République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la recherche Scientifique
Université A.MIRA
Faculté des Sciences Exactes
Département Informatique

Mémoire de Master en Informatique

Option : Administration et Sécurité des Réseaux

Thème

Infrastructure de gestion de clés publiques avec EJBCA

Réalisé par :

-MEZHOUD Karim

-MOKHTARI Hamza

Devant le jury composé de :

-Président : Mr M.SAADI

-Examinateur : Mr M.OMAR

-Examinateur : Mr A.ACHROUFANE

-Encadreur : Mr A.R SIDER

Promotion 2011-2012

Dédicaces

Je dédie ce modeste travail à :

A toutes ma familles, spécialement a mes chers parents qui m'on soutenus pendant ce travaille, sans oublier Chahra et tout mes amis.

Dédicaces

Je dédie ce modeste travail à :

Mes chers parents qui ont été présents à chaque fois que j'avais besoin d'eux.

Mes frères Ali, Nabil et ma sœur Rima.

Tous mes amis.

Remerciements

Nos vifs remerciements sont adressés :

En premier lieu, à Dieu tout puissant pour nous avoir donné la force et le savoir afin d'accomplir ce travail.

A nos parents pour nous avoir soutenus et pour tous les sacrifices consentis, ainsi qu'à tous nos proches.

A notre encadreur, Mr A.R Sider pour nous avoir guidés tout au long de ce projet.

Aux membres de la commission pour avoir jugé notre modeste travail.

Et à tous ceux qui ont participé de près ou de loin à l'accomplissement de ce projet.

TABLE DES MATIERES

LISTE DES FIGURES ET TABLEAUX

FIGURES :

TABLEAUX :

LISTE D'ABREVIATIONS

AC	Autorité de certification
AE	Autorité d'enregistrement
AEL	Autorité d'enregistrement locale
AES	Advanced Encryption Standard
ANT	Another Neat Tool
CMP	Certificate Management Protocol
CRL	Certificates Revocation List
DES	Data Encryption Standard
DN	Distinguished Name
DSA	Digital Signature Algorithm.
EE	End entity (entité d'extrémité)
EJBCA	Entreprise Java Bean Certificate Authority
JCE	Java Cryptography Extension
JDBC	Java Data Base Connectivity
JDK	Java Development Kit
JKS	Java KeyStore
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
ICP	Infrastructure à clé publique
IDEA	International Data Encryption Algorithm
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
LDAP	Lightweight Directory Access Protocol

LDAPS	Lightweight Directory Access Protocol Secure
LGPL	GNU Lesser General Public License
MD5	Message Digest 5
MYSQL	My Structured Query Language
NIST	National Institute of Standards and Technology
OCSP	Online Certificate Status Protocol
PEM	Privacy Enhanced Mail
PKCS	Public-Key Cryptography Standards
PKI	Public-Key Infrastructure
PKIX	Public-Key Infrastructure X.509
RDN	Relative Distinguished Name
RFC	Request For Comments
RSA	R.Rivest, A.Shamir et l.Adleman.
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SSL	Secure Socket Layer
TLS	Transport Layer Security
UIT	Union Internationale des Télécommunications
UIT-T	Union Internationale des Télécommunications section des Télécommunications

INTRODUCTION GENERALE

Le commerce électronique s'accroît de plus en plus ; un nombre important de documents sous forme électronique sont produits, échangés ou archivés (factures, bons de commande, e-mails, courriers, …). Certaines de ces informations dites confidentielles peuvent subir des modifications intentionnelles (par un intrus) ou accidentelles.

En effet, les données qui transitent sur Internet, sont sujettes à diverses attaques lorsque les entités échangent leurs clés publiques. De plus, le nombre d'applications qui utilisent les clés publiques et les certificats numériques pour sécuriser les différentes transactions est en augmentation considérable.

La sécurité des systèmes d'information est donc un enjeu majeur pour toute entreprise ou organisation, surtout depuis l'avènement et le développement d'internet et de l'intégration des systèmes d'information entre eux.

Une Infrastructure de gestion de clés publiques connu sous le nom PKI (Public key Infrastructure) est un système de gestion de clés publiques qui permet de gérer des listes importantes de clés publiques. Elle permet aussi de créer, gérer, distribuer, utiliser, stocker et révoquer des certificats numériques.

La PKI permet de sécuriser les communications réseaux en assurer les quatre principes fondamentaux de la sécurité informatique qui sont : l'authentification, la confidentialité, l'intégrité et la non-répudiation.

L'université de Bejaia, comme toute université, produit, échange, gère de l'information et des documents à travers différentes applications : messagerie électronique, serveur web,…. De plus, de nombreuses clés publiques sont manipulées par les différents services de l'université de Béjaia. Face à cette montée en puissance, la certification de ses applications et

la gestion des listes importantes de clés publiques servant pour leur sécurité est devenu un besoin fondamental.

Ceci nous a amené à nous interroger sur la façon de sécuriser les différentes applications utilisées par les services de l'université de Béjaia. En d'autres termes, quelles sont les solutions possibles pour certifier ses applications et fournir une bonne gestion de clés publiques ?

Pour mieux cerner cette problématique, nous avons pris le cas du centre de calcul de l'université de Béjaia et nous allons mettre en place un système de gestion de clés publiques qui aura comme rôle la gestion, la distribution, l'obtention des clés publiques au moyen de certificats, et la publication des certificats obsolètes.

Les grandes questions que l'on se pose dans ce travail sont les suivantes :

- Comment sécuriser les applications du centre de calcul ?
- Comment fonctionne une PKI ?
- Quelle est la solution la plus puissante pour mettre en œuvre une PKI ?
- Comment organiser les différents certificats générés par la PKI ?
- Comment valider les certificats numériques ?
- Comment faciliter la récupération des certificats ?

Dans ce mémoire, qui s'articule autour de trois chapitres, nous commençons par présenter, dans le premier chapitre, les notions de base de la sécurité informatique. Le deuxième chapitre, quant à lui, sera consacré à la présentation de l'infrastructure de gestion de clés publiques. En dernier lieu, dans le troisième et dernier chapitre, nous allons décrire la mise en place d'une PKI avec EJBCA ainsi qu'une application test.

CHAPITRE I :
NOTIONS DE BASE SUR LA SECURITE INFORMATIQUE

Introduction

La sécurité informatique est définie comme étant l'ensemble des moyens techniques, organisationnels, juridiques et humains nécessaires, mis en place pour conserver, rétablir et garantir la sécurité des systèmes informatiques. Elle a pour objectif d'assurer les quatre principales fonctionnalités suivantes : l'authentification, la confidentialité, l'intégrité et la non-répudiation. Ces exigences sont vitales si l'on désire effectuer une communication sécurisée à travers un réseau informatique tel qu'Internet.

La mise en œuvre de ces services de sécurité est rendue possible grâce à la cryptographie moderne qui offre différents mécanismes de sécurité tels que le chiffrement et la signature numérique.

Dans ce chapitre nous allons décrire les objectifs principaux de la sécurité, ainsi qu'un rappel sur les notions de base de la cryptographie (le chiffrement/déchiffrement, les clés, les familles cryptographiques, la fonction de hachage, la signature numérique, le certificat numérique) et enfin nous allons définir les deux protocoles SSL et TSL.

I.1 Normes et Standards

I.1.1 ISO 7498 :

« L'ISO (Organisation internationale de normalisation) est le plus grand producteur et éditeur mondial de Normes internationales… L'ISO permet ainsi d'établir un consensus sur des solutions répondant aux exigences du monde économique et aux besoins plus généraux de la société.» [REF01].

« ISO7498 décrit le modèle de référence de base pour l'interconnexion de systèmes ouverts(OSI). Elle définit les règles liées à la sécurité des éléments architecturaux qui peuvent être appliquées de façon appropriée dans les circonstances pour lesquelles la protection de la communication entre systèmes ouverts est requise. Il établit, dans le cadre de la référence du modèle, les lignes directrices et les contraintes en vue d'améliorer les normes existantes ou à élaborer de nouvelles normes dans le contexte de l'OSI afin de permettre des communications sécurisées et de fournir ainsi une approche cohérente de la sécurité dans l'OSI.

La norme ISO 7498-2, de 1989, décrit cinq grandes catégories de "services" relatifs à la sécurité : le contrôle d'accès, l'authentification, la confidentialité et l'intégrité des données, la non répudiation. » [REF02].

I.1.2 IETF

« L'Internet Engineering Task Force (IETF, groupe de travail sur les technologies et protocoles de l'Internet) est une des organisations de pointe qui se charge de promouvoir les normes de l'Internet.

Cela signifie que son but est de mieux faire fonctionner l'Internet en produisant des documents techniques pertinents de haute qualité qui influencent la manière dont les gens créent, utilisent et gèrent l'Internet. » [REF03].

Parmi les principaux documents appelés Requests For Comments (RFC) développés par ce groupe, voici ceux qui sont les plus pertinents pour la présente étude :

- ***RFC 5280* - Certificate and Certificate Revocation List (CRL) Profile:** Spécification qui décrit le format du certificat X.509 v3 avec ses extensions, ainsi que le mécanisme de liste de révocation pour son utilisation sur l'internet. Cette RFC constitue la proposition du groupe de travail PKIX[1] pour l'utilisation de l'ICP (Infrastructure à clé publique) sur l'internet *[RFC5280]*.

- ***RFC 2560* - Online Certificate Status Protocol (OCSP) :** Spécification qui décrit le protocole OCSP. Ce protocole permet de déterminer le statut courant d'un certificat à clé publique, sans avoir à recourir à l'utilisation d'une liste de révocation. Le protocole

[1] Le Groupe de travail PKIX a été créé en automne 1995 dans le but d'élaborer des normes internet pour soutenir les infrastructures à clé publique basés sur X.509.

OCSP permet donc à une composante logicielle de faire appel à un tiers de confiance, afin de déterminer si un certificat a fait l'objet d'une révocation ou non. Ce mécanisme a pour avantage de faciliter la gestion des listes de révocation *[RFC2560]*.

- ***RFC 3161*** **- Time-Stamp Protocol (TSP) :** Spécification qui décrit le protocole d'échange avec le service d'horodatage Time Stamping Authority (TSA). Le service TSA permet de générer des assertions qui attestent qu'une donnée existait à un moment précis dans le temps. Ce service est utilisé par les services de non-répudiation *[RFC3161]*.

- ***RFC 4210*** **- Certificate Management Protocol (CMP) :** Spécification qui décrit le protocole CMP. Le protocole CMP est utilisé pour la création, ainsi que la gestion des certificats X.509v3 à distance. Les composantes de l'ICP utilisent ce protocole pour communiquer entre elles. Cette spécification remplace la *RFC 2510 [RFC4210]*.

- ***RFC 2251*** **: - Lightweight Directory Access Protocol (LDAP) v3 :** Spécification qui décrit le protocole LDAP en termes de X.500 comme mécanisme d'accès X.500. Un serveur LDAP doit agir selon la série de recommandations de l'UITX.500(1993) en fournissant le service. Cependant, on n'exige pas qu'un serveur LDAP se serve d'un quelconque protocole X.500 en fournissant ce service, par exemple LDAP peut être mappé sur n'importe quel autre système d'annuaire à condition que le modèle des données et du service X.500 utilisé dans LDAP ne soit pas enfreint dans l'interface LDAP*[RFC2251]*.

- ***RFC 2246 :*** **- Transport Layer Security (TLS) :** Spécification qui décrit le protocole TLS, comporte un dispositif pour négocier le choix d'une méthode de compression des données sans perte au titre du protocole de prise de contact TLS et pour ensuite appliquer l'algorithme associé à la méthode choisie au titre du protocole d'enregistrement TLS. TLS définit une méthode standard de compression qui spécifie que les données échangées via le protocole d'enregistrement ne seront pas compressées *[RFC2246]*.

- ***RFC 1320*** **: Message-Digest Algorithm (MD4) :** Ce document décrit en détail la fonction de hachage MD4 conçu par le professeur Ronald Rivest en 1990.

- ***RFC 1321* : Message-Digest Algorithm (MD5) :** Ce document décrit en détail la fonction de hachageMD5 conçu par le professeur Ronald Rivest en 1991

I.1.3 UIT-T

« L'Union Internationale des Télécommunications ou UIT (International Telecommunication Union ou ITU, en anglais), dont le siège se trouve à Genève, est devenue une institution spécialisée de l'ONU en 1947…Le but de l'UIT est de réglementer, planifier et normaliser les télécommunications internationale de toutes sortes afin de faciliter l'interconnexion des réseaux et des systèmes.

L'UIT-T se compose de cinq organes permanents parmi lesquels on trouve le Comité Consultatif International Télégraphique et Téléphonique (CCITT) en cahrge des réseaux de télécommunications, des signaux téléphoniques et téléinformatiques, de la transmission par câble du téléphone, des données et de la télévision. Suite à une réorganisation de l'UIT fin 1992, le CCITT disparaît et devient UIT-T (UIT – secteur de normalisation des télécommunications). L'UIT-T publie des recommandations désignées par une lettre pour chaque domaine et un numéro. » [REF04].

I.1.4 PKCS (Public Key Cryptography Standards):

« Spécifications développées par les laboratoires de la société RSA en vue d'accélérer le déploiement de la cryptographie à clé publique. Les PKCS servent de référence dans le monde de la cryptographie ». [REF05]

I.2 Les objectifs de la sécurité informatique

La sécurité informatique repose sur quatre principes fondamentaux (ou services) : l'authentification, la confidentialité, l'intégrité et la non-répudiation.

I.2.1 Authentification :

L'authentification est un mécanisme qui sert à s'assurer que l'identité de la personne avec laquelle on communique est bel est bien celle qu'elle prétend être.

Elle est généralement mise en œuvre grâce à une combinaison d'un nom d'utilisateur et d'un mot de passe. Une partie du processus d'authentification consiste à identifier correctement un utilisateur, une application, ou un groupe.

I.2.2 Confidentialité :

La confidentialité est la propriété qu'une information n'est ni disponible ni révélée aux individus, entités ou processus non autorisés. Les données transportées lors d'une communication ne peuvent pas être lues par un adversaire espionnant les communications.

La confidentialité a donc pour but d'assurer que seul le destinataire peut connaître le contenu des messages ou des données sensibles dites «confidentielles » qui lui sont transmises.

I.2.3 Intégrité :

L'intégrité des données est la propriété qui assure que le traitement, transmission, et conservation des données, ne subissent aucun changement ou destruction volontaire (par un intrus) ou accidentelle.

Si une partie intermédiaire modifie les données d'origines envoyées par l'expéditeur, le destinataire final devrait être capable de détecter qu'il y a eu des changements.

I.2.4 Non répudiation :

La non-répudiation est définie par l'impossibilité pour une des entités impliquées dans une communication de nier avoir participé à l'ensemble ou à une partie de la communication. Elle assure, ainsi, une protection contre le faux démenti d'une entité d'être impliquée dans une communication.

I.3 Rappels sur la cryptographie

La *cryptographie* est la science qui utilise les mathématiques pour chiffrer et déchiffrer des données. La cryptographie sert à stocker des données sensibles ou de les transmettre par des réseaux de communication non sûrs comme Internet, dans le but qu'elles ne puissent être interceptées par personne à l'exception du destinataire souhaité.

La *cryptanalyse* est la science de l'analyse et du cassage des communications sécurisées. Elle se base sur les méthodes analytiques, l'application d'outils mathématiques, voir de la détermination et de la chance. La *cryptologie* englobe à la fois la cryptographie et la cryptanalyse.

I.3.1 Chiffrement/déchiffrement :

Le chiffrement est le mécanisme cryptographique par lequel un message (fichier, courriel, etc.) dit « en clair », est transformé à l'aide d'un algorithme mathématique et d'une clé, de manière à le rendre incompréhensible. Le déchiffrement est l'opération inverse qui, par un procédé similaire, applique une transformation sur un message chiffré, de manière à le ramener dans sa forme compréhensible. Le chiffrement a pour objectif d'assurer la ***confidentialité*** des données.

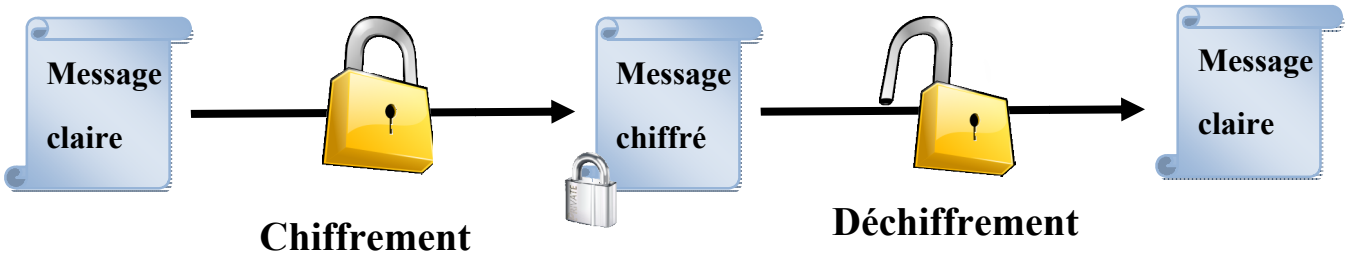

Figure I.1 : Chiffrement et déchiffrement d'un message.

I.3.2 Les clés

« Une clé est une valeur qui est utilisée avec un algorithme cryptographique pour produire un texte chiffré spécifique. Les clés sont, à la base de très grands nombres. La taille d'une clé se mesure en bits; et le nombre qui peut être représenté par une clé de 1024 bits est vraiment immense. Plus grande est la clé, plus grande est la sécurité, mais les algorithmes utilisés pour chaque type de cryptographie sont très différents. Par exemple, en matière de cryptographie à clé publique, plus la clé est grande, plus le chiffrement est sûr. » [REF06].

Il existe deux sortes de clés :

- Clé Publique : quantité numérique, attachée à une ressource ou un individu qui la distribue aux autres afin qu'ils puissent lui envoyer des données chiffrées ou déchiffrer sa signature.
- Clé Privée : quantité numérique secrète attachée à une ressource ou à un individu, lui permettant de déchiffrer des données chiffrées avec la clé publique correspondante ou d'apposer une signature au bas de messages envoyés vers des destinataires.

Bien que la clé publique et la clé privée soient liées, il est très difficile de déduire la clé privée en partant de la seule clé publique; toutefois, il est toujours possible de déduire la clé privée si l'on dispose de suffisamment de temps et de puissance de calcul. Pour une taille allant jusqu'à 512 bits par exemple, il faut faire travailler conjointement plusieurs centaines d'ordinateurs. Il est donc très important de choisir une clé d'une taille convenable. Par sûreté, il est couramment recommandé d'utiliser des clés RSA de taille au moins égale à 2048 bits.

I.3.3 Les familles cryptographiques :

La cryptographie se scinde en deux parties nettement différenciées, d'une part la cryptographie à clé secrète, encore appelée symétrique ou bien classique, et d'autre part la cryptographie à clé publique, dite également asymétrique ou moderne.

I.3.3.1 Cryptographie symétrique (à clé secrète) :

Dans la cryptographie symétrique, aussi appelée chiffrement conventionnel ou à clé secrète, une (seule et même) clé est utilisée à la fois pour le chiffrement et le déchiffrement. Cette clé doit être gardée secrète. La sécurité d'un algorithme à clé symétrique repose donc sur la clé : si celle-ci est dévoilée, alors n'importe qui peut chiffrer ou déchiffrer des messages.

Les algorithmes à clé symétrique sont des algorithmes où la clé de chiffrement peut être calculée à partir de la clé de déchiffrement ou vice versa. Dans la plupart des cas, la clé de chiffrement et la clé de déchiffrement sont identiques. Pour de tels algorithmes, l'émetteur et le destinataire doivent se mettre d'accord sur une clé à utiliser avant d'échanger des messages chiffrés.

La Figure « Figure I.2 » est une illustration du processus du chiffrement à clé secrète.

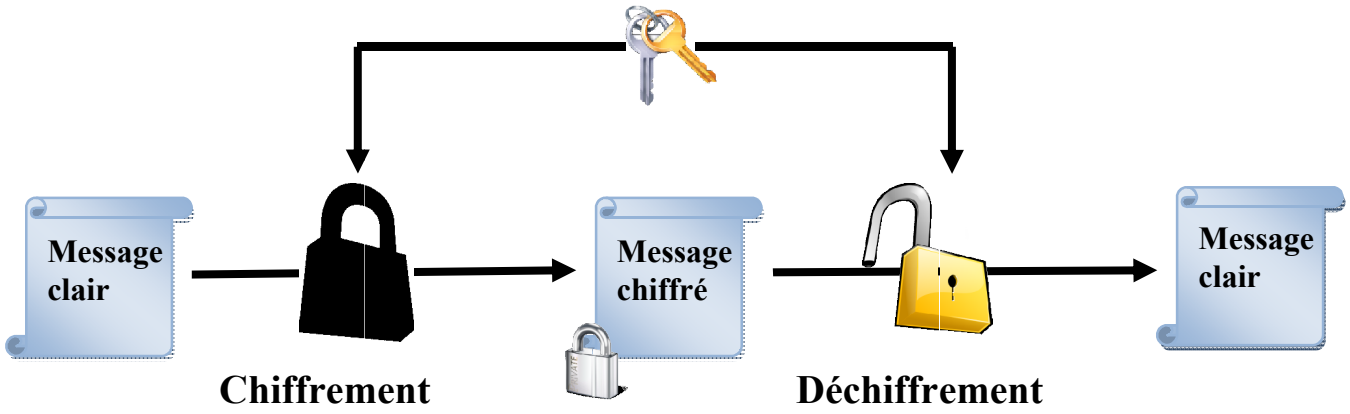

Figure I.2 : Chiffrement à clé secrète.

Il existe plusieurs algorithmes qui fonctionnent sur ce principe :

Algorithme	Description
DES (Data Encryption Standard), 1974	Conçu par IBM, ce système de chiffrement par blocs est fondé sur une clé de 56 bits. Longtemps standard de chiffrement des communications gouvernementales non classées secrètes, il a été remplacé récemment par AES. L'algorithme a été rendu public.
IDEA (International Data Encryption Algorithm), 1990	Conçu par X. Lai et J. Massey, ce système de chiffrement par blocs s'appuie sur une clé de 128 bits. L'algorithme a été rendu public.
AES (Advanced Encryption Standard), 2000	Conçu par J. Daemen et V. Rijmen, ce système de chiffrement par blocs s'appuie sur une clé de 128 à 256 bits. Il s'agit du standard de chiffrement pour les communications gouvernementales non classées secrètes. L'algorithme a été rendu public.

Tableau I.1 : Algorithme symétrique [REF05].

I.3.3.2 Cryptographie asymétrique

Dans les algorithmes à clé secrète, tout reposait sur le secret d'une clé commune qui devait être échangée dans la confidentialité la plus totale. Les problèmes de distribution de clé sont résolus par la cryptographie à clé publique, dont le concept fut inventé par Whitfield Diffie et Martin Hellman en1975.

La cryptographie à clé publique repose sur un schéma asymétrique qui utilise une paire de clés pour le chiffrement: une clé publique, qui chiffre les données et une clé privée correspondante, aussi appelée clé secrète, qui sera utilisée pour le déchiffrement.

Les algorithmes à clé asymétrique ou clé publique sont conçus de telle manière que la clé de chiffrement soit différente de la clé de déchiffrement. Il est donc mathématiquement impossible de déduire la clé privée de la clé publique.

La cryptographie à clé publique est principalement utilisée pour assurer l'intégrité des données, l'authentification, la non-répudiation et l'échange de clés. Elle peut aussi assurer la confidentialité d'un message en utilisant la clé publique du destinataire comme clé de chiffrement. En effet, seul le destinataire possède la clé privée correspondante autorisant le déchiffrement du message.
La Figure « Figure I.3 » est une illustration du processus du chiffrement à clé publique.

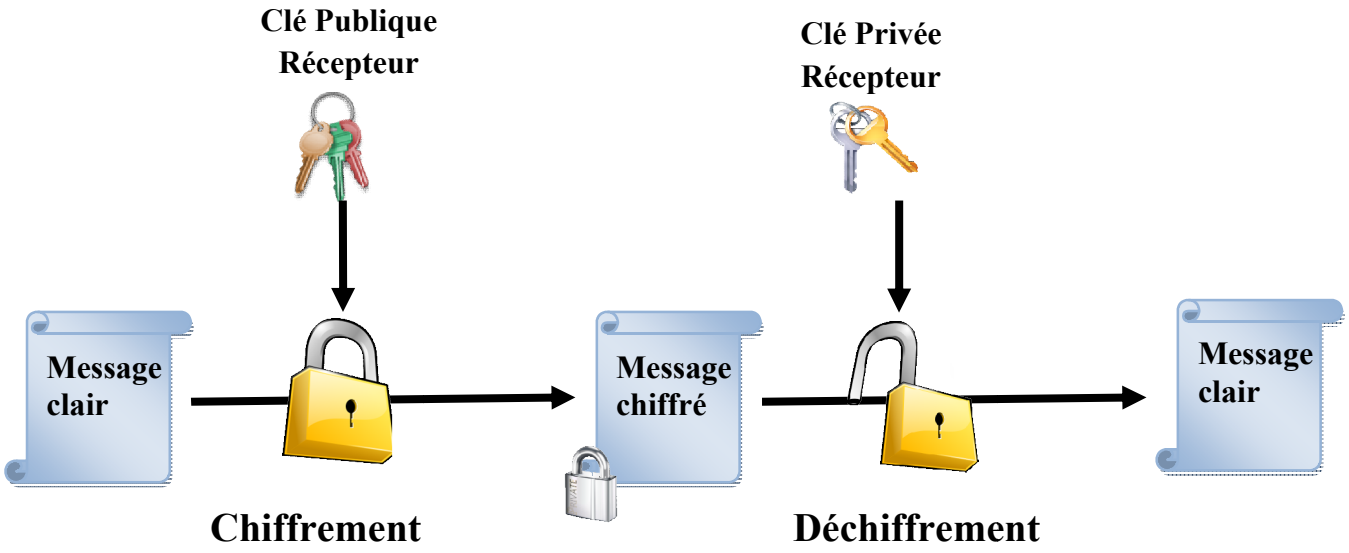

Figure I.3 : Chiffrement à clé publique.

RSA baptisé ainsi d'après le nom de ces créateurs Ron Rivest, Adi Shamir et Leonard Adleman, est l'exemple le plus populaire des algorithmes asymétriques.

Le tableau suivant énumère les principaux algorithmes asymétriques ainsi que l'usage pour lequel ceux-ci ont été conçus.

Algorithme	Description
Diffie-Hellman, 1976	Conçu par W. Diffie et M. E. Hellman, cet algorithme permet de partager un secret commun après un protocole d'échange de données. La sécurité du schéma de Diffie-Hellman repose sur la difficulté de calculer un logarithme discret. L'algorithme a été rendu public.
RSA (Rivest Shamir Adleman), 1978	Conçu par R. Rivest, A. Shamir et L. Adleman, cet algorithme permet de partager un secret commun après un protocole d'échange de données. La sécurité du schéma repose sur la difficulté de la factorisation en nombres premiers. L'algorithme a été rendu public.
Les cryptosystèmes à courbes elliptiques, 1985-2005 : – ECMQV (Elliptic Curve Menezes-Qu-Vanstone) – ECDH (Elliptic Curve Diffie-Hellman)	Introduits par V. Miller et N. Koblitz, de nombreux travaux ont déjà été menés sur ce type de système offrant de solides protections (pour des longueurs de clés plus petites que d'autres types d'algorithmes) contre la cryptanalyse. La sécurité du schéma repose sur la difficulté de calculer un logarithme discret. La société Certicom détient de nombreux brevets dans ce domaine.

Tableau I.2: Algorithme asymétrique [REF05].

I.3.4 Fonction de hachage

Une fonction de hachage est une méthode permettant de caractériser une information. Elle prend en entrée un message de taille quelconque, applique une suite de traitements reproductibles à cette entrée et obtient à la sortie une chaine de caractères hexadécimaux, une empreinte servant à identifier la donnée initiale appelée aussi le condensé. Cette sortie résume en quelque sorte l'information et a une taille fixe qui varie selon les algorithmes.

Les fonctions de hachage les plus connues sont MD4, MD5 (« MD » pour « Message Digest »), SHA (Secure Hash Algorithm) et SHS (Secure Hash Standard).Toutes ces fonctions sont quasiment similaires mais plus ou moins rapides.

- **MD4 :** L'algorithme prend en entrée un message de longueur arbitraire et produit en sortie une128-bit « empreinte digitale » ou « message digest » de l'entrée *[RFC1320]*.

- **MD5 :** L'algorithme MD5 est une extension de MD4, il est légèrement plus lent que MD4, mais il est plus «conservateur» dans la conception. L'algorithme MD5 est placé dans le domaine public pour examen et adoption éventuelle en tant que norme *[RFC1321]*.

.

- **SHA-1 :** C'est une fonction de hachage cryptographique qui fournit une empreinte de 160 bits.

- **SHA-256 :** C'est une fonction de hachage cryptographique dérivée de SHA-1 qui fournit une empreinte de 256 bits.

- **SHS :** Il fournit des empreintes de 160 bits. Sa structure est identique à MD4 et MD5, mais potentiellement plus fiable, la taille de la clé étant de 160 bits au lieu de 128 bits.

Figure I.4 : Fonction de hachage.

I.3.5 Signature numérique :

Une signature numérique est une signature électronique qui est utilisé pour authentifier l'identité de l'expéditeur d'un message ou le signataire d'un document, et sert à garantir que le contenu original du message ou document qui a été envoyé reste inchangé. Les signatures numériques sont facilement transportables, ne peuvent pas être imités par quelqu'un d'autre, et peuvent être automatiquement horodaté. La capacité de s'assurer que le message original signé est arrivé signifie que l'expéditeur ne peut pas facilement répudier plus tard. Une signature numérique peut être utilisée avec n'importe quel genre de message, soit crypté ou non, simplement pour que le récepteur soit sûr de l'identité de l'expéditeur et que le message est arrivé intact. Un certificat numérique contient la signature numérique de l'autorité de certification émettrice afin que chacun puisse vérifier que le certificat est bien réel.

Il existe deux sortes de signature numérique :

- **Signature numérique simple :** le message est signé directement par la clé privée et la vérification est faite en utilisant la clé publique.

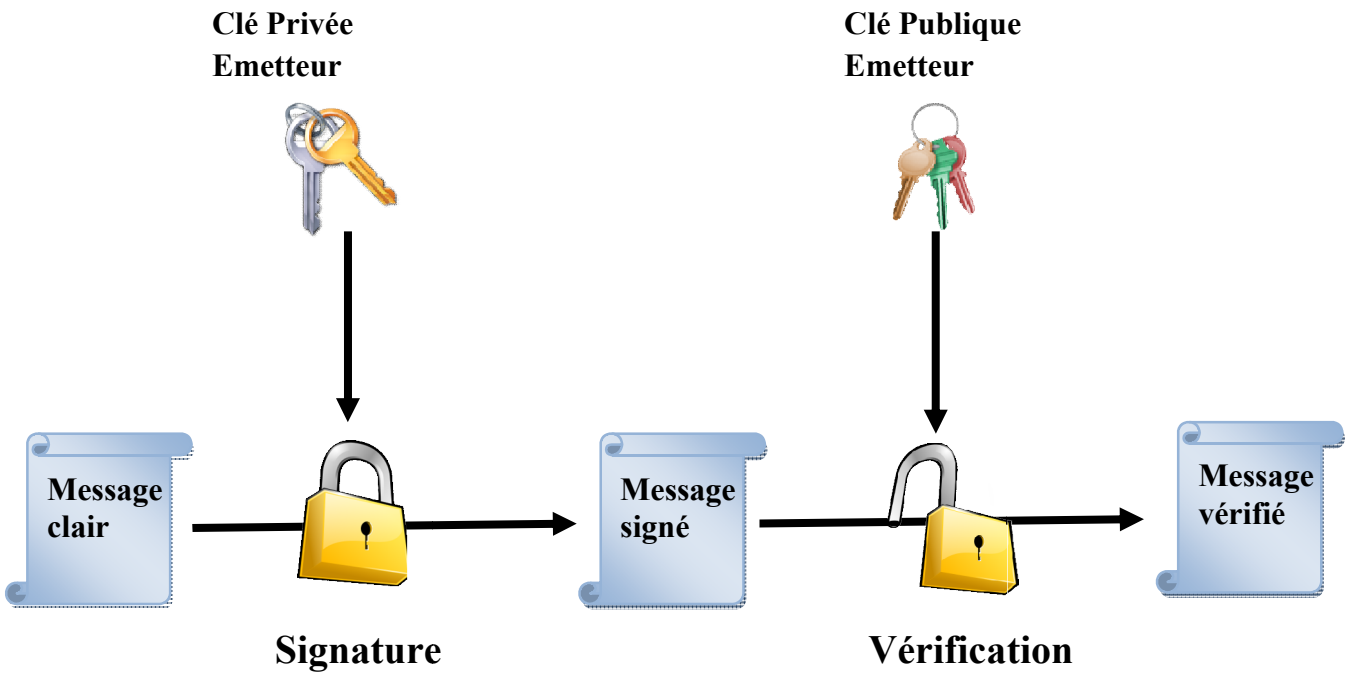

Figure I.5 : Signature numérique simple d'un message.

- **Signature numérique sécurisée :** le message à signer subit d'abord une fonction de hachage produisant ainsi un condensé. Après, le hach résultant est signé avec la clé privée.

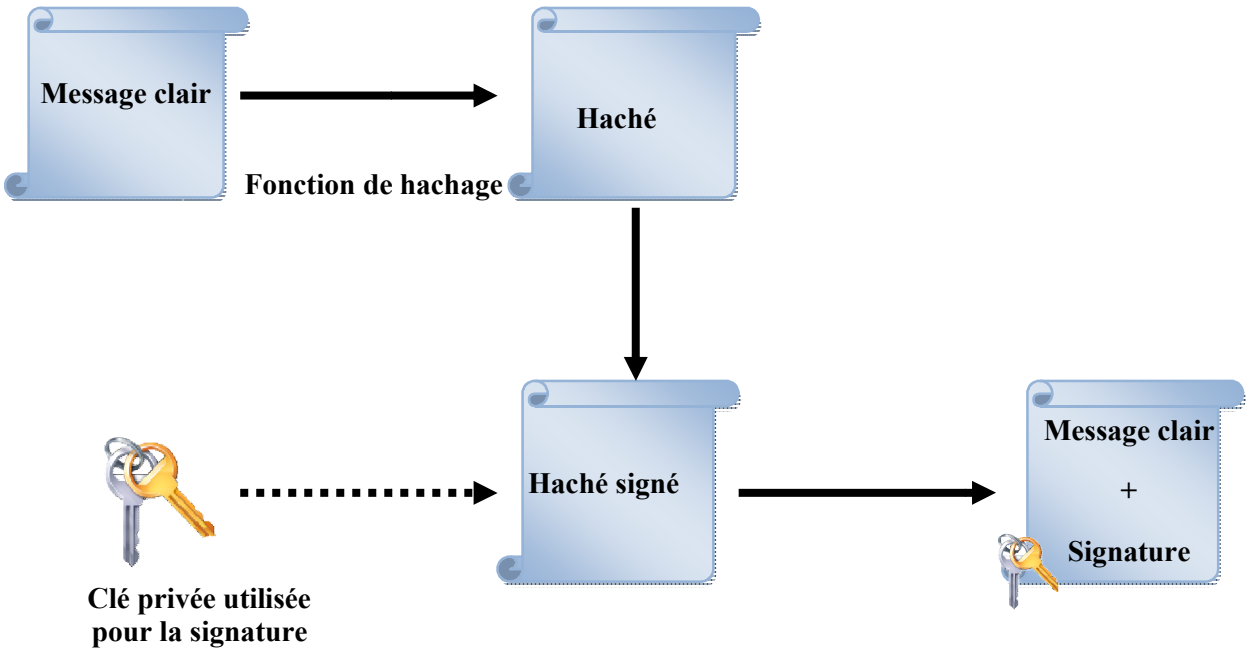

Figure I.6 : Signature numérique sécurisée d'un message.

I.3.6 Certificats numériques

« Un certificat électronique est une carte d'identité numérique dont l'objet est d'identifier une entité physique ou non-physique. Le certificat numérique ou électronique est un lien entre l'entité physique et l'entité numérique (Virtuel). L'autorité de certification fait foi de tiers de confiance et atteste du lien entre l'identité physique et l'entité numérique. Le standard le plus utilisé pour la création des certificats numérique est le X.509 » [REF07].

I.3.6.1 Contenu d'un certificat

Les utilisateurs de certificats étant de plus en plus nombreux, le format de ce certificat doit de ce fait être commun à tous les utilisateurs. Sans cela, il sera impossible d'intégrer ces certificats dans des applications logicielles développées par des différents fournisseurs. Pour cette raison, les certificats numériques sont soumis à un standard.

Bien qu'il existe plusieurs formats de certificats : X.509, PGP (Pretty Good Privacy),…le format X.509 est aujourd'hui la norme de l'industrie qui permet d'assurer l'interopérabilité entre les différents systèmes. Il est normalisé par l'ISO et réalisé par l'IETF.

C'est probablement au niveau des navigateurs Web que l'utilisation des certificats X.509 a été la plus systématique et la plus évidente.

La figure « Figure I.7 » illustre la structure d'un certificat répondant à la norme X.509.

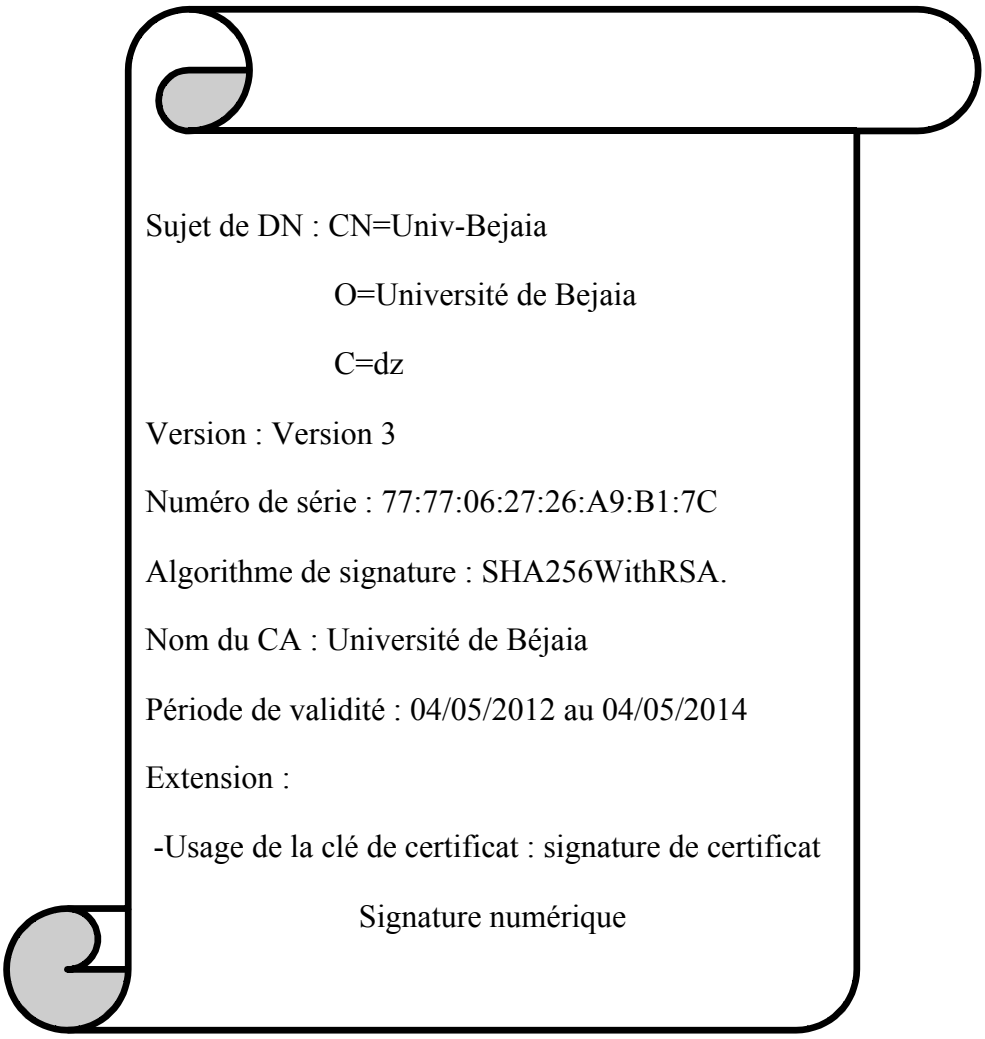

Figure I.7 : Certificat numérique X.509.

Globalement, la composition d'un certificat X.509 est la suivante :

- **Sujet du certificat (DN) :** Ce champ identifie l'identité du propriétaire du couple clés privée/publique à certifier. Il existe là aussi un formalisme pour nommer ce champ.

- **Version du protocole X.509 (v3 actuellement) :** Ce champ identifie la version de la norme X.509 qui est utilisée dans le certificat. À ce jour, trois versions de la norme X.509 ont été définis. la dernière version utilisée est la version 3.

- **Numéro de série (unique par CA) :** Le numéro de série est un numéro unique qui est utilisé pour identifier le certificat X.509.

- **Algorithme de signature de l'autorité de certification :** Ce champ identifie l'algorithme utilisé par l'autorité de certification pour signer numériquement le certificat X.509.

- **Nom du CA (DN) :** Permet d'identifier l'autorité de certification qui a délivré le certificat. Il existe un formalisme bien défini pour attribuer un nom à chaque entité sans ambigüité (la position géographique entre en compte).
- **Période de validité :** Ce champ définit la période pendant laquelle la clé publique du certificat X.509est valide.

- **Extensions (facultatif)** : Ce champ a été introduit dans la version 3 du X.509. Il permet aux autorités de certification de rajouter leurs propres informations aux certificats qu'elles délivrent. Parmi ces informations on peut citer :

    - **authorityKeyIdentifier** : Cette extension fournis un moyen d'identifier la clé publique liée à la clé privée utilisée pour la signature du certificat.
    - **subjectKeyIdentifier** : Cette extension fournis un moyen d'identifier un certificat contenant une clé publique particulière.
    - **keyUsage** : Ce champ renseigne sur l'utilisation qui doit être faite de la clé, par exemple : Digital Signature, NonRepudiation, KeyEncipherment, DataEnciphement, CRLSign, …

- **extendedKeyUsage** : Ce champ indique une ou plusieurs fonctionnalités pour lesquelles la clé publique certifiée peut être employée, en complément des champs fournis par Key Usage. Par exemple : Serveur authentification, client authentification, signature de réponses OCSP,…

- **Basic Constraints** : L'extension des contraintes de base permet de définir si le sujet du certificat est un AC et la profondeur maximale de la chaîne de certification l'incluant.

- **Signature numérique :** Contient l'identifiant de l'algorithme (fonction de hachage) utilisé par l'autorité de certification pour signer le certificat, ainsi que la valeur de la signature numérique.

I.3.6.2 Types de certificat

Le groupe de travail PKIX de l'IETF fait état de l'existence de deux classes de certificats à clés publiques *[RFC5280],* soit :

- ***Les certificats d'entités d'extrémité (end-entity) :*** Ce sont des certificats émis par une AC, pour des entités qui ne sont pas des émetteurs de certificats.
- ***Les certificats d'AC :*** Ceux sont des certificats émis par une AC, pour des entités qui sont elles-mêmes des AC capables d'émettre des certificats à clé publique. Les certificats d'AC peuvent aussi être classés en trois sous-catégories, soit :

  - ***Les certificats auto-émis (self-issuedcertificates) :*** Certificats dont l'émetteur et le sujet représentent la même entité. Une AC pourrait, par exemple, utiliser ce type de certificat durant l'opération de rotation de clés, afin d'offrir la confiance de l'ancienne clé vers la nouvelle.

  - ***Les certificats auto-signés (self-signedcertificates) :*** Cas spécial de certificats auto-émis, où la clé privée utilisée par l'AC pour signer le certificat correspond à la clé publique contenue dans ce même certificat.
    Une autorité de certification à la racine d'un chemin de certification va générer puis signer elle-même sa propre clé puisqu'il n'existe aucune AC au-dessus de celle-ci.

- ***Les certificats croisés (cross-certificates) :*** Certificat d'AC pour lequel l'émetteur et le sujet sont des entités différentes. Ce certificat permet de reconnaitre l'existence d'une autre AC dans un modèle de confiance en réseau.

I.4 Secure Sockets Layer (SSL) / Transport Layer Security (TLS)

I.4.1Définition :

« Conçu et développé par Netscape, le protocole SSL a été développé au-dessus de la couche TCP afin d'offrir aux navigateurs Internet la possibilité d'établir des sessions authentifiées et chiffrées. La première version de SSL date de 1994. La version actuelle est la v3. » [REF05].

Le développement de ce protocole a été repris par l'IETF au sein du groupe TLS (Transport Layer Security). Le protocole TLS v1.0 a été normalisé en 1999 par l'IETF dans la *RFC 2246* et présente quelques évolutions mineures par rapport à la version SSL v3. Ces protocoles ne sont pas compatibles mais la plupart des serveurs et des navigateurs web peuvent mettre en œuvre les deux protocoles.

Théoriquement, le protocole SSL permet la sécurisation de tout protocole applicatif qui s'appuie sur la pile TCP/IP, tels que HTTP, LDAP, SMTP, FTP,...
Dans la pratique, les implémentations de SSL éprouvées s'appliquent surtout à http et LDAP, donnant ainsi naissance aux protocoles biens connus HTTPS (HTTP sur SSL) et LDAPS (LDAP sur SSL).

I.4.2 Objectifs et moyens mis en œuvre

TLS et SSL peuvent servir à sécuriser les données transmises en utilisant le cryptage. TLS/SSL authentifie également les serveurs et, éventuellement, authentifie les clients afin de prouver l'identité des parties engagées dans la communication sécurisée. Il fournit également l'intégrité des données grâce à une valeur de contrôle d'intégrité. Le protocole de sécurité TLS/SSL fourni aussi la confidentialité des données en assurant que la communication entre le client et le serveur n'est pas écouté par un intrus.

I.4.3 Fonctionnement

Les protocoles SSL/TLS fonctionnent entre le protocole transport (TCP ou UDP) et le niveau applicatif pour sécuriser un protocole nativement peu sûr. Citons par exemple le protocole HTTPS (port 443) ou le protocole LDAPS (port 636).

Les protocoles SSL et TLS sont subdivisés en quatre sous protocoles :

- **Handshake :** « Handshake est le protocole d'établissement d'une connexion SSL. Il permet d'authentifier les parties client-serveur et de négocier les paramètres cryptographiques (choix de l'algorithme de chiffrement, de l'algorithme de calcul du code d'authentification MAC, des clés de chiffrement, etc.). » [REF05]

- **Record :** « Record est un protocole d'enregistrement. Pour une même connexion SSL, il offre à la fois les services de confidentialité, par le biais de l'algorithme cryptographique à clé secrète retenu, et d'intégrité des messages échangés, par le biais du calcul du code d'authentification MAC pour chaque message échangé. Des fonctions de fragmentation et de compression des données sont en outre réalisées. » [REF05]

- **Alert :** « Alert est un protocole d'alerte. Il permet d'échanger des messages prédéfini sur les états d'une connexion SSL, tels que la fermeture d'une connexion, notamment lorsqu'un certificat a été révoqué, qu'il a expiré, qu'il est vicié, etc. » [REF05]

- **CCS :** « CCS (Change Cipher Security) est un protocole de modification des spécifications de chiffrement. Il permet de modifier les paramètres de chiffrement d'une connexion SSL. » [REF05]

Ces quatre sous-protocoles s'organisent comme présenté dans la figure suivante :

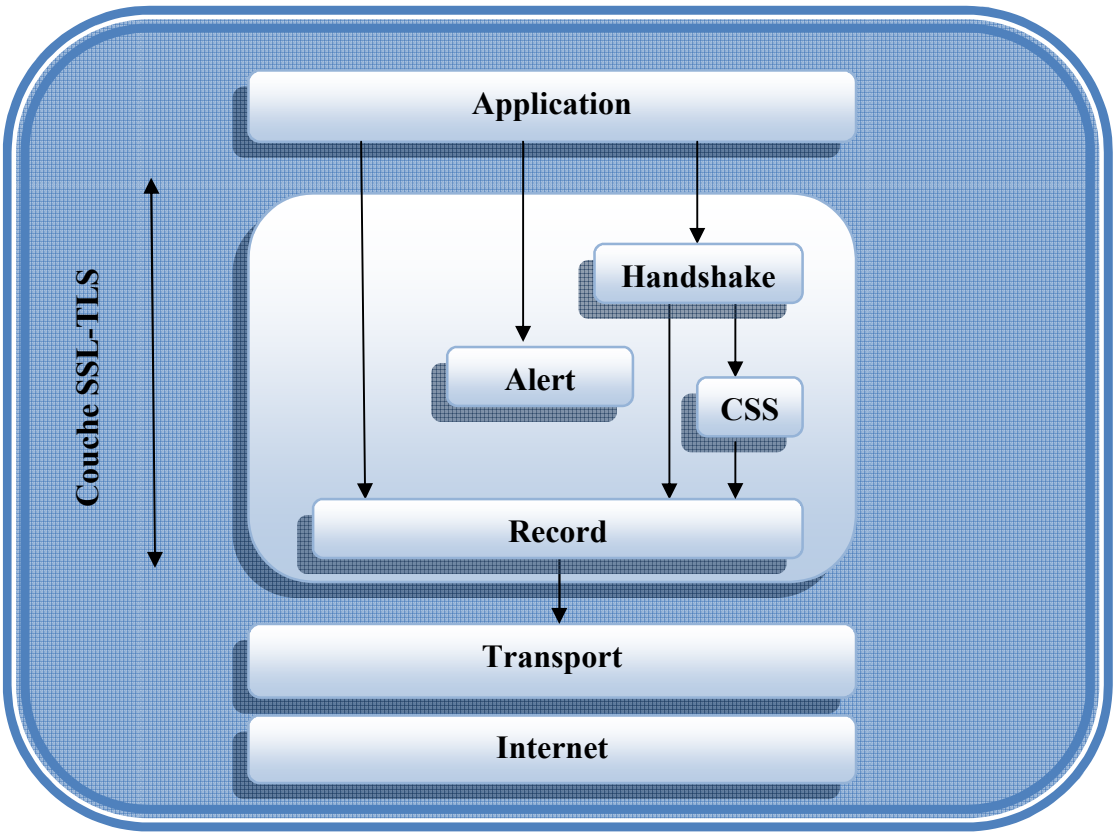

Figure I.8 : Protocole SSL/TLS [REF08].

I.4.4 La négociation SSL

La sécurisation des transactions par SSL est basée sur un échange de clés entre client et serveur.

Le mécanisme en est le suivant :

1. Le client envoie au serveur plusieurs informations pour définir notamment la version du protocole SSL utilisé, les paramètres de chiffrement utilisés, etc.

2. Le serveur envoie ensuite au client son certificat de clé publique. Il demande éventuellement au client qu'il envoie son propre certificat si une authentification du client par certificat est requise.
3. Le client utilise les informations du certificat du serveur pour authentifier le serveur.

4. Le client est alors en mesure d'envoyer au serveur une « pré » clé secrète, qu'il chiffre avec la clé publique du serveur.

5. Le serveur peut déchiffrer ensuite la « pré » clé secrète à l'aide de sa clé privée. Le serveur et le client réalisent une même série d'opérations pour obtenir des clés secrètes de session à partir de la « pré » clé secrète et des données aléatoires échangées dans les étapes précédentes.

6. Le client envoie ensuite un avertissement au serveur indiquant que les prochains messages seront chiffrés. Puis il envoie un message (chiffré cette fois) qui signifie que la phase de négociation est terminée.

7. Le serveur envoie alors lui aussi un avertissement au client qui indique que les prochains messages seront chiffrés. il envoie un message (chiffré cette fois) qui signale que la phase de négociation est terminée. La session SSL est ainsi complètement établie.

I.4.5 Implémentations de SSL et TLS

L'implémentation native de la dernière version de SSL (version 3) par les navigateurs et serveurs Web actuels du marché fait de HTTPS le standard de sécurisation des applications Web. Le protocole SSL est en effet implémenté dans le code du navigateur pour « Internet Explorer » et « Netscape Communicator » et par tous les principaux serveurs http (Iplanet, , Lotus, Apache,…). Cette version 3 de SSL prévoit entre autres une authentification optionnelle du client par certificat.

SSL/TLS est principalement utilisé pour crypter les données confidentielles envoyées sur un réseau non sécurisé comme l'Internet. Dans le protocole HTTPS, les types de données chiffrées incluent l'adresse URL, l'en-tête HTTP, et les données communiquées par le biais des formulaires. Une page Web sécurisé par SSL/TLS a une URL qui commence par «https://».

Conclusion

Le fonctionnement de la cryptographie est simple : chaque utilisateur possède une paire de clés (clé privée/clé publique) reliées mathématiquement par un algorithme très complexe. La clé publique est connue de tous afin que chaque utilisateur, lorsqu'il le désire, puisse envoyer un message à une autre personne. La clé privée est quant à elle connue uniquement par l'utilisateur qui la possède et n'est jamais communiquée. Elle lui permet de décrypter un message dont il est destinataire.

Bien que très efficace, la cryptographie à clé publique comporte cependant un enjeu majeur, soit la gestion des clés publiques. En effet, l'efficacité des mécanismes de sécurité basés sur la cryptographie à clé publique dépend du niveau de certitude que détient l'utilisateur d'une clé publique quant à l'identité de son propriétaire légitime.

Il est donc nécessaire de mettre en place un système de gestion de clés publiques qui permet de gérer des listes importantes de clés publiques et d'en assurer les quatre principes fondamentaux de la sécurité informatique.

CHAPITRE II :
INFRASTRUCTURE DE GESTION DE CLES PUBLIQUES

Introduction

L'infrastructure de gestion de clés publiques (ICP), communément appelée PKI (Public Key Infrastructure), crée un espace de confiance qui permet de gérer tous les aspects de sécurité : authentification des utilisateurs et des entités techniques, confidentialité, intégrité des données et non répudiation des transactions. Elle offre aussi la possibilité de signer du courrier électronique, de crypter des données sur les serveurs ou de mettre en place un site web sécurisé.

Dans ce chapitre, nous allons définir l'infrastructure de gestion de clés publiques ainsi que ses composants, le processus et la politique de certification, les différents protocoles d'une PKI et enfin le processus de publication.

II.1 Définition

Comme son nom l'indique, une infrastructure à clés publiques (ICP) appelée aussi PKI est un ensemble de moyens matériels, de logiciels, de composants cryptographiques, mis en œuvre par des personnes, combinés par des politiques, des pratiques et des procédures requises, qui permettent de créer, gérer, conserver, distribuer et révoquer des certificats basés sur la cryptographie asymétrique.

Cette définition empruntée à l'IETF montre qu'une PKI est bien plus qu'un système technique. Les politiques, pratiques et procédures décrivent le cadre de mise en œuvre des moyens dédiés à la PKI de façon sûre, et qui correspondent à l'attente, en termes de confiance, de ses utilisateurs.

Cette infrastructure va permettre de mettre en œuvre des services de sécurité tels que l'authentification, l'intégrité, la confidentialité et la non-répudiation des échanges électroniques.

L'infrastructure de gestion de clés publiques offre en plus les services suivants :

- service d'enregistrement des utilisateurs ;
- service de génération de certificats ;
- service de renouvellement de certificats ;
- service de révocation de certificats CRL ;
- service de publication de certificats ;
- service de publication des listes de révocation ;
- service d'identification et d'authentification.

II.2 Les composants d'une PKI :

La PKI est une association de plusieurs composants qui interviennent à différentes étapes mises en œuvre depuis la création du certificat jusqu'à l'utilisation de celui-ci. Le groupe PKIX de l'IETF propose une vue simplifiée du modèle architectural d'une PKI X.509 en la subdivisant en cinq (5) composantes distinctes *[RFC5280],* soit :

- ***L'entité d'extrémité (EE) :*** Les entités d'extrémités sont des utilisateurs, tel qu'un cliente-mail, un serveur Web ou un navigateur Web. Elles ne sont pas autorisées à délivrer des certificats à d'autres entités.

- ***L'autorité de certification (AC) :*** Composante s'acquittant de la signature des certificats numériques servant à la diffusion des clés publiques, ainsi que du maintien du statut de révocation de ceux-ci.

- ***L'autorité d'enregistrement (AE) :*** Une AE ou RA (RegistryAuthority)est une fonction administrative qui enregistre les entités de la PKI. L'AE est digne de confiance pour identifier et authentifier les entités en fonction de la politique de CA. Il peut y avoir une ou plusieurs RA connectées à chaque autorité de certification de la PKI.

- ***L'émetteur de liste de révocation de certificats (CRL) :*** Composante qui s'acquitte de la génération des listes de révocation de certificats. Cette tâche est souvent réalisée par l'AC.

- ***Le dépôt* :** Système distribué où sont stockés les certificats ainsi que les listes de révocation, afin que les utilisateurs puissent les récupérer librement. Ce dépôt prend généralement la forme d'un annuaire de type LDAP.

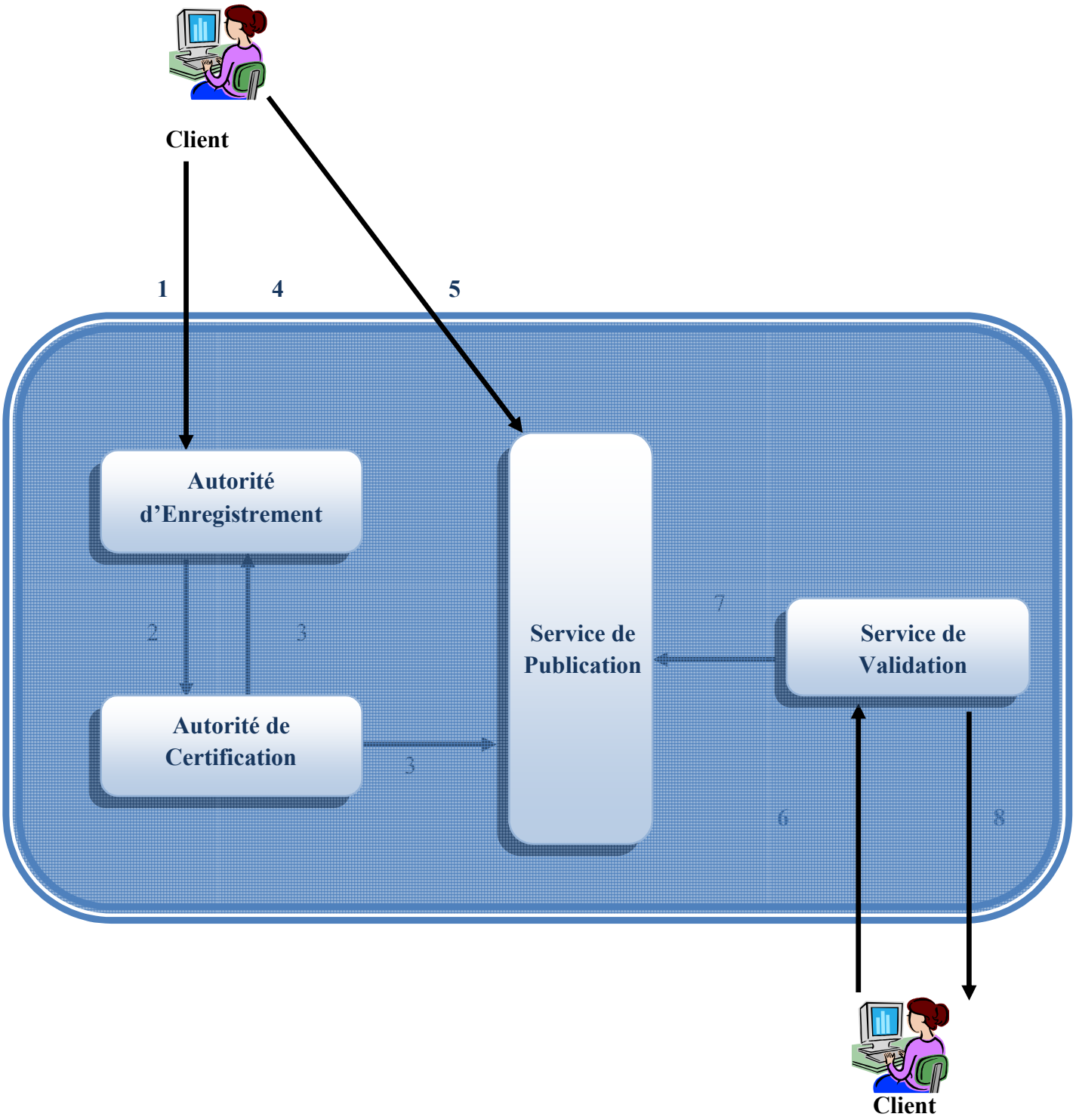

Figure II.1 : Les composants d'une PKI.

La figure « Figure II.1 » montre la place de chaque composante et le processus de gestion des certificats. Ce dernier inclut les étapes suivantes :

1. L'utilisateur fait sa demande à l'autorité d'enregistrement

2. L'autorité d'enregistrement vérifie les données d'identification, la possession de la clé privée et valide la requête qui est transférée à l'autorité de certification

3. L'autorité de certification vérifie la validité de la requête et génère le certificat. Le certificat est publié dans l'annuaire et transmis à L'autorité d'enregistrement

4. L'autorité d'enregistrement avertit l'utilisateur que son certificat est disponible

5. L'utilisateur récupère le certificat dans l'annuaire

6. L'utilisateur envoie au répondeur OCSP une requête pour vérifier l'état du certificat.

7. Le répondeur OCSP récupère la liste des certificats révoqués à partir du service de publication

8. Le répondeur OCSP envoie la réponse à l'utilisateur.

Il existe d'autres composantes non mentionnées dans les documents de l'IETF, mais qui peuvent être nécessaires dans certains cas :

- ***l'Autorité de séquestre :*** Composante qui détient les clés nécessaires pour déchiffrer les données chiffrées. Dans certaines circonstances, l'autorité de séquestre permet à un tiers autorisé d'accéder à ces clés.

- ***Serveur OCSP :*** Composante ayant pour but de vérifier la validité d'un certificat numérique en obtenant son statut de révocation. Le protocole OCSP est définit dans la norme IETF *RFC2560*. Il est un protocole de requête / réponse où un client OCSP récupère le statut de révocation de certificats à partir d'un serveur (répondeur) OSCP au lieu de le récupéré dans la liste de révocation.

- ***Serveur d'horodatage :*** Composante ayant pour mission de garantir électroniquement la date et l'heure d'une opération selon la méthode décrite dans la norme IETF *[RFC3161]*.

Dans certains cas, Il est possible que l'AC s'acquitte des rôles d'autorité d'enregistrement, d'émetteur de listes de révocation ou d'autorité de séquestre.

Cependant, L'AC est une autorité qui suit une politique de sécurité stricte, elle doit donc avoir une confiance totale en ses subordonnées qui doivent subir à leur tour une politique de sécurité plus stricte que celle de l'AC.

II.3 Répartition des AC

Dans une même infrastructure de gestion de clés publique, il peut y avoir plusieurs AC ayant le même rôle. Quelle sont donc les différentes répartitions décrivant la relation entre toutes les ACs? Il existe trois modèles de répartition : hiérarchique, croisé et en graphe.

II.3.1 Modèle hiérarchique

Dans ce modèle, l'autorité de certification racine(ACRoot) délivre un certificat d'attribut qui garantit la délégation du service de certification à une ou plusieurs d'autres autorités (ACs subordonnées), qui elles-mêmes à leurs tours délivrent des certificats à d'autres, et ainsi de suite [REF09].

La figure « Figure II.2 » est une illustration du modèle hiérarchique.

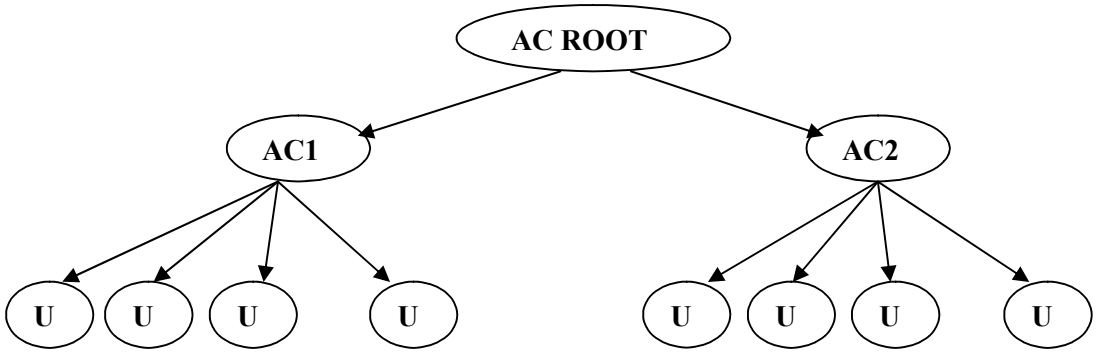

Figure II.2 : Modèle hiérarchique [REF09].

Les autorités subordonnées AC1 et AC2 ont soumis leurs clés publiques à une AC Root qui leur a généré un certificat. L'autorité AC Root peut être définie comme le plus haut niveau d'autorité ; c'est la seule composante qui ait un certificat auto-signé.

II.3.2 Modèle croisé (Peer-to-Peer) :

Dans ce modèle, les relations croisées servent à relier deux hiérarchies d'autorités de certification de deux organisations différentes. L'autorité de certification racine de chaque organisation signe pour l'autre un certificat d'identité. Ainsi, n'importe quel utilisateur de la première hiérarchie peut vérifier la clé publique de n'importe quel autre utilisateur de la deuxième [REF09].

La figure « Figure II.3 » est une illustration du modèle croisé.

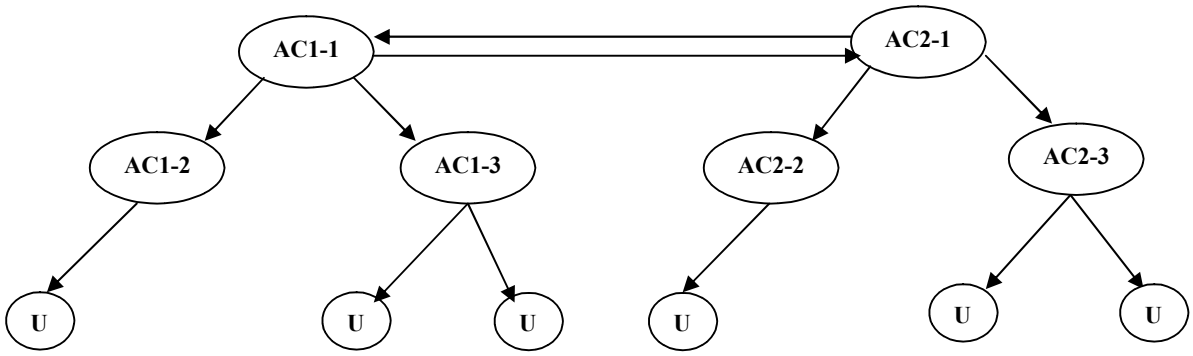

Figure II.3 : Modèle croisé (Peer-to-Peer) [REF09].

II.3.3 Modèle en graphe

Dans ce modèle, chaque autorité de certification délivre un certificat d'identité à l'autorité en qui elle fait confiance. [REF09].

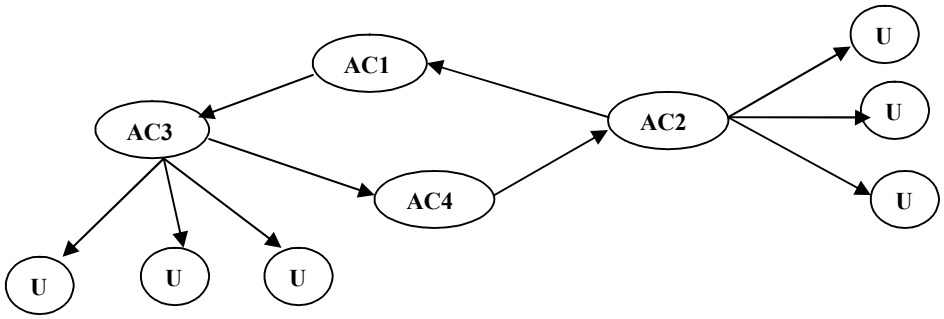

Figure II.4 : Modèle en graphe [REF09].

II.4 Cycle de vie des clés et des certificats

Le cycle de vie d'un certificat commence par la création des deux clés publique et privée. Ilse poursuit par la demande de certificat de la part du détenteur des clés, puis par la validation des justificatifs apportés. Viennent ensuite les étapes d'émission du certificat et de son acceptation. La phase suivante comporte l'utilisation des clés et du certificat, la validation du certificat, sa suspension ou sa révocation. Le cycle s'achève à l'expiration des clés et du certificat et recommence à la phase première.

II.5 La politique d'une PKI

La politique d'une PKI se définit à deux niveaux:

- **La politique de certification** : Une politique de certification est un document qui vise à indiquer quels sont les différents acteurs d'une infrastructure à clé publique(PKI), leurs rôles et leurs fonctions. Ce document est publié dans le périmètre PKI. Par exemple, une politique de certification peut préciser que la clé privée ne peut pas être exportée.

- **La politique de sécurité** : elle est utilisée pour définir l'architecture de base et les processus d'une PKI, et aussi les limites juridiques dans lesquelles opère la PKI. Cette politique fournit trois documents :

    - **Rapport pratique de certification**: qui détermine les critères de certification et la politique de révocation des certificats,
    - **Politique du certificat :** qui explique et limite l'utilisation du certificat numérique,
    - **Considérations légales** : qui sont utilisés pour responsabiliser les utilisateurs en cas de perte ou de fraude à l'intérieur même d'une PKI.

II.6 Les protocoles d'une PKI

Dans cette partie, nous allons décrire les principaux protocoles supportés par les différentes composantes d'une PKI. Ces protocoles permettent aux utilisateurs de faire appel aux différents services offerts par la PKI.

II.6.1 CRL (CertificateRevocation List)

« Lorsqu'un certificat est crée, il contient sa date de création et une date de fin de validité. Passé cette date aucune application ne l'acceptera. Mais cette date n'est pas suffisante pour invalider un certificat dans certains cas. En effet, une personne peut quitter une entreprise ou changer de service ou se faire dérober sa clé secrète. Dans ce cas il faut invalider son certificat courant. La méthode est la même que pour les cartes bancaires. Chaque autorité de certification publie régulièrement la liste des certificats révoqués, qui ne sont plus valides. Cette liste est généralement dans un annuaire LDAP, accessible par le web. Pour garantir son origine et son intégrité, elle est signée par l'autorité de certification…

Un protocole permet d'interroger ces listes en temps réel, OCSP (Online Certificate Status Protocol). Il est déjà standardisé et implémenté dans certains produits. » [REF10].

II.6.2 OCSP (Online Certificate Status Protocol)

Une limitation parfois gênante des CRL est le délai de propagation des informations de révocation. Pour le réduire, le protocole de validation de certificat OCSP (Online Certificate Status Protocol) a été développé.

L'OCSP définit par l'IETF dans la *RFC 2560* est un protocole Internet utilisé exclusivement pour valider l'état d'un certificat numérique X.509. L'information sur la révocation des certificats est disponible « on line » depuis un répondeur OCSP à travers un mécanisme de requête/réponse.

Ce protocole est une alternative réglant certains des problèmes posés par les CRLs dans une infrastructure à clés publiques. Il simplifie les traitements réalisés du côté client et il se base sur la liste de révocation pour une validation plus à jour.

- **Avantage par rapport aux CRLs**

Plusieurs raisons peuvent amener à préférer le protocole OCSP aux traditionnelles CRL :

- OCSP fournit des informations sur le statut du certificat plus à jour.
- Afin de valider un certificat, le client communique qu'avec une seule entité qui est le répondeur OCSP. Cela permet de centraliser la récupération des informations sur le statut du certificat au sein d'une PKI et d'alléger la tache du client qui n'a pas besoin de récupérer et de traiter les CRLs.

- **Composition d'une requête OCSP :**

Les communications OCSP sont de la forme « requête/réponse » et les serveurs OCSP sont appelés répondeurs OCSP.

Dans le protocole OCSP, le serveur demande l'état d'un ou de plusieurs certificats à la fois, au répondeur OCSP. Ce dernier vérifie l'état du certificat demandé et retourne une réponse à l'entité de fin.

Définie dans la *RFC 2560*, une requête OCSP se compose des éléments suivants :

- une version du protocole ;
- une demande de service ;
- certaines informations sur le certificat cible.
- des extensions optionnelles qui peuvent être traités par le répondeur OCSP.

La figure « Figure II.5 » illustre comment la requête OCSP est formée et envoyée vers le serveur OCSP.

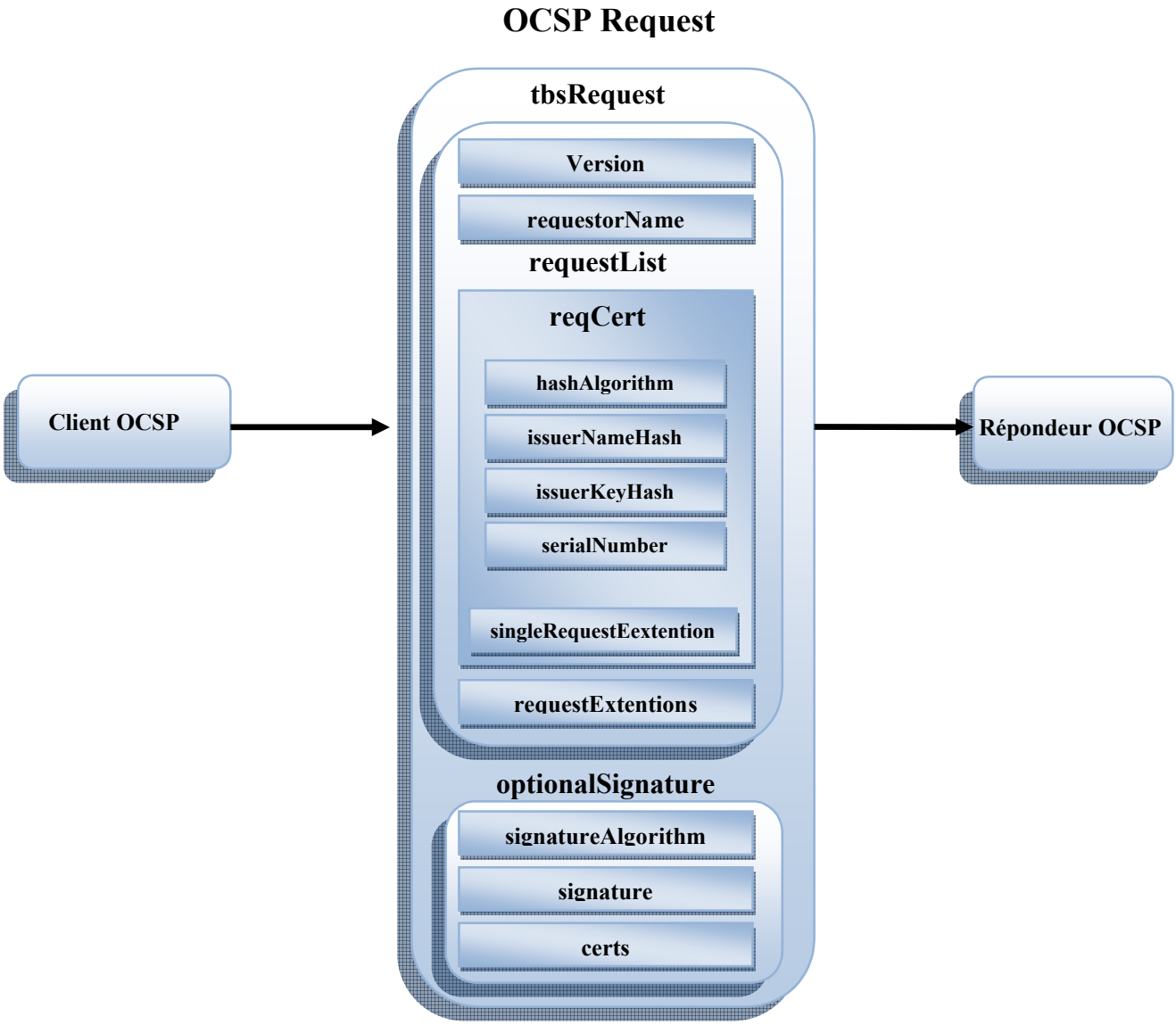

Figure II.5 : Composition d'une requête OCSP [REF11].

À la réception de cette requête, le répondeur OCSP vérifie si:

- le message est bien formé ;
- il est configuré pour fournir le service demandé ;
- la requête contient les informations requises pour être traitée.

Si l'une des trois conditions précédentes n'est pas respectée, le répondeur OCSP produit un message d'erreur, sinon il renvoie une réponse définitive.

Définies dans la *RFC 2560*, les réponses OCSP peuvent être de divers types :

- Bon (good) : cela veut dire que la réponse à la requête émit par le serveur est positive et que le certificat est valide (n'est pas révoqué).

  Révoqué (revoked) : cela veut dire que le certificat sollicité n'est plus valide, c'est-à-dire qu'il est révoqué temporairement ou définitivement.

- Inconnu (unknown) : cela veut dire que le répondeur OCSP n'a pas trouvé d'information sur le certificat demandé.

La figure « Figure II.6 » illustre comment la réponse OCSP est formée et envoyée vers le client OCSP:

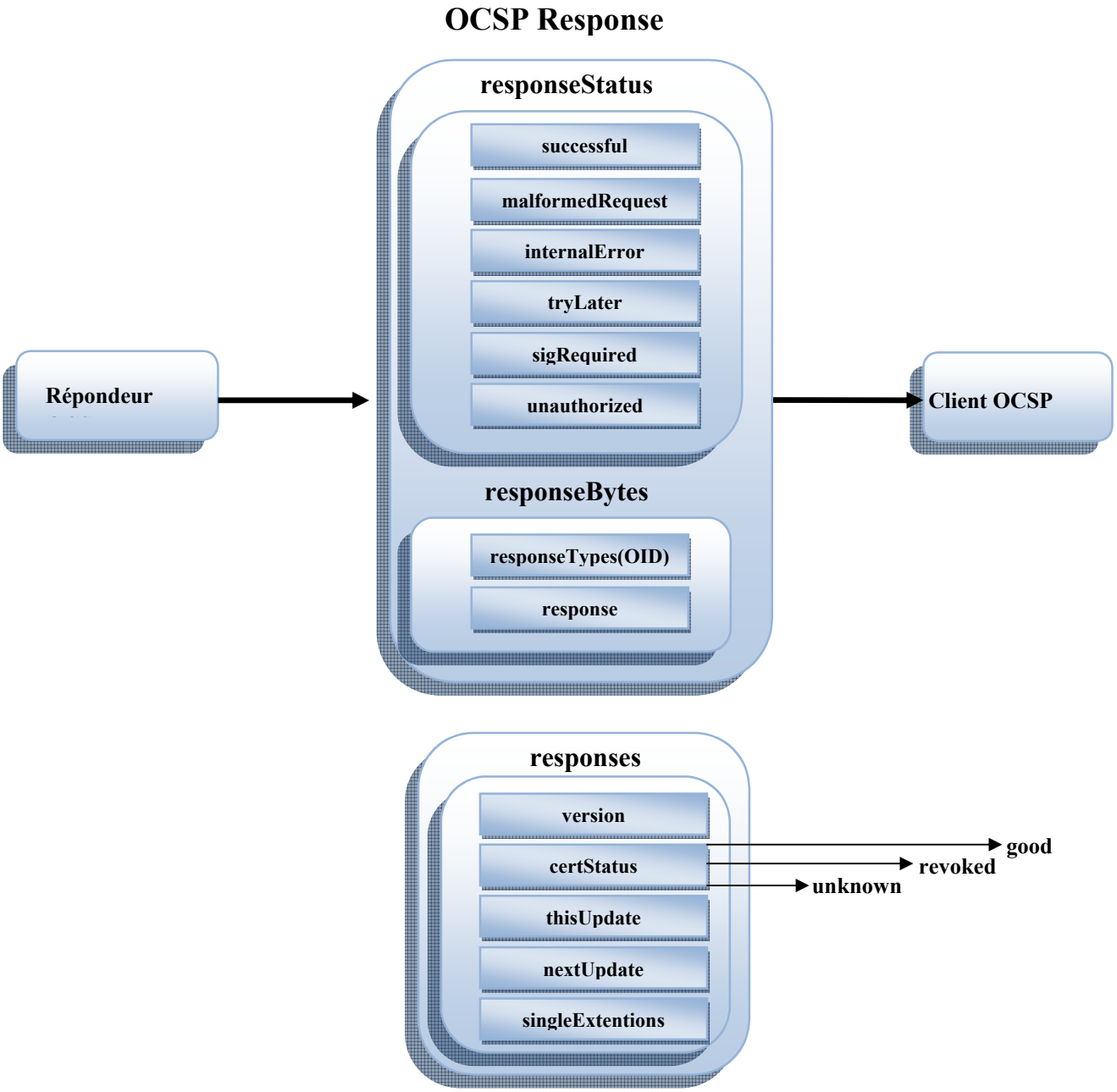

Figure II.6: Composition d'une réponse OCSP[REF11].

- **Délégation d'Autorité de Signature OCSP**

La clé qui signe l'information d'état d'un certificat et la clé qui a signé le certificat ne doit pas être la même. L'émetteur du certificat délègue explicitement son autorité de signature au répondeur OCSP. L'émetteur publie un certificat contenant une valeur unique «extendKeyUsage» dans la signature des certificats OCSP. Ce certificat doit être délivré directement au répondeur par l'AC informée *[RFC2560]*.

II.6.3 CMP *(Certificate Management Protocol)*

Le certificat de gestion de protocole (CMP) est un protocole Internet utilisé pour l'obtention des certificats numériques X.509 dans une infrastructure de gestion de clés.

Il est décrit dans la *RFC 4210* en développant un protocole compréhensible supportant une grande variété de modèles.

Le protocole CMP permet aux utilisateurs d'une PKI de réaliser à distance certaines opérations de gestion des certificats à clé publique :

- s'enregistrer auprès de la PKI et recevoir son certificat;
- recouvrer ses clés;
- effectuer la mise à jour de ses clés;
- renouveler son certificat;
- effectuer une demande de révocation.

II.7Annuaire

II.7.1 Définition

Les certificats émis par les autorités de certification doivent être rendus accessibles afin que les utilisateurs du système puissent y accéder. Pour cela, ils sont publiés à travers un annuaire à accès libre. Il constitue en quelque sorte une base de données centrale accessible en mode de lecture, en mode d'ajout (nouveau certificat) et en mode de suppression (révocation d'un certificat). Il contient également une liste de révocation de certificats(CRL) qui comporte

la liste de l'ensemble des certificats révoqués depuis la création de l'annuaire, datés et signés par les autorités de certification.

II.7.2 Annuaire et PKI

L'annuaire est généralement cité comme un élément essentiel dans la PKI d'un système. Comme les certificats numériques sont largement distribués, il sera nécessaire de disposer d'un système d'annuaire permettant la publication des certificats révoqués et par la même occasion les certificats en cours de validité.

Il offre aussi aux utilisateurs la possibilité de récupérer les clés publiques destinées à décrypter les documents et permet de stocker les clés privées des utilisateurs pour les utiliser en cas de recouvrement de clé.

L'annuaire doit accepter le protocole X.509 pour le stockage des certificats révoqués (CRLs) et le protocole LDAP (Lightweight Directory Access Protocol) qui est le standard pour l'accès aux données par annuaire.

II.7.3 Protocole d'accès au répertoire

Pour comprendre mieux le service d'annuaire, il est nécessaire de bien connaître les bases sur les protocoles d'accès à l'annuaire que sont X .500 et LDAP.

II.7.3.1 X.500

X.500 est certainement le plus ancien modèle d'accès à l'annuaire. Il définit l'ensemble des normes informatiques sur les services d'annuaire définies par l'UIT-T. Mais seule la partie X.509 concernant l'authentification est utilisée actuellement ; pour le reste, la plupart des services d'annuaire actuels utilisent une norme beaucoup moins lourde : LDAP.

X.500 n'est pas devenu un standard pour les applications Internet pour plusieurs raisons :

1. Il est jugé très lourd et complexe
2. Il est basé sur le modèle OSI, qui est difficilement supporté par les applications Internet.

II.7.3.2 LDAP (Lightweight Directory Access Protocol)

LDAP est un protocole permettant l'interrogation et la modification des services d'annuaire. Ce protocole repose sur TCP/IP ; il est très nettement moins lourd en ressources que son prédécesseur X.500. Cette raison a permis d'amener très rapidement ce protocole au niveau d'un standard d'Internet. La version LDAPv3 est définie dans la *RFC2251*.

Un annuaire LDAP respecte généralement le modèle X500 : c'est une structure arborescente dont chacun des nœuds est constitué d'attributs associés à leurs valeurs. Plus précisément, la structure d'un annuaire est comme suit :

- Un annuaire est un arbre d'entrée.
- Une entrée est constituée d'un ensemble d'attributs.
- Un attribut possède un nom, un type et une ou plusieurs valeurs.
- Les attributs sont définis dans des schémas.

Le fait que les attributs puissent être multivalués est une différence majeure entre les annuaires LDAP et les SGBDR. De plus, si un attribut n'a pas de valeur, il est purement et simplement absent de l'entrée.

Chaque entrée a un identifiant unique, le Distinguished Name (DN). Il est constitué à partir de son Relative Distinguished Name (RDN) suivi du DN de son parent. C'est une définition récursive. On peut faire l'analogie avec une autre structure arborescente, les systèmes de fichiers ; le DN étant le chemin absolu et le RDN le chemin relatif à un répertoire. En règle générale le RDN d'une entrée représentant une personne est l'attribut uid.

Exemple :

dc=dz
|
dc=UnivBejaia
/ \
ou=personnes ou=services
|
uid=admin

La communication LDAP s'appuie sur le protocole TCP, et de manière optionnelle sur TLS *[RFC2246]*.

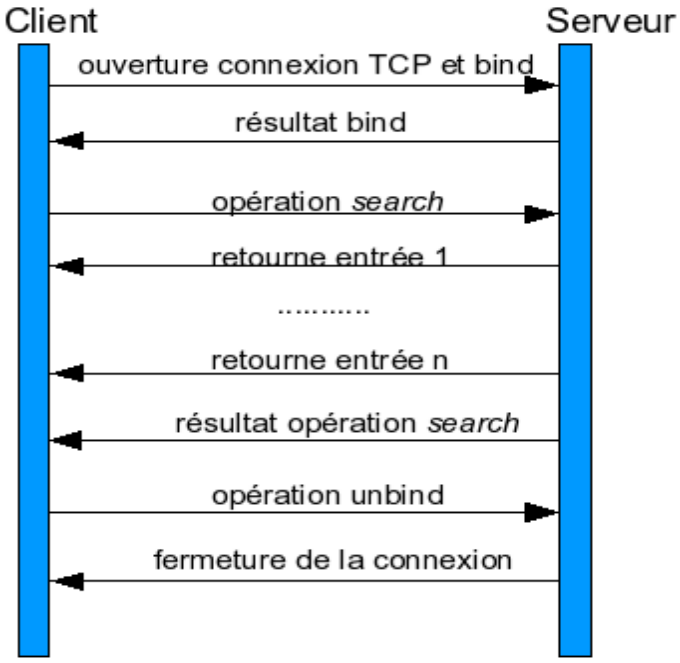

Figure II.7 : Fonctionnement du protocole LDAP [REF12].

Cette figure donne un exemple simple d'un échange entre un client et un serveur LDAP. Le client ouvre une connexion TCP, se connecte au serveur, émet une requête de recherche, et récupère les entrées correspondantes à sa recherche, puis se déconnecte du serveur et ferme la connexion TCP.

Les opérations de base effectuées sur un annuaire LDAP sont illustrées dans le tableau suivant :

Opération LDAP	Description
Search (recherche)	Recherche dans l'annuaire des objets répondant à certains critères. Cette opération est évidemment la plus courante.
Compare (comparaison)	Cette opération consiste à simplement comparer une valeur d'attribut d'une entrée avec une valeur fournie par le client.
Add (ajout)	Cette opération ajoute une ou plusieurs entrées. Les entrées sont reliées à des entrées existantes de l'arbre de données.
Modify (modification)	Pour modifier le contenu d'une entrée, c'est à dire modifier les valeurs de ses attributs. Éventuellement rajouter des valeurs, en effacer ou supprimer, quelques valeurs ou toutes.
Delete (destruction)	Pour effacer une entrée existante.
modifyDN (renommage)	Cette opération permet de modifier la valeur d'un attribut qui sert de RDN.
Bind (association)	Cette opération permet de se connecter a l'annuaire.
Unbind	Déconnexion du serveur. La connexion TCP est ensuite terminée.
Abandon	Abandon d'une opération en cours. La connexion TCP n'est pas terminée.

Tableau II.1 : Les opérations de base sur LDAP.

Conclusion :

Nous avons vu qu'une infrastructure de gestion de clés publiques se construit, c'est donc une structure à la fois technique et administrative. Le domaine des PKI est intéressant : il est possible de les utiliser pour diverses applications telles que le mail chiffré, le web sécurisé, le commerce électronique ...

Comme les PKI intègrent la cryptographie à clé publique et certificat numérique, elles peuvent se confier à des tierces parties de confiance et échanger des données en toute sécurité.

Il existe plusieurs solutions ou produits qui implémentent une PKI et qui offrent la possibilité de sécuriser les données. Le chapitre suivant sera consacré à la mise en place de l'un de ces produits.

Introduction :

L'université de Bejaia, comme toute université, produit, échange, gère de l'information et des documents à travers différentes applications : messagerie électronique, serveur web,….

Dans cette conjoncture, il nous a été proposé dans le cadre de notre projet d'obtention du diplôme Master professionnel, de réaliser une infrastructure de gestion de clés publiques qui s'inscrit dans le cadre de la mise en place d'une configuration sous Linux qui s'occupe de la création des certificats, la validation et la publication de ces derniers. Cette plate-forme s'adresse au centre de calcul de notre université.

III.1 Présentation du projet

Notre projet consiste en la mise en œuvre d'une infrastructure de gestion de clés publique (PKI) pour le réseau du centre de calcul de l'université **Abderrahmane Mira** de **Béjaia.**

Il existe plusieurs solutions ou produit facile à déployer pour réaliser une PKI, parmi ces solutions les deux autorités de certification EJBCA et OpenCA font parties des applications web qui répondent le plus a des attentes de PKI.

OpenCA est un logiciel développé par The OpenCA Labs qui est une organisation dont les résultats sont open source, travaillant sur l'étude des PKIs et le développement de projets qui s'y relatent [REF16].

Entreprise Java Bean Certificate Authority (EJBCA) est une PKI sous licence OpenSource (LGPL) développée en Java/J2EE pouvant gérer plusieurs autorités de certification (AC) [REF13].

Le tableau suivant décrit une comparaison entre les deux applications web, EJBCA et OpenCA.

Paramètres	EJBCA	OpenCA
Facilité de configuration	Très complexe	Complexe
Choix de l'algorithme a utilisé	Oui	Oui
OCSP	Oui	Non
Mise à jour de la CRL	Automatique	Manuel
Cout	Gratuit	Gratuit
Plateforme	Java J2EE	Perl CGI
Support LDAP	Oui	Oui
Module	EJB	Modules Perl
Evolutivité	Bien conçu et évolutif	Evolutivité difficile
Composante Autonome	Elle est présente. La PKI peut être administré complètement à travers la ligne de commande	Elle n'est pas présente. La seule façon d'administrer la PKI est à travers des interfaces Web.

Tableau III.1 : Comparaison entre EJBCA et OpenCA [REF16]

D'après ce tableau, nous constatons que les deux outils sont puissants. Nous avons choisi d'utiliser le produit en logiciel libre **EJBCA** pour les raisons suivantes :

- Entièrement écrit en Java ;
- Implémente OCSP.
- Mise à jour automatique de la CRL.
- Evolutif et permet une administration par ligne de commande.

III.1.1 Présentation du produit EJBCA

III.1.1.1 Définition :

EJBCA est une PKI avec un haut niveau de paramétrage comprenant les composants de bases des PKI disponibles sur le marché, elle fournit la gestion et la délégation des droits d'administration. EJBCA fournit également un serveur OCSP, un serveur d'horodatage et un serveur de signature. Chaque composant peut être déployé de manière autonome ou intégré dans EJBCA et exécutent au sein d'un "conteneur d'EJB," sous la supervision d'un serveur d'application (JBoss).

EJBCA a été construit comme un générateur, permettant de créer des profils de certificat, des formats de requête, des cinématiques de gestion du cycle de vie des certifications, de personnaliser des cartes à puce ou de dongle USB.

III.1.1.2 Architecture d'EJBCA:

EJBCA propose quatre niveaux fonctionnels. Cette figure est une illustration de l'architecture fonctionnelle d'EJBCA.

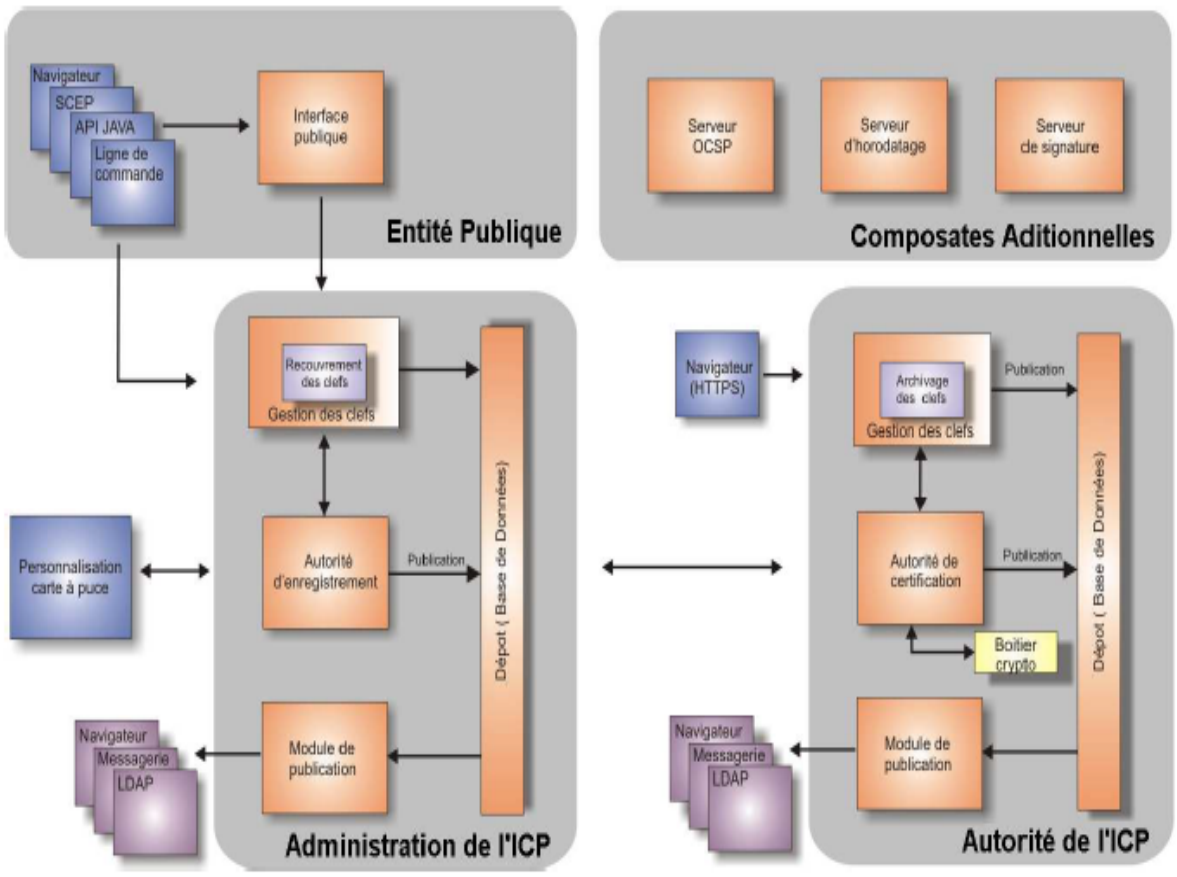

Figure III.1 : Architecture fonctionnelle d'EJBCA [REF14].

D'après cette figure, EJBCA offre quatre composantes qui permettent de réaliser l'ensemble des opérations du cycle de vie des certificats :

- **Autorité de certification (AC) :** Composante à la base de la PKI EJBCA. Celle-ci s'acquitte de l'émission et de la gestion des certificats numériques.

- **Autorité d'enregistrement (AE) :** Composante qui s'acquitte du processus d'inscription des entités d'extrémité. Celle-ci est responsable des fonctions qui lui sont déléguées par l'AC en vertu de la politique de certification.

- **Autorité d'enregistrement locale (AEL) :** Composante qui permet aux personnes, aux machines ou aux agents logiciels d'effectuer des demandes de certificats.

- **Dépôts :** Composantes qui stockent les éléments publics de la PKI tels que les certificats et les listes de certificats qui ont été révoqués (CRL), de manière à les rendre disponibles aux utilisateurs.

III.1.1.3 Caractéristiques techniques :

Les caractéristiques techniques du produit EJBCA sont illustrées dans le tableau suivant :

Caractéristique technique	Description
Éditeur	PrimeKey Solutions
Solution	EJBCA
Version actuelle	4.0.10
Référence	http ://www.ejbca.org/
Plateformes supportées	Plateformes Java™ J2EE 1.3 (EJB 2.0). Serveurs d'applications : · JBoss; · Weblogic; · Glassfish; · OC4J; · Websphere.
Système d'exploitation supporté	Unix, Linux, Microsoft
Fonctionalité	•Multiple AC et niveau d'AC par instance ; •Génération de certificats de manière individuelle ou pour lot ; •Génération de certificats en mode centralisé et décentralisé ; •Administration par ligne de commande ou interface Web ; •Gestion des profils de certificats ; •Gestion des profils d'entités ;

Protocoles et standards	•Certificat x509v3 et CRL v2 (RFC3280) ; •OCSP (protocole de validation - RFC2560) •CMP (protocole de gestion – RFC4210) ; •Horodatage (RFC3261) ; •LDAP v3.
Base de données	Hypersonic, Mysql, PostGresql, Oracle, MS-SQL, Sysbase, informix.
Entité	•Autorité de certification •Autorité d'enregistrement •Gestionnaire de clé (Séquestre et renouvellement) •Autorité d'enregistrement publique •Serveur OCSP interne ou externe •Gestionnaire de carte à puce et dongle USB •Service de journalisation • Service de publication LDAP et courriel
Support cryptographique	RSA (jusqu'à 4096-bits) Elliptic Curve DSA signing MD5, SHA-1, SHA-256
Caractéristiques de certificats	Certificats X.509 v3 Longueur de clés allant jusqu'à 4096-bit pour authentification. Configuration flexible des DN de certificats.
Annuaire LDAP	Sun directory, OpenLDAP, Active directory, LDAPv3

Tableau III.2 : Les caractéristiques technique du produit EJBCA [REF14].

III.2 Mise en œuvre de l'infrastructure de gestion de clés publique :

Le Produit EJBCA est offert sous licence LGPL, celui-ci offre donc toutes les libertés fondamentales associées au logiciel libre. Son utilisateur est donc libre de l'exécuter, de

l'adapter, de l'améliorer et de le redistribuer sans aucune redevance. Il s'agit d'un avantage marqué de cette solution sur les solutions propriétaires.

Nous allons maintenant voir comment mettre en œuvre une PKI à l'aide du produit EJBCA.

III.2.1 Description de l'environnement

Pour mettre en œuvre notre PKI, nous avons eu le besoin de mettre en place trois machines : une machine d'administration EJBCA sous Linux, un serveur OCSP sous Linux et un client sous Windows.

Dans chaque machine il fût installé tous les logiciels requis, la version de chaque logiciel est précisée dans le Tableau III.3**.**

La figure « Figure III.2 » résume l'environnement de travail pour la mise en place d'EJBCA.

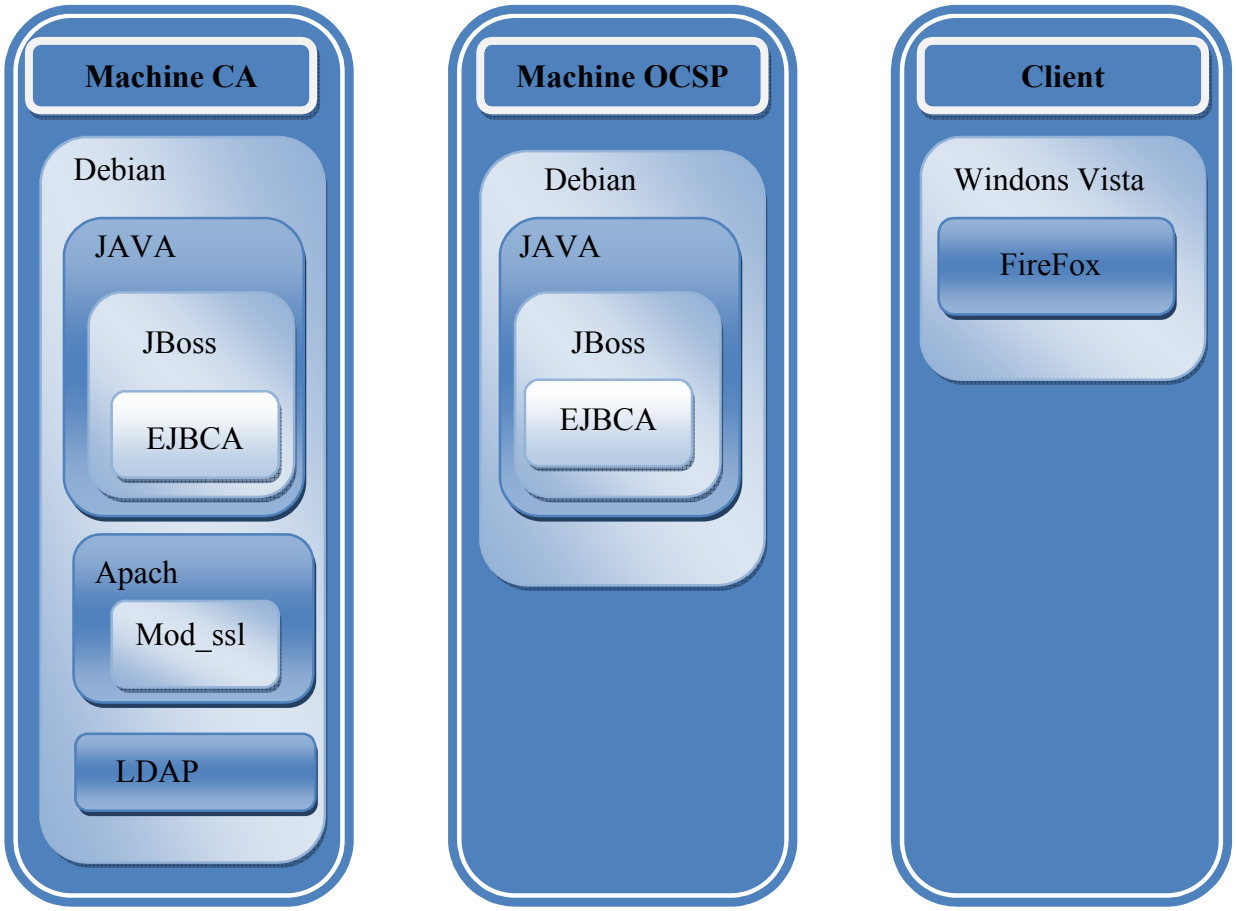

Figure III.2 : Environnement de travail.

III.2.2 Les logiciels requis :

L'installation et l'utilisation d'EJBCA nécessitent le chargement de plusieurs composants. Voici un tableau des composants à charger, ainsi que les versions utilisés.

Modules	Version
Linux	Debian 6
EJBCA	4.0.10
Serveur JBOSS	5.1.10
Installeur ANT	1.7.0
OpenJDK	6
JCE Policy	1.5.0
MySQL	5.0
JDBC	5.0.4

Tableau III.3 : les logiciels requis pour la mise en place d'une PKI avec EJBCA.

- **Linux :**

Linux ou GNU/Linux, est un système d'exploitation libre fonctionnant avec le noyau Linux. Il est idéal pour l'implémentation d'une PKI. De plus, toutes les composantes logicielles requises pour mettre en œuvre la solution proposée sont compatibles avec ce système d'exploitation.

Il existe plusieurs distributions sous Linux (Debian, Red Hat Entreprise Linux, Slackware, SuSe). Nous avons choisie Debian car il se distingue de la plupart des autres distributions citées auparavant par son caractère non commercial et par le mode de gouvernance coopératif de l'association qui gère la distribution. De plus, la plupart des logiciels requis sont intégré dans cette version de linux, ce qui facilite le processus d'installation d'EJBCA.

- **EJBCA :**

EJBCA est une PKI réalisée en JAVA décrit en détail dans la section III.1.1. Tous les composants utilisés fonctionnent en JAVA ou existent sur plusieurs plateformes. Le fonctionnement d'EJBCA est identique sur toutes les plateformes.

- **JBOSS :**

JBoss Application Server est un serveur d'applications J2EE libre entièrement écrit en Java, publié sous licence GNU LGPL. Parce que le logiciel est écrit en Java, JBoss Application Server peut être utilisé sur tout système d'exploitation fournissant une machine virtuelle java (JVM).

Pour s'exécuter, les composantes logicielles de la PKI EJBCA doivent être déployées dans un serveur J2EE. JBoss est un des nombreux serveurs d'applications supportés par le produit EJBCA.

- **Apache ANT :**

Another Neat Tool(Ant) est un projet Open Source de la fondation Apache écrit en Java qui sert essentiellement à compiler et déployer les projets Java.

- **OpenJDK :**

Le JDK (Java Development Kit) permet de développer et d'exécuter des applications écrites en langage Java. Les composantes de la PKI EJBCA, de même que le serveur d'application JBoss ont besoin de l'interpréteur Java pour s'exécuter.

- **JCE Policy :**

Java Cryptography Extension(JCE) est une extension de la plateforme Java qui implémente le cryptage, la génération de clés, …, et permet d'utiliser des clés de grandes tailles.

- **MySQL :**

My Structured Query Language (MySQL) est un serveur de bases de données relationnelles Open Source. Il stocke les données manipulées par des applications (dans

notre cas EJBCA) dans des tables séparées plutôt que de tout rassembler dans une seule table.

- **JDBC :**

Java DataBase Connectivity (JDBC) est une API fournie avec Java permettant de se connecter à des bases de données, dans notes cas on utilise JDBC pour que le serveur JBoss puisse se connecter a la base de donnée MySQL.

III.2.3 Installation

Dans cette partie nous allons citer les étapes essentielles pour l'installation de l'autorité de certification d'EJBCA ainsi que du répondeur OCSP.

Première étape : sur la machine de l'autorité de certification d'EJBCA

1-Installation de la base de données MySql :

Dans ce point nous allons installer l'outil MySql et créer ensuite une base de données ainsi qu'un utilisateur.

- L'installation se fait avec la commande suivante en tant que root

```
#apt-get install mysql-server
```

Un mot de passe de l'utilisateur root est demandé lors de l'installation

- La création de la base de données se fait avec l'exécution des commandes suivantes:

```
#mysql –uroot –p
password :mot de passe root
```

```
mysql>create database ejbca ;
```

- Nous allons maintenant passer à la création d'un utilisateur 'ejbca' identifié par un mot de passe 'ejbca' et lui donnée les privilèges sur la base de donnée 'ejbca' créée précédemment.

```
mysql>create user 'ejbca'@'localhost' identified by
'ejbca';
```

```
mysql>grant create,insert,select,update,delete on ejbca.*
to 'ejbca'@'localhost';
```

2-Configuration des variables d'environnement:

Les variables d'environnement suivantes doivent être renseignées à partir du terminal :

```
#export JBOSS_HOME=/opt/jboss
#export PATH=$PATH:$JBOSS_HOME/bin
#export APPSRV_HOME=$JBOSS_HOME
#export ANT_OPTS=-Xmx512m
#export JAVA_HOME=/usr/lib/jvm/java-6-openjdk/
#export JAVA_OPTS="-server -Xms128m -Xmx512m"
#export JAVAC_OPTS="-Dno-xdoc"
```

3-Installation d'Apache Ant:

L'installation d'Apache Ant s'effectue en exécutant la commande suivante:

```
#apt-get install ant
```

4- Installation de JDK et JCE :

L'installation de JDK et JCE s'effectue en exécutant les commandes suivantes:

```
#apt-get install Openjdk-6-jdk
```

Apres le téléchargement de l'outil JCE dans le répertoire /usr on effectue les opérations suivantes :

```
# cd /usr
# cp jce_policy-6.zip /opt
# cd /opt
# unzip jce_policy-6.zip
# cd jce
# cp *.jar $JAVA_HOME/jre/lib/security/
# cd /opt
# rm -rf jce jce_policy-6.zip
```

5- Installation du serveur JBoss :

Dans ce point, nous avons besoin d'installer le serveur JBoss et le connecteur JDBC pour connecter EJBCA à la base de données Mysql.

- L'installation du serveur JBoss s'effectue en exécutant les commandes suivantes:

```
# cd /usr
# cp jboss-5.1.10.zip /opt
# cd /opt
# unzip jboss-5.1.10.zip
# cd jboss
```

- L'installation du connecteur JDBC

```
# cd /usr
# cp mysql-connector-java-5.0.8.zip /opt
# cd /opt
# unzip mysql-connector-java-5.0.8.zip
# rm -rf mysql-connector-java-5.0.8.zip
# cp mysql-connector-java-5.0.8/mysql-connector-java-
5.0.8-bin.jar /opt/jboss/server/default/lib/
# rm -rf mysql-connector-java-5.0.8/
```

6-Installation et Configuration d'EJBCA:

- L'installation d'EJBCA s'effectue comme suite:

```
# cd /usr
# cp ejbca_4_0_10.zip /opt
# cd /opt
# unzip ejbca_4_0_10.zip
# rm -rf ejbca_4_0_10.zip
# ln -s ejbca_4_0_10/ ejbca
```

- Nous allons maintenant passer à la configuration des différents fichiers dont nous avons besoin pour la compilation et le déploiement d'EJBCA. Ces fichiers qui se trouvent dans le répertoire /opt/ejbca/conf doivent avoir une extension « .properties ». La configuration de ces fichiers consiste en la modification de quelques valeurs et ces modifications seront prises en compte par la suppression du caractère « # » qui sert de commentaire.

    - **Le fichier « database.properties » :**

```
database.name=mysql
database.url=jdbc:mysql://127.0.0.1/ejbca
database.driver=com.mysql.jdbc.Driver
database.username=ejbca
database.password=ejbca
```

- **Le fichier « ejbca.properties » :**

```
appserver.home=/opt/jboss
appserver.type=jboss
ejbca.productionmode=ca
jboss.config=ejbca
```

- **Le fichier « web.properties » :**

```
java.trustpassword=univbejaia
superadmin.cn=AdminUnivBejaia
superadmin.dn=CN=${superadmin.cn},O=Universite de
Bejaia,C=DZ
superadmin.password=ejbca
httpsserver.password=serverpwd
httpsserver.hostname=ca.univ-bejaia.dz
```

- **Le fichier « install.properties » :**

```
ca.name=AdminCA1
ca.dn=CN= AdminCA1,O=Universite de Bejaia,C=DZ
ca.tokenpassword=null
ca.keytype=RSA
ca.keyspec=2048
ca.signaturealgorithm=SHA256WithRSA
```

- **Le fichier « ocsp.properties » :**

```
ocsp.enabled=true
ocsp.defaultresponder=CN=Autorite de Validation,
O=Universite de bejaia,C=DZ
ocsp.signaturealgorithm=SHA256WithRSA
```

- **Le fichier « va-publisher.properties » :**

```
ocsp-datasource.jndi-name=OcspDS
ocsp-database.url=jdbc:mysql://192.168.1.6:3306/ejbca
ocsp-database.driver=com.mysql.jdbc.Driver
ocsp-database.username=ejbcaca
ocsp-database.password=ejbca
```

L'adresse 192.168.1.6 est l'adresse IP de la machine implémentant le serveur OCSP.

- Nous allons maintenant passer à l'étape suivante qui est le déploiement d'EJBCA. Cela ce fait en suivant les étapes suivantes dans l'ordre:

  - compiler les fichiers d'installation d'EJBCA : cela se fait comme suit :

```
# cd /opt/ejbca/
# ant bootstrap
…
Build SUCCESSFUL
Close Command Prompt of EJBCA
```

  - Ouvrir un autre terminale et lancer le serveur JBoss: cela se fait comme suit:

```
#cd /opt/jboos/bin
#./run.sh –b 0.0.0.0
…
Started in 3m:30s:229ms
```

Attendre jusqu'a ce que le serveur JBoss soit complètement démarré pour passer a l'étape suivante :

  - Installer EJBCA : cela ce fait en exécutant la commande suivante:

```
#ant install
```

  - Déployer EJBCA : il est nécessaire de stopper le serveur JBoss en appuyant sur Ctr+c. On pourra désormais déployer EJBCA en exécutant la commande suivante :

```
#ant deploy
…
Build SUCCESSFUL
```

L'installation de l'autorité racine de certification EJBCA est finie.

Afin de pouvoir accéder à l'interface d'administration d'EJBCA, il faut tout d'abord relancer le serveur JBoss. Ensuite il faut installer le certificat d'administration généré lors de l'installation d'EJBCA dans le navigateur web (Mozilla Firefox). Lors de l'importation du certificat d'administration, un mot de passe est demandé pour la sécurité personnelle du navigateur qui permet de protéger le magasin de certificats et un autre pour le fichier PKCS

#12 (.p12) qui est prédéfini dans un fichier de configuration (web.properties) dans la propriété « superadmin.password ».

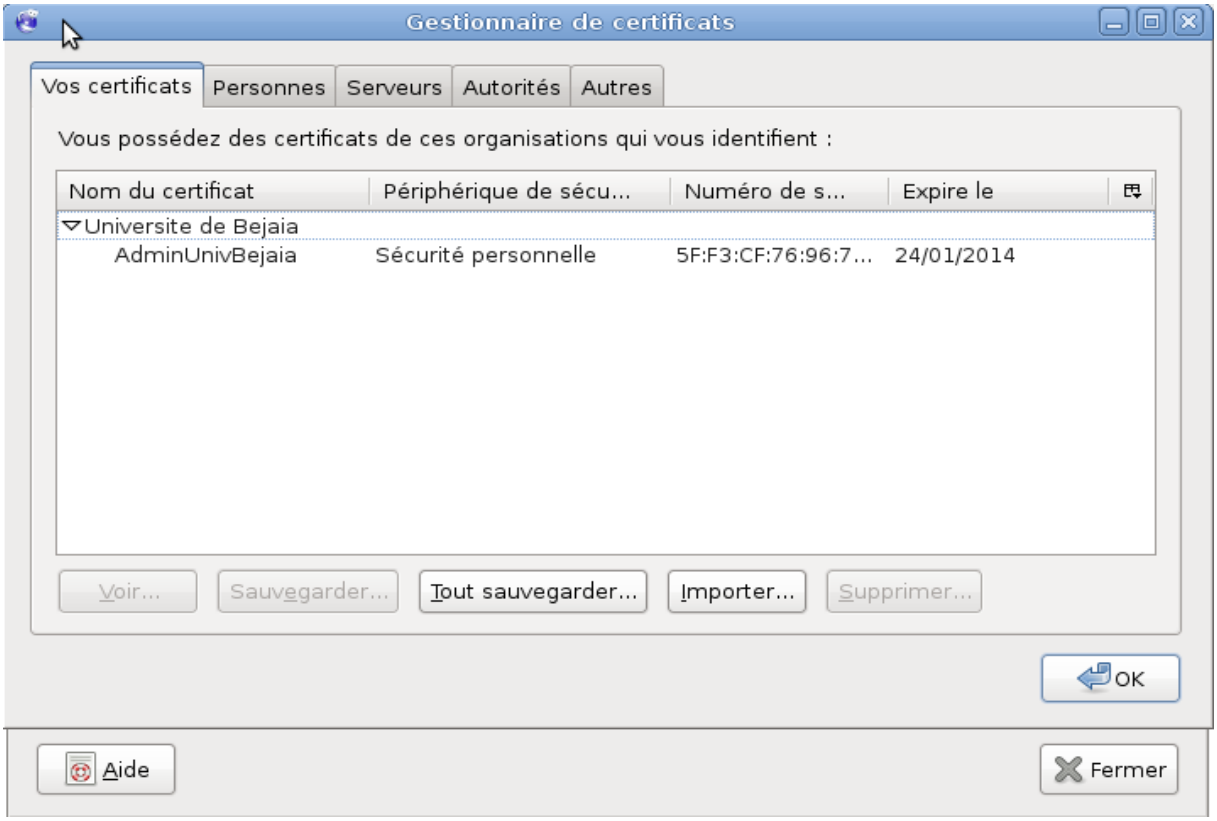

Figure III.3 : installation du certificat d'administration EJBCA.

Une fois le certificat intégré dans le navigateur, on peut se connecter sur l'interface d'administration à l'URL suivante : https://ca.univ-bejaia.dz:8443/ejbca/adminweb/index.jsp

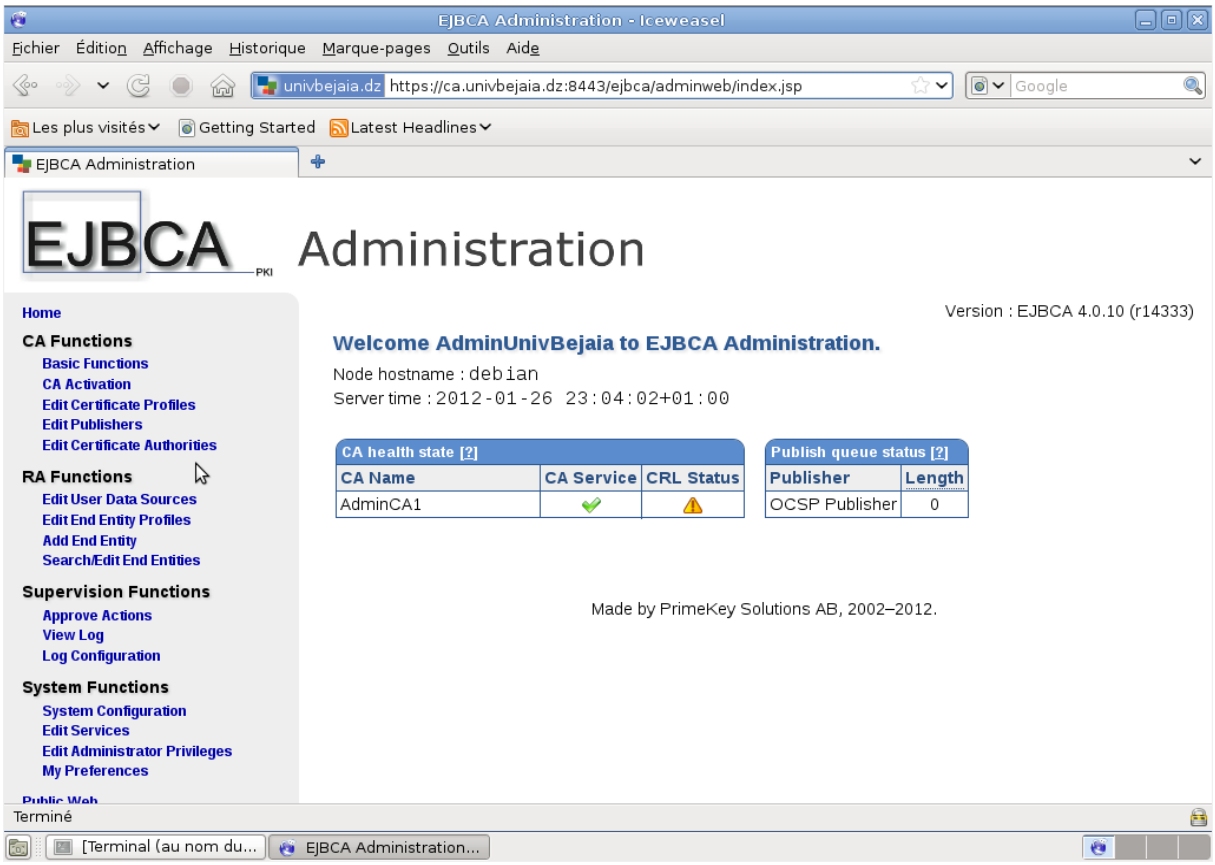

Figure III.4 : Interface d'administration d'EJBCA.

Deuxième étape : sur la machine du répondeur OCSP.

La procédure d'installation du serveur OCSP est semblable à celle de l'installation d'EJBCA de l'étape 1 à l'étape 6, à la seule différance que dans l'étape 6 les fichier a configurer sont ejbca.properties, database.properties, ocsp.properties.

1- Configuration des fichiers :

- **Le fichier « ejbca.properties » :**

```
appserver.home=/opt/jboss
appserver.type=jboss
ejbca.productionmode=ocsp
jboss.config=ejbca
```

- **Le fichier « database.properties » :**

```
database.name=mysql
database.url=jdbc:mysql://127.0.0.1/ejbca
database.driver=com.mysql.jdbc.Driver
database.username=ejbca
database.password=ejbca
```

- **Le fichier « ocsp.properties » :**

```
ocsp.enabled=true
ocsp.defaultresponder=CN=Autorite de Validation,
O=Universite de bejaia,C=DZ
ocsp.signaturealgorithm=SHA256WithRSA
j2ee.web-noconfigure=true
ocsp.keys.storePassword=foo123
```

2- Déploiement d'OCSP :

Le déploiement d'OCSP se fait en exécutant la commande suivante :

```
#cd /opt/ejbca
#ant ocsp-deploy
…
BUILD SUCCESSFUL
Close Command Promp of EJBCA
```

Après cela, il faut lancer le serveur JBoss avec la commande suivante :

```
#cd /opt/jboos/bin
#./run.sh –b 0.0.0.0
…
Started in 2m:30s:100ms
```

III.2.4 Architecture de la PKI:

Pour mettre en œuvre une PKI avec EJBCA, il faut d'abord définir une hiérarchie qui s'adapte à nos besoins. Pour cela on doit créer un modèle qui comporte une autorité racine avec laquelle on a définis un serveur SSL depuis lequel on crée des utilisateurs finaux et un serveur OCSP.

Ce tableau ci-dessous décrit les attributs principaux de chaque élément de la hiérarchie.

Autorité de certification racine	
Signing Algorithm	SHA256WithRSA
Key Usage	Key certificate sign, CRL sign
RSA key size	2048
Validity	3650d
Subject DN	CN=Autorite de Validation,O=Universite de Bejaia,C=DZ
Certificate Profile	AC_Racine_Universite_Bejaia
Default OCSP Service Locator	http://ocsp.univ-bejaia.dz:8080/ejbca/publicweb/status/ocsp
Profile du serveur SSL	
Signing Algorithm	SHA256WithRSA
Key Usage	Digital Signature,Key encipherment
Extended key usage	Server Authentification
RSA key size	2048
Validity	3650d
Available CAs	AC_ Universite_Bejaia
Publisher	OCSP Publisher
Profile du serveur OCSP	
Signing Algorithm	SHA256WithRSA
Key Usage	Digital Signature
Extended key usage	OCSP Signer
RSA key size	2048
Validity	720d
Available CAs	AC_ Universite_Bejaia
Publisher	OCSP Publisher
Entité Finale du serveur SSL	
End Entity Profile	Profile_Utilisateur_SSL
Username	www.univ-bejaia.dz
Password	ejbca
CN, Common name	www.univ-bejaia.dz
O, Organization	Universite de Bejaia
C, Country	DZ
DNS Name	www.univ-bejaia.dz
Certificate Profile	Profile_Serveur_SSL

CA	AC_Universite_Bejaia
Token	P12
Entité Finale du serveur OCSP	
End Entity Profile	Profile_Utilisateur_OCSP
Username	Repondeur_OCSP
Password	foo123
CN, Common name	Repondeur_OCSP_UnivBejaia
O, Organization	Universite de Bejaia
C, Country	DZ
Certificate Profile	Profile_Serveur_OCSP
CA	AC_Universite_Bejaia
Token	P12

Tableau III.4 : Hiérarchie de la PKI.

III.2.5 Création des autorités de certification et entités

Apres avoir installé EJBCA, tous les logiciels requis cité dans le Tableau III.3 et définie la hiérarchie de notre PKI, on peut procéder à la création des différentes autorités de certification et entité avec les propriétés définies dans le Tableau III.4.

Cette procédure se fait à partir de l'interface d'administration d'EJBCA en se connectant sur l'URL suivant : https://ca.univ-bejaia.dz:8443/ejbca/adminweb/index.jsp

- **Création des autorités de certification de la PKI:**

La création d'une autorité de certification se fait a partir de l'interface d'administration d'EJBCA, mais avant de procéder à cela, on doit tout d'abord créer un profile de certificat à partir de l'option « Create Certificate Profile » ou l'on doit spécifier les différentes propriétés définies dans le Tableau III.4.

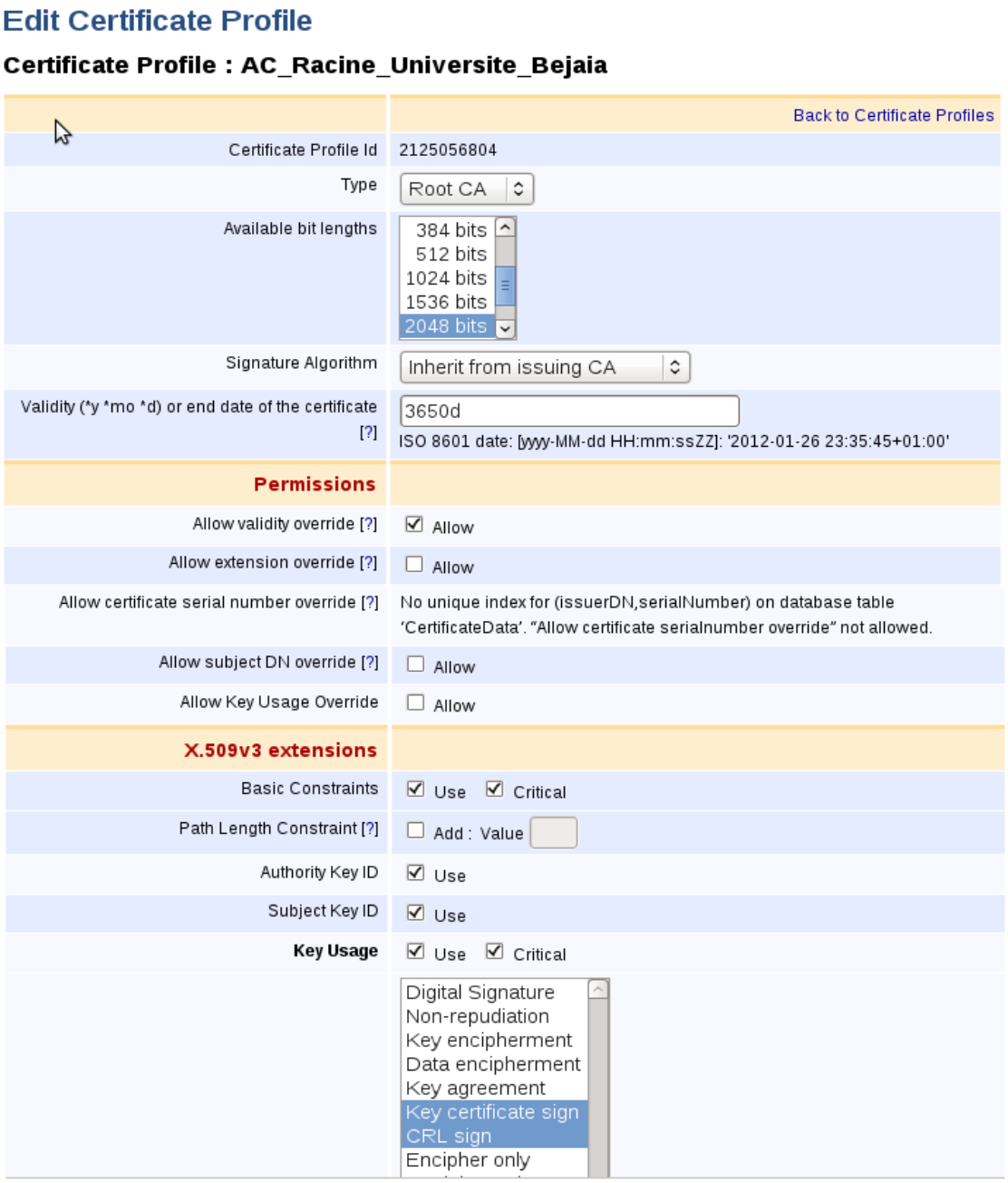

Figure III.5 : Création du profile de l'autorité de certification racine.

On peut désormais procéder à la création de l'autorité de certification à partir du profile crée auparavant en utilisant l'option « Create Certificate » et en se basant sur les propriétés défini dans le Tableau III.4.

Figure III.6 : Création de l'autorité de certification racine.

- **Création des entités finales de la PKI :**

Pour les besoins de ce projet, il a fallu procéder à la création des deux entités finales à partir des deux profils : serveur SSL, Serveur OCSP.

Avant la création des entités finales, on doit tout d'abord créer leurs profiles à partir de la fonction « Create End Entity Profile » où l'on doit spécifier les différentes propriétés définies dans le Tableau III.4.

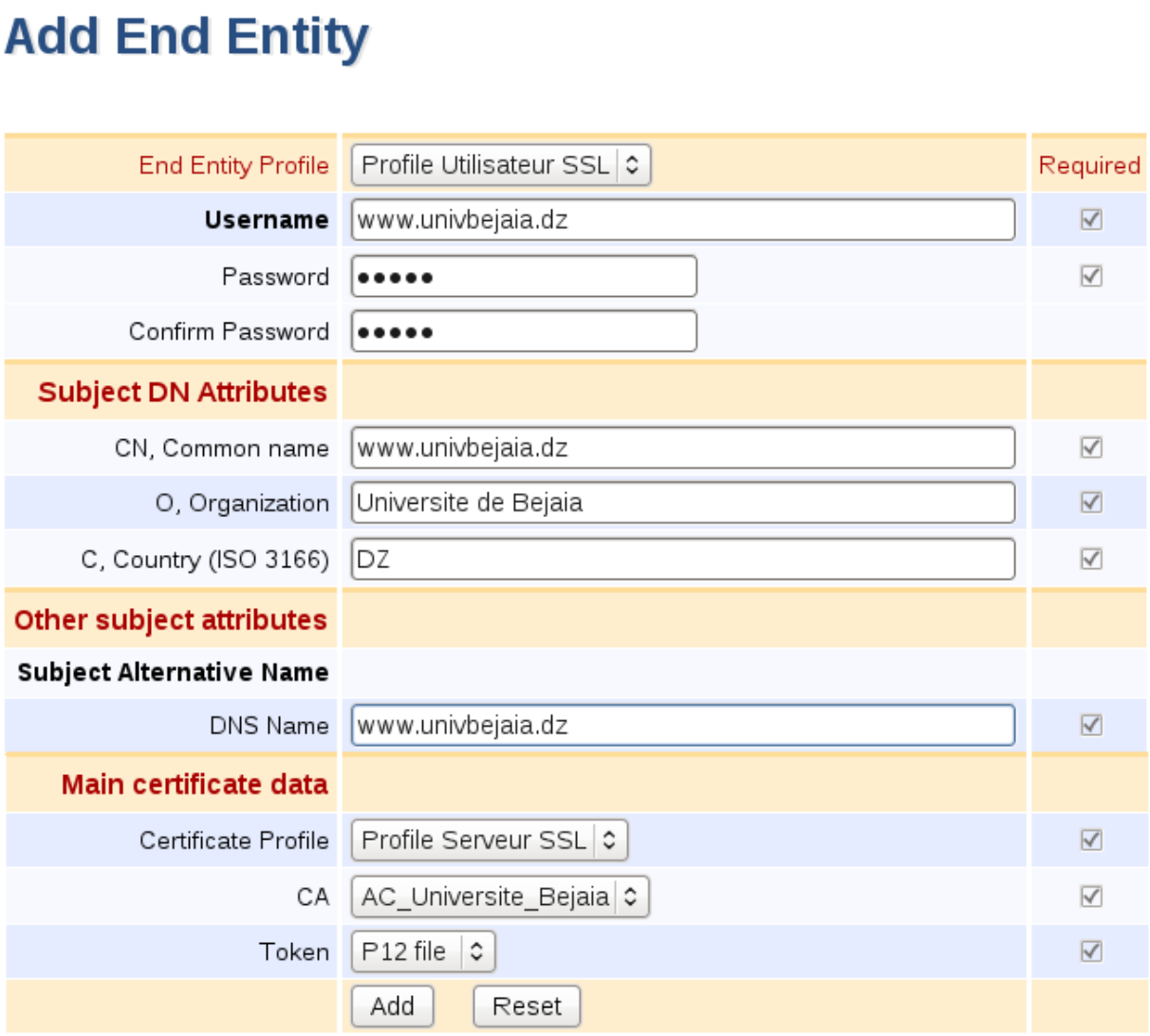

Figure III.7 : Création des entités finales.

Après la création des deux entités finales de notre PKI, l'étape suivante consiste à créer le certificat à clé publique de l'entité issue du profile du serveur SSL, ce qui suit décrit la méthode adoptée pour générer un certificat pour l'authentification du serveur web apache.

III.2.6 Implémentation d'Apache avec SSL

Dans cette partie nous allons décrire l'implémentation du protocole SSL sur un serveur Apache et ainsi lui donner la possibilité d'établir des connexions dites sécurisées.

Apache est un serveur web permettant à des clients d'accéder à des pages web à partir d'un navigateur (aussi appelé browser) installé sur leur ordinateur distant.
Il existe de nombreux modules **Apache** disponibles en vue d'ajouter des fonctionnalités au serveur de base. Le module **mod_ssl** est l'un des modules Apache qui utilise la bibliothèque OpenSSL pour offrir un chiffrement solide à l'aide des protocoles SSL (SSL v2/v3) et TLS. Ce module fourni tout ce qui est nécessaire à la demande de certificats signés auprès d'une autorité de certification connue de façon à pouvoir faire tourner un serveur web sécurisé.

- **Authentification du Serveur**

Pour pouvoir sécuriser le serveur apache, on doit tout d'abord générer son certificat à clé publique signé par notre autorité de certification. Cela se fait sur la machine sur laquelle s'exécute le serveur en générant la demande de signature et la clé privée de l'entité grâce à l'outil OpenSSL. Cela nécessite de fournir certaines informations sur l'identité de l'entité et le mot de passe qui protège la clé privée.
La demande de signature et la clé privée est générée avec la commande suivante :

```
#openssl req -new -newkey rsa:2048 -keyout cle.key -out
req.csr
Generating a 2048 bit RSA private key
.........................++++++
...++++++
writing new private key to 'cle.key'
Enter PEM pass phrase:*********
Verifying - Enter PEM pass phrase:*********
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished
Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:DZ
State or Province Name (full name) [Some-State]:Bejaia
```

```
Locality Name (eg, city) []:Bejaia
Organization Name (eg, company) [Internet Widgits Pty
Ltd]:Universite de Bejaia
Organizational Unit Name (eg, section) []:.
Common Name (eg, YOUR name) []:www.univbejaia.dz
Email Address []:.

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

L'étape suivante consiste à faire signer la demande par l'autorité de certification. Pour ce faire, il faudra accéder à l'interface web publique d'EJBCA avec l'URL http://ca.univ-bejaia.dz:8080/ejbca/publicweb et choisir la fonction « Create Certificate from CSR ». Pour la suite il faudra avoir en main les informations suivantes :

- Le nom d'utilisateur et mot de passe de l'entité faisant la demande de signature (voir la figure « Figure III.7 »**)**. Ces informations sont transmises a l'entité de vive voix ou par courriel.
- La demande de signature (req.csr) générée précédemment avec l'outil OpenSSL. Ensuite on peut soit copier son contenu dans le formulaire prévu pour la requête, soit l'importer directement.

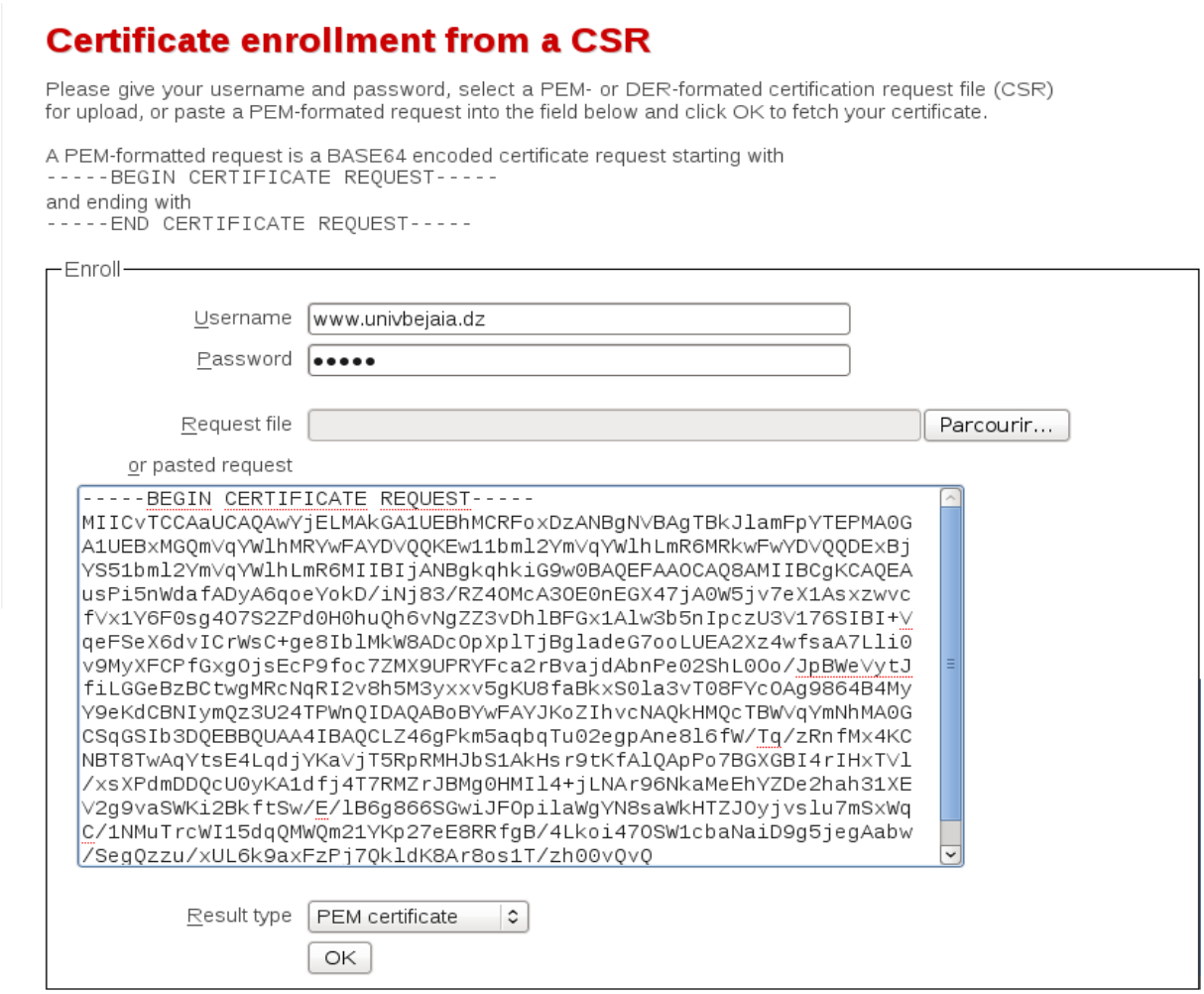

Figure III.8 : Signature du certificat SSL par L'AC.

Une fois le formulaire saisie avec les bonnes informations, on valide la demande de signature afin que l'autorité de certification valide les informations et signe le certificat avec sa clé privée. On aura comme résultat le fichier (www.univbejaia.dz.pem) qui contient le certificat et la clé privée.

Il nous reste maintenant à compléter la configuration du serveur Web Apache pour qu'il puisse utiliser le certificat obtenu lors de l'étape précédente. Il suffira de séparer le certificat signé et la clé privée que contient le fichier (www.univbejaia.dz.pem) en deux fichiers distincts (cert.crt et cle.key) et les placer dans le répertoire /etc/apache2/ssl. Ensuite on procède modification du fichier /etc/apache2/sites-available/default-ssl comme suite :

```
SSLCertificateFile              /etc/apache2/ssl/cert.crt
SSLCertificateKeyFile           /etc/apache2/ssl/cle_pri.key
```

SSLCertificateFile : spécifie le chemin vers le certificat du serveur web apache.
SSLCertificateKeyFile : spécifie le chemin vers la clé privée du certificat

Si dans la hiérarchie il existe des autorités intermédiaires, on utilise la directive « SSLCertificatePathChainFile » qui spécifie le chemin vers la chaine qui contient tous les certificats de la hiérarchie d'AC qui sont nécessaires à la validation du serveur. Le fichier (chain.pem) est téléchargeable à partir de l'option « Fetch CA & OCSP Certificate » de l'interface publique.

Nous allons maintenant procéder à un test sur le navigateur mais avant cela on a besoin de modifier le ficher /etc/hosts en ajoutant la ligne suivante :

```
127.0.0.1 www.univ-bejaia.dz
```

On relance le serveur apache avec la commande suivante :

```
#/etc/init.d/apache2 restart
```

On tente maintenant d'accéder à l'URL https://192.168.1.5/ à partir du notre navigateur du poste client :

Figure III.9 : teste de validation du certificat serveur**.**

D'après la figure, il est impossible de procéder à la validation du certificat serveur. Cela est du au fait que le certificat racine de notre PKI qui permet de valider le certificat du serveur n'est pas encore installé sur le navigateur du poste client.

L'installation du certificat racine se fait à partir de la fonction « Fetch CA & OCSP Certificate » de l'interface publique, en le téléchargement directement dur le navigateur et choisir l'action pour identifier des sites web.

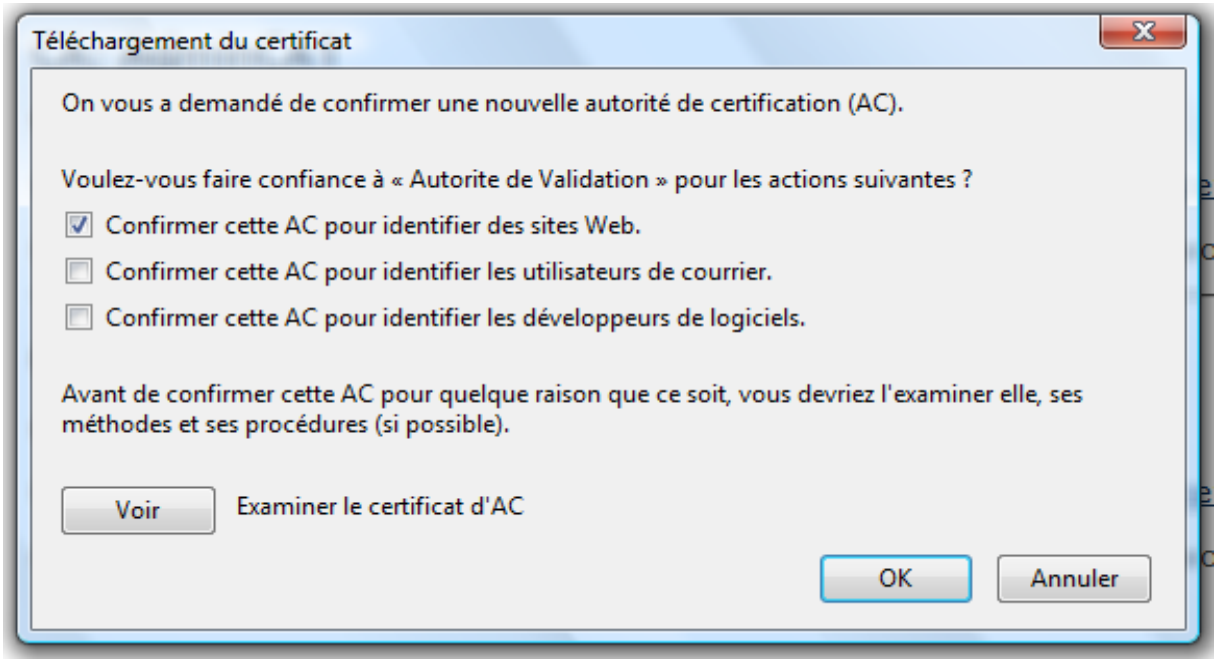

Figure III.10 : Installation du certificat racine dans le navigateur.

Suite à l'installation de notre certificat racine, il est désormais possible d'accéder à la page principale de notre serveur de test de sécurité en évitant l'avertissement rencontré auparavant

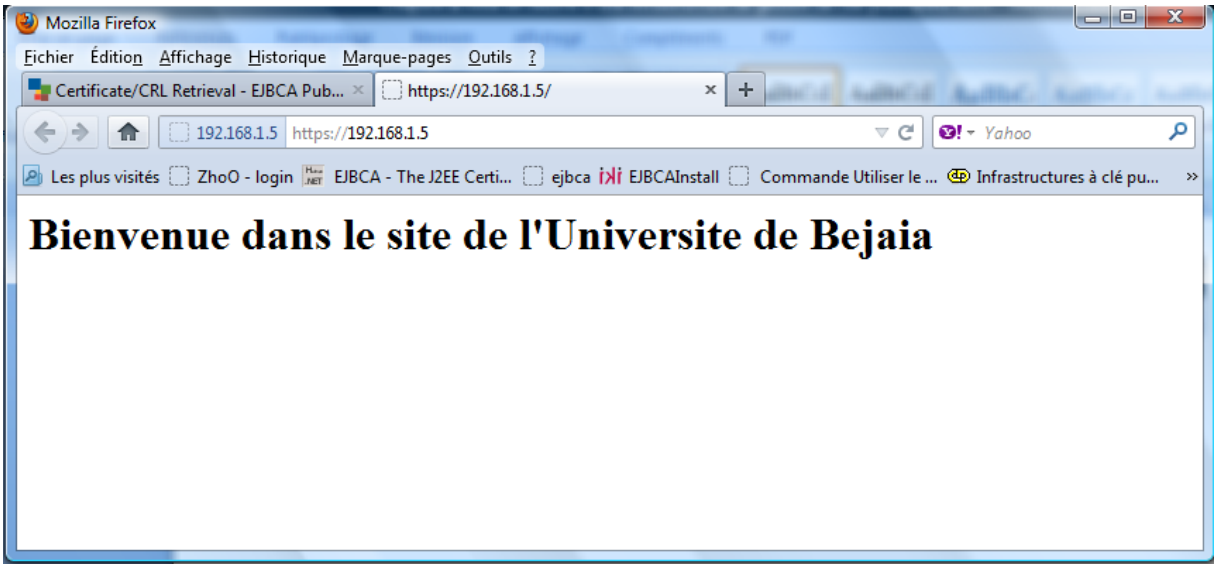

Figure III.11 : Page d'accueil du serveur de test.

III.2.7 Vérification de la validité des certificats numérique par le serveur OCSP

Apres avoir créé le profile de certificat OCSP (voir la section III.2.3.3). On peut maintenant implémenter les points essentiels pour le bon fonctionnement du répondeur OCSP.

- **Création d'un utilisateur pour la machine EJBCA**

Cette étape s'effectue sur la machine ou le serveur OCSP est installé.la création d'un utilisateur « ejbcaca » identifié par un mot de passe « ejbca » avec des privilèges sur la base de données « ejbca » créé précédemment s'effectue comme suite :

```
#mysql –uroot –p
password :mot de passe root
```

```
mysql>use ejbca ;
```

```
mysql>create user 'ejbcaca'@'192.168.1.5' identified by
'ejbca';
```

L'adresse ip 192.168.1.5 est l'adresse de la machine EJBCA.

```
mysql>grant create,insert,select,update,delete on ejbca.* to
'ejbcaca'@'192.168.1.5';
```

- **Création du service de publication OCSP**

A partir de la page d'administration d'EJBCA, on choisi la fonction « Edit Publishers » et on créé un service de publication nommé « OCSP Publisher » avec les propriétés nécessaires. Et on sauvegarde et on test la connexion du service de publication OCSP avec le bouton « Save and Test Connection ».

Figure III.12 : Connexion au serveur OCSP.

- **Exportation de l'entité OCSP**

Cette étape s'effectue sur la machine EJBCA à partir de l'interface publique. On doit télécharger le certificat de l'entité OCSP « Repondeur_OCSP_unvibejaia.p12 » en accédant à la fonction « Create Browser certificate » et on saisissant un nom et un mot de passe spécifier lors de la création de l'entité OCSP.

Ensuite on doit copier ce fichier et le placer dans la machine OCSP à l'emplacement suivant : /opt/jboss/bin/keys.

- **Synchronisation de la base de données**

Il est nécessaire d'effectuer une synchronisation initiale de la base de données en exécutant sur la machine EJBCA la commande suivante :

```
#mysqldump -uroot -p ejbca CertificateData > sync.sql
```

On copie le fichier « sync.sql », on le place dans la machine OCSP dans le répértoir /opt/
Et on l'importe avec la commande suivante :

```
#mysql -uroot -p ejbca < sync.sql
```

- **Tester la validité d'un certificat**

Avant d'effectuer un test on doit redémarrer le serveur JBoss sur les deux machines (EJBCA et OCSP), on télécharge ensuite le fichier (www.univbejaia.dz.pem) de l'entité créé a partir du profile serveur SSL avec l'option « Search/Edit End Entities» de l'interface d'administration d'EJBCA, et le fichier (AutoritedeValidation.pem) de l'autorité de certification « AC_Univ_Bejaia» dans l'option « Fetch CA & OCSP Certificates » de l'interface publique.
On peut maintenant faire notre test en utilisant l'outil OpenSSL avec la commande suivante :

```
#openssl ocsp –issuer AutoritedeValidation.pem –cert
www.univbejaia.dz.pem –CAfile AutoritedeValidation.pem –text –
url http://192.168.1.6:8080/ejbca/publicweb/status/ocsp
```

Le résultat de cette commande s'affiche sur le terminal comme suite :

```
OCSP Request Data:
    Version: 1 (0x0)
    Requestor List:
        Certificate ID:
          Hash Algorithm: sha1
          Issuer Name Hash:
4145F8A5CCF07E01EBF1D22D40A1E29392B1E02E
          Issuer Key Hash:
FE537B40381C97926B154ED8E9288BDF47B422AA
          Serial Number: 4C3BB2CF27678EE8
    Request Extensions:
        OCSP Nonce:
```

```
            0410F4EA2A76F9CEBA624EF13A75A2F25792
Response verify OK
www.univ-bejaia.dz.pem: good
      This Update: JUN 19 13:50:26 2012 GMT
```

Sur le terminal JBoss du répondeur OCSP s'affiche :

```
13:50:26,007 INFO  [OCSPServletBase] Received OCSP request for
certificate with serNo: 4a4dde76be1b9d51, and issuerNameHash:
ed5a88e311dfc0b56bfa2e1254e8cf014e24fa84. Client ip
192.168.1.5.
13:50:26,116 INFO  [OCSPServletBase] Adding status information
(good) for certificate with serial '4a4dde76be1b9d51' from
issuer 'CN=Autorite de Validation,O=Universite de
Bejaia,C=DZ'.
```

III.2.8 Publication LDAP

Dans cette partie, nous allons mettre en œuvre un annuaire LDAP pour stocker les certificats à clé publique.

- **Présentation d'Open LDAP :**

Lightweight Directory Access Protocol, ou LDAP, est un protocole réseau pour l'interrogation et la modification des services d'annuaire sur TCP / IP. OpenLDAP est une implémentation libre du protocole LDAP développée par « The OpenLDAP Project ». C'est un annuaire qui fonctionne sur le modèle client/serveur. Il contient des informations de n'importe quelle nature qui sont rangées de manière hiérarchique.

- **Installation et configuration d'Open LDAP :**

Pour installer Open LDAP on doit procéder comme suit :

  - Installation du démon « sldap » et un ensemble d'outils standard « ldap-utils » : cela se fait avec les commandes suivantes :

```
#apt-get install slapd
#apt-get install ldap-utils
```

Lors de l'installation de sldap, il faudra fournir les informations suivantes :

```
Omit OpenLDAP server configuration? No
DNS domain name: univ-bejaia.dz
Organization name? univ-bejaia.dz
Administrator password: {PASSWORD}
Confirm password: {PASSWORD}
Database backend to use: HDB
Do you want the database to be removed when slapd is purged?
No
Allow LDAPv2 protocol? No
```

- Modifier le fichier « ldap.conf » qui se trouve dans le répértoire /etc/ldap de la façon suivante :

```
BASE dc=univbejaia,dc=dz
URI ldap://ldap.unvibejaia.dz
```

- **Publier les certificats :**

Pour publier les certificats il faut accéder à la page « Edit publisher » de l'interface d'administration d'EJBCA, ajouter un annuaire « LDAP publisher » et remplir les informations suivantes :

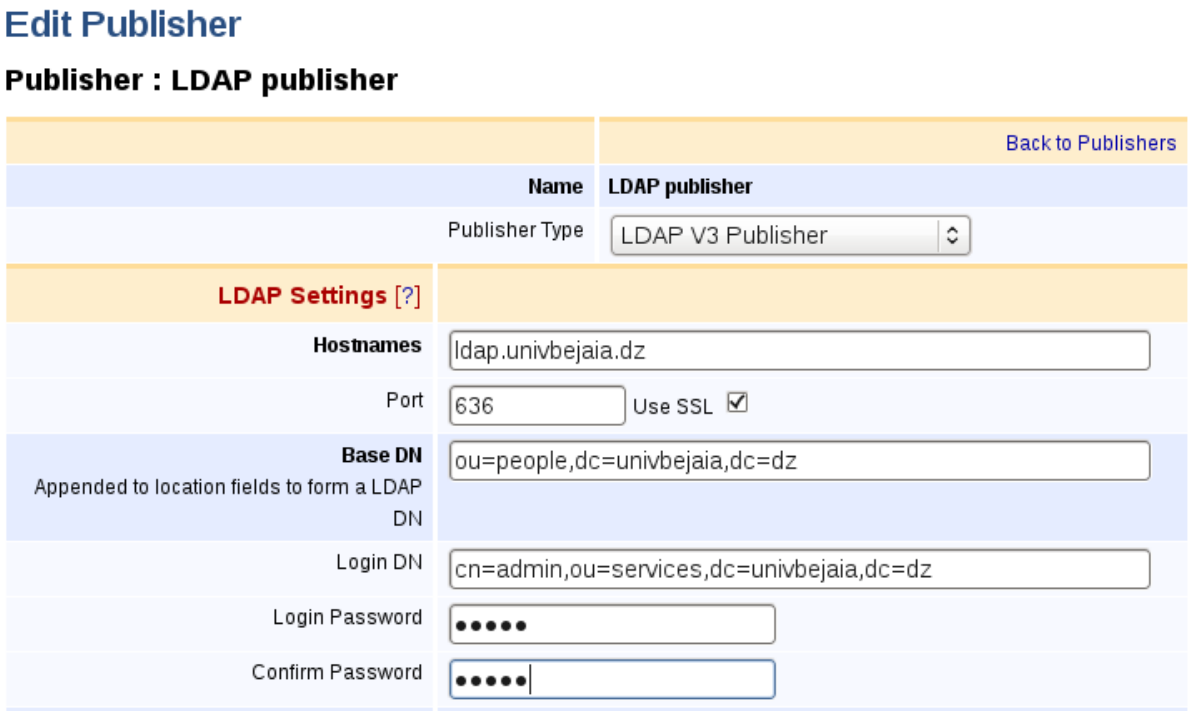

Figure III.13 : Configuration de LDAP.

Le mot de passe doit être celui décrit dans le processus de création du certificat.

Après cela, il faut tester la connexion à l'annuaire LDAP en cliquant sur le bouton « Save and test Connection ».
La prochaine étape consiste à ajouter l'annuaire LDAP aux différents profils d'EJBCA en éditant ces derniers et en sélectionnant l'annuaire LDAP dans le champ « Publisher ».
Enfin, il faut republier les certificats pour que ces derniers soient publiés dans tous les annuaires actifs.

```
#cd /opt/ejbca/
# bin/ejbca.sh ca republish 'AC_Univ_Bejaia'
```

Une fois que les certificats sont republiés, on peut effectuer des recherches grâce à la commande « ldapsearch ». On peut maintenant vérifier dans la base de données la liste des utilisateurs et les certificats par la commande suivante:

```
#ldapsearch -Y EXTERNAL -H ldapi:/// '(userCertificate=*)' userCertificate
```

Conclusion

Dans ce chapitre, Nous avons vu comment mettre en place une infrastructure de gestion de clés publique avec EJBCA. Nous avons créé une autorité de certification, un serveur web Apach sécurisé, un serveur OCSP pour valider les certificats et un annuaire LDAP pour stocker ces derniers.

Comme nous avons pu le constater, EJBCA est une PKI très puissante, complète et facile à administrer, mais qui s'avère très complexe lors de son installation. Elle facilite la gestion des certificats en intégrant le serveur OCSP et la mise à jour automatique des CRL.

CONCLUSION GENERALE

Durant notre projet, nous avons pu mettre en œuvre une infrastructure de gestion de clés publiques avec EJBCA. Ainsi nous avons eu une idée sur le fonctionnement de la PKI et son importance dans le cadre du centre de calcul. La mise en place d'une PKI au sein du centre de calcul a permis de faciliter la gestion des clés publiques et la sécurisation des sites web et autre applications par le moyen des certificats numériques.

La conclusion principale à la quelle nous sommes arrivées à travers ce modeste travail est que l'infrastructure de gestion de clés publique EJBCA est une solution efficace et facile à réaliser pour remédier à la problématique de gestion des clés publiques et certificats numériques.

Les principaux résultats auxquels nous sommes parvenus à dégager de cette étude, et qui constituent les réponses aux questions posées dans la problématique, sont résumés dans ce qui suit :

L'infrastructure de gestion de clés publique est un moyen efficace pour assurer les principes de base de la sécurité informatique et d'échanger ainsi des clés publiques et des certificats numériques dans divers environnements sécurisés. Elle se compose principalement des entités finales qui sont les utilisateurs voulant certifier leurs clés publiques, des autorités de certifications qui délivrent les certificats numériques aux entités finales, des autorités d'enregistrement et des annuaires pour le stockage des certificats.

Nous avons installé EJBCA sur une machine d'administration et un serveur OCSP sur une autre machine. Nous avons établi une architecture hiérarchique comportant une autorité racine avec laquelle on a définis un serveur SSL depuis lequel on crée des utilisateurs finaux et un serveur OCSP. Si les utilisateurs finaux veulent vérifier la validité des certificats numériques, cela sera possible en envoyant une requête au serveur OCSP qui se charge de vérifier la validité du certificat et d'envoyer la réponse aux utilisateurs.

Pour faciliter l'accessibilité aux différents certificats numériques disponibles générés, nous avons établi un annuaire LDAP qui se charge de stocker les certificats, les clés publiques et les listes de révocation (CRL).

En perspective, il faudra installer l'autorité de certification EJBCA , ainsi que le répondeur OCSP et enfin un annuaire LDAP dans les différents services du centre de calcul .

Nous voudrions aussi étendre notre solution en intégrant l'authentification par carte à puce pour authentifier les étudiants de l'université de Bejaia.

Bibliographie

REF02: Implementation of ISO 7498-2, Information processing systems - Open systems Interconnection - Basic reference mode, Part 2: Security architecture.

REF04 : DUBUISSON Olivier, ANS.1, *Communication entre systèmes hétérogènes*, Paris, Springer-Verlag France et CNET France Télécom, 1999, p 60.

REF05 : LLORENS Cédric et al**.** *Tableaux de bord de la sécurité réseau,* 2e édition, Paris, Eyrolles, 2006.

REF06 : Une Introduction à la Cryptographie, 1998, document officiel de Network Associates Inc. (NAI), Traduction française par news:fr.misc.cryptologie

REF09: MAWLOUD Omar, cours de sécurité 2, Université de Béjaïa, 2012.

REF10: ARCHIMBAUD Jean-Luc, *Les IGC, infrastructures de gestion de clés*, Les Cahiers du numérique, 2003 /3 Vol. 4, p 116.

REF13: Ayesha Ghothi et Asra Parveen, George Mason University-Fall, 2006, Traduction de l'article : PKI administraion using ejbca and openCA ,p 12.

Webographie

REF01: ISO copyright office, http://www.iso.org/iso/fr/about.htm**,** Genève.

REF03: Internet Society, http://www.internetsociety.org/fr/%C3%A9v%C3%A9nements/r%C3%A9unions-ietf, États-Unis et Suisse, 2012.

REF07: Adrien BERNARD, http://www.techno-science.net/?onglet=glossaire&definition=6133, SARL CLEVACTI, date consultation: 13 mai 2012.

REF08: Vincent LIMORTE, François VERRY et Sébastien FONTAINE, http://www.authsecu.com/ssl-tls/ssl-tls.php**,** date de publication : 23 décembre 2006.

REF11 : Erik ANDERSEN, http://www.x500standard.com/index.php?n=X509W.OCSP, date de publication : 06 Mai 2009,

REF12 : Michel Gardie , http://www-public.it-sudparis.eu/~gardie/LDAP/Articles/annuaires-LDAP.html, date consultation: 20 mai 2012.

REF14 : http://www.ossir.org/sur/supports/2008/Presentation-EJBCA-FR_1.1.pdf

Resumé

Le centre de calcul de l'université de Béjaia veut sécuriser ses différentes applications au moyen d'une infrastructure de gestion de clés publiques (PKI) qui se charge de gérer ses différentes clés publiques et de certifier ses applications.

Pour cela nous avons mis en œuvre une PKI avec EJBCA dans laquelle on a définis une autorité de certification, un serveur OCSP, et un serveur SSL.

Nous avons donné la possibilité au client de vérifier son certificat en envoyant une requête au serveur OCSP, et on lui a permet aussi de consulter les différents certificats générés au moyen d'un annuaire LDAP.

Mots clés : PKI, clé publique, EJBCA, autorité de certification, serveur OCSP, serveur SSL, LDAP.

Abstract

The computing center of the University of Bejaia wants to secure its various applications using a Public Key Infrastructure (PKI) which is responsible for managing its differents public keys and certify his applications.

For this we implemented a PKI with EJBCA in which we have defined a certification authority, an OCSP server and an SSL server.

We gave the possibility to the client to verify his certificate by sending a request to the OCSP server, and we also permitted him the possibility to consult the differents certificates generated using an LDAP directory.

Keywords: PKI, public key, EJBCA, certification authority, OCSP server, SSL server, LDAP.