الجمهورية الجزائرية الديمقراطية الشعبية
République algérienne démocratique et populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique
المركز الجامعي لعين تموشنت
Centre Universitaire Belhadj Bouchaib d'Ain-Temouchent
Institut des Sciences et de la Technologie
Département de Génie Electrique

Projet de fin d'études
Pour l'obtention du diplôme de Master en :
Domaine : SCIENCE ET TECHNOLOGIE
Filière : Electronique
Spécialité : Génie des Télécommunications

Développement d'une Infrastructure à Clés Publiques

Présenté Par :
1) Ramdane Youssef

Devant les jurys composés de :

Dr. Benaissa Mohammed	MCA	C.U.B.B (Ain Temouchent)	Encadreur
Dr. MANA Mohammed	MCA	Université de Tlemcen	Co-encadreur
Dr. Ferouani .S	MCA	C.U.B.B (Ain Temouchent)	Examinatrice
Mr. Benazza .B	MAA	C.U.B.B (Ain Temouchent)	Président
Mr. Krim .M	MAA	Université Sidi Belabbes	Invité

Année universitaire 2014/2015

Remercîment

باسم الله و الصلاة و السلام على رسول الله

Avant d'exposer ce thème, je remercie le bon Dieu pour toute la force et patience qui m'avait donnée pour achever à ce niveau modeste et pour ma réussite pour arriver à ce point dans mes études.

Je remercie mes chère parents pour leur aide immense et de pouvoir me soutenir dans mon chemin de vie et d'études de puis le premier jour de l'école jusqu'à ce moment où vous lisez ce modeste thème.

Je remercie mon encadreur qui m'a aidé à faire ce travail malgré les responsabilités et les empêchements qu'il avait, et pour son temps qui ma donner pour faire ce travail.

Un spécial dédicace à monsieur Ben Azza qui m'a aidé le long de mon chemin Licence et Master et à notre chère Dr Ferouani qui mérite du succès et de bonheur dans sa vie, et plus de réussite dans son chemin et carrière, elle mérite que du bien dans sa vie, et au Dr Ben Aissa aussi qui m'a trop inspirer et supporté et à Dr Soltane qui m'a aidé à réaliser beaucoup de chose et qui ma soutenue le long de mes études ici.

Je remercie tous les personne qui ont une empreinte dans ma vie comme étudiant de proche ou de loin.

Sommaire

Sommaire

Chapitre II LA SIGNATURE ELECTRONIQUE

Sommaire

Chapitre III L'infrastructure à clef publique ICP

Chapitre IV Réalisation en C

List des figures

Chapitre I : INTRODUCTION A LA CRYPTOGRAPHIE

Chapitre II : LA SIGNATURE ELECTRONIUE

Chapitre III : L'INFRASTRUCTURE à CLEF PUBLIQUE

CHAPITRE IV : REALISATION EN C

List des tableaux

Chapitre I : INTRODUCTION A LA CRYPTOGRAPHIE

Introduction générale

Dans une société où la sécurité et la confidentialité des communications sont de plus en plus important, les certificats numériques, véritables cartes d'identités pour les utilisateurs de réseaux informatiques, sont un domaine en plein essor. Les certificats numériques sont diffusés dans une infrastructure à clés publiques, infrastructure de plus en plus répandue dans les entreprises.

Une infrastructure à clefs publiques est un ensemble de solutions techniques basées sur la cryptographie à clé publique. Cette dernière emploie une combinaison de clefs publiques et privées, de signatures numériques, des certificats numériques, et des autorités de certification (AC), pour répondre à des nouvelles exigences de sécurité.

Un certificat numérique (aussi appelé certificat électronique) est un fichier permettant de certifier l'identité du propriétaire d'une clé publique, un peu à la manière d'une carte d'identité. Un certificat est généré dans une infrastructure à clés publiques par l'autorité de certification (Certification Authority , CA) qui a donc la capacité de générer des certificats numériques contenant la clé publique en question.

Dans le cadre de notre travail, nous allons déployer une infrastructure à clefs publiques. Notre objectif consiste à générer des clefs privées/publiques et générer et signer des certificats électroniques.

Le manuscrit s'articule autour de quatre chapitres :

Le premier chapitre présente une introduction à la cryptographie. Le deuxième chapitre décrit les certificats électroniques. Le troisième chapitre définit l'infrastructure à clefs publiques et le quatrième chapitre montre les résultats de notre application développée en langage C.

Résumé

Résumé

Le PKI ou ICP est un system baser sur la cryptographie a clef publique qui gêner une paire de clef « publique et privée » qui sera utiliser pour le chiffrement et déchiffrement des informations qui circule entre deux entité et à l'aide d'une fonction d'hachage utiliser pour vérifier l'identité des parties communicante par la vérification de la signature des certificats délivré avec les messages et les informations qui circule . Beaucoup de langues de programmation peuvent être utilisé pour faire une modélisation des PKI comme le OpenSSL et le Java et le C++, nous dans notre travaille on a réalisé un modèle de PKI en C comme langage de programmation, et on a choisis le RSA comme algorithme pour la génération des paire de clef « publique et privé », avec un modèle de certificat numérique et une signature MD5.

L'intégration d'un PKI dans les systèmes de communication est devenue une nécessité dans nos jours, sont utilisation assure la confidalité dans l'échange des informations sur internet et dans les transactions Bankers et dans la sécurité des smartphones et dans tous les systèmes de communication électronique et tous sont fonctionnement se faire automatiquement dans un temps trop court.

Mots-clés : Cryptographie, cryptage à clef publique RSA, PKI, chiffrement, déchiffrement, hachage, vérification d'identité, certificat numérique.

ملخص :

PKI أو الـ ICP هو نضام تشفير بني على التشفير بالمفاتيح العامة, الذي ينشأ زوج من المفاتيح " العامة و الخاصة " التي تستعمل في تشفير و فك تشفير الرسائل و المعلومات التي تتداول بين هيئتين, بمساعدة دالة Hache تعمل على التأكد من هوية المرسل عن طريق مراجعة إمضاء الوثيقة الإلكترونية التي ترفق مع المعلومات المرسلة. الكثير من لغات البرمجة تسمح لنا بتجسيد نموذج للـ PKI مثل لغة البرمجة OpenSSL و JAVA و C++, في النموذج الذي جسدناه اخترنا الـ C كلغة برمجة و الـ RSA كخوارزمية لإنشاء زوج المفاتيح " العامة و الخاصة " , مع نموذج لوثيقة إلكترونية و خوارزمية إمضاء MD5 .

أصبح من المهم جدا إدماج الـ PKI في مختلف أنظمة الاتصال , لأنه يضمن مراجعة هوية الهيئات التي تتبادل الرسائل و المعلومات على الانترنت و يستعمل أيضا في تأمين التبادلات البنكية الإلكترونية و في حماية الأجهزة الذكية و في جميع أنظمة التواصل الإلكترونية و كل هذا يعمل أوتوماتيكية و في وقت قصير جدا.

كلمات مفتاحية : تشفير, التشفير بمفتاح عمومي , RSA, PKI , تشفير, فك التشفير, hachage , مراجعة الهوية, وثيقة إلكترونية .

I. INTRODUCTION A LA CRYPTOGRAPHIE

Résumé

Dans ce chapitre on a introduit la cryptographie en générale et ces services et ces rôles important dans notre vie quotidienne et on a introduit aussi ces deux catégories, le cryptage et la cryptanalyse de chaque méthode de cryptage qui apparut depuis la préhistoire, la cryptographie à base de décalage des lettres de Jule César et le cryptage symétrique et asymétrique.

Comme on a expliqué dans ce chapitre le cryptage de César étais baser sur le décalage des lettres avec où les maths s'introduit pas, et ce modèle comme tous les autres modèles de cryptage à décalage étais faible devant l'analyse fréquentielle. On a vue on suit le cryptage moderne qui est diviser en deux grandes partie, le cryptage à clé secrete et cryptage a clé publique, où dans le cryptage à clé secrète on va voir les deux grand méthodes le DES et le AES par détaille, et dans le cryptage a clé publique on a expliquer le RSA comme puisque le PKI est baser sur ce dernier.

Mots-clés : Chiffrement, César, cryptage asymétrique, cryptage symétrique, AES, DES, RSA.

I.1 INTRODUCTION

La cryptographie est l'art du cryptage. Crypter un message, on dit aussi chiffrer un message, consiste à le transformer selon un procédé recélant un certain secret en un message inintelligible appelé cryptogramme. La connaissance du procédé inverse de décryptage ou de déchiffrement permet alors de retrouver le message d'origine à partir du cryptogramme. Le schéma suivant illustre le principe de chiffrement et de déchiffrement.

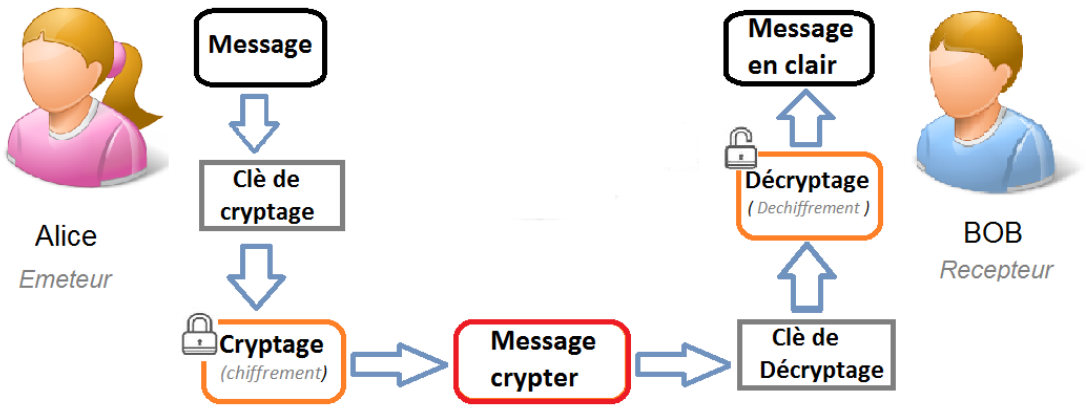

Figure 1 Schéma de base de processus de cryptage

L'algorithme de cryptage souvent connu, cela permet une plus grande souplesse. Le secret réside alors le plus souvent en l'utilisation d'une clé secrète qui apparait comme un paramètre spécial de l'algorithme.

Le cryptage ne sert pas seulement à rendre secret un message, il permet également d'authentifier ou signer un message, crypté ou non. Ce qui s'avère très utile dans une société de plus en plus informatisée comme la nôtre.

La Cryptographie est aujourd'hui essentielle pour le développement du Commerce électronique, des cartes à puce, de la téléphonie mobile, et particulièrement cruciale dans le secteur bancaire, elle est devenue une discipline qui concerne un public de plus en plus important.

I.2 CRYPTOGRAPHIE ET SERVICES DE SECURITE

La cryptographie permet d'assurer les services de sécurité suivant :

a. Confidentialité des données : seule le destinataire peut prendre connaissance du message
b. Intégrité des données : le message ne peut pas être falsifié sans qu'on s'en aperçoive
c. Authentification : 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 clef de déchiffrement. Le receveur est sûr de l'identité de l'émetteur grâce à une signature
d. Non-répudiation : l'émetteur ne peut nier avoir écrit le message (non-répudiation d'origine) et le receveur ne peut nier avoir reçu le message (non-répudiation de réception). On distingue aussi la non-répudiation de transmission dont l'émetteur du message ne peut nier avoir envoyé le message.

I.3 CLASSIFICATION

On distingue la cryptographie classique et la cryptographie moderne.

3.1 CRYPTOGRAPHIE CLASSIQUE

La cryptographie classique se base d'une manière absolue sur la complexité du langage naturel et sur le secret du code utilisé.

Il y a historiquement deux grandes familles de codes classiques avec des hybrides :

- Les codes à répertoire
- Les codes à clefs secrètes qui se subdivisent en deux familles :

- Les codes de substitution qui peuvent être des codes par blocs ou par flots
- Les codes de transposition ou de permutation qui sont des codes par blocs.

a. Codes à répertoire

Ces codes consistent en un dictionnaire qui permet de remplacer certains mots par des mots différents.

On peut par exemple créer le dictionnaire suivant :

Rendez-vous : 2000

Dimanche : étudiants

Midi : inscrits

Place : au centre universitaire

Des Martyrs : Ain Témouchent

La phrase en claire :

Rendez-vous Dimanche midi place des martyrs

Devient avec ce code :

2000 étudiants inscrits au centre universitaire Ain Témouchent

Ces codes présentes diverses inconvénients, par exemple l'ajout de nouveaux mots fait accroitre la taille du dictionnaire démesurément et toute modification dans le code nécessite l'envoi du dictionnaire ce qui augmente le risque d'interception du code.

b. Codes de substitutions

Codage : Dans les codes de substitution par flots ou par blocs les lettres sont remplacées par des symboles d'un nouvel alphabet suivant un algorithme précis.

Exemple : code de César

Pour coder on remplace chaque lettre par son rang dans l'alphabet.

A=0, B=1, C=2,....,M=12, N=13,...,S=19,...,X=23, Y=24, Z=25

Lettre codée=lettre claire+3 modulo 26

Le message en clair :

RENDEZ-VOUS DIMANCHE MIDI PLACE DES MARTYRS

Devient :

UHQGHC YRXV GLPDQFKH PLGL SODFH GHV PDUWBUV

On peut considérer toute la famille des codes :

Lettre codée=lettre claire +n modulo 26 où n est un entier entre 0 et 25 appelé la clef du code.

Décodage :

Le décodage se fait en utilisant la relation :

Lettre claire=lettre codée -n mod 26

On a affaire à un code en continu ou par flots symétrique ou à clef secrète.

c. Codes de permutation ou de transposition

Codage : Dans les codes de permutation On partage le texte en blocs, on garde le même alphabet mais on change la place des lettres à l'intérieur d'un bloc (on les permute).

Un exemple historique dont le principe est encore utilisé est la méthode de la grille. On veut envoyer le message suivant :

RENDEZ-VOUS DIMANCHE MIDI PLACE DES MARTYRS

L'expéditeur et le destinataire du message se mettent d'accord sur une grille de largeur fixée à l'avance (ici une grille de 5 cases de large).

L'expéditeur écrit le message dans la grille en remplaçant les espaces entre les mots par le symbole #. Il obtient :

Tableau 1 Exemple de codage de message à l'aide de table de transposition

R	E	N	D	E
V	**#**	**V**	**O**	**U**
S	**#**	**D**	**I**	**M**
A	**N**	**C**	**H**	**E**
#	**M**	**I**	**D**	**I**
#	**P**	**L**	**A**	**C**
E	**#**	**D**	**E**	**S**
#	**M**	**A**	**R**	**T**
Y	**R**	**S**	**#**	**#**

Il lit le texte en colonne et obtient ainsi le message crypté :

RVSA##E#YE##NMP#MRNVDCILDASDOIHDAER#EUMEICST#

Pour augmenter la sécurité du code sans toucher à son principe les deux interlocuteurs peuvent décider l'ajout d'une clef. Le but est de pouvoir changer facilement le cryptage d'un message tout en gardant le même algorithme de codage. Pour cela on rajoute une clé secrète constituée par l'ordre de lecture des colonnes.

Exemple : clé=ordre

On numérote les colonnes en fonction du rang des lettres du mot ORDRE dans l'alphabet c'est-à-dire : 3, 4, 1, 5,2 et on lit les colonnes dans l'ordre indiqué.

Tableau 2 Exemple de codage de message à l'aide de table de transposition en augmentant la sécurité

O	R	D	R	E
3	4	1	5	2
R	**E**	**N**	**D**	**E**
V	#	**V**	**O**	**U**
S	#	**D**	**I**	**M**
A	**N**	**C**	**H**	**E**
#	**M**	**I**	**D**	**I**
#	**P**	**L**	**A**	**C**
E	#	**D**	**E**	**S**
#	**M**	**A**	**R**	**T**
Y	**R**	**S**	#	#

NVDCILDASEUMEICST#RVSA##E#YE##NMP#MRDOIHDAER#

On a 5! Codes différents.

Décodage : Le décodage consiste à ranger les messages chiffrés en colonne sur la grille en suivant l'ordre des colonnes donné par la clé. Pour éviter d'allonger démesurément la hauteur de la grille et pour éviter d'avoir à coder la totalité du message avant de commencer la transmission, on travaille sur des blocs de taille m = k × l où k est la largeur de la grille et l sa hauteur. On a alors un système de codage par blocs, symétrique ou à clef secrète.

d. Code de Vigénère

Le principe est le suivant :

- On se fixe une longueur de bloc m.
- On découpe le message en blocs de m-lettres.
- On chiffre par blocs de m lettres. On décide par exemple que la première lettre d'un bloc de m est codée avec un code de César de clef n1, la deuxième avec un code de César de clef n2 et la m-ième par un code de César de clef nm.

Le code de César est un code mono-alphabétique. Le code de Vigénère est un code poly-alphabétique ou par blocs.

Exemple :

M=5, n1=2, n2=3, n3=1, n4=5, n5=7

Message en claire : RENDEZ-VOUS DIMANCHE MIDI PLACE DES MARTYRS

On partage en blocs de taille 5 en partant de la gauche

RENDE ZVOUS DIMAN CHEMI DIPLA CEDES MARTY RS###

Les trois # sont ajoutées pour compléter le dernier bloc.

La manière de compléter le dernier bloc peut être une faiblesse du code. Dans chaque bloc on code la première lettre avec le code de César de clef n1 = 2,..., la cinquième lettre du bloc avec le code de César de clef n5 = 7.

Le message codé devient :

THOIL BYPZZ FLNFU EKFRP FLQQH EHEJZ ODSYF TV###

Pour le décodage on fait l'inverse ;

Lettre clair(i)=lettre codée(i)-clé (i) mod 26

e. Chiffrement de Hill

Ce crypto-système généralise celui de Vigénère. Il a été publié par L. S. Hill en 1929.

Codage : On choisit un alphabet de n lettres (on prendra dans nos exemples n = 26) et une taille m pour les blocs. Chaque caractère est d'abord codé par un nombre compris entre 0 et n -1 (exemple son rang dans l'alphabet diminué de 1). Les caractères sont alors regroupés par blocs de m caractères formant un certain nombre de vecteurs X=(x1, x2, …, xm). Les nombres xi étant compris entre 0 et n - 1, on peut les considérer comme des éléments de Z/nZ et X est alors un élément de (Z/nZ)m.

On construit alors une matrice m × m d'éléments de Z/nZ : A.

Le bloc X est alors chiffré par le bloc Y = AX, le produit s'effectuant modulo n.

Décodage

$X=A^{-1}Y$ avec A^{-1} est la matrice inverse de A, le produit s'effectuant modulo n.

3.2 CRYPTOGRAPHIE MODERNE

Dans la cryptographie moderne, on distingue les codes de chiffrement symétriques ou à clef secrète tel que DES et AES et les codes de chiffrement asymétriques ou codes à clef publique tel que RSA qui fait l'objet de notre projet.

a. CODE A CLEF SECRETE

a.1 Code DES (Data Encryption Standard) : DES est un système cryptographique basé sur le schéma de Feistel. Il répète 16 fois un algorithme, appelé fonction d'étage qui dépend

d'un paramètre, la clef d'étage. La répétition de cet algorithme mélange les bits du message en clair en respectant les principes de C. Shannon, confusion et diffusion.

La confusion gomme les relations entre le texte clair et le texte chiffré pour éviter les attaques par analyse statistique. Autrement dit un changement d'un seul bit dans le texte clair doit affecter un grand nombre de bits (idéalement tous) du texte chiffré.

La diffusion disperse la redondance du texte clair dans le texte chiffré, par exemple deux lettres doublées ne doivent pas rester côte à côte après cryptage.

- Schéma de Feistel

Le schéma de Feistel est un algorithme de chiffrement itératif agissant sur des demi-blocs de message. Une fonction de ronde suivant un schéma de Feistel est comme suit :

Soit X un bloc de longueur 2×l, le bloc sera découpé en deux sous blocs G et D de longueur l

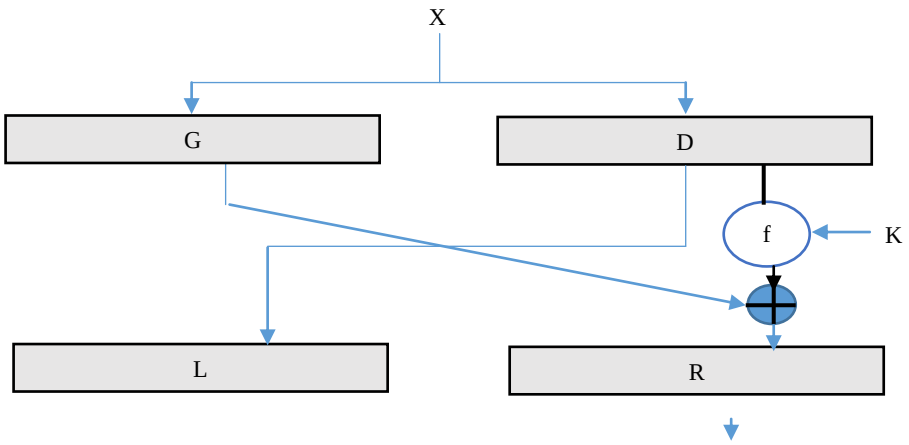

Figure 2 Un tour du schéma de Feistel

L'image du bloc (G, D) est le bloc (L, R) tel que :

L=D

R= G⊕ f_K(D)

Cette transformation est bijective, car si on a un tel couple (L, D), on peut facilement reconstruire le couple (G, D).

D=L

G=R⊕f_K(L)

La figure suivante illustre le principe de chiffrement/déchiffrement de Feistel :

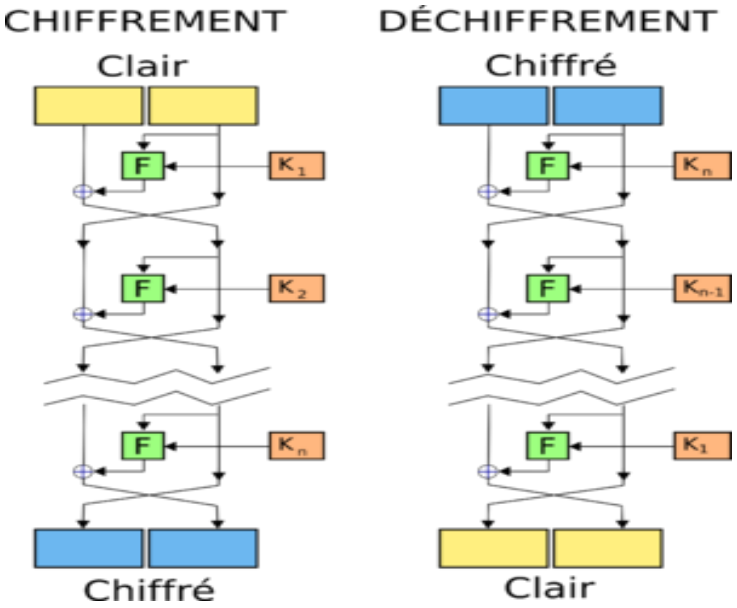

Figure 3 Chiffrement et déchiffrement avec le schéma de Feistel

- **Aspect technique du DES**

  - DES utilise des clefs de longueur 64 bits dont seulement 56 bits utiles. Les autres bits (8, 16, 24, 32, 40, 56, 64) sont utilisés pour tester la parité.
  - DES agit sur des blocs de 64 bits
  - DES est un système de chiffrement itéré à 16 étages.

a.2 Code AES (Advanced Encryption Standard)

Ces aspects techniques sont :

- AES est un système de chiffrement itéré opérant sur des blocs de 128 bits
- On notera N_e le nombre d'étage
- La fonction d'étage 'g' est répétée N_e fois avec N_e=10, 12 ou 14 et d'une clef secrète de 128 bits, 192 bits ou 256 bits
- Un algorithme de diversification de clefs, EXPANDKEY[i] permet de créer une clef pour chaque étage i à partir de la clef secrète K
- La fonction d'étage 'g' est l'ensemble des opérations que l'on fait subir au texte qui arrive à l'étage i. Elle est la même pour tous les étages sauf le dernier
- La fonction 'g' consiste en l'application successive de 4 opérations :
  - SUBBYTES
  - SHIFTROWS
  - MIXCOLUMNS
  - ADDROUNDKEY

Pour le dernier étage, on applique seulement les opérations :

- SUBBYTES
- SHIFTROWS
- ADDROUNDKEY

A partir de la clef secrète K, on génère des sous clefs d'étages par l'algorithme de diversification de clef, EXPANDKEY[i]

On obtient les clefs, K^1, K^2, …, K^{Ne}

- Schéma de codage AES (N_e=10)

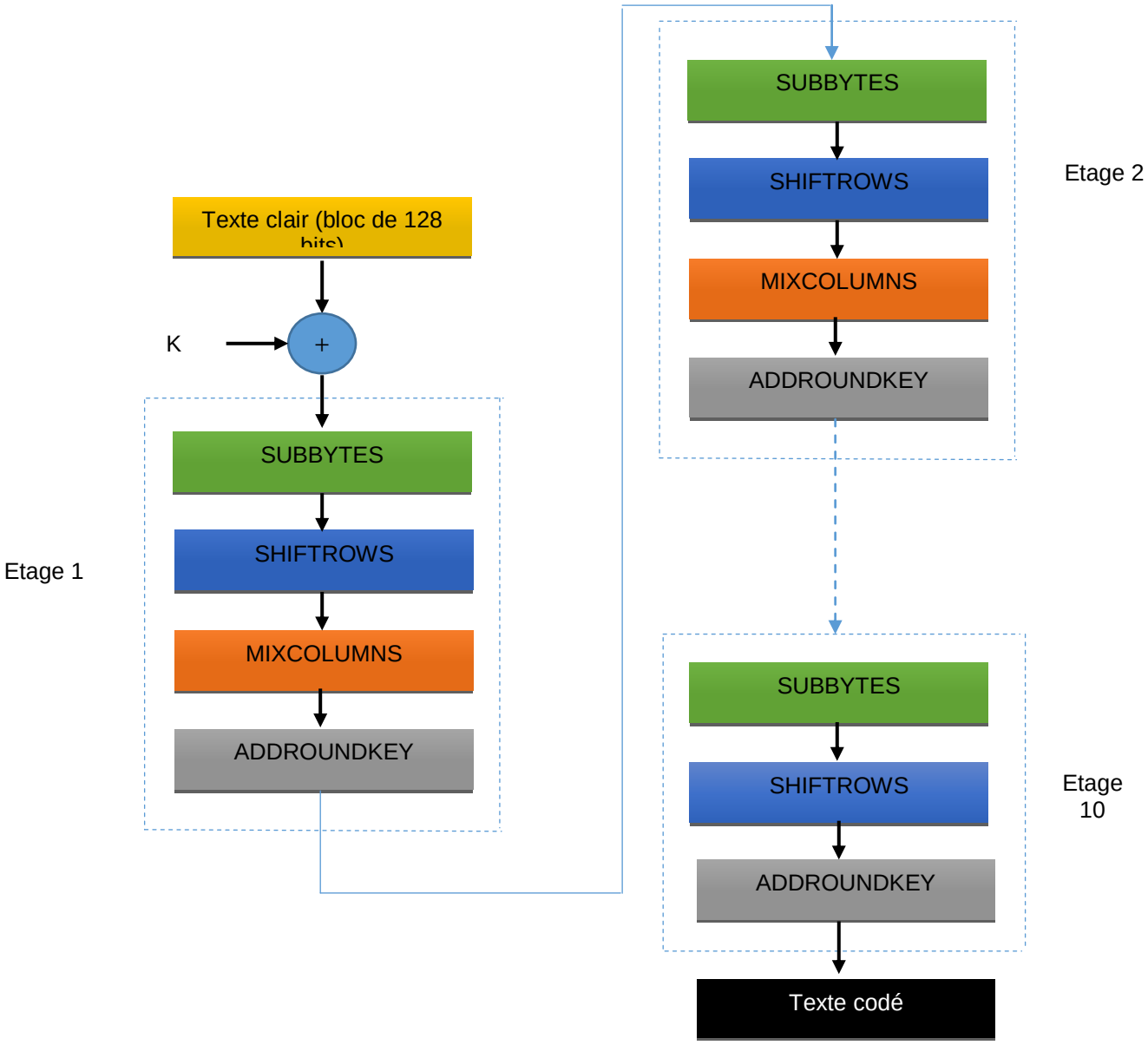

Figure 4 Schéma de codage AES (N_e=10)

- Schéma de décodage AES (Ne=10)

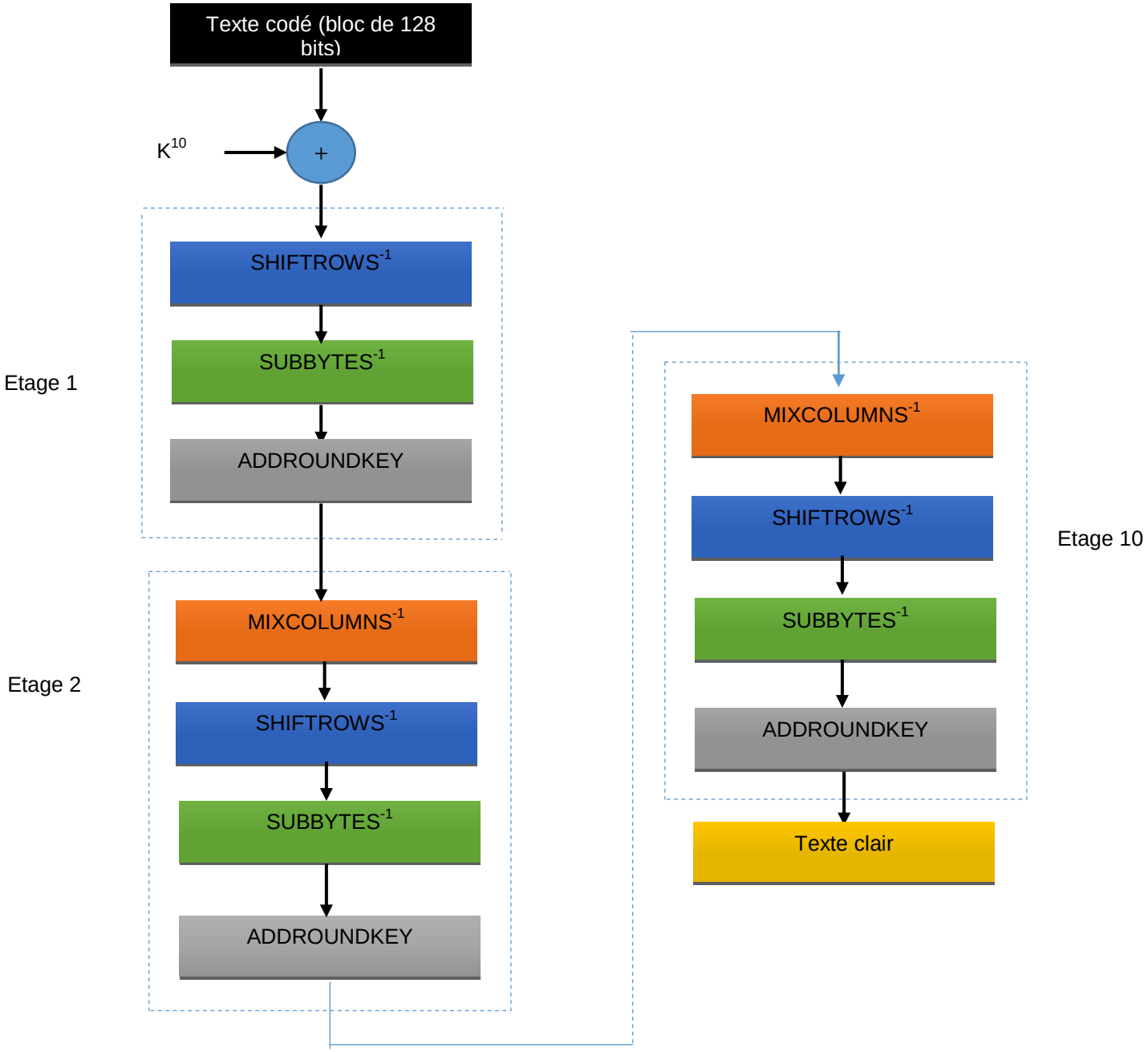

Figure 5 Schéma de décodage AES (Ne=10)

Description de l'algorithme de codage

- On prend le bloc de 128 bits que l'on considère comme une suite de 16 octets que l'on range dans une matrice 4×4.
- On note STATE cette écriture matricielle et qui représente aussi le flux d'entrée-sortie de chaque étage.

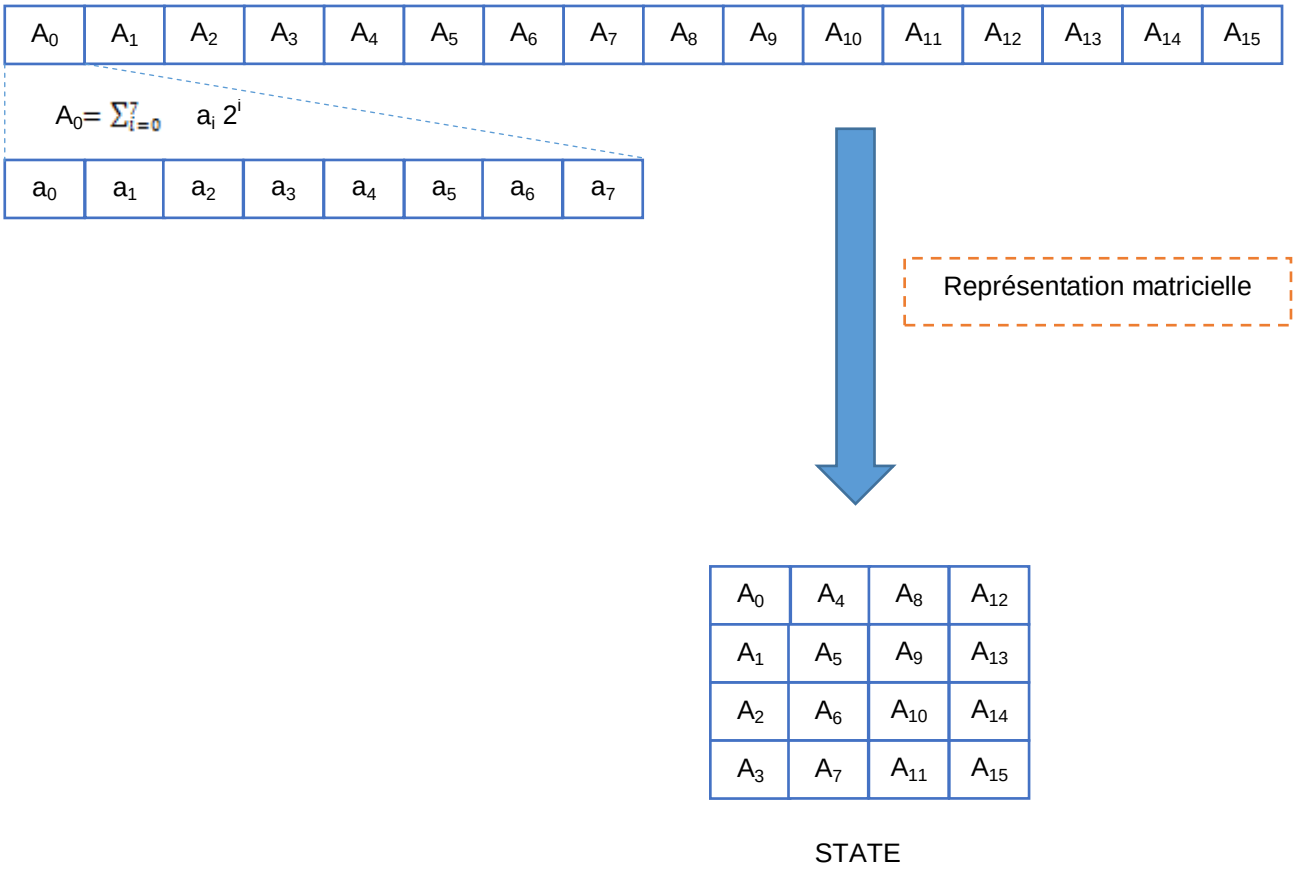

Figure 6 Schéma explicatif de matrice de STATE

L'algorithme se déroule de la manière suivante :

1- Calculer STATE=STATE⊕K

2- Pour i de 1 à N_e-1 effectuer sur STATE les opérations suivantes :

a- Opération de substitution SUBBYTES

Cette opération agit sur chaque octet et utilise la table de substitution S-Box ci-dessous :

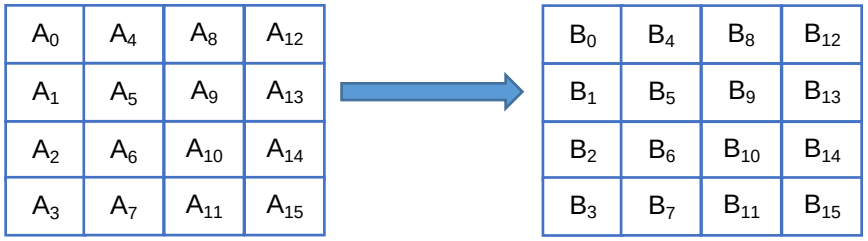

Figure 7 Opération de substitution SUBBYTES

a- Opération SHIFTROWS : sur le résultat de SUBBYTES on effectue une permutation notée SHIFTROWS (permutation circulaire sur les éléments des lignes de la matrice des octets B_{ij})

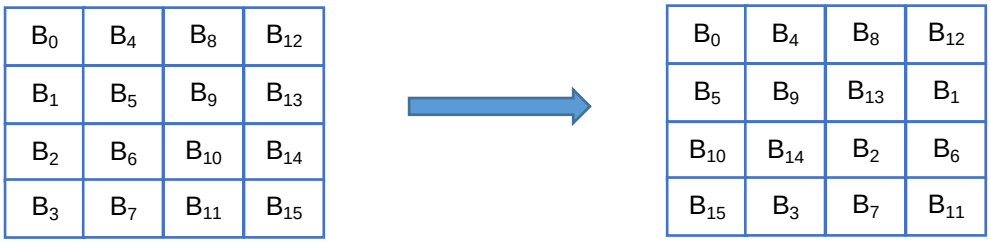

Figure 8 Opération SHIFTROWS

Remarque 2: le même processus est appliqué lors du déchiffrement mais avec un décalage dans le sens inverse.

b- Opération MIXCOLUMNS : Cette opération consiste à réaliser pour chaque colonne de la matrice STATE résultant de l'opération SHIFTROWS le produit matriciel suivant :

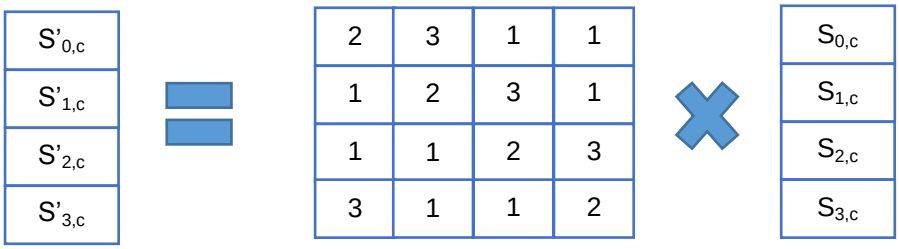

Figure 9 Opération MIXCOLUMNS

$S_{0,c}$ représente l'octet de la ligne 0 et la colonne c de la matrice STATE

Remarque 3: l'opération MIXCOLUMNS assure la diffusion. En effet, une différence sur un octet d'entrée se propage sur les quatre octets de sortie.

Remarque 4: lors du déchiffrement, on utilise le produit matriciel suivant :

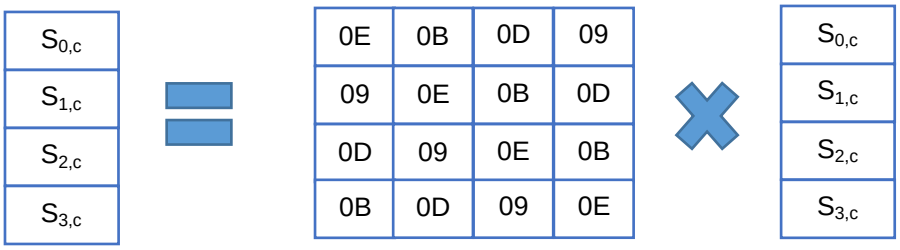

Figure 10 Opération SHIFTROWS (Déchiffrement)

c- Opération ADDROUNDKEYS[i]

Cette opération consiste en un XOR de la matrice STATE résultant de l'opération MIXCOLUMNS et de la clef du tour.

Expansion de clefs

- L'AES utilise la clef de chiffrement K et la soumet à une routine d'expansion
- Cette expansion génère $4\times(N_e+1)$ mots
- Le résultat de cette expansion consiste dans un tableau linéaire noté [Wi] avec i variant de 0 à $4\times(N_e+1)-1$

Contenu du [Wi]

- Les N_k (k=4, 6 ou 8) premiers mots (mot=32 bits) [w0, …, WN_{k-1}] contient la clef K
- Les mots suivant sont calculés en faisant un XOR du mot précédent [Wi-1] et du mot situé N_k positions avant [WN_{k-1}]
- Pour les mots situés sur une position multiple de N_K, une transformation est appliquée à [Wi-1] avant le XOR
- Cette transformation consiste en une permutation cyclique nommée '' ROTWORD'' d'un cran vers la gauche suivie d'une application de la S-Box ''SUBWORD'' séparément sur chaque octet du mot puis d'un XOR avec un vecteur dépendant du tour noté Rcon (i/N_K)
- Rcon (i/N_K) contient la valeur [X^{i-1}, 00, 00, 00]

Rcon(i) sont définies par :

Rcon (i) = X^{i-1} mod $X^8+X^4+X^3+X+1$

Rcon (1)=X^0=01

Rcon (2)=X=02

Rcon (3)=X^2=04

Rcon(4)=X^3=08

Rcon(5)=X^4=10

Rcon(6)=X^5=20

Rcon(7)=X^6=40

Rcon(8)=X^7=80

Rcon(9)=X^8 mod $X^8+X^4+X^3+X+1=X^4+X^3+X+1$=1B

Roc(10)= X^9 mod $X^8+X^4+X^3+X+1=X^5+X^4+X^2+X$=36

…

Remarque 5 : $F_{256}=F_2[X]/ X^8+X^4+X^3+X+1$

Remarque 6: Il est important de noter que l'expansion dans le cas d'une clef de 256 bits est différente des deux autres cas (clefs de 128 bits et 192 bits). En effet, dans ce cas si le rang i-4 est un multiple de 8 la fonction SUBWORD est encore appliquée à Wi-1 avant le XOR.

Exemple : N_k=6 → clef de 192 bits

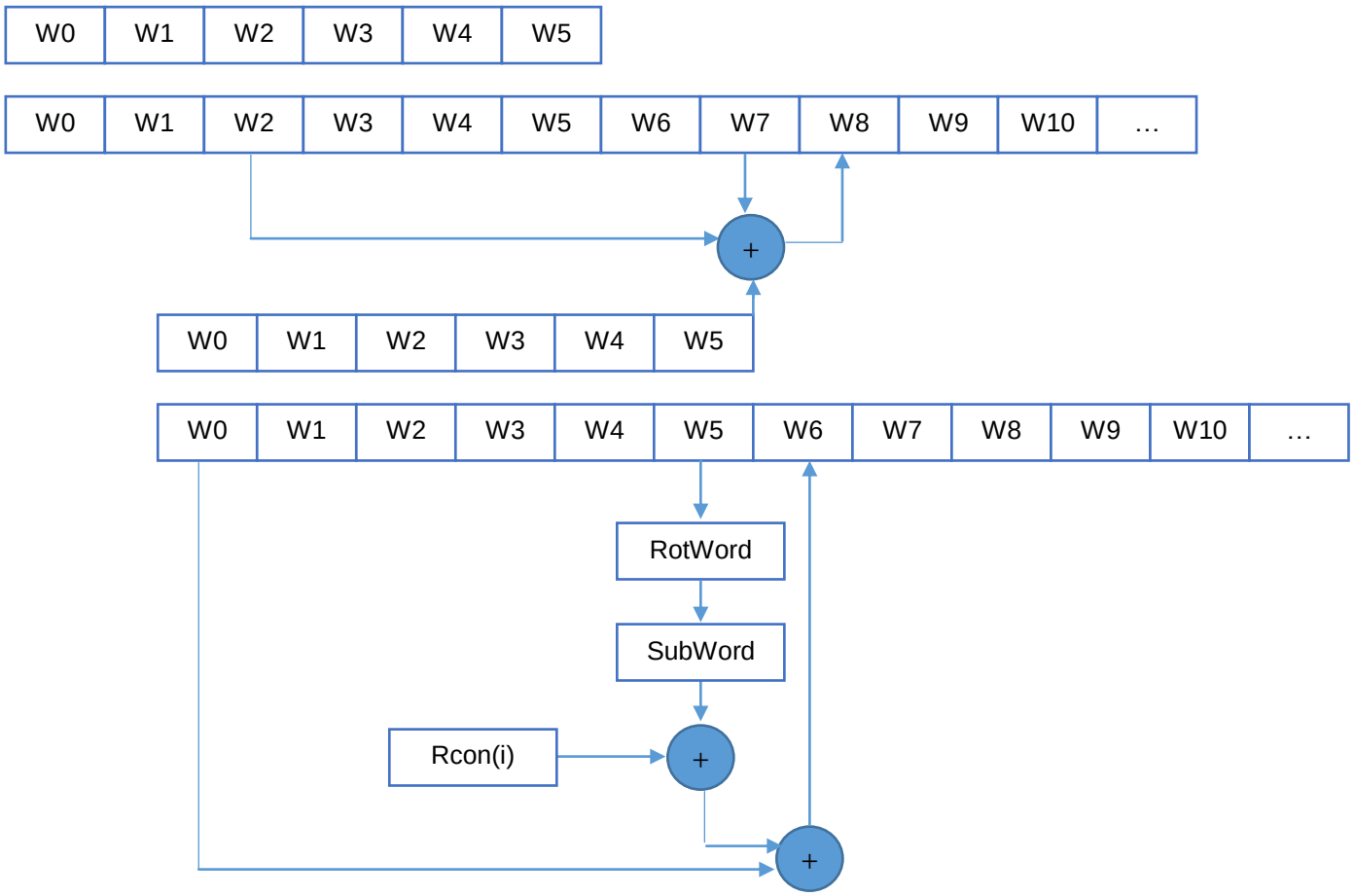

Figure 11 Schéma explicatif de Contenu du [Wi]

b. CODE A CLEF PUBLIQUE

Dans cette partie, on va aborder uniquement le code à clef publique RSA. Cet algorithme a été décrit en 1977 par Ronald Rivest, Adi Shamir et Leonard Adleman.

b.1 Mise en œuvre de RSA

On considère deux personnes appelées traditionnellement Alice et Bob qui veulent sécuriser leurs communications.

- Bob possède un message confidentiel qu'il souhaite transmettre à Alice.
- Alice construit deux clés :
  - une clé de chiffrement publique qu'elle transmet à Bob.
  - une clé de déchiffrement privée qu'elle conserve soigneusement.
- Bob utilise la clé publique pour chiffrer le message, et le transmets à Alice.
- Alice utilise la clé privée pour déchiffrer le message reçu.

Génération de clefs

- Soient deux grands nombres premiers « aléatoirement » choisis : p et q.
- Notons : $n = p*q$ et $\varphi = (p-1)*(q-1)$
- Soient d un grand entier « aléatoirement » choisi, premier avec φ. Et e l'inverse de d modulo φ.
- La clé publique de chiffrement est le couple (n,e), la clé privée de déchiffrement est le couple (n,d).

Chiffrement

- Avant d'être chiffré, le message original doit être décomposé en une série d'entiers M de valeurs comprises entre 0 et $n-1$.
- Pour chaque entier M il faut calculer $C \equiv M^e [n]$.
- Le message chiffré est constitué de la succession des entiers C.

Déchiffrement

- Conformément à la manière dont il a été chiffré, le message reçu doit être composé d'une succession d'entiers C de valeurs comprises entre 0 et $n-1$.
- Pour chaque entier C il faut calculer $M \equiv C^d [n]$.
- Le message original peut alors être reconstitué à partir de la série d'entiers M.

b.2 Fiabilité de RSA

La sécurité de l'algorithme RSA repose sur la difficulté à factoriser *n*. Pour décrypter le message, il est nécessaire de trouver *d* connaissant *e*, ce qui nécessite de recalculer *φ*, et donc de connaître *p* et *q*, les deux facteurs premiers de *n*. Or, la factorisation d'un entier (de très grande taille) en facteurs premiers est extrêmement difficile, cette opération nécessitant une capacité de calcul très importante. Pour exemple : en 2010, l'INRIA et ses partenaires ont réussi à factoriser une clé de 768 bits. Il leur a fallu deux ans et demi de recherche, et plus de 10^20 calculs. C'est à ce jour le meilleur résultat connu de factorisation.

Afin de se prémunir contre les puissances de calculs grandissantes, il est régulièrement recommandé d'utiliser des tailles de clés de plus en plus grandes (actuellement de 2048 bits).

I.4 Conclusion

Dans ce chapitre, nous avons présenté les méthodes de chiffrement classiques et modernes. Les méthodes classiques manipulent des caractères tandis que les méthodes modernes manipulent des bits (série de 0 et 1). La sécurité des méthodes classiques se base sur le secret de l'algorithme. Ces méthodes ne sont plus utilisées maintenant. Les méthodes de chiffrement modernes reposent sur le même principe de chiffrement que les méthodes classiques, elles effectuent des substitutions et transpositions sur les bits du message en clair pour produire le message chiffré. La sécurité de ces méthodes repose sur le secret de la clef de chiffrement, l'algorithme étant publique (connu de tout le monde).

Dans la cryptographie moderne, on distingue les méthodes de chiffrement à clefs secrètes ou symétriques qui utilisent la même clef pour chiffrer et déchiffrer et les méthodes de chiffrement à clefs publiques ou asymétriques qui utilisent une paire de clefs, privée et publique. La clef publique sert à chiffrer un message en clair, tandis que la clef privée est utilisée pour déchiffrer les messages chiffrés avec la clef publique.

II. *LA SIGNATURE ELECTRONIQUE*

Résumé

Dans ce chapitre on a expliqué la signature numérique (électronique), ces caractéristiques essentielles, les différents types de signature, qui sont basés sur la cryptographie asymétrique. Les conditions de validité de la signature électronique, la création des certificats électronique et l'introduction de signature numérique dans le certificat électronique , l'intégrité des document et la vérification des signature et le chiffrement intégrer dans cette opération.

La signature électronique est l'élément de base dans la génération des certificats électroniques, c'est l'élément qui aide dans la vérification de l'identité des personne qu'on est en train de communiquer avec, et dans notre temps présent la vérification ce fait automatiquement.

Mots-clés : signature numérique, certificat électronique, chiffrement, authentification, autorité de certificat, confiance, hachage, fonction d'hachage**.**

II.1 Introduction

La signature électronique est une application très concrète de la cryptographie asymétrique qui fut inventée dans le milieu des années 70. Elle va permettre à chacun de signer un acte électronique avec une valeur juridique identique à celle que l'on accorde actuellement à la version papier de cet acte. Elle doit garder une marque de la volonté de s'engager sur ce que l'on signe. A ce titre cet outil va prendre une place importante dans la vie quotidienne des responsables d'entreprises et à terme des citoyens.

Pour le spécialiste de la sécurité ceci permet d'assurer la traçabilité des actes effectués et l'adaptabilité des actions. Pour le spécialiste des questions relatives à la vie privée ceci peut également devenir une menace si la signature électronique est utilisée comme un moyen de surveillance des individus par pistage de leurs actions.

La signature électronique n'a pas d'autre finalité que celle de la traditionnelle signature manuscrite, à savoir identifier le signataire d'un acte et manifester son consentement aux obligations qui découlent de cet acte. Le reste est l'application des techniques cryptographiques à d'autres besoins de sécurité.

II.2 Définition

Le terme 'signature électronique est une dénomination générique désignant des éléments de niveaux de sécurité variables. Ainsi, la simple signature électronique a une définition très large qui peut par exemple s'appliquer à une signature manuscrite faxée. Par contre, la signature

électronique avancée est déjà plus restrictive et plus sécurisée. Elle doit ainsi répondre aux quatre exigences suivantes :

- être liée uniquement au signataire.
- permettre l'identification de celui-ci.
- être créée par des moyens que le signataire puisse garder sous son contrôle exclusif.
- être liée aux données auxquelles elle se rapporte (garantie d'intégrité).

Les signatures électroniques avancées peuvent notamment avoir la forme de signatures numériques fondées sur la cryptographie asymétrique. Il s'agit alors de la signature à clé publique (PKI) composée de deux clés : l'une privée, détenue par le seul signataire, et l'autre publique, accessible à tous *via* l'annuaire du prestataire de service de certification (CA).

La signature électronique qualifiée est un autre type de signature avancée à clés mais présentant un niveau élevé de sécurité et ayant force probante (recevabilité en justice).

Si elle n'est pas qualifiée, une signature électronique ne pourra néanmoins pas être refusée (d'un point de vue juridique) au seul motif que :

- la signature se présente sous forme électronique, ou
- qu'elle ne repose pas sur un certificat qualifié, ou
- que ce certificat qualifié n'est pas délivré par un prestataire accrédité de service de certification, ou
- qu'elle n'est pas créée par un dispositif sécurisé de création de signature.

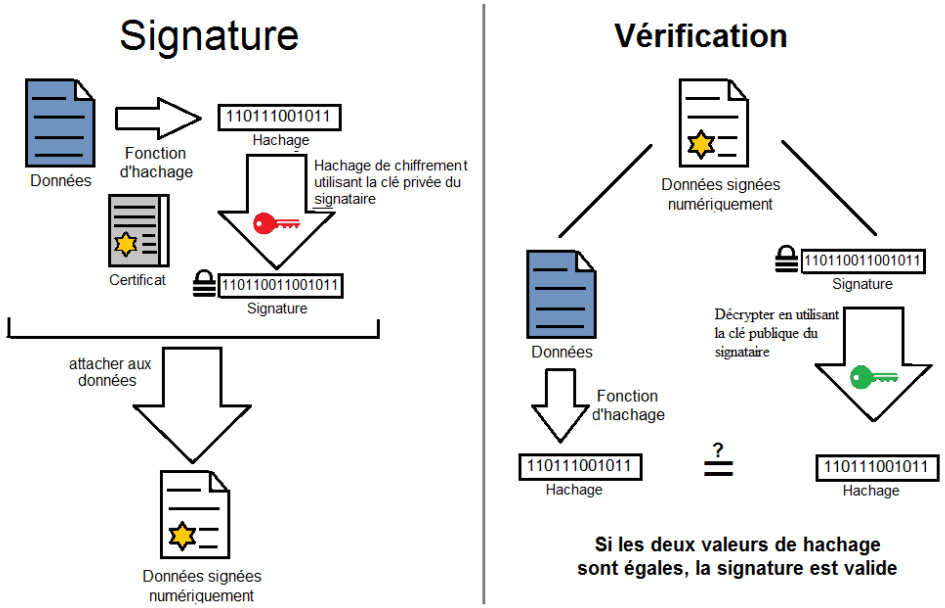

Figure 1 Signature et vérification de signature de certificats electroique

II.3 Les caractéristiques essentielles de la signature électronique

3.1 Singularité : Elle doit singulariser l'objet, c'est-à-dire ôter tout risque de confusion avec un objet, même similaire. Selon ce critère, la signature_manuscrite pour les personnes, bien que supérieure à la simple écriture du nom, demeure plus faillible que les signatures biométriques.

3.2 Produit dérivé : Elle doit être issue de, ou engendré par cet objet. En cela, la signature se distingue de l'identifiant ; ce dernier est souvent attribué arbitrairement, et s'obtient par l'intermédiaire d'une table de correspondance. Par exemple, les signatures biométriques ou manuscrites sont des produits d'une personne, le numéro de sécurité sociale est une forme d'identifiant attribué.

3.3 Invariance : L'objet doit produire la même signature quel que soit son état, sa représentation, ou son apparence. Distinguons plusieurs cas :

- Pour une personne, cela signifie que sa signature doit rester la même quel que soit son âge, son humeur ou son état physique. Ainsi, le code génétique est une excellente signature car elle seule demeure de la conception et par-delà la mort, et résiste à l'amputation ou toute autre altération de la personne.

II.4 Caractéristiques non essentielles d'une bonne signature électronique

4.1 Concision : La signature doit être aussi courte que possible ; en particulier la comparaison des signatures doit être plus simple que celle des objets eux-mêmes. Selon ce critère, la séquence complète du génome est une signature très médiocre de la personne. En réalité, seuls d'infimes fragments d'ADN servent de signature (appelée empreinte génétique).

4.2 Production aisée : Générer la signature de l'objet doit être une opération la moins coûteuse possible. Ce point est plus particulièrement important en mathématiques notamment pour les graphes, nœuds, groupes, etc.

II.5 Les différentes signatures

5.1 Signature manuscrite :

La signature d'une personne se présente généralement comme une forme personnalisée de l'écriture à la main .Cette forme peut être simplifiée, calligraphiée, dessinée de

diverses manières, et associée à des effets de style (traits, courbes, points, etc.) qui sont mis au point par l'individu pour personnaliser cette signature et la rendre à la fois unique.

Il existe maintenant des systèmes (machines à signer) permettant de reproduire mécaniquement la signature d'une personne. Ils sont notamment utilisés par les personnes qui doivent signer de grandes quantités de documents, comme par exemple des célébrités, des chefs d'états ou des dirigeants d'entreprises.

5.2 Signatures par cachet :

Plusieurs cultures utilisant des systèmes d'écriture non alphabétiques ne partagent pas la notion occidentale de la signature manuscrite : dans ces cultures, l'action de signer d'un nom ne diffère en rien de l'écriture normale. Le chinois, le japonais ou le coréen sont des exemples, ils existent des gens qui utilisent le graphe cachet à la place de Signature manuscrite.

5.3 Signature aveugle :

On appelle signature aveugle une signature effectuée sur un document qui a été masqué avant d'être signé.

II.6 Les conditions de validité de la signature électronique

La recevabilité en justice des signatures électroniques et la qualification de signature électronique " avancée ", reposent sur des conditions relatives :

- aux certificats.
- aux tiers de certification**.**
- au processus de création de la signature électronique.

6.1 Les certificats :

Le certificat est au cœur du processus de signature électronique. Il est porteur d'une valeur juridique puisqu'il va permettre l'identification de la personne, mais il a également une définition technique. Selon la définition qui en est donnée par l' « ISO », c'est « un objet informatique qui permet de lier de façon intangible une entité (une personne, une ressource) à certaines caractéristiques de cette entité ».

Le certificat est, ainsi, un message électronique par lequel un témoin privilégié, le certificateur, contrôle la concordance et l'adéquation entre l'identité du signataire et la clé publique.

On distingue le certificat électronique simple et le certificat qualifié. Le premier est un document qui se présente sous la forme électronique et qui atteste du lien entre les données de vérification de signature électronique et Série de critères un signataire. Le deuxième doit avoir été délivré par un prestataire capable de délivrer ce type de certificats et comporter certaines indications telles que :

- Une mention indiquant que ce certificat est délivré à titre de certificat électronique qualifié.
- L'identité du prestataire de services de certification électronique ainsi que l'état dans lequel il est établi.
-Le nom du signataire ou un pseudonyme, celui-ci devant alors être identifié comme tel.
-Le cas échéant, l'indication de la qualité du signataire en fonction de l'usage auquel le certificat est destiné.
-Les données de vérification de signature électronique qui correspondent aux données de création de signature électronique.
-L'indication du début et de la fin de la période de validité du certificat électronique.
-Le code d'identité du certificat électronique.
-La signature électronique sécurisée du prestataire de services de certification électronique qui délivre le certificat électronique.

Les certificats électroniques permettent de contrecarrer les tentatives de substitution de la clé d'une personne par une autre. Elle se compose de trois éléments :

• Une clé publique.
• Des informations sur le certificat. (Informations sur l'« identité » de l'utilisateur, Telles que son nom, son ID utilisateur, etc.)
• Une ou plusieurs signatures électroniques. La signature électronique d'un certificat permet de déclarer que ses informations Ont été attestées par une autre personne ou entité.

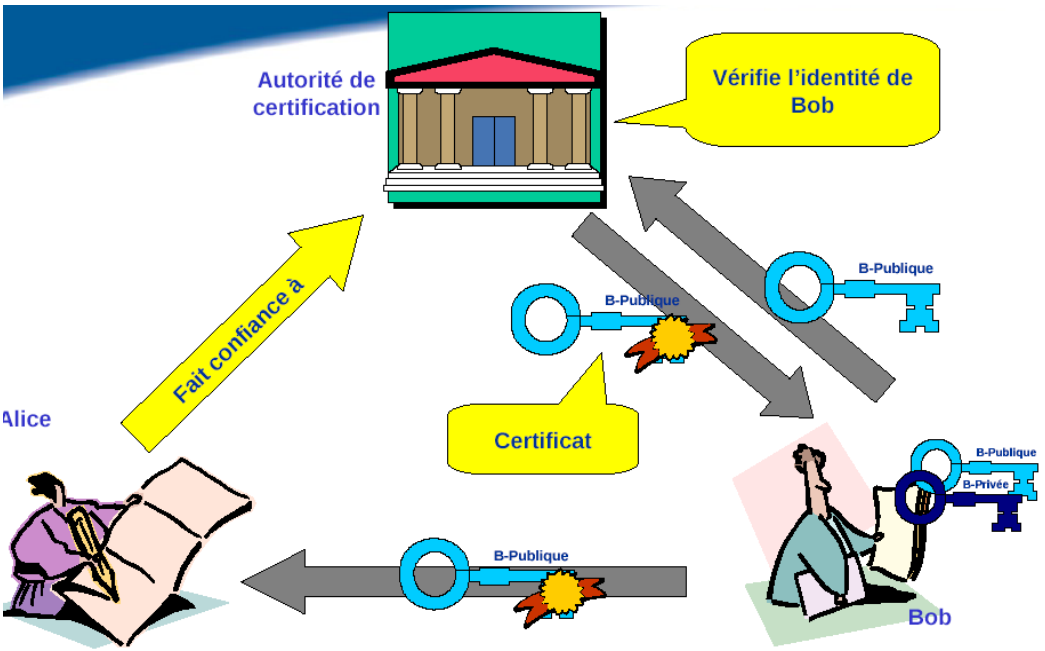

Figure 2 La création de certificats électronique

II.7 CREATION DE CERTAFICAT

7.1 Les tiers de certification :

Si la fourniture de services de certification ne peut être soumise à une autorisation préalable, et peut être assurée par toute personne physique ou morale, les Etats membres doivent cependant instaurer un système de contrôle des tiers de certification**.** Les prestataires de service de certification doivent :

- faire la preuve qu'ils sont suffisamment fiables pour fournir des services de Certification.
- assurer le fonctionnement d'un service d'annuaire rapide et sûr.
- veiller à ce que la date et l'heure d'émission et de révocation d'un certificat puissent être déterminées avec précision.
- vérifier, par des moyens appropriés et conformes au droit national, l'identité.
- employer des personnes ayant les connaissances spécifiques, l'expérience compétences et les Qualifications nécessaires à la fourniture des services.
- utiliser des systèmes et des produits fiables qui sont protégés contre les modifications et Qui

assurent la sécurité technique.

- enregistrer toutes les informations pertinentes concernant un certificat qualifié pendant le délai utile. Ces enregistrements peuvent être effectués par des moyens électroniques.
- ne pas stocker ni copier les données afférentes à la création de signature de la personne à laquelle le prestataire de service de certification a fourni des services de gestion de clés.
- avant d'établir une relation contractuelle avec une personne demandant un certificat à l'appui de sa signature électronique, informer cette personne par un moyen de communication durable des modalités et conditions précises d'utilisation des certificats.
- utiliser des systèmes fiables pour stocker les certificats sous une forme vérifiable, de sorte que :

- seules les personnes autorisées puissent introduire et modifier des données.
- L'information puisse être contrôlée quant à son authenticité.
- Les certificats ne soient disponibles au public pour des recherches que dans les cas où le titulaire du certificat a donné son consentement.

7.2 Le dispositif de création de la signature électronique

Il est nécessaire pour une signature électronique que soit assurée l'intégrité du document. En effet, il suffirait de modifier l'acte après apposition de la signature pour modifier la teneur de l'engagement contractuel. Les dispositifs sécurisés de création de signature doivent au moins garantir, par les moyens techniques :

- les données utilisées pour la création de la signature ne puissent, pratiquement, se rencontrer qu'une seule fois et que leur confidentialité soit raisonnablement assurée.
- les données utilisées pour la création de la signature ne puissent être trouvées par déduction et que la signature soit protégée contre toute falsification.
- les données utilisées pour la création de la signature puissent être protégées de manière fiable par le signataire légitime contre leur utilisation par d'autres.

Donc les dispositifs sécurisés de création de signature ne doivent pas modifier les données à signer ni empêcher que ces données soient soumises au signataire avant le processus de signature.

II.8 Nécessité de garantir l'intégrité du document

L'un des traits caractéristiques de la signature électronique réside en ce qu'elle fait l'objet d'une télétransmission. Or, pendant cette transmission, la signature peut être altérée, comme, d'ailleurs, le message lui-même. Cette altération peut être due aux conditions techniques ou à l'intervention de personnes mal intentionnées. Le message, à son arrivée, peut ne pas correspondre exactement à celui qui a été envoyé. Ce sont ces risques qui expliquent la nécessité d'une garantie, voulue par les utilisateurs, de l'intégrité des messages électroniques et donc de la signature, qui en est l'une des données.

II.9 Intégrité et vérification de la signature

Le terme « *intègre* » qualifie l'état d'un objet qui n'a pas été modifié, intentionnellement ou non, par rapport à un état antérieur. C'est, dans le cas de la signature électronique, la transmission qui pourra être à l'origine de la modification du fichier, ce qui explique la nécessité de contrôler le bon état du fichier à l'arrivée. Egalement, le destinataire pourrait être tenté de modifier la teneur du contrat par exemple, limiter son engagement. La signature électronique tend également à protéger le destinataire car elle va sceller l'engagement contractuel de l'expéditeur. En ce sens, la signature électronique se veut protectrice de toutes les parties au contrat.

L'intégrité recourt, en informatique, à l'utilisation de la technique. Elle sera mise en œuvre par le contrôle du condensé ou « *hash* ». En effet, le message à signer va tout d'abord être haché par un logiciel. De ce hachage va résulter un condensé[1], sorte de chaîne alphanumérique, qui sera le résultat du contenu même du message. Ainsi, à chaque message correspond un condensé numérique unique. Toute modification du message, jusqu'à la suppression d'une virgule, engendrerait un condensé différent. Ensuite, grâce à un dispositif de création de signature électronique, le condensé va pouvoir être chiffré par la clé privée de l'expéditeur. Il en résultera un cryptogramme. Ainsi, c'est pourquoi l'on emploie généralement le terme de **« *signature électronique* ».**

Lors de la réception du message, les données de la signature électronique devront être vérifiées. On utilisera, pour ce faire, un dispositif de vérification de la signature électronique, qui

permettra de s'assurer de l'identité du signataire (grâce au certificat) et de l'intégrité du message. Il va falloir ainsi défaire ce qui a été fait par le signataire.

Le cryptogramme sera déchiffré grâce à la clé publique de l'expéditeur, ce qui va permettre de retrouver le résumé du message ou « *hash* », garant de l'intégrité. Parallèlement à cela, le message sera haché par le destinataire. Il suffira alors de comparer les deux résumés : s'ils coïncident, la signature est validée, sous réserve de la validité du certificat et du non répudiation de la bi-clé.

Les systèmes de vérification de la signature électronique sont également soumis au décret du 30 Mars 2001. Ces dispositifs sont définis comme « un matériel ou un logiciel destiné à mettre en application les données de vérification de signature électronique ».

Pour vérifier la validité de la signature électronique, il faut vérifier les certificats :

- Vérifier la période de validité du certificat du signataire
- Vérifier que le certificat n'est pas contenu dans la liste de révocation ou de suspension du PSC
- Vérifier que le certificat du PSC qui a signé le certificat du signataire n'est pas contenu dans la liste de révocation.

Après avoir contrôlé la validité du certificat, les paramètres identifiant le hash utilisé lors de la création de la signature sont retirés du certificat. Le "fingerprint" du document est recalculé en appliquant ces informations. La signature est décryptée à l'aide de la clé publique contenue dans le certificat. Le "fingerprint" recalculé et la signature décryptée sont comparés. Lorsqu'ils sont identiques, le document a bel et bien été signé par le détenteur du certificat et n'a pas été changé après la signature.

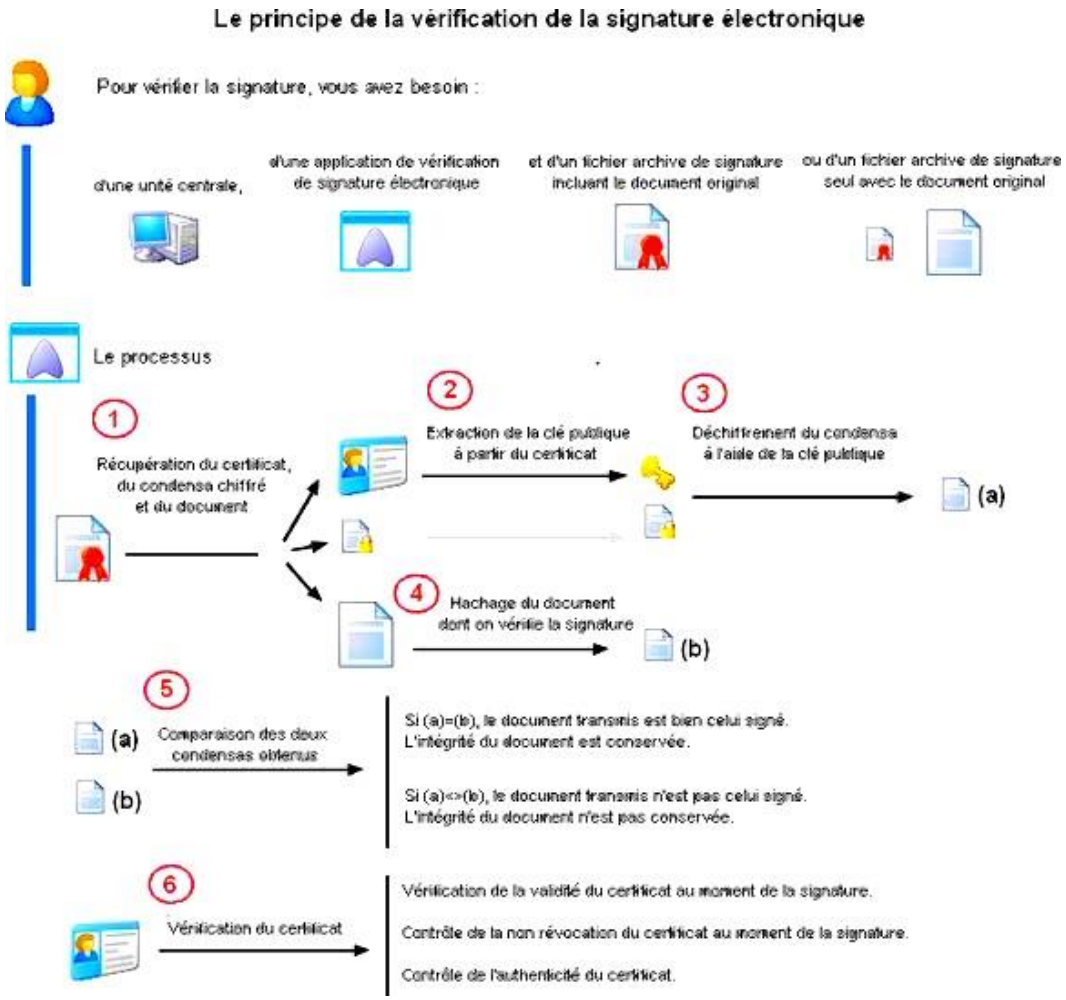

Figure 3 Les étapes principales de la vérification de signature numérique

II.10 L'intégrité dans le temps : l'archivage.

L'archivage est l'action de mettre en archive, d'archiver. Employé surtout à l'origine pour les seuls documents électroniques, comme un synonyme de stockage ou de sauvegarde, il tend de plus en plus à être utilisé pour tous les documents, quels qu'en soient la nature et le support, et à remplacer conservation

La conservation est révélatrice du fait que la technique doive servir le droit. Elle démontre l'ascendant du droit sur la technique. Parmi les Etats membres de l'Union européenne, le Portugal[2] est l'un des premiers pays à s'être interrogé sur la question de la conservation, la

nécessité de garantir l'intégrité du document, et les changements apportés à ceux-ci durant le temps de la conservation, lesquels sont susceptibles de les transformer en un nouveau document.
Le Conseil d'Etat, dans son rapport intitulé « *Internet et les réseaux numériques* », de Septembre 1998, indique que la conservation doit être « *durable* ».

Le Code civil, en son article 1348 alinéa 2, définit la durabilité comme engendrant une modification irréversible du support, ce qui ne parait pas adapté à un stockage informatique sur disques durs. Ainsi, la conservation doit uniquement être fiable.
Dans le cadre d'une infrastructure à clé publique mettant en œuvre une signature électronique, il faut archiver beaucoup de données et de documents. En effet, il faudrait conserver le certificat, les différents contrats signés et ceux conclus avec le prestataire, les copies des pièces justificatives d'identité et de qualité[3], toutes les informations qui pourraient se révéler nécessaires pour faire la preuve en justice de la certification électronique[4] et les données à caractère personnel.
La différence entre l'archivage publique et privé est qu'aux termes des dispositions de l'article L.211-4 du code du patrimoine, les archives publiques sont :

- Les documents qui procèdent de l'activité de l'Etat, des collectivités territoriales, des établissements et entreprises publics.
- les documents qui procèdent de l'activité des organismes de droit privé chargés de la gestion des services publics ou d'une mission de service public.
- Les minutes et répertoires des officiers publics ou ministériels.

Les archives privées sont définies par opposition à la définition d'archives publiques. En effet, l'article L.211-5 du code du patrimoine prévoit que les archives privées sont l'ensemble des documents définis à l'article L.211-1 cité ci-avant, qui n'entrent pas dans le champ d'application de l'article L.211-4 ci-dessus relatif aux archives publiques.
Ce sont, en substance, l'ensemble des documents des personnes physiques ou morales de droit privé.

II.11 Chiffrage et signature numérique

En combinant les signatures électroniques avec la cryptographie à clé publique, La signature électronique donne une preuve de l'identité de son auteure et le chiffrement offre la confidentialité. En matière de correspondance aussi il est prudent de signer avant de chiffrer. C'est nous seulement plus sûr. Un adversaire ne peut retirer la signature d'un message pour y mettre la sienne, mais il faut prendre en compte des considérations légales :

Si le signataire ne peut pas voir le texte qu'il signe, sa signature pourrait alors être contestée au vues de la loi [1323]. De plus, il existe des attaques crypte-analytiques contre cette technique utilisée avec des signatures RSA.

Une clé peut être mise en dépôt sans affecter à l'autre, et les clés peuvent avoir des tailles différentes ou encore expirer à des dates différentes.

Bien sûr, les messages doivent être datés pour éviter qu'ils ne soient réutilisés. La datation peut aussi aider à prémunir contre d'autres ennuis.

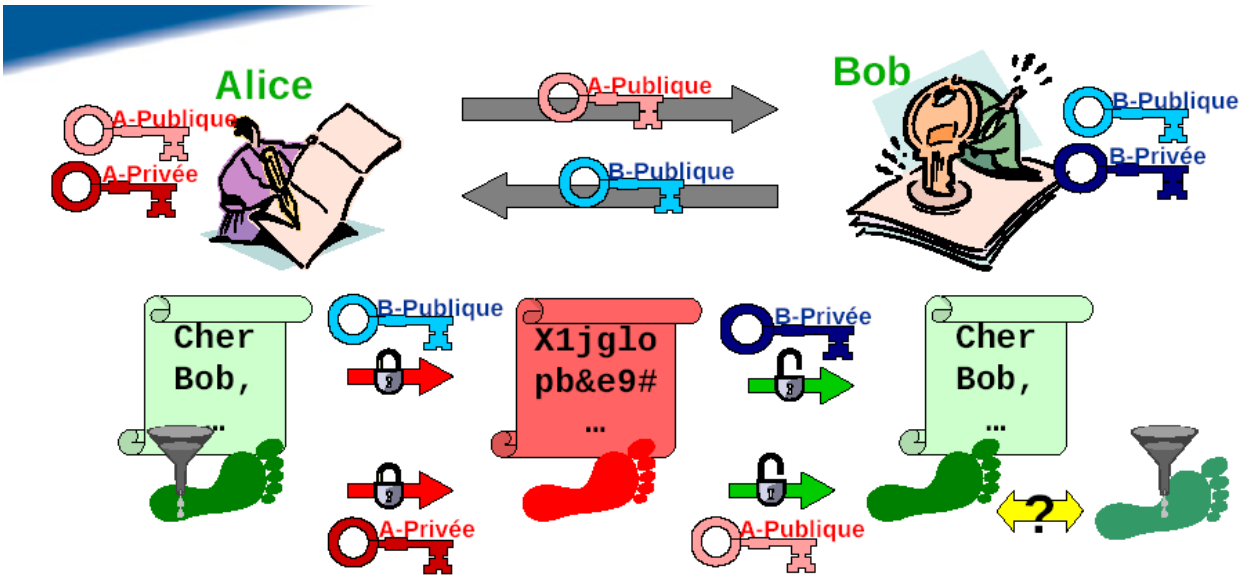

Figure 4 Le chiffrement dans la signature numérique

EXEMPLE

1. Alice signe le message avec sa clé privée.

   SA(M)

2. Alice chiffre le message avec la clé publique de Bernard et lui envoie le résultat.

   EB (SA(M))

3. Bernard déchiffre le message avec sa clé privée.

DB (EB (SA(M)))=SA(M)

4. Bernard vérifie avec la clé publique d'Alice et obtient le message initial.

VA (SA(M))=M

Il semble naturel de signer avant de chiffrer. Quand Alice écrit une lettre, elle la signe et met ensuite dans l'enveloppe. Si elle met la lettre non signée et signe ensuite l'enveloppe, alors Bernard pourrait craindre que la lettre n'ait été remplacée. Si Bernard montrait à Christine la lettre et l'enveloppe d'Alice, Christine pourrait l'accuser de mentir sur la lettre qui est arrivée dans l'enveloppe.

Alice n'a aucune raison d'utiliser la même paire clé publique, clé privée pour chiffrer et signer. Elle peut avoir deux paires de clés : une pour chiffrer et une pour signer.

II.12 Conclusion

Dans ce chapitre, nous avons présenté la signature électronique. Cette dernière vise à identifier le signataire d'un acte et manifester son consentement aux obligations qui découlent de cet acte. La signature électronique a des caractéristiques essentielles telles que la singularité, l'invariance et qu'elle doit être dérivée et d'autres caractéristiques non essentielles à savoir la concision et le coût de génération.

La condition de validité d'une signature électronique repose sur les certificats. Selon la définition de l'ISO, un certificat est un objet informatique qui permet de lier de façon intangible une entité (une personne, une ressource) à certaines caractéristiques de cette entité. Les certificats sont générés par des autorités de certification.

Dans le chapitre suivant, nous allons aborder l'infrastructure à clé publique et nous allons montrer comment générer et signer un certificat par une autorité de certification.

III. L'INFRASTRUCTURE A CLEF PUBLIQUE

Résumé

Dans ce chapitre on a bien expliqué le fonctionnement de la PKI, ces services et ces différents éléments qui la construisent. La génération de certificat signer à travers un PKI passe par la demande de certificat dédié à l'autorité d'enregistrement qui transmettr une demande à l'autorité de certificat qui gère une certificat signer à la personne qui à demander, tout ça est expliquer par détaille dans ce chapitre .

Mots-clés : PKI, ICP, BOB, Identité, clef publique, enregistrement, révocation, Annuaire LDP.

III.1 Introduction

En raison de la connectivité des réseaux d'aujourd'hui, le réseau d'un organisme est exposé à des utilisateurs non autorisés qui pourraient tenter d'accéder et de manipuler les données critiques de mission ou les données confidentielles de ses clients. La nécessité de l'authentification de l'identité des utilisateurs, des ordinateurs et même d'autres organisations, a conduit au développement de l'infrastructure à clé publique (PKI).

III.2 Définition de la PKI

On appel PKI (Public Key Infrastructure, ou en français infrastructure à clé publique (ICP), parfois infrastructure de gestion de clés (IGC), l'ensemble des solutions techniques basées sur la cryptographie à clé publique. La cryptographie à clés publiques emploie une combinaison de clefs publiques et privées, de signatures numériques, des certificats numériques, et des autorités de certification (AC), pour répondre à des nouvelles exigences de sécurité. Une PKI permet de centraliser l'ensemble de ces besoins en un seul point. Une PKI (Public Key Infrastructure) est un arrangement en cryptographie qui facilite l'examen des tiers, et attestant les identités d'utilisateur.

La PKI permet la liaison de clés publiques pour les utilisateurs. Ces clés publiques sont le plus souvent stockées dans des certificats. Cette fixation des clés publiques des utilisateurs est généralement réalisée par un logiciel dans un emplacement central, en coordination avec les autres composants logiciels associés installés dans les endroits distribués.

III.3 Fonctions de la PKI

Une PKI effectue les fonctions suivantes :

- **Enregistrement** : C'est un procédé par lequel le propriétaire qui sera le sujet du certificat se fera connaître auprès d'une AC. Ceci peut aussi se faire en passant par l'entremise d'une AE ou directement par l'AC. Le nom et les attributs utilisés pour identifier le propriétaire doivent être validés en relation avec les pratiques de certification (PC) dont dépend l'AC. L'AC suit alors les directives de la politique de certification (PC) pour valider si les détails fournis par le propriétaire sont corrects, avant de délivrer un certificat.
- **Certification** : C'est un processus par lequel une AC délivre un certificat pour un propriétaire de clé publique et adresse ce certificat au propriétaire ou le publie dans un dépôt de stockage.
- **Génération de clé** : Dans certains cas, le propriétaire peut générer des paires de clefs depuis un environnement local et transmettre la requête d'enregistrement vers l'AC. Dans d'autres cas, les paires de clefs peuvent être générées par une AC ou une AE via un générateur de clefs. Dans ce cas, le processus de distribution de la clef privée et du certificat vers le propriétaire sera réalisé de manière sécurisée.
- **Recouvrement de clefs** : Les clefs utilisées pour chiffrer des données, pour le transport ou les échanges peuvent avoir besoin d'être archivées pour satisfaire à des exigences de politique de sécurité. L'opération d'archivage nécessite par ailleurs une étape de récupération en cas de perte. Le recouvrement est un processus qui autorise la récupération de clefs quand une clef est perdue et que des informations précédemment chiffrées ont besoin d'être utilisées. Les opérations d'archivage et de recouvrement peuvent être réalisées soit par une AC soit par un système de recouvrement de clefs séparé de l'AC.
- **Mise à jour des clefs** : Toutes les clefs doivent pouvoir être remplacées sur une base régulière, quand une clef expire ou qu'elle est compromise. Si la mise à jour des clefs se produit en réponse à une expiration de sa date de péremption, la transition vers la nouvelle clef doit se produire avec facilité, cela exige néanmoins des mécanismes de support de notification de mise à jour. Dans le cas d'une compromission de clef, le certificat doit être déclaré invalide, et la nouvelle validité du certificat et sa disponibilité doivent être annoncées. Le support de mise à jour pour les clefs d'AC est

aussi nécessaire. Utilisée de façon transparente, cette technique sera, avec le temps, essentielle à l'opération de lissage de la PKI. La mise à jour des clefs et des certificats d'AC peut être supportée par la même procédure et le même protocole que pour l'expiration des clefs. En complément, il est nécessaire d'avoir un support pour la notification hors connexion (out-of-band) quand un événement se produit, tel qu'une compromission de clef d'AC. Une compromission de clef d'AC est un événement catastrophique, cela mène à la révocation du certificat d'AC et de tous les certificats publiés qui se subordonnent à elle.

- **Certification croisée** : Le processus de certification croisée permet à des utilisateurs d'un domaine de certification de faire confiance à des certificats délivrés par une AC d'un domaine de certification différent. On définit un certificat croisé comme un certificat publié par une AC pour certifier la paire de clefs privée/publique d'une autre AC. Un certificat croisé peut être publié dans le même domaine administratif ou à travers d'autres domaines administratifs. La certification croisée peut être utilisée dans une seule direction (une AC certifie une autre AC) ou dans les deux sens (les AC se certifient l'une et l'autre).
- **Révocation** : Dans la plupart des cas, un certificat reste valide jusqu'à ce que sa période de validité expire. Il y a, cependant, un certain nombre de scénarios qui rendent nécessaire la révocation avant la fin de la période de validité d'un certificat. Ceux-ci incluent :
    - le propriétaire change de nom
    - un employé quitte la société qui lui avait délivré le certificat
    - la clé privée du propriétaire est compromise

Les AC sont responsables du maintien de l'information sur l'état des certificats. Cela inclut le support pour la révocation des certificats qui sont devenus invalides avant la période d'expiration du certificat. Les CRL peuvent être utilisées comme mécanisme pour transporter des informations sur le statut des certificats révoqués. Si un certificat est révoqué, une entrée est ajoutée à la prochaine CRL publiée. Alternativement, les AC peuvent utiliser un mécanisme de notification de révocation en ligne tel qu'OCSP (Online Certificat Statu Protocol) pour réduire le temps de latence entre la révocation effectuée depuis l'autorité de certification et sa notification auprès de l'entité d'extrémité (EE). A la différence du classique mécanisme de publication de CRL, l'entité d'extrémité (EE) employant des méthodes de

service en ligne pour la validation certificat, doit pouvoir identifier le fournisseur du service en ligne.

III.4 Architecture de la PKI

Autorité de certification : entité qui crée des certificats. C'est une autorité morale qui définit les règles d'entrée des ressources et des individus dans la PKI. En pratique, il s'agit de définir les critères et les modalités d'attribution de certificats numériques.

Autorité d'enregistrement : entité chargée de recueillir les demandes de certificats et de contrôler l'identité de la personne ainsi que les critères d'attribution.

Certificat : Une identité électronique qui est émise par une tierce partie de confiance pour une personne ou une entité réseau. Chaque certificat est signé avec la clé privée de signature d'une autorité de certification. Il garantit l'identité d'un individu, d'une entreprise ou d'une organisation. En particulier, il contient la clé publique de l'entité et des informations associées à cette entité.

Certificat Auto signé : un certificat auto signé contient comme tout certificat une clé publique. Sa particularité réside dans le fait que ce certificat est signé avec la clé secrète associée. Dans ce cas précis, l'autorité de certification est donc le détenteur du certificat.

Certificat X.509 : Il s'agit d'une norme sur les certificats largement acceptée et conçue pour supporter une gestion sécurisée et la distribution des certificats numériquement signés sur le réseau Internet sécurisé. Le certificat X.509 définit des structures de données en accord avec les procédures pour distribuer les clés publiques qui sont signées numériquement par des parties tierces.

Certificat X.509v3 : Les certificats X.509v3 ont des extensions de structures de données pour stocker et récupérer des informations pour les applications, des informations sur les points de distribution des certificats, des CRLs et des informations sur les politiques de certification. Chaque fois qu'un certificat est utilisé, les capacités de X.509v3 permettent aux applications de vérifier la validité de ce certificat. Il permet aussi à l'application de vérifier si le certificat est dans une CRL. Ces certificats et CRLs sont normalisés auprès de l'IETF dans la RFC 2459.

Clé : Une quantité utilisée en cryptographie pour chiffrer/déchiffrer et signer/vérifier des données.

CRL : (Certificat Révocation List) se sont les listes de révocations de certificats.

Clé Publique : quantité numérique, attachée à une ressource ou un individu, qui la distribue aux autres afin qu'ils puissent lui envoyer des données chiffrées ou déchiffrer sa signature.

Clé Privée : quantité numérique secrète attachée à une ressource ou à un individu, lui permettant de déchiffrer des données chiffrées avec la clé publique correspondante ou d'apposer une signature au bas de messages envoyés vers des destinataires.

Bi-clé : couple de clés composé d'une clé privée et d'une clé publique

Schéma démonstratif de l'architecture du PKI

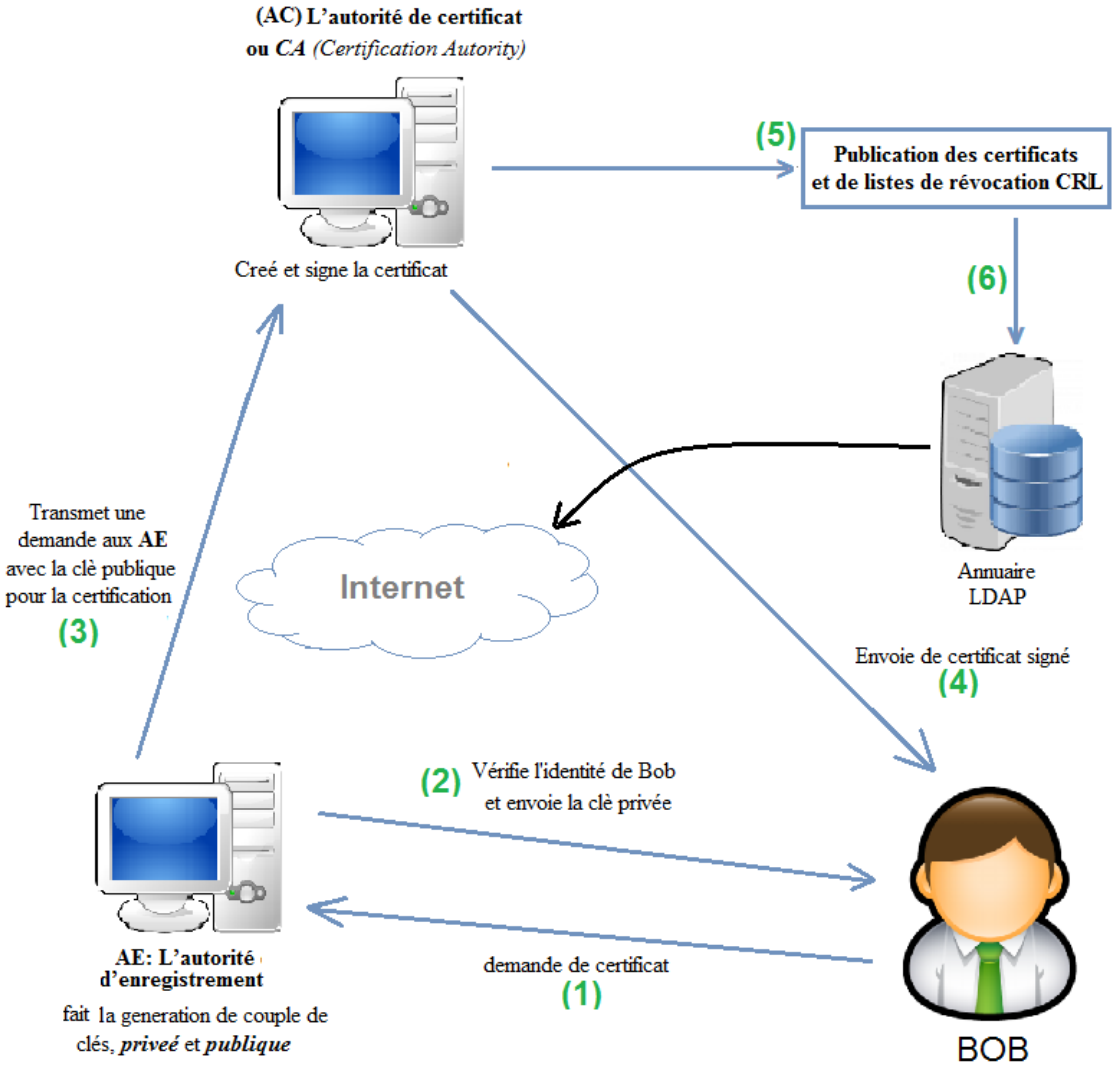

Figure 1 Schéma démonstratif de PKI

III.5 Conclusion

Une PKI est une infrastructure qui se construit, c'est donc une structure à la fois technique et administrative, avec 80% d'organisationnelle et 20% de technique. Le domaine des PKI est intéressant : il est possible de les utiliser pour des applications tels que mail chiffré, web sécurisé VPN (notamment IPSEC), commerce électronique...

Dans le chapitre suivant, nous allons développer une PKI. A cet effet, nous avons développé une application de génération de clefs et certificats électroniques.

IV. REALIASTION EN C

Résumé

Dans ce chapitre de réalisation, on va voire la génération de clefs pour un PKI en utilisant le RSA comme algorithme et on va suivre par un modèle de certificat numérique qui contient l'information principale de certificat.

Mots-clefs : génération des clefs, RSA, certificat numérique.

IV.1. Introduction

Dans la réalisation on a choisit de réaliser une modélisation de processus de création de certificat numérique de PKI en langage C, comme algorithme de chiffrement asymétrique on a choisis le RSA.

IV.2. Procédure

2.1 La génération de pair de clé

La génération de pair de clé « publique et privée » se passe à travers les méthodes de création de clé de RSA comme il est l'algorithme choisis dans la réalisation.

1. L'utilisateur choisis deux grands nombres premiers **p** et **q** au hasard.
2. Calcule leur : **n** = p*q.
3. Calcule leur : **φ(*n*)** = (q-1)*(p-1).
4. L'utilisateur choisit un autre nombre e premier avec **φ(*n*)** et en même temps inférieur à **φ(*n*)**, ce chiffre « **e »** appelé : *exposant de chiffrement*.
5. Calcule de **d** l'inverse modulo de e modulo **φ(*n*)**, tel que : **e*d= 1 mod φ(*n*)**, ce chiffre « **d »** appelé : *exposant de chiffrement*.
6. La clé publique : **(n,e)**.
7. La clé privé : **(n,d)**.

Tout cela a était réaliser dans un programme qui demande le user de donner deux nombre premier **p** et **q** et qui calcule leur **n** et **φ(*n*)** et puis demande **e** et calcule **l'inverse modulo d**, et affiche la **clé publique (n,e)** et la clé **privé (n,d)**.

2.2 La création de certificat numérique

Comme on a dit dans les chapitres précédents, le certificat électronique (numérique) contient les infos de la partie qui demande le certificat plus une clé publique et une date de primité. Et c'est ce qu'on a réalisé dans notre programme, où l'user donne :

- Son nom.
- Son prénom.
- Son e-mail.
- Une date de validité.

On attachons avec ça bien sûr la clé publique qui générer par l'algorithme de chiffrement qu'on a réalisé le RSA.

IV.3. Réalisation

Dans notre réalisation on a choisit de programmer un annuaire des nombres premier qui varie entre 1000 et 4500 pour faciliter la tâche à l'utilisateur et pour gagner de temps dans la phase de vérification de primitivité des nombres.

1 – Le user choisis les deux nombres premiers **p** et **q**, de l'annuaire devon qui lui facilite les choses et gagne le temps de vérification de primitivité de chiffre.

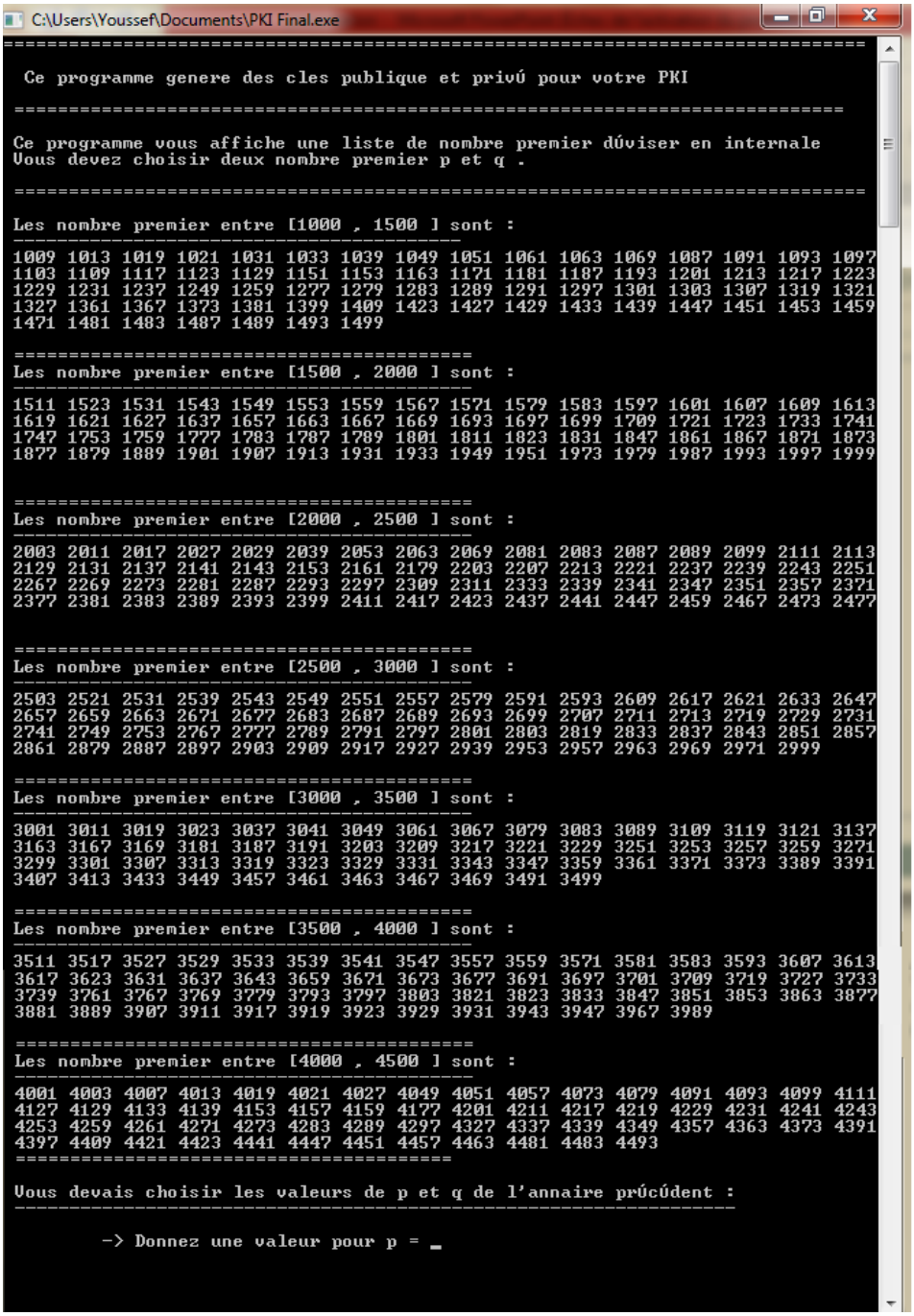

Figure 1 Annuaire de nombre premier pour choisir **p** et **q**

2- On a fait un choix :

q = 3359

p = 4073

3 – le calcule de ***φ*** et **n**, ce fait automatiquement :

n = *p*q*

φ = *(p-1)*(q-1)*

4 – *La valeur de e :*

e = 2201597.

5 - *la vérification de primitivité entre* ***e*** *et* ***φ*** *ce fait automatiquement aussi.*

Figure 1 Le calcule de n et fi après le choix de **p** et **q** et **e** par le user

6 – La clef publique (**n**,**e**) :

CPb (13681207,2201597) .

7 – Calcule de **d** :

d = 2201597 MOD 13673776

d = 12434725.

8 - clef privée (**n**,**d**) :

CPR (13681207**,**12434725).

Le programme maintenant va demander le user de donnée des donné pour crée un certificat.

IV.4 la génération des certificats :

9 - Le programme va demander le user de donée :

- Le nom :
- Le prénom :
- E-mail :
- Date de validation de certificat :
  - De : jj/ mm / année
  - Au : jj/ mm / année

Et puis le certificat sera crée automatiquement à l'aide de ces informations données, comme est montrer par la suite :

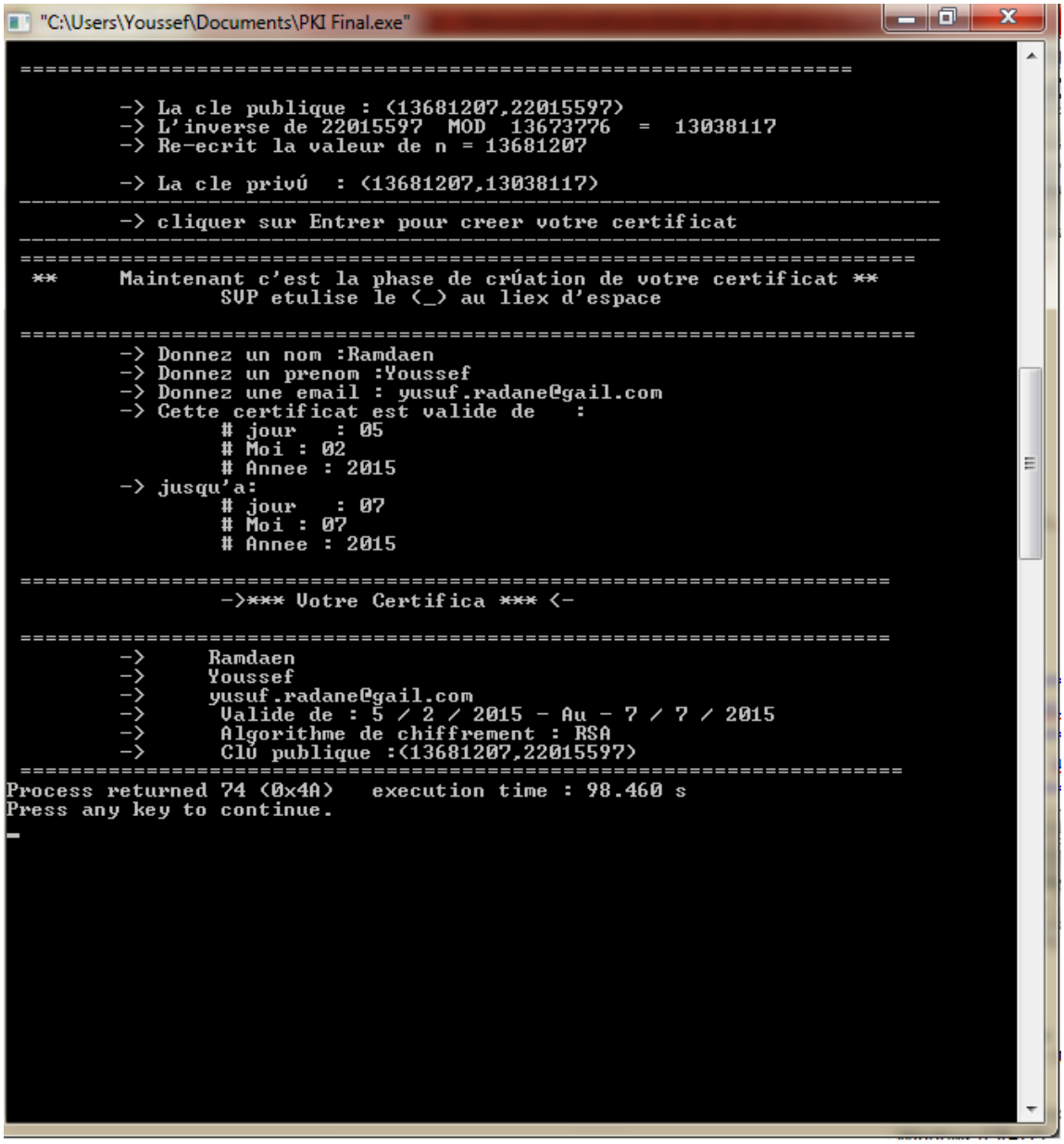

Figure 2 génération de certificats électronique à l'aide des données remplies par l'user

Et là on a un certificat qui contient les informations nécessaire de demandeur et ça durée et l'algorithme de chiffrement utiliser et la clef publique que le correspondant pourra utiliser dans le déchiffrement et la vérification de validité de cette certificat.

IV.5 conclusion

Les valeurs de p et q et e choisis dans notre exemple, sont juste des exemples, un autre user peut choisir autres valeurs, de préférence ces valeurs soit grandes, pour avoir des clefs de grandes tailles. Dans les différentes modèles de PKI le user soit donnes des valeurs pour p et q et e soit les valeurs de p et q seront donnée aléatoirement par le programme à l'aides des fonction dédié à cette tâche. Après le calcule de n et φ le programme affiche la clef publique et privée, et puis demande de remplir le certificat électronique.

Nous avant présenter qu'un simple modèle de PKI, qui contient la génération des clefs « publique, privée » et la génération d'un certificat. Ça c'était qu'un modèle simple, il y a autre méthode pour crée un PKI en OpenSSL mais ça reste qu'un modèle aussi, car la réalisation d'un PKI réel est trop difficile qui nécessite du matérielles et équipement et de programme de haut performance.

Algorithme de génération des paires de clef et certificats des PKI

```
/* Algorithme de génération des certificats des PKI*/

/* Réaliser par : Ramdane Youssef */

#include <stdio.h>

#include <conio.h>

void euclide(int e,int fi);

void main(void)

{

   int i , j , z,t ,flag,C,k,l,vale,p,q,e,fi,n,d,PGCD,u,y,t1,R,J,M,An,JF,MF,AnF;

   char Nom[60];

   char prenom[60];

   char email[200];

   char MP[90];

printf("================================================================
========== \n ");

   printf(" Ce programme génère des clefs publique et privé pour votre PKI \n ");

   printf("\n
================================================================================
= ");

   printf(" \n \n Ce programme vous affiche une liste de nombre premier diviser en intervalle
\n Vous devez choisir deux nombre premier p et q . \n ");

   printf("\n
================================================================================
===");

/* Le programme va afficher tous les nombres premiers qui existe dans l'intervalle [ 0,1000]
*/
```

```
    printf(" \n Les nombre premier entre [1000 , 1500 ] sont : \n ----------------------------------------
-- \n ");

                for(i=1000; i<=1500; ++i)

                {

                   flag=0;

                   for(j=2; j<=i/2; ++j)

                   {

                    if(i%j==0)

                    {

                      flag=1;

                      break;

                    }

                   }

                   if(flag==0)

                    printf("%d ",i);

                }

          printf("\n \n ======================================== \n Les nombre
premier entre [1500 , 2000 ] sont : \n ---------------------------------------- \n ");

                for(i=1500; i<=2000; ++i)

                {

                   flag=0;

                   for(j=2; j<=i/2; ++j)

                   {

                    if(i%j==0)

                    {

                      flag=1;

                      break;
```

```
      }
    }
    if(flag==0)
      printf("%d ",i);
  }
 printf("\n \n ========================================= \n Les nombre premier entre [2000 , 2500 ] sont : \n ---------------------------------------- \n ");
  for(i=2000; i<=2500; ++i)
  {
    flag=0;
    for(j=2; j<=i/2; ++j)
    {
      if(i%j==0)
      {
        flag=1;
        break;
      }
    }
    if(flag==0)
      printf("%d ",i);
  }
 printf("\n \n ========================================= \n Les nombre premier entre [2500 , 3000 ] sont : \n ---------------------------------------- \n ");
  for(i=2500; i<=3000; ++i)
  {
    flag=0;
    for(j=2; j<=i/2; ++j)
```

```
    {
      if(i%j==0)
      {
        flag=1;
        break;
      }
    }
    if(flag==0)
      printf("%d ",i);
  }
  printf("\n \n ======================================== \n Les nombre premier entre [3000 , 3500 ] sont : \n ---------------------------------------- \n ");
  for(i=3000; i<=3500; ++i)
  {
    flag=0;
    for(j=2; j<=i/2; ++j)
    {
      if(i%j==0)
      {
        flag=1;
        break;
      }
    }
    if(flag==0)
      printf("%d ",i);
  }
```

```
printf("\n \n ========================================= \n Les nombre premier entre [3500 , 4000 ] sont : \n ---------------------------------------- \n ");

        for(i=3500; i<=4000; ++i)

        {

           flag=0;

           for(j=2; j<=i/2; ++j)

           {

            if(i%j==0)

            {

              flag=1;

              break;

            }

           }

           if(flag==0)

            printf("%d ",i);

        }

    printf("\n \n ========================================= \n Les nombre premier entre [4000 , 4500 ] sont : \n ---------------------------------------- \n ");

        for(i=4000; i<=4500; ++i)

        {

           flag=0;

           for(j=2; j<=i/2; ++j)

           {

            if(i%j==0)

            {

              flag=1;

              break;
```

```
                    }
                   }
                   if(flag==0)
                    printf("%d ",i);
                 }
printf("\n ======================================== ");
printf(" \n \n Vous devais choisir les valeurs de p et q de l'annaire précédent : \n ----------
---------------------------------------------- ");
/* L'utilisateur doit choisir les valeurs de p et q de la liste des nombres premier afficher
dessous */
printf("\n \n \t -> Donnez une valeur pour p = ");
scanf("%d",&p);
printf("\t -> Donnez une valeur pour q = ");
scanf("%d",&q);
n=q*p;
fi=(q-1)*(p-1);
printf(" \t -> la valeur de n=%d\n ",n);
printf(" \t -> La valeur de fi=%d \n ",fi);
/* Choix et vérification de primitivité de e avec fi {fi(n)} */
for(z=0;z<1000;z++){
    printf(" \t -> Donnez une valeur pour e ( inferieur à fi ), e =");
    scanf("%d",&e);
    printf(" \t -> S.V.P Reecrit la valeur de fi : ");
    scanf("%d",&fi);
    for(t=1; t<=e || t<=fi; t++)
  {
    if(e%t==0 && fi%t==0)
```

```
        PGCD=t;
   }
   if(PGCD!=1) { printf(" \n \t -> OOps ! %d et %d ne sont pas premier entre eux . \n \n \t -> veiller donnez une autre valeur pour e \n ",e,fi);}
   else if(PGCD==1)  { printf(" \t -> %d et %d sont premier entre eux \n  \t -> Danc : \n",e,fi);
                    break ; }
   }
    printf(" \n \t -> Re-ecrit la valeur de e :");
    scanf("%d",&u);
    printf("\n \t -> Re-ecrit la valeur de fi : ");
    scanf("%d",&y);
/* L'affichage des cles publique et privé */
   printf("
========================================================================= \n");
   printf(" \t \t ->*** Voici votre clef publique et clef prive *** <- \n  ");
   printf(" \n
========================================================================= \n ");
   printf(" \n \t -> La clef publique : (%d,%d)",n,e);
   euclide(u,y);
/* génération de certificat */
printf("\n
========================================================================= \n ");
printf(" ** \t Maintenant c'est la phase de création de votre certificat ** \n \t \t SVP etulise le (_) au lieux d'espace \n  ");
printf("\n
========================================================================= \n ");
   printf("\t -> Donnez un nom :");
   scanf("%s",Nom);
```

```
    printf("\t -> Donnez un prenom :");
    scanf("%s",prenom);
    printf("\t -> Donnez une email : ");
    scanf("%s",email);
    printf("\t -> Cette certificat est valide de   : \n ");
    printf("\t \t # jour   : ");
    scanf("%d",&J);
    printf("\t \t # Moi : ");
    scanf("%d",&M);
    printf("\t \t # Annee : ");
    scanf("%d",&An);
    printf("\t -> jusqu'a: \n ");
    printf("\t \t # jour   : ");
    scanf("%d",&JF);
    printf("\t \t # Moi : ");
    scanf("%d",&MF);
    printf("\t \t # Annee : ");
    scanf("%d",&AnF);
    printf(" \n
========================================================================== \n");
    printf(" \t \t ->*** Votre Certificat *** <- \n  ");
    printf(" \n
========================================================================== \n ");
    printf("\t -> \t"); puts(Nom);
    printf("\t -> \t"); puts(prenom);
    printf("\t -> \t"); puts(email);
    printf("\t ->\t Valide de : %d / %d / %d - Au - %d / %d / %d \n ",J,M,An,JF,MF,AnF);
```

```
   printf("\t -> \t Algorithme de chiffrement : RSA \n " );
   printf("\t ->\t Clé publique :(%d,%d)\t",n,e);
   printf(" \n
========================================================================== ");
}
void euclide(int e,int fi)
{
    int ww = e;
    int mm = fi;
    int t0 = 0;
    int t1 = 1;
    int q1 = (int) mm/ww;
    int r1= mm - q1 * ww;
    int tmp = 0;
    int n;

  while(r1 > 0)
  {
      tmp = t0 - q1 * t1;

      if (tmp >= 0)
          {
          tmp = tmp % fi;
       }
       else
       {
```

```
            tmp = fi - ((-tmp) % fi);
        }
            t0 = t1;
            t1 = tmp;
            mm = ww;
            ww = r1;
            q1 = (int) mm/ww;
            r1 = mm - q1 * ww;
    }
    if (ww!=1)
    {
            printf("%d   n'a pas d'inverse modulo   %d\n\n",e,fi);
    }
    else    {
        printf ("\n \t -> L'inverse de %d  MOD  %d  =  %d\n ",e,fi,t1);
        printf(" \t -> Re-ecrit la valeur de n = ");
        scanf("%d",&n);
        printf ("\n \t -> La clef privé  : (%d,%d) \n",n,t1);
        printf(" ---------------------------------------------------------------------- \n \t -> cliquer sur Entrer
pour creer votre certificat \n ---------------------------------------------------------------------- ");
          }
getch();
}
```

Code source de signature (Hachage) en MD5

```
/*
**********************************************************************
** md5.h -- Header file for implementation of MD5                   **
** RSA Data Security, Inc. MD5 Message Digest Algorithm             **
** Created: 2/17/90 RLR                                             **
** Revised: 12/27/90 SRD,AJ,BSK,JT Reference C version              **
** Revised (for MD5): RLR 4/27/91                                   **
**   -- G modified to have y&~z instead of y&z                      **
**   -- FF, GG, HH modified to add in last register done            **
**   -- Access pattern: round 2 works mod 5, round 3 works mod 3    **
**   -- distinct additive constant for each step                    **
**   -- round 4 added, working mod 7                                **
**********************************************************************
*/

/*
**********************************************************************
** Copyright (C) 1990, RSA Data Security, Inc. All rights reserved. **
**                                                                  **
** License to copy and use this software is granted provided that   **
** it is identified as the "RSA Data Security, Inc. MD5 Message     **
** Digest Algorithm" in all material mentioning or referencing this **
** software or this function.                                       **
```

```
**                                                **
** License is also granted to make and use derivative works        **
** provided that such works are identified as "derived from the RSA **
** Data Security, Inc. MD5 Message Digest Algorithm" in all        **
** material mentioning or referencing the derived work.            **
**                                                **
** RSA Data Security, Inc. makes no representations concerning      **
** either the merchantability of this software or the suitability   **
** of this software for any particular purpose.  It is provided "as **
** is" without express or implied warranty of any kind.            **
**                                                **
** These notices must be retained in any copies of any part of this **
** documentation and/or software.                                  **
**********************************************************************
*/

/* typedef a 32 bit type */
typedef unsigned long int UINT4;

/* Data structure for MD5 (Message Digest) computation */
typedef struct {
  UINT4 Tab1[2];                /* number of _bits_ handled mod 2^64 */
  UINT4 buf[4];                                 /* scratch buffer */
  unsigned char in[64];                         /* input buffer */
  unsigned char digest[16];     /* actual digest after MD5Final call */
} MD5_CTX;
```

```
void MD5Init ();

void MD5Update ();

void MD5Final ();


/*

 ***********************************************************************

 ** End of md5.h                                    **

 ****************************** (cut) ********************************

 */


/*

 ***********************************************************************

 ** md5.c                                          **

 ** RSA Data Security, Inc. MD5 Message Digest Algorithm          **

 ** Created: 2/17/90 RLR                                 **

 ** Revised: 1/91 SRD,AJ,BSK,JT Reference C Version              **

 ***********************************************************************

 */


/*

 ***********************************************************************

 ** Copyright (C) 1990, RSA Data Security, Inc. All rights reserved. **

 **                                                **

 ** License to copy and use this software is granted provided that   **

 ** it is identified as the "RSA Data Security, Inc. MD5 Message     **
```

```
** Digest Algorithm" in all material mentioning or referencing this **
** software or this function.                                        **
**                                                                   **
** License is also granted to make and use derivative works         **
** provided that such works are identified as "derived from the RSA **
** Data Security, Inc. MD5 Message Digest Algorithm" in all          **
** material mentioning or referencing the derived work.             **
**                                                                   **
** RSA Data Security, Inc. makes no representations concerning       **
** either the merchantability of this software or the suitability    **
** of this software for any particular purpose.  It is provided "as **
** is" without express or implied warranty of any kind.              **
**                                                                   **
** These notices must be retained in any copies of any part of this **
** documentation and/or software.                                    **
**********************************************************************
*/

/* -- include the following line if the md5.h header file is separate -- */
/* #include "md5.h" */

/* forward declaration */
static void Transform ();

static unsigned char PADDING[64] = {
  0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

```
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

/* F, G and H are basic MD5 functions: selection, majority, parity */
#define F(x, y, z) (((x) & (y)) | ((~x) & (z)))
#define G(x, y, z) (((x) & (z)) | ((y) & (~z)))
#define H(x, y, z) ((x) ^ (y) ^ (z))
#define I(x, y, z) ((y) ^ ((x) | (~z)))

/* ROTATE_LEFT rotates x left n bits */
#define ROTATE_LEFT(x, n) (((x) << (n)) | ((x) >> (32-(n))))

/* FF, GG, HH, and II transformations for rounds 1, 2, 3, and 4 */
/* Rotation is separate from addition to prevent recomputation */
#define FF(a, b, c, d, x, s, ac) \
  {(a) += F ((b), (c), (d)) + (x) + (UINT4)(ac); \
   (a) = ROTATE_LEFT ((a), (s)); \
   (a) += (b); \
  }
#define GG(a, b, c, d, x, s, ac) \
```

```
  {(a) += G ((b), (c), (d)) + (x) + (UINT4)(ac); \
   (a) = ROTATE_LEFT ((a), (s)); \
   (a) += (b); \
  }
#define HH(a, b, c, d, x, s, ac) \
  {(a) += H ((b), (c), (d)) + (x) + (UINT4)(ac); \
   (a) = ROTATE_LEFT ((a), (s)); \
   (a) += (b); \
  }
#define II(a, b, c, d, x, s, ac) \
  {(a) += I ((b), (c), (d)) + (x) + (UINT4)(ac); \
   (a) = ROTATE_LEFT ((a), (s)); \
   (a) += (b); \
  }

void MD5Init (mdContext)
MD5_CTX *mdContext;
{
  mdContext->Tab1[0] = mdContext->Tab1[1] = (UINT4)0;

  /* Load magic initialization constants.
   */
  mdContext->buf[0] = (UINT4)0x67452301;
  mdContext->buf[1] = (UINT4)0xefcdab89;
  mdContext->buf[2] = (UINT4)0x98badcfe;
  mdContext->buf[3] = (UINT4)0x10325476;
```

```
}

void MD5Update (mdContext, inBuf, inLen)
MD5_CTX *mdContext;
unsigned char *inBuf;
unsigned int inLen;
{
  UINT4 in[16];
  int mdi;
  unsigned int i, ii;

  /* compute number of bytes mod 64 */
  mdi = (int)((mdContext->i[0] >> 3) & 0x3F);

  /* update number of bits */
  if ((mdContext->Tab1[0] + ((UINT4)inLen << 3)) < mdContext->Tab1[0])
    mdContext->Tab1[1]++;
  mdContext->Tab1[0] += ((UINT4)inLen << 3);
  mdContext->Tab1[1] += ((UINT4)inLen >> 29);

  while (inLen--) {
    /* add new character to buffer, increment mdi */
    mdContext->in[mdi++] = *inBuf++;

    /* transform if necessary */
    if (mdi == 0x40) {
```

```
    for (Tab1 = 0, ii = 0; Tab1 < 16; i++, ii += 4)
      in[Tab1] = (((UINT4)mdContext->in[ii+3]) << 24) |
            (((UINT4)mdContext->in[ii+2]) << 16) |
            (((UINT4)mdContext->in[ii+1]) << 8) |
            ((UINT4)mdContext->in[ii]);
    Transform (mdContext->buf, in);
    mdi = 0;
   }
  }
}

void MD5Final (mdContext)
MD5_CTX *mdContext;
{
  UINT4 in[16];
  int mdi;
  unsigned int Tab1, ii;
  unsigned int padLen;

  /* save number of bits */
  in[14] = mdContext->Tab1[0];
  in[15] = mdContext->Tab1[1];

  /* compute number of bytes mod 64 */
  mdi = (int)((mdContext->Tab1[0] >> 3) & 0x3F);
```

```
 /* pad out to 56 mod 64 */
 padLen = (mdi < 56) ? (56 - mdi) : (120 - mdi);
 MD5Update (mdContext, PADDING, padLen);

 /* append length in bits and transform */
 for (Tab1 = 0, ii = 0; Tab1 < 14; Tab1++, ii += 4)
  in[Tab1] = (((UINT4)mdContext->in[ii+3]) << 24) |
        (((UINT4)mdContext->in[ii+2]) << 16) |
        (((UINT4)mdContext->in[ii+1]) << 8) |
        ((UINT4)mdContext->in[ii]);
 Transform (mdContext->buf, in);

 /* store buffer in digest */
 for (Tab1 = 0, ii = 0; Tab1 < 4; i++, ii += 4) {
  mdContext->digest[ii] = (unsigned char)(mdContext->buf[Tab1] & 0xFF);
  mdContext->digest[ii+1] =
   (unsigned char)((mdContext->buf[Tab1] >> 8) & 0xFF);
  mdContext->digest[ii+2] =
   (unsigned char)((mdContext->buf[Tab1] >> 16) & 0xFF);
  mdContext->digest[ii+3] =
   (unsigned char)((mdContext->buf[Tab1] >> 24) & 0xFF);
 }
}

/* Basic MD5 step. Transform buf based on in.
 */
```

```
static void Transform (buf, in)
UINT4 *buf;
UINT4 *in;
{
  UINT4 a = buf[0], b = buf[1], c = buf[2], d = buf[3];

  /* Round 1 */
#define S11 7
#define S12 12
#define S13 17
#define S14 22
  FF ( a, b, c, d, in[ 0], S11, 3614090360); /* 1 */
  FF ( d, a, b, c, in[ 1], S12, 3905402710); /* 2 */
  FF ( c, d, a, b, in[ 2], S13,  606105819); /* 3 */
  FF ( b, c, d, a, in[ 3], S14, 3250441966); /* 4 */
  FF ( a, b, c, d, in[ 4], S11, 4118548399); /* 5 */
  FF ( d, a, b, c, in[ 5], S12, 1200080426); /* 6 */
  FF ( c, d, a, b, in[ 6], S13, 2821735955); /* 7 */
  FF ( b, c, d, a, in[ 7], S14, 4249261313); /* 8 */
  FF ( a, b, c, d, in[ 8], S11, 1770035416); /* 9 */
  FF ( d, a, b, c, in[ 9], S12, 2336552879); /* 10 */
  FF ( c, d, a, b, in[10], S13, 4294925233); /* 11 */
  FF ( b, c, d, a, in[11], S14, 2304563134); /* 12 */
  FF ( a, b, c, d, in[12], S11, 1804603682); /* 13 */
  FF ( d, a, b, c, in[13], S12, 4254626195); /* 14 */
  FF ( c, d, a, b, in[14], S13, 2792965006); /* 15 */
```

```
  FF ( b, c, d, a, in[15], S14, 1236535329); /* 16 */


  /* Round 2 */
#define S21 5
#define S22 9
#define S23 14
#define S24 20
  GG ( a, b, c, d, in[ 1], S21, 4129170786); /* 17 */
  GG ( d, a, b, c, in[ 6], S22, 3225465664); /* 18 */
  GG ( c, d, a, b, in[11], S23,  643717713); /* 19 */
  GG ( b, c, d, a, in[ 0], S24, 3921069994); /* 20 */
  GG ( a, b, c, d, in[ 5], S21, 3593408605); /* 21 */
  GG ( d, a, b, c, in[10], S22,   38016083); /* 22 */
  GG ( c, d, a, b, in[15], S23, 3634488961); /* 23 */
  GG ( b, c, d, a, in[ 4], S24, 3889429448); /* 24 */
  GG ( a, b, c, d, in[ 9], S21,  568446438); /* 25 */
  GG ( d, a, b, c, in[14], S22, 3275163606); /* 26 */
  GG ( c, d, a, b, in[ 3], S23, 4107603335); /* 27 */
  GG ( b, c, d, a, in[ 8], S24, 1163531501); /* 28 */
  GG ( a, b, c, d, in[13], S21, 2850285829); /* 29 */
  GG ( d, a, b, c, in[ 2], S22, 4243563512); /* 30 */
  GG ( c, d, a, b, in[ 7], S23, 1735328473); /* 31 */
  GG ( b, c, d, a, in[12], S24, 2368359562); /* 32 */


  /* Round 3 */
#define S31 4
```

```
#define S32 11

#define S33 16

#define S34 23

  HH ( a, b, c, d, in[ 5], S31, 4294588738); /* 33 */

  HH ( d, a, b, c, in[ 8], S32, 2272392833); /* 34 */

  HH ( c, d, a, b, in[11], S33, 1839030562); /* 35 */

  HH ( b, c, d, a, in[14], S34, 4259657740); /* 36 */

  HH ( a, b, c, d, in[ 1], S31, 2763975236); /* 37 */

  HH ( d, a, b, c, in[ 4], S32, 1272893353); /* 38 */

  HH ( c, d, a, b, in[ 7], S33, 4139469664); /* 39 */

  HH ( b, c, d, a, in[10], S34, 3200236656); /* 40 */

  HH ( a, b, c, d, in[13], S31,  681279174); /* 41 */

  HH ( d, a, b, c, in[ 0], S32, 3936430074); /* 42 */

  HH ( c, d, a, b, in[ 3], S33, 3572445317); /* 43 */

  HH ( b, c, d, a, in[ 6], S34,   76029189); /* 44 */

  HH ( a, b, c, d, in[ 9], S31, 3654602809); /* 45 */

  HH ( d, a, b, c, in[12], S32, 3873151461); /* 46 */

  HH ( c, d, a, b, in[15], S33,  530742520); /* 47 */

  HH ( b, c, d, a, in[ 2], S34, 3299628645); /* 48 */


  /* Round 4 */

#define S41 6

#define S42 10

#define S43 15

#define S44 21

  II ( a, b, c, d, in[ 0], S41, 4096336452); /* 49 */
```

```
 II ( d, a, b, c, in[ 7], S42, 1126891415); /* 50 */
 II ( c, d, a, b, in[14], S43, 2878612391); /* 51 */
 II ( b, c, d, a, in[ 5], S44, 4237533241); /* 52 */
 II ( a, b, c, d, in[12], S41, 1700485571); /* 53 */
 II ( d, a, b, c, in[ 3], S42, 2399980690); /* 54 */
 II ( c, d, a, b, in[10], S43, 4293915773); /* 55 */
 II ( b, c, d, a, in[ 1], S44, 2240044497); /* 56 */
 II ( a, b, c, d, in[ 8], S41, 1873313359); /* 57 */
 II ( d, a, b, c, in[15], S42, 4264355552); /* 58 */
 II ( c, d, a, b, in[ 6], S43, 2734768916); /* 59 */
 II ( b, c, d, a, in[13], S44, 1309151649); /* 60 */
 II ( a, b, c, d, in[ 4], S41, 4149444226); /* 61 */
 II ( d, a, b, c, in[11], S42, 3174756917); /* 62 */
 II ( c, d, a, b, in[ 2], S43,  718787259); /* 63 */
 II ( b, c, d, a, in[ 9], S44, 3951481745); /* 64 */

 buf[0] += a;
 buf[1] += b;
 buf[2] += c;
 buf[3] += d;
}

/*
 ***********************************************************************
 ** End of md5.c                                      **
 ******************************** (cut) ********************************
```

```
 */

/*
 **********************************************************************
 ** md5driver.c -- sample routines to test                           **
 ** RSA Data Security, Inc. MD5 message digest algorithm.            **
 ** Created: 2/16/90 RLR                                             **
 ** Updated: 1/91 SRD                                                **
 **********************************************************************
 */

/*
 **********************************************************************
 ** Copyright (C) 1990, RSA Data Security, Inc. All rights reserved. **
 **                                                                  **
 ** RSA Data Security, Inc. makes no representations concerning      **
 ** either the merchantability of this software or the suitability   **
 ** of this software for any particular purpose.  It is provided "as **
 ** is" without express or implied warranty of any kind.             **
 **                                                                  **
 ** These notices must be retained in any copies of any part of this **
 ** documentation and/or software.                                   **
 **********************************************************************
 */

#include <stdio.h>
```

```
#include <sys/types.h>

#include <time.h>

#include <string.h>

/* -- include the following file if the file md5.h is separate -- */

/* #include "md5.h" */


/* Prints message digest buffer in mdContext as 32 hexadecimal digits.

   Order is from low-order byte to high-order byte of digest.

   Each byte is printed with high-order hexadecimal digit first.

 */

static void MDPrint (mdContext)

MD5_CTX *mdContext;

{

  int Tab1;


  for (Tab1 = 0; Tab1 < 16; Tab1++)

    printf ("%02x", mdContext->digest[Tab1]);

}


/* size of test block */

#define TEST_BLOCK_SIZE 1000


/* number of blocks to process */

#define TEST_BLOCKS 10000


/* number of test bytes = TEST_BLOCK_SIZE * TEST_BLOCKS */
```

```
static long TEST_BYTES = (long)TEST_BLOCK_SIZE * (long)TEST_BLOCKS;


/* A time trial routine, to measure the speed of MD5.
   Measures wall time required to digest TEST_BLOCKS * TEST_BLOCK_SIZE
   characters.
 */
static void MDTimeTrial ()
{
  MD5_CTX mdContext;
  time_t endTime, startTime;
  unsigned char data[TEST_BLOCK_SIZE];
  unsigned int Tab1;

  /* initialize test data */
  for (Tab1 = 0; Tab1 < TEST_BLOCK_SIZE; Tab1++)
    data[Tab1] = (unsigned char)(Tab1 & 0xFF);

  /* start timer */
  printf ("MD5 time trial. Processing %ld characters...\n", TEST_BYTES);
  time (&startTime);

  /* digest data in TEST_BLOCK_SIZE byte blocks */
  MD5Init (&mdContext);
  for (Tab1 = TEST_BLOCKS; Tab1 > 0; Tab1--)
    MD5Update (&mdContext, data, TEST_BLOCK_SIZE);
  MD5Final (&mdContext);
```

```
  /* stop timer, get time difference */
  time (&endTime);
  MDPrint (&mdContext);
  printf (" is digest of test input.\n");
  printf
    ("Seconds to process test input: %ld\n", (long)(endTime-startTime));
  printf
    ("Characters processed per second: %ld\n",
     TEST_BYTES/(endTime-startTime));
}

/* Computes the message digest for string inString.
   Prints out message digest, a space, the string (in quotes) and a
   carriage return.
 */
static void MDString (inString)
char *inString;
{
  MD5_CTX mdContext;
  unsigned int len = strlen (inString);

  MD5Init (&mdContext);
  MD5Update (&mdContext, inString, len);
  MD5Final (&mdContext);
  MDPrint (&mdContext);
```

```
  printf (" \"%s\"\n\n", inString);
}

/* Computes the message digest for a specified file.
   Prints out message digest, a space, the file name, and a carriage
   return.
 */
static void MDFile (filename)
char *filename;
{
  FILE *inFile = fopen (filename, "rb");
  MD5_CTX mdContext;
  int bytes;
  unsigned char data[1024];

  if (inFile == NULL) {
    printf ("%s can't be opened.\n", filename);
    return;
  }

  MD5Init (&mdContext);
  while ((bytes = fread (data, 1, 1024, inFile)) != 0)
    MD5Update (&mdContext, data, bytes);
  MD5Final (&mdContext);
  MDPrint (&mdContext);
  printf (" %s\n", filename);
```

```
  fclose (inFile);
}

/* Writes the message digest of the data from stdin onto stdout,
   followed by a carriage return.
 */
static void MDFilter ()
{
  MD5_CTX mdContext;
  int bytes;
  unsigned char data[16];

  MD5Init (&mdContext);
  while ((bytes = fread (data, 1, 16, stdin)) != 0)
    MD5Update (&mdContext, data, bytes);
  MD5Final (&mdContext);
  MDPrint (&mdContext);
  printf ("\n");
}

/* Runs a standard suite of test data.
 */
static void MDTestSuite ()
{
  printf ("MD5 test suite results:\n\n");
  MDString ("");
```

```
 MDString ("a");
 MDString ("abc");
 MDString ("message digest");
 MDString ("abcdefghijklmnopqrstuvwxyz");
 MDString
  ("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789");
 MDString
  ("12345678901234567890123456789012345678901234567890\
12345678901234567890123456789012345678901234567890");
 /* Contents of file foo are "abc" */
 MDFile ("foo");
}

void main (argc, argv)
int argc;
char *argv[];
{
 int Tab1;

 /* For each command line argument in turn:
 ** filename         -- prints message digest and name of file
 ** -sstring         -- prints message digest and contents of string
 ** -t               -- prints time trial statistics for 1M characters
 ** -x               -- execute a standard suite of test data
 ** (no args)        -- writes messages digest of stdin onto stdout
 */
```

```
 if (argc == 1)
   MDFilter ();
 else
   for (Tab1 = 1; Tab1 < argc; Tab1++)
     if (argv[Tab1][0] == '-' && argv[Tab1][1] == 's')
       MDString (argv[Tab1] + 2);
     else if (strcmp (argv[Tab1], "-t") == 0)
       MDTimeTrial ();
     else if (strcmp (argv[Tab1], "-x") == 0)
       MDTestSuite ();
     else MDFile (argv[Tab1]);
}

/*
 ********************************************************************
 ** End of md5driver.c                                             **
 ******************************** (cut) ********************************
 */
```

Conclusion générale

L'apparition des ordinateurs a aidé le développement de la cryptographie, en commençons par la cryptographie à clef secret jusqu'à la cryptographie a clef publique qui a donné naissance aux PKI.

Le PKI est une révolution dans le domaine de sécurité de l'information, car il contribue dans la sécurité des échanges de monnaies électronique et dans la sécurité des applications pour les smartphones et dans la constitution des VPN et dans autre domaine.

Les PKI comme on a dit dans le chapitre précédent sont difficile à crée, car il nécessite des équipements matérielle et logiciel, mais avec nos propre moyens simple on a essayé de présenter les choses.

Dans ce travaille modeste on a développé un PKI. A cet effet, nous avons développé une application de génération de clefs et certificats électroniques, on a mis notre propre tache, on ajouton des choses qui facilites la taches aux user comme l'annuaire des nombre premier et les étapes claire qui montre aux user les étapes de génération des clefs « publique, privé » par le RSA et en a reliée ça avec son certificat numérique qui a rempli ces propre information.

On espère que ce travaille modeste vous a plu et qu'il était expliqué facilement et par détaille et que les démarche de réalisation on était simple et facile pour vous, et que le code sources étais claire et bien facile et bien détaillé.

Merci et on vous souhaite de bon courage dans vos projets de fin d'étude.

REFERENCE

[1] Jonathan Katz, Yehuda Lindell, Introduction to Modern Cryptography: Principles and Protocols (Chapman & Hall/CRC Cryptography and Network Security Series) , August 31, 2007.

[2] R.L. Rivest, A. Shamir, and L. Adleman « A Method for Obtaining Digital Signatures and Public-Key Cryptosystems » Communications of the ACM, 1978.

[3] Jonathan Katz, Université du Maryland, College Park « Cryptographie ».

Conférence online : https://www.coursera.org/course/cryptography

[4] Comment ça marche, Encyclopédie, Sécurité / Législation

Lien : http://www.commentcamarche.net/contents/securite-legislation-9

[5] Encyclopédie Wikipédia, Portail de la cryptologie**,** Chiffrement
Lien : http://fr.wikipedia.org/wiki/Chiffrement

[6] David Eklund, luau (Lib Update/Auto-Update) : Simple Update Library, Copyright (C) 2003.

Code C de MD5 signature numérique, lien : http://goo.gl/YNzaHw

[7] The Euclidean Algorithm and the Extended Euclidean Algorithm.

Lien : http://www.di-mgt.com.au/euclidean.html