REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTRE DE L'ENSEIGNEMENT SUPERIEUR

ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE SAAD DAHLEB BLIDA

Faculté des sciences
Département : Informatique

CDTA

Mémoire

Pour l'obtention du diplôme de

Master

En informatique

Option : sécurité des systèmes d'information

Présenté par :

BOUDERBALA Fatma Zohra

THEME

Mise en place d'une solution de confiance basée sur les PKI pour l'intégration de la signature électronique dans La plateforme de numérisation du CDTA.

Devant le jury composé de :

Mme GHABGOUB Yasmine	USDB	Promotrice
Mme BOUZIDI Fatma Zohra	CDTA	Encadreur
- Mme BOUSTIA Narhimene	USDB	Président
-M. BENYAHIA Mohamed	USDB	Examinateur

Année Universitaire 2019/2020

REMERCIMENT :

Avant tout, je remercie **ALLEH** le tout puissant de m'avoir guidé, aidé et donné la foi, la force et le courage pour accomplir ce travail.

En préambule à ce mémoire, il m'est agréable de citer et adresser mes remerciements les plus sincères aux personnes qui m'ont apporté leurs aides et qui ont contribué à l'élaboration et au bon déroulement de ce travail :

A ma promotrice **M^{me} GHABGHOUB Yasmine**

A mon encadreur **M^{me} BOUZIDI Fatma zohra**
Je tiens aussi à remercier **M^{me} SEMAR Kahina** chef de département au centre de développement des technologies avancées.
Je désire aussi remercier **Mr REZOUG Amar** Professeur à l'École Nationale Supérieure de technologie d'Alger.

« Mes remerciements les plus respectueux de m'avoir aidé dans ce travail, le regard critique , juste et avisé que vous aviez porté sur mes travaux ne peut que m'encourager a être engagée dans mes recherches »

À tout le corps enseignants et le personnel du département génie informatique qui ont contribués de près ou de loin à ma formation.

Aux membres de jury qui auront à juger et ce travail et d'avoir accepté de l'examiner.

À la fin, il nous est agréable d'adresser nos vifs remerciements à tous ceux qui m'ont aidé de près ou de loin à élaborer ce mémoire.

DEDICACE :

Je dédie ce modeste travail en signe de reconnaissance et de respect à:

Mes chandelles de vie, qui m'ont donnée la vie et qui ont toujours été là pour moi.

« Vous avez tous sacrifié pour vos enfants n'épargnant ni santé ni efforts. Vous m'avez donné un magnifique modèle de labeur et de persévérance. Je suis redevable d'une éducation dont je suis fière »

Mon mari REZOUG Amar

«Aucun mot ne saurait t'exprimer mon profond attachement et ma reconnaissance pour l'encouragement et le soutiens dont tu m'as toujours entouré. Cher mari j'aimerai bien que tu trouves dans ce travail l'expression de mes sentiments de reconnaissance les plus sincères car grâce à ton aide et à ta patience avec moi que ce travail a pu voir le jour»

Mes deux sources de joie mes enfants Ayham Abdelillah & Mohamed Djawed.

«Vous resterez pour toujours le rayon du soleil qui égaye ma vie»

Mon frère Kamel, mes deux perles frangines : Ichrak, Maroua, ma chère belle sœur SOUMIA & mon petit poussin Mohamed Rassim.

« Qui m'ont toujours entouré et motivé sans cesse tout le long de ce projet, a qui je souhaite un avenir radieux plein de réussite »

Tous ceux qui portent le nom « BOUDERBALA » et «REZOUG»

Aux personnes qui étaient toujours à mes côtés, mes aimables amies, mes collègues.

À tous ces intervenants, je présente mon respect et ma gratitude.

Résumé

La signature numérique est un mécanisme permettant de garantir l'intégrité d'un document électronique et d'en authentifier l'auteur, par analogie avec la signature manuscrite d'un document papier. La signature numérique repose sur la technique de la cryptographie (asymétrique), c'est-à-dire, le document est chiffré par deux clés. Le présent travail propose une solution de signature numérique basée sur les PKI pour le Centre de Développement des Technologies Avancées garantissant l'identité du signataire, ainsi que l'intégrité et la confidentialité, des documents échangés conférée par la signature numérique de tous formats (word/pdf/txt). En premier lieu, on a présenté le domaine de la signature numérique et la cryptographie. En second, on a fait le choix d'intégrer un provider de sécurité afin d'utiliser les standards PKCS et fournir les algorithmes de hachage et de cryptographie, où nous avons choisi l'algorithme SHA pour le hachage et l'algorithme RSA pour le cryptage asymétrique en basant sur PKCS12 pour l'utilisation des certificats et l'échange de clés. En dernier la solution a été mis en place.

Mots clés : signature numérique , document électronique , cryptographie asymétrique, PKI, PKCS, PKCS12, certificats, échange de clés

Abstract

The digital signature is a mechanism that guarantees the integrity of an electronic document and authenticates the author, by analogy with the signature of a paper document. The numerical signature is based on the technique of cryptography (asymmetric), i.e. the document is encrypted by two keys. The present work proposes a digital signature solution based on the PKI for the Center for the development of Advanced Technology guaranteeing the identity of the signatory, as well as the integrity and confidentiality of exchanged documents conferred by the digital signature of all formats (word / pdf / txt). First of all, we studied the field of digital signature and cryptography. Second, we made the choice to integrate a security provider in order to use the PKCS standards and provide the hashing and cryptography algorithms, where we chose the SHA algorithm for the hashing and RSA algorithm for the asymmetric cryptography based on PKCS12 for the use of certificates and the exchange of keys. Lastly, the solution was realized.

Keywords: digital signature, electronic document, asymmetric cryptography, PKI, PKCS, PKCS12, certificates, key exchange

ملخص

التوقيع الرقمي هو آلية لضمان سلامة الوثيقة الإلكترونية والمصادقة على المؤلف ، بالقياس مع التوقيع بخط اليد على المستند الورقي. يعتمد التوقيع الرقمي على تقنية التشفير (غير المتماثل) ، أي أن الوثيقة مشفرة بمفتاحين. يقترح هذا العمل حلاً للتوقيع الرقمي يعتمد على البنية التحتية للمفاتيح العمومية لمركز تطوير التقنيات المتقدمة لضمان هوية الموقّع ، فضلاً عن سلامة وسرية المستندات المتبادلة الممنوحة بالتوقيع الرقمي لجميع التنسيقات . أولاً ، درسنا مجال التوقيع الرقمي والتشفير. ثانيًا ، قمنا باختيار دمج موفر الأمان من أجل استخدام معايير وتوفير خوارزميات التجزئة والتشفير ، حيث اخترنا خوارزمية للتجزئة وخوارزمية لـ تشفير غير متماثل قائم على لاستخدام الشهادة وتبادل المفاتيح. أخيرا تم تنفيذ الحل.

الكلمات الرئيسية: التوقيع الرقمي ، المستند الإلكتروني ، التشفير غير المتماثل ، الشهادات ، تبادل المفاتيح

Table des matières:

CHAPITRE IV La cryptographie dans le langage JAVA et le package Bouncy Castel

CHAPITRE V

Liste des figures

Liste des tableaux

Introduction générale

En Algérie, la numérisation de l'administration est en plein essor dont plusieurs services publics sont devenus totalement ou partiellement numérisés. Il est évidant que la digitalisation des services a un d'impact économique et social avéré à savoir, permettre aux usagers d'accéder plus rapidement à de multiples services sans effort de déplacement, permettre de limiter l'utilisation de la paperasse ce qui va engendrer par conséquent des économies d'achat du papier et de le gérer soient par son brulement en cas de fin de validité ou bien son archivage. Malgré tous les progrès réalisés, que ce soit en Algérie ou bien à l'étranger, ce champ de développement reste toujours ouvert et d'actualité dont beaucoup de chantiers restent inachevés ou pas encore entamés.

Dans toute organisation la dématérialisation des tâches constitue un véritable moyen pour gagner du temps, être efficace, et faire des économies substantielles. Par exemple, lorsque un individu, quel que soit son rang dans un établissement, doit imprimer un document en vue le signer puis le scanner à nouveau, une signature électronique va lui permettre de gagner du temps et de se consacrer à d'autres tâches.

Le Centre de Développement des Technologies Avancées (CDTA) a envisagé de s'en servir des avantages de la numérisation afin de réduire les dépenses, gagner du temps et de la flexibilité. La dématérialisation de nombreuses procédures administratives est en phase de réalisation en donnant naissance à la numérisation des actions de la structure et entre autres intégration de la signature numérique dans les échanges des documents administratifs. Avec de telle initiative, il ne s'agit plus simplement de vérifier l'identité d'un individu ou de garantir qu'une information ne peut être accessible que par les personnes autorisées. Il convient désormais également de s'assurer que, d'une part les données n'ont pas été modifiées par un tiers, et d'autre part qu'une organisation ou qu'un individu impliqué dans un échange ne peut pas nier son rôle dans l'échange. En d'autres termes, la confiance doit être assurée.

Le travail qu'a été demandé consiste à proposer et réaliser un système de signature numérique des documents administratifs internes de différents formats (PDF, Word, txt...), la solution doit être basée sur les infrastructures à clé publique, dont le CDTA possède une architecture de certification et d'archivage, ce que signifie que la solution doit être basée sur les certificats publiques et l'échange de clés offraient par la PKI. La solution que nous avons développé et que sera exposée le long de ce mémoire de Master consiste à signer des documents en se basant sur la cryptographie a clés publiques et en garantissant l'identité du signataire par l'utilisation des certificats publiques et des certificats personnels de l'extension .p12. Pour

cela, on a utilisé des providers des services cryptographiques (CSP) du langage Java et on a intégré le provider bouncy castel qui se conforme avec le Framework JCE (java cryptographic extention). Nous avons basé la solution proposée sur les standards PKCS, en particulier PKCS1 pour la cryptographie RSA, et PKCS 12 qui définit une syntaxe pour transférer des informations personnelles d'identité (comme les clés privés, les certificats, des secrets quelconques, et des extensions de certificat). On a utilisé également les algorithmes SHA256 pour le hachage et l'algorithme RSA pour le cryptage.

Le mémoire est organisé en Cinque chapitres :

Le premier chapitre est consacré à la définition des concepts de bases et les principes essentiels dans le domaine de la cryptographie.

Le deuxième chapitre présente une vue sur les infrastructures de gestion de clé et la signature numérique.

Le troisième chapitre présente la cryptographie dans le langage Java, la notion de CSP (Cryptographique Service Provider), le provider Bounty Castle, qui sont utilisés pour la réalisation de notre système.

Le quatrième chapitre traite l'analyse des besoins exprimé par le CDTA et la conception de la solution que nous avons proposée.

Le cinquième chapitre regroupe les outils utilisés pour la réalisation de ce système et quelques interfaces de notre application.

Nous terminons ce mémoire par une conclusion générale et quelques perspectives.

Chapitre1: Concepts et principes de la Cryptographie

1 INTRODUCTION

La signature électronique sécurisée est étroitement liée aux technologies de la cryptographie à clé publique ou cryptographie asymétrique. La cryptographie permet de réaliser les objectifs de sécurité, de confidentialité, d'intégrité, d'authentification, et de disponibilité via ses outils de chiffrement. La connaissance de ces concepts est donc indispensable pour la mise en place de ce système de signature. Dans le présent chapitre, nous allons définir les concepts de base et les principes essentiels dans le domaine de la cryptographie.

2 CONCEPTS DE BASE

2.1 La cryptologie

La cryptologie est un mot composé qui tire son origine du grec : cryptos qui signifie secret et logie qui signifie science. En fait, c'est la science du secret et ne peut être vraiment considérée ainsi que depuis peu de temps. Elle englobe la cryptographie, l'écriture secrète et la cryptanalyse, l'analyse de cette dernière[39]. On peut dire que la cryptologie est un art ancien et une science nouvelle : un art ancien car Jules César l'utilisait déjà et il fait son apparition dans l'ancien testament sous la forme du code Atbash ; une science nouvelle parce que ce n'est que depuis les années 1970 qu'elle est devenue un thème de recherche scientifique. Cette discipline est liée à beaucoup d'autres, par exemple la théorie des nombres, l'algèbre, la théorie de l'information, ou encore les codes correcteurs [6].

2.2 Les composants de la cryptologie

a- La cryptanalyse : La cryptanalyse est la technique qui consiste à déduire un texte en clair d'un texte chiffré sans possédé la clé de chiffrement. Le processus par lequel on tente de comprendre un message en particulier est appelé une attaque. Une attaque est souvent caractérisée par les données dont elle a besoin :

- Attaque sur texte chiffré seul (ciphertext-only en anglais) : Le cryptanalyste possède des exemplaires chiffrés des messages, il peut faire des hypothèses sur les messages originaux qu'il ne possède pas. Le cryptanalyse est plus ardue à cause de manque d'information à disposition.
- Attaque à texte clair connu (known-plaintext attack en anglais) : Le cryptanalyste possède des messages ou des parties de messages en clair ainsi que les versions chiffrées. La cryptanalyse linéaire fait partie de cette catégorie.
- Attaque à texte clair choisi (chosen-plaintext attack en anglais) : Le cryptanalyste possède des messages en clair, il peut créer les versions chiffrées de ces messages avec l'algorithme que l'on peut considérer comme boîte noire. La cryptanalyse différentielle est un exemple d'attaque à texte clair choisi.
- Attaque à texte chiffré choisi (chosen-ciphertext attack en anglais) : Le cryptanalyste possède des messages chiffrés et demande la version en clair de certains de ces messages pour mener l'attaque [6].

b- La cryptographie : est l'art de cacher l'information. Elle désigne l'ensemble des techniques permettant de chiffrer des messages, c'est-à-dire permettant de les rendre incompréhensibles [20]. Cela permet la protection de messages ou données en concevant des procédés ou algorithmes de chiffrement utilisant des secrets ou clés, assurant ainsi la confidentialité, l'authenticité et l'intégrité. Elle est utilisée depuis l'Antiquité, mais certaines de ses méthodes les plus importantes, comme la cryptographie asymétrique, datent de la fin du XXe siècle.

Les éléments constituant la cryptographie sont [6]:

➢ **Chiffrement ou cryptage**

Est un procédé de cryptographie grâce auquel on souhaite rendre la compréhension d'un document impossible à toute personne qui n'a pas la clé de déchiffrement.

➢ **Clé de chiffrement**

Paramètre constitué d'une séquence de symboles et utilisé, avec un algorithme cryptographique, pour transformer, valider, authentifier, chiffrer ou déchiffrer des données.

➢ **Déchiffrement**

C'est la méthode ou l'algorithme utilisé pour transformer un texte chiffré en un texte en clair.

➢ **Texte en clair**

C'est le message à protéger

➢ **Texte chiffré**

C'est le résultat du chiffrement du texte en clair.

➢ **Crypto-système**

C'est l'algorithme de chiffrement.

3 Historique de la cryptographie

3.1 Les premières méthodes de chiffrement (Antiquité)

Lors de l'Antiquité, la cryptographie était restreinte à un petit groupe de personnes qui avaient les capacités d'imaginer et de développer une telle idée (n'oublions qu'il n'y avait là aucun manuel ou aucune aide quelconque car la cryptographie n'existait pas encore vraiment). Ainsi, le risque que les messages codés ne soient découverts et décodés par d'autres personnes était faible. En effet, la plus grande partie des gens n'auraient même pas pu imaginer ce concept et même s'ils y avaient songé, les risques restaient minimes étant donné que la majorité de la population était illettrée.[20]

- **1900 av. J.-C.** Un scribe égyptien utilise des hiéroglyphes, qui ne sont pas standards, racontant la vie de son maître. Le but n'était pas de rendre le texte incompréhensible mais plutôt de lui donner un caractère plus solennel.
- **1600 av. J.-C.** Le premier document chiffré connu remonte à l'Antiquité. Il s'agit d'une tablette d'argile, retrouvée en Irak. Un potier y avait gravé sa recette secrète en supprimant des consonnes et en modifiant l'orthographe des mots.
- **600 av. J.-C.** Un roi de Babylone écrit sur le crâne rasé de ses esclaves, attend que leurs cheveux aient repoussé, et les envoie à ses généraux. Il suffit ensuite de raser à nouveau le messager pour lire le texte.
- **600 av. J.-C.** La Mésopotamie, grande civilisation de l'antiquité, avait atteint un niveau cryptologique étonnamment moderne. On a retrouvé en Iran des fragments de tablettes où des nombres correspondaient à des mots.
- **500 av. J.-C**. Des scribes hébreux emploient le ATBASH, un simple algorithme de chiffrement par substitution utilisant l'alphabet renversé, afin de transcrire le livre de Jeremiah. (Par exemple bonjour devient _ ruojnob _).
- **487 av. J.-C.** Des grecs utilisent une scytale, aussi appelé bâton de Plutarque (historien et moraliste de la Grèce Antique). Il s'agit d'un bâton autour duquel on enroulait une longue et mince bande de cuir sur laquelle on écrivait notre message secret (souvent codé). Une fois la bande déroulée, il était difficile de retrouver le message. Seule le destinataire, connaissant le diamètre du bâton de celui ayant servi à écrire le message, pouvait le déchiffrer.

Figure 1: Une Scytale.

La principale faiblesse de ce système est qu'un bâton de diamètre approximativement égal suffit pour déchiffrer le texte.

3.2 Les premiers systèmes cryptographiques (période classique)

Dans le reste du monde la situation restait à peu près semblable jusqu'à environs 50 av. J.-C. L'homme prend enfin l'initiative de développer des techniques de protections de l'information confidentielle de façon plus efficace.

- **150 av. J.-C.** Polybe, historien grec, a imaginé un procédé de chiffrement très innovant pour son temps. Cette méthode utilise un système de transmission basé sur un carré de 25. Les spécialistes en cryptologie moderne ont vu dans cette méthode plusieurs caractéristiques très intéressantes dont la conversion de lettres en chiffres, la représentation de chaque lettre par deux éléments séparés.

- **50 av. J.-C.** Jules César a utilisé une substitution dans l'alphabet pour les communications gouvernementales. En effet, il décalait la lettre qu'il souhaitait coder de 3 lettres vers la droite (en revenant au début de l'alphabet si besoin). ''Le papyrus de Leyde'' est le plus ancien manuscrit connu concernant l'alchimie. Il utilise un algorithme de chiffrement pour cacher les parties importantes de certaines recettes.
- **1499** Jean Trithème (1462-1516), ancien abbé, est considéré comme un des pères de la cryptographie. En effet, il est l'auteur d'un des premiers systèmes poly-alphabétiques et il a créé une technique de sténographie (fait de cacher un message au sein d'un autre) où les lettres sont remplacées par des mots choisis de manière à former, par leur réunion, une prière par exemple.
- **1560** Blaise de Vigenère (1523 - 1596) est l'auteur de l'un des premiers systèmes de substitution poly-alphabétique, il a utilisé donc une clé. Cette méthode restera dominante pendant trois siècles. Sa particularité est qu'il n'utilise non pas un alphabet, mais 26 alphabets décalés pour chiffrer un message. On peut résumer ces décalages avec un carré de Vigenère. Ce chiffre utilise une clé qui définit le décalage pour chaque lettre du message [36].

3.3 La Cryptographie moderne (Temps Moderne)

L'éducation, l'accès à l'information et la connaissance sont devenus accessibles à presque tout le monde, avantageant énormément la créativité et la nouveauté et donc la complexité. Les méthodes de cryptographies se sont décuplées et la difficulté de
cassage du code également. Cependant, la cryptographie a évolué uniquement dans des milieux fermés tels, les gouvernements, les services secrets ou les armées. C'est pourquoi, pendant tant d'années, elle est restée une science secrète.

- **1918** Gilbert Vernam met au point l'algorithme ''One Time Pad'' (traduit masque jetable) aussi appelé chiffre de Vernam. En effet, il fonctionne sur le même principe que le chiffrement de Vigenère, avec quelques règles supplémentaires : Une clé ne doit être utilisée qu'une seule fois, elle doit être de la même taille que le message et elle doit être générée aléatoirement. Ce système est donc reconnu comme étant l'algorithme de chiffrement le plus sécuritaire. Cependant, la communication des clés pose problème car n'oublions pas que la clé doit être de la même taille que le message codé, il est donc hors de question d'utiliser Internet comme passerelle. Pour donner un exemple, on peut penser à la valise diplomatique que les gouvernements utilisent pour communiquer les clés privées de façon sûre à leurs ambassades.
- **1923** Le Dr Arthur Scherbius, hollandais résidant en Allemagne, met au point une machine nommée Enigma qui sert à encoder des messages. Pendant la guerre, des versions d'Enigma sont utilisées pour pratiquement toutes les communications radio allemandes ainsi que pour les communications télégraphiques. Même les bulletins météos sont codés avec Enigma. Elle continuera à être utilisée dans l'armée encore jusqu'en 1939 (date à laquelle le code de cette machine a été cassé).

Enigma est une machine à chiffrer inventée initialement par Arthur Scherbius et Richard Ritter en 1918. Elle se présente sous la forme d'une caisse en bois de 34×28×15 cm, et pèse une douzaine de kilos. Elle est composée :

- D'un clavier alphabétique.
- D'un tableau de connexion.
- De 3 rotors mobiles à 26 positions.
- D'un rotor renvoi à 26 positions (le réflecteur).
- D'un tableau de 26 ampoules correspondant aux 26 lettres de l'alphabet.

Figure 2 : la machine Enigma [10]

- **1976** IBM publie un algorithme basé sur Lucifer (l'une des premières méthodes de chiffrement moderne destiné à un usage civil). Il devient le DES (Data Encryption Standard). C'est un chiffrement qui transforme des blocs de 64 bits avec une clé secrète de 56 bits au moyen de permutations et de substitutions. Le DES est considéré comme étant raisonnablement sécuritaire.
- **1978** Le RSA est inventé par Ronald L. Rivest, Adi Shamir et Leonard M.Adleman. Il est un des plus populaires des systèmes à clés publiques.
- **1992** Le MD5 (Message Digest 5) est développé par Ronald L. Rivest. Il s'agit d'une fonction de hachage très utilisée sur l'Internet, mais n'est pas considéré comme étant un algorithme sûr.
- **2000** L'AES (évolution de l'algorithme Rijndael) devient le standard du chiffrement avancé pour les organisations du gouvernement des Etats-Unis.
- **2005** La cryptographie quantique, qui repose sur la physique quantique, serait considérée comme sûre à presque 100%. Ce sera peut-être la méthode du futur car elle est toujours en cours d'expérimentation[21]

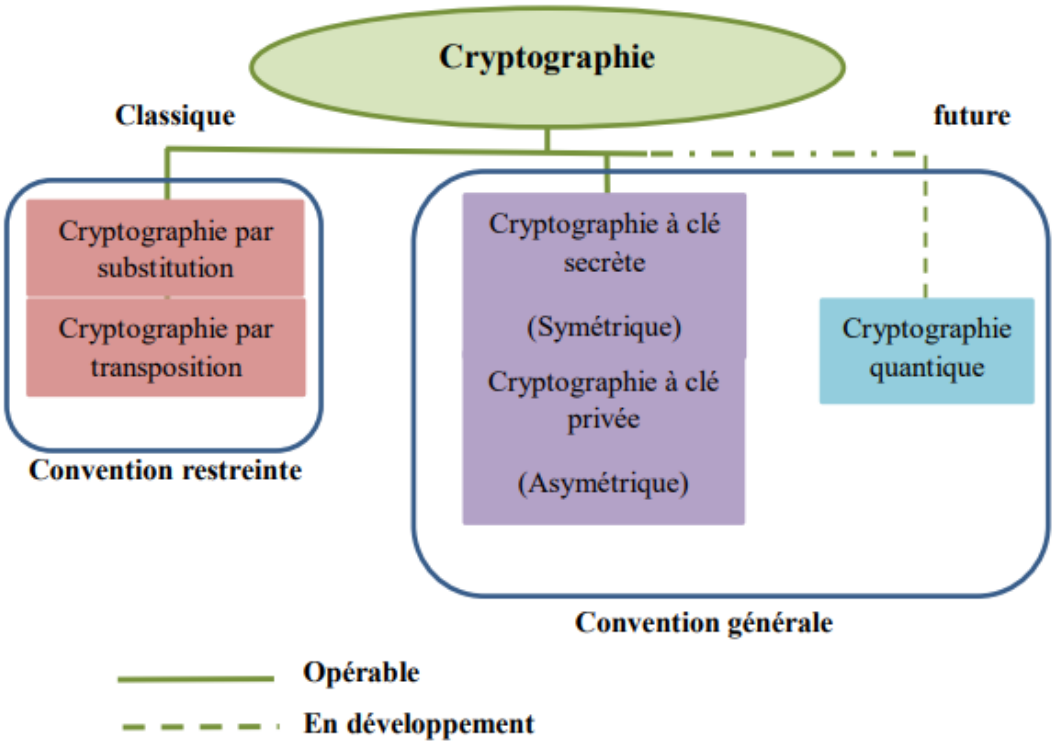

Figure 3 :Les systèmes de chiffrement[29]

4 Fonctionnement de la cryptographie moderne

À côté de la fonction de chiffrement, qui permet de préserver le secret des données lors d'une transmission, et qui a été utilisée depuis très longtemps, la cryptographie moderne a développé de nouveaux buts à atteindre et qu'on peut énumérer de manière non exhaustive: confidentialité, intégrité des données, authentification des divers acteurs, non-répudiation d'un contrat numérique, signature numérique, certification, contrôle d'accès, gestion des clés, preuve de connaissance.

La cryptologie moderne a pour l'objet l'étude des méthodes qui permettent d'assurer les services d'intégrité, d'authenticité et de confidentialité dans les systèmes d'information et de communication. Elle recouvre aujourd'hui également l'ensemble des procédés informatiques devant résister à des adversaires[14].

4.1 La cryptographie symétrique

Le principe de la cryptographie symétrique, encore appelée cryptographie conventionnelle ou à clé secrète est d'utiliser la même clé pour chiffrer et déchiffrer l'information. L'avantage de ce type de cryptographie est la rapidité des processus de chiffrement et déchiffrement. C'est pour cette raison qu'elle est largement utilisée pour protéger des données de taille importante. Cependant, utiliser la cryptographie conventionnelle pour la transmission des messages peut

rapidement revenir très cher, à cause de la difficulté de partager la clé secrète avec un destinataire que l'on ne connaît pas ou avec qui l'on n'a aucun contact physique.

La cryptographie symétrique est donc recommandé pour le stockage de données car le chiffrement est très rapide, mais il est vivement déconseillé de l'utiliser seule sans autre moyen de distribution de clés dans le cas de données à transmettre.

Les algorithmes de cryptographie conventionnelle les plus connus sont DES (Data Encryption Standard), 3DES, AES (Advanced Encryption Standard), RC4 (Ron's Code), RC5[12].
le DES (Data Encryption Standard) est L'algorithme symétrique le plus célèbre qui fonctionnait avec des clés de 64bits remplacé par l'AES (Advanced Encryption System, qui fonctionne avec des clé allant jusqu'à 256 bits)[39].

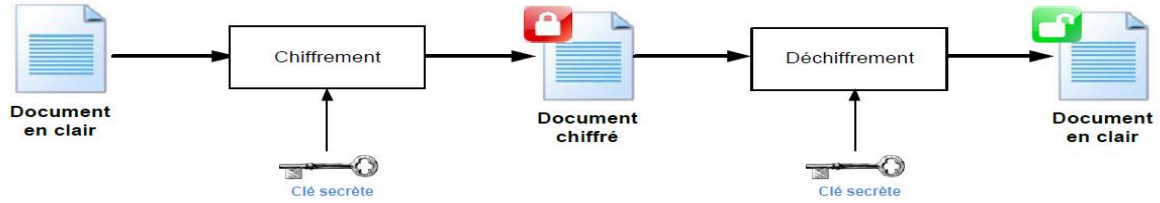

Figure 4: Principe du chiffrement symétrique

La limitation de l'utilisation de la cryptographie symétrique réside dans la problématique de la transmission en toute confidentialité de la clé secrète. Si le réseau est composé de n nœud alors il faudra gérer n.(n-1)/2 clés, ce qui ne s'adapte pas au facteur d'échelle. Avec 500 nœud, on arrive déjà à plus de 12 millions de clés à gérer[12]

Deux grandes catégories de systèmes de chiffrement à clés ~~clef~~ secrètes :

- Le chiffrement par flux. [41]
- Le chiffrement par blocs.

4.1.1 **Le** chiffrement par flux (stream cipher)

Le chiffrement par flux est un chiffrement a clé symétrique permet de traiter des donnés de longueur quelconque. Il fonctionne en générant, à partir de la clé K, une suite de symboles, appelée suite chiffrante, de la même longueur que le message à chiffrer. Les bits du texte clair sont généralement combinés par opération XOR avec un flux de bits pseudo-aléatoire (*keystream*).Un des algorithmes de chiffrement par flot le plus repandu est RC4, il a été conçu en 1987 par Ronald Rivest [9]. Ces algorithmes sont généralement plus

rapides mais moins résistants que les chiffrements par blocs.

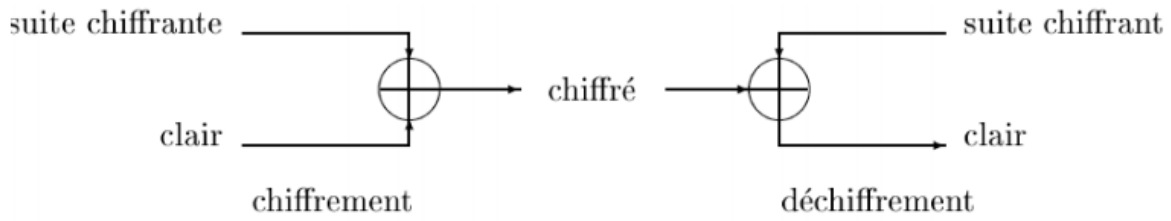

Figure 5: mécanisme de chiffrement par flot[8]

4.1.2 Le chiffrement par bloc (Block Chiper)

Le chiffrement par bloc permet de travailler sur des blocs de taille fixée. Le texte clair est préalablement découpé en "blocs de message" qui sont traités séparément. Comme la plupart des microprocesseurs traitent des mots de plusieurs bits (de 32 bits, souvent), cette opération s'avère rentable une fois mise en œuvre. Dans la pratique, on utilise des modes de chiffrement hybrides : on considère le texte clair comme un flot de blocs de message qui sont traités "au vol". On appelle "mode opératoire" ce mécanisme de traitement `a partir d'une fonction de chiffrement par blocs[8].

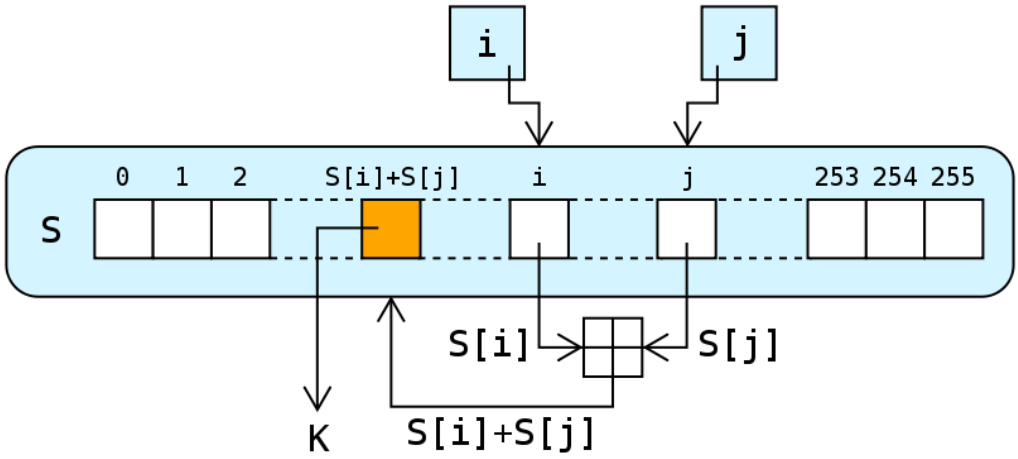

Figure 6: mécanisme de chiffrement par bloc

Un exemple de taille de bloc et de clé utilisés par les algorithmes les plus connus:

- DES : blocs de 64 bits, clé de 56 bits.
- IDEA : blocs de 64 bits, clé de 128 bits.
- AES : blocs de 128 bits, clé de 128 à 256 bits.

Pour cette catégorie, nous allons présenter deux algorithmes très connus DES et AES en mettant l'accent sur l'AES, car il est devenu le standard recommandé pour le chiffrement symétrique[32].

4.2 La cryptographie asymétrique

Le fonctionnement repose sur une paire de clés liées mathématiquement. On considère l'une d'entre elle comme privée, et l'autre comme publique (qui pourra donc être diffusée). Le contenu de la première clé ne peut être retrouvé à partir de la seconde clé. Les opérations se font à sens unique, on ne peut revenir en arrière (principe de la trappe). Ce retour en arrière est toutefois possible en possédant la clé privée[5].

Un cryptosystéme à clé publique se comporte comme un coffre-fort dont seule une personne possède la clé. Il laisse son coffre ouvert à disposition de toute personne désirant lui envoyer un message, celle-ci referme lors la porte et seul le destinataire peut ensuite l'ouvrir. En pratique, le destinataire publie à l'intention de ceux qui veulent lui envoyer des messages. C'est une méthode de chiffrement que lui seul est capable de déchiffrer, on voit donc bien pourquoi ces systèmes sont dits asymétriques[16].

L'un des principaux avantages de la cryptographie de clé publique est qu'elle offre une méthode d'utilisation des signatures numériques. Les algorithmes de cryptographie asymétrique les plus utilisés sont : Diffie-Hellman, RSA (Ron Rivest, Adi Shamir et Len Adleman), Elgamal, DSA (Digital Signature Algovrithm) [45].

Figure 7: Chiffrement Asymétrique [14]

Le problème posé dans cette méthode de cryptographie est comment garantir qu'une clé publique correspond bien à l'entité avec qui on communique ? si la clé publique n'est pas distribuée d'une manière sécurisée. Un schéma asymétrique peut subir une attaque de type "Man in the Middle", une telle attaque est illustrée dans le scénario ci-après :

Figure 8: Man in the middle[12]

La solution au problème dit "man in the middle" est l'usage d'un certificat numérique qui assure la liaison entre l'identité et la clé publique correspondante dans un document numérique signé par une tierce partie de confiance dite autorité de certification.

4.2.1 Les algorithmes de chiffrement asymétrique

Un chiffrement asymétrique est défini par trois algorithmes

- Algorithme de génération des clés,
- Algorithme de chiffrement,
- Algorithme de déchiffrement.

Certains algorithmes asymétriques comme RSA offrent aussi des opérations pour la génération de signature numérique et sa vérification.

L'utilisation d'une paire de clés publique/privée permet d'assurer la confidentialité, l'authentification, l'intégrité et l'échange de la clé secrète. Cependant, les algorithmes asymétriques ne réalisent pas tous ces fonctions[14]. La Table II.1 donne un résumé des opérations cryptographiques pouvant être réalisés par les algorithmes asymétriques les plus connus.

Tableau 1: Les applications des crypto systèmes asymétriques[5]

Algorithme	Chiffrement/Déchiffrement	Signature numérique	Echange de clé
RSA	Oui	Oui	Oui
Diffie-Hellman	Non	Non	Oui
DSA	Non	Oui	Non
EC-Elliptic Curves	Oui	Oui	Oui

a- Cryptage RSA :

Le cryptage RSA, du nom de ses concepteurs, Ron Rivest, Adi Shamir et Leonard Adleman, est le premier algorithme de chiffrement asymétrique. Il a été découvert en 1977 au Massachusetts Institute of Technologie. Le principe de ce cryptage est d'utiliser une clé publique pour crypter les données et une clé privée qui servira à les décrypter[14].

Les opérations de chiffrement et de déchiffrement dans le protocole RSA sont basées sur un calcul d'exponentiation modulaire, définies respectivement par les expressions : C =ME mod N (1) M =CD mod N (2) Où M est le message en clair, C est le message chiffré et N est le modulo. Ces deux opérations sont généralement exécutées après une étape de générations de deux clés. La première (E, N) est rendue publique. La seconde (D, N) est privée. Les deux exposants E et D sont calculés comme suit :

1. Génération deux nombres premiers p et q.

2. Calcul du modulo N, tel que N=p×q.

3. Calcul de la fonction d'Euler: Φ (N)= (p-1) × (q-1).

4. E est choisi, tel que 2 < E [30]

b- Protocole d'échange de clés DH (Diffie-Hellman) :

C'est un algorithme à clé publique d'échange de clé, basé sur le problème de logarithme discret, développé par Diffie et Hellman en 1976. Ce protocole permet à deux tiers de générer un secret partagé sans avoir aucune information préalable l'un sur l'autre et en échangeant uniquement leurs clés publiques[14].

c- DSA (Digital Signature Algorithm):

L'algorithme de Diffie-Hellman permet de créer un secret commun, mais contrairement à RSA, il ne permet pas signer des documents. C'est pour cette raison que l'algorithme de Diffie-Hellman est souvent associé à DSS (Digital Signature Standard, un autre algorithme) ou DSA (DSS permet de signer les documents). Le DSA est une norme promulguée par le NIST. Il est uniquement utilisé pour les signatures numériques. DSA génère des signatures plus rapides, et peut vérifier les signatures RSA[14].

4.3 La cryptographie hybride

Une solution peut d'utiliser les deux systèmes de cryptographie et de prendre l'avantage de chacun. C'est ce qu'on appelle la cryptographie hybride, elle utilise la cryptographie à clé publique pour échanger la clé secrète qui va être utilisée pour le chiffrement des données. Cela a l'avantage de protéger la clé secrète et d'être rapide car c'est la cryptographie symétrique qui va être utilisée pour le chiffrement [4].

5 Notion de hachage

Une fonction de hachage est une fonction qui transforme en un résumé court, de taille fixe. L'image d'un message par une fonction de hachage s'appelle le condensé du message, L'empreinte du message, le résumé du message ou encore le message haché. Une fonction de hachage doit posséder deux qualités indispensables:

_ Résistance à la détermination d'une primage, ce qui signifie qu'il doit être impossible en pratique, à partir d'un résumé m, de trouver un message M ayant ce résumé tel que m=h(M).

_ Résistance aux collisions, ce qui signifie qu'il est impossible en pratique de construire deux messages M1 et M2 ayant le même résumé : h(M1)=h(M2) [5].

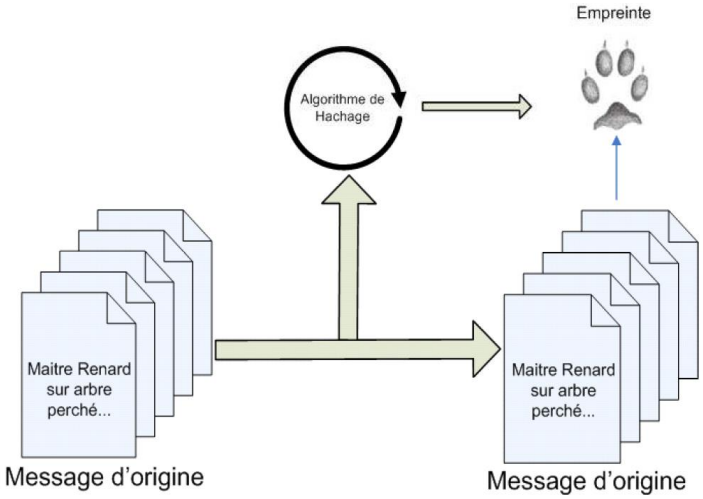

Figure 9: fonctionnement de hachage [21]

Les algorithmes de hachage les plus utilisés sont :

- **MD4** : Message Digest 4 (RFC 1320) qui génère une empreinte de 12 bits. Cet algorithme est désormais abandonné à cause d'une faiblesse de conception et tant la probabilité d'avoir le même résultat pour deux messages différents est important.
- **MD5** : Message Digest 5 (RFC 1321) améliore MD4, l'empreinte reste sur 128 bits. Mais, cet algorithme est désormais considéré comme non sûr pour l'usage cryptographique : une équipe de Chinois a pu démontrer pouvoir reproduire à partir d'une empreinte e1 (calculée à partir d'un message X), un nouveau message (Y) capable de produire une empreinte e2 identique à e1. Ce qu'on appelle une collision complète (dans un contexte peu sécurisé puisque le second message n'a pas été trouvé de manière aléatoire…). Il n'empêche qu'il est encore fortement utilisé pour vérifier l'empreinte de fichiers téléchargés sur Internet, pour le cryptage de mot de passe sur les sites Web (fonctions PHP disponibles).
- **SHA-0 :** Secure Hash Algorithm 0, cette première version mise au point en 1993 fut vite abandonnée à cause de deux failles permettant des collisions.
- **SHA-1** : Secure Hash Algorithm 1 est sorti en 1995 sous la coupelle de la NSA, il produit une empreinte de 160 bits, mais reste parmi les algorithmes peu sûrs, bien qu'il possède une puissance de calcul importante pour permettre une collision qui ne sera trouvée à partir d'un message aléatoire.
- **SHA-2** (aussi appelé SHA-224, SHA-256, SHA-384, SHA-512) Secure Hash Algorithm C'est un algorithme dérivé de SHA1, publié par la NSA en 2000, les versions 256 et 512 sont répandues et dits sûrs, une étude de 2003 a montré que l'algorithme n'avait pas la fragilité des algorithmes rencontrés sur MD5 et SH1.
- **Whirlpool :** conçu dans le cadre d'un projet européen, cet algorithme génère des empreintes de 512 bits. La longueur de l'empreinte et le fait que l'algorithme travaille sur des registres 64 bits en font un système assez consommateur pour le processeur et la mémoire (notamment sur environnements embarqués et basés sur des architectures 32 bits), mais il reste particulièrement robuste, il est normalisé ISO 10118-3 en 2004 pour sa version finale et il est libre de droits.

D’autres algorithmes de hachage peuvent être rencontrés mais présentent pour la plupart : 1- une implémentation peu rencontrée, 2- des failles (collisions) mises en évidence. Citons notamment FFTH, RIPEMD, LANMAN Hash, Tiger [21].

6 Conclusion

Un concepteur de système cryptographique est toujours en train d’essayer d’élaborer un système de chiffrement plus sûr mais au même temps des intrus essayent de casser ce dernier, ils se livrent constamment une bataille, mais les enjeux sont toujours énormes.

Dans ce chapitre, nous avons défini les concepts de base de la cryptographie, les différents algorithmes de cryptage symétrique et asymétrique ainsi que les algorithmes de hachage. Dans le chapitre qui suit, on va définir la gestion de privilèges basée sur les certificats d'identité avec aperçu sur les infrastructures de gestion de clé publique ainsi qu’une étude sur les différents types et formats de la signature numérique.

CHAPITRE II infrastructure a clé publique et signature numérique

1- Introduction

Dans le présent chapitre, on montra l'intérêt des infrastructures de gestion de clé pour la gestion des privilèges basé sur les certificats d'identité. Ensuite, on présentera une étude descriptive sur les différents standards de clé publique ainsi que le mécanisme et les différents formats de la signature numérique.

2- La gestion des privilèges

2-2 Les modèles classiques de gestion des privilèges

2-2-1 Le contrôle d'accès mandataire (MAC)

Dans les politiques mandataires (MAC), le contrôle d'accès est effectué sous l'égide du système plutôt que des utilisateurs. Le sujet n'a accès à une information que si le système l'y autorise. Les règles d'une politique mandataire différent selon qu'il s'agisse de maintenir des propriétés de confidentialité ou d'intégrité. Les politiques mandataires les plus souvent utilisées sont des politiques multi-niveaux (MLS), qui reposent sur la notion de classes de sécurité. Le modèle de Bell-LaPadula (1973), qui a été utilisé pendant longtemps dans les systèmes militaires, est l'un des plus influents pour la confidentialité. Pour l'intégrité, nous citons le modèle de Biba (1977), le modèle de Clark et Wilson (1987) ainsi que le modèle de la Muraille de Chine de Brewer et Nash en1989 [38].

Figure 10: Les niveaux de sensibilité dans le modèle MAC[42]

2-2-2 Le contrôle d'accès discrétionnaire (DAC)

DAC (Discretionary Access Control) a été proposé par Lampson et popularisé par le système d'exploitation UNIX. Dans ce modèle ce sont les utilisateurs qui attribuent les permissions sur les ressources qu'ils possèdent. Ils déniassent librement les droits d'accès pour eux, le groupe et les autres utilisateurs. Ce type de mécanisme est utilisé principalement dans les systèmes d'exploitation modernes. Les permissions sont représentées par une matrice, dans laquelle chaque ligne corresponde à un utilisateur et chaque colonne à une ressource. Le contenu de chaque élément de cette matrice définit les droits d'accès (lecture, écriture et exécution) pour l'utilisateur sur la ressource. La mise en œuvre de ce modèle est coûteuse en mémoire lorsque le

nombre des utilisateurs est important. Alors, le regroupement des utilisateurs peut être envisagé afin de limiter la taille de la matrice, par exemple en utilisant le concept de rôles[42]

Tableau 2 :Matrice de permissions DAC

	Objet 1	Objet 2	...	Objet n
Sujet 1	*Lire*			
Sujet 2		*Ecrire*		*Lire*
...				
Sujet n			*Exécution*	

2-2-3 Le contrôle d'accès basé sur les rôles

Dans le modèle de contrôle d'accès basé sur les rôles, ou RBAC, la politique de sécurité ne s'applique pas directement à des utilisateurs comme dans le modèle précédent. Le rôle est ici le concept central de la politique de sécurité. D'un côté les permissions sont accordées aux rôles ; de l'autre, les utilisateurs se voient affecter un ou plusieurs rôles. Les utilisateurs obtiennent les permissions accordées aux rôles qu'ils jouent.

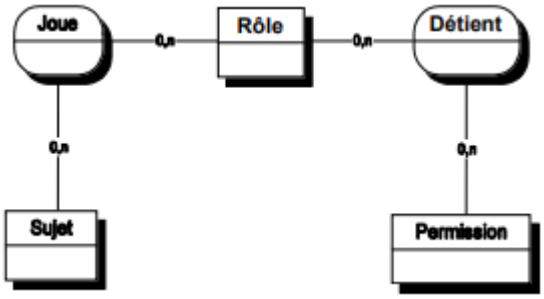

Figure 11 : Le modèle RBAC

Il est possible de raffiner ce modèle en incluant les concepts de session et de hiérarchie de rôles. Dans une même session, un utilisateur a la possibilité de ne pas activer tous ses rôles, mais uniquement un sous-ensemble de ses rôles nécessaires à la réalisation de la tâche à accomplir. La hiérarchie de rôles permet de mettre en place un mécanisme d'héritage des permissions entre les rôles et simplifie d'autant l'administration de ce modèle. Le modèle RBAC est complété par des contraintes. La séparation des pouvoirs, par exemple, peut être exprimée à l'aide d'une contrainte indiquant qu'un utilisateur n'est pas autorisé à jouer simultanément certains rôles dans une même session, comme ceux d'anesthésiste et de chirurgien lors d'une opération. Les inconvénients du modèle RBAC sont les suivants. Tout d' abord, le concept de permission est primitif. En effet, dans le modèle RBAC, rien n'est dit sur l'usage ou la structure des permissions, considérant qu'ils sont dépendants de l'application concrète du modèle. Le concept de hiérarchie de rôles est quelque peu ambigu. Il est en général incorrect de considérer que la hiérarchie de rôles correspond à la hiérarchie organisationnelle [1]

2-3 La gestion de privilèges basée sur les certificats d'identité

2-3-1 Approche générale de l'infrastructure de gestion de clés

L'infrastructure de gestion de clés (IGC) ou Public Key Infrastructure (PKI) permet de sécuriser de façon globale l'accès à un réseau, à des informations et des données. L'IGC est défini par la IETF comme : l'ensemble des algorithmes, protocoles et services utilisés pour gérer et sécuriser les échanges d'information. Elle s'appuie principalement sur la manipulation de certificats d'identité et l'utilisation de la cryptographie[23].

Le PKI (Public Key Infrastructure) repose sur la cryptographie à clé publique. il utilise deux clés différentes, où chaque clé est utilisée pour une opération bien précise. La clé privée est réservée au chiffrement et la clé publique est dédiée à l'opération de déchiffrement. L'autorité de certification CA génère les deux clés, publique et privée simultanément. Elle délivre la clé privée à son propriétaire et garde la clé publique dans sa base de données pour la partager[7].

L'IGC offre les quatre services de base de la sécurité essentiels aux échanges des informations :

1. **Confidentialité :** Assurer le caractère privé de l'information,
2. **Intégrité :** Attester que l'information n'a pas été manipulée,
3. **Authentification :** Attester de l'identité d'un individu ou d'une application,
4. **Non-répudiation :** Assurer que l'information ne pourra être plausiblement désavouée.

L'IGC est utilisée dans des domaines tels que : courrier électronique, e-commerce, réseau privé virtuel, extranet, e-gouvernement, administration électronique, gestion électronique de documents, etc. Elle permet d'assurer de bout en bout la sécurisation des accès et des transferts de données. Plusieurs éléments entrent en jeu dans ce système, notamment les tiers de confiance [23].

2-3-2 Les composants de l'infrastructure de gestion de clés[15]

L'infrastructure de gestion des clés est basée sur plusieurs composants qui sont indispensables à son fonctionnement. Nous répertorions les principaux composants suivants :

- **L'autorité de certification (CA)** : On peut dire que, c'est le composant le plus important de l'infrastructure PKl du fait de son rôle central dans les différentes cinématiques d'échanges à l'intérieur d'une PKl. La CA est chargée de délivrer et gérer les certificats. En effet, elle génère des certificats à clés publiques et assure l'intégrité et l'authenticité des informations contenues en les signant avec sa clé privée. Pour émettre des certificats, elle doit recevoir, au préalable, les requêtes de certification contenant la clé publique de l'entité qui le sollicite.
- **L'autorité d'enregistrement (RA) :** Elle joue le rôle d'intermédiaire entre l'utilisateur et la CA et dépende de cette dernière. Elle a comme responsabilité de vérifier tout ce qui concerne l'utilisateur, son identité, la concordance entre clés privées/publiques, de certifier et d'assurer qu'il possède les droits nécessaires pour demander des certificats. En résumé, cette autorité a pour tâche de gérer les requêtes de certificat qu'elle reçoit des différentes entités et de concevoir les paires de clés qui leur sont spécifiques.

- **Les certificats :** Ils assurent la sécurité d'une clé publique afin d'éviter les failles de sécurité liées à l'usurpation d'identité et à la modification écrite. Leur rôle dans le fonctionnement des PKl sera abordé plus en profondeur dans la suite du document.
- **Les services d'archivage et de publication :** L'archivage est un service qui permet le stockage des paires de clés pour une restitution en cas de perte de la clé privée. En effet, il a pour mission de stocker en toute sécurité les clés de chiffrement émis au sein de l'infrastructure. La publication est un service qui répertorie les différents certificats à clés publiques émis par la CA afin de les rendre disponibles aux éventuels futurs utilisateurs, c'est pourquoi on se réfère communément à lui par le terme de dépôt. Ainsi, un annuaire peut être utilisé (LDAP ou Xsoo par exemple), un serveur Web ou encore un outil de messagerie, etc. Ce service est contraint par plusieurs exigences telles que, par exemple, le délai de mise à jour des listes de révocation ou la disponibilité de ces listes. Le dépôt est également responsable de la publication de la CRL (Liste de Révocation de Certificat).
- **Les utilisateurs :** Ce sont les personnes ou entités organisationnelles ayant émises ou émettant des demandes de certificat, ou souhaitant simplement vérifier la validité et les informations sur l'identité d'un certificat préalablement reçu.

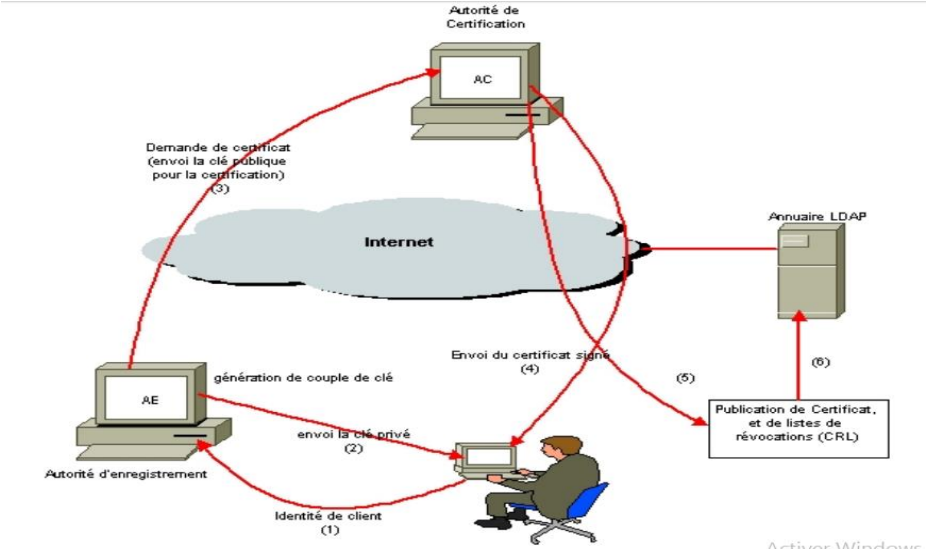

Figure 12: L'Organisation d'une PKI[17]

2-3-3 Le certificat d'identité

Plusieurs définitions existent des certificats électroniques (soit d'identité, soit d'attribut). Nous résumons par : "le certificat électronique est le document émis et signé par un tiers de confiance (organisme certificateur ou un utilisateur normal), associant une clé publique à des informations relatives au propriétaire du certificat". Cette définition de certificat électronique nous semble la plus générale. Particulièrement, le certificat d'identité sert à identifier le propriétaire d'une clé publique. Il est l'équivalent électronique d'un passeport. Il contient

l'information que l'on peut utiliser pour authentifier l'identité du détenteur (exemple : son nom, son adresse IP, etc.)[23].

Dans la section suivante, nous présentons les différents types de certificats d'identité et leurs architectures de confiance :

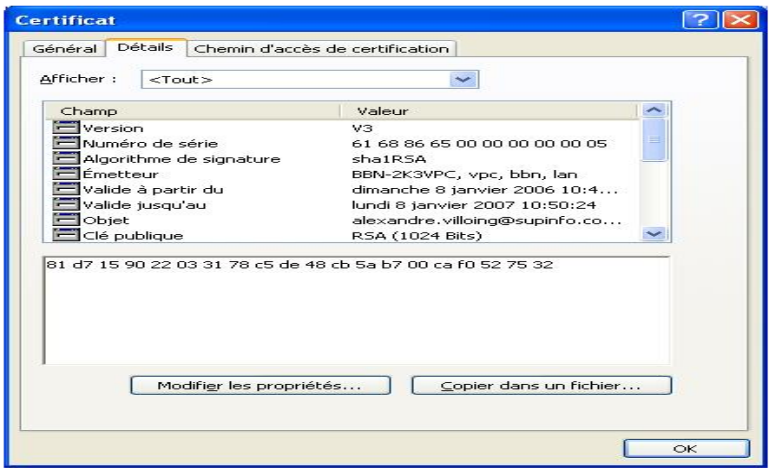

Figure 13: exemple de certificat numérique[44]

2-3-4 Norme X.509

X.509 est une norme de cryptographie de l'Union Internationale des Télécommunications pour les infrastructures à clés publiques (PKI). L'infrastructure de gestion de clé X.509 est reconnue par l'acronyme PKIX (en anglais: Public Key Infrastructure X.509). PKIX est aussi le nom d'un groupe de standardisation de l'IETF [46]. Il a pour but de développer un ensemble de normes qui définit une infrastructure à clés publiques basée sur les certificats X.509 [31]. Ces normes sont orientées pour être utilisées sur Internet.

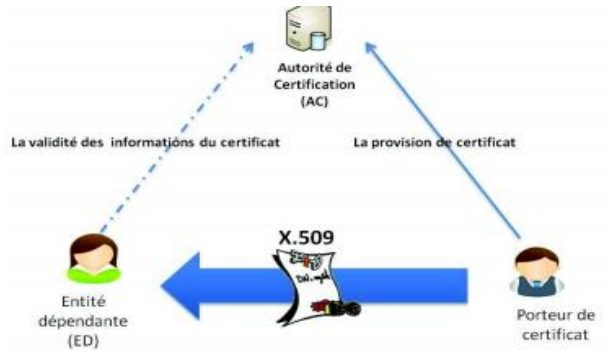

Figure 14 : Modèle de confiance du standard X.509 [45]

2-3-4-1- Génération de certificat X.509

Un certificat est généré après qu'une demande de certification a été initiée par une entité de l'infrastructure à clé publique. La demande est suivie par un enregistrement, sous la responsabilité de l'AR, qui recueille et vérifie l'identité du propriétaire du certificat et toutes les autres informations utiles à la délivrance d'un certificat. Le RA transmet ensuite ces informations à l'autorité de certification qui doit émettre un certificat. Avec les informations

d'enregistrement et la clé publique du propriétaire du certificat, l'autorité de certification peut émettre le certificat. Pour ce faire, il est imposé de signer numériquement le certificat à l'aide de la clé de signature privée de l'AC. Après, que le certificat a été délivré et vérifié comme étant correct par le propriétaire du certificat, il peut être distribué et utilisé par d'autres entités [17].

2-3-4-2- Cycle de vie d'un certificat X509

La figure suivante illustre le cycle de vie d'un certificat de sa délivrance et à sa destruction.

Figure 15 : Cycle de vie d'un certificat numérique

2-3-5- Le certificat X.509 v3

La caractéristique principale de certificat X.509 version 3 est d'attacher une clé à une entité, de cette manière le certificat devient un certificat d'identité. L'objectif de la version 1 était uniquement d'authentifier le propriétaire du certificat et c'est aux applications de déterminer les privilèges du détenteur du certificat. La version 3 de X.509 a permis d'étendre ce principe en ajoutant au certificat des extensions pour associer des privilèges, des informations du certificat ou des informations d'administration du certificat. Les extensions standards de la norme X509 permettent d'indiquer les privilèges suivants :

- **Key Usage :** indique les fonctions possibles du certificat (signature, chiffrement, signature de certificats, signature de liste de certificats révoqués, etc.).
- **Private Key Usage Period** : permet à l'émetteur du certificat d'indiquer une période de validité pour la clé privée.
- **Certificate Policies** : indique une ou plusieurs politiques de certification pour déterminer la politique avec laquelle le certificat a été créé et aussi déterminer les usages du certificat.
- **Policy Mappings** : indique les politiques de l'émetteur (issuerdomainpolicy) et les politiques du propriétaire (userdomainpolicy) du certificat pour les comparer et délimiter les usages du certificat.

- **Subject Alternative Name** : représente un autre identificateur pour le propriétaire du certificat, par exemple un rôle.
- **Issuer Alternative Name** : représente un autre identificateur pour l'émetteur du certificat.
- **Basic Constraints** : indique que le propriétaire du certificat a les privilèges d'un tiers de confiance.
- **Policy Constraints** : utilisé pour indiquer les limites des politiques de certifications.
- **Extended Key Usage** : indique les objectifs du certificat, supplémentaires ou complémentaires à l'extension Key Usage.
- **Authority Information Access :** décrit comment accéder aux services et à l'information du tiers de confiance.
- **Subject Information Access** : décrit les types de services du certificat et comment y accéder, ainsi que les services et politiques du tiers de confiance.

Les extensions sont donc un mécanisme par lequel les certificats peuvent être étendus de façon standard pour inclure des informations additionnelles. Toutes ces informations sont encapsulées et signées par une autorité de confiance (AC).

3- Les standards de cryptographie à clé publique (PKCS)

Les standards de cryptographie à clé publique sont des spécifications écrites par les Laboratoires RSA en coopération avec des développeurs des systèmes sécurisés du monde entier. Publiés pour la première fois en 1991 par un petit groupe de précurseurs dans l'adoption de la technologie des clés publiques, les document ***PKCS*** ont été largement référencés et implémentés. Les contributions venant de la série des ***PKCS*** ont été utilisées dans de nombreux formats et devenus de facto des standards tels que les documents ANSI X9, PKIX, SET, S/MIME et SSL. [37]

a- PKCS #1 - Cryptographie RSA

Ce standard recommande des implémentations de clé publiques basée sur l'algorithme RSA, notamment sur:

- les primitives de cryptographie,
- les plans schématiques de cryptage,
- les plans schématiques de signature avec annexe,
- la syntaxe ASN.A pour représenter les clés et identifier les schémas.

b- PKCS #3 - Accord sur clé Diffie-Hellman

Ce standard décrit une implémentation de l'accord sur clé de Diffie-Hellman, où deux parties, sans aucun accords préalables, peuvent s'accorder sur une clé secrète qui ne sera connue que

par eux seuls. Cette clé secrète peut être alors utilisée par exemple pour chiffrer des communications ultérieures. Les communications visées sont celles transitant par les couches ISO90 a et b de transport et de réseaux.

c- PKCS #5 - Cryptographie par mot de passe

Ce document recommande des implémentations basées sur une cryptographie à mot de passe, couvrant:

- les fonctions de dérivation de clé,
- les plans schématiques de cryptage et d'authentification de messages,
- la syntaxe ASN.1 identifiant les techniques utilisées.

d- PKCS #6 - Syntaxte des extensions de certificat

Ce standard décrit une syntaxe pour formuler les extensions de certificats. Un certificat avec extension, est appelé un certificat étendu. Il consiste en un certificat X.509 d'une clé publique et un jeu d'attributs, le tout signé par l'émetteur des certificats X.509. Les attributs et le certificat peuvent être vérifiés par une simple opération à clé publique, et le certificat X.509 peut être extrait si besoin.

Le but d'inclure un jeu d'attribut est d'étendre le processus de certification à d'autres informations comme une adresse email par exemple. Une liste non exhaustive est disponible dans le PKCS #9. L'application immédiate de ce standard se trouve dans le standard PKC numéro 7 définissant les messages de communication cryptographique.

e- PKCS #7 - Syntaxe des messages de cryptographie

Ce standard décrit la syntaxe générale à utiliser pour des données pouvant avoir été transformée par des procédés de cryptographie, comme les signatures digitales, ou les enveloppes digitales. Ce standard permet aussi l'existence d'attributs, le fait d'être authentifié avec le contenu du message, ou la présence de Co ou contre signatures. Une utilisation dégénérée de cette syntaxe fournie permet de diffuser des certificats et les listes de révocation de certificats.

Ce standard est compatible avec le format PEM (Privacy-Ehanced-Mail), car les données signées ‘signed-data’ et les enveloppes de données signées ‘signed-and-enveloped-data’, construites dans un mode compatible PEM, peuvent être converties en messages PEM sans aucune opération cryptographique. De manière similaire, les messages PEM peut être converti en données signées ou en enveloppes de données signées.Les données en elle-même sont encodées selon la règle BER (Binary Encoding Rule), et sont donc des chaînes d'octets.

Leur envoie par Email, ou par des media ne supportant que l'ASCII est possible en suivant par exemple le format PEM (RFC 1422) ou encore les solutions évoquées dans la RFC 1421.

f- PKCS #8 - Syntaxe d'information sur les clés privées

Ce standard décrit une syntaxe pour donner des informations sur une clé privée. Ces informations incluent la clé privée afin de suivre des algorithmes à clé publique et un jeu d'attribut. Ce standard décrit aussi un standard pour les clés privées cryptées, par un algorithme à base de mot de passe par exemple.

Le fait d'inclure un jeu d'attributs avec la clé est un moyen simple pour un utilisateur d'établir une relation de confiance avec des informations comme le nom distinctif ou la clé publique d'une Autorité de Certification racine. Bien qu'une telle relation de confiance puisse aussi s'établir via des signatures digitales, le chiffrage par une clé secrète connue seulement de l'utilisateur est aussi efficace, et surement plus simple à implanter. Une liste non exhaustive d'attributs est standardisée par le PKCS #9.

g- PKCS #9 - Types d'attributs choisis

Ce document contient des définitions d'attributs standards à utiliser dans les certificats étendus conformes au PKCS #6, dans les informations de clé privée conformes au PKCS #8, et dans les requêtes de certification standard PKC #10.

h- PKCS #10 - Syntaxe de demande de certification

Une demande de certification est composée d'un nom distinctif, d'une clé publique et optionnellement d'attributs, le tout signé par l'entité demandante. Ces demandes sont envoyées à l'autorité de certification, qui transforme alors la requête en un certificat X.509 lié à la clé publique. La manière de retourner le certificat n'est pas abordée ici, ce standard a été pensé comme un application primaire du PKCS #7.

i- PKCS #11 - Interface de jeton cryptographique

Ce standard définie une interface applicative de programmation (API) nommée Cryptoki pour des périphériques contenant des informations de cryptographies et pouvant effectuer des opérations cryptographiques.

Cryptoki se prononce "crypto-key" et est un acronyme de "cryptographic token interface". Son schéma suit une approche objet, en essayant d'être le plus possible indépendant de la technologie utilisée et d'autoriser le partage des ressources (permettant à plusieurs applications d'accéder à plusieurs périphériques). Cette API permet de proposer aux applications une logique commune d'utilisation des périphériques appelés "jetons cryptographiques".

j- PKCS #12 - Syntaxe d'échange d'informations personnelles

Ce standards définie une syntaxe pour transférer des informations personnelles d'identité, comme les clés privés, les certificats, des secrets quelconques, et des extensions de certificat. Ainsi, les applications ou machine (navigateur, café Internet, etc) respectant ce standard seront capable d'importer, d'exporter et d'utiliser un ensemble d'information personnelle d'identité. Le transfert peut se faire directement sous plusieurs modes d'intégrité et d'intimité:

Le plus sécurisé: Ce mode n'est possible que si les plateformes émettrice et réceptrice aient chacune une paire de clés privée/publique de confiance utilisables pour la signature et l'encryptage.

Moins sécurisés: Ces modes sont basés sur des mots de passe et ne sont utilisés que si les paires de clés suscités ne sont pas disponibles.

k- PKCS #13 - Cryptographie à courbe elliptique

Ce standard est encore sous-développement, il abordera de nombreux aspects de la cryptographie à courbe elliptique, comme la génération de la clé et sa validation, les signatures digitales, les encryptages à clés publiques, les accords de clé et la syntaxe ASN.1.

La cryptographie à courbe elliptique est une branche prometteuse de la cryptographie à clé publique, en effet, elle offre potentiellement la même sécurité que les systèmes de cryptographie à clé publique, mais avec des clés de tailles réduites. Depuis son apparition dans les années 1980, beaucoup d'aspects de son implémentation ont été améliorés, la rendant ainsi plus compréhensible et standardrisable.

l- PKCS #15 - Formatage des informations sur les jetons de cryptographie

Les buts des spécifications du standard 15 de cryptographie à clé publique sont de:

- permettre l'inter-opérabilité entre des composants de plate-formes variées, (platform neutral),
- permettre aux applications d'utiliser les avantages de produits et de composants provenant de fabriquants différents, (vendor neutral),
- permettre l'utilisation des avancées technologiques sans re-écrire le logiciel d'application, (application neutral),
- maintenir la cohésion avec les standards existants.

4- La signature numérique

La signature électronique est un code attaché au corps du message à transmettre, elle permet de vérifier l'identité de l'expéditeur. Elle est l'équivalent numérique de la signature manuscrite. Elle est même plus sécurisée que la signature manuscrite, car on peut imiter cette dernière alors qu'il est presque impossible de produire une fausse copie de la signature électronique.

La signature numérique repose sur un système de chiffrement à clé publique et clé privée permettant d'authentifier l'émetteur d'un document. La clé privée sert à signer, la clé publique sert à vérifier cette signature. Les objectifs d'une signature électronique sont les suivants :

- Vérifier l'intégrité d'un message ou d'un document : détecter toute modification éventuelle.
- Authentifier de manière certaine la provenance du message.

Contrairement au chiffrement qui est utilisé à des fins de confidentialité, les signatures électroniques sont, en quelque sorte, annexées aux données et laissent le texte qui vient d'être signé totalement en clair [3]. Pour cela, les propriétés suivantes doivent être vérifiées :

- une signature ne peut pas être falsifiée,
- une signature donnée n'est pas réutilisable sur un autre document,
- un document signé est inaltérable,
- une signature ne peut pas être reniée.

5- Législation en matière de signature et certificat d'identité

5-1- Signature électronique

La loi 15-04, dans son Article 2 Alinéa 1 et au niveau de son Article 6, défini La signature électronique comme un ensemble de données électroniques jointes ou logiquement liées servant de méthode d'authentification de l'identité du signataire et de l'adhésion de ce dernier au contenu de l'écrit sous forme électronique.
La Loi attribue, la qualité d'équivalence à une signature manuscrite, à la seule signature électronique qualifiée (bénéficiant d'un certificat électronique qualifié), tout en ne privant pas toutes autres signatures électroniques de leur efficacité juridique, sauf si à l'origine cette signature était conditionnée par son caractère manuscrit auquel cas sauf la signature électronique certifiée peut assurer cette efficacité (Articles 7,8 & 9).

En sus l'Article 7 ajoute que pour qu'une signature électronique soit désignée comme qualifiée, elle doit satisfaire aux principales exigences suivantes :

- être réalisée sur la base d'un certificat électronique qualifié,
- être conçue par des moyens sécurisés de création de signature électronique, de telle sorte que toute modification ultérieure des données soit détectée,
- être créé par des moyens que le signataire puisse garder sous son contrôle exclusif.

5-2- Certificat électronique qualifié

Tel que défini par la loi type de la Commission des Nations Unies pour le Droit Commercial International (CNUDCI) et la loi n°15-04, le Certificat électronique désigne un document sous forme électronique attestant le lien entre les données de vérification d'une signature électronique qualifiée et le signataire.
Le certificat électronique qualifié est un certificat numérique qui répond aux principales exigences suivantes :

- Etre délivré par un tiers de confiance agréé par l'Autorité gouvernementale de certification électronique pour ce qui est des intervenants dans la **Branche gouvernementale** et par un Prestataire de services agréé par l'Autorité économique de certification électronique pour ce qui est des autres opérateurs et du public,
- Doit contenir principalement :

- Une indication que le certificat électronique est délivré à titre de certificat qualifié,
- L'identification du prestataire de services de certification ou du tiers de confiance autorisé ayant délivré le certificat électronique et le pays dans lequel il / elle, est établie,
- Le nom du signataire ou un pseudonyme pour l'identifier,
- Les données de vérification de signature qui correspondent aux données de création de signature électronique et notamment la Clef cryptographique publique délivrée par l'une des deux Autorités gouvernementale ou économique selon le cas,
- 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 qualifiée du fournisseur de certification électronique ayant délivré le certificat électronique,
- Les limites de valeur des transactions pour lesquelles le certificat électronique peut être utilisé le cas échéant.

La Branche gouvernementale regroupe les institutions, administrations et établissements publiques tels que définis par la législation en vigueur.

6- Les types de signatures

6-1- La signature électronique simple

L'usage de plus en plus répandu de l'ordinateur et l'arrivée des réseaux non protégés comme Internet a donné une nouvelle impulsion à la cryptographie et surtout avec la cryptographie à clé publique.

La cryptographie à clé publique rend possible l'utilisation des signatures électroniques. Celles-ci permettent de corroborer l'origine d'un message. Pour signer un message, on utilise une fonction mathématique. Le résumé obtenu est chiffré à l'aide de la clé privée de l'expéditeur. Le résultat, qui constitue la signature électronique, est annexé au message. Le destinataire du message peut ensuite s'assurer de l'origine du message et de l'intégrité de l'information.

6-2- La multi-signature électronique

La multi-signature électronique aussi appelée signature de groupe se base sur les mêmes principes développés dans la signature numérique. Elle est nécessaire quand plusieurs entités doivent signer un document, par exemple un bon de commande, un contrat de travail, un projet de groupe, etc[12].

7 Le principe de la signature

On extrait une empreinte du document source, que l'on chiffre avec la clé privée. Cela donne la signature électronique qu'on adjoint au document. Pour vérifier la signature, il suffit d'extraire à nouveau l'empreinte du document, de déchiffrer la signature avec la clé publique puis de comparer le résultat à l'empreinte.

Signature : C = Sign S (P)
Vérification de la signature : P = Verif T (C) avec VerifT (Sign S (P)) = P et « S » et « T » des clés

'C' : texte chiffré.
'P' :texte clair. [3]

La confiance dans la signature numérique est donc basée sur :

- La confiance dans les algorithmes le condensat et de chiffrement asymétrique. Si un algorithme est avéré défaillant, par exemple une fonction de condensat qui produit trop de collisions, ou un algorithme de chiffrement asymétrique qui pourrait chiffrer des documents déchiffrable par la clé publique sans recours à la clé privée, tout le système de signature numérique deviendrait caduque.
- La confiance dans la clé, ou plus exactement la confiance dans le fait que seul le signataire est bien la seule personne à disposer de la clé privée.

Concernant les algorithmes de condensat et de chiffrement, des évolutions voient continuellement le jour. C'est pour cela qu'il est conseillé maintenant de signer des documents avec des algorithmes récents (SHA pour l'empreinte et RSA ou IDEA pour le chiffrement), et une longueur de clé suffisante (on utilise actuellement des clés RSA de 2048 bits).

Concernant la confiance dans la clé, il existe (comme souvent) deux méthodes :
- une méthode déconcentrée (GPG/PGP)
- une méthode centralisée (PKI).

Les deux méthodes utilisent un certificat, qui contient entre autre le nom du détenteur, la clé publique et quelques informations complémentaires.

- **La méthode déconcentrée** est basée sur le principe de transitivité de la confiance (les amis de mes amis sont mes amis). On fait confiance dans un certain nombre de personnes (ou plus exactement de certificats) qui vont faire confiance eux aussi dans d'autres certificats.Pour prouver sa confiance dans un certificat, on signe simplement le certificat avec sa clé privée. Bien entendu, ce fonctionnement est récursif et les clés privées associées aux certificats ainsi signées peuvent à nouveau être utilisées pour signer d'autres certificats.

  L'avantage de ce système est qu'il ne repose sur aucun point central. Les certificats sont déposés dans des dépôts mais ces dépôts n'ont pas d'influence directe sur le système de confiance.

  L'inconvénient de ce système réside dans le fait que seul le détenteur de la clé privée associée à un certificat peut révoquer ce certificat. La gestion de la fin de vie des certificats est donc peu fiable. D'autre part, cette méthode donne vie à des « signing

parties » des moments où des personnes se rencontrent uniquement pour vérifier les certificats des autres personnes et les signer pour prouver au monde leur valeur.

- **La méthode centralisée** est basée sur des autorités de certification. Ces autorités peuvent signer des certificats et les révoquer. Ils gèrent pour cela des listes de révocations qui sont alors publiées. La fin de vie des certificats est ainsi beaucoup mieux gérée. Par contre, cela implique que tout le monde fasse confiance dans certaines autorités de certification (ceux généralement préconfigurés dans les logiciels de courriers électroniques) qui ont ainsi un pouvoir immense dans le système de confiance.

On peut illustrer ici de manière très schématique quel est le processus complet de génération d'une signature électronique :

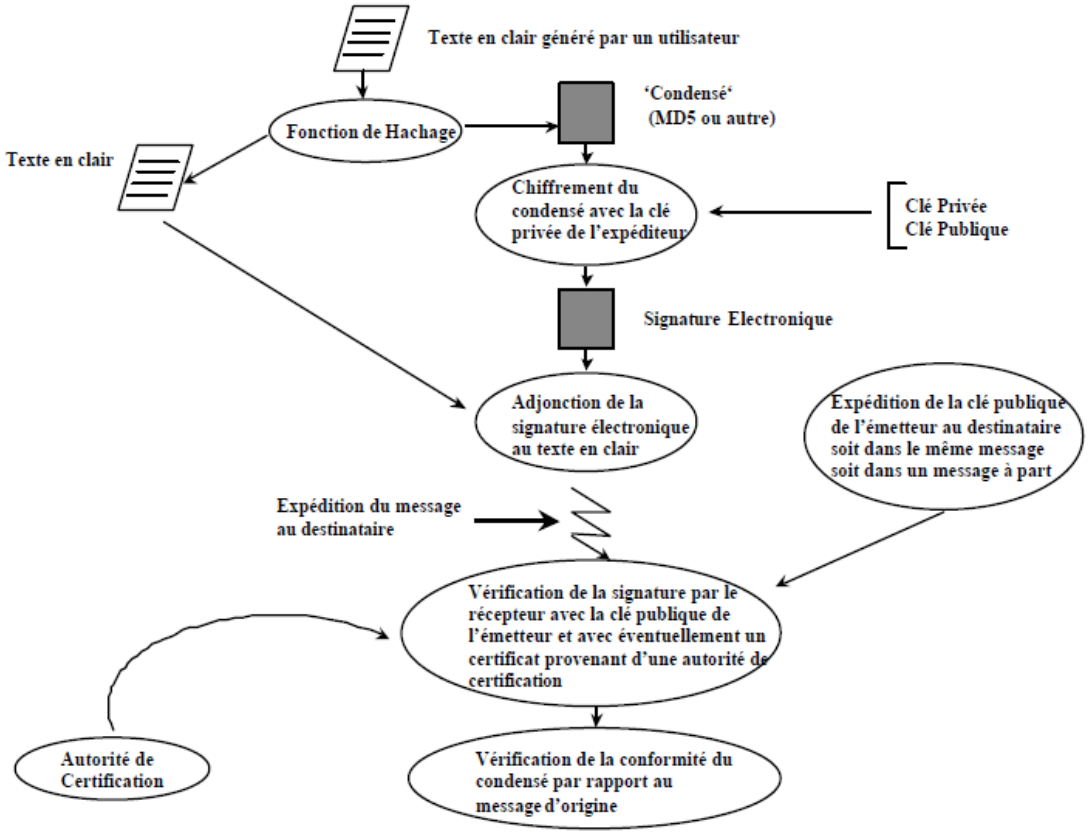

Figure 16: processus complet de génération d'une signature Format de signature de document

8 Format de signature numérique

8.1 PKCS #7, CMS

PKCS#7 v1.5 (Public Key Cryptography Standard), nommé également Cryptographic Message Syntax, est un standard de RSA Labs publié en novembre 1993, qui définit un format pour l'échange ou le stockage de données cryptographiques (signées, chiffrées, signées et chiffrées, etc.) ou simplement l'échange ou le stockage de certificats (dans ce cas, aucune donnée n'est signée ou chiffrée).

Ce standard définit un type de donnée ContentInfo possédant deux éléments : le type de contenu ContentType et le contenu lui-même content.

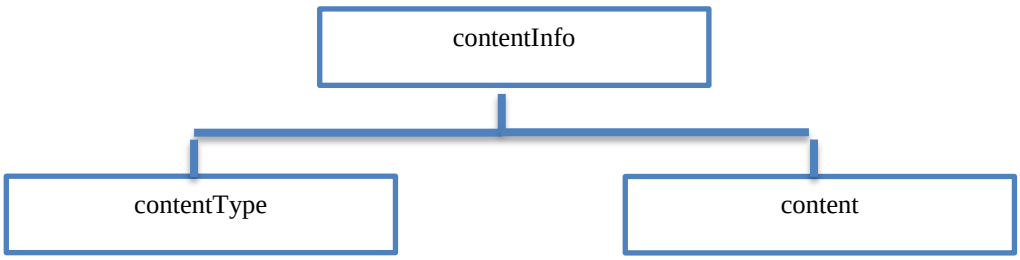

Figure 17 : PKCS#7 - Structure de base

ContentInfo peut avoir 6 types de contenus différents : data, signedData, envelopedData, signedAndEnvelopedData, digestedData, et encryptedData.

Les types envelopedData et encryptedData sont tous les deux utilisés pour l'échange de données chiffrées. Dans le premier cas (envelopedData), on chiffre le message avec une clé aléatoire et on chiffre également cette clé avec la (les) clé(s) publique(s) du (des) destinataire(s). Dans l'autre cas (encryptedData), on chiffre simplement le message avec une clé donnée, et la distribution des clés doit se faire par d'autres moyens.[13]

Analysons le cas où le contenu est de type SignedData :

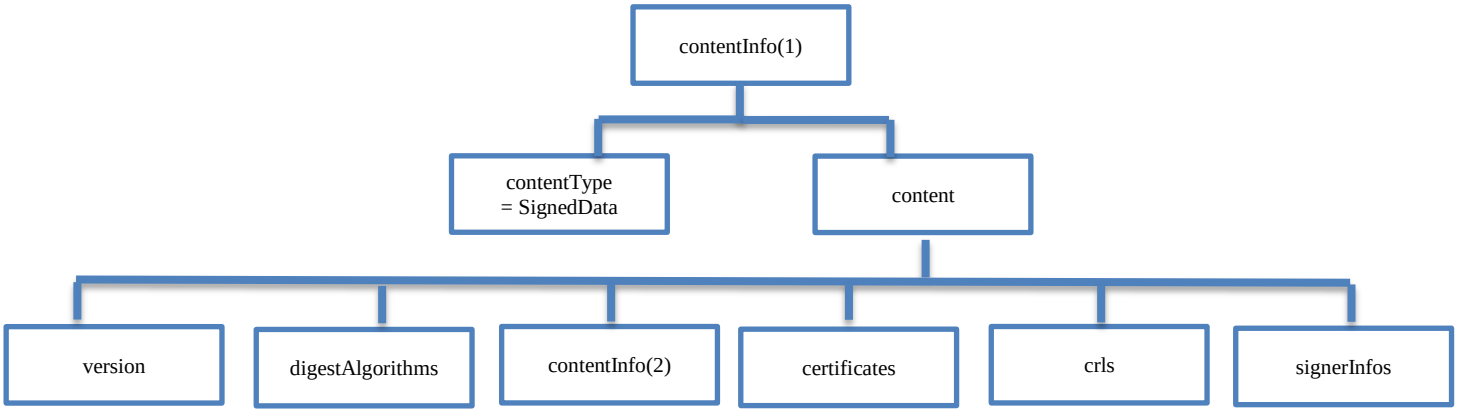

Figure 18: PKCS#7 - Structure détaillée avec signedData

L'élément content contient lui-même un élément contentInfo (2) qui peut contenir un des six éléments déjà cités. Cette flexibilité permet d'imbriquer à volonté des objets contentInfo.

- **version** indique la version de l'objet.
- **digestAlgorithms** contient l'ensemble non ordonné des algorithmes de hachage utilisés pour signer le contenu de contentInfo (2). Plusieurs signataires peuvent utiliser des algorithmes de hachage différents.
- **contentInfo(2)** contient les données à signer. En général, il s'agit d'un objet data (les données originales qu'on désire signer).
- **certificats** est l'ensemble, non ordonné, des certificats des signataires et éventuellement d'autres certificats faisant partie de la chaîne. Les certificats sont placés 'en vrac' dans cet ensemble et il peut y avoir plus de certificats que nécessaire. Il s'agit en quelque sorte d'un magasin de certificats. Lors de la vérification il faudra rechercher la chaîne de certificat pour chaque signataire.
- **crls (certificate revocation lists)** est un ensemble non ordonné de listes de révocation de certificats.
- **signerInfos** contient des informations sur chaque signataire : algorithme de hachage, identification du certificat (nom de l'émetteur du certificat et numéro de série), signature, et éventuellement d'autres paramètres optionnels signés (par exemple la date de signature), ou non signés (par exemple une contresignature). Le type SignedData, signerInfos peut ne pas contenir d'élément (cas particulier où encore aucune personne n'a signé les données, ou lorsqu'on ne veut transmettre que des certificats ou listes de révocation).

Il est également possible de créer une signature externe, c'est-à-dire détaché du message. Dans ce cas contentInfo (2) ne contient pas d'élément. Ceci est pratique pour éviter la réplication des données stockées sur bande ou support non réinscriptible, ou pour des applications réseau où il est nécessaire de ne renvoyer que la signature.

PKCS#7 v1.5 a été publié en mars 1998 dans le RFC 2315 de l'IETF. Depuis lors, l'IETF a repris le travail de développement et de maintenance et ce standard a évolué sous le nom CMS (pour Cryptographic Message Syntax) :

- RFC 2630 [CMS1] (juin 1999),
- RFC 3369 [CMS2] (août 2002),
- RFC 3852 (juillet 2004).

8.2 S/MIME

S / MIME (Secure / Multipurpose Internet Mail Extensions) fournit un moyen cohérent pour envoyer et recevoir les données MIME sécurisées. Basé sur norme Internet MIME populaire.
S / MIME fournit les éléments suivants :

- Services de sécurité cryptographique pour la messagerie électronique.
- Applications: authentification, intégrité des messages et non-répudiation d'origine (à l'aide de signatures numériques) et la confidentialité et la sécurité des données (en utilisant le cryptage).
- S / MIME peut être utilisé par les agents utilisateurs de messagerie (MUA) traditionnels pour ajouter des services de sécurité cryptographique au courrier envoyé et à interpréter les services de sécurité cryptographique dans le courrier reçu.

Cependant, S / MIME n'est pas limité au courrier; il peut être utilisé avec n'importe quel mécanisme de transport qui transporte des données MIME, comme HTTP. [18]

S/MIME permet la signature ou le chiffrement de données MIME[11]. Le standard de courrier électronique sécurisé S/MIME version 2 est basé sur PKCS#7 v1.5 (ou RFC 2315), tandis que S/MIME version 3 est basé sur le RFC 2630. Deux méthodes sont fournies pour la réalisation de la signature :

- soit le contenu MIME est encapsulé dans la signature CMS (dans contentInfo). Dans ce cas, si le logiciel de messagerie du récepteur n'implémente pas S/MIME, le récepteur ne pourra pas lire le message.
- soit le contenu MIME est détaché du message (signature externe). Cette deuxième solution est la plus utilisée car elle permet aux anciens logiciels de messagerie qui n'implémentent pas S/MIME de pouvoir lire le message. La signature se présente comme un fichier attaché.[13]

8.3 XML Signature

« XML-Signature Syntax and Processing » est un standard produit par le groupe de travail XML Signature de l'IETF/W3C. Les signatures XML peuvent être appliquées à tout contenu numérique (objet de données), y compris XML[19]. Il existe trois types de signature :

- **une signature enveloppée** : la signature XML est incluse dans le document XML signé.
- **une signature enveloppante :** la signature XML contient le document signé XML.
- **une signature détachée :** la signature fait référence à un ou plusieurs documents signés qui peuvent être de n'importe quel type.

La signature XML définit une syntaxe XML et des règles de traitement pour signature et vérification de signatures sur une ou plusieurs données. Elle utilise un ensemble de références indirectes à chaque objet de données signé, pour chaque objet signé une

référence qui pointe vers l'objet via un URL, contient une valeur de résumé calculée sur cet objet, l'ensemble complet de références est regroupé sous un élément SignedInfo. [35]

Chaque *Reference* possède :

- **une URI (*Uniform Resource Identifier*)** d'un fichier binaire ou de données XML (sauf si les données XML sont incluses dans la référence-même).
- **une séquence de transformations (*Transforms*)** à effectuer avant la fonction de hachage. Ces transformations permettent par exemple de convertir des données XML en données binaires, ou de sélectionner seulement une partie des données (pour signer une partie d'un formulaire).
- **une empreinte numérique *DigestValue*** définie par son type *DigestMethod*. *SignedInfo* contient également la mise en forme canonique *CanonicalizationMethod* pour convertir la structure XML *SignedInfo* en données binaire.[13]

8.4 XadES (XML Advanced Electronic Signature)

XadES, spécification technique de l'ETSI en cours d'évolution, définit une extension de XML Signature afin de créer des signatures électroniques avancées telles que définies dans la « directive 1999/93/CE du Parlement européen et du Conseil du 13 décembre 1999 sur un cadre communautaire pour les signatures électroniques ». Voir section 5.4.[13]

La référence de cette spécification est « ETSI TS 101 903 Vx.y.z » où Vx.y.z est la version du document:

- **V1.1.1 :** http://uri.etsi.org/01903/v1.1.1/ (février 2002). Publié aussi comme note du W3C sur http://www.w3.org/TR/XAdES/
- **V1.2.1** : non disponible (mars 2004)
- **V1.2.2** : http://uri.etsi.org/01903/v1.2.2/ (avril 2004)
- **V1.3.2** : http://uri.etsi.org/01903/v1.3.2/ (mars 2006)

8.5 PGP, OpenPGP

Pretty Good Privacy (PGP) est un logiciel commercial, son fonctionnement repose sur le processus de cryptographie à clés asymétriques. Au début, Le PGP a été couramment utilisé pour courriers électroniques. Le concept PGP a été créé par Phil Zimmermann et sa version 1.0 est sortie en 1999[40].L'IETF a publié en 1998 le standard OpenPGP (sur base de PGP 5.x), laissant la porte ouverte à d'autres implémentations libres comme Gnu Privacy Guard (GPG).

Les messages PGP signés et/ou chiffrés sont convertis en base 64 avant d'être transférés. Afin de reconnaître le message PGP, un entête contenant BEGIN PGP MESSAGE est ajouté au début du message. PGP n'utilise pas ASN.1 ni les certificats X.509. Le système PGP possède son propre format de données16. Il possède également sa propre définition de certificat. De manière similaire à S/MIME, PGP permet la signature de message électronique. Les signatures sont détachées du document signé. Une application de messagerie ne supportant pas PGP pourra tout de même visualiser le message. vPGP possède une structure de certification particulière c'est-à-dire que n'importe qui peut facilement signer le certificat d'une autre personne et jouer ainsi le rôle d'autorité de certification[13, 40]

9 Service d'archivage et journalisation

L'archivage des documents à valeur probatoire au sein d'une entreprise est un problème important. Les coffres forts électroniques émergent sur le marché. Souvent, une entreprise ayant le besoin de stocker des documents importants (archivage de factures, de contrats, etc.) fait appel à des sociétés tiers d'archivage. Celles-ci mettent en place des systèmes de stockage redondants et dont l'accès est sécurisé. Elles prennent sur elles la responsabilité du stockage (dédommagement en cas de pertes de données). L'archivage demande donc un investissement non négligeable.

La journalisation consiste à garder en mémoire toutes les opérations effectuées, envoi, réception, archivage, destruction d'un document, etc. La journalisation est importante pour la sécurité[13]

10 Conclusion

La signature numérique est l'analogue de la signature sur papier. Elle se différencie de la signature écrite par le fait qu'elle n'est pas visuelle, mais correspond à une suite de caractères. Pour qualifier le mécanisme de signature, il faut garantir l'intégrité d'un document électronique et l'authentification de l'auteur, le mécanisme de signature se repose sur un dispositif de chiffrement qui dépend de la force du cryptage asymétrique retenu et de la taille des clés choisies. La signature numérique reste soumise au problème de la diffusion/authentification de la clé publique. Ce problème est en général résolu par la mise en place d'une infrastructure à clés publiques, entraînant toutes les vulnérabilités inhérentes à ce type de dispositif.

Dans ce chapitre nous avons présenté une vue globale sur les infrastructures de gestion de clé puis, les différents standards de clé publique et nous avons défini par la suite le mécanisme et les différents formats de la signature numérique.

CHAPITRE III : Analyse et conception

1- Introduction

Un projet, au sens commun du terme, est un ensemble d'activités et d'actions coordonnées, qui mobilisent des ressources dans un intervalle de temps précis, avec un début et une fin, afin de répondre à un besoin clairement identifié. L'analyse et la conception sont des parties primordiales dans le cycle de vie d'un projet. Pour la réalisation de notre système de signature, nous avons choisi d'axer notre travail sur la méthode de développement 2TUP qui est née des travaux poussés vers la standardisation et la flexibilité, et ce pour répondre aux contraintes actuelles de gestion et de développement.

2- Processus de développement 2TUP

2TUP signifie « 2 Track Unified Process» .C'est un processus qui répond aux caractéristiques du Processus Unifié. Le processus 2TUP apporte une réponse aux contraintes de changement continuel imposées aux systèmes d'information de l'entreprise. En ce sens, il renforce le contrôle sur les capacités d'évolution et de correction de tels systèmes. « 2 Track» signifie littéralement que le processus suit deux chemins. Il s'agit des « chemins fonctionnels » et « d'architecture technique », qui correspondent aux deux axes de changement imposés au système d'information[2] .

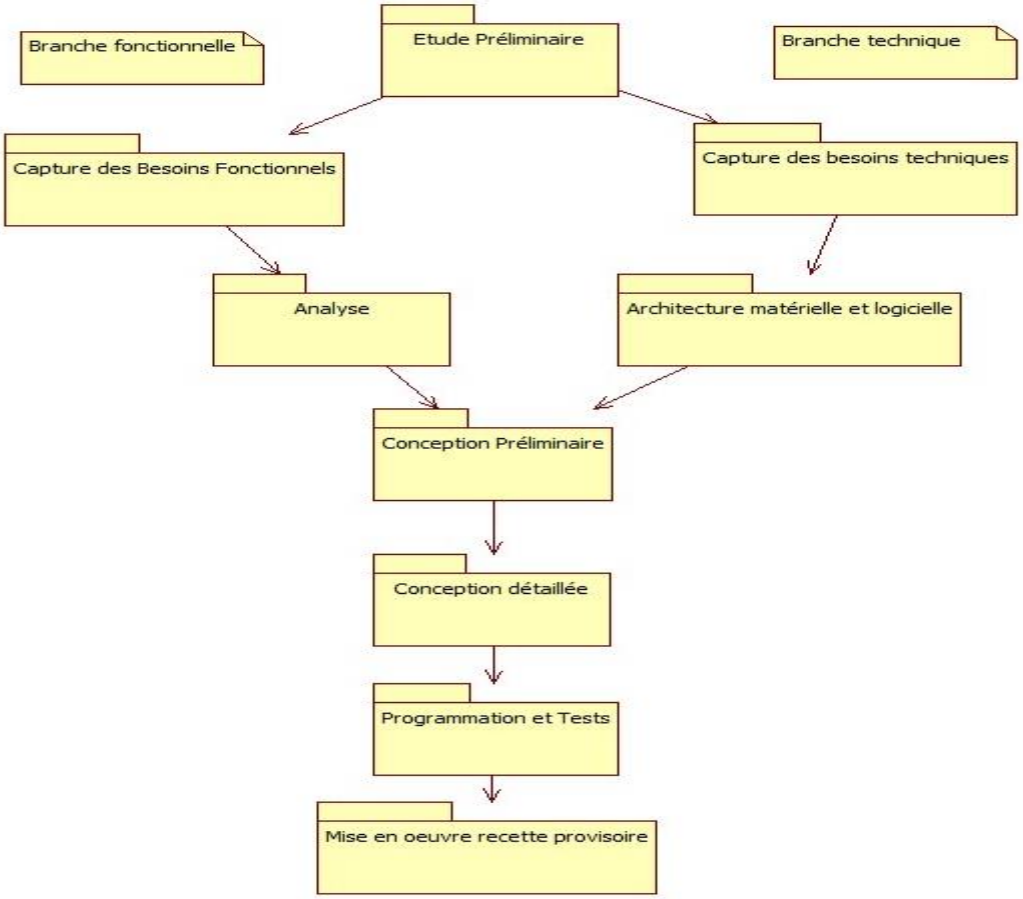

Figure 19:Le processus de développement en Y

3- L'étude préliminaire

3-1 Identification de besoin

Le Centre de Développement des Technologies Avancées CDTA dispose un nombre important d'employés (plus de 600 employés), cela nécessite et consomme beaucoup de documents administratifs en papier. Le traitement des documents physiques exige des efforts importants au quotidien. En calculant le temps qu'il faut pour, imprimer chacun de ces documents, le faire signer, le préparer à l'envoi et l'attendre en courrier afin de le remette à leur destinataire, les Managers au CDTA veulent passer à la dématérialisation, qui offre un gain de temps potentiel avec la signature numérique. Cette solution technologique devienne d'autant incontournable dans le monde des documents dématérialisés, elle permet de passer les procédures administratives, d'achat ou de gestion, à la vitesse supérieure.

La réalisation de la plateforme de signature numérique nécessite la mise en place d'une solution de confiance pour que les identités électroniques et les identifiants de signature soient vérifiés par une tiers de confiance « une autorité de certification » ou bien toute une infrastructure de confiance.

3-2 Description du travail à réaliser

Le présent projet de Master vise la proposition d'une solution qui permet aux responsables de signer des documents administratifs internes, en garantissant l'identité de signataire, ainsi que l'intégrité et la confidentialité. Cela se fait par des mécanismes et des algorithmes de sécurité, en particulier la cryptographie asymétrique et les fonctions de hachage.

Pour garantir l'identité de signataire, la solution doit être basée sur une infrastructure de gestion de clés, que le Centre de Développement des technologies avancées le dispose dans son Data Center. Des configurations devront être misent en place pour délivrer des certificats de type utilisateur afin de signer les documents. Et pour valider une signature, un processus de vérification de la signature doit être mis en place.

3-3 Recueil des besoins fonctionnels

La capture des besoins fonctionnels est focalisée sur le métier des utilisateurs. Elle qualifie plus tôt le risque d'un système inadapté aux utilisateurs. Notre analyse consiste à étudier précisément la spécification fonctionnelle de manière à obtenir une idée de ce que va réaliser le système de signature en terme de métier.

Afin d'identifier les besoins fonctionnels, des scenarios de signatures ont été discutés avec des responsables administratifs et techniques du Centre de développement des Technologies Avancées. Mais, cette discussion n'était pas vraiment suffisante à cause de la situation actuelle relative au confinement causé par le Covid-19. Pour cela on a beaucoup plus se basé sur la recherche des travaux connexes, dont on a conclu que la signature des documents doit être standard et générale. En complétant le support de gestion au fur à mesure. Le processus développé de la signature est ci-dessous :

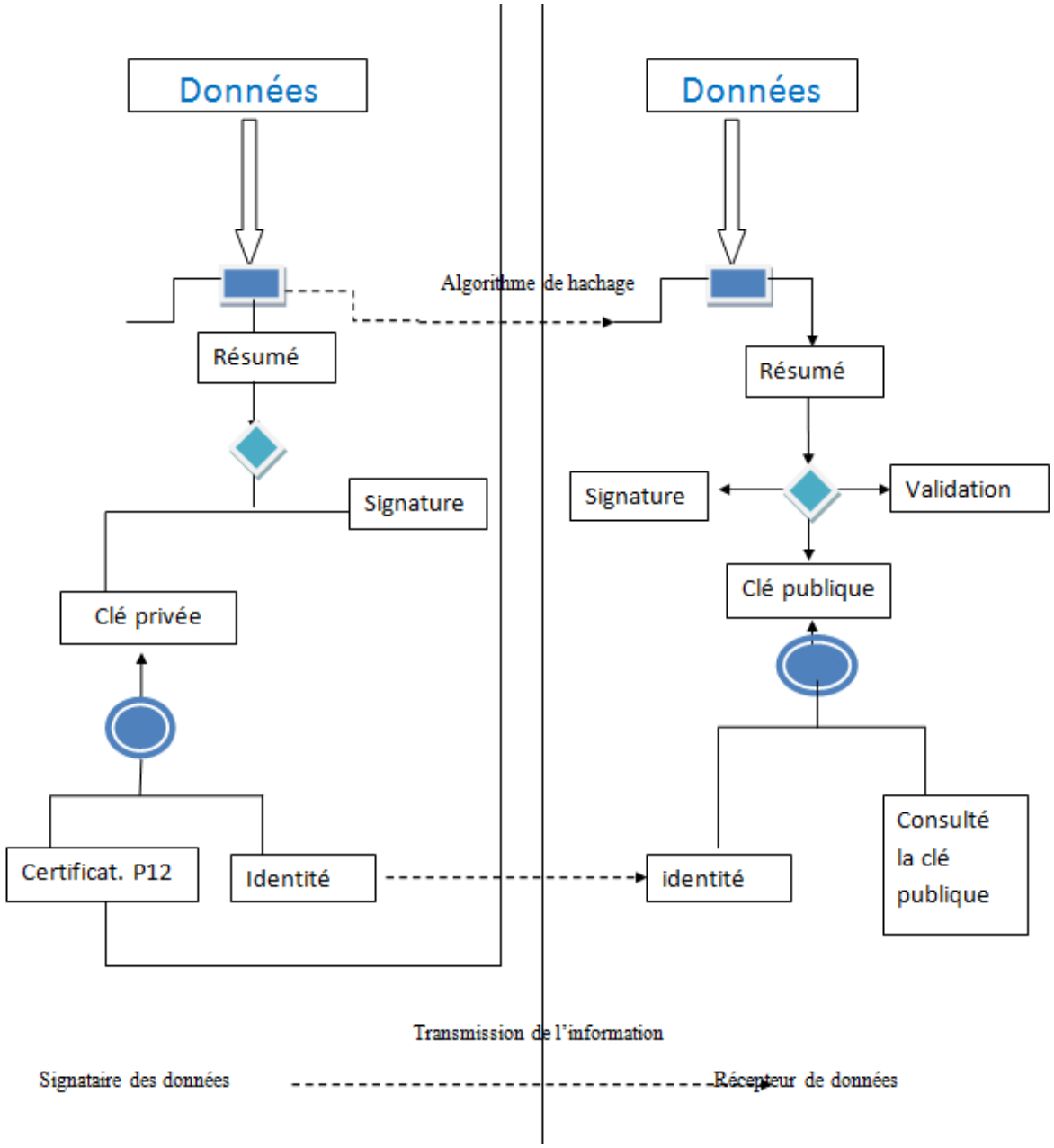

Figure 20 Processus de la signature à réaliser

Le processus métier de la signature numérique des documents se résume en deux parties essentielles :

a- Le signataire

À partir des informations à signer :

- On calcule la valeur du hachage cryptographique des données ;
- On chiffre cette valeur avec sa propre clé privée, ce que implique la signature ;

- On transmet les données, la signature, l'identité du signataire et les deux algorithmes utilisés ;

b- Le récepteur (le demandeur de la signature)

- Reçoit les données D et la signature chiffrée SC ;
- Applique le hachage cryptographique à D et trouve H1 ;
- Récupère la clé publique du signataire à partir de son identité ;
- L'utilise pour déchiffrer la signature reçue SC et trouve H2 ;
- Compare H1 et H2 ;
- La signature est valide s'ils sont identiques ;

3-4 Recueil des besoins techniques

La capture des besoins techniques, qui recense toutes les contraintes et les choix dimensionnant la conception du système, les outils et le matériel sélectionnés ainsi que la prise en compte des contraintes d'intégration avec l'existant (pris requis d'architecture technique).

Les choix techniques adoptés pour le projet sont comme suit :

- La modélisation du système avec UML et l'outil visuel paradigme.
- Le langage JAVA et l'IDE NetBeans pour le développement de la solution .
- Les API : JCA (Java Cryptography Architecture) et JCE (Java Cryptography Extension).

- JCA (Java Cryptography Achitecture) qui définit l'architecture générale du framework et les fonctionnalités cryptographiques de base (fonctions de hachage, signatures numériques, clés, certificats, ...)
- JCE (Java Cryptography Extension) qui fournit des fonctionnalités cryptographiques de haut niveau (chiffrement/déchiffrement avec algorithmes symétriques/asymétriques, authentification de messages (HMAC), ...)
- Le Provider Boucy castel.

- La bibliothèque itext5 pour la manipulation des fichiers PDF.
- L'algorithme sha-256 pour le hachage.
- L'algorithme RSA pour le cyptage asymétrique.
- PKCS pour les standards de cryptographie à clé publique.

3-5 Architecture matérielle et logicielle

La figure ci-dessous présente l'architecture de Data Center du CDTA. La solution de signature impose l'utilisation des deux serveurs respectivement le serveur de CA (autorité de Certification) et le serveur de fichier.

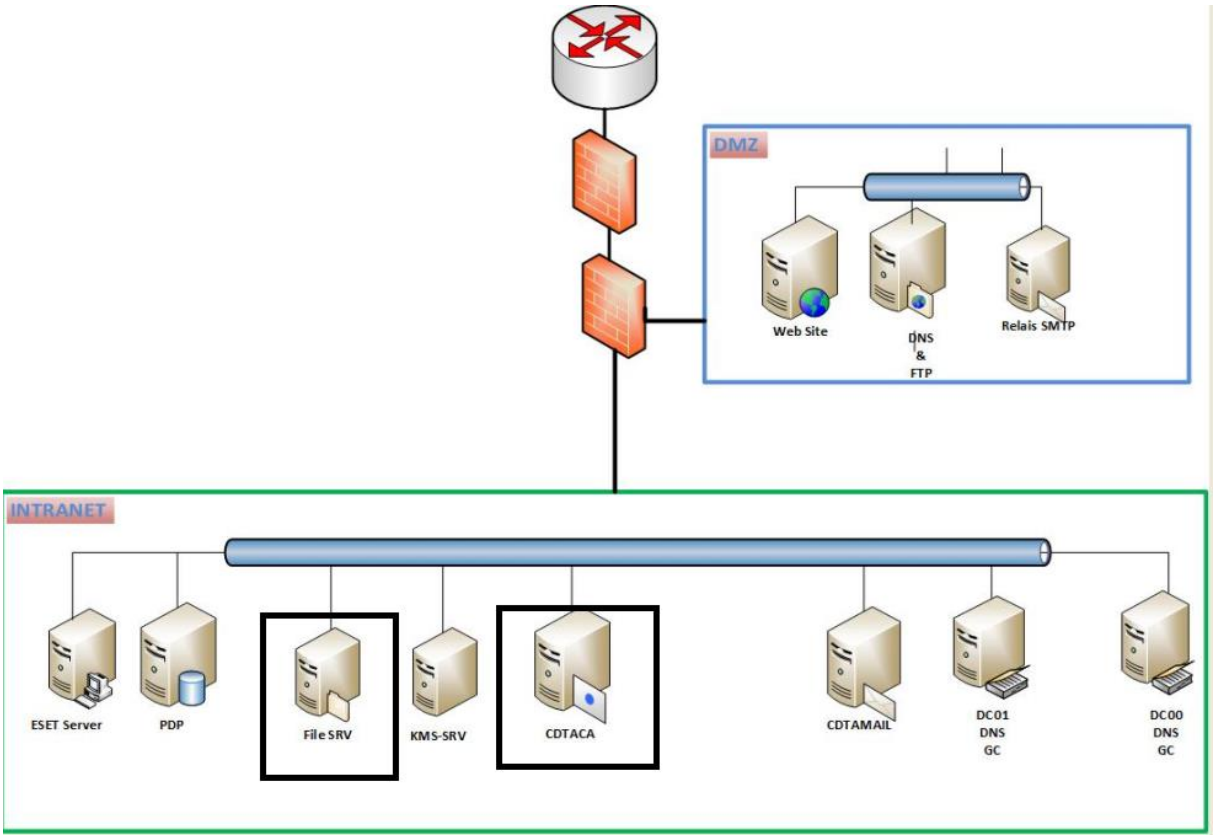

Figure 21 architecture générale de Data Center de CDTA.

3-6 Analyse fonctionnelle et définition des objectifs

La capture des besoins fonctionnels, produit un modèle de besoins focalisé sur le métier des utilisateurs. Cette étape élimine le risque d'avoir un système inadapté aux besoins des utilisateurs.

3-6-1 Identification des cas d'utilisation

3-6-1-1 L'outil Visual Paradigme

Pour une bonne présentation des diagrammes UML on a choisi Visual paradigme version 7.1 qui est un logiciel de création de diagrammes, conçu pour les développeurs logiciels afin de modéliser des systèmes et de gérer les processus de développement. Le logiciel présente de nombreux repères pour accéder facilement à ses fonctionnalités.

3-6-1-2 Diagramme de cas d'utilisation

Le modèle de cas d'utilisation capture les exigences d'un système et les fonctionnalités futures que doit l'implémenter. Les Cas d'Utilisations sont un moyen de communication avec les utilisateurs et d'autres parties prenantes ce que le système est destiné à faire. Ils montrent l'interaction entre le système et les entités externes au système. Il scinde la fonctionnalité du système en unités cohérentes.

3-6-1-3 Identification des acteurs

Le système de signature nécessite trois types d'acteurs principaux dont chacun a son rôle. Le tableau ci-dessus résume le rôle de chaque acteur :

Tableau 3: les acteurs et leurs rôles

Acteur	Rôle
Administrateur	Il gère les utilisateurs et leurs certificats
Signataire	En générale c'est un responsable qui signe des documents administratifs. Il utilise sa clé privée qui est intégrée dans un certificat p12. Il peut visualiser son certificat (.cer).
Récepteur ou demandeur	C'est un demandeur de signature. Il vérifie la signature lors de la réception de document signé.

3-6-1-4 Cas d'utilisation de l'administration des utilisateurs et des certificats

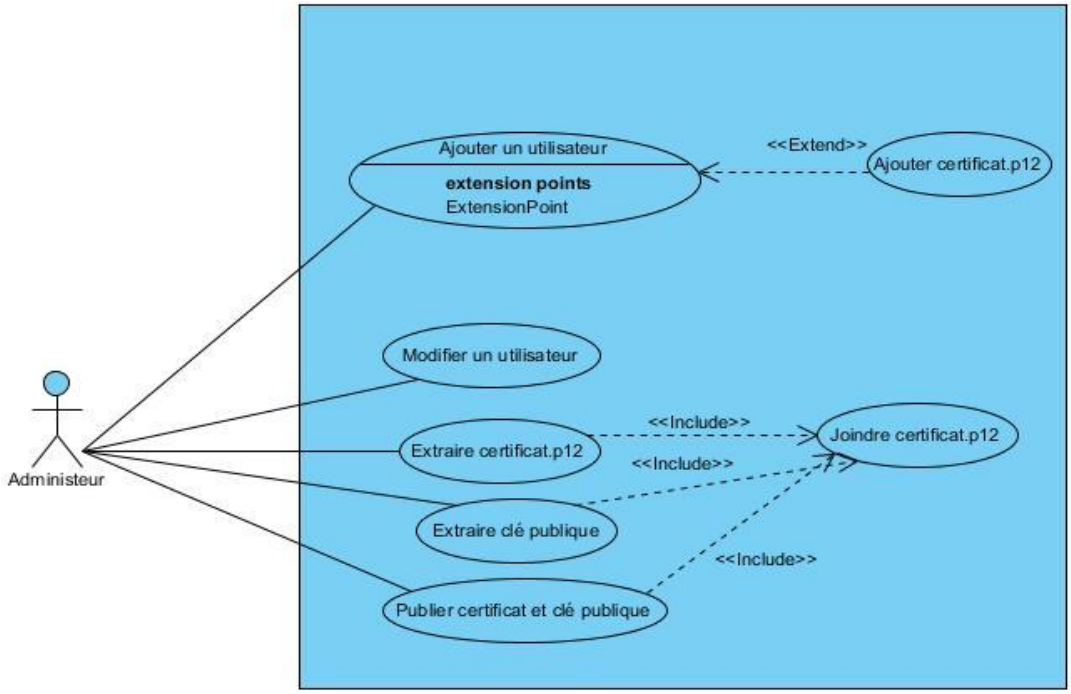

Figure 22. Cas d'utilisation de l'administrateur.

- **Description des cas d'utilisations de l'administration**

Tableau 4. Description des cas d'utilisation de l'administration.

Acteur	**Cas d'utilisation**	**Message émis/reçus par les acteurs**

Administrateur	Ajouter utilisateur	**Emet :** saisir les informations et ajouter un compte utilisateur. Joindre Certificat**.** **Reçoit :**le Certificat p12 depuis l'autorité de certification.
	Modifier utilisateur	**Emet :** mettre à jours les informations d'un utilisateur. **Reçoit :** une demande. Consulter Certificat.
	Extraire certificat.cer	**Emet :** Certificat.cer **Reçoit :** Certificat.p12
	Extraire clé publique	**Emet :** fichier contient clé publique. **Reçoit :** Certificat.p12
	Publier Certificat et clé publique	**Emet :** Certificat et clé publique aux utilisateurs de l'application.

3-6-1-5 Cas d'utilisation de la signature

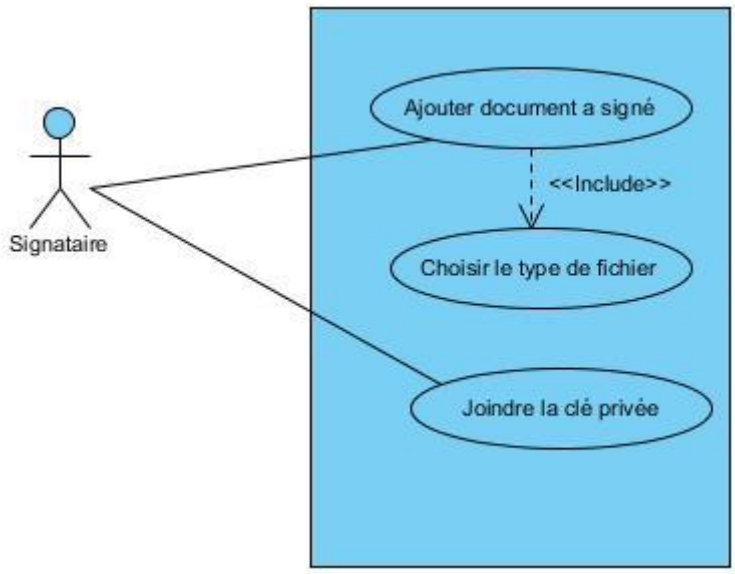

Figure 23.Cas d'utilisation de la signature

- **description de cas d'utilisation de la signature**

Tableau 5.Description de cas d'utilisation de la signature.

Acteur	Cas d'utilisation	Message émis/reçus par les acteurs
signataire	Ajouter document à signer	**Emet :** joindre document au formulaire selon le type. **Reçoit :** un document pour signer.
	Joindre la clé privée	**Emet :** joindre clé privée. **Reçoit :** Certificat .p12

3-6-1-6 Cas d'utilisation de demande de signature

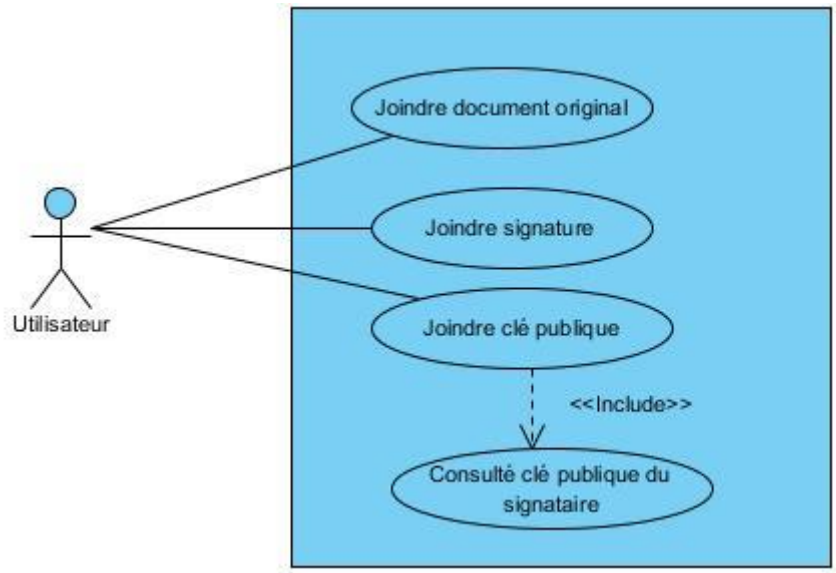

Figure 24. Cas d'utilisation de la vérification de signature

- **Description des cas d'utilisations de la vérification de la signature d'un document**

Tableau 6.Description des cas d'utilisations de la vérification de la signature d'un document

Acteur	Cas d'utilisation	Message émis/reçus par les acteurs
Utilisateur (demandeur de signature)	Joindre document original	**Emet :** document à signer (original).
	Joindre signature	**Emet :** joindre signature.

		Reçoit : document signé + signature.
	Joindre clé publique	**Emet :** envoie la clé publique au système de vérification. **Reçoit :** la clé publique du l'identité du signataire

3-7 La conception.

3-7-1 Diagramme d'activité

3-7-2-1 Diagramme d'activité de la signature

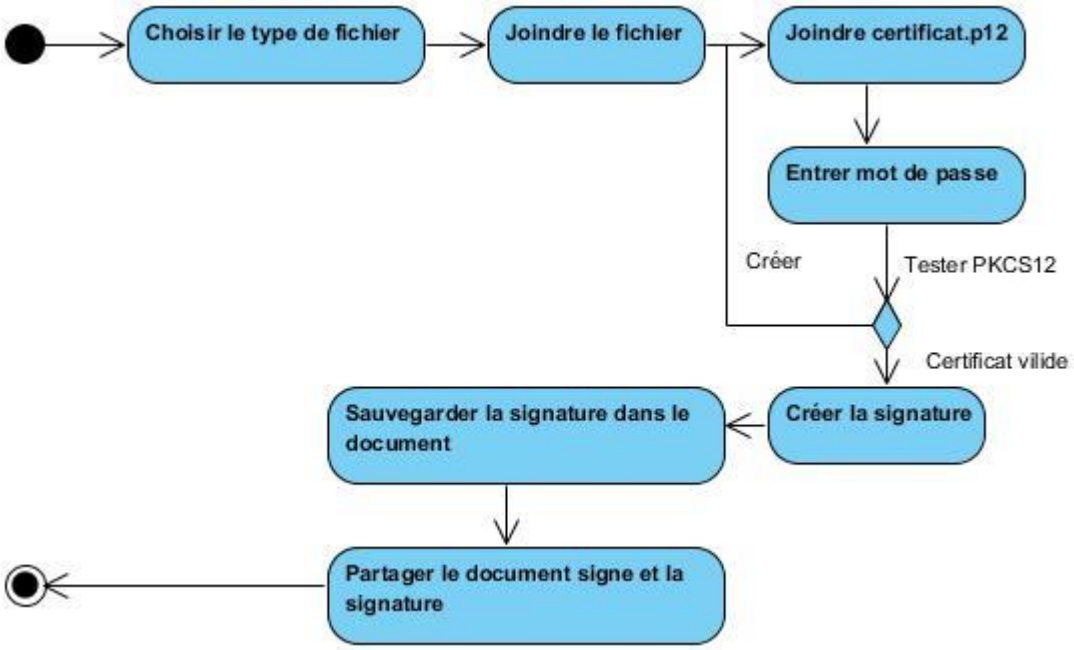

Figure 25.Diagramme d'activité de la signature

3-7-2-3 Diagramme d'activité de la vérification de la signature

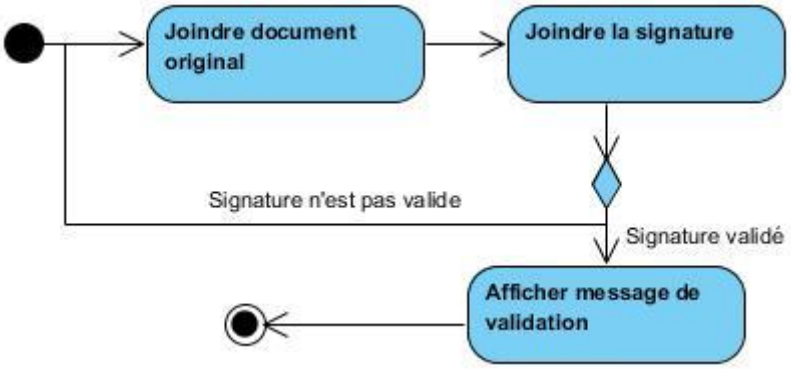

Figure 26.Diagramme d'activité de la vérification de signature

.

3-7-2-4 Diagramme d'extraction de la clé publique et du certificat

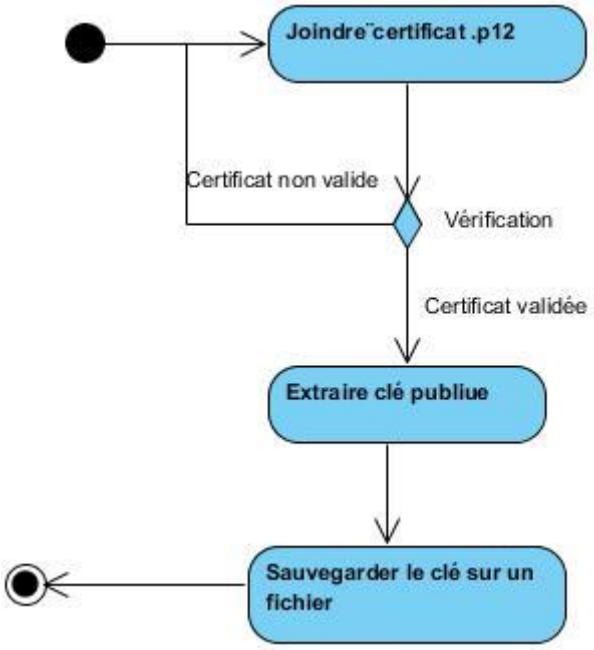

Figure 27.Diagramme d'extraction de la clé publique

3-7-3 diagramme de séquence

Un diagramme de séquence montre chronologiquement (de haut en bas) les interactions entre un ensemble d'objets. Dans cette partie nous décrivons deux diagrammes essentiels (métier) de notre système qui sont, le diagramme de séquence de la signature et le diagramme de séquence de la vérification de la signature.

3-7-1-1 Diagramme de séquence de la signature d'un document

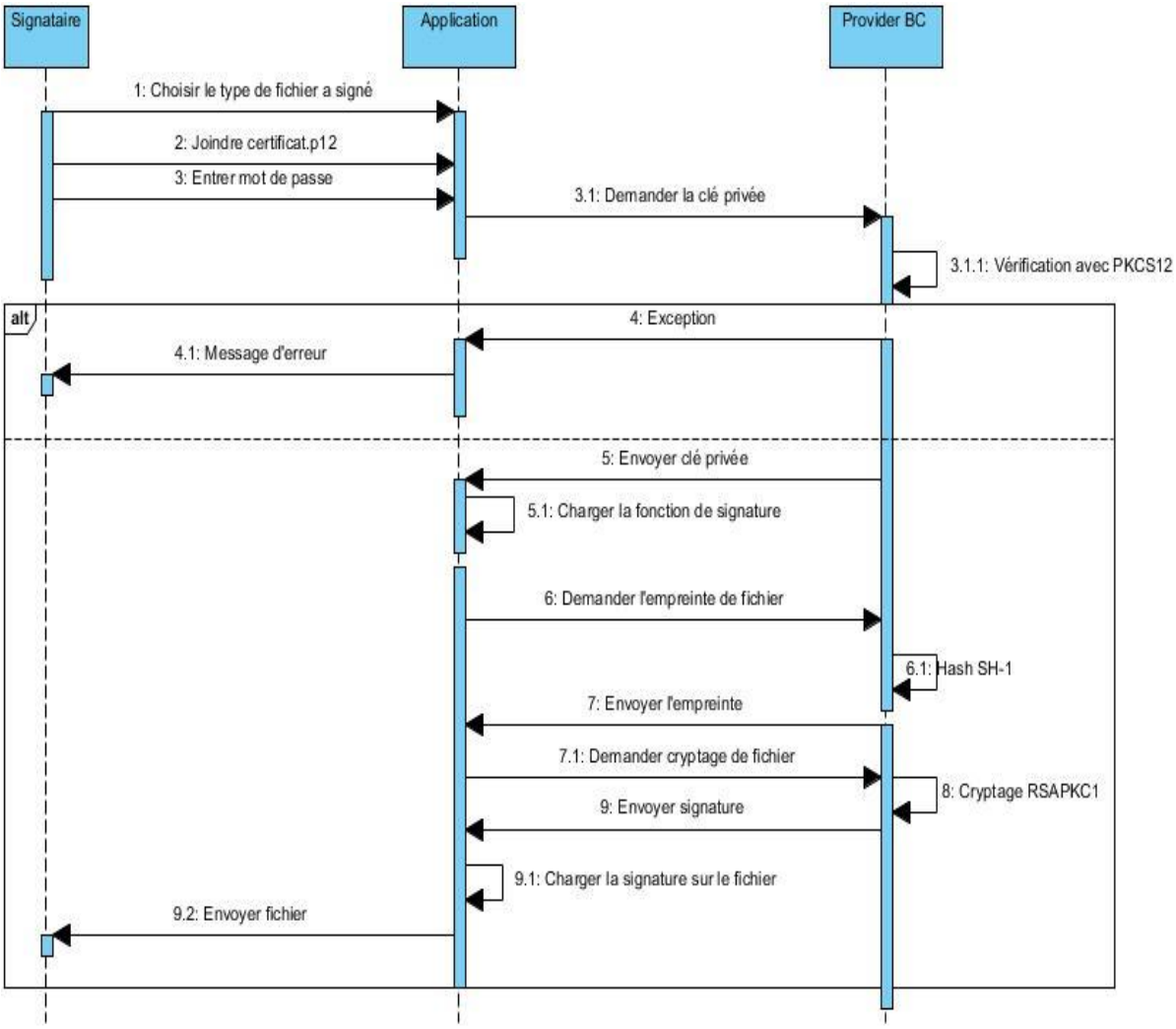

Figure 28. Diagramme de séquence de la signature d'un document

3-7-1-2 Diagramme de séquence de la vérification de la signature

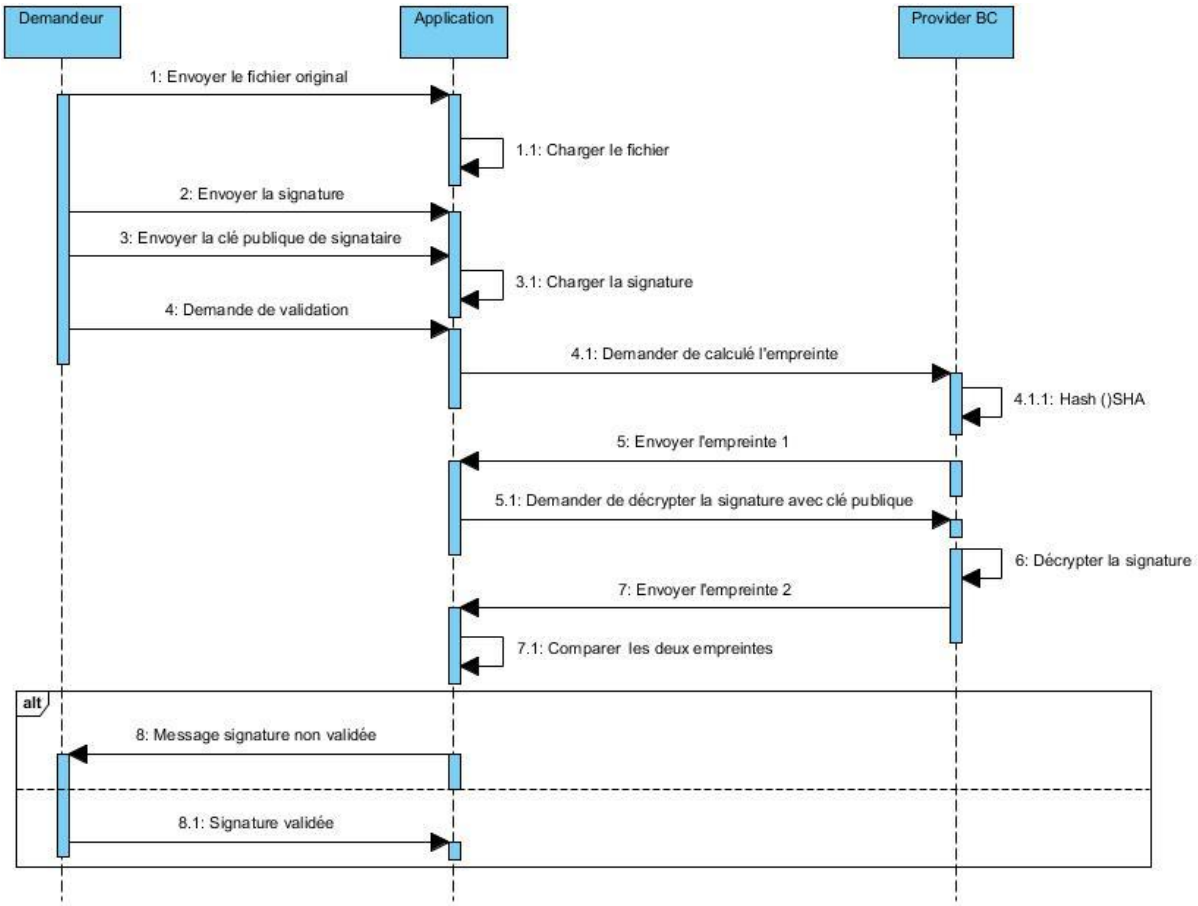

Figure 29.Diagramme de séquence de la vérification de la signature

4- Conclusion

Dans ce chapitre, nous avons modélisé un système de signature numérique afin de signer des différents formats de document (pdf, doc, txt), pour se faire nous avons utilisé le langage UML pour la conception de la solution proposée. Dans le chapitre qui suit nous allons aborder la partie implémentation.

CHAPITRE IV : la cryptographie dans le langage JAVA et le package Bouncy Castel

1- Introduction :

Java fournit au développeur un Framework de sécurité pour écrire des applications, et aussi à l'utilisateur ou l'administrateur un ensemble d'outils pour la gestion sécurisée des applications. Les APIs de sécurité Java couvrent plusieurs secteurs.Ces interfaces vérifiant l'authentification et le contrôle d'accès, permettent aux applications de se protéger contre les accès non autorisés aux ressources.
Dans ce chapitre nous montrons les principes de conception de JCA/JCE, la notion de CSP (Cryptographic Service Provider). Nous présentons les concepts introduits dans l'API de sécurité comme les classes de moteur (engine classes) et l'interface de provider de service SPI (Service Provider Interface) pour l'implémentation des providers de service. Dans la classe Provider, nous montrons comment installer et configurer un provider et comment utiliser ses implémentations. Nous Présentons par la suite le provider Bouncy Castle.

2- La sécurité du langage Java

Le langage Java est conçu pour être de type sécurisé et facile à utiliser. Il contient une architecture de sécurité destinée à protéger les ressources locales de code chargé à distance.[43]. Java permet la construction des systèmes sans virus et inviolables. Les techniques d'authentification sont basées sur le cryptage à clé publique.[26]

Le compilateur Java produit un programme Java au format byte-code qui s'exécute sur une machine virtuelle Java indépendamment de la machine et de la plateforme OS. Un vérificateur du byte-code est utilisé automatiquement pour s'assurer que le programme exécuté dans l'environnement d'exécution Java (JRE) est en byte-code. Une fois le byte-code est vérifié, l'environnement d'exécution Java le prépare pour l'exécution. Le byte-code est un code binaire, ce qui permet un traitement plus rapide que le code source Java, et qui rassemble (compile) tous les codes dispersés dans les différents fichiers lors de l'écriture de programme. C'est par rapport à ces caractéristiques que le Java est à la fois robuste et sécurisé[26].

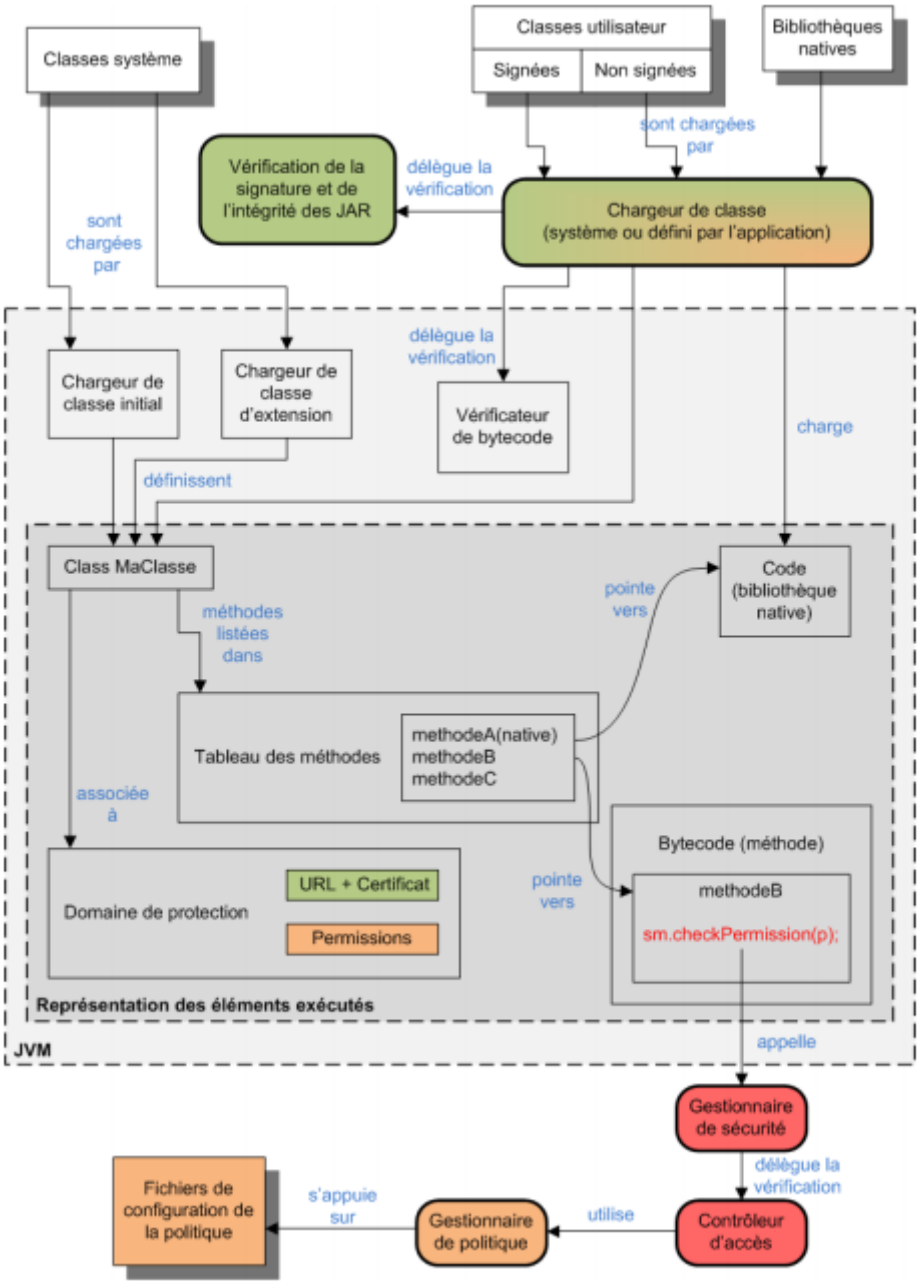

Figure 30 : Architecture générique du système de contrôle d'accès de Java[33]

3- Cryptographie dans Java

La spécification de l'API de sécurité Java utilise des termes techniques pour désigner des concepts cryptographiques. On donne ci-dessous la signification exacte de ces termes utilisés**:**

- Chiffrement et déchiffrement (Encryption and Decryption): Le chiffrement est le processus qui prend des données en entrée (plaintetxt) et une clé (key) pour produire des données chiffrées (ciphertext). Le déchiffrement est le processus

inverse qui prend le texte chiffré et une clé pour reproduire les données originales en texte clair.

- Chiffrement basé sur mot de passe (PBE-Password-Based Encryption) : Le chiffrement basé sur mot de passe (PBE) permet de créer une clé de chiffrement à partir d'un mot de passe afin d'éviter les attaques de mot de passe (ou attaque de dictionnaire).
- Agrément de clé (Key Agreement) ou établissement de clé : C'est un protocole qui permet d'établir des clés cryptographiques entre deux ou plusieurs parties.
- Code d'authentification de message (MAC-Message Authentification Code) : Un code d'authentification de message (MAC) fournit un moyen pour vérifier l'intégrité de l'information transmise ou stockée dans un endroit incertain, basé sur une clé secrète. Le mécanisme MAC qui est basé sur une fonction de hachage et un secret, est désigné sous le nom *HMAC*. Un *HMAC* peut être utilisé avec toutes fonctions de hachage, e.g. MD5 ou SHA1, en combinaison avec une clé secrète partagée [14].

4- Fournisseurs de services cryptographiques

Les fournisseurs contiennent un package (ou un ensemble de packages) qui fournit des implémentations concrètes pour les algorithmes cryptographiques annoncés. java.security.Provider est la classe de base pour tous les fournisseurs de sécurité. Chaque CSP contient une instance de cette classe qui se caractérise par le nom du fournisseur et le répertorie de tous les services / algorithmes de sécurité qu'il implémente. Lorsqu'une instance d'un algorithme particulier est nécessaire, la structure JCA consulte la base de données du fournisseur et si une correspondance appropriée est trouvée alors l'instance sera créée.

L'indépendance de l'algorithme est obtenue en définissant une interface de programmation d'application (API) générique de haut niveau que, toutes les applications utilisent pour y accéder à un type de service. L'indépendance d'implémentation est obtenue en faisant en sorte que toutes les implémentations de fournisseurs soient conformes à des interfaces bien définies. Les instances de classes de moteur sont ainsi "soutenues" par des classes d'implémentation qui ont les mêmes signatures de méthode. Les appels d'applications sont acheminés via la classe du moteur et sont livrés à l'implémentation de support sous-jacente. L'implémentation gère la demande et renvoie les résultats appropriés.

Les méthodes d'API d'application de chaque classe de moteur sont acheminées vers les implémentations du fournisseur via des classes qui implémentent l'interface de fournisseur de services (SPI) correspondante. Autrement dit, pour chaque classe de moteur, il existe une classe SPI abstraite correspondante qui définit les méthodes que l'algorithme de chaque fournisseur de services cryptographiques doit implémenter. Le nom de chaque classe SPI est le même que celui de la classe de moteur correspondant, suivi de SPI. **Par exemple**, la Signatureclasse de moteur donne accès à la fonctionnalité d'un algorithme de signature

numérique. L'implémentation réelle du fournisseur est fournie dans une sous-classe de SignatureSpi. Les applications appellent les méthodes API de la classe de moteur, qui à leur tour appellent les méthodes SPI dans l'implémentation réelle.

Pour chaque classe de moteur dans l'API, des instances d'implémentation sont demandées et instanciées en appelant la méthode d'usine getInstance() dans la classe de moteur. Une méthode d'usine est une méthode statique qui renvoie une instance d'une classe. Les classes de moteur utilisent le mécanisme de sélection du fournisseur de framework pour obtenir l'implémentation de support réel (SPI), puis créent l'objet moteur réel. Chaque instance de la classe de moteur encapsule (en tant que champ privé) l'instance de la classe SPI correspondante, connue sous le nom d'objet SPI. Toutes les méthodes API d'un objet API sont déclarées finales et leurs implémentations invoquent les méthodes SPI correspondantes de l'objet SPI encapsulé[25].

Exemple de code pour obtenir une instance d'une classe de moteur

```
import javax.crypto. *;

   Cipher c = Cipher.getInstance ("AES");

   c.init (ENCRYPT_MODE, clé);
```

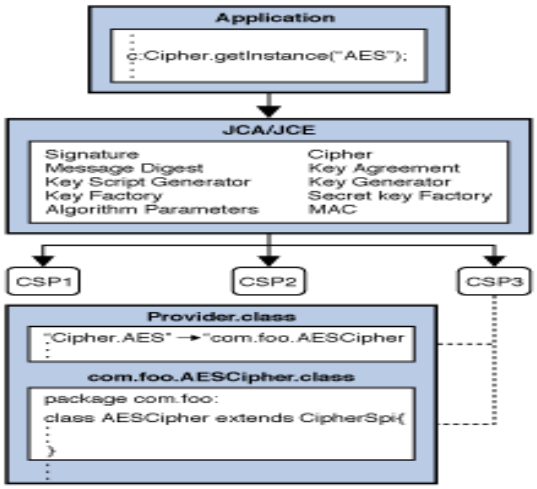

Figure 31: L'application récupère l'instance de chiffrement «AES»

2-1- Installation et configuration de providers :

Il y a deux étapes pour installer un provider : installer les classes de package du provider et configurer le provider. Il y a deux façons pour enregistrer un provider :

- **Enregistrement statique** : on ajoute le nom du provider à la liste des providers existants dans le fichier java.security du sous répertoire lib/security du JRE. Pour chaque provider disponible il y a une ligne correspondante de la forme :

security.provider.n= masterClassName

Cette ligne permet de déclarer un provider et spécifier son ordre de préférence n. L'ordre de préférence est celui dans lequel les providers sont recherchés pour les algorithmes demandés. L'ordre 1 est le plus préféré, suivi de 2 et ainsi de suite. Master Class Name doit spécifier la classe maitre du provider. Cette classe est toujours une sous classe de la classe Provider. Le constructeur de sous classe positionne les valeurs des propriétés qui sont nécessaires pour l'API de cryptographie Java pour chercher les algorithmes implémentés par le provider.

Enregistrement dynamique : peut être réalisé par l'application client qui appelle des méthodes de la classe Security telle que : Security.addProvider, ou insertProviderAt.
Ce type d'enregistrement n'est pas persistant et peut être seulement réalisé par des programmes sûrs et de confiance. Chaque instance de classe Provider a un nom, un numéro de version, et une chaine décrivant le provider et ses services. La classe Provider a des méthodes pour avoir ces informations, on peut interroger l'instance Provider en appelant les méthodes suivantes : getName(),getVersion(), et getInfo().[14]

4-1- Architecture JCA/JCE

L'API de sécurité fait partie du noyau du Java, elle est conçue pour permettre aux développeurs d'incorporer des fonctionnalités de sécurité dans leurs programmes. Deux API fournies depuis Java 1.4 permettent la mise en œuvre de la cryptographie :

- **JCA (Java Cryptography Achitecture**) qui définit l'architecture générale du framework et les fonctionnalités cryptographiques de base (fonctions de hachage, signatures numériques, clés, certificats, ...)
- **JCE (Java Cryptography Extension)** qui fournit des fonctionnalités cryptographiques de haut niveau (chiffrement/déchiffrement avec algorithmes symétriques/asymétriques, authentification de messages (HMAC), ...)

Historiquement, les API de cryptographie de Java étaient séparées en deux parties afin de permettre une restriction de diffusion :

- **le package *java.security*** contient des classes qui ne possèdent pas de restriction de diffusion.
- **le package *javax.crypto*** contient des classes qui ont été pendant un moment diffusées sous restriction.

l'API JCA est contenue dans le package java.security et l'API JCE est contenue dans le package javax.crypto.

Tableau 7 : contenue des API JCA/JCE

JCA	JCE
java.security	javax.crypto
java.security.acl	javax.crypto.interfaces
java.security.cert	javax.crypto.spec
java.security.interfaces	
java.security.spec	

4-1-1- JCA (Java Cryptographie Architecture)

L'API de sécurité dans JDK 1.1 a introduit l'architecture de la cryptographie Java-JCA (Java Cryptography Architecture), qui est un Framework pour l'accès et le développement de fonctionnalités cryptographiques pour la plateforme Java. JCA comprenait des APIs pour les signatures numériques et les empreintes de messages. C'est une architecture basée provider permettant d'implémenter un ou plusieurs providers de services cryptographiques interopérables, connus comme providers de services cryptographiques- CSPs (Cryptographic Service Providers) [28].

4-1-2- JCE (Java Cryptographie Extension)

JCE est une API qui propose de standardiser l'utilisation de la cryptographie en restant indépendant des algorithmes utilisés. Elle prend en compte le chiffrement/déchiffrement de données, la génération de clés et l'utilisation de la technique MAC (Message Authentification Code) pour garantir l'intégrité d'un message.

JCE a été intégrée au JDK 1.4. Pour pouvoir utiliser cette API, il faut obligatoirement utiliser une implémentation développée par un fournisseur (provider). Avec le JDK 1.4, Sun fournit une implémentation de référence nommée SunJCE. Les classes et interfaces de l'API JCE sont regroupées dans le package.

JCA/JCE ont des mécanismes simples pour permettre aux développeurs d'ajouter et de choisir des providers particuliers.

La Figure ci-dessous montre comment les différents composants de l'application font appel aux classes appropriées des APIs JCA/JCE, ces dernières appellent les classes dans un provider qui fournit des implémentations pour les classes SPI. Les classes utilisent le code interne dans le provider pour fournir la fonctionnalité demandée [14].

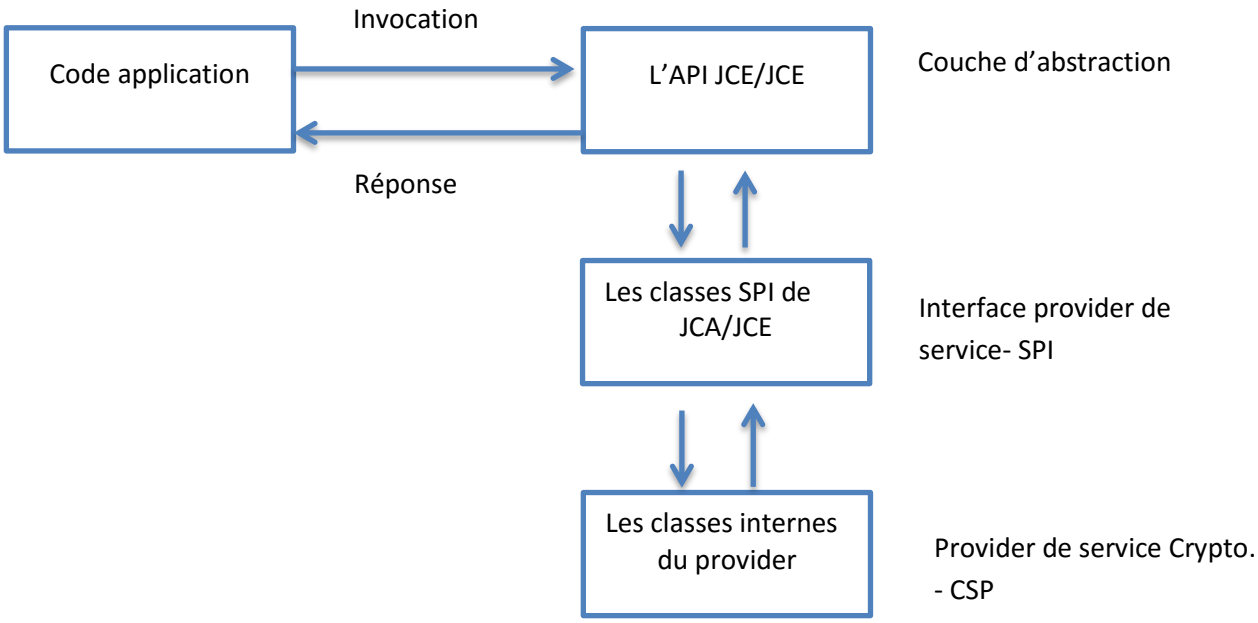

Figure 32 : L'API JCA/JCE et les providers de service

5- Magasins de clés

Une base de données appelée "keystore" peut être utilisée pour gérer un référentiel de clés et de certificats. Les magasins de clés sont disponibles pour les applications qui ont besoin de données à des fins d'authentification, de chiffrement ou de signature.

Les applications peuvent accéder à un magasin de clés via une implémentation de la KeyStoreclasse, qui se trouve dans le java.securitypackage. Depuis JDK 9, le type de fichier de clés par défaut et recommandé (format) est le "pkcs12", qui est basé sur la norme de syntaxe d'échange d'informations personnelles RSA PKCS12. Auparavant, le type de fichier de clés par défaut était "jks", qui est un format propriétaire. D'autres formats de magasin de clés sont disponibles, tels que, "jceks", et "pkcs11", ce dernier est basé sur la norme RSA PKCS11 et prend en charge l'accès aux jetons cryptographiques tels que, les modules de sécurité matériels et les cartes à puce [25].

6- Le package de Bouncy Castle

6-1- Historique

Bouncy Castle a commencé lorsque deux collègues étaient fatigués de devoir réinventer un ensemble de bibliothèques de cryptographie à chaque fois qu'ils changeaient de travail dans Java SE côté serveur . L'un des développeurs était actif dans le développement de Java ME (J2ME à l'époque) comme passe-temps et une considération de conception était d'inclure la plus grande gamme de machines virtuelles Java pour la bibliothèque, y compris celles sur

J2ME. Cette considération de conception a conduit à l'architecture qui existe dans Bouncy Castle[34].

Le projet, fondé en mai 2000, a été initialement écrit en Java uniquement, mais a par la suite ajoutée une API C # en 2004. Le 18 octobre 2013, une association à but non lucratif, la Legion of the Bouncy Castle Inc. a été créée dans l'État de Victoria, en Australie, par les principaux développeurs et d'autres pour s'approprier le projet et soutenir le développement continu du Apis. L'association a été reconnue comme un organisme de bienfaisance australien avec un but la promotion dans l'éducation est un but qui est bénéfique pour la communauté, cette association a été criée par l' Australian Charities et Not-For-Profits le 7 novembre 2013[34]. L'association a été autorisée à collecter des fonds pour soutenir ses objectifs le 29 novembre 2013 par Consumer Affairs Victoria. Les API de crypto Bouncy Castle sont gérées par une organisation caritative australienne, la Légion du Bouncy Castle Inc. , qui s'occupe de l'entretien et de l'alimentation des API de Bouncy Castle[27].

6-2- L'architecture de Bouncy Castel

Les algorithmes du package BC sont conformes au Framework JCE. Lorsque l'accès aux librairies JCE n'est pas possible dans certains cas (comme le cas des applets Java ou les MIDlets Java exécutés sur les petits dispositifs avec ressources limitées). Le package BC contient une API légère (lightweight) appropriée à l'environnement mobile comme la plateforme J2ME/MIDP. L'API BC existe aussi en version C# et fournit les même fonctionnalités que celles du Java.

6-3- La version certifier de Bouncy Castel

Le provider Boucy Castel fournit une version FIPS (certifier) et une autre non FIPS. Dans une JVM, nous ne pourrons pas inclure les deux versions, pour cela nous devrons exclure la version actuelle avant d'inclure l'autre.

6-3-1- Le standard FIPS 140

La norme FIPS (Federal Information Processing Standard) 140-2 est une norme gouvernementale américaine et canadienne qui spécifie les exigences de sécurité pour les modules cryptographiques et opérationnelles des modules des systèmes qui protègent les informations sensibles. Ces modules utilisent des fonctions de sécurité approuvées par le NIST (National Institute of Standards and Technology) telles que, les algorithmes cryptographiques, la taille des clés, la gestion des clés et les techniques d'authentification.

6-3-2- Intégration avec le fournisseur FIPS de Bouncy Castle

L'intégration de fournisseur FIPS de Bouncy Castle prend en charge trois phases:

- **Hybride** pour la transition des clés privées du magasin de clés par défaut vers le magasin de clés Bouncy Castle.
- **Non-hybride** pour commencer à stocker les clés privées uniquement dans le magasin de clés Bouncy Castle.
- **Approuvé uniquement** pour imposer l'utilisation de matériaux cryptographiques approuvés par FIPS uniquement tels que, la taille de clé, l'algorithme et le fichier de clé.

Plusieurs propriétés du *fichier <pf_install>* /pingfederate/bin/run.properties nous permettent de configurer ces phases comme indiqué dans le tableau suivant[22].

Tableau 8 : configuration du fichier<pf_install>

Phase	Propriétés
Hybride	pf.hsm.mode=BCFIPS pf.hsm.hybrid=true
Non hybride	pf.hsm.mode=BCFIPS pf.hsm.hybrid=false
Approuvé uniquement	pf.hsm.mode=BCFIPS org.bouncycastle.fips.approved_only=true pf.hsm.hybrid=false

7- CONCLUSION

Dans ce chapitre nous avons présenté les providers JCA /JCE de cryptographie dans Java et le package cryptographique Bouncy Castel, qui seront utilisés pour la réalisation de système de signature numérique propos

CHAPITRE V: Réalisation

1 Introduction

Le choix du langage s'est porté sur le **Java**, qui étant Orienté Objet à la base et conçu pour être de type sécurisé et facile à utiliser. Il contient une architecture de sécurité destinée à protéger les ressources locales de code chargé à distance. Notre choix s'est porté sur l'EDI **NetBeans,** qui nous fournit le confort et la simplicité nécessaires à un développement propre et rapide. Dans le présent chapitre nous présentons le système que nous avons conçu. Les principales interfaces du système sont montrées par des captures d'écran.

2 Les outils utilises pour le codage

2-1 L'environnement de développement Netbeans

C'est un environnement de développement intégré (IDE) pour Java, placé en open source par Sun en juin 2000 sous licence CDDL (Common Développement and Distribution License). En plus de Java, NetBeans permet également de supporter de différents autres langages, comme Python, C, C++, XML et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages web) [24].

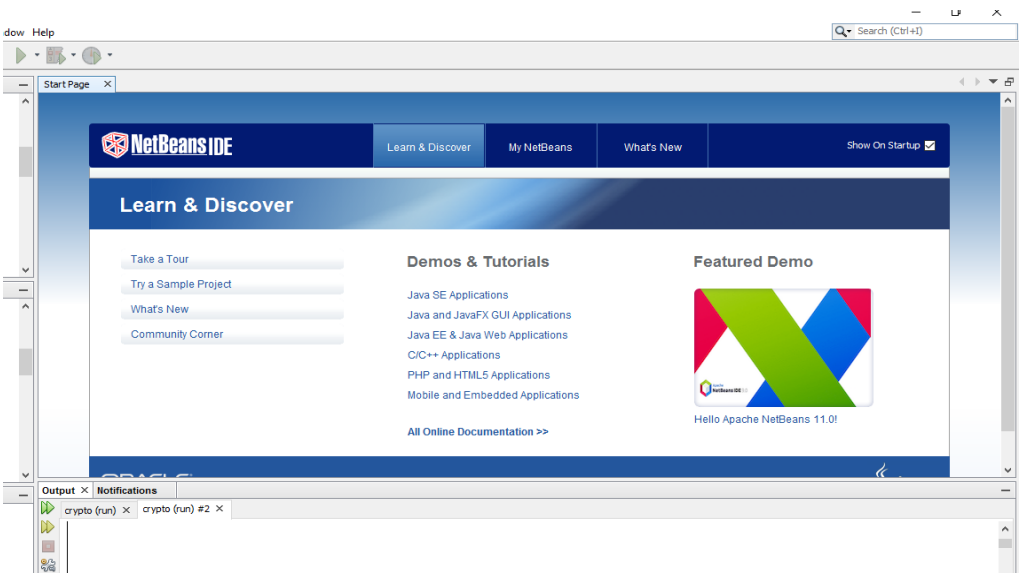

Figure 33 l'IDE NetBeans

2-2 L'API JCA/JCE

Pour utiliser les fonctions de cryptographie java nous avons ajouté le package JCE dans notre JDK (voir chapitre IV).

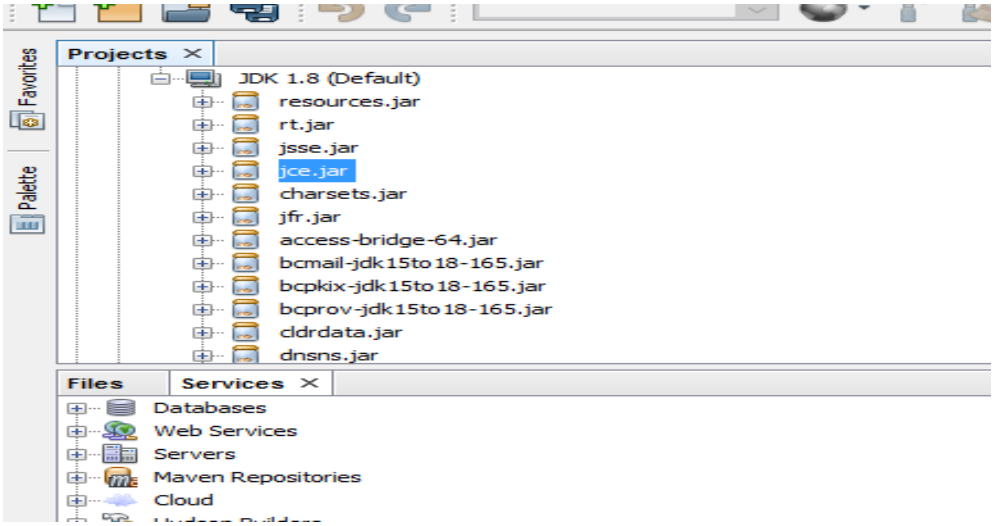

Figure 34. Intégration du package jar

2-3 La bibliothèque Bouncy Castel

BouncyCastle est une bibliothèque Java qui complète l'extension Java Cryptographic Extension (JCE) par défaut (voir chapitre IV).

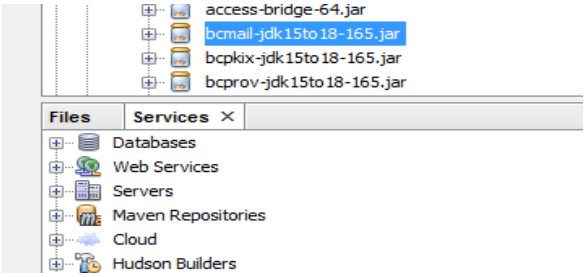

Figure 35.Intégration des packages bouncy castel dans le JCE

2-4 Le logiciel IText

est une interface de programmation partiellement à code source ouvert servant à créer et manipuler des documents PDF. Écrit en langage Java, en .NET (iTextSharp) ainsi qu'en Java compatible avec Android (iTextG). Il est distribué sous licence AGPL et propriétaire pour certaines parties, *iText* permet par exemple de:

- créer un fichier PDF , et l'afficher dans un navigateur ;
- créer des documents dynamiques à partir de sources telles que, des fichiers XML ou des bases de données ;
- ajouter ou supprimer de l'interactivité (par exemple, rendre un formulaire interactif non modifiable une fois qu'il est rempli) ;
- ajouter des marque-pages, des numéros de pages, des filigranes ;
- découper, concaténer et manipuler des pages de fichiers PDF ;

- automatiser le remplissage de formulaires au format PDF. Par exemple pré-remplir le nom dans une facture ;

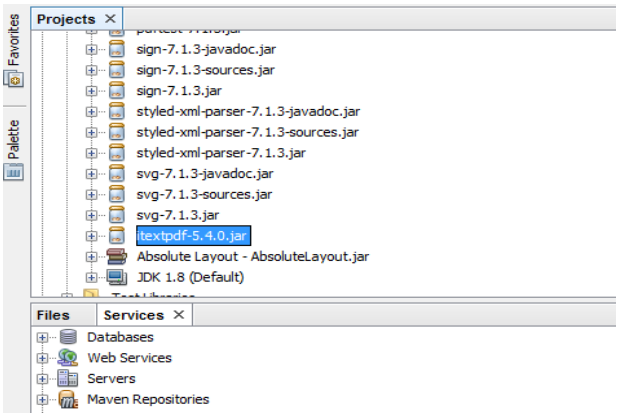

Figure 36.bibliothèque ITEXT

3 Les interfaces essentielles de l'application développée

3-1 l'authentification

Figure 37 l'interface de l'authentification

3-2 Signature d'un document

Figure 38 . Interface de signature

3-3 Signer un document PDF

Pour manipuler un document PDF on a utilisé la bibliothèque ITEXT5**,**

```
import com.itextpdf.text.Document;
import com.itextpdf.text.DocumentException;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.pdf.BaseFont;
import com.itextpdf.text.pdf.PdfContentByte;
import com.itextpdf.text.pdf.PdfReader;
import com.itextpdf.text.pdf.PdfStamper;
import com.itextpdf.text.pdf.PdfWriter;
```

Figure 39 Packages ITEXT utilisé pour manipuler le PDF à signer

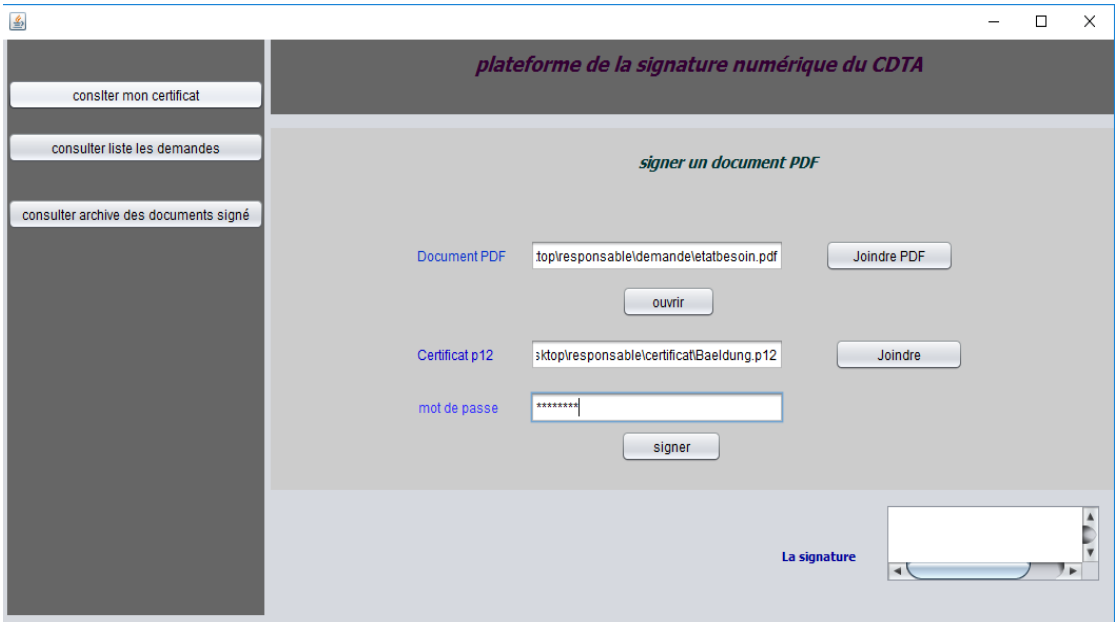

Figure 40. Interface de signature d'un document PDF

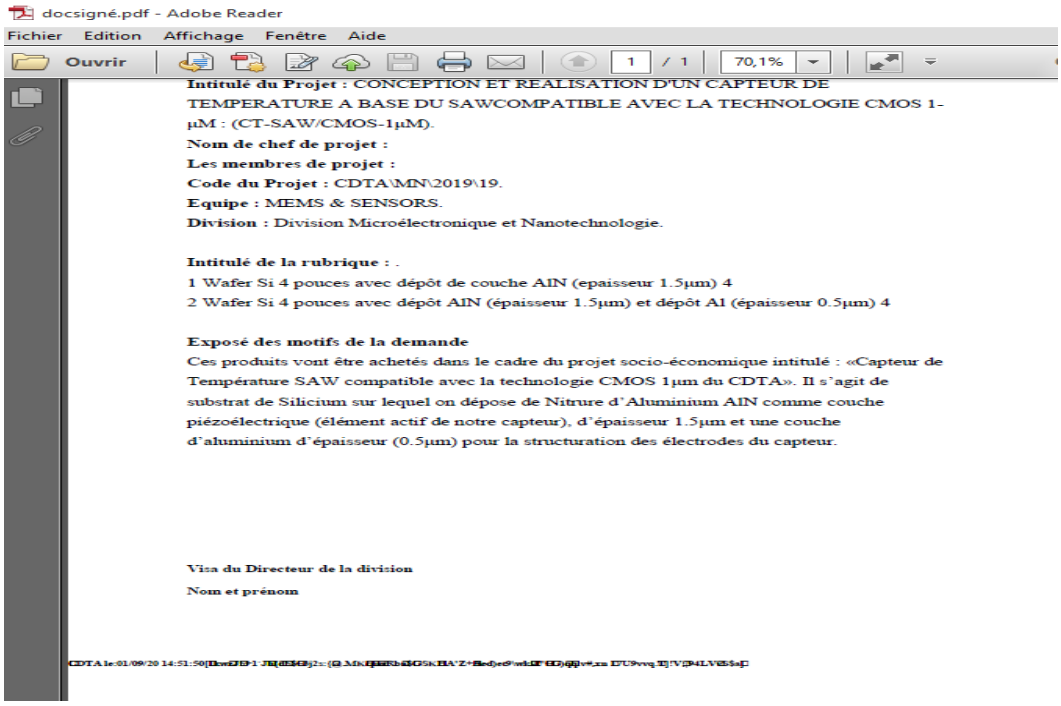

Figure 41. Document PDF signé

3-4 Signer document word

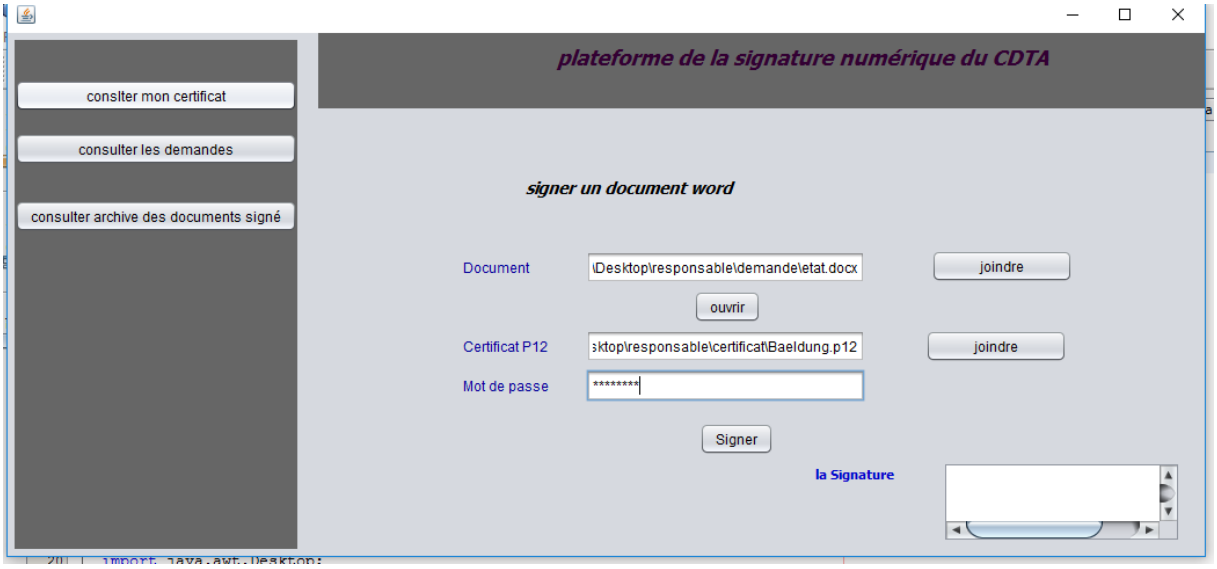

Figure 42 interface de signature d'un document WORD.

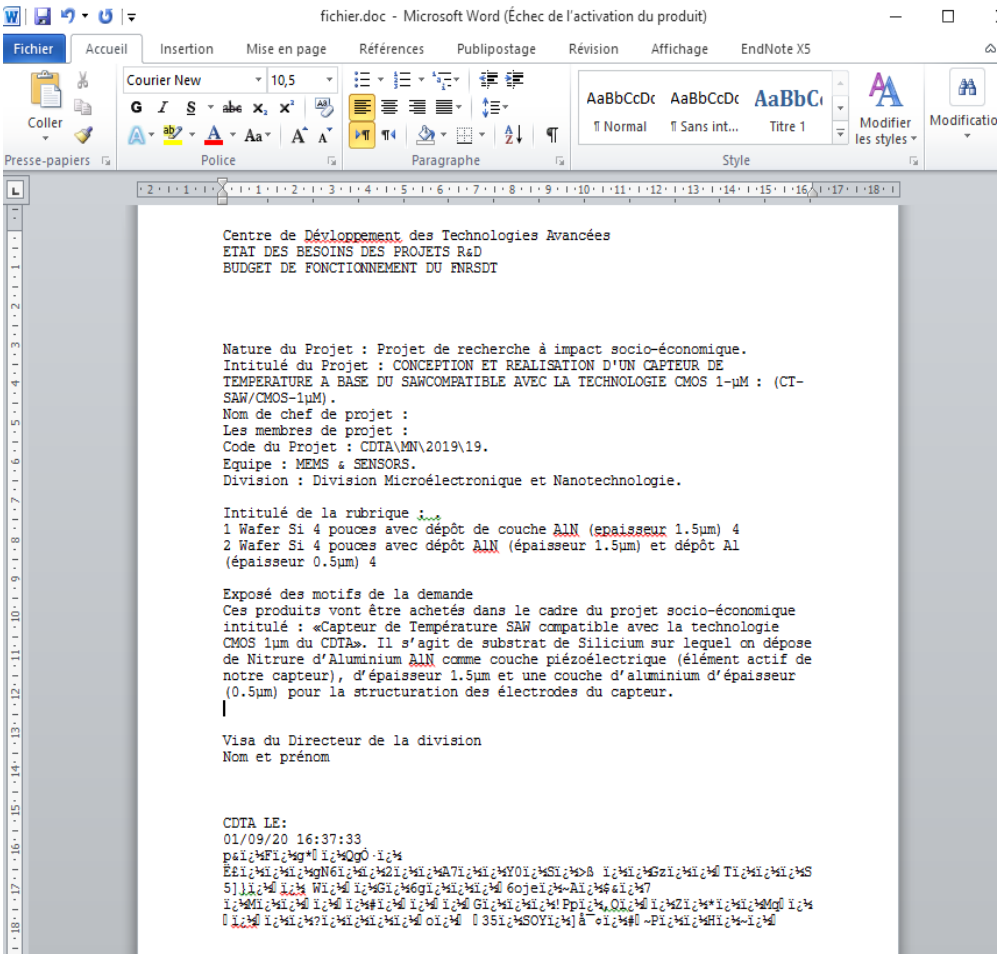

Figure 43 Document Word signé

3-5 Signer document text (.txt)

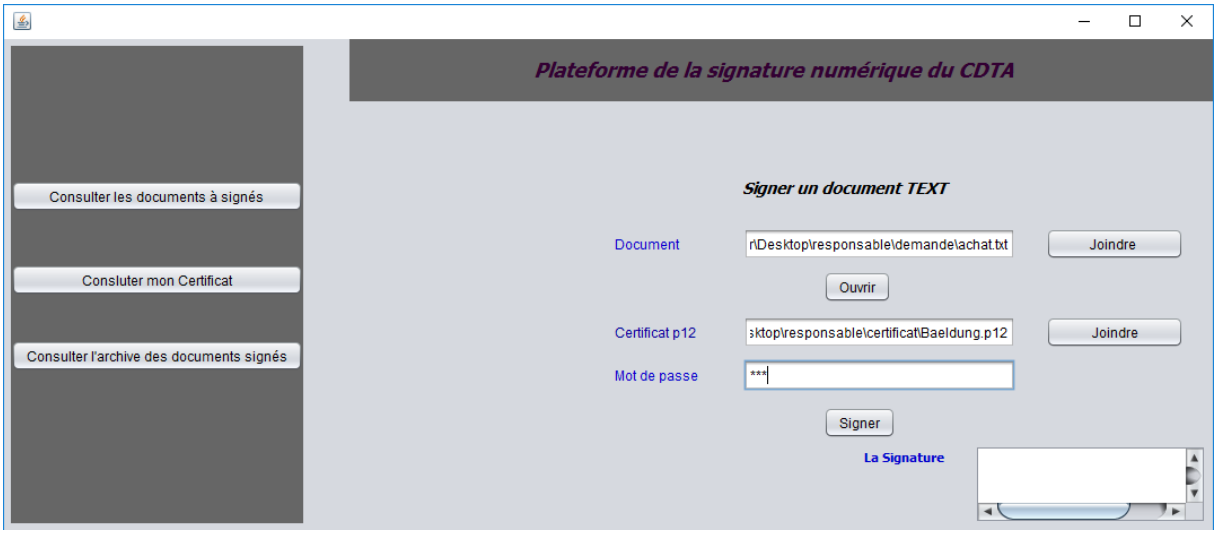

Figure 44. Interface de signature d'un document txt

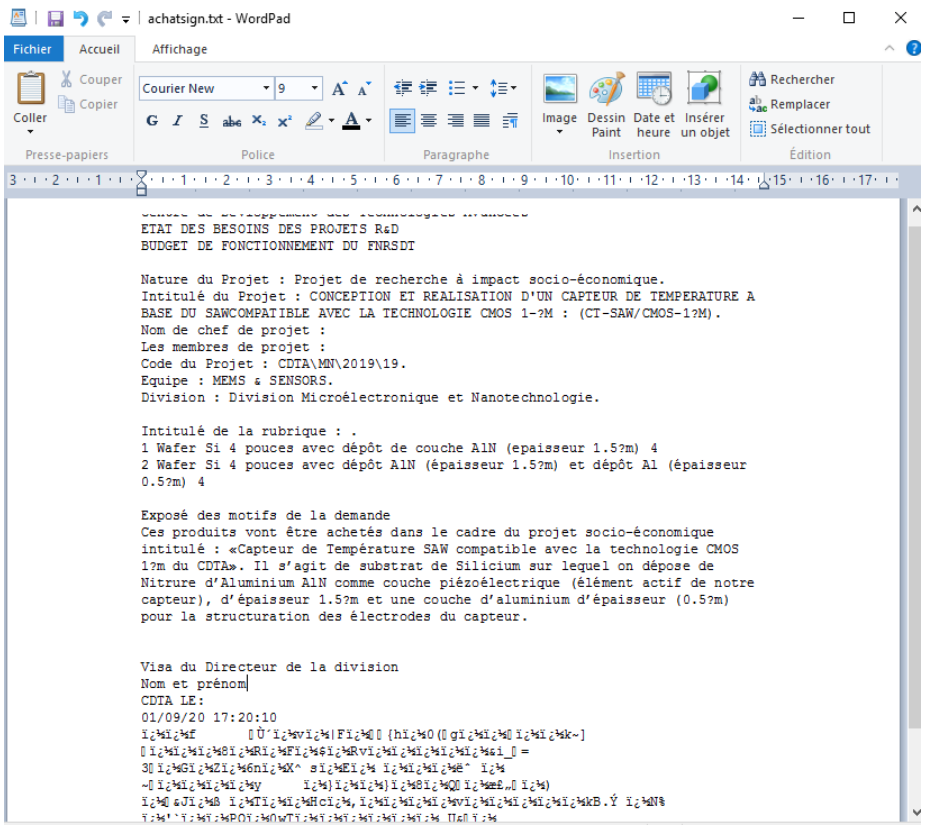

Figure 45. Document .txt signé

On a utilisé l'algorithme SHA pour calculer l'empreinte de document.

```
MessageDigest md = MessageDigest.getInstance("SHA-256", "BC");
byte[] empreinte = Hex.encode(md.digest(buffer));
```

Concernant le cryptage avec la clé privée on a utilisé l'algorithme RSA.

```
Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
```

3-6 Consulter mon certificat

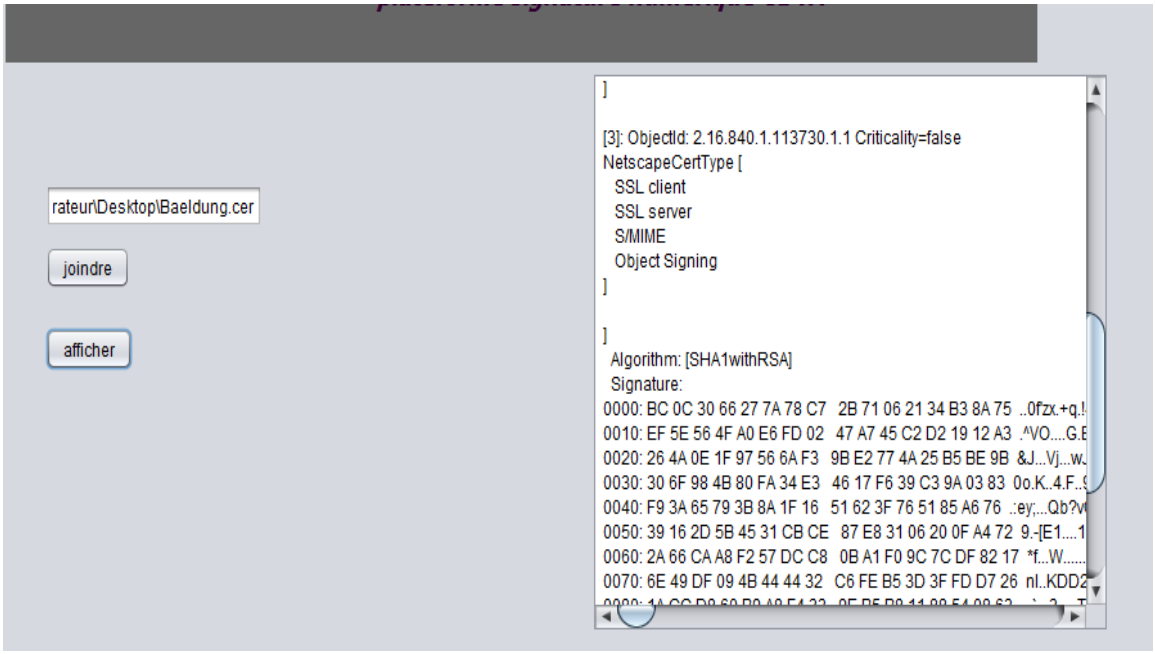

Figure 46. Afficher un certificat

3-7 Extraire clé publique

On a utilisé Java *KeyStore* qui est une base de données qui peut contenir des clés. Un Java KeyStore est représenté par la classe KeyStore(java.security.KeyStore). ainsi que le standard pkcs12.

```
Security.addProvider(new BouncyCastleProvider());
try {
    ks = KeyStore.getInstance("PKCS12");
```

Figure 47 Extraire clé publique

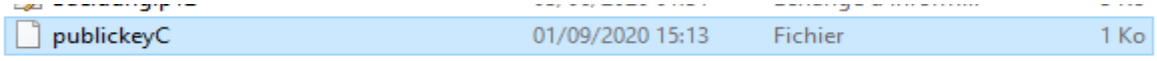

Figure 48 Création de la clé

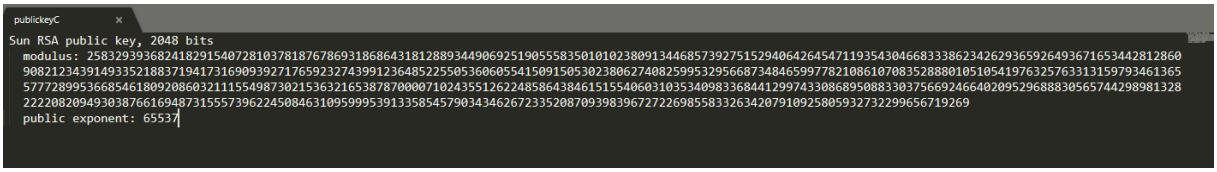

Figure 49. Affichage de la clé

3-8 La vérification de la signature

La vérification se fait par le décryptage de la signature et la vérifier avec l'empreinte du document

```
cipher.init(Cipher.DECRYPT_MODE, publickey);
        byte[] newMD = cipher.doFinal(cipherText);

  Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
```

Figure 50. L'instance de l'algorithme utilisé

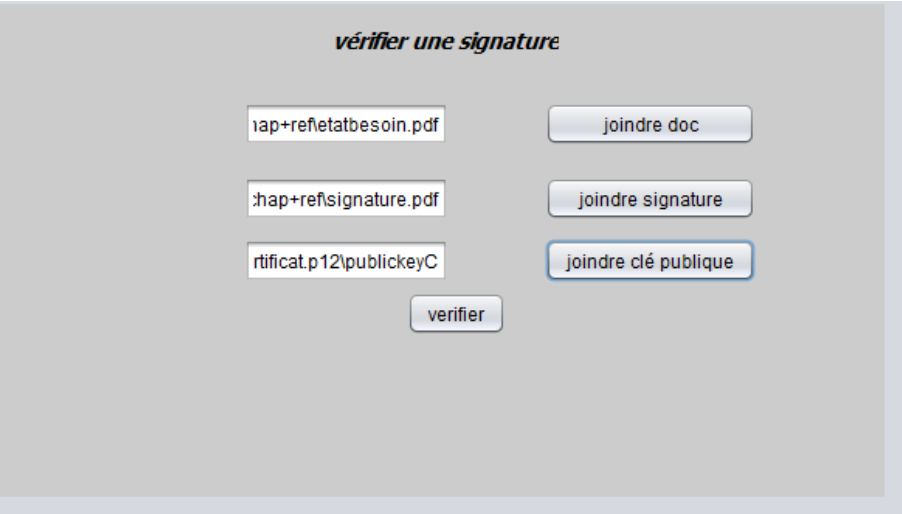

Figure 51. Vérification de la signature

4 CONCLUSION

Dans ce chapitre, nous avons présenté les différents outils et technologies qui ont été utilisés pour réaliser cette solution, nous avons achevé les parties essentielles de la solution en respectant la conception de l'application. Quelques interfaces ont été présentées pour bien clarifier les étapes d'utilisation de la solution proposée dans ce projet de fin d'étude.

Conclusion générale

La signature numérique est un mécanisme permettant de garantir l'intégrité d'un document électronique et d'en authentifier l'auteur, par analogie avec la signature manuscrite d'un document papier. La signature numérique va permettre aux établissements qu'ont l'adopté de réduire les dépenses et de gagner du temps et de la flexibilité. Ce projet de master a été réalisé afin de répondre à un besoin exposé au niveau de Centre de Développement des Technologies Avancées concernant le développement d'un système de signature numérique.

Afin de fixer les idées sur le domaine de la signature numérique et sa terminologie, une étude théorique approfondie a été donnée tout au début de ce mémoire. L'analyse et la conception du système réalisé ont été détaillées dans un chapitre séparé, ces détails nous ont permis de réaliser d'une façon solide le système de signature. Le système réalisé a été exposé dans la dernière partie de ce mémoire.

Dans ce projet nous avons réalisé un système de signature numérique des documents de différents formats (PDF, WORD, TXT) basée sur les PKI. Ce dernier est basé sur des forces cryptographiques considérables, ou on a réalisé cet outil avec le langage Java et on a intégré des providers de service cryptographique, et on s'est appuyé sur les standard PKCS (les standards cryptographiques à clé publique).

Notre travail est considéré comme une première version de projet « signature numérique » au sien de Centre de Développement des Technologies Avancées et au même temps est un maillon très important dans le système de numérisation du Centre. Au même temps le système de signature numérique réalisé est générique car il est utilisable pour et par tout Enterprise qui le souhaite et qui dispose des moyens pour l'adopter.

Ce travail constituant un grand chantier, des améliorations ultérieures sont à venir. Ainsi, en guise de perspectives, nous prévoyons de transformer cette application en plusieurs applets, et on les intègre dans des workfows de différentes applications de numération pour la flexibilité de la signature. L'amélioration vers la signature multiple est une étape très importante pour la mise en valeur de la signature numérique.

Les références

1 Anas Abou El Kalam, Rania El Baida, Philippe Balbiani, Salem Benferhat, Frédéric Cuppens, Yves Deswarte, Alexandre Miege, Claire Saurel, and Gilles Trouessin, 'Or-Bac: Un Modèle De Contrôle D'accès Basé Sur Les Organisations', *Cahiers francophones de la recherche en sécurité de l› information. II, 30,* 43 (2003).

2 Mr ABDERRAHIM Amine, Mr CHIKH Azzeddine, and Mr CHOUITI S-Mohammed, 'Suivie Des Enseignements Du Lmd Par Application De La Méthode 2tup'.

3 Serge Aumont, Roland Dirlewanger, and Olivier Porte, 'L'accès Sécurisé Aux Données', *3ème journée des réseaux-JRES,*(1999).

4 Oussama AZZOUZI, 'Système Embarqué Flexible Pour Un Chiffrement Hybride Symétrique/Asymétrique',Thèse Magister, Ecole Nationale Supérieure d'informatique,(2012).

5 Pierre Barthélemy, Robert Rolland, and Pascal Véron, 'Cryptographie', *Edition Hermès-Lavoisier* (2005).

6 Toufik Bekkouche, 'Développement Et Implémentation Des Techniques De Cryptage Des Données Basées Sur Les Transformées Discrètes', Thèses de doctorat, (2018).

7 Amina Bendouma, 'Développement D'une Infrastructure De Gestion De Clés De Cryptage Dans Les Réseaux Ad Hoc Véhiculaires (Vanet)', Thèse , Université du Québec à Trois-Rivières, (2017).

8 Côme Berbain, 'Analyse Et Conception D'algorithmes De Chiffrement À Flot', Thèse de doctorat en Informatique, Paris 7, (2007).

9 Ianis BERNARD, 'Cryptanalyse De Chiffrement Par Flot', International university, cours,(2015).

10 Ryma Dr BOUSSAYOUD, Chehla BENHADJI, Labiba CHIOUKH, Doha AFER, Djihan BOUCHAIR, and Nesrine TITI, 'Cryptage/Chiffrement & Tatouage Des Données Numériques' , Mémoire MastreII , Université de JIJEL ,(2019).

11 Jon Callas, Lutz Donnerhacke, Hal Finney, and Rodney Thayer, 'Openpgp Message Format', RFC 2440, (November 1998).

12 YACINE CHALLAL, 'Infrastructures À Clés Publiques', in *Ingénierie des Protocoles et Logiciels Sécurisés*, cours, (2015).

13 Rodolphe CARDON DE LICHTBUER, 'Signatures Électroniques Dans Les Applications Internet',Mémoire d'ingéniorat, ECOLE ROYALE MILITAIRE Bruxelles, (2006).

14 BESMA DEBBAGH, and NOURA BOUNEGEB, 'Eude Et Comparaison De Principaux Systèmes Crypto Fournis Par Le Package De Bouncy Castle Plate Forme Java Sdk',Mémoire MasterII, université Ouargla(2016).

15 Saidou Diop, 'Une Infrastructure À Clés Publiques (Pki) Pour Sécuriser Les Messages Dans Un Réseau V2g', Thèse, Université du Québec à Trois-Rivières, (2018).

16 Yevgeniy Dodis, and Nelly Fazio, 'Public Key Broadcast Encryption for Stateless Receivers', in *ACM Workshop on Digital Rights Management*Springer, (2002), pp. 61-80.

17 Walid DOUCENE, 'Infrastructures À Clés Publiques Basées Sur La Technologie Blockchain', Mémoire MasterII, UNIVERSITE MOHAMED BOUDIAF-M'SILA-FACULTE MATHEMATIQUES ET DE L'INFORMATIQUE,(2019).

18 S Dusse, P Hoffman, B Ramsdell, L Lundblade, and L Repka, 'Rfc2311: S/Mime Version 2 Message Specification', RFC Editor,(1998).

19 Donald Eastlake, Joseph Reagle, David Solo, Frederick Hirsch, and Thomas Roessler, 'Xml-Signature Syntax and Processing', *W3C recommendation,* (2002).

20 Mr Mohamed EL MARRAKI, Nasser Yassine, and Ouyous Mina, 'Rapport Sur L'étude Et L'implémentation De Quelques Algorithmes De Chiffrement Et De Signature',Master informatique,Faculté des science Rabat,2014.

21 'Explications Sur La Cryptographie', Tice-Education, (Publication : 16 mai 2020).

22 Riccardo Focardi, Francesco Palmarini, Marco Squarcina, Graham Steel, and Mauro Tempesta, 'Mind Your Keys? A Security Evaluation of Java Keystores', in *NDSS*, (2018).

23 Paul Axayacatl Frausto Bernal, 'Icare-S2: Infrastructure De Confiance Sur Des Architectures De Réseaux Pour Les Services De Signature Évoluée', Paris, ENST, (2004).

24 Inaàm Sihem GHAFFOUR, and Rafika BENLEDGHEM, 'Réalisation D'une Application Client/Serveur (Gestion D'une Banque) Avec Jdbc Et Mysql Selon Le Modèle À Trois Couches Sous Netbeans',Mémoire Master II,Université de Telemcen(2013).

25 Li Gong, Gary Ellison, and Mary Dageforde, *Inside Java 2 Platform Security: Architecture, Api Design, and Implementation*Addison-Wesley Professional, (2003).

26 James Gosling, 'Java: An Overview', in *Sun Microsystems Laboratories: The First Ten Years*Citeseer, (1995), pp. 1991-2001.

27 Clare Healy, Colin Riordan, and Jack L Kelly, 'Bouncy Castle Burns', *Burns: journal of the International Society for Burn Injuries,* 32 (2006), 920-21.

28 David Hook, '*Beginning Cryptography with Java*',book, ISBN: 978-0-764-59633-9 (August 2005).

29 'Institut D'électronique Et D'informatique Gaspard-Monge (Igm)', (2002).

30 M Issad, M Anane, B Boudraa, and N Anane, 'Implémentation Du Crypto Système Rsa Dans un Environnement Sopc', Conference: "Sixième Edition du Séminaire sur les Systèmes de Détection : Architectures et Technologies " DAT' (2014) At: alger, algerie.

31 X ITU-T RECOMMENDATION, 'Information Technology–Open Systems Interconnection–the Directory: Public-Key and Attribute Certificate Frameworks', (2000).

32 Jérémy Jean, 'Cryptanalyse De Primitives Symétriques Basées Sur Le Chiffrement Aes', *These de doct. Ecole Normale Supérieure* (2013).

33 Jérôme Lafosse, *Java Ee: Guide De Développement D'applications Web En Java*Editions ENI, 2009).

34 Graham Jenkins, "Utiliser PGP avec Java et Bouncy Castle", Linux Gazette n°98 , (Février 2004).

35 Michael McIntosh, and Paula Austel, 'Xml Signature Element Wrapping Attacks and Countermeasures', in *Proceedings of the 2005 workshop on Secure web services*, 2005), pp. 20-27.

36 Richard A Mollin, *An Introduction to Cryptography*CRC Press, 2000).

37 'Pkcs', glasnost.entrouvert.org,(Accessed 2003).

38 Hind Rakkay, 'Approches Formelles Pour La Modélisation Et La Vérification Du Contrôle D'accès Et Des Contraintes Temporelles Dans Les Systèmes D'information', École Polytechnique de Montréal, (2009).

39 Bruce Schneier, 'Cryptographie Appliquée, Algorithmes, Protocoles Et Codes Source En C', *Seconde édition, Vuibert Informatique* (2001).

40 Kamarudin Shafinah, and Mohammad Mohd Ikram, 'File Security Based on Pretty Good Privacy (Pgp) Concept', *Computer and Information Science,* 4 (2011), 10.

41 Gaël Thomas, 'Design Et Analyse De Sécurité Pour Les Constructions En Cryptographie Symétrique', Thèse de doctorat en Mathematiques et applications, École doctorale Sciences et ingénierie pour l'information, mathématiques,Limoge,(2015).

42 BELLAL Toufik, Mr E Cousin, Mr F Cuppens, and Mme N Cuppens, '*Expression D'une Politique De Sécurité Dans Un Réseau Social',Article,* département d'informatique médicale,UNIV EUROPEENNE, (2015).

43 Julien Vayssière, 'Une Architecture De Sécurité Pour Les Applications Réflexives: Application À Java', Thèse de doctorat, Nice, (2002).

44 Alexandre VILLOING, 'Implémenter Une Infrastructure À Clés Publiques Dans Un Environnement Windows Server 2003', (posté en 2006).

45 Ahmad Samer Wazan, Romain Laborde, François Barrere, and Abdelmalek Benzekri, 'Etude Du Concept De Confiance Pour Les Infrastructures À Clés Publiques', 10ème Colloque francophone Gestion de REseaux et de Services (GRES 2014), 1 December 2014 - 3 December 2014 (Paris, France).

46 Von Welch, Ian Foster, Carl Kesselman, Olle Mulmo, Laura Pearlman, Steven Tuecke, Jarek Gawor, Sam Meder, and Frank Siebenlist, 'X. 509 Proxy Certificates for Dynamic Delegation', in *3rd annual PKI R&D workshop*, 2004).