et la technologie de cryptographie par courbe elliptique a fait l'objet du dépôt de nombreux brevets à travers le monde. Cela peut rendre son utilisation très coûteuse.

1.5. Fonction de hachage, signature numérique et autorité de certification

1.5.1. Fonction de hachage cryptographique

C'est un algorithme symétrique par bloc. Il permet l'usage du chiffrement asymétrique sans engendrer trop de ralentissement, mais également d'assurer la provenance d'un fichier ainsi que son intégrité. Dans la majorité des cas, elles seront utilisées pour créer une signature numérique qui permet d'authentifier les expéditeurs.
On les utilise en cryptographie pour fournir un "condensé" ou "haché" ou encore "empreinte" de taille fixe. Un condensé est la caractéristique d'un texte ou de données uniques dont l'objectif est d'être représentatif d'une donnée particulière et bien définie ce qui permet une détection simplifiée des changements dans un message.[19]

Les fonctions de hachage sont de la forme $h = H\ (m)$ et possèdent de nombreuses propriétés :

- Il doit être facile de calculer $h = H\ (m)$ pour n'importe quel message m.

- Elles produisent un résultat de longueur constante.
- Pour un h donné, il est impossible de trouver x tel que $H(x) = h$. On parle de propriété à sens unique.
- Pour un x donné, il est impossible de trouver y tel que $H(y) = H(x)$. Résistance faible de collision.
- Il est impossible de trouver x, y tels que $H(y) = H(x)$. Résistance forte de collision.

1.5.2. La signature numérique

Souvent on essaye de protéger les deux entités communicantes d'un intrus, mais il faut les protéger aussi l'une de l'autre, car en cas de conflits, rien n'empêche la répudiation du ou des messages. C'est ici qu'interviennent les signatures numériques. Elles permettent notamment de vérifier l'auteur, la date et l'heure de la signature, d'authentifier le contenu d'un message et peuvent être vérifiées par des tiers pour résoudre ces conflits. [20]
Une signature digitale doit être:

- Authentique : la signature doit permettre au récepteur du message d'identifier son expéditeur (identifier la personne qui a apposé la signature).
- Infalsifiable : la signature ne peut être falsifié (on ne peut pas utiliser la signature d'une tierce personne).
- Non réutilisable : la signature doit dépendre du message à signer (deux messages différents impliquent deux signatures différentes).
- Inaltérable : un message signé ne peut plus être modifié (le message n'a pas été altéré entre le moment de sa signature et le moment de sa réception).
- Signature facile à mémoriser.
- Irrévocable : le signataire ne peut pas nier sa signature.

1.5.3. Autorité de certification

Dans le chiffrement à clé publique et dans la validation des signatures numériques, l'utilisateur doit utiliser la clé publique de son destinataire. On sait que les clés publiques sont rangées dans un annuaire accessible par tout le monde, et si ce dernier n'est pas sécurisé, on est exposé au risque d'usurpation d'identité. Un intrus « I » peut se faire passer pour un utilisateur A, en remplaçant la clé publique de A par la sienne. Lorsqu'un autre utilisateur B reçoit la signature de « I », il va croire que c'est celle de A. Pour remédier à ce problème, nous utilisons l'autorité de certification, qui est une entité de confiance qui émet des certificats numériques, à savoir des fichiers de données servant à relier cryptographiquement une entité à une clé publique.
Les certificats numériques incluent des données sur l'entité émettrice du certificat ainsi que des données de chiffrement servant à vérifier l'identité de l'entité liée au certificat numérique.
Habituellement, ils contiennent des renseignements sur l'entité pour laquelle ils ont été créés, notamment sa clé publique et la date d'expiration du certificat, le nom de l'entité, ses

coordonnées et d'autres informations sur l'entité certifiée. En fonction du niveau de sécurité, on distingue trois classes de certificats : [21]

- Certificats de classe 1 : l'utilisateur ne fournit qu'une adresse e-mail.
- Certificats de classe 2 : l'autorité réclame des documents officiels.
- Certificats de classe 3 : Ces certificats ne peuvent être délivrés que par la présence physique d'un utilisateur.

Une entité ou une personne qui a besoin d'un certificat numérique peut en faire la demande à une autorité de certification ; après vérification de son identité, l'autorité créera un certificat numérique pour ce demandeur et le signera numériquement avec sa clé privée. Le certificat numérique pourra ensuite être authentifié à l'aide de la clé publique de l'autorité de certification. Un certificat a une durée de vie. Pendant sa validité, le certificat peut être suspendu ou révoqué par l'utilisateur ou par l'autorité de certification.

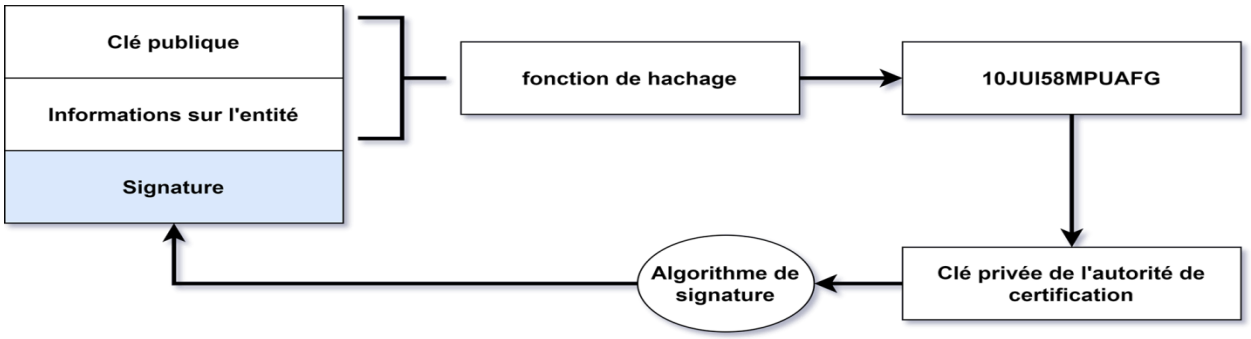

Figure 12. Certificat à clé publique.

1.6. Cryptanalyse

La cryptanalyse est l'étude des probabilités de succès des attaques possibles sur les cryptosystèmes afin de déceler leurs éventuelles faiblesses.

Bien évidemment, la qualité la plus importante que doit posséder un cryptosystème est la sécurité, c'est-à-dire qu'il doit pouvoir résister à une quelconque attaque qui aurait pour but de le casser et d'en extraire le texte en clair original. On peut distinguer deux types réels d'attaques :

- Les attaques passives : qui ont pour but d'intercepter un message et d'exploiter les informations contenues dans celui-ci. Cette attaque ne modifie en rien les informations interceptées.
- Les attaques actives :
  - Impliquent certaines modifications du flot de données ou la création d'un flot frauduleux.
  - Visent à ralentir, dégrader ou même empêcher la communication.

- Envoient des informations parasites dans le but de saturer des systèmes.
- Modifient les informations afin de tromper le destinataire ou de faire carrément disparaître ces informations.

Il existe quatre catégories d'attaques : interruption, interception, modification, fabrication : [22]

- **Interruption :** c'est une attaque portée à la disponibilité. La destruction d'une pièce matérielle (tel un disque dur), la coupure d'une ligne de communication, ou la mise hors service d'un système de gestion de fichiers en sont des exemples.
- **Interception :** c'est une attaque portée à la confidentialité. Il peut s'agir d'une personne, d'un programme ou d'un ordinateur. Une écoute téléphonique dans le but de capturer des données sur un réseau, ou la copie non autorisée de fichiers ou de programmes en sont des exemples.
- **Modification :** il s'agit d'une attaque portée à l'intégrité. Changer des valeurs dans un fichier de données, altérer un programme de façon à bouleverser son comportement ou modifier le contenu de messages transmis sur un réseau sont des exemples de telles attaques.
- **Fabrication :** c'est une attaque portée à l'authenticité. Il peut s'agir de l'insertion de faux messages dans un réseau ou l'ajout d'enregistrements à un fichier.

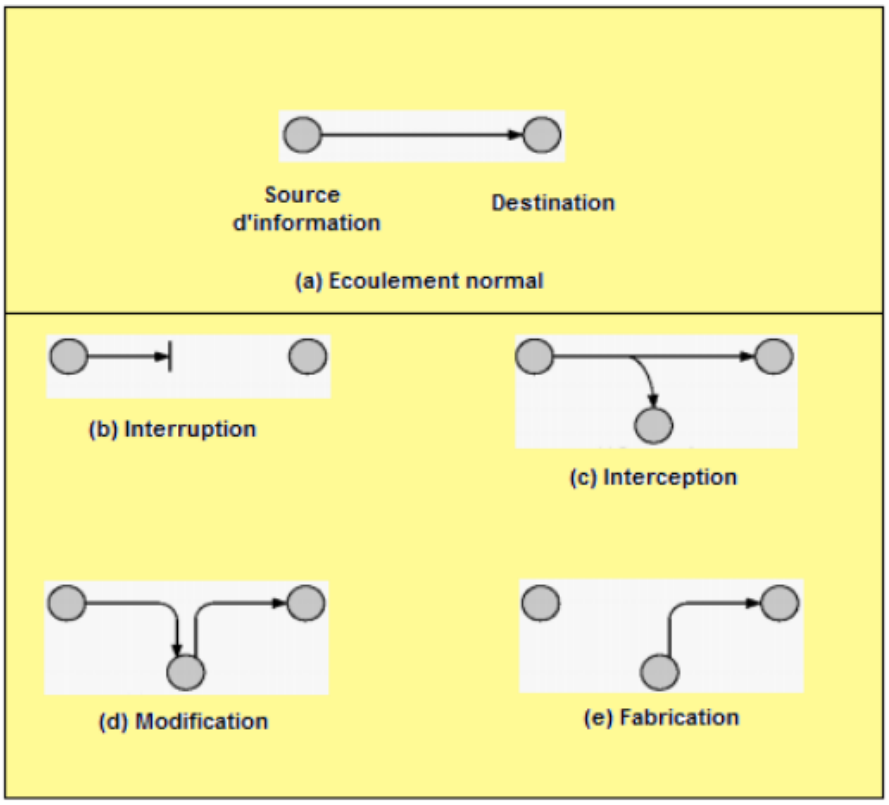

Figure 13. Catégories d'attaques.

1.6.1. Classification des attaques

Une attaque est souvent caractérisée par les données qu'elle nécessite [5]:

- **Attaque sur texte chiffré seul (ciphertext-only) :** 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-plaintextattack) :** le cryptanalyste possède des messages ou des parties de messages en clair ainsi que les versions chiffrées.
- **Attaque à texte clair choisi (chosen-plaintextattack) :** le cryptanalyste possède des messages en clair, il peut générer les versions chiffrées de ces messages avec l'algorithme que l'on peut dès lors considérer comme une boîte noire.
- **Attaque à texte chiffré choisi (chosen-ciphertextattack) :** le cryptanalyste possède des messages chiffrés et demande la version en clair de certains de ces messages pour mener l'attaque.

1.6.2. Familles d'attaques cryptanalytiques

Il existe plusieurs familles d'attaques cryptanalytiques : [5]

➢ Analyse fréquentielle

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

➢ Indice de coïncidence

L'indice de coïncidence 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.

➢ 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.

➢ 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.

- **Attaque par force brute**

L'attaque par force brute consiste à tester toutes les solutions possibles de mots de passe ou de clés.

1.7. Conclusion

Dans ce chapitre nous avons abordé essentiellement la cryptographie qui satisfait les objectifs de la sécurité informatique, qui sont : confidentialité, authentification, disponibilité, intégrité et non-répudiation. Nous avons décrit les algorithmes cryptographiques les plus répandus, nous avons présenté aussi les fonctions de hachages cryptographiques, signature numérique et les autorités de certification et nous avons terminé par donner un petit aperçu sur la cryptanalyse ses catégories et ses familles d'attaques.

Les chapitres suivants porteront sur le premier et le plus célèbre algorithme de chiffrement à clé publique, le **RSA,** sur lequel est basé notre travail.

Chapitre 2

Le cryptosystème RSA

2.1. Introduction

Le chiffrement symétrique existe depuis la création de la cryptographie. Il ne fut remis en question qu'en 1976 par Diffie et Hellman. Suite à la publication de leur article qui introduit l'idée de chiffrement à clé publique (appelé aussi chiffrement asymétrique) trois chercheurs nommés Ron Rivest (Cryptologue), Adi Shamir (Expert en cryptanalyse) et Len Adleman (Chercheur en informatique et biologie moléculaire) avaient décidé de travailler ensemble sur le fait que le nouveau système de codage révolutionnaire de W.Diffie et M.Hellman était une impossibilité logique (il présentait des failles). Ils tentèrent de prouver qu'un cryptosystème asymétrique ne pouvait exister. Mais après de nombreux essais infructueux, ils découvrirent au contraire un candidat possible de cryptosystème asymétrique, aujourd'hui connu sous le nom de RSA. [22] [23]

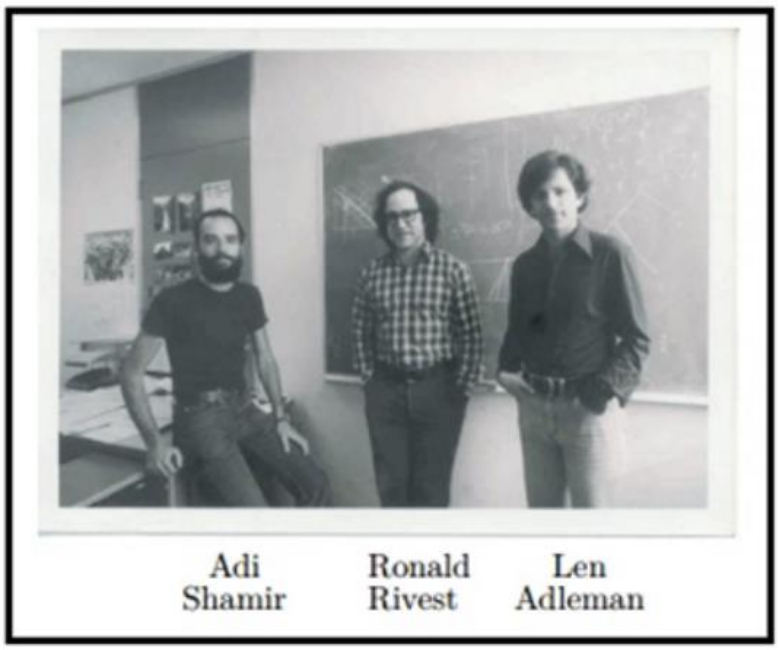

Figure 14. Rivest, Shamir, Adleman.

Dans ce chapitre, nous présenterons le principe de fonctionnement du RSA, ainsi que les principales attaques qui ont été menées contre RSA depuis son apparition.

2.2. Définition et applications

RSA est un algorithme cryptographique qui fait partie de la catégorie des algorithmes à clé publiques ou systèmes asymétriques. L'acronyme RSA, vient du nom de ses trois inventeurs (Rivest, Shamir et Adleman). Ce chiffrement est le plus connu en termes d'algorithmes cryptographiques de type asymétrique. [24]

RSA est aujourd'hui un système universel servant dans une multitude d'applications. On l'utilise dans les transactions sécurisées sur internet, il est implanté dans de nombreux

standards informatiques. Egalement intégré dans les cartes Ethernet, les Cartes à puce Bancaires, mais aussi dans grand nombre d'institutions gouvernementales, militaires et universitaires. En résumé, le RSA est le cryptosystème asymétrique le plus utilisé aujourd'hui dans le monde.

2.3. Définition de l'algorithme

L'algorithme **RSA** est utilisé pour le chiffrement à clé publique et est basé sur le fait qu'il est facile de multiplier deux grands nombres premiers mais difficile de factoriser le produit. C'est l'algorithme de chiffrement asymétrique le plus courant, toujours considéré comme sûr, avec la technologie actuelle, pour des clés suffisamment grandes (***1024, 2048*** *voire* ***4096 bits***). [25]

2.4. Principe de fonctionnement du RSA

RSA est fondé sur l'utilisation d'une paire de clés, composée d'une clé publique pour le chiffrement et d'une clé privée pour le déchiffrement. Il se base sur : la génération des clés, le chiffrement et le déchiffrement, la signature et la vérification de la signature. [30]

2.4.1. Génération des clés

- RSA repose sur le choix d'un couple de deux nombres premiers généralement appelés p et q que l'on doit absolument garder secrets. Les nombres premiers choisis doivent être les plus grands possible afin de compliquer au maximum la tâche à celui qui veut attaquer le système. Ce couple élu va engendrer d'autres nombres qui vont constituer la clé du procédé. Alors le RSA fonctionne à partir de deux nombres premiers distincts.

- On note N le produit de p et q , appelé module de chiffrement tel que : $N = p \times q$.
- On calcule l'indicatrice d'Euler de N: $\varphi(N) = (p-1) \times (q-1)$.
- On choisit e, un entier premier avec $\varphi(N)$, appelé exposant de chiffrement (d'où le choix de la lettre e pour encryption).
- Comme e est premier avec $\varphi(N)$, il est, d'après le théorème de Bachet-Bézout, inversible $mod\ \varphi(N)$, c'est-à-dire qu'il existe un entier d tel que $e.d \equiv 1\ mod\ \varphi(N)$ où d est l'exposant de déchiffrement (d'où le choix de la lettre d pour décryption).
- Maintenant, nous avons nos couples de clés (publique et privée) :
  - ✔ Le couple (N, e) est appelé clé publique.
  - ✔ Le couple (N, d) est appelé clé privée.

2.4.2. Chiffrement et déchiffrement des messages

- Un message m est chiffré en $C = m^e \ mod \ N$, où e est l'exposant de chiffrement et C le message chiffré.
- Le déchiffrement d'un chiffré C n'est autre que : $C^d \ mod \ N$, tel que d est l'exposant de déchiffrement.

Pour chiffrer un message, il suffit de connaitre e et N. D'autre part pour déchiffrer, il faut d et N.

2.4.3. Signature

Après la confidentialité de la transmission d'un message subsiste un autre problème : son authenticité. La signature par RSA permet de résoudre ce problème et ceci en chiffrant le message haché que nous notons $h, (H(m) = h)$ en utilisant la clé privée. Le signataire envoie donc le message m et sa signature S.

2.4.4. Vérification de la signature

A la réception, le récepteur calcule le haché du message m, $(H(m) = h)$ en utilisant la même fonction de hachage que le signataire. Puis il calcule $h' = S^e mod \ N$ avec (e, N) la clé publique du signataire. A la fin, il compare ce résultat obtenu à h, si le résultat h' obtenu est identique au haché h du message alors la signature est authentique sinon elle sera rejetée.

2.5. Algorithmes de base constituant le RSA : [30]

Entrée : taille de la clé //exprimée en bits.
Sortie : la clé publique (N, e) et la clé privée (N, d)

1. Prendre deux nombres premiers **p** et **q** suffisamment grands (de taille à peu près égale).
2. Calculer N = p.q,
3. Calculer $\varphi(N)$= (p-1) × (q-1),
4. Choisir un nombre e tel que $1 < e < \varphi(N)$ et le $pgcd\ (e, \varphi(N)) = 1$,
5. Prendre un nombre e qui n'a aucun facteur en commun avec $\varphi(N)$,
6. Calculer d tel que $ed \ mod \ \varphi(N)$=1.
7. **Return** N, e, d.

Algorithme 1. Génération de clés avec RSA.

Entrée : (N, e) et m // la clé publique et le texte en clair m avec $m \in$ [0.N-1].
Sortie : C // texte chiffré.

1. Calculer **C=m^e mod N**
2. **Return** C

Algorithme 2. Chiffrement avec RSA.

Entrée : (d, e) et C	// la clé privée et le texte chiffré.
Sortie : m	// texte clair.
1. Calculer $m = C^d$ **mod N**	
2. **Return** m	

Algorithme 3. Déchiffrement avec RSA.

Entrée : (N,d) et m	// la clé privée et le texte en clair m avec $m \in$ [0.N-1].
Sortie : S, m	// la signature et le texte clair.
1. Calculer h=H(m)	// le hachage du message m
2. Calculer S= h^d mod N	
3. **Return** S, m	

Algorithme 4. Signature avec RSA.

Entrée : (e, N), m,S
Sortie : Acceptation ou rejet de la signature
1. Calculer h=H(m)
2. Calculer h'=S^e mod N
3. Accepter si h=h', rejet sinon

Algorithme 5. Vérification de la signature.

- **Exemple applicatif :** [26]

Alice choisit deux nombres premiers :

$p = 11$ et $q = 19$ alors $N = pq$ donne $N = 209$.

$\varphi(N) = 10 \times 18 = 180.$

Alice choisit par exemple 7 qui est premier avec $\varphi(N) = 180$.

La clé publique est (7, 209).

Alice envoie la clé publique (7, 209) à Bob.

Alice calcule maintenant d tel que le reste de $7d$, dans la division par $\varphi(N) = 180$, soit 1.

Elle trouve 103 (on vérifie que cette fois 7×103=721 a bien pour reste 1 dans la division par 180).

La clé secrète est 103.

Notons que pour trouver cette clé secrète, Alice a eu besoin de $\varphi(N) = (11 - 1)(19 - 1)$et donc deux nombres facteurs premiers 11 et 19 qui ne sont connus que d'elle.

Bob choisit le nombre 63.

Il a reçu la clé publique (7,209) et doit calculer $m^e \ mod \ N$ soit $63^7 \ mod \ 209$.

Il trouve 123 et envoie cette valeur à Alice.

Alice reçoit 123. Elle utilise sa clé secrète $d = 103$ pour calculer $123^d \ mod \ N$ soit $123^{103} \ mod \ 209$.

Elle trouve 63.

Elle a bien trouvé le nombre choisi par Bob.

Ce calcul a nécessité l'utilisation de deux facteurs p et q choisis par Alice et inconnus de Bob.

Ces facteurs sont indispensables pour trouver la clé secrète.

Et c'est justement la détermination de ces deux facteurs qui pose une réelle difficulté lorsque ces nombres sont très grands.

2.6. Preuve RSA

2.6.1. Propriétés mathématiques utilisées

Si $N = pq$ avec p et q premiers alors $\varphi(N) = (p-1)(q-1)$ avec $\varphi(N)$ dénommé fonction d'Euler ou indicatrice d'Euler.

Petit théorème de Fermat:

Si p est un nombre premier,
Si a est un nombre premier avec p (c'est-à-dire que $pgcd(a,p) = 1$) alors

$$a^{p-1} \equiv 1 \ mod \ p$$

Théorème d'Euler:

Si $pgcd(a,N) = 1$ alors $a^{\varphi(N)} \equiv 1 \ mod \ N$.

La démonstration à suivre repose sur ces propriétés mathématiques des nombres premiers.

2.6.2. Démonstration

Nous noterons m le message saisi par Bob, e l'exposant de chiffrement et N le modulo.

Le résultat repose sur les choix effectués par Alice pour les valeurs de e et d et enfin de

$N = pq$ avec p et q deux nombres premiers.

Alice a choisit les entiers e et d tel que $ed \equiv 1 \ mod \ \varphi(N)$.

Nous savons donc qu'il existe un nombre entier k tel que $ed = k\varphi(N) + 1$.

La clé publique est le couple (N, e).

La clé secrète est d.

Bob a envoyé le nombre $C \equiv m^e \ mod\ N$ à Alice.

Alice a alors calculé $C^d \equiv (m^e)^d \ mod\ N$.

Nous allons montrer qu'elle trouve bien la valeur de m envoyé par Bob.

Nous avons :

$C^d \equiv (m^e)^d \ mod\ N \equiv m^{ed} \ mod\ N \equiv m^{k\varphi(N)+1} mod\ N \equiv m^{k\varphi(N)}\, m \ mod\ N$

Retenons avec $C^d \equiv m^{ed} \ mod\ N$

$$m^{ed} \equiv m^{k\varphi(N)}\, m \ mod\ N \ldots\ldots\ldots\ldots\ldots (1)$$

Plusieurs cas se présentent :

a. Premier cas : $pgcd(m, N) = 1$

D'après le théorème d'Euler si $pgcd(m, N) = 1$ alors $m^{\varphi(N)} \equiv 1 \ \ mod\ N$ Ainsi

$C^d \equiv m^{k\varphi(N)}\, m \ mod\ N$

$C^d \equiv (m^{\varphi(N)})^k\, m \ mod\ N$

$C^d \equiv 1^k\, m \ mod\ N$

$C^d \equiv m \ mod\ N$……. CQFD.

b. Deuxième cas $pgcd(m, N)$ différent de 1 :

Comme $N = pq$ avec p et q premiers, m est divisible par p seulement ou par q seulement ou par les deux à la fois.

- **Si m est divisible par p seulement (donc premier avec q).**

Alors m peut s'écrire $m = kp$ avec k entier.

Ainsi

$kp \equiv 0 \ mod\ p$ soit $m \equiv 0 \ mod\ p$ et on a aussi $m^{ed} \equiv (kp)^{ed} mod\ p$

$m^{ed} \equiv 0 \ mod\ p$

De $m \equiv 0 \ mod\ p$ et $m^{ed} \equiv 0 \ mod\ p$ on obtient $m^{ed} \equiv m \ mod\ p$

On peut donc dire qu'il existe un entier A tel que

$$m^{ed} - m = Ap \ldots\ldots\ldots\ldots\ldots (2)$$

Par ailleurs on a avec (1) :

$m^{ed} \equiv m^{k\varphi(N)}\, m \bmod N$

$\equiv m^{\varphi(N)\mathrm{k}}\, m \bmod$

$\equiv m^{(q-1)(p-1)k}\, m \bmod N$

$\equiv m^{(q-1)k(p-1)}\, m \bmod N$

D'après le petit théorème de Fermat, comme m est premier avec q, nous avons :

$m^{(q-1)} \equiv 1 \bmod q$ d'où $m^{ed} \equiv m^{(q-1)k(p-1)} m \equiv 1^{k(p-1)} m \equiv m \bmod q$

On obtient $m^{ed} \equiv m \bmod q$ on peut dire donc qu'il existe un entier B tel que

$$m^{ed} - m = Bq \quad \ldots\ldots\ldots\ldots\ldots (3)$$

Grâce a (2) et (3), nous pouvons écrire que p et q divisent $m^{ed} - m$, donc N

divise $m^{ed} - m$ (car $N = pq$ avec p et q premiers).

Cela signifie que : $m^{ed} - m \equiv 0 \bmod N$ c'est-à-dire que :

$m^{ed} \equiv m \bmod N$ et en fin que $C^d \equiv m \bmod N$……....CQFD

- **Si m est divisible par p et par q (premiers et donc premiers entre eux)**

Alors $m \equiv 0 \bmod pq$ et $m^{ed} \equiv 0 \bmod pq$ c'est-à-dire comme $N = pq$, le résultat $m^{ed} \equiv m \bmod N$ est trivial et $C^d \equiv m \bmod N$……CQFD

2.7. Conseils d'utilisation du RSA

Pour une bonne sécurité pour le chiffrement RSA, il faut respecter certaines règles tel que :

- Ne pas chiffrer des blocs courts.
- $(p-1) \times (q-1)$ doit être grand (grand facteur premier).
- Utiliser N très grand.
- Ne pas utiliser de N communs à plusieurs clés.
- Si (N, d) est compromise ne plus utiliser N.

2.8. Catégories d'attaques sur RSA

La communauté cryptographique étudie la solidité du RSA face à une panoplie d'attaques. Ces attaques ont été réparties en trois 03 catégories selon la faille qu'elles exploitent : [30]

2.8.1. Attaques mathématiques

Les attaques mathématiques sont des attaques qui cherchent à trouver une faille dans les fondements mathématiques mêmes de l'algorithme.

➢ Factorisation de modulo N

Le RSA est basé sur la difficulté de factoriser un entier ayant plusieurs milliers de chiffres décimaux, on peut facilement démontrer qu'un tel nombre est composé, pour autant les méthodes utilisées ne fournissent en général aucune information sur ses diviseurs premiers. Il existe certaines méthodes permettant de les déterminer, nous allons décrire l'une de ces méthodes qui est la division successive.

La méthode des divisions successives, qui est la plus simple, consiste à déterminer les diviseurs premiers de N plus petits que $\sqrt{N}$, ensuite il suffit de diviser N successivement par tous les nombres premiers retrouvés jusqu'à atteindre p, qui est le plus petit diviseur premier de N ($p \leq \sqrt{N}$) .

Cette méthode est une technique d'attaque par force brute, bien qu'elle ne permette pas de factoriser des entiers n'ayant que des grands facteurs premiers et par rapport aux autres méthodes elle est inefficace, elle est néanmoins incontournable, et c'est la première méthode que l'on doit essayer afin de factoriser N. [28]

➢ Attaque de Wiener

Proposée par le cryptologue Michael J. Wiener. Cette attaque permet de retrouver facilement la clé privée à partir de la clé publique (e, N) , lorsque les conditions $d < \frac{1}{3}N^{\frac{1}{4}}$ (d trop petit) et $q<p<2q$ (p et q trop proches) sont remplies, il est facile de retrouver d.
Cette attaque a été améliorée par Dan Boneh et Glen Durfee pour tous les exposants d inférieurs ou égaux à $N^{0.292}$. [29]

➢ Attaque en connaissant quelques bits de la clé privée

Une autre technique d'attaque apparait lorsqu'on possède quelques bits de la clé privée d. D. Boneh et al ont montré que si la taille de la clé d est de k bits alors la connaissance de $\frac{k}{4}$ bits de poids faible est largement suffisante pour récupérer la clé d. [31]

2.8.2. Attaques de protocole

Même si RSA est solide, la façon dont on l'utilise n'est pas neutre. Par exemple si on envoie le même message à 3 personnes différentes, chiffrés avec 3 clés RSA de ces personnes, on peut facilement retrouver le message en clair à partir des 3 messages chiffrés en utilisant la propriété de multiplicativité de la fonction RSA **:** $f(x \times y) = f(x) \times f(y)$.
Il est également risqué de chiffrer plusieurs messages liés au moyen de la même clé publique RSA. [32]

➢ Attaque d'Hastad

L'attaque de Hastad (attaque de Broadcast) est une attaque qui s'applique au cas où un même message, ou plus généralement des messages liés entre eux par une relation connue, sont chiffrés et envoyés à différentes personnes.
Pour illustrer l'attaque sur un cas très simple : supposons d'abord que Bob utilise le même exposant de chiffrement *e* pour envoyer à *k* personnes différentes le même message *m*.
Chaque personne P_i possède sa propre clé publique (N_i, e). On suppose que le message *m* est inférieur à tous les N_i. Pour envoyer *m* à la i-ème personne P_i, Bob chiffre naïvement le message *m* avec $C_i = m^e \ mod \ N_i$. L'attaquant Marvin peut espionner la communication et obtenir chacun des *k* messages chiffrés.
Pour simplifier, on peut supposer que l'exposant public *e* est égal à 3. On voit facilement que dans ce cas, Marvin peut obtenir *m* si le nombre de chiffrés *k* est supérieur ou égal à 3. En effet, supposons que Marvin obtienne C_1, C_2, C_3 où : $C_1 = m^3 mod \ N_1$, $C_2 = m^3 mod \ N_2$, $C_3 = m^3 mod \ N_3$.
On peut supposer que $pgcd(N_i, N_j) = 1$ pour tout $i \neq j$, car sinon, Marvin pourrait factoriser au moins deux modules N_i .En appliquant le théorème des restes chinois (CRT) à C_1, C_2, C_3 on obtient un entier C'compris entre 0 et $N_1.N_2.N_3$ Tel que
$C' = m^3 mod \ N_1.N_2.N_3$.Comme le message *m* est inférieur à chacun des modules N_i, on a $m^3 < N_1.N_2.N_3$.
L'égalité $C' = m^3$ se vérifie par conséquent sur les entiers, et Marvin retrouve *m* en calculant (dans $\mathbb{Z}$) la racine cubique de C'. On voit que l'attaque se généralise à tout exposant de chiffrement *e*, à condition que le nombre de chiffrés *k*, soit supérieur ou égal à *e*. L'attaque n'est donc réalisable que pour *e* petit. [34]

2.8.3. Attaques physiques

Si l'on suppose que la fonction RSA est mathématiquement solide, on peut alors construire des protocoles surs pour le chiffrement et qui admettent des preuves de sécurité. Malgré ces preuves, certaines attaques restent possibles. Celles-ci ne s'attaquent pas au problème mathématique, mais elles utilisent le fait que le dispositif électronique qui fait les calculs de chiffrement, n'est pas une abstraction mathématique : dans le monde physique, il met un certain temps à calculer, il consomme du courant électrique, il peut faire des erreurs de calcul…etc. Avec ces paramètres supplémentaires, l'attaquant parvient parfois à ses fins. [35]

➢ Attaque sur le temps de calcul

L'algorithme de calcul de la fonction RSA est toujours le même. En particulier il fait intervenir une boucle pour le calcul de $C^d mod\, N$ (ou d est la clé secrète).$d = d_1, d_2, d_3, \dots, d_n$, chaque bit $d_i$est égal à 0 ou 1. Or dans la boucle de calcul, on trouve une instruction du type 'si $d_i = 1$ alors faire tel calcul sinon ne pas le faire'.
En mesurant les temps de calcul pour de nombreuses valeurs initiales de C (message crypté), on peut déduire si le calcul qui est fait lorsque $d_i = 1$ a été réellement effectué et de retrouver la clé secrète d bit par bit. On peut contourner ce type d'attaque en masquant les différences de temps de calcul. [35]

➢ Attaque sur la consommation électrique

Le même type d'attaque peut être effectué avec la consommation électrique pour chacun des calculs, on mesure la courbe de consommation électrique du composant qui fait le calcul. En analysant la statistique de la consommation électrique à chaque étape du calcul, on déduit au fur et à mesure la valeur de la clé secrète d.
Des méthodes existent pour fausser la consommation électrique et rendre cette information inutilisable. [35]

➢ Attaque par injection de faute

Les attaques par faute sont une famille de techniques qui consistent à produire volontairement des erreurs dans le cryptosystème. Ces attaques peuvent porter sur des composants matériels ou logiciels. Elles ont pour but de provoquer un comportement inhabituel des opérations cryptographiques dans le but d'en extraire des informations secrètes (comme une clé de chiffrement). Une attaque par faute peut être couplée à d'autres attaques comme l'attaque sur la consommation électrique et l'attaque sur le temps de calcul .

Les attaques sont possibles sous l'hypothèse que l'attaquant peut affecter l'état interne du système en écrivant des valeurs par exemple en mémoire ou sur un bus informatique. [35]

2.9. Outils mathématiques

2.9.1. Factorisation des entiers

La factorisation des entiers est un problème crucial en cryptographie, car elle permet de casser le RSA. La méthode la plus élémentaire pour factoriser un entier N consiste à prendre tous les entiers inférieurs à N, et à tester s'ils divisent N (algorithme de force brute). C'est bien sûr un algorithme inutilisable si N est grand. Un premier raffinement consiste à ne prendre que les entiers inférieurs à racine de N (si N n'est pas premier, N admet forcément un diviseur inférieur à racine de N). C'est beaucoup mieux, mais encore insuffisant pour les entiers de 1000 chiffres que l'on souhaite factoriser.

L'idée utilisée par les algorithmes modernes est due à l'arithméticien français Pierre de Fermat : [36]

si on trouve deux entiers x et y, non égaux, non opposés, tels que $x^2 = y^2 \, mod \, N$, alors$(x - y)(x + y) = 0 \, mod \, N$, et $pgcd \, (x + y, N)$ou $pgcd \, (x - y, N)$ donne un diviseur non trivial de N Il reste à trouver de tels nombres x et y. Posons x un nombre juste supérieur à racine de N. x^2 est juste supérieur à N, et $x^2 = a \, mod \, N$, avec a petit. Il est donc facile de factoriser a, et avec un peu de chances, en essayant plusieurs x, on peut espérer trouver un a tel que $a = y^2$.

Exemple : Soit à factoriser N = 3337, sa racine carrée vaut 57.
On teste :

– $58^2 = 27 = 3^3 \, mod \, 3337$: ne convient pas.
– $59^2 = 144 = 12^2 \, mod \, 3337$: convient !

Alors, $pgcd(59 + 12{,}3337) = 71$ donne un diviseur non trivial de N. Le second facteur premier est 49.

2.9.2. Petit théorème de Fermat amélioré

Nous connaissons le petit théorème de Fermat :

Théorème: (Petit théorème de Fermat). [37]
Si p est un nombre premier,
Si a est un nombre premier avec p (c'est-à-dire que $pgcd(a, p) = 1$) alors

$$a^{p-1} \equiv 1 \, mod \, p$$

Théorème: (Petit théorème de Fermat amélioré). [37] Soient p et q deux nombres premiers distincts et soit $N = pq$. Pour tout $a \, \epsilon \, Z$ tel que $pgcd(a, N) = 1$ alors :

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

on note $\varphi(N) = (p - 1)(q - 1)$ la fonction d'Euler l'hypothèse $pgcd(a, N) = 1$ équivaut ici à ce que a ne soit divisible ni par p, ni par q.

Par exemple pour p = 5 et a = 7 :
$\varphi(N) = 4 \times 6 = 24$ Alors pour a = 1, 2, 3, 4, 6, 8, 9 , 11, 12, 13, 16, 17, 18, ... on a bien :

$$a^{24} \equiv 1 \, mod \, 35$$

Démonstration : Notons $C \equiv a^{(p-1)(q-1)}$ Calculons C modulo p:

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

Où l'on a appliqué le petit théorème de Fermat : $a^{(p-1)} \equiv 1\ mod\ p$ car p ne divise pas a.
Calculons ce même C mais cette fois modulo q :

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

Où l'on a appliqué le petit théorème de Fermat : $a^{(q-1)} \equiv 1\ mod\ q$ car q ne divise pas a.
Conclusion partielle : $C \equiv 1\ mod\ p$ et $C \equiv 1\ mod\ q$.
Nous allons en déduire que $C \equiv 1\ mod\ pq$.
Comme $C \equiv 1\ mod\ p$ alors il existe $\alpha\ \epsilon\ Z$ tel que $C = 1 + \alpha p$. [37]

2.9.3. Algorithme d'Euclide

L'algorithme d'Euclide sur deux nombres entiers positifs a et b avec $a > b \geq 0$ procéde comme suit :

Si $b = 0$ l'algorithme termine et rend la valeur a ;

Sinon l'algorithme calcule le reste r de la division euclidienne de a par b puis recommence avec $a \coloneqq b$ et $b \coloneqq r$.

Formellement l'algorithme d'Euclide construit deux suites finies d'entiers par récurrence : la suite (q_n) des quotients et la suite (r_n) des restes :

$r_0 = a$, $r_1 = b$;

Pour $n \geq 1$, r_{n+1} et $q_n$sont le reste et le quotient de la division euclidienne de r_{n-1} par r_n , Définis par les deux conditions :

$r_{n-1} = r_n q_n + r_{n+1}$ et $0 \leq r_{n+1} < r_n$.

Par définition la suite r_n est une suite strictement décroissante d'entiers positif : elle est donc finie et s'arrête au premier n tel que $r_n = 0$. [38]

Exemple

Soit à calculer le $pgcd$ de 21 et 15.

On réalise la division euclidienne de $a = 21$ et $b = 15$,

Le quotient est 1 et le reste 6.L'algorithme continue avec $a := b$ est $b :=$ le précédent reste, jusqu'à trouver un reste nul. L'algorithme s'arrête alors est retourne le dernier reste non nul trouvé, Ici $r_3 = 3$ qui est bien le $pgcd$ de 21 et 15.

2.9.4. L'algorithme d'Euclide étendu

L'algorithme d'Euclide étendu est une variante de l'algorithme d'Euclide qui permet, à partir de deux entiers a et b, de calculer non seulement leur plus grand commun diviseur

$(pgcd)$, mais aussi un couple de coefficients de Bézout, c'est-à-dire deux entiers u et v tel que

$$au + bv = pgcd(a, b).$$

Cet algorithme est particulièrement utilisé lorsqu'on souhaite calculer l'inverse multiplicatif d'un entier. La question importante est comment calcule-t-on les coefficients u et v. L'idée principale de l'algorithme est d'effectuer les mêmes étapes que pour l'algorithme d'Euclide, mais en exprimant à chaque itération le reste comme une combinaison linéaire de a et b. Puisque le dernier reste est le $pgcd$, celui-ci sera alors exprimé comme une combinaison linéaire de a et b. [39]

2.10.Conclusion

Dans ce chapitre, nous avons mis l'accent sur l'algorithme RSA, nous avons détaillé son fonctionnement, puis nous nous sommes intéressés à sa cryptanalyse et ses trois catégories d'attaques qui sont les attaques mathématiques, de protocole et attaques physiques, et nous avons introduit les outils mathématiques utilisés pour la simulation de nos deux attaques (par factorisation et de Wiener) dans ce qui suit nous allons présenter cette simulation.

Chapitre 3

Conception et simulation de quelques attaques sur le RSA

3.1. Introduction

Ce chapitre va être consacré aux aspects pratiques pour la réalisation de notre application «Simulation d'attaques sur RSA».
Son but est de simuler deux attaques à savoir l'attaque par factorisation et l'attaque de Wiener sur le module de RSA lors du transfert des messages afin de pouvoir vérifier la partie intégrité et authenticité du message.

3.2. Ressources utilisée

3.2.1. Ressources matérielles

- Processeur **Intel® Core™ i3-6100U** CPU d'une fréquence de **2.30 GHZ**.
- Une mémoire vive d'une capacité de **4Go.**
- Système d'exploitation **64 bits**.
- Une carte graphique de **512 Mo.**

3.2.2. Ressources logicielles

- IntelliJ IDEA Community Edition 2020.2.2 x64.
- JavaFX.
- Scene Builder.

3.3. Présentation de l'application

3.3.1. Objectif

L'objectif de notre travail est de réaliser une application de simulation de l'attaque par factorisation et l'attaque de Wiener sur le système cryptographique RSA afin de démontrer que malgré la robustesse de RSA celui-ci reste sensible aux attaques lorsque la clé est de petite taille.

3.3.2. Description

Notre application simule l'attaque par factorisation et l'attaque de Wiener sur le système cryptographique RSA, en mettant en évidence deux scénarios et cas possibles qui sont :

– Le mode normal.
– Le mode simulation d'attaque.

3.3.3. Description de la fenêtre principale

Les composants de la fenêtre principale de notre application sont les suivants:

➢ **Partie supérieur**

- **Boutons :**

1. «Changer de type d'attaque » permet de choisir l'attaque à simuler à savoir l'attaque par factorisation ou l'attaque de Wiener.
2. «Changer les rôles » sert à intervertir les rôles entre l'expéditeur et le destinataire.
3. « Reset » utilisé pour réinitialiser l'opération.
4. Boutons radios pour choisir le mode d'utilisation de l'application (mode normal ou mode simulation d'attaque).
5. « Générer les clés » permet de générer les clés publiques et privées pour l'expéditeur et le destinataire.
6. « Envoyer » permet à l'expéditeur d'envoyer un message au destinataire.

- **Champs de saisie :**

1. « N, Clé publique et Clé Privée » qui sont initialement vide et qui seront automatiquement remplis après avoir cliqué sur le bouton générer les clés.
2. « Message » permet à l'expéditeur de saisir le message a envoyé et au destinataire d'afficher le message reçu.

- **Liste déroulante :**

1. Offre la possibilité de choisir le nombre de bits de P et Q.

➢ Partie inferieur

- **Boutons :**

1. « Intercepter » permet d'intercepter le message crypté qui a été envoyé par l'expéditeur.
2. « attaquer » sert à factoriser N de manière à trouver les deux nombres premiers p et q puis calculer l'exposant de déchiffrement d qui servira à décrypter le message envoyé par l'expéditeur.
3. « Envoyer » offre à l'attaquant la possibilité d'envoyer le message au destinataire.

- **Champs de saisies :**

1. « Message » sert à modifier ou pas le messager initial envoyé par l'expéditeur.

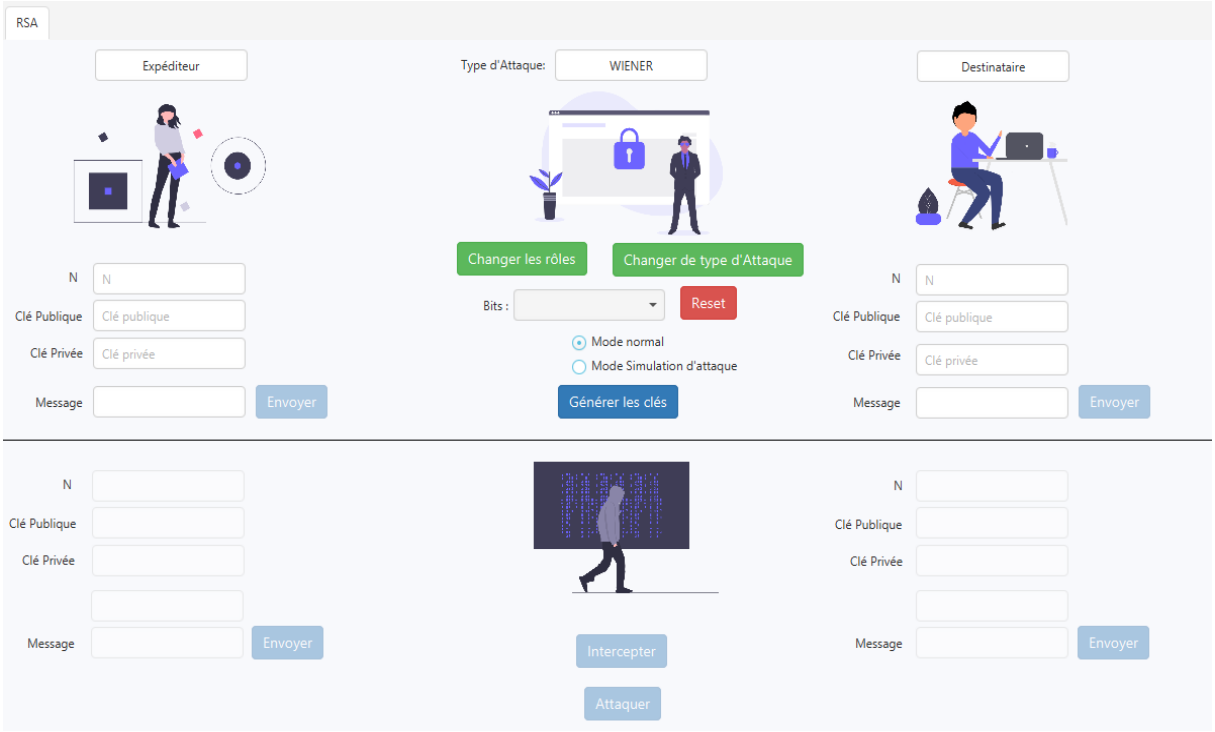

Figure 15. Interface principale.

3.4. Utilisation

3.4.1. Attaque par Factorisation/attaque de Wiener : Mode normal

Afin de pouvoir envoyer un message en mode normal autrement dit sans intervention d'une tierce personne à partir de l'expéditeur vers le destinataire il faudra :

a. Déterminer l'expéditeur et le destinataire.
b. Choisir l'attaque à utiliser (Factorisation/ Wiener).
c. Choisir le nombre de bits de P et Q.
d. Activer le mode normal.

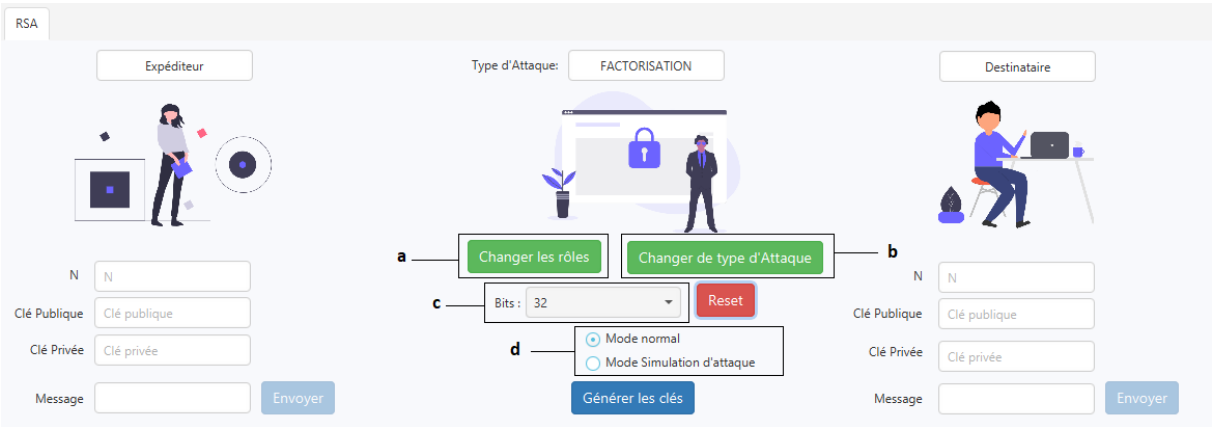

Figure 16. Attaque (factorisation/Wiener) en mode normal.

e. Générer les clés (clés publiques, clés privées) pour l'expéditeur et le destinataire.

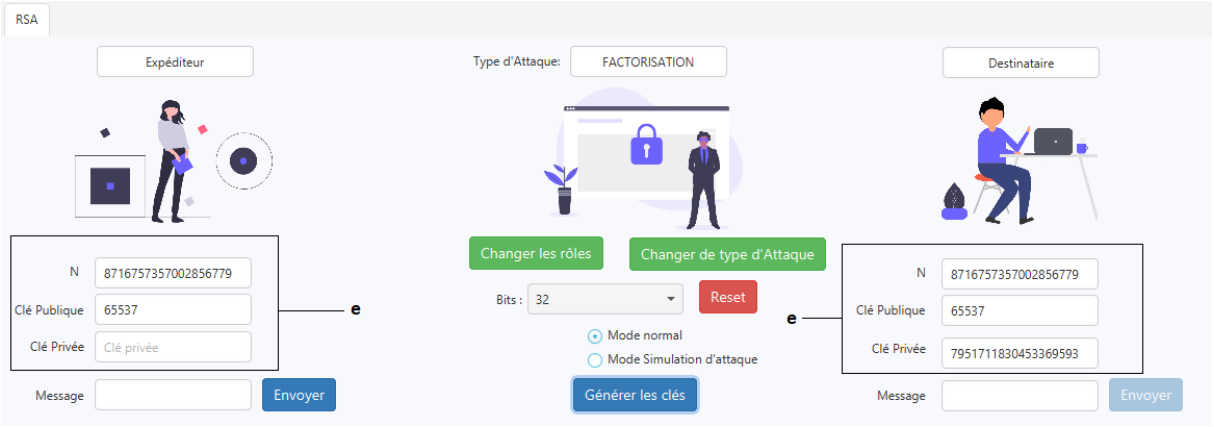

Figure 17. Génération de clés pour l'expéditeur et le destinataire (Mode normal).

f. Saisir et envoyer le message par l'expéditeur.

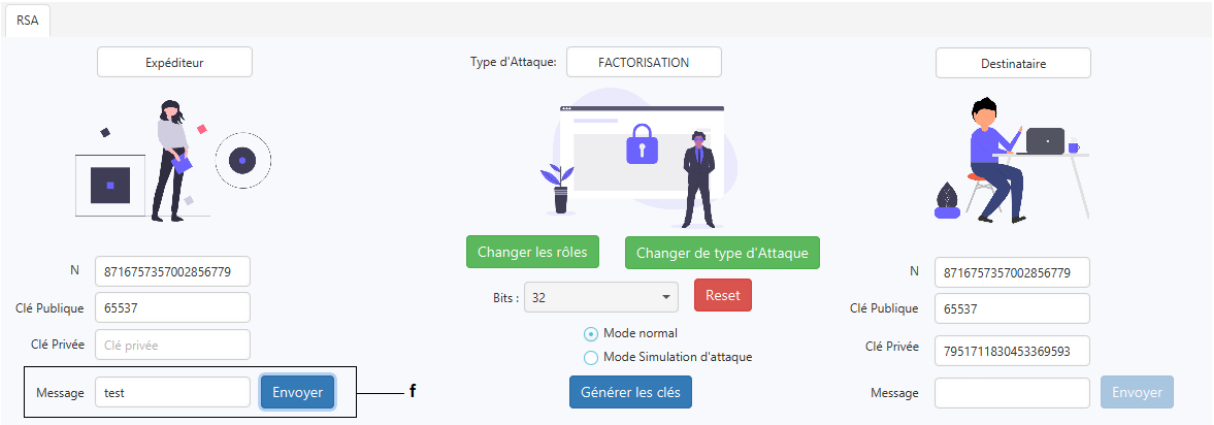

Figure 18. Saisie et envoi du message par l'expéditeur (Mode normal).

g. Réceptionner le message par le destinataire.

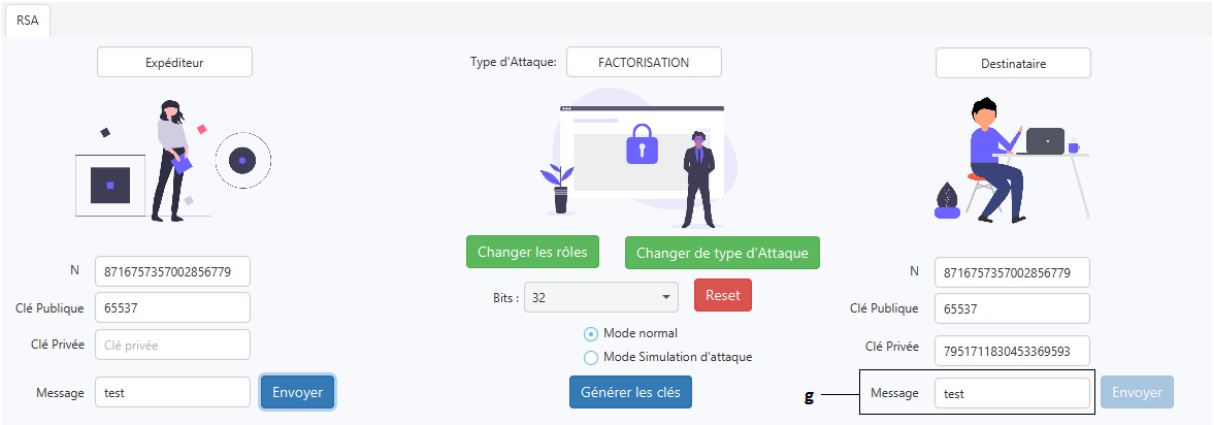

Figure 19. Réception du message par le destinataire (Mode normal).

3.4.2. Attaque par factorisation/ Attaque de Wiener : mode simulation d'attaque

a. Après avoir déterminé l'expéditeur et le destinataire, choisir le type d'attaque (par factorisation ou de Wiener) et le nombre de bits de P et Q, il faudra activer le mode simulation d'attaque puis cliquer sur le bouton « Générer les clés » ainsi les clés seront générées automatiquement pour l'expéditeur et le destinataire.

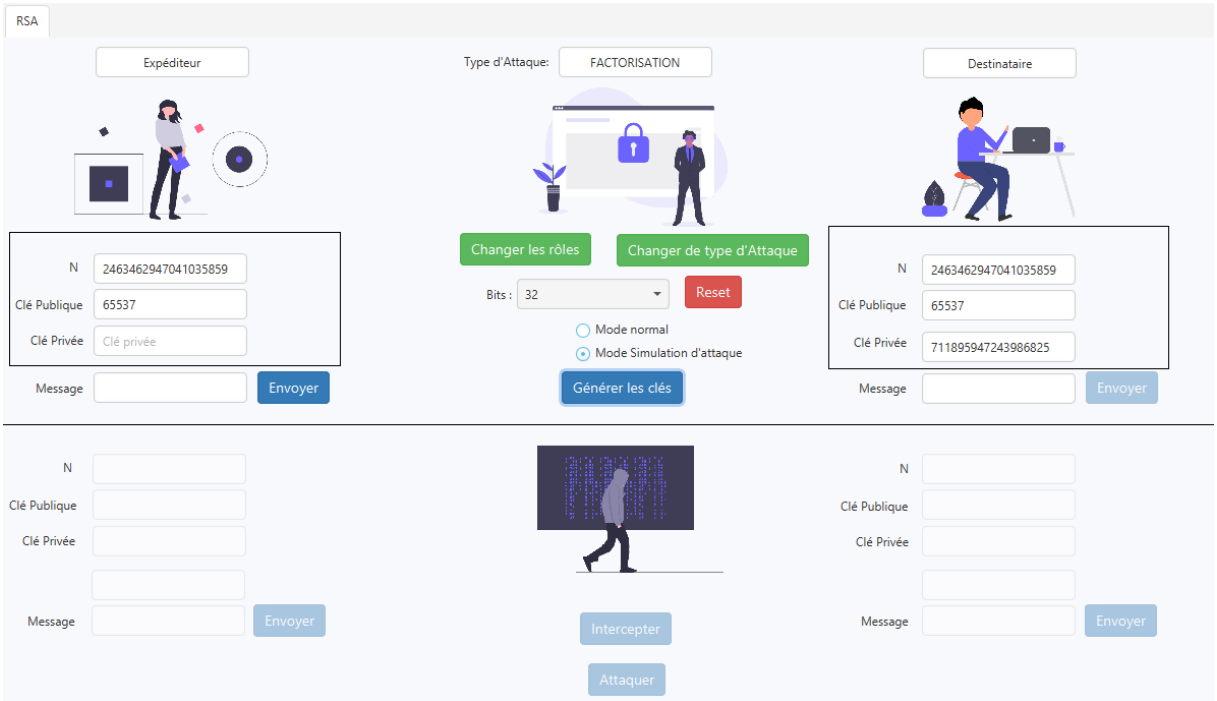

Figure 20. Génération de clés pour l'expéditeur et le destinataire (mode simulation d'attaque).

b. A ce moment l'expéditeur saisi le message et clique sur le bouton « Envoyer » et cela implique l'activation du bouton « Intercepter » permettant à l'attaquant d'intercepter le message envoyé par l'expéditeur.

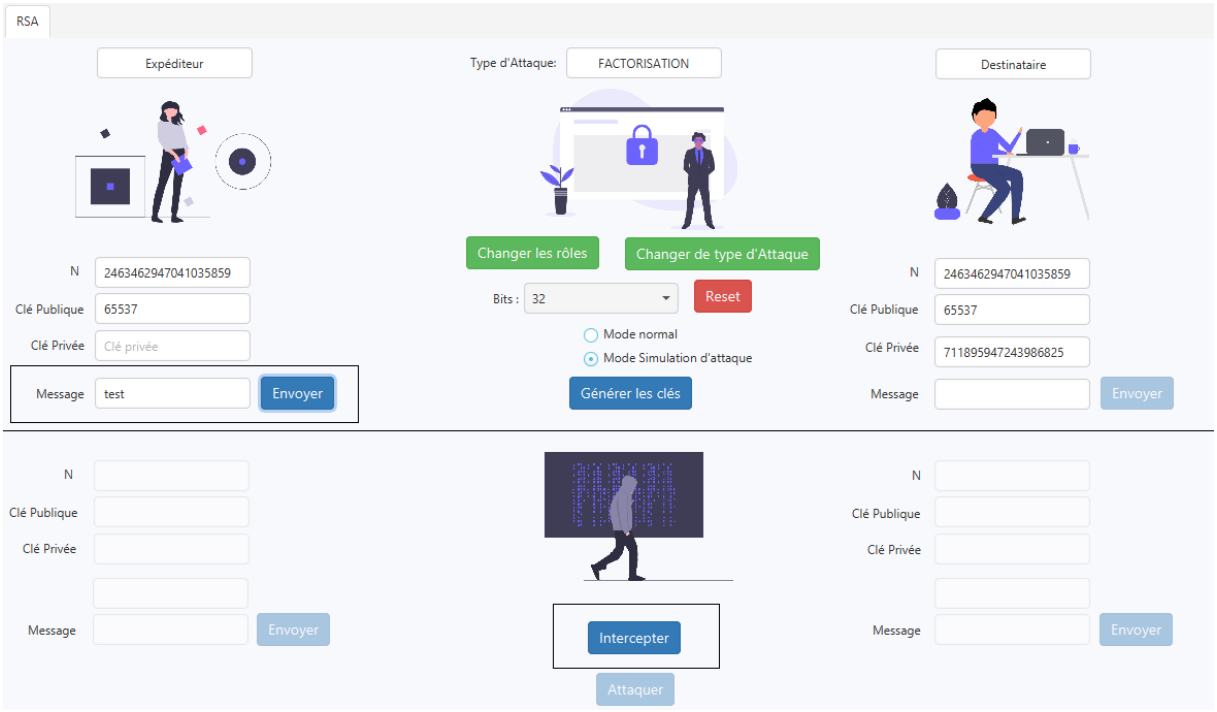

Figure 21. Envoi du message par l'expéditeur et activation du bouton « Intercepter ».

c. En cliquant sur le bouton « intercepter » l'intrus intercepte le message chiffré envoyé par l'expéditeur en destination du destinataire, et ainsi la possibilité de déchiffrer le message s'offre à lui et ceci grâce au bouton « attaquer » qui est maintenant activé.

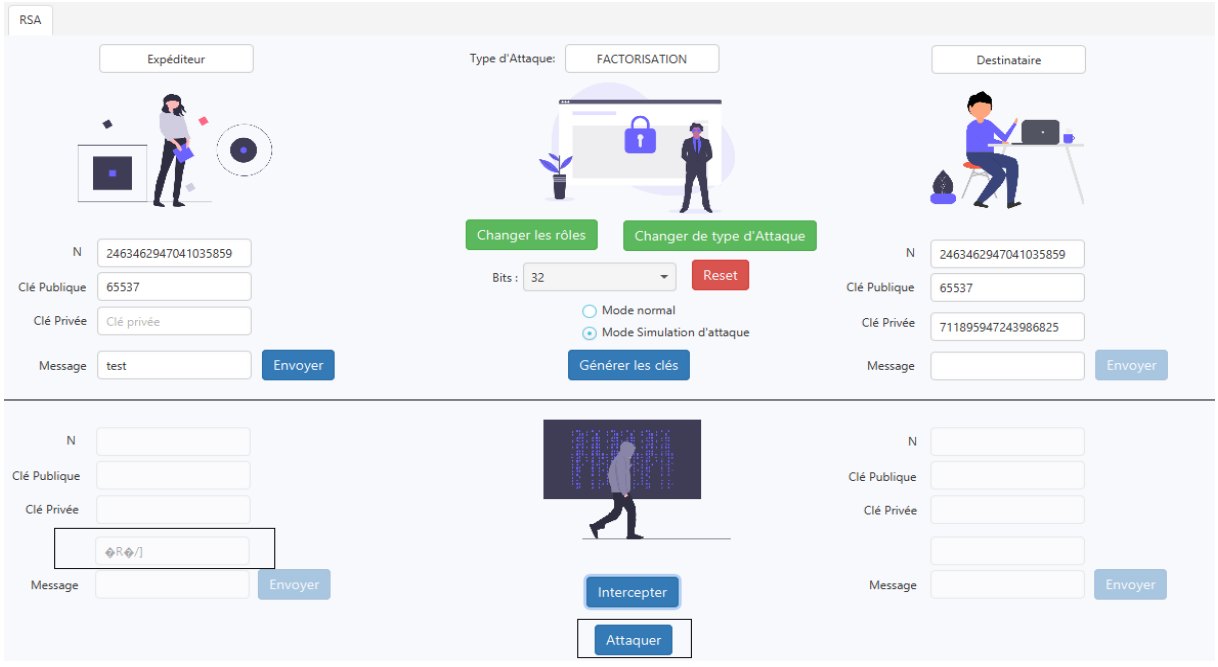

Figure 22. Interception du message et activation du bouton « Attaquer ».

d. Apres que l'intrus ait cliqué sur le bouton « Attaquer », une pop up sera affichée cette dernière permettra de trouver les nombres premiers P et Q qui permettront par la suite de calculer D (Clé privée).

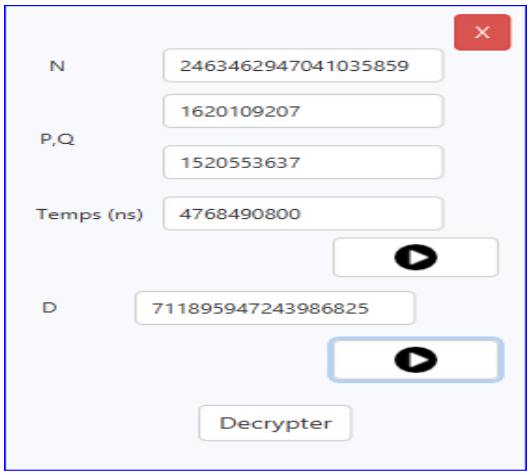

Figure 23. Factorisation de la clé privée et calcule du D.

e. Une fois que les calculs ont été effectués l'intrus peut enfin déchiffrer le message en cliquant sur le bouton « Décrypter », cette opération permet d'afficher le message en clair.

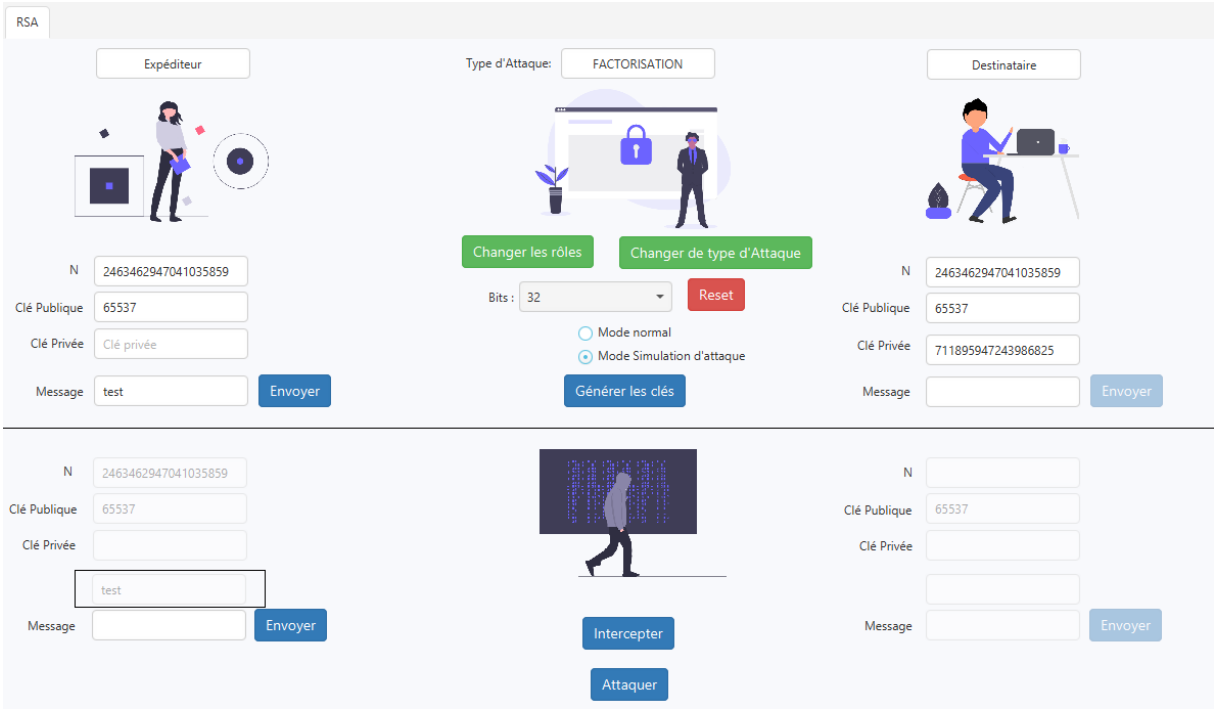

Figure 24. Déchiffrement du message.

f. A ce moment l'intrus a le choix de renvoyer le message tel quel au destinataire principal ou de le modifier et envoyer un message différent du message initialement envoyé par l'expéditeur, en cliquant sur « Envoyer » le message sera envoyer au destinataire.

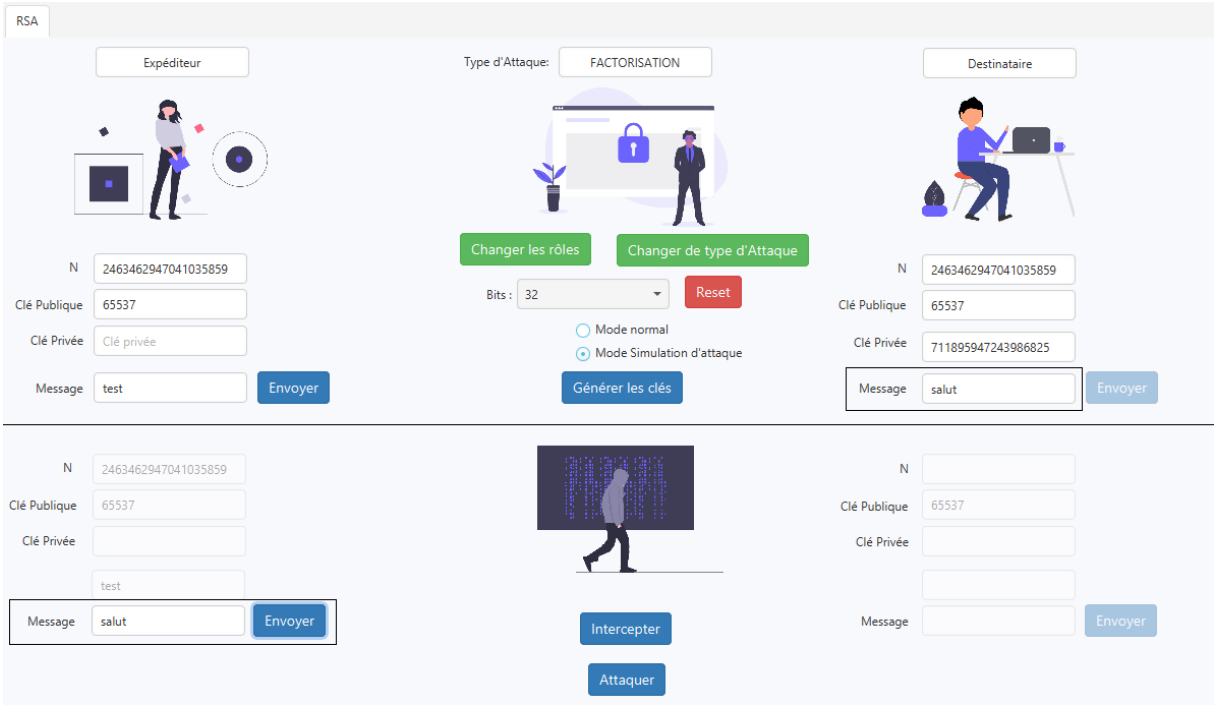

Figure 25. Modification ou pas du message initial puis envoi au destinataire.

3.5. Procédure suivie pour la réalisation des attaques

3.5.1. Procédure suivie pour la réalisation de l'attaque par factorisation

Pour réaliser l'attaque par factorisation il a fallu :

1. Calculer $\sqrt{N}$.
2. Calculer $N \ mod \ QP$ tel que $QP = \sqrt{N} - 1$
   Si $N \ mod \ QP = 0$ alors on attribue la valeur QP à P ou bien à Q
   Sinon $QP \leftarrow QP - 1$ (on soustrait 1 à chaque itération jusqu'à ce que $N \ mod \ QP = 0$)
3. Calculer q
   Soit $N = p \times q \ \rightarrow \ q = \frac{N}{p}$
4. Calculer D (exposant de déchiffrement)

$$ed = 1 \ mod \ \varphi(N) \qquad \rightarrow \qquad d = \frac{1 \ mod \ \varphi(N)}{e}$$

Tel que : $\varphi(N) \ = \ (p-1) \times (q-1)$

Et e un entier choisi de manière à être premier avec $\varphi(N)$ (appelé exposant de chiffrement).

5. Déchiffrer le message m

$m = C^d mod\ N$

Tel que : C est le texte chiffré.

```
//region Calcul de P et Q par factorisation
public boolean PQfinder(BigInteger n, BigInteger e){
    this.setN(n);
    this.setE(e);
    BigInteger QP;

    BigInteger square = n.sqrt(); //Calculer la racine carré de N => RC;
    long startTime = System.nanoTime(); // Démarrer le chronomètre

    for(QP = square.subtract(BigInteger.ONE); QP.compareTo(BigInteger.ZERO) > 0; QP = QP.subtract(BigInteger.ONE)){
            // Boucle servant à soustraire 1 de RC => QP
        BigInteger res = n.mod(QP); // Calculer le résultat = N mod QP
        if(res.compareTo(BigInteger.ZERO) == 0){ // Si le résultat = 0 arrêter la boucle et attribuer QP à P ou Q
            break;
        }
    }

    this.PQ = n.divide(QP); // N = p * q donc q = N/p
    this.QP = QP;

    long finishTime = System.nanoTime(); // Arrêter le chronomètre

    setTime(finishTime - startTime); // Calculer le temps écoulé

    return true;
}
//endregion Calcul de P et Q
```

Figure 26. Code d'attaque par factorisation.

3.5.2. Procédure suivie pour la réalisation de l'attaque de Wiener

Afin de mettre en œuvre l'attaque de Wiener il faut utiliser l'algorithme suivant

$(q_1, \ldots, q_m; r_m) \leftarrow$ EUCLIDEAN ALGORITHM(b, n)
$c_0 \leftarrow 1$
$c_1 \leftarrow q_1$
$d_0 \leftarrow 0$
$d_1 \leftarrow 1$
for $j \leftarrow 2$ **to** m
do {
  $c_j \leftarrow q_j c_{j-1} + c_{j-2}$
  $d_j \leftarrow q_j d_{j-1} + d_{j-2}$
  $n' \leftarrow (d_j b - 1)/c_j$
  comment: $n' = \phi(n)$ if c_j/d_j is the correct convergent
  if n' is an integer
  then {
    let p and q be the roots of the equation
      $x^2 - (n - n' + 1)x + n = 0$
    if p and q are positive integers less than n
      then return (p, q)
  }
}
return ("failure")

Algorithme 6. Algorithme de Wiener.

1. Calculer les fractions continues à l'aide de l'algorithme d'Euclide

$$(q_1, \ldots, q_m; r_m)$$

$r_0 \leftarrow a$
$r_1 \leftarrow b$
$m \leftarrow 1$
while $r_m \neq 0$
do {
  $q_m \leftarrow \lfloor \frac{r_{m-1}}{r_m} \rfloor$
  $r_{m+1} \leftarrow r_{m-1} - q_m r_m$
  $m \leftarrow m + 1$
}
$m \leftarrow m - 1$
return $(q_1, \ldots, q_m; r_m)$
comment: $r_m = \gcd(a, b)$

Algorithme 7. Algorithme d'Euclide.

2. Calculer C_i et D_i selon l'algorithme de Wiener

$$C_i = q_i c_{i-1} + c_{i-2}$$

$$D_i = q_i d_{i-1} + d_{i-2}$$

$$n' = \frac{(D_i \times b) - 1}{C_i} \qquad selon\ notre\ code\ b = E1$$

3. Résoudre l'équation du second degré

$$x^2 - (n - n' + 1)x + n = 0$$

a. Calculer $n - n' + 1$

b. Calculer $\Delta = b^2 - 4\,ac$

c. Calculer X_1 et X_2

$$X_1 = \frac{-b + \sqrt{\Delta}}{2a} \qquad et \qquad X_2 = \frac{-b - \sqrt{\Delta}}{2a}$$

6. Calculer D (exposant de déchiffrement)

$$ed = 1\ mod\ \varphi(N) \qquad \rightarrow \qquad d = \frac{1\ mod\ \varphi(N)}{e}$$

Tel que : $\varphi(N) = (p - 1) \times (q - 1)$

Et e un entier choisi de manière a être premier avec $\varphi(N)$ (appelé exposant de chiffrement).

7. Déchiffrer le message m

$m = C^d mod\ N$

Tel que : C est le texte chiffré.

```
//region Algorithme de Wiener
public void WienerAttack(BigInteger E, BigInteger N)
{
    BigInteger N1 = N;
    BigInteger E1 = E;
    List<BigInteger> Pub = EuclidAlgo(N1, E1);// Calculer les fractions continues à l'aide de l'algorithme d'Euclide
    List<BigInteger> c = new ArrayList<>();
    List<BigInteger> d = new ArrayList<>();
    BigInteger n1 = BigInteger.ZERO;
    BigInteger p1 = BigInteger.ZERO;
    BigInteger p2 = BigInteger.ZERO;

    c.add(BigInteger.ONE);
    c.add(Pub.get(0));
    d.add(BigInteger.ZERO);
    d.add(BigInteger.ONE);

    long startTime = System.nanoTime();

    for (BigInteger i = BigInteger.ONE; i.compareTo(BigInteger.valueOf(Pub.size())) <= 0; i = i.add(BigInteger.ONE))
    {
        Calculer Ci et Di selon l'algorithme de Wiener

        //region Résolution de l'équation du second degré [x² - (n - n' + 1)*x + n = 0]
        if (c.get(i.intValueExact()).compareTo(BigInteger.ZERO) != 0)
        {
            Calculer n' = (Di * b - 1) / Ci ; b = E1

            if (n1.remainder(BigInteger.ONE).compareTo(BigInteger.ZERO) == 0)
            {
                BigInteger prod = N1.subtract(n1).add(BigInteger.ONE); // Calculer (n - n' + 1) => prod
                BigInteger descrim = prod.multiply(prod).subtract(BigInteger.valueOf(4).multiply(N1)); // Δ = b²-4ac
                                                                        // b = prod ; a = 1 ; c = N1

                Calculer X1 => P et X2 => Q
                else
                {
                    break;
                }
            }
        }
        //endregion Résolution de l'équation du second degré [x² - (n - n' + 1)*x + n = 0]
    }
```

Figure 27. Code de l'attaque de Wiener.

3.6. Factorisation du module N

Lors de la génération des clés, l'attaquant va tenter de factoriser le module de RSA N pour trouver p et q et calculer par la suite la clé privée pour déchiffrer les messages interceptés.

3.6.1. Factorisation du module N (Attaque par factorisation)

Le tableau suivant montre le temps nécessaire pour factoriser le module N de différentes tailles à savoir 8 bits, 16 bits et 32 bits respectivement.

N	P	Q	D	Temps (ns)
8651	211	41	2273	730400
1484852851	59659	24889	117208641	16426800
11704833825257785 1	2731321963	428540977	36439526357933200 1	626066800

Tableau 1. Temps nécessaire pour la factorisation de N avec l'attaque par factorisation.

Le graphe suivant montre la variation du temps calculé pour l'attaque par factorisation en fonction de la taille de p et q.

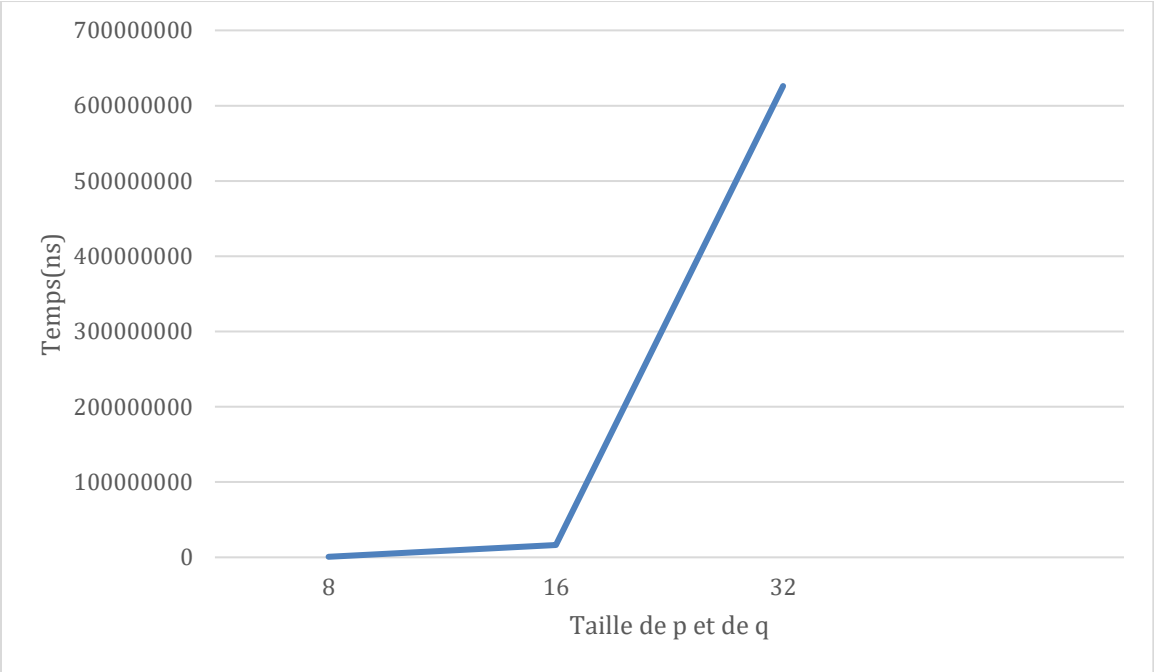

Figure 28. Graphe du temps calculé pour l'attaque par factorisation.

3.6.2. Factorisation du module N (Attaque de Wiener)

Le tableau suivant montre le temps nécessaire pour factoriser le module N de différentes tailles à savoir 16 bits, 32bits et 64bits respectivement

N	P	Q	D	Temps (ns)
590230651	26339	22409	147545481	5892800
5132328808 556915029	2894260741	1773277969	9271	8452300
12240059491 986103907628 2479658102833751	1511989170 8975564557	8095335421 430355443	87283313	8932300

Tableau 2. Temps nécessaire pour la factorisation de N avec l'attaque de Wiener.

Le graphe suivant montre la variation du temps calculé pour l'attaque de Wiener en fonction de la taille de p et q.

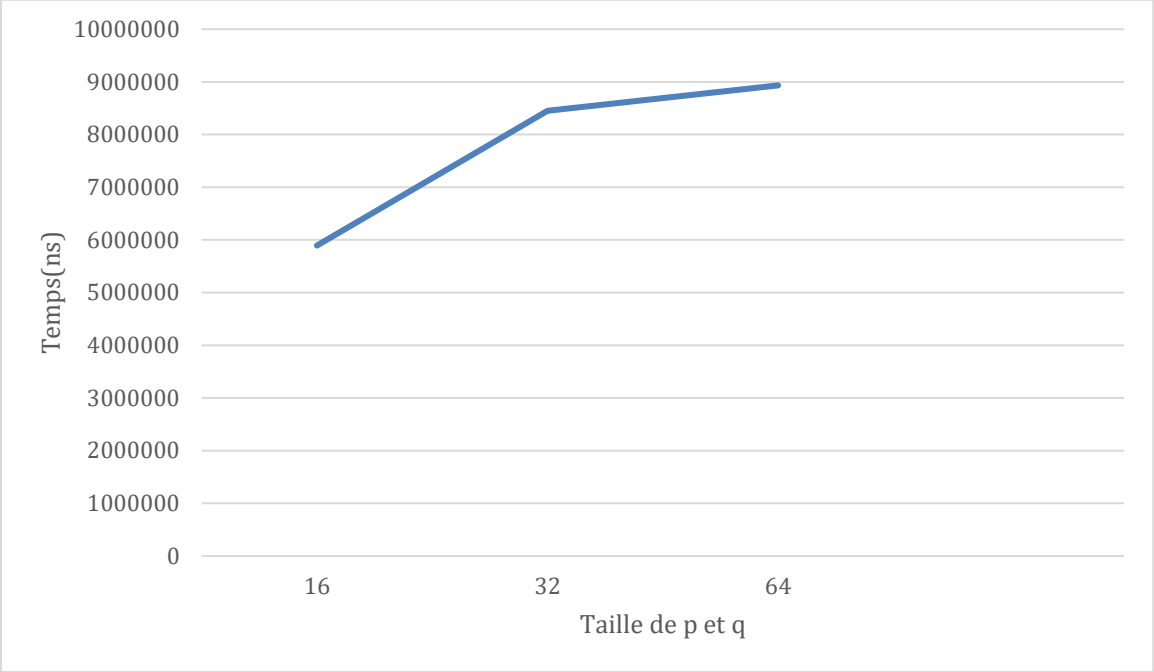

Figure 29. Graphe du temps calculé pour l'attaque de Wiener.

3.6.3. Discussion

D'après le graphe du temps calculé pour les deux attaques, par factorisation et Wiener, on remarque que dans les deux cas, plus la taille de p et de q qui sont les deux facteurs de N qui constitue le module de RSA est grande, plus le temps de calcul pour réaliser l'attaque (soit par factorisation ou Wiener) et retrouver le d est important.

3.7. Conclusion

Dans ce chapitre, nous avons présenté notre application qui nous permet de dérouler l'algorithme RSA, et de simuler deux attaques contre le RSA, l'attaque par factorisation et l'attaque de Wiener. Les résultats obtenus par les différents scénarios testés montrent que la sécurité du cryptosystème RSA est liée à la taille de la clé, plus la taille de la clé est grande, plus l'attaque est difficile à mener et plus le risque diminue.

Conclusion générale

Les cryptographes n'ont cessé de redoubler d'ingéniosité, faisant se succéder des dizaines de systèmes de chiffrement plus recherchés les uns que les autres, mais chaque système a ses limites et ne doit pas être un substitut aux autres mesures de sécurité.

La cryptographie profite de l'évolution de la technologie, mais elle est en même temps victime de cette évolution, car les intrus peuvent utiliser ces nouvelles technologies dans la cryptanalyse, en mettant en œuvres des attaques contre les différents algorithmes de chiffrement, citons parmi ces algorithmes, le RSA qui est l'algorithme le plus populaire de nos jours et sur lequel nous avons mené notre étude, plus exactement sur sa cryptanalyse.

Tout au long de la préparation de notre projet de fin d'études, nous avons essayé de mettre en pratique les connaissances acquises durant nos études universitaires et cela dans le but de réaliser une application qui nous permet de simuler deux attaques contre le RSA qui sont l'attaque par factorisation et l'attaque de Wiener.

Nous avons observé, après plusieurs tests, que les deux attaques dépendent de la taille de la clé, plus la clé est grande plus le temps nécessaire pour casser l'algorithme est important.

De nos jours, avec la perpétuelle augmentation des domaines d'utilisation des systèmes embarqués et avec la multiplication des objets connectés, un chiffrement des échanges est nécessaire ainsi qu'une accélération des performances et bien sur une protection contre les attaques et pour satisfaire ces objectifs les systèmes embarqués s'intéressent aux courbes elliptiques qui sont principalement utilisées dans les environnements à faibles ressources car elles ne nécessitent pas de longues clés pour assurer un haut niveau de sécurité contrairement aux autres algorithmes de chiffrement à clé publiques tel que le RSA qui utilise des clés trop longues.

Annexe

Annexe

1. Génération des clés

1.1. Génération de clés pour l'attaque par factorisation

Afin de générer des clés pour une attaque par factorisation il faudra :

a. Générer en premier lieu un nombre aléatoire « p » puis vérifier que ce nombre est premier, si « p » n'est pas premier soustraire « 1 » puis vérifier à nouveau, répéter l'opération jusqu'à obtention d'un nombre premier.
b. Générer en second lieu un nombre aléatoire « q » puis vérifier que ce nombre est premier, si « q » n'est pas premier soustraire « 1 » puis vérifier à nouveau, répéter l'opération jusqu'à obtention d'un nombre premier.
c. Calculer $\varphi(N)$

$$\varphi(N) = (p-1)(q-1)$$

d. Calculer N

$$N = p \times q$$

e. Attribuer à la clé publique un exposant de chiffrement e qui est un nombre premier de Fermat.
f. Calculer la clé privée

$$d = e^{-1} mod\ \varphi(N)$$

```
//region Génération de clés pour l'attaque par factorisation
public void genererCles(int n){
    Random r = new Random();
    BigInteger p = new BigInteger(n, r); // Générer un nombre aléatoire "P"
    while (!p.isProbablePrime( certainty: 1)) { // Vérifier ce nombre s'il est premier ou pas
        p = p.subtract(new BigInteger( val: "1")); // Tant que le nombre n'est pas premier soustraire "1"
                                                   // et vérifier une nouvelle fois cette condition
    }

    Random rr = new Random();
    BigInteger q = new BigInteger(n, rr); // Générer un nombre aléatoire "Q"
    while (!q.isProbablePrime( certainty: 1)) { // Vérifier ce nombre s'il est premier ou pas
        q = q.subtract(new BigInteger( val: "1")); // Tant que le nombre n'est pas premier soustraire "1"
                                                   // et vérifier une nouvelle fois cette condition
    }

    BigInteger temp = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE)); //Calculer Phi = (p-1)(q-1)

    this.N = p.multiply(q); // Calculer N = p * q
    this.E = BigInteger.valueOf(65537); // Attribuer à la clé publique un nombre premier de Fermat 3,5,17,257,65537
    this.D = E.modInverse(temp); // Calculer la clé privée d = e^-1 mod Phi
}
//endregion Génération de clés pour l'attaque par factorisation
```

Figure 30. Code génération de clés pour l'attaque par factorisation.

1.2. Génération de clés pour l'attaque de Wiener

Pour générer des clés pour une attaque de Wiener il faudra utiliser l'algorithme d'Euclide étendu ainsi que le théorème suivant :

Théorème: Soit $N = p \times q$ un module RSA où les nombres premier p et q sont de même taille $(q < p < 2q)$ **.** Puisque $N = p.q > q^2$, Alors $q < \sqrt{N}$
D'autre part on a

$$N - \varphi(N) = p + q - 1 < 2q + q - 1 < 3q < 3\sqrt{N}$$

Si e est une clé publique et d la clé privée, alors $d \equiv e^{-1}(mod\ \varphi(N))$, où $\varphi(N)$ est l'indicateur d'Euler.
Donc il existe un entier k tel que $e.d - k.\varphi(N) = 1$. Puisque

$$k = \frac{e.d - 1}{\varphi(N)} < \frac{ed}{\varphi(N)} < d$$

Alors

$$\left|\frac{e}{N} - \frac{k}{d}\right| = \left|\frac{ed - kN}{Nd}\right| = \frac{|ed - k\varphi(N) - kN + k\varphi(N)|}{Nd}$$

$$= \frac{|1 - k(N - \varphi(N))|}{Nd} < \frac{3k\sqrt{N}}{Nd} < \frac{3k}{d\sqrt{N}}$$

Puisque $k < d < \frac{1}{3}N^{\frac{1}{4}}$ alors

$$\frac{3k}{d\sqrt{N}} < \frac{N^{\frac{1}{4}}}{d\sqrt{N}} = \frac{1}{dN^{\frac{1}{4}}} < \frac{1}{2d^2}$$

Ainsi, en appliquant le théorème de convergence $\frac{k}{d}$ est une convergente de $\frac{e}{N}$. Connaissant d et k, on peut alors calculer $\varphi(N)$ par la relation

$$\varphi(N) = \frac{ed - 1}{k}$$

Ainsi, on peut calculer p et q et donc trouver la factorisation de N.

Algorithme d'Euclide étendu

$a_0 \leftarrow a$
$b_0 \leftarrow b$
$t_0 \leftarrow 0$
$t \leftarrow 1$
$s_0 \leftarrow 1$
$s \leftarrow 0$
$q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor$
$r \leftarrow a_0 - qb_0$
while $r > 0$
do $\begin{cases} temp \leftarrow t_0 - qt \\ t_0 \leftarrow t \\ t \leftarrow temp \\ temp \leftarrow s_0 - qs \\ s_0 \leftarrow s \\ s \leftarrow temp \\ a_0 \leftarrow b_0 \\ b_0 \leftarrow r \\ q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor \\ r \leftarrow a_0 - qb_0 \end{cases}$
$r \leftarrow b_0$
return (r, s, t)
comment: $r = \gcd(a, b)$ and $sa + tb = r$

Algorithme 8. Algorithme d'Euclide étendu.

Pour générer des clés pour une attaque de Wiener il faudra :

a. Générer en premier lieu un nombre aléatoire « p » puis vérifier que ce nombre est premier, si « p » n'est pas premier, alors générer un autre nombre, répéter jusqu'à trouver p premier.

b. Générer en second lieu un nombre aléatoire « q » puis vérifier que ce nombre est premier, si « q » n'est pas premier, alors générer un autre nombre, répéter jusqu'à trouver q premier.

c. Calculer N

$$N = p \times q$$

d. Calculer $\varphi(N)$

$$\varphi(N) = (p-1)(q-1)$$

e. Calculer e à partir de $d < \frac{1}{3}N^{\frac{1}{4}}$ et l'algorithme d'Euclide étendu.

f. Vérifier si $q > p$ puis vérifier si $p > 2q$

g. Calculer la valeur de d à partir de e et $\varphi(N)$.

```
//region Génération de clés pour l'attaque Wiener
public void NPQ(int n) throws ParseException {
    boolean A, B;
    BigInteger p, q, N, Phi;
    List<BigInteger> E;
    boolean isEmpty;

    do {
        SecureRandom ran1 = new SecureRandom();

        do {
            p = new BigInteger(n, ran1); // Générer un nombre aléatoire "P"
        } while (!p.isProbablePrime( certainty: 1)); // Tant que le nombre n'est pas premier générer de nouveau un autre

        SecureRandom ran2 = new SecureRandom();

        do {
            q = new BigInteger(n, ran2); // Générer un nombre aléatoire "Q"
        } while (!q.isProbablePrime( certainty: 1)); // Tant que le nombre n'est pas premier générer de nouveau un autre



        N = p.multiply(q); // Calculer N = p * q
        Phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE)); // Calculer Phi = (p-1)(q-1)
        E = Gen(Phi, N); // Calculer E à partir de : 3*d < N^1/4 et l'algorithme d'Euclide étendu
        isEmpty = E.isEmpty(); // Vérifier la présence d'une valeur de E
        A = q.compareTo(p) > 0; // Vérifier si Q > P
        B = p.compareTo(q.multiply(BigInteger.TWO)) > 0; // Vérifier si P > 2*Q
    } while (A || B || isEmpty); // Boucle vérifiant les conditions que Q < P, P < 2*Q, E est présent

    setN(N); // Attribuer à N sa valeur
    setE(E.get(1)); // Attribuer à E sa valeur à partir de la liste
    setD(this.getE().modInverse(Phi)); // // Attribuer à D sa valeur en le calculant à partir de E et Phi
}

Calculer E à partir de : 3*d < N^1/4 et l'algorithme d'Euclide étendu

Algorithme d'Euclide étendu

//endregion Génération de clés pour l'attaque Wiener
```

Figure 31. Code de génération de clés pour l'attaque de Wiener.

Bibliographie

[1] Introduction à la sécurité informatique https://www.commentcamarche.net

[2] https://www.openhost-network.com

[3] Raphael Yende. SUPPORT DE COURS DE SÉCURITÉ INFORMATIQUE ET CRYPTO. Master. Congo- Kinshasa. 2018 https://hal.archives-ouvertes.fr

[4] Renaud Dumont, Notes de cours provisoires, 2009 - 2010 https://www.cours-gratuit.com/cours-divers/support-de-cours-de-cryptographie-et-securite-informatique

[5] Mémoire de fin d'études en informatique, Université ABDELHAMID IBN BADIS Mostaganem, 2012/ 2013. Lotmani Zakaria Ismail et Elhomr Youcef, « Simulation d'une attaque sur le cryptosysteme RSA ».

[6] Arnaud jaques, Qu'est-ce que la stéganographie ?,2018. https://www.securiteinfo.com/attaques/divers/steganographie.shtml

[7] Jean-françois pillou, Cryptographie, Mai 2015 https://www.commentcamarche.net/contents/203-cryptographie

[8] Bastien L, Sécurité, 4 avril 2019 https://www.lebigdata.fr/chiffrement-des-donnees-tout-savoir

[9] Culture Informatique, Comment ça marche le cryptage ?, 18 mars 2016 https://www.culture-informatique.net/comment-ca-marche-cryptage

[10] Preuve de sécurité https://fr.wikipedia.org/wiki/Preuve_de_sérurité

[11] Explication, Décryptage https://chiffrer.info

[12] Travail de Bachelor, Haute École de Gestion de Genève (HEG-GE) Filière Informatique de Gestion, 5 juin 2015, Daniel LAMAS, « La cryptographie ». https://core.ac.uk

[13] Mémoire de fin d'études en recherche opérationnelle, Université HOUARI BOUMEDIENNE, 2007, Louiza REZALLAH, « De la cryptographie classique a la cryptographie moderne théorie et application ». http://repository.usthb.dz

[14] D'après un cours de Daniel Barsky & Ghislain Dartois, Cryptographie, Paris 13 le 1 octobre 2010 https://www.math.univ-paris13.fr

[15] Généralité sur la cryptographie http://dspace.univ-tlemcen.dz/bitstream/112/6836/1/Etude-comparative-entre-la-cryptographie.pdf

[16] Data Encryption Standard https://www.geeksforgeeks.org

[17] Houda FERRADI, Chiffrement par bloc (AES), Université Paris 13 Villetaneuse 01/02/2016 https://www.di.ens.fr/~ferradi/coursAES.pdf

[18] Chiffrement RSA https://www.lemagit.fr

[19] Thèse de doctorat de l'université Pierre et Marie Curie Spécialité Informatique, 2012, Stéphane Jacob « Protection cryptographique des bases de données : conception et cryptanalyse » https://tel.archives-ouvertes.fr

[20] Renaud Dumont, Cryptographie et Sécurité informatique, Université de Liège Faculté des Sciences Appliquées, 2009 – 2010.

[21] S.Bellataf, Cours de sécurité informatique,UMMTO, 2017/2018.

[22] Jacques Stern, Louis Granboulan. "Conception et preuves d'algorithmes cryptographiques ",2004. https://www.di.ens.fr/~granboul/enseignement/crypto/CoursCrypto.pdf

[23] Renaud Berthe, Anthony Monier. "RSA", 2010. https://repo.zenk-security.com

[24] Nicholas G. McDonald. "A Research Review : PAST, PRESENT, AND FUTURE METHODS OF CRYPTOGRAPHY AND DATA ENCRYPTION". https://interstices.info/nombres-premiers-et-cryptologie-lalgorithme-rsa

[25] Jean-Sébastien Coron. "Cryptanalyses et preuves de sécurité de schémas à clé publique",2001.

[26] Mémoire de fin d'étude de l'université MOULOUD MAMMERI de Tizi-Ouzou,juillet 2018,Mohand-Amokrane BIR, Lyes DAHMOUNI « Etude et implémentation d'algorithmes de chiffrement à clé secrète et à clé publique : Application au cryptage de laparole. »

[27] Math.univ-lille1.fr/~bodin/_chier/ch-crypto.pdf.

[28] Cours de cryptographie MM067 - 2009/10, Méthodes de factorisation, Université Pierre et Marie Curie, auteur : Alain Kraus https://www.math.univ-paris13.fr/~boyer/enseignement/crypto/Chap6.pdf

[29] Cours de cryptographie, Université Aix-Marseille, A. Dragut http://pageperso.lif.univ-mrs.fr/~andreea.dragut/enseignementCLAA/CryptoChap3RSA2012.pdf

[30] Thèse de doctorat, Université Mouloud MAMMERI Tizi-Ouzou, 2019/ 2020. Rebiha HADAOUI, «le chiffrement RSA dans les systèmes à ressources limités».

[31] Techniques de cryptanalyse de RSA, Christophe Grenier, 2009 ftp://ftp.irisa.fr/local/caps/DEPOTS/BIBLIO2009/Grenier_Christophe.pdf

[32] http://www.goubin.fr/papers/strategies.pdf

[33] J. Hastad. "Solving simultaneous modular equations of low degree", 1988.

[34] Mémoire de fin d'études en Informatique, UNIVERSITE LARBI BEN M'HIDI OUM EL BOUAGHI, juin 2016, MIROUD Amina et NOUADRI Mouna « Cryptanalyse de RSA : Etude comparative de deux approches ».

[35] Christophe Clavier. De la sécurité physique des crypto-systèmes embarqués. Cryptographie et sécurité. Université de Versailles Saint-Quentin-en-Yvelines, 2007 https://hal.archives-ouvertes.fr/tel02487098/document.

[36] Mémoire de fin d'études en informatique, Université ABDELHAMID IBN BADIS Mostaganem, 2012/ 2013. Lotmani Zakaria Ismail et Elhomr Youcef, « Simulation d'une attaque sur le cryptosysteme RSA ».

[37] Hervé Lehning, cryptographie & codes secrets, Pole, Paris, 2013.

[38] Algorithme d'Euclide, http://www.bibmath.net/ .

[39] http ://defeo.lu/in310/poly/euclide-bezout

[40] http ://www.bibmath.net/crypto/index.php.