MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE
LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABDELHAMID IBN BADIS MOSTAGANEM

Faculté des Sciences & Technologies
Département d'Informatique

MEMOIRE DE FIN D'ETUDES
Pour l'Obtention du Diplôme de Master en Informatique
Option : Ingénierie des systèmes d'information

Thème

SIMULATION D'UNE ATTAQUE SUR LE CRYPTOSYSTEME RSA

Présenté par :

Lotmani Zakaria Ismail
Elhomr Youcef

Encadré par:

Chahinez Mérièm Bentaouza

Année Universitaire 2012/ 2013

Résumé

Ce projet est la réalisation d'une application en simulant une attaque sur le crypto-système RSA en utilisant la factorisation. La cryptographie consiste à protéger des messages en assurant la confidentialité, l'intégrité et l'authentification en utilisant une clé. Cette clé est utilisée pour dissimuler les messages aux yeux de certains utilisateurs mal intentionnés. Ainsi le RSA, un très bon algorithme de cryptographie asymétrique qui utilise deux clés, une clé publique pour le chiffrement et une clé privée pour le déchiffrement mais il est victime de beaucoup d'attaques pour le casser.

Abstract

This project is the realization of an application by simulating an attack on the RSA cryptosystem using the factorization. The cryptography is used to protect messages to ensure confidentiality, integrity and authentication using a key. This key is used to conceal messages to some malicious users. So the RSA considered as a good algorithm asymmetric cryptography uses two keys, a public key for encryption and a private key for decryption, but he suffered several attacks in order to break the algorithm.

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 et le chiffrement demeurent une partie indispensable de la sécurité informatique moderne. Néanmoins, tout utilisateur non averti peut ne pas comprendre l'utilité du chiffrement et risque de s'en passer. L'objectif principal de document est d'introduire différentes notions et quelques exemples de techniques cryptographiques généralement utilisées ainsi que leur cryptanalyses.

Etant une science de coder des messages secrets, la cryptographie préserve le texte en clair de l'indiscrétion des criminels informatiques, fournissant non seulement la confidentialité des données mais aussi l'authentification, l'intégrité, le non désaveu et le contrôle d'accès.

Ce document est divisé en quatre chapitres :

Chapitre 1, présente les notions de cryptographie (définition, domaine d'utilisation…), ainsi que la cryptographie au cours de l'histoire et la cryptographie moderne en citant les deux catégories symétrique et asymétrique.

Le chapitre 2, présente l'algorithme à clef publique RSA, la description du protocole du chiffrement, le déroulement de l'algorithme ainsi que la force de l'algorithme.

Le chapitre 3, parle des attaques et cryptanalyses réalisées sur les différents systèmes cryptographiques et leurs classifications, le chapitre expose les attaques réalisées sur l'algorithme RSA avec leurs différents types physiques et algorithmiques.

Le chapitre 4, explique et expose l'application réalisée en montrant ses différentes fonctionnalités (génération des clefs, transferts des messages), et les scénarios possibles (attaque sur la clef privée, changement de signature, usurpation d'identité).

Enfin nous terminerons ces chapitres par une conclusion.

SOMMAIRE

Chapitre 4

Conception et réalisation

Table des figures

Liste des abréviations

MD: Message Digest

DES:Data Encryption Standard

AES:Advanced Encryption Standard

MIT:Massachusetts Institute of Technology

NSA:National Security Agency

RSA:Rivest, Shamir and Adleman

PGP: Pretty Good Privacy

DRM: Digital Right Management

SHA: Secure Hach Algorithm

IDEA: International Data EncryptionAlgorithm

Chapitre 1

LA CRYPTOGRAPHIE

1. Introduction

Le développement considérable des réseaux de communication a favorisé l'expansion de nouveaux moyens de paiement à distance avec notamment la carte à puce, le e-commerce, et l'e-Banking. Ces nouveaux types de communications nécessitent de plus en plus de moyens de transaction dits sûrs, pouvant résister aux diverses attaques cryptanalystiques. De cette lutte effrénée entre d'une part les cryptographes qui mettent en place des systèmes ou des algorithmes cryptographiques, et d'autre part les cryptanalystes qui développent des techniques pour briser les systèmes de sécurité.

2.Cryptographie

La cryptographie a été inventée, à ses débuts, dans le but de sécuriser des communications, jugées secrètes par l'expéditeur. 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 le secret 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. Elle concerne la transformation d'un message (texte, image, chiffres) intelligible vers un message codé, incompréhensible à tous sauf pour les détenteurs d'un code de déchiffrement. Il ne faudra d'ailleurs pas confondre avec **la sténographie**, qui consiste à dissimuler un message sans transformation « logique ».

Particulièrement la sténographie, voici quelques exemples de procédés sténographiques utilisés

- **L'encre invisible :**
  Au **1er**siècle av. **J-C**, apparu une méthode de dissimulation de message consistant à écrire avec du lait ou du jus de citron. Il suffisait de chauffer le support pour faire réapparaître le message.
- **Le crâne rasé :**
  A l'antiquité, les Grecs utilisaient parfois la tête des esclaves pour tatouer des messages. Ils étaient ensuite envoyés chez le destinataire et après un long voyage, il fallait raser leur tête dont les cheveux avaient repoussés, pour lire le message.
- **Le micro point :**
  Pendant la seconde guerre mondiale, les Allemands utilisaient la technique du micro-point, consistant à dissimuler des photos ou des textes dans un point de ponctuation d'une lettre. Il suffisait alors d'agrandir le point pour voir apparaître le message.

L'opération de chiffrement transforme un texte en clair en un texte chiffré, appelé cryptogramme, au moyen d'une clé (qu'on dénomme *la clé de chiffrement*). Le déchiffrement est le traitement des données qui retransforme le texte chiffré en texte en clair.

- **La cryptographie** est la science qui consiste à créer de tels systèmes de chiffrement.
- **La cryptanalyse** est la science complémentaire qui consiste à déterminer certaines propriétés de ces systèmes dans le but de reconstituer le texte en clair, souvent en l'absence des paramètres qui sont nécessaires pour le déchiffrement.
- **La cryptologie** englobe la cryptographie et la cryptanalyse.
- **Le chiffrement** est généralement accompli avec un ***algorithme*** bien défini et une ***clé***. Ceci s'applique aussi au déchiffrement bien que les algorithmes ne soient pas nécessairement les mêmes pour ces deux étapes.

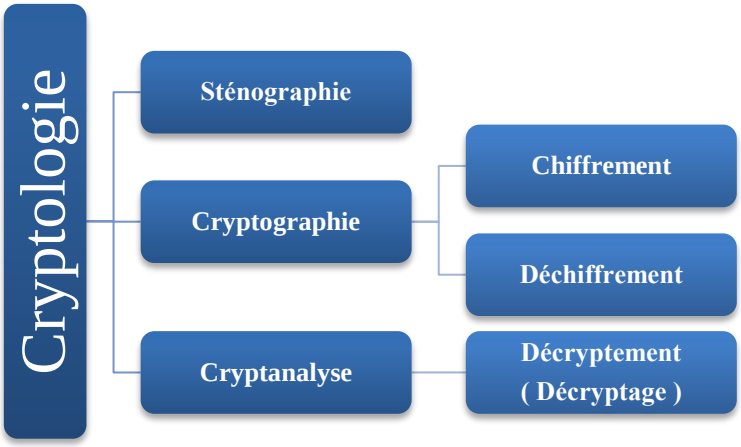

Figure 1.Organigramme de Cryptologie

3. Quelques repères historiques

Tout au long de l'histoire, la cryptographie s'est insérée dans les différents contextes historiques, et a progressé avec les évolutions mathématiques. La plupart du temps elle est utilisée dans un milieu politique ou militaire **[1].**

Historiquement le développement de cryptographie est passé par trois âges :

3.1 Âge Artisanal (systèmes manuels)

Jusqu'à au **XX^{ème}** siècle (fin de 1^{èr} guerre mondiale), on entend habituellement les méthodes cryptographiques qui requièrent simplement du papier et un crayon. Il n'est pas surprenant que ces systèmes soient les plus anciens.

Les 1ères traces de cryptographie remontent au **XVI^{ème}** siècle avant **J-C**.On a retrouvé à cette époque sur les bords du **Tigre** en **Irak** le plus vieux document chiffré connu. Dans l'antiquité on remarque alors

l'émergence de nouveaux modes de cryptographie, pour la plupart propres à un peuple. C'est pourquoi nous allons ici organiser l'étude avec ces différentes civilisations.

a. Hébreux (chiffre Atbesh)

La plus connue et la plus simple est sans doute le chiffre **Atbesh**. Datant du **V**ème siècle avant **J-C**, ce procédé consiste à faire correspondre l'alphabet classique avec l'alphabet inversé, et d'associer ainsi à chaque lettre la lettre correspondante en position dans l'alphabet inversé. Voici un exemple pour l'alphabet latin

b. Grecs(Scytale)

A la même époque que les hébreux, les grecs commencèrent à développer quelques méthodes de codage. Parmi celles-ci: la scytale (ou scytale spartiate).

La scytale est une des premières techniques de codage par transposition. Elle date du **V**ème siècle av **J-C**. Le principe est le suivant: celui qui est l'auteur du message va préalablement enrouler une bande (de parchemin ou de cuir) en hélice autour d'un bâton régulier de diamètre fixé sur laquelle il va pouvoir écrire. Après écriture, l'auteur déroule la bande qui présente alors une succession de caractères n'ayant aucun sens logique. Celui qui veut lire le message doit alors connaître le diamètre du bâton initial puis y enrouler de nouveau la bande.

Figure 2. Scytalespatiate

c. Romains (le chiffre de César)

Jules César utilisait un mécanisme de confidentialité rudimentaire, où chaque lettre d'un message était remplacée par celle située trois positions plus loin dans l'alphabet. La méthode se généralise et prend le nom de substitution.

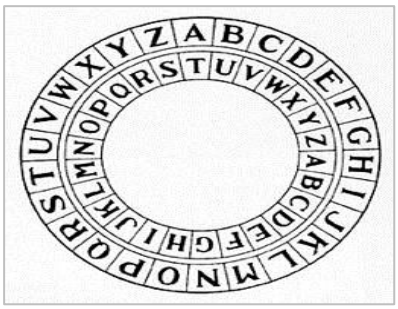

Figure 3. Roue de César

3.2 Âge technique (Systèmes mécaniques et électro-mécaniques)

Le **XX^{ème}** siècle (1919 - 1975), utilisant des appareils mécaniques ont donc été mises au point. L'âge technique garde les substitutions et les permutations mais les met en œuvre à l'aide de machines mécaniques ou électro-mécaniques. Les plus célèbres sont l'**Enigma** et la **Hebern**.

a. Machine Enigma

La machine **Enigma** reste l'une des plus remarquables machines à crypter toutes périodes confondues; elle a longtemps servi de modèle et a marqué son époque.

La naissance de la machine Enigma est initiée en 1919 par un ingénieur hollandais, *Hugo Alexander*, qui breveté un dispositif de codage électromécanique ; ses idées sont réutilisées par le *Dr Arthur Scherbius* qui donne le nom Enigma à sa machine initialement développée pour les civils.

Cependant, sa société fera faillite peu après, en revanche, les militaires allemands vont lui accorder un grand intérêt pendant la seconde guerre mondiale **[2].**

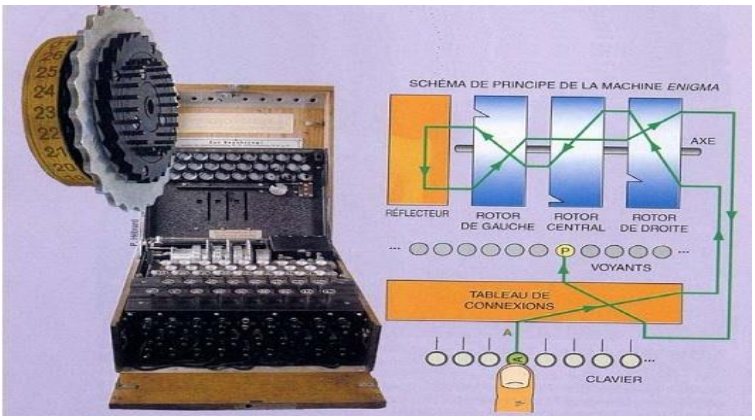

Figure 4. La machine Enigma

b. Machines de Hebern

En 1910, *Edward H. Hebern* commença à mettre au point des machines de chiffrement aux États-Unis. Au début des années 1920, il inventa la machine à code électrique qui utilisait seule roue codeuse. Il conçut aussi, en 1924, des appareils à trois et cinq roues codeuses comme prototypes pour la marine américaine. Très différentes de l'Enigma, les machines Hebern utilisaient des roues codeuses dont le filage interne pouvait facilement être changé. De plus, ces roues codeuses pouvaient fonctionner dans le sens de rotation conventionnel ou d le sens contraire. Par contre, ces machines possédaient toutes des faiblesses du point de cryptanalytique. Ces faiblesses sont décrites par *William Friedman* dans la référence **[3].**

3.3 Âge scientifique (Moderne)

A commencé 1976 dont on utilise les méthodes modernes, avec la prolifération des ordinateurs et l'essor de leur connectivité, plusieurs méthodes cryptographiques utilisent maintenant des logiciels plutôt que des machines électromécaniques.

Il voit l'introduction de mécanismes donnant des réponses positives à des questions a priori hors d'atteinte :

- *Comment assurer un service de confidentialité sans avoir au préalable établi une convention secrète commune ?*
- *Comment assurer un service d'authenticité basé sur la possession d'un secret sans révéler la moindre information sur le secret ?*

a. Système symétrique DES

Les principes des codes symétriques commerciaux modernes du type **DES** (**D**ata **E**ncrytion **S**tandard) ont été mis au point dans les années 1970 par **IBM** avec l'aide de la **NSA** (**N**ational **S**ecurity **A**gency), ce sont des hybrides de codes de substitutions et de codes de transpositions.

Ils restent très sûrs avec des clés assez courtes de 128 bits à 256 bits sureté est non prouvée. Leur cryptanalyse a fait des progrès (crypta linéaire et différentielle), ce qui permet de mieux cerner leur sureté **[4].**

b. Système asymétrique RSA

L'algorithme à clé publique le plus répandu est l'algorithme RSA (*Rivest-Shamir & Adleman*), publié en 1978. Il est fondé sur les propriétés des nombres premiers. Pour casser l'algorithme, il faut factoriser un grand nombre entier (fourni par la clé publique) **[5].**

L'algorithme **RSA** est malheureusement beaucoup plus coûteux à mettre en œuvre que le **DES**, et son usage est généralement utilisé pour le codage de données de taille limitée. C'est en particulier une excellente solution pour la diffusion des clés du DES.

4.Algorithme cryptographique

Un algorithme cryptographique, ou Chiffre, est une fonction mathématique utilisée dans le processus de chiffrement et de déchiffrement et qui fonctionne en combinaison avec une *clé* qui peut être un mot, un nombre, ou un texte chiffré différent si l'on utilise des clés différentes. La sécurité des données chiffrées est entièrement dépendante de deux choses : la force de l'algorithme cryptographique et le secret de la clé. Un algorithme cryptographique, plus toutes les clés possibles et tous les protocoles qui le font fonctionner constituent un crypto-système **[6]**.

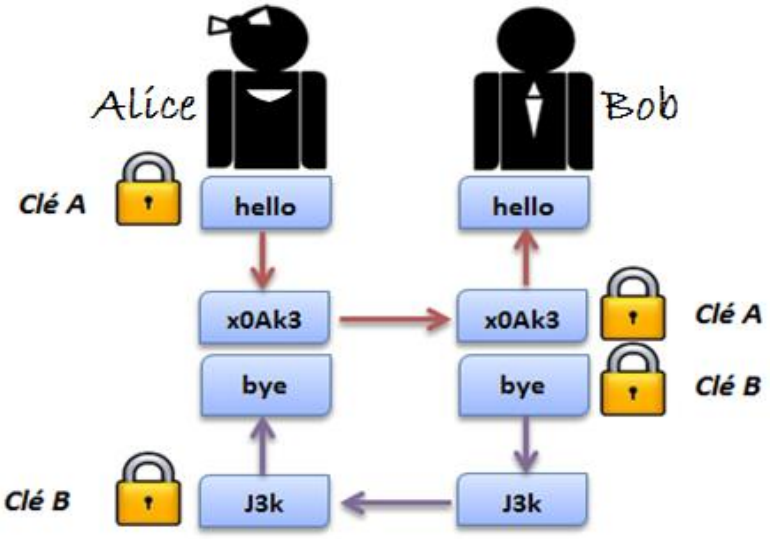

Figure 5. Exemple de la cryptographie de transfert de message

5. Objectifs fondamentaux et Qualités d'un crypto-système

De nos jours, les outils cryptographiques sont conçus pour remplir de plus en plus de fonctionnalités différentes. Parmi ces nombreuses fonctionnalités, il y en a quatre qui sont particulièrement importantes, les protocoles cryptographiques qui doivent être assuré sont résumés par les mots clés suivants **[4]:**

- **Intégrité des données:** le message ne peut pas être falsifié sans qu'on s'en aperçoive.
- **Identité des interlocuteurs du message:** l'émetteur est sûr de l'identité du destinataire c'est à dire que seul le destinataire pourra prendre connaissance du message car il est le seul à disposer de la clé de déchiffrement.
- **Authentification :** le receveur est sûr de l'identité de l'émetteur grâce à une signature.
- **Non-répudiations :**qui se décompose en trois :
  1. ***non-répudiation d'originel*** : l'émetteur ne peut nier avoir écrit le message.
  2. ***non-répudiation de réception :*** le receveur ne peut nier avoir reçu le message.
  3. ***non-répudiation de transmission*** l'émetteur du message ne peut nier avoir envoyé le message.

6.Systèmes de chiffrement

Il existe deux systèmes de chiffrement :

6.1. Systèmes de chiffrement à clé secrète, ou systèmes symétriques

Ils reposent sur le partage entre deux interlocuteurs en communication, d'une même clé **secrèteS** qui sert à paramétrer un algorithme à la fois pour le chiffrement d'un message et pour son déchiffrement. La clé **S** doit faire l'objet d'un échange physique préalablement à toute communication. Pour le stockage de messages, le principe est le même avec un seul interlocuteur. Cette clé prend en général la forme d'un ensemble de bits de taille limitée. Un procédé, connu sous le nom d' « *attaque par force brute* », utilisé pour retrouver le contenu des communications, consiste à essayer toutes les clés possibles. Leur nombre dépend de la taille de ces clés : pour une clé de n bits, il y a $\mathbf{2^n}$ clés possibles ; la complexité d'un produit est donc bornée par la taille de cet ensemble.

En général, la clé secrète commune **S** n'est pas utilisée directement pour chiffrer les messages, mais pour chiffrer une autre clé **K** qui est un nombre tiré au hasard par l'émetteur à chaque session et qui sert comme clé secrète pour chiffrer les messages. Cette clé **K** chiffrée est envoyée en début de session ou de message ou, dans le cas de stockage, conservée avec le message.

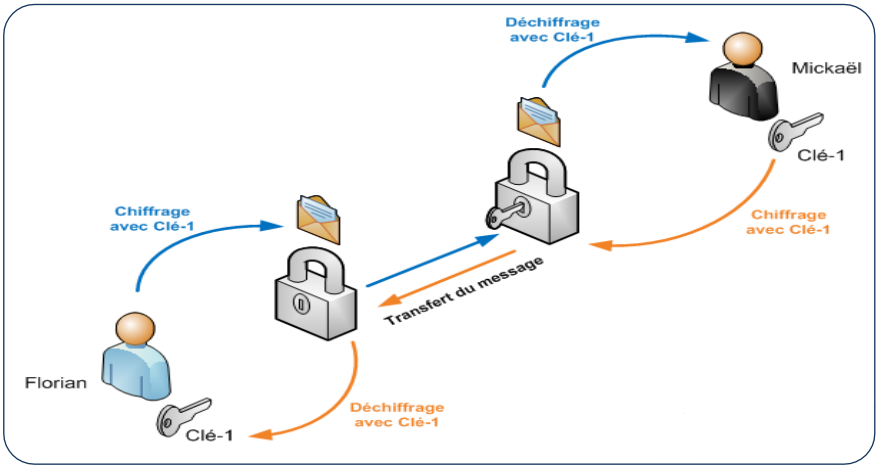

Figure 6. Schéma de cryptographie symétrique

6.2. Systèmes de chiffrement à clé publique ou systèmes asymétriques

Les algorithmes à clé publique servent à chiffrer des messages, mais aussi à calculer des signatures numériques. Une signature numérique est une valeur qui dépend du message, considéré alors sous sa forme numérisée comme un nombre, et de l'identité du signataire, qui doit être le seul à pouvoir calculer cette signature. Un message signé est composé du message en clair et de cette signature numérique. Vérifier une signature consiste, en appliquant la fonction inverse de la signature, à retrouver le message en clair.

Chaque utilisateur possède son propre couple de clés différentes **S** et **P** :

- La clé **S** est gardée secrète par son propriétaire qui l'utilise pour déchiffrer des messages reçus ou signer des messages.
- La clé **P** est rendue publique. Elle dépend de la clé **S** par une fonction à sens unique : la fonction est facilement calculable, mais son inversion est extrêmement difficile (on ne sait pas déduire **S** de **P**). Elle sert à quiconque pour chiffrer les messages destinés au propriétaire du couple de clés, ou à vérifier les signatures.

Pour **chiffrer** un message destiné à une personne *A*, le correspondant *B* applique la fonction définie par $\mathbf{P}_A$, la clé publique de *A*. A le déchiffrera avec sa clé secrète $\mathbf{S}_A$ qu'elle est seule à détenir.

Pour **signer** un message, *B* lui applique la fonction définie par sa clé secrète $\mathbf{S}_B$ pour calculer une signature. Pour vérifier cette signature, *A* lui applique la fonction inverse de la fonction de signature, définie par la clé publique $\mathbf{P}_B$ de **B**, ce qui lui permet de retrouver en clair le message initial. Seul B, qui détient SB, a pu calculer cette signature.

B envoie un message chiffré et/ou signé à A	
B chiffre avec $\mathbf{P_A}$	**A** déchiffre avec $\mathbf{S_A}$
B signe avec $\mathbf{S_B}$	**A** vérifie avec $\mathbf{P_B}$

Comme les algorithmes à clé publique sont lents à exécuter, on chiffre toujours les messages avec des algorithmes à clé symétrique, et on utilise le dispositif à clé asymétrique pour chiffrer la clé de session générée aléatoirement, comme dans le cas des systèmes à clé secrète **[7].**

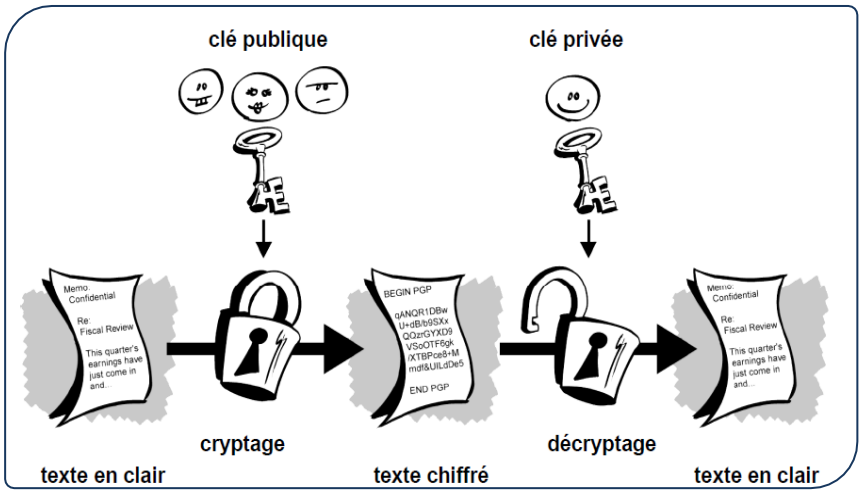

Figure 7. Schéma de cryptographie asymétrique

6.3. Systèmes de chiffrement hybride

Le concept de chiffrement hybride fait appel aux deux techniques, symétrique et asymétrique, comme présenté **(Figure 8).**La cryptographie asymétrique est très lente à cause des calculs complexes qui lui sont associés alors que l'un des principaux avantages de la cryptographie symétrique est sa rapidité. Cependant il réside une grosse faille dans cette dernière: la transmission de la clé secrète. C'est là qu'intervient la cryptographie asymétrique où sa paire de clé est utilisée pour sécuriser le transport de la clé secrète sur le réseau. Et comme cette clé est relativement petite, son chiffrement par un système asymétrique est beaucoup moins coûteux. Ces Caractéristique principales sont Utilise la rapidité de l'algorithme symétrique, et la sécurité de l'asymétrique, Et utilisé dans la plupart des outils actuels, parmi les algorithmes de ce chiffrement on a le **PGP** (**P**retty **G**ood **p**rivacy)**[8].**

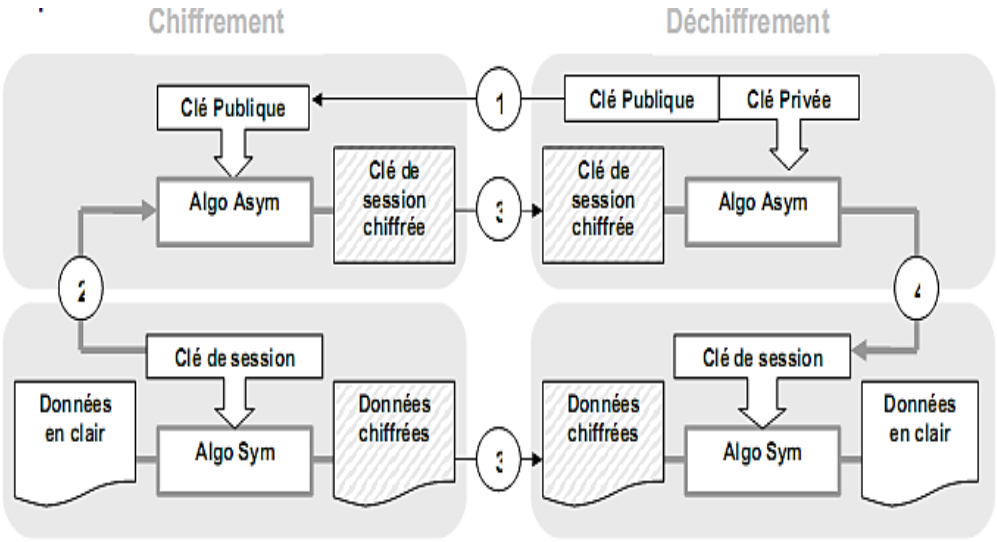

Figure 8. Systèmes de chiffrement hybride

7. Quel crypto-système choisir

Chacun des crypto-systèmes existant codes par **flots**, codes par **blocs** à clé secrète, codes à clés publiques ont leurs avantages et leurs inconvénients. Ils sont chacun des applications privilégiées et sont souvent utilisés en association.

a. Codes par flots basés sur les registres à décalage sont très utilisés en télécommunications (*télévision à péage*), à cause de leur rapidité. Ils permettent de faire du codage et du décodage à la volée en temps réel.

b. Codes par blocs à clé sécrète sont les plus employés car ils réalisent un excellent compromis entre rapidité et sécurité. Ils ne nécessitent que des clés assez courtes pour un bon niveau de sécurité (*128/256 bits*). Par contre ils nécessitent d'échanger de nombreuses cales secrètes avec un risque pour la sécurité. On peut les associer à un crypto-système à clé publique pour échanger les clés qui doivent être

changées très régulièrement pour garder une bonne sécurité et aussi pour réaliser les signatures et authentifications.

Les crypto-systèmes asymétriques sont plus lents. Ils nécessitent des clés longues (1*024/2048 bits*) sauf les codes elliptiques (*128/256 bits*). Ils ne nécessitent pas d'échange de clés secrètes et permettent de réaliser facilement des fonctionnalités très utiles (*signature, authentification, non répudiation,...*)**[8].**

8. Applications de la cryptographie

Jusqu'au 20ème siècle la cryptographie (l'égale) était essentiellement réservée aux militaires et aux diplomates et accessoirement aux industriels et banquiers.

Le développement des moyens de communications électromagnétiques facile à intercepter, des stockages de données confidentielles dans des sites assez faciles à pénétrer et sur des supports (bandes ou disques, magnétiques optiques) faciles à dupliquer ont généré une demande de crypto systèmes sûrs faciles d'emploi et rapides pour des usages civils et commerciaux.

Les codes symétriques ou à clés secrètes, type **DES** (*Data Encryption Standard*), **IDEA** (*International Digital Encryption Algorithm*), **AES** (*Advanced Encryption Standard*) ont été créés pour couvrir ces besoins. L'application première de la cryptographie est reste encore la transmission sécurisée de données sensibles.

Le commerce en ligne, la consultation des données bancaires sur internet, le paiement par carte bancaire ont entrainé une explosion de l'utilisation de la cryptographie mais ont aussi une extension de ses applications. Cette extension a été facilitée par la mise au point des procèdes de transferts de clés de *Diffie* et *Hellman* et des codes asymétriques ou à clés publiques de type **RSA** ou **El Gamal**.

Les codes à clé publique permettent de résoudre avec des protocoles légers les questions de transfert de clés, d'identification, d'authentification, de signature entre correspondants.

Aujourd'hui les plus gros utilisateurs de cryptosystèmes sont les institutions financières (banques, bourses,...), les télécommunications (téléphonie mobile, WIFI, Internet,...), les sites d'achats en ligne,…

Il est probable que le tatouage (watermarking) des données numériques ainsi que la gestion des droits numériques (**D**igital **R**ight **M**anagement ou **DRM**) vont entrainer une demande accrue de cryptographie. Le mariage de la cryptographie et de la théorie de la complexité aboutit à des extensions spectaculaires du champ de la cryptographie classique **[9].**

9. Conclusion

La cryptographie est une technique qui change progressivement en fonction des conditions politiques, économiques et des législations des pays (loi concernant la protection de la vie privée etc...). On peut aussi dire que la cryptographie n'évoluerait pas si personne n'essayait de casser les codes de cryptages, car il n'y aurait plus aucune raison de les améliorer étant donné que personne ne parviendrait à les casser. Les méthodes de cryptage seraient donc jugées sûres à 100% et elles le resteraient en attendant que quelqu'un réussisse à passer les mesures de sécurités**.**

Chapitre 2

RSA

1. Introduction

Le **RSA** est l'un des plus connus des algorithmes asymétriques. C'est à dire qu'on utilise une clé publique que tout le monde connaît qui permet de crypter le texte et une clé privée, dont seul le destinataire du message est censé la connaître afin de décrypter le message. Ce qui est important avec ce système, c'est qu'on ne peut pas retrouver la clé privée à partir de la clé publique.

Il a été inventé en **1978** par *Ronald L. Rivest, Adi Shamir* et *Leonard M. Adleman*, Des chercheurs au **MIT** (*Massachusetts Institute of Technology*).

Cette méthode est très utilisée de nos jours par exemple, dans les navigateurs Internet afin de garantir la sécurité de certains sites ou encore pour chiffrer des emails. Il est aussi le standard de chiffrement du secteur bancaire de plusieurs pays **[1]**.

Afin de crypter, nous avons besoin d'un message codé (il faut donc que le message d'origine soit transformé en nombres). Le RSA utilise l'arithmétique des congruences modulos **N**, c'est à dire qu'il utilise les restes des divisions. Il est alors possible, pour certains nombres, de trouver deux autres entiers qui nous permettent, par le calcul de puissance, de retomber sur le reste de départ (ainsi un premier calcul crypte et un second nous ramène sur ce chiffre de départ, donc décrypte). Un des principaux avantages du RSA, c'est qu'il crypte par blocs de caractères, donc il ne conserve pas la fréquence d'apparition des lettres.

2. Historique

L'algorithme **RSA** a été inventé en **1977** par *Ronald **R**ivest, Adi **S**hamir* et *Leonard **A**dleman*. C'est la première instance de méthodes de cryptographie à clé publique, dont le principe a été introduit à l'état de concept, par ***Whitfield Diffie*** et ***Martin Hellman***. A l'heure actuelle, il s'agit sans doute du crypto-système asymétrique le plus connu et le plus utilisé à travers le monde. Sa popularité en fait aussi un des algorithmes les plus étudiés, que ce soit d'un point de vue théorique ou pratique.

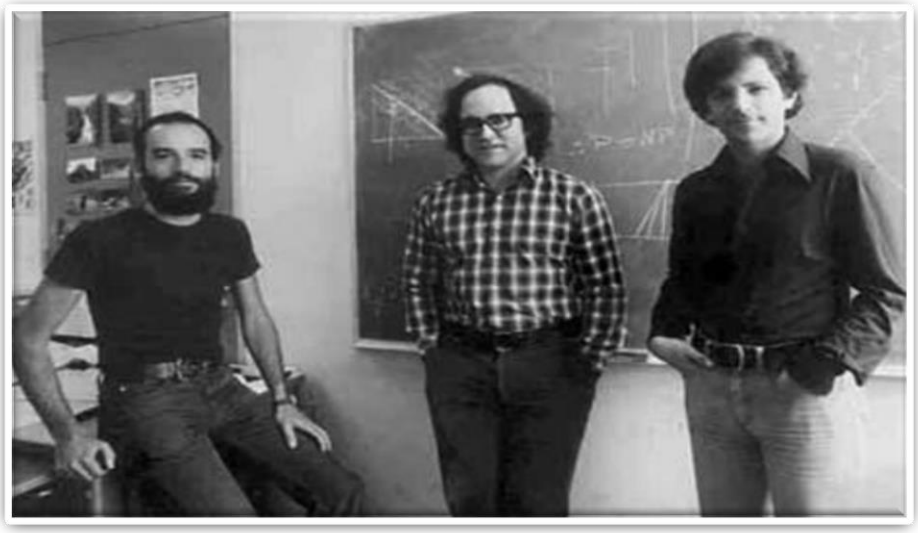

Figure 9. De gauche à droite : Adi Shamir, Ronald Rivest et Leonard Adleman

3. Crypto-système RSA

Le chiffrement symétrique existe depuis la création de la cryptographie. Il ne fut remis en question qu'en 1976 par *Diffie* et *Hellman*. Leur article **[2]** proposait de rompre la symétrie des rôles de l'expéditeur et du destinataire, en introduisant l'idée de chiffrement à clef publique (*appelé aussi chiffrement asymétrique*) selon laquelle seul le déchiffrement resterait confidentiel. Cela revient à rendre la clef de chiffrement publique (même pour l'adversaire), la clef de déchiffrement restant privée. Ces deux clefs sont en fait reliées mathématiquement (parce que la fonction de déchiffrement est l'inverse de la fonction de chiffrement), mais il doit être impossible en pratique de retrouver la clef privée à partir uniquement de la clef publique. La situation rappelle dans une certaine mesure l'exemple des cadenas que n'importe qui peut refermer une fois ouverts, mais que seul le propriétaire de la clef peut ouvrir. Les propriétés requises pour le cryptosystème sont plus fortes qu'en cryptographie symétrique, si bien que Diffie et Hellman ne purent illustrer par aucun exemple leur nouvelle théorie.

Suite à la publication de l'article **[2]**, trois chercheurs du **M.I.T**., nommés Rivest, Shamir et Adleman, 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 **[3]**, aujourd'hui connu sous le nom de RSA, et qui fut présente pour la première fois dans le numéro d'aout 1977 de *Scientific American*.

4. Définition de l'algorithme

L'algorithme **RSA** est utilisé pour la cryptographie 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'exemple le plus courant de cryptographie asymétrique, toujours considéré comme sûr, avec la technologie actuelle, pour des clés suffisamment grosses (***1024, 2048*** *voire* ***4096 bits***) **[4].**

4.1 Présentation de RSA

La sécurité du RSA repose sur la difficulté supposée du problème de la factorisation d'entiers. A ce jour, il n'existe pas des méthodes de résolution dont la complexité est meilleure que sous-exponentielle. Le plus grand module RSA factorisé par ce moyen, en utilisant une méthode de calculs distribués est long de 768 bits. Afin de rendre impossible la factorisation du module, on choisit en pratique des tailles de module au moins égales à **1024 bits**, voire à **2048 bits**.

En pratique, le RSA se décline en deux versions. La version classique, ou version ***RSA standard***, et la version ***CRT.*** Cette dernière version est quatre fois plus rapide que la version standard, c'est pourquoi elle est très utilisée pour implanter le RSA dans les systèmes embarqués **[5].**

4.2Fonctionnement

4.2.1 Chiffrement

On travaille principalement avec des nombres premiers (donc entiers naturels). A priori, il est assez facile de dire si un petit nombre est premier ou non. Cependant, dès que ce nombre devient plus conséquent, il est très rapidement difficile de dire s'il s'agit d'un premier ou non. Voici un atout principal du RSA **[1].**

- On choisit donc deux nombres premiers très grand **p** et **q** qui serviront à former les clés publiques et privées.
- On calcul **N**, qui est un constituant de la clé publique et de la clé privée en faisant : $\boldsymbol{N = p * q}$ **.**
- Ensuite on calcul **e**, qui fait partie de la clé publique, avec : **φ(N)= (p-1)*(q-1)** et l'exposant public **e** de telle sorte qu'il est un nombre premier avec **(p-1)*(q-1)**.C.à.d. Le **PGCD(e,φ(N))=1**.
- La couple **(N, e)**forme ainsi la clé publique de chiffrement.

On peut commencer le cryptage. On va se servir de la clé publique de chiffrement. Tout d'abord, il faut un message préalablement codé. On doit donc transformer en nombres le message d'origine (en utilisant la valeur **ASCII** de chaque lettre ou encore en remplaçant chaque lettre par son rang dans l'alphabet par exemple).

On a un message codé nommé$\boldsymbol{M}$. Il faut ensuite découper le message en blocs strictement inférieurs à **N**. On nomme alors chacun des blocs codés $\boldsymbol{M_i}$, le texte crypté $\boldsymbol{C}$ et chacun des blocs cryptés $\boldsymbol{C_i}$. Dans la réalité, les blocs sont très longs et les clés contiennent une centaine de chiffres (voir plus).

Afin de crypter, on fait $\boldsymbol{C_i = {M_i}^e \ mod \ N}$ .Autrement dit, $\boldsymbol{C_i}$ (étant le bloc de texte crypté), équivaut au reste de la division de $\boldsymbol{{M_i}^e}$par$\boldsymbol{N}$.

4.2.2 Déchiffrement

- Il s'agit désormais de calculer, à partir de **p** et **q**, la clé privée **d**. Pour cela, il faut satisfaire l'équation$\boldsymbol{d.e \equiv 1 \ mod \ \varphi}$(**N**).
- Avec la clé $\boldsymbol{d}$ générée, on peut décrypter bloc par bloc le texte crypté. Onretrouve alors le message$\boldsymbol{M}$.
- $\boldsymbol{M_i = {C_i}^d \ mod \ N}$. Autrement dit, $\boldsymbol{M_i}$ (étant le bloc de texte décrypté),et qui vaut au reste de la division de $\boldsymbol{{C_i}^d}$par$\boldsymbol{N}$. On retrouve alors nos blocs codéset il ne reste plus que les transférer vers leur équivalant dans l'alphabet défini **[6].**

4.2.3 Signature RSA

Comme pour l'opération de déchiffrement, la signature RSA **S** d'un message $\boldsymbol{M}$ consiste en une autre exponentiation modulaire avec l'exposant privé : $\boldsymbol{S = M^d \bmod N}$. La validité de cette signature est vérifiée en utilisant la clé publique du signataire :$\boldsymbol{M}$ =? $\boldsymbol{S^e \bmod N}$.

Dans la pratique, on préfère, signer une empreinte du message $\boldsymbol{M}$ plutôt que de signer directement sa valeur. Pour ce faire, on utilise une fonction de hachage $\boldsymbol{H}$ pour calculer l'empreinte du document à signer$\boldsymbol{\widetilde{M} = H(M)}$. Dans ce cas, la signature devient$\boldsymbol{S = M^d \bmod N}$.

De même, le calcul de l'empreinte est aussi nécessaire pour la vérification de signature **[5].**

4.3 Algorithme RSA

4.3.1 Description du protocole

Le but de l'algorithme est bien sûr de pouvoir transmettre un message codé, que seul le récepteur "officiel" puisse décrypter, c'est-à-dire qui ne puisse pas être décrypté par un tiers qui intercepterait ledit message. Nous appellerons **Bob le** destinataire du message, et **Alice** l'émettrice.

4.3.2 Déroulement de l'algorithme

La première étape est donc la création de la clé publique de Bob.

Etape 1 : Création de clé publique pour RSA

Output : Bob possède une clé publique ***(N, e)*** et une clé privée ***d***

Begin	
	Bob choisit deux grands nombres premiers ***p***et ***q*** Bob calcule ***N = p*q et φ(N) = (p − 1)(q − 1)*** Bob choisit un entier **e** premier avec ***φ(N)*** Bob calcule ***d*** l'inverse de e modulo ***φ(N)***
End	

Le nombre ***φ(N),*** appelé **indicateur d'Euler**, est le nombre d'entiers premiers avec ***N*** compris entre ***1***et ***N − 1***.**Bob** peut désormais proposer sa clé publique qui est la paire d'entiers ***(N, e)***. Sa clé privée est **d**. La sécurité du protocole RSA vient du fait que seul **Bob** connaît la factorisation de **N**. Il est le seul à pouvoir calculer ***φ(n)*** et par conséquent à pouvoir inverser ***e*** pour obtenir sa clé privée ***d***.

Désormais, toute personne ayant accès à la clé publique de **Bob** peut lui envoyer des messages sécurisés. A aucun moment, **Alice** n'est intervenue dans la création de la clé publique de **Bob**. Pourtant elle peut désormais lui envoyer un message sécurisé selon le protocole de RSA.

Etape 2 : Protocole RSA

Data: Alice connaît la clé publique de Bob ***(N, e)*** et Bob a calculé sa clé privée ***d***

Input : Alice possède un message ***M*** avec ***M<N***

Output : Bob possède ce message ***M***

Begin	
	Alice calcule $C = M^e \bmod N$ Alice envoie ***C*** à Bob calcule $M = C^b \bmod N$ Bob calcule $M = C^b \bmod N$
End	

5. Dans la Pratique

La norme **RSA** était de choisir un nombre **N** de *155chiffres*. En 1999, un nombre de 155 chiffres (qui servait justement de clé à un système RSA) a été décomposé en produit de deux premiers de 78 chiffres chacun. Le temps cumulé de calcul a été évalué à *8000 millions d'instructions* par *secondes* durant un *an*. Depuis, la Société ***RSA Data Security*** recommande des nombres de *309, voire 617 chiffres*. Les deux nombres premiers secrets **p** et **q** sont si grands (ils dépassent les *100 chiffres*) qu'il est quasi impossible de les retrouver connaissant le nombre public **N** et donc il est quasi impossible de retrouver la clé privée **d**.

Pour le chiffrement, on utilise généralement une combinaison de RSA et d'un algorithme symétrique. On chiffre tout d'abord les données à l'aide d'une clé symétrique aléatoire puis ensuite on chiffre cette clé à l'aide de la clé. Le destinataire déchiffrera cette clé symétrique avec sa clé RSA privée et ensuite il déchiffrera les données avec sa clé symétrique.

Si on souhaitait doubler la taille de la clé RSA, on multiplierait par 3 le temps des opérations utilisant la clé publique, par 8 le temps des opérations utilisant la clé privée et par 16 le temps de génération des clés. On voit encore une fois pourquoi il est préférable qu'on utilise le RSA avec un algorithme symétrique **[1].**

6. Limites du RSA

L'lorsqu'il s'agit de programmer un logiciel de cryptage / décryptage RSA, nous nous heurtons à un nouveau problème : le dépassement de capacité des variables. En effet, le plus grand type d'entier prévu dans le langage C est le type **unsigned long int**, qui ne peut gérer que des nombres de **32 bits** au maximum. Ceci est notoirement insuffisant pour un cryptage efficace. Notre programme se contentera donc de fonctionner pour des « petits nombres », c'est à dire pour des nombres **p**et **q** compris entre **0** et

255. Nous allons donc obtenir une clé **n** comprise entre **0** et **65535**. Mais pour les raisons évoquées ci-dessus, il ne sera pas possible de crypter par blocs de plusieurs caractères (il nous faudrait **n > 65536** pour pouvoir coder par blocs de **2** caractères). Notre code est donc d'une grande simplicité à casser, c'est pour cela que notre programme n'a qu'un but pédagogique et ne devra en aucun cas être utilisé pour un « usage commercial » **[6].**

7. Démonstration Mathématique

- On commence donc par définir **p** et **q** qui **[1]** :
  - Doivent être des nombres entiers naturels premiers.
  - Doivent être le plus grand possible et ils ne doivent pas être très proches l'un de l'autre par question de sécurité, de façon qu'ils ne soient pas facilement retrouvés à l'aide de **N**.
  - Doivent être connus que par le récepteur du message, puisqu'ils constituent la solution de génération des clés.
- On définit la clé publique $\boldsymbol{N}$ avec : $\boldsymbol{N = p * q}$ **.**
- On définit la clé publique $\boldsymbol{e}$ avec : $\boldsymbol{pgcd\big(e, (p-1)(q-1)\big) = 1.}$

  Il y a alors plusieurs possibilités pour le choix de **e** mais il doit bien évidemment être inférieur à $\boldsymbol{(p-1)(q-1)}$
- Le couple $(\boldsymbol{N}, \boldsymbol{e})$forme la clé publique de cryptage.

  Tout le monde peut la connaître afin de crypter un message, mais« personne » ne peut décrypter les messages sans la clé privée.
- On nomme$\boldsymbol{M}$, $\boldsymbol{M_i}$, $\boldsymbol{C}$, $\boldsymbol{C_i}$:
  - $\boldsymbol{M}$est le message d'origine déjà codé.
  - $\boldsymbol{M_i}$ est un bloc du message d'origine **M** (avec $\boldsymbol{M_i} < N$)
  - $\boldsymbol{C}$est le message crypté
  - $\boldsymbol{C_i}$ est un bloc du message crypté$\boldsymbol{C}$ (avec $\boldsymbol{C_i} < N$)
- Maintenant il faut trouver la clé privée **d**.

I. Preuve de l'existence de d et k

Le petit *théorème de Fermat* dit que si **p** est un nombre premier alors **1** est le reste de la division de $\boldsymbol{M_i^{p-1}}$par **p**, soit$\boldsymbol{M_i^{p-1} = 1\ mod\ p}$ (pour $\boldsymbol{1 < M_i < N}$).

Le fait que **e** et **d** soient des entiers naturels tel que :$\boldsymbol{e * d - (p-1)(q-1) = 1}et\boldsymbol{e}$ premier avec $\boldsymbol{(p-1)(q-1)}$ est prouvé avec *le Théorème de Bezout*.

Théorème de Bezout :

$\boldsymbol{x}$ et $\boldsymbol{y}$ sont premiers entre eux s'il existe $\boldsymbol{u}$ et $\boldsymbol{v}$dans $\mathcal{R}$ tel que : $\boldsymbol{xu + vy = 1}$. On donne : **x=e** et $\boldsymbol{y = (p-1)(q-1)}$
donc pour que $\boldsymbol{e * u + v(p-1)(q-1) = 1}$ il faut que $\boldsymbol{e}$ soit premier avec $\boldsymbol{(p-1)(q-1)}$

Et justement, c'est le cas.

Avec $\boldsymbol{e} * u + v(p-1)(q-1) = 1$ on a alors $\boldsymbol{u}$ et $\boldsymbol{v}$ dans$\mathcal{R}$, c'est-à-dire qu'ils ne sont pas forcément réels. Comment alors trouver **u** et **v** entiers réels de manière que $x = \boldsymbol{d}$et$\boldsymbol{v = k}$?

On peut avoir u comme entier réel mais alors v serait forcément négatif, et donc il ne pourrait pas valoir.

Notons tout de même ce que cela donnerait :

$$\boldsymbol{e} * \boldsymbol{d} + (-v) * ((p-1)(q-1)) = 1$$

Afin d'avoir v entier, la solution est évidente :

$$\boldsymbol{e} * \boldsymbol{d} - \boldsymbol{v}(p-1)(q-1) = 1$$

$\boldsymbol{v}$peut désormais valoir $\boldsymbol{k}$tel que $\boldsymbol{e} * \boldsymbol{d} - \boldsymbol{k}(p-1)(q-1) = 1$

Théorème de Fermat :

Ce théorème utilise la notion de factoriel et de combinaisons d'éléments, ce qui dépasse mon niveau de mathématique. C'est pourquoi je ne prouverai que brièvement ce théorème.

Brève preuve du Théorème de Fermat :

Puisque **e** est premier avec ***(p-1)*(q-1)*** il existe des entiers ***d*** et ***k*** tel que ***e*d - k(p–1)(q-1) = 1***.

On peut enfin résoudre **d** avec **: *e*d = k (p–1)(q–1) +1***ou encore ***e*d = 1mod (p–1)(q–1)***

Car : $\boldsymbol{k(p-1)(q-1) + 1 = 1 mod\ (p-1)(q-1)}$

Par exemple avec$\boldsymbol{p = 3, q = 5, (p-1)\ (q-1) = 8}$

Donc $\boldsymbol{k * 8 + 1 = 1\ mod\ 8}$

$$\boldsymbol{k * 8 + 1 = 2 * 8 + 1 = 17}$$

Ainsi : $\boldsymbol{17 - 16 = 1}$

Et$\boldsymbol{17 = 1\ mod\ 8}$

Donc$\boldsymbol{e} * \boldsymbol{d} - \boldsymbol{k}(p-1)(q-1) = 1$ $(avec\ \mathbf{0} < d < (p-1)(q-1))$.

Afin de trouver **d**, il faut connaître$(p-1)(q-1)$**.** Si on connaît **p** et **q**, il est facile de le calculer, or si quelqu'un voulait décoder un message, il lui sera difficile de trouver **p** et **q** seulement à l'aide de

n (Evidemment **p**, **q** et n doivent être très grands). Néanmoins il existe des algorithmes pour trouver **p** et **q** mais ils sont très lents surtout si **N** l'est aussi.

II. Preuve que si p et q sont premiers on a :

$$\boldsymbol{pour\ tout\ M_i < N :\ M_i^{k(p-1)(q-1)+1} = M_i mod\ N\ \ (modulo\ N, car\ N\ =\ p * q)}$$

Preuve :

$$\boldsymbol{M_i^{k(p-1)(q-1)+1} = M_i * M_i^{k(p-1)(q-1)} mod\ N}$$

$\boldsymbol{k * (p-1)(q-1)}$**mod N**…… *(Loi des exposants)*

Donc$\boldsymbol{M_i^{k(p-1)(q-1)} = 1\ mod\ N}$

Où $\boldsymbol{M_i^{ed} = M_i mod\ N}$car :$\boldsymbol{k * (p-1)(q-1) + 1 = e * d}$, Car d'après le *théorème de Fermat*$\boldsymbol{M_i^{p-1} = 1\ mod\ p}et\boldsymbol{M_i^{q-1} = 1\ mod\ q}$(Evidemment, avec **p<N**, et donc **q<N**…)

Donc$\boldsymbol{M_i^{ed} = M_i\ mod\ p}et\boldsymbol{M_i^{ed} = M_i\ mod\ q}$

Dans ce cas,$\boldsymbol{M_i^{(p-1)(q-1)} = 1\ mod\ N}$**, 1 modulo N** car $\boldsymbol{N = p*q}$**, et 1** car le pgcd de deux nombres premiers est forcément 1 .

Mi est le reste de la division de$\boldsymbol{M_i^{k(p-1)(q-1)+1}}$par **N**. $\boldsymbol{(ou\ p * q = N\ et\ k(p-1)(q-1) + 1 = e * d\ )}$**.**

On peut alors dire que$\boldsymbol{M_i^{k(p-1)(q-1)+1} = M_i^{ed}}$

On revient donc au point où :$\boldsymbol{M_i^{ed} = M_i mod\ N}$

Dans ce cas si$\boldsymbol{C_i = M_i^e\ mod\ N}$

Alors pour retrouver la valeur initiale de **Mi** on utilise :$\boldsymbol{M_i = C_i^d\ mod\ N}$

Donc le cryptage se fait alors avec $\boldsymbol{C_i = M_i^e\ mod\ N}$

Et le Décryptage avec $\boldsymbol{M_i = C_i^d\ mod\ N}$

---

8. Exemple Simple

- On prend un alphabet simple de quatre lettres :

A – B – C

- On code cet alphabet de façon que :

A = 1

B = 2

C = 3

- On choisit les deux nombres premiers **p** et **q** :

  p = 2

  q = 5

- On calcule la clé publique **n** avec :

  n = p*q = 2*5 = 10

- On définit la clé publique e de manière que **e** soit premier avec **(p–1)*(q–1)**.

  On a alors : **(p–1)*(q–1) = (2–1)*(5–1) = 1*4 = 4**

  Le **pgcd (e, 5) = 1** donc **e = 3**

- On calcul **d** avec **e*d – k((p–1)(q–1)) =1** par exemple :

  e*d – k((p–1)(q–1)) =1

  3*d – k(4) _1

  3*3 – 2*4 _1 car 9 = 3 * 3 + 1

  Donc la clé privée **d = 3**

  - Clé publique : **(N, e) = (10, 3)**
  - Clé privée :**(N, d) = (10, 3)**

  (Dans la pratique, la clé publique n'est jamais égale à la clé privée)

Cryptage du message : $C_i = M_i^e \bmod N$

Message : **BAC**

Message Codé : **3 2 4**

Crypter A : **M_i = A = 1**, donc **$C_i = 1^3 \bmod 10 = 1$**

Crypter B : **M_i = B = 2**, donc **$C_i = 2^3 \bmod 10 = 8$**

Crypter C : **M_i = C = 3**, donc **$C_i = 3^3 \bmod 10 = 7$**

Message Crypté : **8 1 7**

Décryptage du message : $M_i = C_i^d \bmod N$

Message Crypté : 8 1 7

Décrypter 8 : **C_i = 8**, donc **$M_i = 8^3 \bmod 10 = 2$**

Décrypter 1 : **C_i = 1**, donc **$M_i = 1^3 \bmod 10 = 1$**

Décrypter 7 : **C_i = 7**, donc **$M_i = 7^3 \bmod 10 = 3$**

Message Codé retrouvé : **2 1 3**

Message décodé : **B A C**

9. Complexité

Le chiffrement et le déchiffrement RSA nécessitent tous deux une exponentiation modulo **n**. Dans le cas général où les exposants **d** et **e** sont choisis aléatoirement, leur taille est donc celle de $\phi(n)$, soit celle de **n** puisque $\phi(n) = (p-1)*(q-1)$. Avec les algorithmes usuels d'exponentiation, le chiffrement et le déchiffrement coûtent donc tous deux $O(\log n)$ multiplications modulo **n**. Comme ces multiplications peuvent s'effectuer **[4]**en temps $O(\log^2 n)$, on obtient une complexité de chiffrement et de déchiffrement de $O(\log^3 n)$.

Cette complexité, bien que polynomiale, n'est pas négligeable, notamment pour des environnements à puissance de calcul réduite tels que la **carte à puce**. Aussi est-il tentant de choisir soit un petit **e**, soit un petit **d**, de façon à accélérer le chiffrement ou le déchiffrement. On évoquera plus tard les problèmes posés par ces choix particuliers d'exposants. Si **d** n'est pas petit, on peut légèrement accélérer le déchiffrement à l'aide des restes chinois*. En effet, pour calculer $m^d \bmod n$, il suffit de calculer $m^d \bmod p$ et $m^d \bmod q$, puis d'appliquer les **μ** restes chinois dont le coût est négligeable par rapport au calcul de $m^d \bmod n$ si **d** est grand. Or $m^d \equiv m^{d \bmod (p-1)} \bmod p$ *et* $m^d \equiv m^{d \bmod (q-1)} \bmod q$, de sorte que si l'on conserve les valeurs de **p** et **q**, on peut ramener le calcul de $m^d \bmod n$ à deux exponentielles d'exposant de taille divisée par deux, avec un module de taille également divisée par deux. Comme la complexité d'une exponentielle est de **O(log3n),** on divise ainsi approximativement le temps de calcul global par quatre : seule la constante de la complexité est améliorée**[7].**

> ***Notamment des restes de chinois :***
> *Le théorème des restes chinois n'énonce que l'application*
> $x \in \mathbb{Z}_n \rightarrow (x_i \bmod p_i^{ei}) i \in I$
> *est un isomorphisme. Les éléments de* $\mathbb{Z}_n$ *qui ont un inverse multiplicatif forment un groupe noté* $\mathbb{Z}_n^*$. *Ce groupe est isomorphe au produit :* $\prod_{i \in I} \mathbb{Z}\, p_i^{ei}$

10. Force de l'algorithme

Malgré une apparente simplicité, le système RSA reste l'un des plus sûrs. Jusqu'a très récemment. La plupart des gens s'accordaient sur l'idée que décoder un message sans connaitre la clef était équivalent à factoriser l'entier n (trouver p et q).

D'autre part, il est très difficile dans la pratique de factoriser **N** : même s'il existe des méthodes beaucoup plus efficaces que le procédé naturel (tester tous les nombres impaires jusqu'à$\sqrt{n}$**),** le temps de calcul nécessaire reste incomparablement plus long que celui dont ont besoin **Alice** et **Bob**.

L'algorithme étant assez sûr et assez lent. Dans la pratique, il sert le plus souvent à transmettre une clef servant à déchiffrer un message codé selon une autre méthode plus rapide, typiquement une méthode de chiffrement dite symétrique, par exemple **IDEA** : on code et on décode un message de la

même façon, il est donc nécessaire de transmettre le message codé et la clef pour le décoder. C'est ce que fait par exemple **PGP** (***P**retty **G**ood **P**rivacy*), logiciel de mise en œuvre de cryptographie publique (RSA) et symétrique (IDEA)**[8].**

11. Mise en œuvre pratique de RSA

Même si le protocole de RSA est assez simple, sa mise en œuvre pose toutefois quelques problèmes pour **Alice**, notamment la construction de deux « gros » nombres premier (**p** et **q**), ainsi que la détermination du couple ***(d,e)***, le problème confronté est celui d'élever de façon efficace un « grand » nombre à une grosse puissance, modulo **N**.

Donc les atouts principaux du **RSA** sont le fait que ce langage travaille avec les nombres premiers. Or factoriser un très grand nombre premier en deux autres prend beaucoup de temps. Trouver de grands nombres premiers est lent. Dans la plupart des cas on utilise des algorithmes **probabilistes de primalité** afin de déterminer **p** et **q**, comme le test de *Miller-Rabin*. Ces nombres ont donc une certaine probabilité de primalité.

Le fait que le **RSA** utilise une clé publique et une clé privée est un bon point étant donné que le problème de communications des clés est résolu (*contrairement aux algorithmes symétriques*).De plus, avec cette méthode, on crypte par bloc de caractères, ce qui empêche à un cryptanalyste d'utiliser l'analyse des fréquences afin d'essayer de casser les clés.

Le problème avec une telle méthode, c'est qu'il faut utiliser de grandes clés, et leur génération prend quand même un certain temps. De plus le cryptage également, puisqu'on travaille avec de très grandes clés, donc puissances (*c'est pourquoi on l'utilise avec un algorithme symétrique, comme cité plus haut*). Il existe des algorithmes pour casser les clés, mais ceux-ci sont très lents et donc la plupart des fois inefficaces dans des délais raisonnables **[8].**

12. Conclusion

Les atouts principaux du **RSA** sont le fait que ce langage travaille avec les nombres premiers. Or factoriser un très grand nombre premier en deux autres prend beaucoup de temps.

Le fait que le RSA utilise une clé publique et une clé privée est un bon point étant donné que le problème de communications des clés est résolu (contrairement aux algorithmes symétriques).

Le problème avec une telle méthode, c'est qu'il faut utiliser de grandes clés, et leur génération prend quand même un certain temps. De plus le cryptage également, puisqu'on travaille avec de très grandes clés, donc puissances (c'est pourquoi on l'utilise avec un algorithme symétrique, comme cité plus haut)
Il existe des algorithmes pour casser les clés, mais ceux-ci sont très lents et donc la plupart des fois inefficaces dans des délais raisonnables.

Chapitre 3

ATTAQUES SUR RSA

1. Introduction

Les composants de sécurité, parce qu'ils contiennent des informations confidentielles, font l'objet d'attaques. Celles-ci tentent généralement de porter atteinte à la confidentialité, à l'intégrité ou à l'authenticité des données protégées par les algorithmes de cryptographie embarqués sur ces composants. Les attaques dites **matérielles** ou physiques exploitent les faiblesses de l'implantation matérielle de ces algorithmes. Contrairement aux attaques **cryptanalytiques** (lesquelles sont purement mathématiques).

2. Classification des attaques

Une attaque est souvent caractérisée par les données qu'elle nécessite **[1]**:

- **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. La cryptanalyse linéaire fait partie de cette catégorie.
- **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. La cryptanalyse différentielle est un exemple d'attaque à texte clair choisi.
- **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.

3. Familles d'attaques cryptanalytiques

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

- **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. C'est le seul moyen de récupérer la clé dans les algorithmes les plus modernes et encore inviolés comme AES.
- **Attaque par paradoxe des anniversaires:** Le paradoxe des anniversaires est un résultat probabiliste qui est utilisé dans les attaques contre les fonctions de hachage. Ce paradoxe permet de donner une borne supérieure de résistance aux collisions d'une telle fonction. Cette limite est de l'ordre de la racine de la taille de la sortie, ce qui signifie que, pour un algorithme comme **MD5** (empreinte sur **128 bits**), trouver une collision quelconque avec **50%** de chance nécessite **264 hachages** d'entrées distinctes.

4. Stratégies d'attaque du système RSA

4.1 Attaques physiques

Il s'agit dans ce cas d'attaques se basent sur des propriétés physiques de l'appareil en charge de l'implémentation du protocole :

- **Attaques 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 $\boldsymbol{y^d mod\, n}$ (ou $\boldsymbol{d}$ est la clé secrète). $\boldsymbol{d = d_1 d_2 d_3 \ldots d_n}$, chaque bit $\boldsymbol{d_i}$est égal à **0** ou **1**.Or dans la boucle de calcul, on trouve une instruction du type '*si* $\boldsymbol{d_i = 1}$ *alors faire tel calcul sinon ne pas le faire*'.

En mesurant les temps de calcul pour de nombreuses valeurs initiales de y (message crypté), on peut déduire si le calcul qui est fait lorsque $\boldsymbol{d_i = 1}$ 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.

- **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 de proche en proche la valeur de la clé secrète$\boldsymbol{d}$.

Des méthodes existent pour fausser la consommation électrique et rendre cette information inutilisable.

- **Attaques par injection de fautes**

Cette attaque a été conçue en **1996** et présentée comme un nouveau modèle d'attaque physique sur les cartes à puce : « Cryptanalyse en présence d'erreurs de calcul dans les processeurs », Elle se base sur l'utilisation d'une signature erronée puis de la signature correcte **[1]**.

4.2 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: $\boldsymbol{f(X * y) = f(x) * f(y).}$

Il est également risqué de chiffrer plusieurs messages liés au moyen de la même clé publique RSA.

4.3 Problème mathématique

La génération des clés RSA s'appuie sur l'utilisation d'un grand nombre $\boldsymbol{n}$ produit de deux grands nombres premiers $\boldsymbol{p}$et $\boldsymbol{q}$: '$\boldsymbol{n=p*q}$ et $\boldsymbol{e}$ premier avec$\boldsymbol{(p-1)*(q-1)}$, $\boldsymbol{(n,e)}$forment la clé publique, et $\boldsymbol{d}$l'inverse de $\boldsymbol{e}$modulo$\boldsymbol{(p-1)*(q-1)}$**,** la clé privée.

La première façon d'attaquer l'algorithme RSA est de factoriser $\boldsymbol{n}$et de retrouver $\boldsymbol{p}$et $\boldsymbol{q}$, En 1999, un nombre de **512bits** a été factorisé avec une puissance de calcul de **104 Mips/an** (Soit $\mathbf{10^{10}}$ **instructions/s** pendant **un an**).

En prenant la **loi de Moore** comme référence (La puissance double tous les 18 mois) on peut estimer que les clés de **1024 bits** pourront être cassées vers l'an 2010 et que les clés de **2048bits** pourront l'être vers 2030.

D'autres attaques sont possibles à condition de faire des hypothèses sur l'exposant secret d'inverse de $\boldsymbol{e}$ modulo$\boldsymbol{(p-1)*(q-1)}$, Il est facilement possible de retrouver $\boldsymbol{d}$ à partir de $\boldsymbol{n}$ et $\boldsymbol{e}$ par *l'attaque de Wiener* lorsque $\boldsymbol{d}$est inférieur à$n^{\frac{1}{4}}$.

D'autres attaques sont possibles si l'on suppose que certains bits de la clé$\boldsymbol{d}$ sont connus si $\boldsymbol{d}$ a une taille de **k bits**, la connaissance des **k/4 bits** de poids faibles (les moins significatifs) est suffisant de reconstituer complètement la clé.

5. Modèle de l'attaque

L'objectif de l'attaquant n'est pas nécessairement de retrouver la clef privée associée à la clef publique; il s'agit de l'objectif le plus ambitieux, et on parle dans ce cas de *cassage total*. Le plus souvent, l'objectif de l'attaquant est de pouvoir retrouver le message clair associé à un message chiffré

donné. L'attaquant peut aussi poursuivre un objectif plus modeste: obtenir seulement la moitié du texte clair, ou même seulement un bit du message. Le rôle d'un schéma de chiffrement étant de préserver la confidentialité d'un message, un attaquant ne devrait pas pouvoir obtenir une quelconque information sur le message clair à partir du chiffré.

On dit qu'un schéma de chiffrement à clé publique est *sémantiquement sûr* s'il est impossible de distinguer le chiffrement de deux messages. Plus précisément, on considère un attaquant qui, après avoir reçu la clef publique, renvoie deux messages m_0 em_1t de même taille. On chiffre l'un des deux messages et on envoie le chiffré y à l'attaquant. Ce dernier doit déterminer de quel message ($m_0$ou m_1) y est le chiffré, avec probabilité significativement supérieure à **1/2**. Le schéma de chiffrement est sémantiquement sûr s'il n'existe pas de tel attaquant **[2].**

6. Attaques de RSA

6.1Attaques sur les implémentations du RSA

On va d'abord s'intéresser aux attaques concernant les implémentations du RSA. Ces attaques montrent qu'une implémentation du RSA ne doit pas seulement se protéger des attaques mathématiques du chiffrement RSA.

Attaques par erreur

Il était avantageux d'utiliser les restes chinois pour accélérer le calcul de $m^d mod\ n$. *Boneh, De Millo et Lipton* **[7]** ont remarqué que cette méthode présentait le danger potentiel suivant. Supposons que par accident (ou par intervention d'un adversaire), la carte fasse une seule erreur d'instruction lors du calcul de la signature, de sorte que seule la valeur de$m^d mod\ p$ soit correctement calculée, et non celle de $m^d mod\ q$.

La signature incorrecte s vérifiera donc$s^e \equiv m\ mod\ p$et s^e ! $\equiv m\ mod\ q$, et donc le $pgcd$ de s$s^e - m$ et n révélera la factorisation de n **[3]**.

6.2 Attaque sur l'exposant du chiffrement

Pour rendre plus efficace le chiffrement RSA, la valeur **e=3** est souvent utilisée. Si Alice correspond avec 3 entités différentes, possédant des modulos n_1, n_2, n_3, et qu'elle envoie le même message m ($m < n_i$ **pour i=1, 2, 3**) chiffré respectivement par c_1, c_2, c_3 (noté qu'on a alors$c_i = m^3 mod n_i$), il est alors possible de retrouver le message m en utilisant le théorème des restes chinois. En effet, le système ($x = c_1 mod n_1, x = c_2 mod n_2, x = c_3 mod n_3$) possède alors une seule solution (une solution seram^3) : Il suffira ensuite de trouver la racine cubique de x pour retrouver m.

L'utilisation d'un exposant de chiffrement petit n'est donc pas interdite à condition de prendre garde à ne pas envoyer le même message plusieurs fois **[4].**

6.3 Timing Attacks

Les attaques de RSA les plus percutantes et les plus novatrices sont certainement ce qu'on appelle **les timing attacks**. RSA consiste en la programmation de la fonction$\boldsymbol{R = y^x mod\, n}$, avec $\boldsymbol{n}$ public et $\boldsymbol{y}$ pouvant être intercepté par l'attaquant. Le but de l'attaquant sera alors de trouver **x**, la clef secrète. Pour l'attaque, la victime doit calculer $\boldsymbol{y^x mod\, n}$ pour quelques valeurs de $\boldsymbol{y}$où y, $\boldsymbol{n}$ et le temps de calcul sont connus de l'attaquant [**5].**

Notons au passage que si un nouvel exposant $\boldsymbol{x}$est choisi pour chaque opération l'attaque ne fonctionne plus. L'information nécessaire et les mesures de temps de calcul peuvent être obtenus à l'aide d'un protocole interactif puisqu'un attaquant pourrait enregistrer les messages reçus par la cible et mesurer le temps mis pour répondre à chaque **y**. L'attaque peut être traitée comme un problème de détection de signal. Le signal consiste en la variation de temps due aux exposants inconnus et les propriétés du signal déterminent le nombre de mesures du temps requises pour l'attaque. Etant donnés **j** messages$\boldsymbol{y_0, y_1, \ldots, y_{j-1}}$, avec des mesures de temps correspondantes $\boldsymbol{T_0, T_1, \ldots, T_{j-1}}$, la probabilité qu'une réponse $\boldsymbol{x_b}$ pour le premier bit exposant $\boldsymbol{b}$ soit correcte est proportionnelle à :

$$\boldsymbol{P(x_b) \alpha \prod_{i=0}^{j-1} F(T_i - t(y_{i-} x_b))}$$

Où$\boldsymbol{t(y_{i-} x_b)}$ est le temps requis pour les **b** premières itérations du calcul de $\boldsymbol{y_i^x mod\, n}$ en utilisant le bit exposant$\boldsymbol{x_b}$, et $\boldsymbol{F}$ est la probabilité attendue de distribution de $\boldsymbol{(T_i - t(y_{i-} x_b)}$ sur toutes les valeurs de $\boldsymbol{y}$ et $\boldsymbol{x_b}$ correct. Comme $\boldsymbol{F}$ est définie comme la distribution de probabilité de $\boldsymbol{(T_i - t(y_{i-} x_b)}$, si $\boldsymbol{x_b}$ est correcte c'est la meilleure fonction pour prédire$\boldsymbol{(T_i - t(y_{i-} x_b)}$

6.4Attaques élémentaires

a. Fuite d'information

On remarque qu'un chiffré RSA révèle au moins un bit d'information sur le message clair. En effet si $\boldsymbol{c = m^e\, mod n}$on a, comme $\boldsymbol{e}$ est nécessairement impair :

$$\boldsymbol{\left(\frac{c}{n}\right) = \left(\frac{m}{n}\right)^e = \left(\frac{m}{n}\right)}$$

Le symbole de Jacobi du message$\boldsymbol{m}$ peut donc se déduire du chiffré $\boldsymbol{c}$**.** Une notion de sécurité forte doit empêcher de déduire facilement du chiffré toute information sur le message.

b. Messages courts

Supposons que l'on chiffre un message court$\boldsymbol{m}$. Le chiffré de $\boldsymbol{m}$ est$\boldsymbol{c = m^e\, mod n}$. Si$\boldsymbol{m}$ est tellement court que $\boldsymbol{m < n^{1/e}}$(ce qui ne peut arriver que si $\boldsymbol{e}$ est bien plus petit que$\boldsymbol{\phi}$(**n**), par exemple si $\boldsymbol{e = 3}$),

alors $\boldsymbol{c = m^e}$dans Z, et donc on peut efficacement décrypter$\boldsymbol{c}$ en extrayant une racine$\boldsymbol{e^{iéme}}$dans Z. Ainsi, il n'y a absolument aucune sécurité si l'on chiffre une clef de session **128 bits** au moyen de **RSA-1024** d'exposant 3 avec la fonction de chiffrement de base. Là encore, cet exemple montre qu'il faut randomiser les messages.

c. Recherche exhaustive

On remarque que la fonction de chiffrement RSA $\boldsymbol{m \rightarrow m^e mod\ n}$ est déterministe : un message est toujours chiffré en le même chiffré. Etant donné un chiffré $\boldsymbol{c}$et un message $\boldsymbol{m}$,on peut donc vérifier si $\boldsymbol{c}$ est le chiffrement de$\boldsymbol{m}$. Ainsi, si l'ensemble des messages possiblesest connu et de petite taille, alors on peut décrypter par recherche exhaustive, en testant tous les messages possibles. Par exemple, si l'on chiffre un texte clair en chiffrant au moyen de RSA chaque lettre du texte séparément, alors n'importe qui pourra décrypter.

Pour éviter la recherche exhaustive, il est indispensable de randomiser les messages avant chiffrement. On ne peut atteindre de bonnes notions de chiffrement qu'en ayant une fonction de chiffrement probabiliste (en rajoutant de l'aléa chaque chiffrement), et non déterministe. C'est pourquoi une fonction à sens unique `a trappe doit être transformée avant d'être utilisée en chiffrement asymétrique **[3].**

6.5 Cryptanalyses élémentaires de RSA

La cryptanalyse élémentaire sur le système RSA consiste à factoriser le module pour retrouver les valeurs de p et q**[6].**

6.5.1 Cryptanalyse de RSA connaissant φ(N)

Proposition : Soit N un module de RSA, Si on connait φ(N), alors on peut factoriser N.

Démonstration : Supposons que φ(N) est connu. Ainsi, on dispose d'un système de deux équations en p et q : $\begin{cases} p * q = N \\ p + q = N + 1 - \varphi(\mathrm{N}) \end{cases}$

Qui donnent l'équation en p : $p^2 - \big(N + 1 - \varphi(\mathrm{N})\big)\mathrm{p} + \mathrm{N} = 0$

On obtient ainsi :$p = \frac{N+1-\varphi(\mathrm{N})+\sqrt{(\mathrm{N}+1-\varphi(\mathrm{N}))^2-4\mathrm{N}}}{2}$

$$q = \frac{N + 1 - \varphi(\mathrm{N}) + \sqrt{(\mathrm{N} + 1 - \varphi(\mathrm{N}))^2 - 4\mathrm{N}}}{2}$$

6.5.2 Utilisation du même module et deux exposants différents

Proposition :

Soit N un module RSA. Soient e_1 et e_2 deux exposants premiers entre eux. SI un message clair M est chiffré avec e_1 et e_2, alors on peut calculer M.

Démonstration :

Soient C_1 et C_2 deux messages chiffrés par :

$$C_1 \equiv M^{e_1}(mod\ N),$$
$$C_2 \equiv M^{e_2}(mod\ N),$$

Et supposons que C_1 et C_2 sont rendus publiques. si $pgcd(e_1, e_2) = 1$, alors il existe deux entiers x_1 et x_2 tels que $x_i < \frac{1}{2} e_i$ vérifiant $e_1 x_1 - e_2 x_2 = \pm 1$. Ces deux entiers peuvents être déterminter par l'algorithme d'Elucide. D'autre part, ils vérifient

$$\left|\frac{e_1}{x_1} - \frac{e_2}{x_2}\right| = \frac{|e_1 x_1 - e_2 x_2|}{x_1 e_1} = \frac{1}{x_1 e_1} < \frac{1}{2x_1^2}$$

Ainsi x_1 et x_2 peuvent être déterminées comme dénominateurs et numérateur de l'une des convergentes de $\frac{e_1}{e_2}$. Si $e_1 x_1 - e_2 x_2 = 1$, on obtien alors

$$C_1^{x_1} C_2^{-x_2} \equiv M^{e_1 x_1} M^{-e_2 x_2} \equiv M^{e_1 x_1 - e_2 x_2} \equiv M(mod\ N)$$

Ce qui donne le message M. si $e_1 x_1 - e_2 x_2 = -1$, on calcule $C_1^{-x_1} C_2^{x_2}$ et on obtient le même résultat.

6.5.3 Utilisation de modules différents pour le même message

Dans cette attaque, on considère qu'un message M est envoyé un certain nombre de fois en utilisant plusieurs clés publiques (N_i, e_i). Ce scénario peut se produire si le même message doit être diffusé à plusieurs destinataires. Cette attaque, due à Hastad, est basée sur le théorème des restes chinois.

Théorème : Si les entiers $N_1, N_2, \dots, N_k$ sont deux à deux premiers entre eux, alors le système :

$$\begin{cases} x = a_1(mod\ N_1) \\ x = a_2(mod\ N_1) \\ \quad \dots \\ x = a_k(mod\ N_k) \end{cases}$$

Admet une solution unique modulo $\mathbf{N} = \prod_{i=1}^{k} \mathbf{N_i}$. Cette solution est :

$\mathbf{x \equiv \sum_{i=1}^{k} a_i p_i M_i\ (mod\ N)}$,

Avec $\mathbf{p_i = \frac{N}{N_i}}$ **et** $\mathbf{M_i \equiv p_i^{-1} (mod\ N_i)}$,

Démonstration :

Soit$\boldsymbol{N = \prod_{i=1}^{k} N_i}$, $\boldsymbol{p_i = \frac{N}{N_i}}$, **et** $\boldsymbol{M_i \equiv p_i^{-1}(mod\ N_i)}$. Alors, pour chaque $\boldsymbol{i, 1 \leq i \leq k}$, on a $\boldsymbol{p_i M_i \equiv 1(mod\ N_i)}$ **et** $\boldsymbol{N_i | pj}$ **si** $\boldsymbol{i \neq j}$. Ainsi, avec $\boldsymbol{x \equiv \sum_{i=1}^{k} a_i p_i M_i\ (mod\ N)}$, on a pour $\boldsymbol{1 \leq i \leq k}$, en considérant $\boldsymbol{x\ modulo\ N_i}$:

$$\boldsymbol{x \equiv a_i p_i M_i + \sum_{j \neq i}^{k} a_j p_j M_j \equiv a_i + \sum_{j \neq i}^{k} a_j p_j M_i\ .\frac{p_j}{N_i} \equiv a_i (mod\ N_i).}$$

Ceci montre que x est la solution du système. Si $\boldsymbol{x' - x \equiv 0\ (mod\ N_i).}$

Puisque les entiers $\boldsymbol{N_i, 1 \leq i \leq k,}$ sont premiers entre deux à deux, alors$\boldsymbol{x' - x \equiv 0\ (mod\ N)}$

Ainsi, puisque $|x' - x| < N$, on a $\boldsymbol{x' = x,}$ ce qui montre l'unicité de la solution.

6.6 Cryptanalyse de RSA par factorisation continues

- **Attaque de Wiener**

Théorème :

Soit $N = pq$ un module RSA où les nombres premier p et q sont de même taille ($q < p < 2q$). Puisque $\boldsymbol{N = pq > q^2,}$ **alors** $\boldsymbol{q < \sqrt{N}.}$

D'autre part on a

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

Si e est une clé publique et d la 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 $ed - k\varphi(N) = 1$. Puisque

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

Alors

$$\left|\frac{\boldsymbol{e}}{\boldsymbol{N}} - \frac{\boldsymbol{k}}{\boldsymbol{d}}\right| = \frac{|\boldsymbol{ed - kN}|}{\boldsymbol{Nd}}$$

$$= \frac{|\boldsymbol{ed - k\varphi(N) - kN + k\varphi(N)}|}{\boldsymbol{Nd}}$$

$$= \frac{|\boldsymbol{1 - k(N - \varphi(N))}|}{\boldsymbol{Nd}}$$

$$< \frac{\boldsymbol{k\big(N - \varphi(N)\big)}}{\boldsymbol{Nd}}$$

$$< \frac{\boldsymbol{3k\sqrt{N}}}{\boldsymbol{Nd}}$$

$$= \frac{\boldsymbol{3k}}{\boldsymbol{d\sqrt{N}}}$$

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**.

L'attaque de *Wiener* peut donc être résumée dans l'algorithme suivant :

Entrée	**Un module RSA N et un exposant publique e pour lequel la clé secrète vérifie $d < \frac{1}{3}N^{\frac{1}{4}}$**
Sortie	La factorisation de **N** et la clé secrète **d**.
	1. Déterminer la liste des résuidte de $\left\lfloor \frac{e}{N} \right\rfloor$.
	2. Pour chaque réduite xy telle que $y < \frac{1}{3}N^{\frac{1}{4}}$ **:**
	3. Calculer $\vartheta = \frac{ey-1}{x}, \delta = (N+1-\vartheta)^2 - 4N, et\ p_2 = \left\lfloor \frac{(N+1-\vartheta+\sqrt{\delta})}{2} \right\rfloor$
	4. **Si** pgcd$(p_2, N) > 1$**alors**
	5. Afficher $d = y, p = p_2\ et\ q = \frac{N}{p_2}$.
	6. **Fin Si**

Chapitre 4

CONCEPTION ET REALISATION

1. Introduction

Ce chapitre va être consacré aux aspects pratiques pour la réalisation de notre application « RSA Attack Simulator » (Simulation de l'attaque sur RSA).

Son but étant de simuler une attaque par factorisation sur le module de RSA lors des transferts des messages puis de voir la partie intégrité et authentification du message.

2. Ressources utilisées

2.1 Les ressources matérielles

- Processeur **Intel Core™ « i3 »**CPU d'une fréquence de **2.53 GHZ**.
- Une mémoire vive d'une capacité de **2 Go**.
- Une carte graphique de **512 Mo** de mémoire dédiée sur **2267 Mo** de mémoire graphique totale disponible.

2.2 Les ressources logicielles

- Système d'exploitation : **Windows 7**.
- Langage de programmation **C++.**
- L'éditeur utilisé est le **CodeGear™ C++Builder® 2009**.

Pourquoi C++ BUILDER

Notre choix s'est porté sur celui-ci car :

- Il permet d'intégrer une interface graphique, en utilisant la syntaxe du langage C++.
- Il offre selon la guise du programmeur un travail en orienté objet, ou en programmation fonctionnelle.
- Offre de nombreux outils pour un développement rapide et efficace.
- Il regroupe toute la puissance du c et c++.
- Il offre les tâches de base qui constituent le cœur d'un programme Windows.

Sa structure nous offre la manipulation facile des images et fichiers.

3. Conception

Le but principal de notre application est de faire une simulation de l'attaque sur RSA par la méthode de factorisation, pour sa réalisation nous passerons par plusieurs étapes :

- Génération des clefs publiques et privées pour Alice et Bob.
- Chiffrement du message lors de son transfert en selon la méthode choisit :
  - Sans signature : Chiffrement du message avec la clef publique du destinataire.

- Avec Signature : Chiffrement du message haché avec la clef privée de l'expéditeur.

- Déchiffrement du message reçu selon la méthode choisit durant l'envoi :
  - Sans signature : Déchiffrement avec la clef privée de l'expéditeur.
  - Avec Signature : Déchiffrement de la signature avec la clef publique du de l'expéditeur et comparaison avec le message haché reçu afin de garantir l'authenticité.
- Attaque sur le module n de RSA afin de retrouver les deux facteurs p et q qui permet de calculer la clef privée.
- Interception et déchiffrement des messages transmis entre Alice et Bob.

Ces étapes peuvent se résumer par l'organigramme suivant :

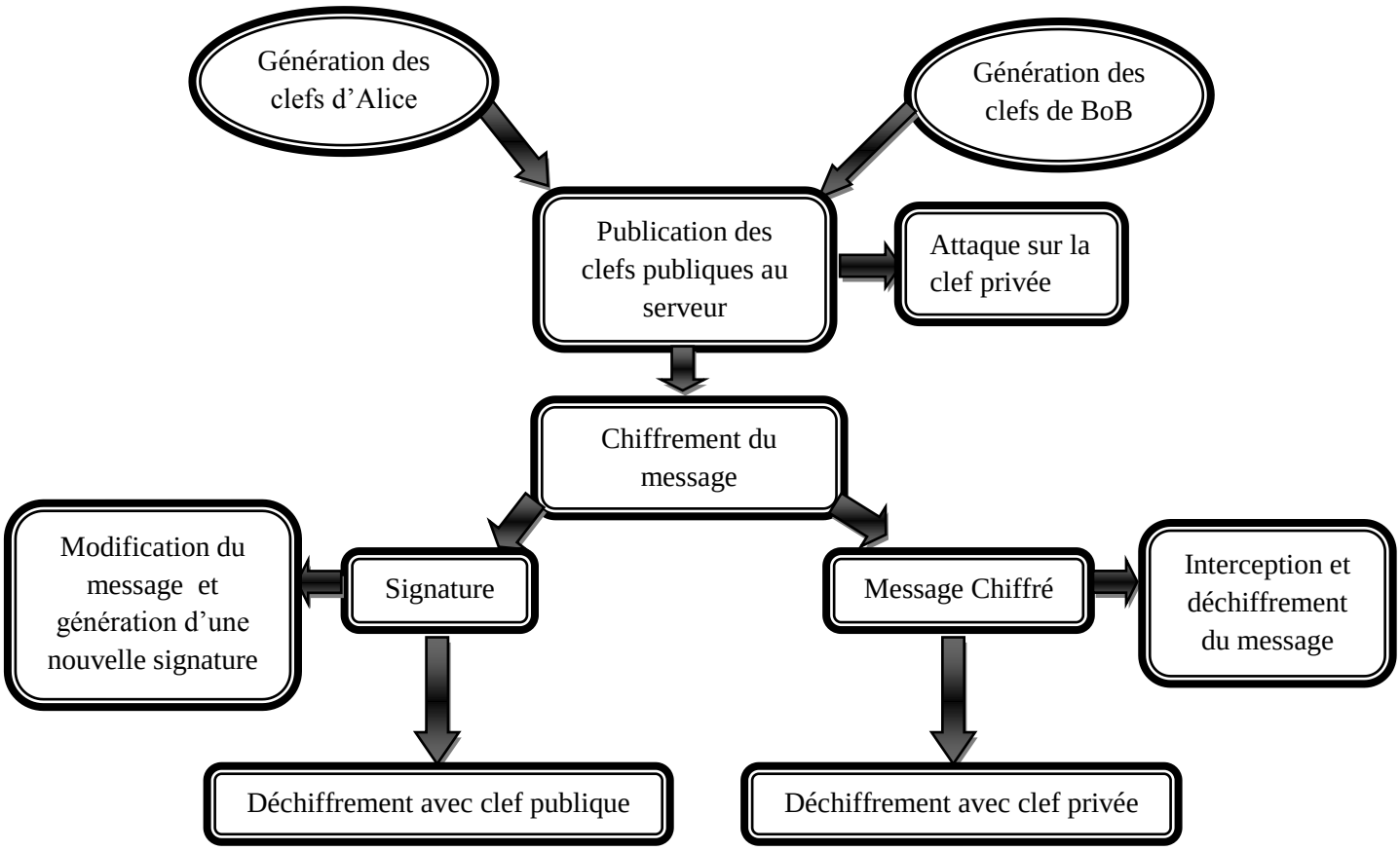

Figure 10. Organigramme de l'application

4. Présentation de l'application

4.1 Objectif

L'objectif de notre travail est de réaliser une application de simulation d'une attaque par factorisation sur le système cryptographique RSA.

4.2 Description de l'application

Notre application « RAS » (RSA Attack Simulator) simule l'attaque par factorisation sur le système cryptographique RSA, en mettant en point plusieurs scénarios et cas possibles **(Figure 11)**.

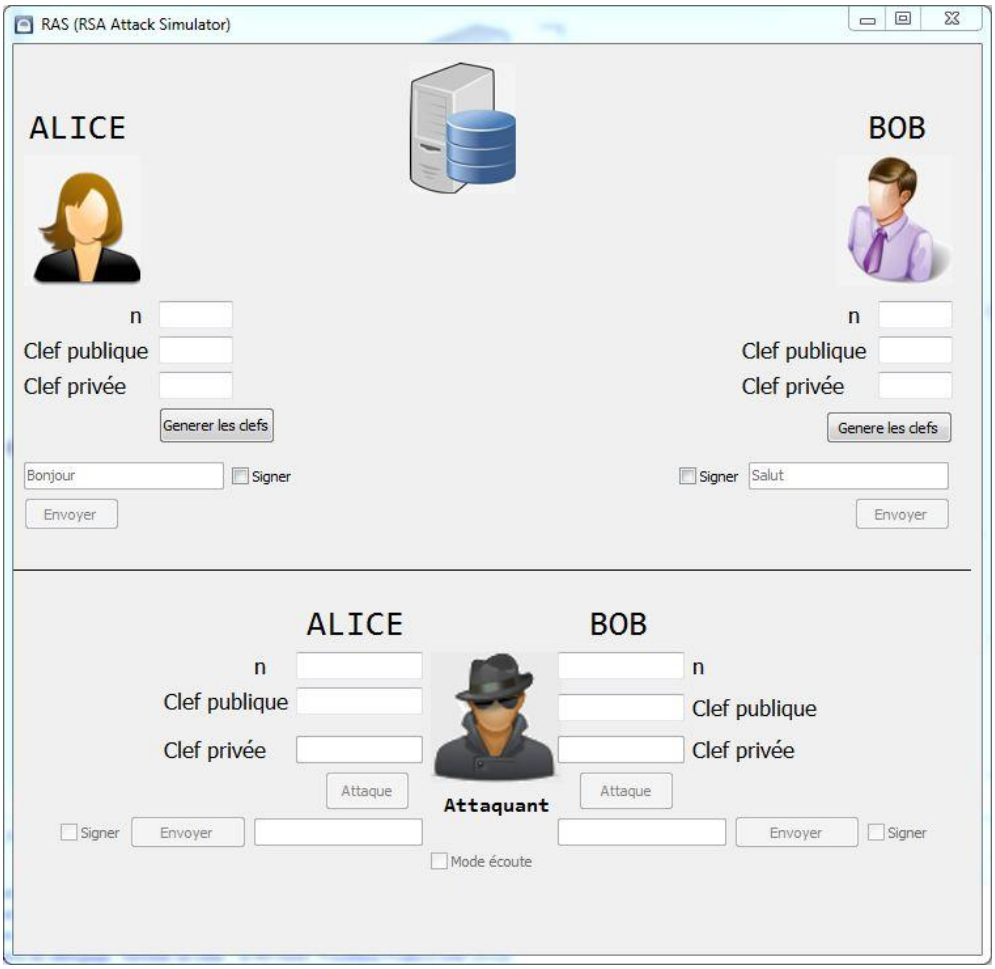

Figure 11. RAS au premier lancement

4.3 Démarches d'utilisation

a) L'application nécessite d'abord la génération des clefs (clef publique, clef privée) soit pour Alice ou Bob ou les deux **(Figure 12)**.

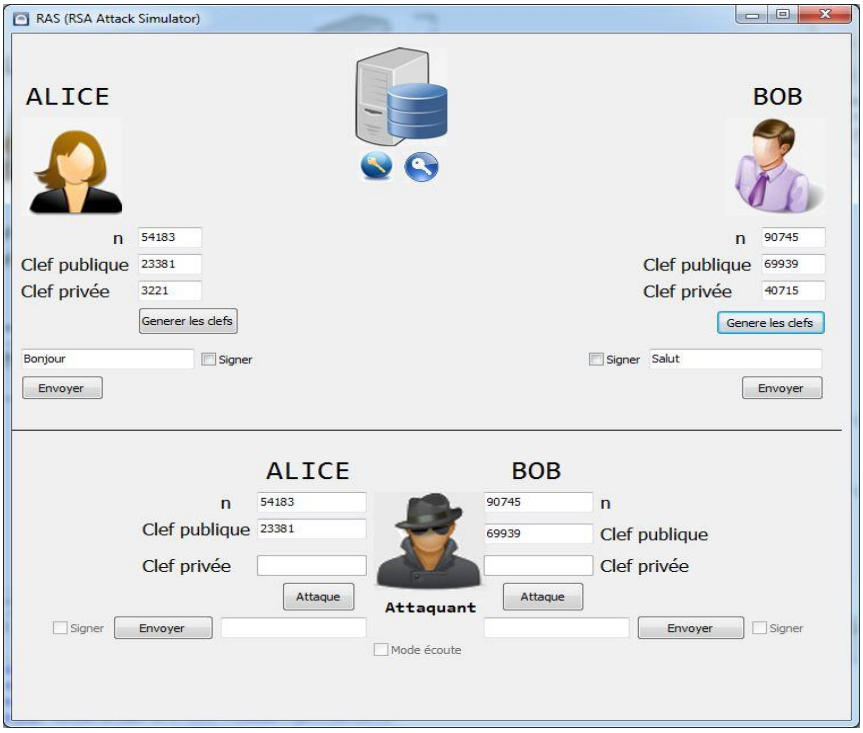

Figure 12. Génération des clefs pour Alice et Bob

b) Après génération des clefs plusieurs options seront possibles à utiliser telles que l'envoie des messages signés ou non signés entre Bob et Alice.

c) L'attaquant pourra ensuite réaliser l'attaque par factorisation sur la clef privée de Alice et Bob **(Figure 13)**.

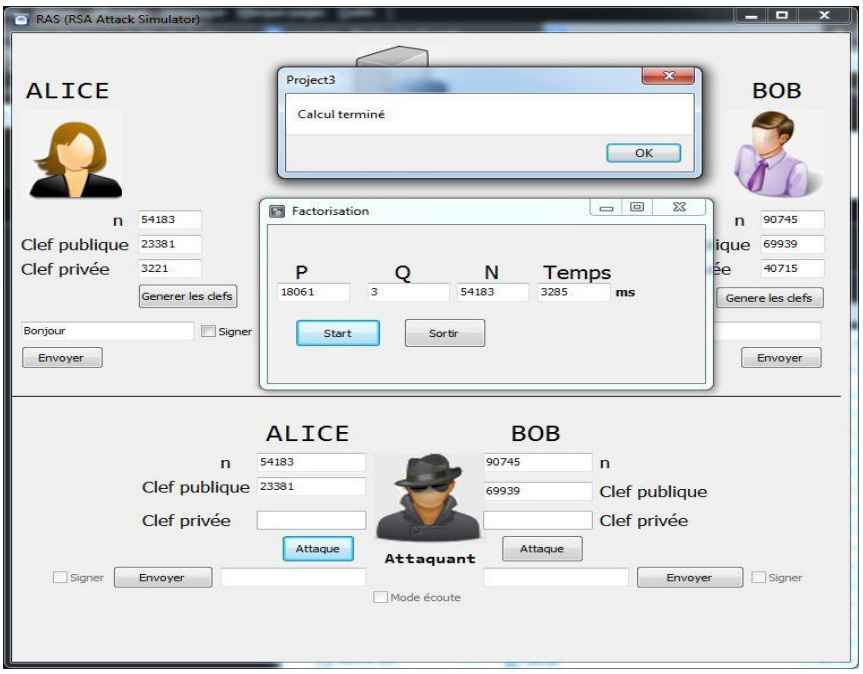

Figure 13. Factorisation de la clef privée d'Alice

d) Une fois que l'attaquant à les clefs privées il pourra intercepter et déchiffrer les messages transmis entre Alice et Bob en activant le mode écoute **(Figure 14)**.

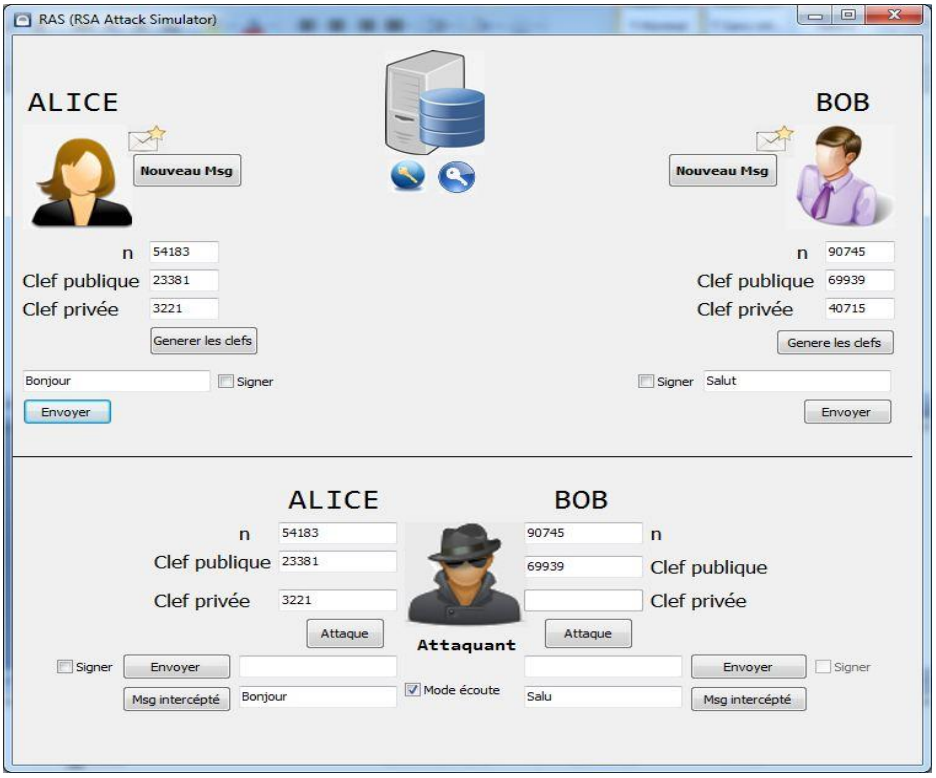

Figure 14. Interception et déchiffrement des messages

e) Si l'attaquant ne signe pas le message lors d'un transfert de messages signés entre Alice et Bob alors les destinataires seront alertés en leur indiquant que le message à été modifié **(Figure 15).**

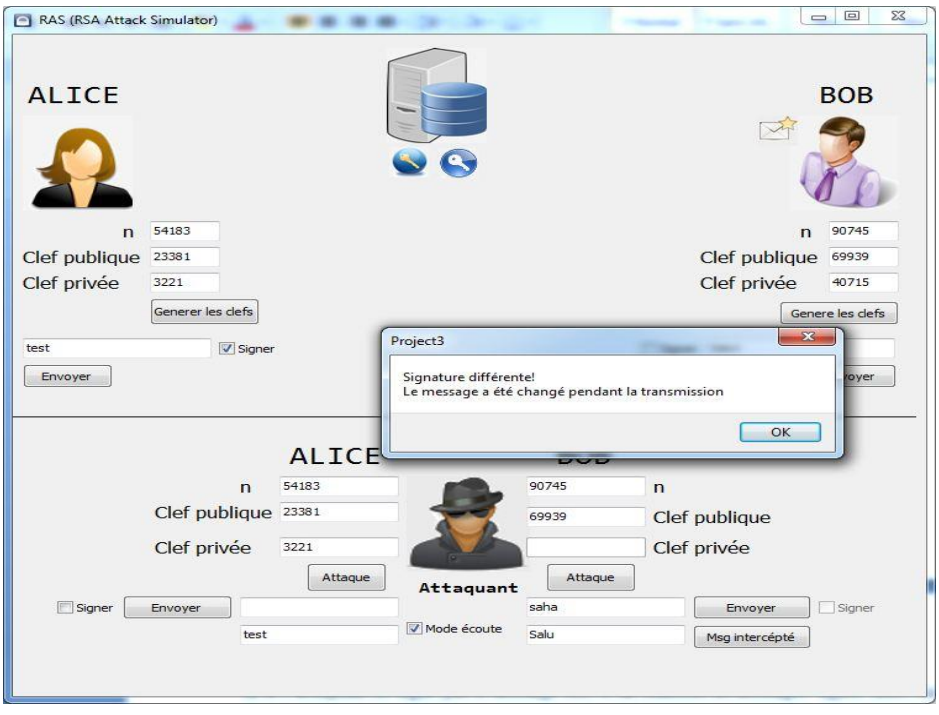

Figure 15. Alerte d'un changement de message pendant la transmission

5. Résultats

5.1 Factorisation du module n

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

Le tableau suivant montre le temps nécessaire pour factoriser le module n de différentes tailles.

N	P	Q	D	Temps (millisecondes)
33337	629	53	27699	1515
54183	18061	3	3221	3285
72563	149	149	2609	5430
84043	367	229	51279	7635
90745	18149	5	40715	8175
121037	17291	7	19155	14625
761581	5479	139	1151	559575

Tableau 1. Temps nécessaire pour factoriser le module n

5.2 Signature

Dans le cas où le message est signé, il doit être d'abord haché avec l'algorithme SHA1, une fois le message haché, il sera chiffré avec la clef privée de l'émetteur pour former une signature, l'attaquant doit d'abord factoriser le module n de RSA, puis calculer la clef privée d avec les deux facteurs p et q retrouvés par la factorisation, ensuite il devra signer le nouveau message pour usurper l'identité de l'expéditeur et contourner l'authenticité.

Le tableau suivant montre le temps nécessaire pour factoriser et signer le message.

N	P	Q	D	Temps (millisecondes)
33337	629	53	27699	4560
54183	18061	3	3221	5655
72563	149	149	2609	8280
84043	367	229	51279	10110
90745	18149	5	40715	11655
121037	17291	7	19155	18555
761581	5479	139	1151	563505

Tableau 2. Temps nécessaire pour la factorisation et la signature

Discussion

On remarque que plus la valeur de n qui constitue le module RSA est grande plus le temps de calcul par factorisation est important, on remarque aussi que lorsque la clef privée est cassée on peut contourner l'authenticité et usurper l'identité du véritable expéditeur.

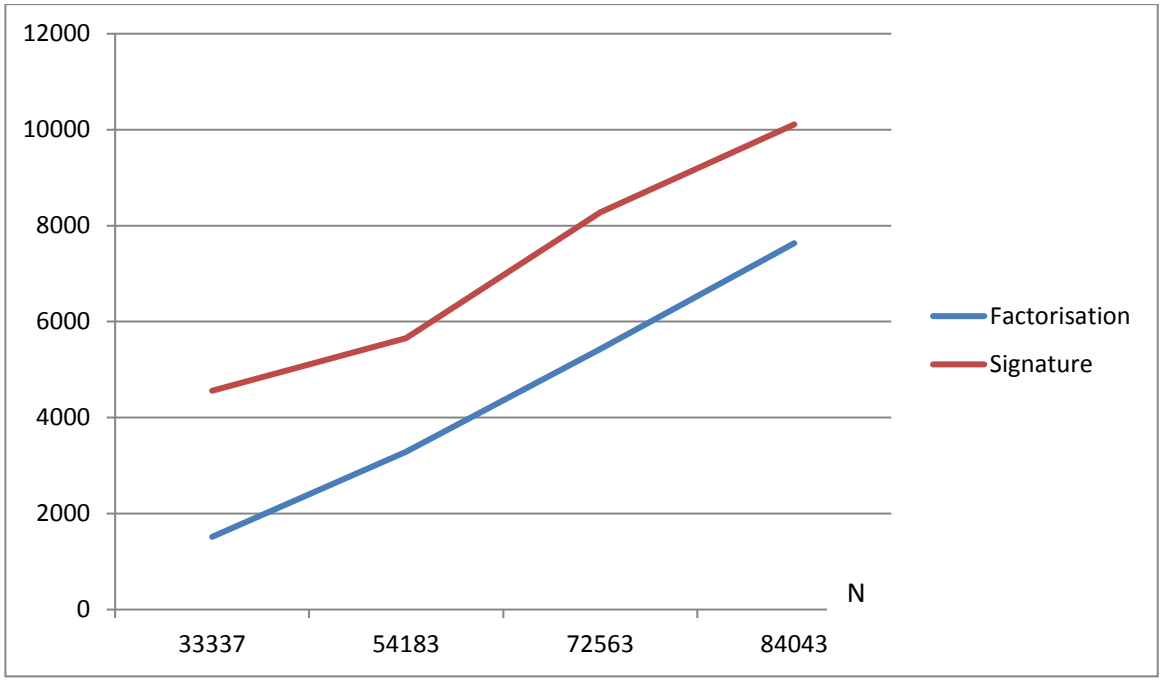

Figure 16. Graphe du temps calculé pour la factorisation et la signature

6. Conclusion

Les résultats obtenus par les différents scénarios testés montrent que la sécurité du système cryptographique RSA est liée avec le choix de la clef, plus la taille de la clef est grande plus la difficulté de l'attaque augmente et plus le risque diminue.

Conclusion générale

Malgré ses avantages, la cryptographie a ses limites et ne doit pas être un substitut aux autres mesures de sécurité. Bien utilisée, c'est un outil puissant mais si le chiffrement est utilisé improprement, il peut nuire aux objectifs réels de l'entreprise.

De plus, la gestion des clés est un problème fondamental et constitue la partie la plus difficile de la cryptographie. Il est donc indispensable de sécuriser chaque étape intervenant dans cette gestion : génération, transfert, stockage, longévité ou encore destruction des clés.

L'organisation de concours pour tester la puissance des algorithmes de chiffrement est très en vogue actuellement. Cette pratique couramment utilisée a surtout pour volonté de prouver qu'aucun algorithme n'est infaillible et ainsi faire en sorte que les recherches dédiées à la cryptographie ne cessent d'évoluer.

Dans ce mémoire nous avons présenté les différentes attaques et cryptanalyses sur les algorithmes de cryptographie et en détaillant l'attaque par factorisation sur RSA qui est le sujet de notre thème.

Les tests ont montré que l'attaque dépend de la taille de la clef privée ainsi que la puissance des ressources matérielles.

Nous suggérons pour des travaux futurs l'intégration de nouvelles méthodes d'attaque sur l'algorithme de RSA et de comparer et analyser les résultats obtenus de ces méthodes.

Bibliographie

Chapitre I

[1] U. Feige and A. Fiat and A. Shamir. Zero-Knowledge Proofs of Identity, J. Cryptology, 1.1988.77–94.

[2] COMPAGNON L, DERMAUX S -La machine Enigma : étude bibliographique. Ecole des Mines de Douai. 2005.

[3] The Friedman Legacy: A Tribute to William and Elizebeth Friedman, NSA Publication: Sources in Cryptologic History, n 3, 1992.

[4] Daniel Barsky . Cours de Cryptographie (version préliminaire 2005/2006) février 2006 page 26.

[5] Rivest RL. Shamir A. Adleman L. "A method for obtaining digital signature and public keycryptosystems".

[6] Thèse de Doctorat en sciences en Informatique, Université Mentouri de Constantine, Octobre 2008.Chikhi Samia : « Contribution à l'authentification souple d'images digitales par des techniques de marquage numérique ».

[7] Mireille CAMPANA .Cours La cryptographie. chapitre 10 page 57.

[8] Jean-Pierre MILLIET. Université Arles : Cours « Programmation Cryptographie » page 15, 16,17 Mai 2000.

[9] Daniel Barsky février 2006Cours de Cryptographie (version préliminaire 2005/2006) chapitre 6 page 43

Chapitre II

[1] ES Morges-Beausobre - Vincent. S 9VSB 3 - La Cryptographie & Le RSA. 2005

[2] Diffie W. et Hellman M. E. – New directions in cryptography. IEEE Trans. Inform. Theory, vol. IT-22, Nov 1976, pp. 644–654.

[3] Rivest R. L., Shamir A. et Adleman L. M. – A method for obtaining digital signatures and public-keycryptosystems. Communications of the ACM, vol. 21, N 2, 1978, p. 120–126.

[4] futura-sciences.com

[5] Thèse Doctorat de l'Université de Versailles Saint-Quentin en-Yvelines. Analyse cryptographique des altérations d'algorithmes. Alexandre Berzati. 29 Septombre 2010

[6] DESTREE Lucile – MARCHAL Mickaël .Mini-RSA Programme d'initiation au chiffrement RSA

[7] Louis Granboulan, Phong Nguyen,David Pointcheval.Edition 2004.Cours de magistère M.M.F.A.I.´ Ecole normale supérieure. Conception et preuves d'algorithmes cryptographiques

[8] B. Schneier, Cryptographie appliquée : protocoles, algorithmes et codes sources en C, J. Wiley, 1997.

Chapitre III

[1] Portail de la cryptologie http://fr.wikipedia.org/wiki/Wikipédia:Portail_Cryptologie

[2] http://www.jscoron.fr/thesis/node6.html

[3] Louis Granboulan ,PhongNguyen,DavidPointcheval.Edition 2004.Cours de magistère M.M.F.A.I.´ Ecole normale supérieure.Conception et preuves d'algorithmes cryptographiques

[4] http://www.cryptis.fr/picsi/attaque/x6138.html

[5] http://www.jeulin.net/crypto6/RSA/rsa.htm

[6] Abderrahmane Nitaj .Laboratoire de Mathèmatiques Nicolas Oresme Université de Caen, France.Version du 28 juin 2009. CRYPTANALYSE DE RSA .page 15

[7] Boneh D., DeMillo R. et Lipton R. – On the importance of checking cryptographic protocols for faults. In Proc. of Eurocrypt '97. Lecture Notes in Computer Science, volume 1233, p. 37–51. – Springer-Verlag, 1997.