# Liste des figures et tableaux

# Table des figures

# Liste des tableaux

# Glossaire

| | |
|---|---|
| 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 |
| 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 |
| IIS | Internet Information Security |

# Glossaire

| | |
|---|---|
| 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 |

# Table des matières

# Introduction Générale

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.[**W8**]

La sécurité des systèmes d'information est donc un enjeu majeur des technologies numériques modernes. Avec le développement de l'Internet, les besoins de sécurité sont de plus en plus importants. En effet, les besoins tels que, l'identification des entités communicantes, l'intégrité des messages échangés, la confidentialité des transactions, l'authentification des entités, etc., liées à la sécurité des communications sont à satisfaire.

La PKI (Public Key Infrastructure) est un ensemble de composants physiques (ordinateurs, équipements cryptographiques logiciels ou matériel type HSM ou encore des cartes à puces), de procédures humaines (vérifications, validation) et de logiciels (système et application) destiné à gérer les clés publiques des utilisateurs d'un système. Elle permet aussi de créer, gérer, distribuer, utiliser, stocker et révoquer des certificats numériques.

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

## Contexte

Toutes structures informatisés, 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. Face à cette montée en puissance, la certification de ses applications et de 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 ces services . En d'autres termes, quelles sont les solutions possibles pour certifier ses applications et fournir une bonne gestion de clés publiques ?

## Problématique

Pour mieux cerner cette problématique, nous avons pris un exemple de site non sécuriser 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 ?,
— 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 ? ;

## Objectif

Notre objectif c'est de mettre en place une infrastructure de gestion de clés afin d'implémenter la signature électronique . Cette infrastructure aura la capacité de délivrer des certificats d'utilisateurs signés par une autorité intermédiaire elle-même possédant un certificat signé par une autorité racine qui possède un certificat auto-signé.

## Organisation du memoire

Ce mémoire s'articule autour de trois chapitres :

— 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.
— Dans le troisième et dernier chapitre, nous allons décrire la mise en place d'une PKI avec EJBCA ainsi qu'une application test.

# 1 Chapitre I : Cryptographie et sécurité informatique

**Introduction** :

La sécurité informatique c'est l'ensemble des moyens mis en œuvre pour réduire la vulnérabilité d'un système contre les menaces accidentelles ou intentionnelles. Il convient d'identifier les exigences fondamentales en sécurité informatique. 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.

## 1.1 Normes et Standards

### 1.1.1 ISO 27001

L'ISO (Organisation internationale de normalisation) est une organisation internationale non gouvernementale, indépendante, dont les 165 membres sont les organismes nationaux de normalisation.

Comme toutes les autres normes de systèmes de management de l'ISO, la certification selon ISO/IEC 27001 est une possibilité, mais pas une obligation. Certains utilisateurs décident de mettre en œuvre la norme simplement pour les avantages directs que procurent les meilleures pratiques. D'autres font le choix de la certification pour prouver à leurs clients qu'ils suivent les recommandations de la norme. De nombreuses organisations dans le monde sont certifiées à ISO/IEC 27001. La norme ISO 27001 :2013 (également connue sous le nom de BS EN 27001 :2017) fournit un cadre pour un système de gestion de la sécurité de l'information (ISMS) qui permet le main-

tien de la confidentialité, de l'intégrité et de la disponibilité des informations, ainsi que la conformité légale. La certification ISO 27001 est essentielle pour protéger vos actifs les plus vitaux.

### 1.1.2 IETF

- **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 PKIX1 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 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

### 1.1.3 UIT-T

La Recommandation UIT-T X.509 | ISO/CEI 9594-8 définit des cadres pour l'infrastructure de clé publique (PKI) et l'infrastructure de gestion de privilège (PMI). Elle présente le concept fondamental de techniques de chiffrement asymétrique. Elle spécifie les types de données suivants : certificat de clé publique, certificat d'attribut, liste de révocation de certificats (CRL) et liste de révocation de certificats d'attribut (ACRL).

Elle définit aussi plusieurs certificats et extensions de liste CRL, ainsi que les informations relatives au schéma d'annuaire permettant de stocker dans un annuaire les données relatives aux infrastructures PKI et PMI. Elle définit en outre des types d'entités, tels que l'autorité de certification, l'autorité d'attribut, la partie utilisatrice, le vérificateur de privilège, le courtier de confiance et l'ancre de confiance. Elle spécifie les principes régissant la validation de certificat, le trajet de validation, la politique de certificat, etc. Elle inclut une spécification des listes de validation d'autorisation qui permettent d'effectuer une validation rapide et d'imposer des restrictions aux communications.

### 1.1.4 PKCS

Ensemble de standards fait pour la cryptographie à clé publique développé en coopération avec un consortium informel (Apple, DEC, Lotus, Microsoft, MIT, RSA, et Sun) qui comprend des standards de mise en oeuvre spécifiques à des algorithmes ou indépendants de tout algorithme. Les spécifications définissant la syntaxe des messages et d'autres protocoles sont contrôlés par RSA Data Security Inc.

## 1.2 Mathématique pour la cryptographie

### 1.2.1 Groupe

Loi de composition, ou opération..[12]

Soit E, un ensemble non vide. On dit q'une fonction $* : E \times E \rightarrow E$ est une loi de composition sur E, ou une opération binaire sur E On note $(E, *)$ le fait que l'ensemble E est muni d'une opération binaire. Dans ce cas, on utilise souvent une notation infixe, c'est-à-dire que

$* : E \times E \rightarrow E$, avec (x,y) $\rightarrow x * y$, où l'image de (x, y) par la fonction « $*$» est notée $x * y$.

Parmi les lois de composition, certaines possèdent des propriétés particulières qui les rendent plus intéressantes. Le choix de ces propriétés n'est pas arbitraire. En effet, c'est une vaste expérience mathématique qui a permis de dégager qu'elles sont les propriétés qui donnent à une loi de composition une structure suffisamment riche pour qu'elle ait un impact important sur l'étude d'un contexte dans lequel elle apparaît. Nous aurons maintes fois l'occasion de constater qu'une fois mises en évidence ces propriétés apparaissent toutes naturelles. On dit q'une loi de composition (opération) $*$, est

1. associative si $x * (y * z) = (x * y) * z$, pour tout x, y, z $\in$ E
2. commutative si $x * y = y * x$, pour tout x,y $\in$ E

On remarque que, si $*$ est associative, alors on peut écrire $x*y*z$ au lieu de $(x*y)*z = (x*y)*z$, puisqu'il n'y a pas d'ambiguïté sur la façon de faire le calcul. Bien entendu, toutes les lois ne sont pas associatives.

Par exemple, on a

— Les opérations usuelles d'addition « + » et de multiplication « . » d'entiers (dans $\mathbb{Z}$) sont toutes deux commutatives et associatives. Il en est de même pour

les entiers modulo n, c.-à-d. dans $\mathbb{Z}n = \mathbb{Z}/n\mathbb{Z}$. Dans ce qui suit, on suppose que l'ensemble $\mathbb{Z}n$ est identifié à 0, 1,...,n

— La loi de composition $* : (x, y) \longrightarrow xy + 1$ sur $\mathbb{N}$ est commutative, mais pas associative. En effet, pour x, y, z $\in \mathbb{N}, ona$

$(x * y) * z = (xy + 1) * z =$ (xy + 1)z + 1 = xyz + z + 1, et

$x * (y * z) = x * (yz + 1) =$ x(yz + 1) = xyz + x + 1.

Les résultats sont donc manifestement différent si $x \neq z$.

— On vérifie facilement que l'opération $x * y := x^y$, pour x et y dans $\mathbb{N}$, n'est ni associative ni commutative.

— Dans l'ensemble $\mathbb{M}_n(R)$ des matrices n x n à coefficients réels, l'addition est une loi associative et commutative, tandis que la multiplication est une loi associative, mais pas commutative en général

### 1.2.2 Anneau

[13]

— A est un groupe abélien pour l'addition, (on note 0 son élément neutre),

— la multiplication est associative, c'est-à-dire :

x(yz) = (xy)z pour tous x, y, z ∈ A.

— la multiplication est distributive sur l'addition à gauche et à droite, c'est-'à-dire :

x(y + z) = xy + xz et (x + y)z = xz + yz pour tous x, y, z ∈ A.

On dit que l'anneau A est commutatif si de plus la multiplication est commutative, c'est-à-dire :

xy = yx pour tous x, y ∈ A.

On dit que A est unitaire si de plus la multiplication admet un élément neutre 1.

x.1 = 1.x = x pour tout x ∈ A.

Exemples :

1. L'ensemble $\mathbb{Z}$ des entiers est un anneau commutatif unitaire. Il en est de même de $\mathbb{Q}$, $\mathbb{R}$ et $\mathbb{C}$.
2. L'ensemble des matrices carrées d'ordre n $\geq 2$ à coefficients réels est un anneau non-commutatif (pour le produit matriciel) unitaire (de neutre multiplicatif la matrice identité). Il en est de même de l'anneau des endomorphismes d'un espace vectoriel (pour la loi $\circ$).

3. L'anneau nul est l'anneau 0 formé d'un unique élément
4. Pour tout intervalle I de R, l'ensemble $F($ I, $\mathbb{R}$ $)$ des applications de I dans $\mathbb{R}$ est un anneau commutatif (la multiplication étant le produit des fonctions défini par (fg)(x) = f(x)g(x) pour tout x $\in$ R) unitaire (de neutre multiplicatif la fonction constante égale à 1). Il en est de même pour l'ensemble $\mathbb{R}^{\mathbb{N}}$ des suites de réels.

### 1.2.3 Corps

[14]

Un corps est un anneau (K, +, $\cdot$ ) avec unité, non nul, où tous les éléments $\neq 0$ sont inversibles pour la multiplication $\cdot$ . Un corps non commutatif est aussi appelé un corps gauche.

Exemple (non commutatif ) : le corps gauche des quaternions :
H=$\{ \begin{pmatrix} a & -\bar{b} \\ b & \bar{a} \end{pmatrix} \} : a, b \in \mathbb{C}$
Exemples commutatifs : $\mathbb{F}p := \mathbb{Z}/p\mathbb{Z}$ (p premier), $\mathbb{Q}, \mathbb{R}, \mathbb{C}, \mathbb{Q}(\sqrt{2}), \mathbb{Q}(i), \mathbb{C}(X, Y), \mathbb{C}(T)$,
$\mathbb{C}((T)) = \sum_{n \geq n_0} a_n T : n_0 \in \mathbb{Z}, \forall n \geq n_0, a_n \in \mathbb{C}$ ,
$\mathbb{Z}[i]/7, \mathbb{Z}[\sqrt{2}]/3$,
$\{ \begin{pmatrix} a & 2b \\ b & a \end{pmatrix} \} : b \in \mathbb{F}_5$ sont des corps finis à 49, 9 et 25 éléments

## 1.3 Généralité sur la cryptographie

### 1.3.1 Cryptosystème

Un cryptosystème est un terme utilisé en cryptographie pour désigner un ensemble composé d'algorithmes cryptographiques et de tous les textes en clair, textes chiffrés et clés possibles (définition de Bruce Schneier).

Cette dénomination est toutefois ambiguë, car très souvent associée à la cryptographie asymétrique avec l'utilisation d'une clé privée et d'une clé publique pour les opérations de chiffrement et de déchiffrement.

Le RFC 28281 déconseille fortement l'utilisation de ce terme pour remplacer « système cryptographique » [4]

**L'exemple le plus connu de cryptosystème est RSA :**

Le cryptosystème RSA est basé sur le théorème d'Euler , un des résultats les plus connus et fondamentaux de l'arithmétique, qui dit que pour des entiers a et N avec pgcd(a, N) = 1 on a :

$$\alpha^{\varphi(N)} \equiv 1 \pmod{N}$$

où $\varphi$ est l'indicatrice d'Euler.Dans le cas de RSA, N est le produit pq de deux nombres premiers distincts p et q, et

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

Maintenant, si A veut recevoir un message privé, il a besoin d'établir un couple de clé publique–privée comme suit : Choisir deux grands nombres premiers p et q et prendre N = pq. Choisir un exposant e avec $gcd(e, \varphi(N)) = 1$. Calculer l'exposant de déchiffrement d tel que ed $\equiv 1 \pmod{\varphi)(N)}$ . Puis A publie (N, e) comme sa clé publique, et garde d comme clé privée, en se débarrassant de p, q et $\varphi(N)$ de manière sûre. L'avantage majeur de ce cryptosystème sur RSA est qu'il se généralise facilement à un groupe abélien arbitraire. En particulier, il a donné naissance à la cryptographie à base de courbes elliptiques

Il en existe bien d'autres cryptosystème, parmi lesquels :

— le cryptosystème de ElGamal ;

— le cryptosystème de Rabin ;

— le cryptosystème de Merkle-Hellman ;

— le cryptosystème de Paillier ;

— les cryptosystèmes à courbe elliptique.

### 1.3.2 Modélisation d'un cryptosystème

Mathématiquement un procédé (ou algorithme) de chiffrement, aussi appelé cryptosystème ou encore système de chiffrement, se présente comme la donnée de : M, C ,K, e, d, respectivement appelés ensemble des messages clairs, des messages chiffrés , des clefs secrètes et

$e : M \times K \longrightarrow C$, une fonction de chiffrement et

$d : C \times K \longrightarrow M$ une fonction de déchiffrement

vérifient : $\forall K_e \in K, \exists K_d \in K, \forall m \in M, d(e(m, K_e), K_d) = m$

Il existe deux grandes classes de cryptosystèmes discriminées par la manière dont est géré le secret sur $E_k et D_k$ :

— Les cryptosystèmes à clef secrète ou symétriques ou conventionnels dans lesquels la clé secrète k n'est connue qu'aux interlocuteurs. Les fonctions $e_k etd_k$ sont alors des ressources secrètes partagées par les deux interlocuteurs si $k_e = k_d$ .
— L'autre classe de procédés de chiffrement est celle des cryptosystèmes à clé publique ou asymétriques si connaissant $k_e$ il est calculatoirement très difficile de déterminer $k_d$.

## 1.4 Les objectifs de la sécurité informatique

La sécurité informatique repose sur ces principes fondamentaux (ou services) : l'authentification, la confidentialité, l'intégrité, la non-répudiation et la disponibilité.[10]

### 1.4.1 Authentification

— L'authentification est la procédure qui consiste, pour un système informatique, à vérifier l'identité d'une entité afin d'autoriser l'accès de cette entité à des ressources (systèmes, réseaux, applications...).,

Elle protège de l'usurpation d'identité

— Les entités à authentifier peuvent être :

une personne,

un programme qui s'exécute (processus),

une machine dans un réseau (serveur ou routeur)

### 1.4.2 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.

### 1.4.3 Intégrité

L'intégrité des données consiste à vérifier qu'elles n'ont pas été altérées accidentellement ou frauduleusement au cours de leur transmission ou de leur stockage.

Utilisation de fonctions de hachage ( MD5, SHA-1, SHA-2, SHA-3 etc ) avec le cacul d'une empreinte (foot-print) du message

### 1.4.4 Non répudiation

Un mécanisme de non-répudiation permet d'empêcher à une personne de nier le fait qu'elle a une personne a effectué une opération (exemple : envoi d'un ration (exemple : envoi d'un message, passage d'une commande). message, passage d'une commande).

Pour assurer la non-répudiation d'un message, on peut, par exemple, utiliser la signature électronique

### 1.4.5 Disponibilité

l'objectif est de garantir l'accès à un service, à une application, au réseau ou à une donnée et que cela soit possible en tout temps, afin de garantir le bon fonctionnement du système d'information. On retrouve ici les principes de sauvegarde, de haute disponibilité et de mise en place d'un plan de continuité (BCP ou DRP en anglais).

## 1.5 Rappel sur la cryptographie

La cryptographie est une branche des mathématiques permettant de créer des preuves mathématiques (à savoir des codes et des chiffres), offrant un très haut niveau de sécurité et pouvant servir à protéger ou à dissimuler une information.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 technique qui consiste à déduire un texte en clair d'un texte chiffré sans posséder la clé de chiffrement.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.

### 1.5.1 Chiffrement/déchiffrement

Le chiffrement est une transformation cryptographique qui transforme un message clair en un message inintelligible (dit message chiffré), afin de cacher la signification du message original aux tierces entités non autorisées à l'utiliser ou le lire [1].

Le déchiffrement est l'opération qui permet de restaurer le message original à partir du message chiffré.

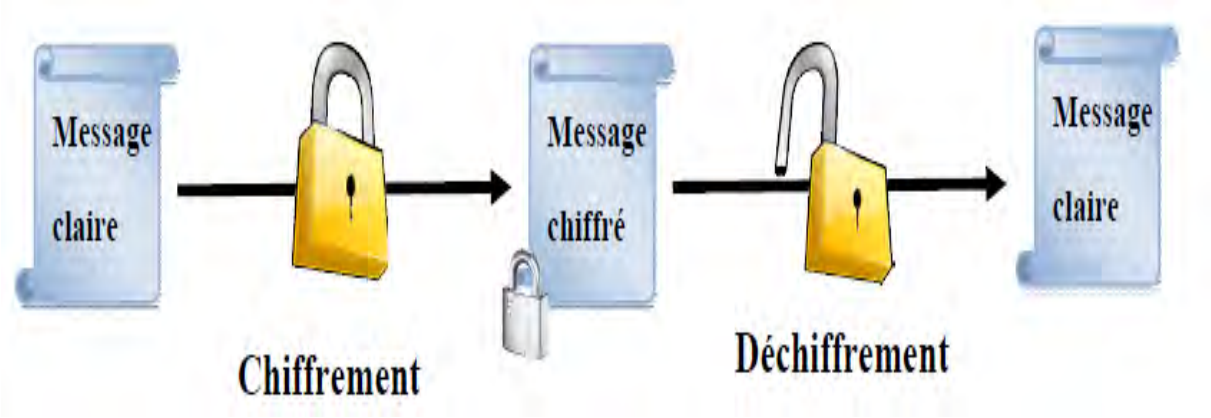



**Figure 1** – Chiffrement/déchiffrement

### 1.5.2 Les clés

Une clé est un paramètre utilisé en entrée d'une opération cryptographique (chiffrement, déchiffrement, scellement, signature numérique, vérification de signature).

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. Il existe deux sortes de clés :

— **Clé Publique :**

La clé publique est créée lors de la génération d'un CSR(Certificate Signing Request) et peut être distribuée au public. Par exemple, une clé publique est utilisée pour crypter des informations que seul le propriétaire de la clé privée est autorisé à recevoir.

— **Clé Privée :**

Une clé privée est créée en convertissant une portion de texte générée automatiquement en un fichier clé à l'aide d'un algorithme mathématique, ce qui lui donne une valeur unique. Ce fichier clé est utilisé pour générer un CSR, et ensuite pour créer un certificat SSL.

Bien que la clé publique et la clé privée soient liées, il est très difficile de dé-

duire 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.

### 1.5.3 Les familles cryptographiques

1. **Cryptographie symétrique (à clé secrète) :**

   Dans le chiffrement symétrique, une même clé est partagée entre l'émetteur et le récepteur. Cette clé dite symétrique est utilisée par l'émetteur pour chiffrer le message et par le récepteur pour le déchiffrer en utilisant un algorithme de chiffrement symétrique.

   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 cryptographie symétrique est très utilisée et se caractérise par une grande rapidité (cryptage à la volée, "on-the-fly"), des implémentations aussi bien software (Krypto Zone, firewalls logiciels type Firewall-1 et VPN-1 de Checkpoint) que hardware (cartes dédiées, processeurs cryptos 8 à 32 bits, algorithmes cablés...) ce qui accélère nettement les débits et autorise son utilisation massive.

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

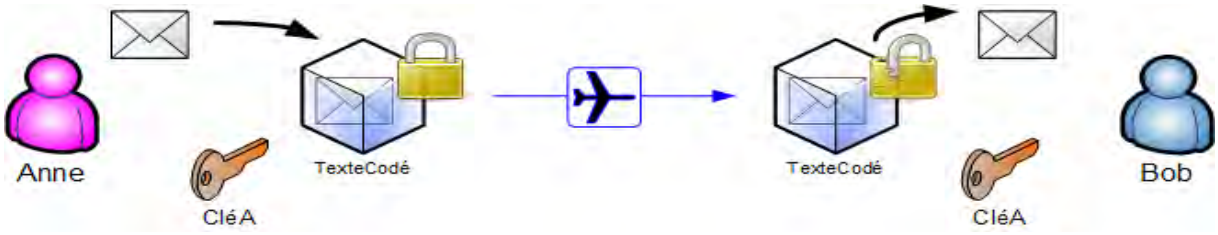



**Figure 2** – Chiffrement à clé secrète

Il existe plusieurs algorithmes qui fonctionnent sur ce principe :

| Algorithme | Description |
|---|---|
| DES (Data Encryption Standard) | Conçu par IBM, l'algorithme DES transforme un bloc de 64 bits en un autre bloc de 64 bits. Il manipule des clés individuelles de 56 bits, représentées par 64 bits (avec un bit de chaque octet servant pour le contrôle de parité). Ce système de chiffrement symétrique fait partie de la famille des chiffrements itératifs par blocs, plus particulièrement il s'agit d'un schéma de Feistel (du nom de Horst Feistel à l'origine du chiffrement Lucifer). 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 ),1991 | Conçu par Xuejia Lai et James Massey, il manipule des blocs de texte en clair de 64 bits. Une clé de chiffrement longue de 128 bits (qui doit être choisie aléatoirement) est utilisée pour le chiffrement des données. L'algorithme a été rendu public. |
| AES ( Advanced Encryption Standard ),2000 | Conçu par J. Daemen et V. Rijmen,l'algorithme prend en entrée un bloc de 128 bits (16 octets), la clé fait 128, 192 ou 256 bits. Il s'agit du standard de chiffrement pour les communications gouvernementales non classées secrètes. L'algorithme a été rendu public. De plus, son utilisation est très pratique car il consomme peu de mémoire et n'étant pas basé sur un schéma de Feistel, sa complexité est moindre et il est plus facile à mettre en œuvre. |

**Table 1** – Algorithme symétrique

2. **Cryptographie asymétrique (à clé publique) :**

   La cryptographie à clé publique, ou cryptographie asymétrique, est une méthode de chiffrement qui utilise deux clés qui se ressemblent mathématiquement mais qui ne sont pas identiques : une clé publique et une clé privée.[**W5**]

   Dans un système asymétrique, le récepteur génère une paire de clés asymétrique : une clé publique qui est diffusée à tout le monde et une clé privée maintenue secrète chez le récepteur. La particularité de cette paire de clé est que tout message chiffrée avec la clé publique ne peut être déchiffré qu'avec la clé privée correspondante.

   D'où la confidentialité des messages chiffré avec la clé publique d'un récepteur. Bien évidemment la clé privée correspondante ne peut être calculée à partir de la clé publique correspondante.

   Puisque les clés publiques doivent être partagées mais sont souvent trop longues pour s'en souvenir, elles sont conservées directement au sein du certificat numérique, ce qui permet de les transporter et de les partager de façon sécurisée.

   Les clés privées, elles, ne devant pas être partagées, sont conservées dans le logiciel ou le système d'exploitation utilisés, sur un appareil de type clé USB cryptographique ou module HSM sur lequel le driver permettant d'utiliser la clé aura été installé.

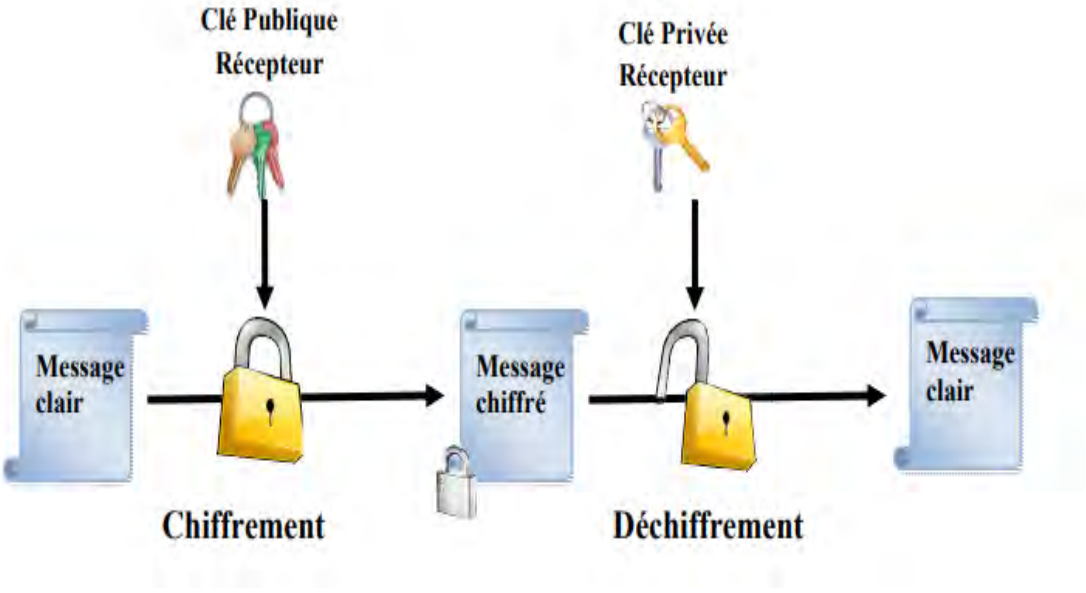



**Figure 3** – Chiffrement à clé publique

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 Whitfield Diffie et Martin E. Hellman,il est également connu sous le nom d'échange de clés exponentiel. La sécurité du schéma de DiffieHellman repose sur la difficulté de calculer un logarithme discret. Si une tierce partie interceptait l'échange, il lui serait difficile en termes de puissance de calcul de déterminer la clé secrète. En pratique, si de grands nombres sont utilisés, cette action demanderait d'énormes ressources aux supercalculateurs modernes pour obtenir le résultat dans un délai raisonnable. L'algorithme a été rendu public. |
| RSA (Rivest Shamir Adleman), 1978 | Conçu par Ron Rivest, Adi Shamir et Leonard Adleman du Massachusetts Institute of Technology. Avec la cryptographie RSA, la clé publique comme la clé privée peuvent chiffrer un message. La clé opposée à celle utilisée pour chiffrer un message sert à le déchiffrer. La sécurité de l'algorithme RSA provient du caractère complexe de la factorisation de grands entiers produits de deux grands nombres entiers. 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. Or c'est sur la difficulté du logarithme discret que s'appuient une partie des primitives cryptographiques utilisées pour l'échange de clé ou la signature |

**Table 2** – Algorithme asymétrique [**W5**]

### 1.5.4 Fonction de hachage

Une fonction de hachage en cryptographie est une fonction publique à sens unique sans trappe (donc facile à calculer et difficile à inverser pour tout le monde), qui transforme un message de longueur quelconque en un message de longueur fixe.

De plus la probabilité, pour qu'il y ait une collusion doit être faible [c'est-à-dire il doit être difficile de trouver x diffèrent de x' tels que h(x) = h(x')] ;

On les utilise en cryptographie pour :

— fournir un condensé de taille fixe ;

— représenter précisément les données : la détection des changements dans le message est simplifiée.

L'intérêt est que cela permet l'usage de la cryptographie asymétrique sans engendrer trop de ralentissement, mais également d'assurer la provenance d'un fichier ainsi que son intégrité. Dans la majorité des cas, elles seront utilisées pour créer une signature numérique. Enfin, la qualité principale de ces fonctions est que les algorithmes sont publics. On parle de "haché", de "résumé", ou de "condensé" pour nommer la caractéristique d'un texte ou de données uniques.

La probabilité d'avoir deux messages avec le même haché doit être extrêmement faible. Le haché ne contient pas assez d'informations en lui-même pour permettre la reconstitution du texte original.

L'objectif est d'être représentatif d'une donnée particulière et bien définie (en l'occurrence le message).

**Propriétés**

Les fonctions de hachage possèdent de nombreuses propriétés :

— Elles peuvent s'appliquer à n'importe quelle longueur de message M.

— Elles produisent un résultat de longueur constante.

— Il doit être facile de calculer h = H(M) pour n'importe quel message M.

— Pour un h donné, il est impossible de trouver x tel que H(x) = h. On parle de propriété à sens unique.

— Pour un x donné, il est impossible de trouver y tel que H(y) = H(x) => résistance faible de collision.

— Il est impossible de trouver x ; y tels que H(y) = H(x) =>résistance forte de collision.

**Fonctions de hachage : Intégrité**

Si M est un message alors pour garantir l'intégrité de M, on envoie ou stocke le couple (M, H(M)) où H(M) est l'empreinte de M via une fonction de hachage H. Le message est considéré intègre s'il est bien accompagné par son empreinte qu'on ne peut falsifier. Les fonctions de hachage sont utilisées pour :

— L'intégrité
— Construire des générateurs aléatoires crypto graphiquement sûrs ;
— Pour la modélisation théorique des fonctions à sens unique tel que le modèle de l'oracle aléatoire.

**Fonction de hachage : Algorithmes**

Les fonctions de hachages comptent deux familles

1. Celles utilisant des clés :
   — MAC (Message Authentication Code)
2. Celles n'utilisant pas de clés
   — MD (Message Digest) :
   — MD4, Rivest, 1990 ;
   — MD5, Rivest, 1992, empreinte sur 128 bits, RFC 1321

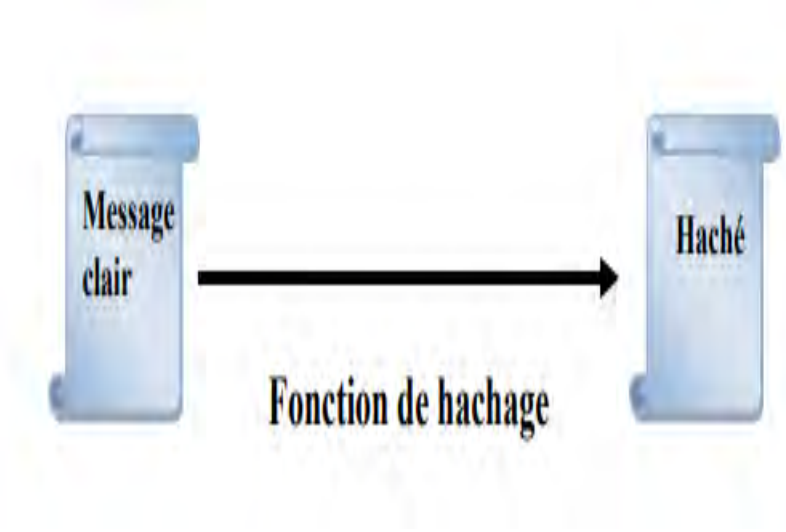



**Figure 4** – Fonction de hachage

### 1.5.5 Signature numérique

La signatures numériques sont les primitives à clé publique de l'authentification des messages. Dans le monde physique, il est courant d'utiliser des signatures manuscrites sur des messages manuscrits ou dactylographiés. Ils sont utilisés pour lier le signataire au message.

De même, une signature numérique est une technique qui lie une personne / entité aux données numériques. Cette liaison peut être vérifiée indépendamment par le récepteur ainsi que par tout tiers.

La signature numérique est une valeur cryptographique calculée à partir des données et d'une clé secrète connue uniquement du signataire.

Dans le monde réel, le destinataire du message a besoin de l'assurance que le message appartient à l'expéditeur et il ne devrait pas être en mesure de répudier l'origine de ce message. Cette exigence est très cruciale dans les applications métier, car la probabilité d'un litige sur les données échangées est très élevée. Il existe deux sortes de signature numérique :

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

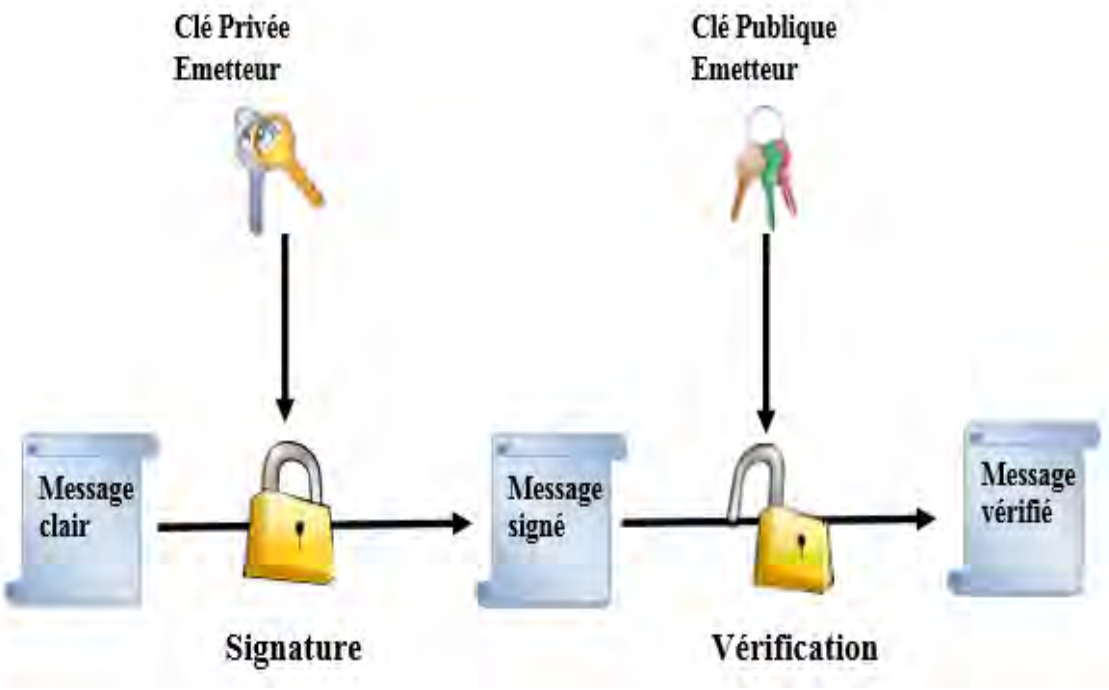



**Figure 5** – Signature numérique simple d'un message

2. **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.

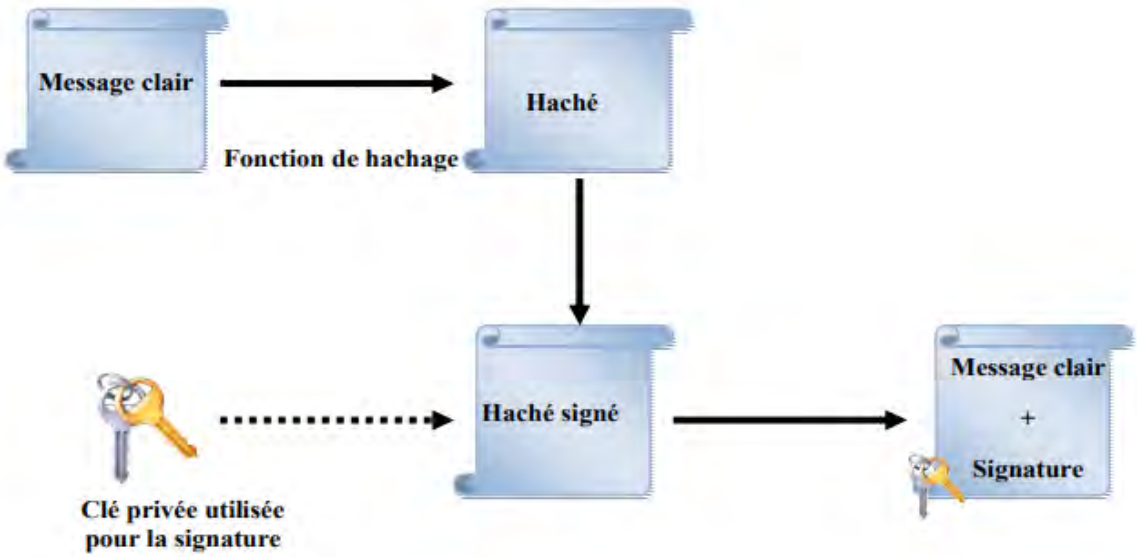



**Figure 6** – Signature numérique sécurisée d'un message

### 1.5.6 Certificats numériques

Les certificats numériques permettent l'identification unique d'une entité ; il s'agit de cartes d'identité électronique émises par des parties dignes de confiance. Les certificats numériques permettent à un utilisateur de vérifier l'identité de la personne à laquelle le certificat est délivré ainsi que l'identité de l'émetteur du certificat.

Les certificats numériques sont le vecteur utilisé par SSL pour la cryptographie de clé publique. La cryptographie de clé publique utilise deux clés cryptographiques différentes : une clé privée et une clé publique. La cryptographie de clé publique est également connue sous le nom de cryptographie asymétrique, car vous pouvez chiffrer les informations avec une clé et les déchiffrer avec la clé complémentaire d'une paire de clés publique-privée donnée.

1. Contenu d'un certificat

   Le certificat contient un certain nombre de champs obligatoires et des extensions dont certaines sont facultatives :

**Version** : indique à quelle version de X509 correspond ce certificat

**Numéro de série** : Numéro de série du certificat

**Algorithme de signature :** identifiant du type de signature utilisée

**Emetteur** : Distinguished Name (DN) de l'autorité de certification qui a émis ce certificat

**Valide à partir de :** la date de début de validité de certificat

**Valide jusqu'à :** la date de fin de validité de certificat

**Objet :** Distinguished Name (DN) de détenteur de la clef publique

**Clé publique :** infos sur la clef publique de ce certificat

**Extensions X509v3 :** Extensions génériques optionnelles, introduites avec la version 3 de X509

**Signature :** Signature numérique de l'AC sur l'ensemble des champs précédents

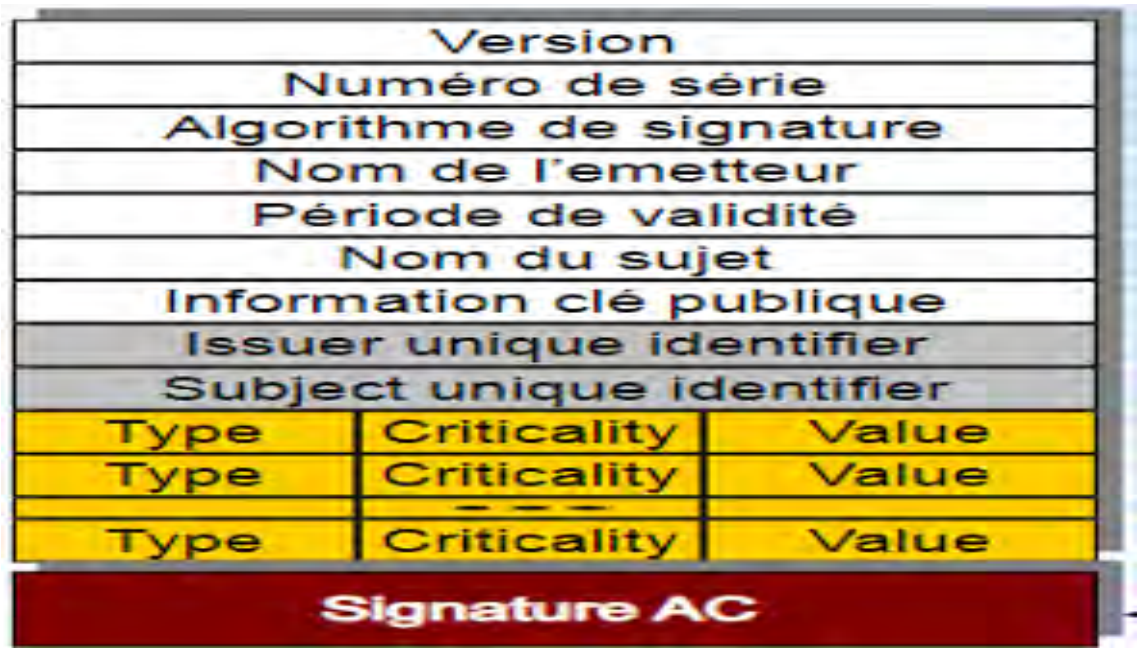



**Figure 7** – format d'un certificat X.509 V3

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

**Version** : 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 de l'émetteur** : 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.

**Nom du sujet** : Ce champ identifie l'identité du propriétaire du couple clés privée/publique à certifier. Il existe là aussi un formalisme pour nommer ce champ.

**Information clé publique** : nom de l'algorithme à clé publique, paramètres concernant la clé.

**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 :

**Issuer unique identifier** : Cette extension fournis un moyen d'identifier la clé publique liée à la clé privée utilisée pour la signature du certificat.

**Subject unique identifier** : Cette extension fournis un moyen d'identifier un certificat contenant une clé publique particulière.

**Signature AC** : 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.

2. Type de certificat

**Certificats SSL / TLS** Les certificats SSL / TLS sont utiles pour sécuriser la transmission de données entre le serveur d'un site Web et les navigateurs des utilisateurs. Ces certificats varient en termes de : .validation - validation de domaine (DV) , validation d'organisation (OV) et validation étendue (EV) .fonctionnalité (ce qu'ils couvrent ou sécurité) - domaines uniques, multi domaines , sous - domaines à un seul niveau (wildcards), et plusieurs sites et sous

- domaines (wildcards multi domaine).

**Certificats d'authentification personnels** Les certificats d'authentification personnels, également appelés certificats de signature d'e-mails et certificats S / MIME, sont utiles pour sécuriser les communications par e-mail en insérant une signature numérique et en utilisant la fonctionnalité de hachage. Ils peuvent également être utilisés pour l'authentification client.

**Certificats de signature de code** Les certificats de signature de code sont parfaits pour protéger les logiciels, pilotes et exécutables téléchargeables. Ces certificats sont disponibles au format organisationnel (OV) ou de validation individuelle (IV), ainsi que la validation étendue et aident les développeurs de logiciels à éviter les avertissements Windows SmartScreen.

**Certificats de signature de documents** Les certificats de signature de documents sont utiles pour sécuriser les documents partagés sur Internet en insérant une signature numérique et en utilisant la fonctionnalité de hachage.

Nous parlerons plus tard de chacun de ces types de certificats et de leurs utilisations individuelles. En attendant, comprenons où les clés de certificat PKI entrent en jeu.

## 1.6 SSL/TLS

### 1.6.1 Définition

Lorsque la prochaine version du protocole a été publiée en 1999, elle a été normalisée par l'IETF (Internet Engineering Task Force) et a reçu un nouveau nom : Transport Layer Security, ou TLS. Comme le note la spécification TLS, «les différences entre ce protocole et SSL 3.0 ne sont pas dramatiques». Ainsi, ce n'est pas vraiment une question de TLS vs SSL ; au contraire, les deux forment une série de protocoles continuellement mis à jour, et sont souvent regroupés en SSL / TLS.

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).

### 1.6.2 Objectifs et moyens mis en œuvre

SSL / TLS proposent les fonctionnalités suivantes :

— Authentification – Le client doit pouvoir s'assurer de l'identité du serveur. Depuis SSL 3.0, le serveur peut aussi demander au client de s'authentifier. Cette fonctionnalité est assurée par l'emploi de certificats.

— Confidentialité – Le client et le serveur doivent avoir l'assurance que leur conversation ne pourra pas être écoutée par un tiers. Cette fonctionnalité est assurée par un algorithme de chiffrement.

— Identification et intégrité – Le client et le serveur doivent pouvoir s'assurer que les messages transmis ne sont ni tronqués ni modifiés (intégrité), qu'ils proviennent bien de l'expéditeur attendu. Ces fonctionnalités sont assurées par la signature des données.

  SSL/TLS reposent donc sur la combinaison de plusieurs concepts cryptographiques, exploitant la fois le chiffrement asymétrique et le chiffrement symétrique.

  SSL / TLS se veulent en outre évolutif, puisque le protocole est indépendant des algorithmes de chiffrement et d'authentification mis en oeuvre dans une transaction. Cela lui permet de s'adapter aux besoins des utilisateurs et aux législations en vigueur. Cela assure de plus une meilleure sécurité, puisque le protocole n'est pas soumis aux évolutions théoriques de la cryptographie (Si un chiffrement devient obsolète, le protocole reste exploitable en choisissant un chiffrement réputé sûr).

### 1.6.3 Fonctionnement

Les protocoles SSL et TLS se décomposent en deux couches principales (quatre en réalité) :

— Handshake :SSL et TLS Handshake Protocol choisit la version de SSL et TLS qui sera utilisée, réalise l'authentification par l'échange de certificats et permet la négociation entre le client et le serveur d'un niveau de sécurité au travers du choix des algorithmes de cryptage. C'est le protocole de configuration de la transaction.

— Record :SSL et TLS Record Protocol encapsule et fragmente les données. C'est le protocole de transmission des données.

— Alert :Ce protocole spécifie les messages d'erreur que peuvent s'envoyer clients et serveurs. Les messages sont composés de deux octets. Le premier est soit warning soit fatal. Si le niveau est fatal, la connexion est abandonnée. Les autres connexions sur la même session ne sont pas coupées mais on ne peut pas en établir de nouvelles. Le deuxième octet donne le code d'erreur.

— 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. »

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

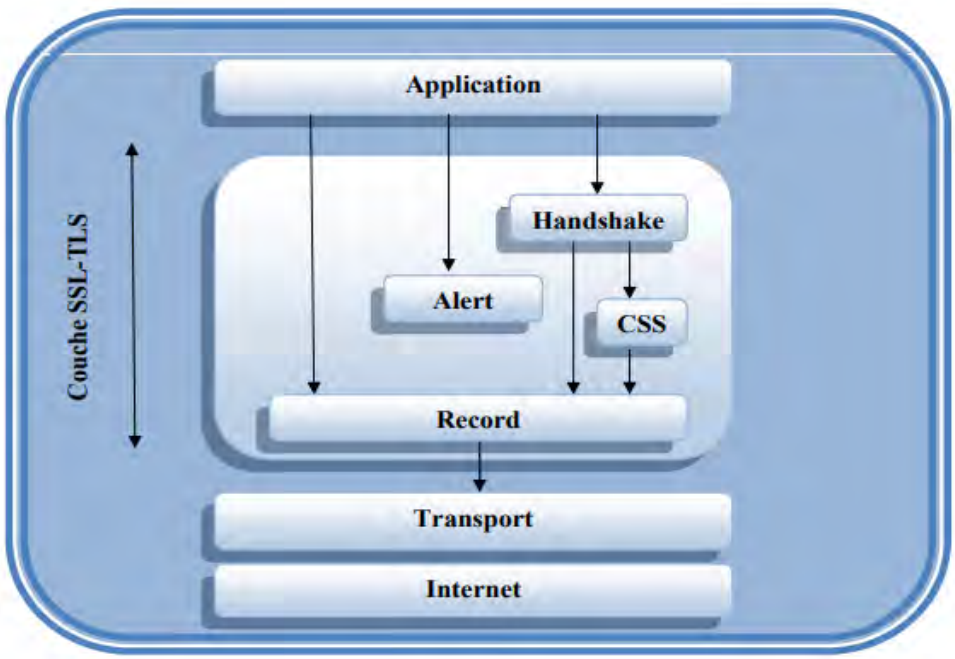



**Figure 8** – Protocole SSL/TLS

### 1.6.4 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. Message « Client Hello »

   Informations que le serveur doit communiquer au client via SSL. Elles incluent le numéro de version SSL, les paramètres de chiffrement, les données spécifiques

de la session.

2. Message « Server Hello »

   Informations que le serveur doit communiquer au client via SSL. Elles incluent le numéro de version SSL, les paramètres de chiffrement, les données spécifiques de la session.

3. Authentification et secret pré-maître

   Le client authentifie le certificat de serveur (par exemple, Nom commun/Date/Émetteur). Le client (en fonction du chiffrement) crée le secret pré-maître pour la session, chiffre avec la clé publique du serveur et envoie le secret pré-maître chiffré au serveur.

4. Déchiffrement et secret maître

   Le serveur utilise sa clé privée pour déchiffrer le secret pré-maître. Le serveur et le client exécutent tous les deux les étapes de génération du secret maître avec le chiffrement convenu.

5. Chiffrement avec la clé de session

   Le client et le serveur échangent des messages pour s'informer mutuellement que les futurs messages seront chiffrés.

### 1.6.5 Implémentations de SSL/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 différentes principes fondamentaux de la sécurité informatique.

# 2 Chapitre II : Infrastructure de gestion de clés publiques(PKI)

**Introduction** :

La PKI peut etre definie comme un dispositif technologique qui permet de créer des Autorités de Certifications (AC) pour identifier les entités. L'autorité de certification a donc pour rôle de délivrer les certificats numériques. Ces derniers permettent d'entreprendre des opérations cryptographiques telles que le chiffrement (ou cryptage), l'authentification et la signature numérique. Ces opérations servent à garantir la confidentialité, l'intégrité et la non-répudiation lors des transactions électroniques.

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.

## 2.1 Notion de PKI

Une PKI assure la sécurité des transactions électroniques et l'échange de renseignements sensibles grâce à des clefs cryptographiques et à des certificats. Une PKI offre divers services : confidentialité, contrôle d'accès, intégrité, authentification, services de non-répudiation pour les transactions commerciales électroniques et les applications informatiques connexes.[**W3**]

En outre, elle gère la production et la distribution des paires de clefs publique et privée, et diffuse la clef publique (ainsi que l'identification de l'utilisateur) sous forme de « certificat » sur des babillards électroniques publics.

Une PKI comprend ce qui suit :

— Autorité de certification ;
— Annuaire de certificats ;
— Autorité Système de révocation de certificats ;
— Système de sauvegarde et de récupération de clefs ;
— Autorité Soutien à la non-répudiation ;
— Mise à jour automatique des clefs ;
— Autorité Gestion de l'historique des clefs ;
— Horodatage ;

— Logiciel client interagissant de manière fiable et continue avec tout ce qui est énuméré ci-dessus.

## 2.2 Les composants d'une PKI

L'infrastructure à clé publique est construite autour d'un ensemble de composants et de procédures de gestion des paires de clés publiques et privées.

Une PKI typique comprend les fonctions suivantes. Ceux-ci peuvent être remplis par une procédure ou un composant technologique et, dans certains cas, par un mélange des deux.

— Autorité de certification (CA) : émet le certificat d'une entité et agit comme un composant de confiance au sein d'une PKI privée. Tout certificat émis par l'autorité de certification est approuvé par toutes les entités qui font confiance à l'autorité de certification. Le rôle exact d'une autorité de certification dépendra de sa position au sein d'une hiérarchie de l'autorité de certification .**[W2]**

— Certificat : Un document numérique, signé par une autorité de certification, et utilisé pour prouver le propriétaire d'une clé publique, dans une PKI. Le certificat a un certain nombre d'attributs, tels que l'utilisation de la clé, l'authentification du client, l'authentification du serveur ou la signature numérique et la clé publique. Le certificat contient également le nom du sujet qui est une information identifiant le propriétaire. Cela peut être, par exemple, un nom DNS ou une adresse IP.

— Autorité d'enregistrement (RA) : Reçoit les demandes de signature de certificat et vérifie l'identité d'une entité finale. L'AE approuvera une demande avant que le certificat ne puisse être émis par l'AC. Il s'agit d'une étape très importante du processus et elle implique souvent une procédure d'inscription des entités finales dans l'ICP.

— Autorité de validation (VA) : Une VA permet à une entité de vérifier qu'un certificat n'a pas été révoqué. Le rôle d'AV est souvent effectué par une installation en ligne hébergée par une organisation qui gère l'ICP. Une autorité de validation utilisera souvent OCSP ou CRL pour annoncer les certificats révoqués.

— Stockage sécurisé : Une méthode de stockage sécurisé d'une clé privée est requise à la fois pour l'autorité de certification (CA) et l'entité finale, afin de protéger la clé contre toute compromission.

— Paire de clés publique / privée - Une clé privée et la clé publique associée sont mathématiquement liées l'une à l'autre. La clé publique peut être largement partagée. La clé privée prouve la propriété de l'identité et doit être gardée secrète.

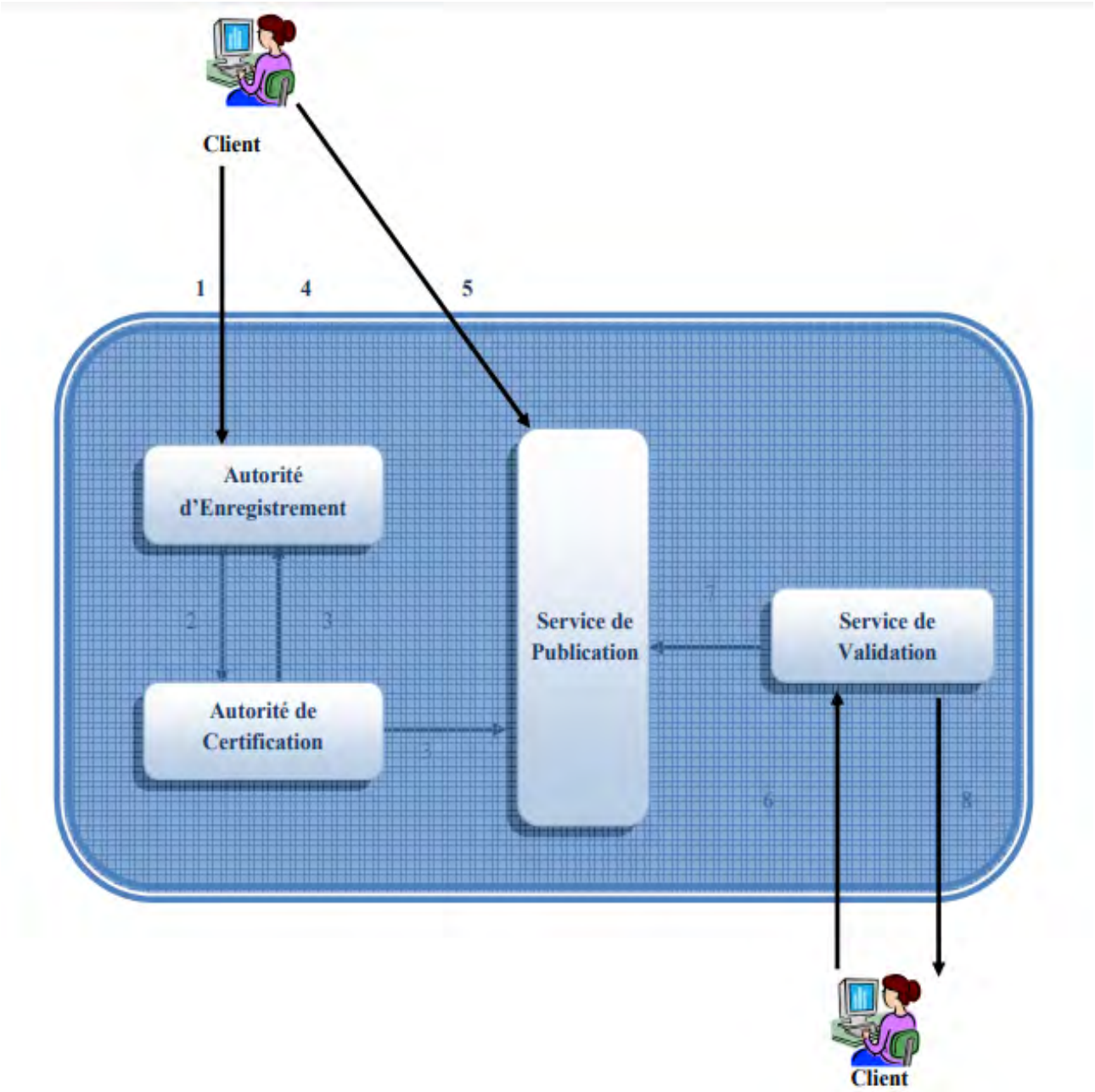



**Figure 9** – Composants et principe de fonctionnement des PKIS

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 :

— **Autorité de certification ;**
— **Annuaire de certificats ;**
— **Autorité Système de révocation de certificats ;**
— **Système de sauvegarde et de récupération de clefs ;**
— **Autorité Soutien à la non-répudiation ;**
— **Mise à jour automatique des clefs ;**
— **Autorité Gestion de l'historique des clefs ;**
— **Horodatage ;**
— **Logiciel client interagissant de manière fiable et continue avec tout ce qui est énuméré ci-dessus.**

## 2.3 Répartition des AC

les autorités de certifications fonctionnent par une chaîne de confiance. Quelles seraient donc les conséquences si une chaîne venait à se briser.

Toutes les autorités de certifications certifiées par l'autorité de certification supérieure seront donc remises en cause car elles n'auraient plus aucun moyen de prouver ses certificats car la communication entre elles serait brisée (figure 1).

Pour parer à ce problème, il existe plusieurs modèles pour structurer les autorités de certifications :[**W8**]

### 2.3.1 Modèle hiérarchique

Modèle hiérarchique : Les autorités CA1 et CA2 ont soumis leurs clefs publiques à un CARoot qui leur a généré un certificat. L'autorité CARoot peut être défini comme le plus haut niveau d'autorité. C'est le seul composant qui ait un certificat auto-signé.

Un certificat auto-signé est le seul certificat qui permette d'assurer l'intégrité et non l'authenticité, d'où la chaîne de confiance. Par conséquence, CA1 et CA2 deviennent des CA subordonnées de CARoot .

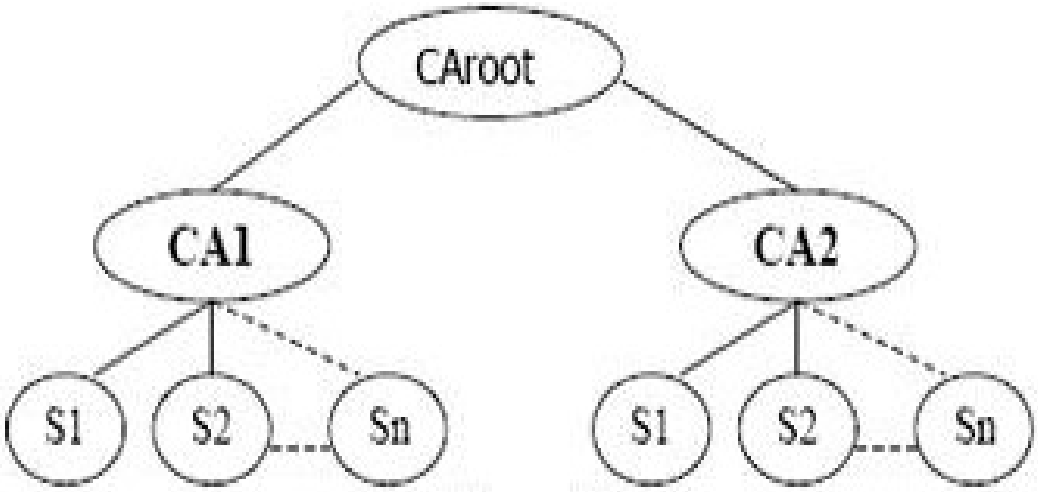



**Figure 10** – Modele hierarchique

### 2.3.2 Modèle croisé (Peer-to-Peer)

Modèle Peer-to-Peer : Le modèle hiérarchique ne règle pas notre problème, c'est pourquoi, il existe le modèle Peer-to-Peer qui permet que différentes autorités de certification soient au même niveau. Si des autorités de certifications sont au même niveau, les certificats qu'elles génèrent sont co-signé, autrement dit, que CA1 peut signer des certificats pour CA2 et vice-versa. Ils sont responsables mutuellement des certificats de leur homologue (figure 3).

Le problème de ce modèle est que toutes les autorités de certifications de même niveau doivent s'échanger leur clef publique pour pouvoir générer des certificats pour leur homologue, de ce fait, plus il y a d'autorités de certification, plus il y aura d'échange de clef publique (pour N autorités de certification il faut générer $N^2$-N/2 certificats pour certifier toutes les autorités).

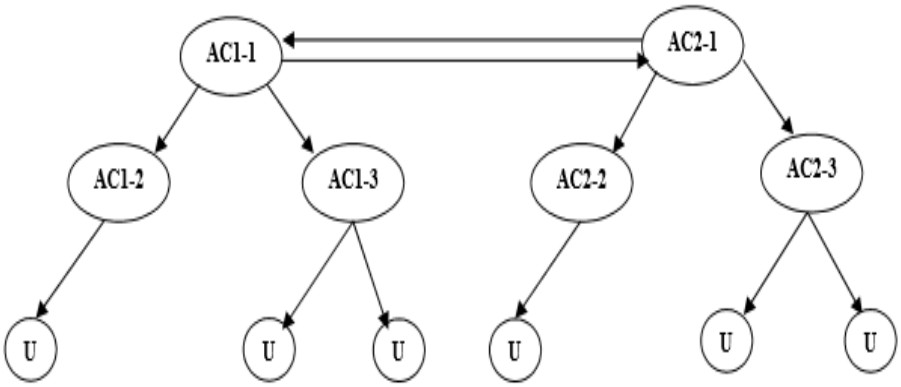



**Figure 11** – Modèle Croisé(Peer-to-Peer)

### 2.3.3 Modèle Bridge

Modèle Bridge : Le modèle en pont ou Bridge est une alternative aux deux autres modèles. En effet, le modèle hiérarchique ne permet pas d'avoir une structure stable et le modèle Peer-to-Peer nécessite un nombre important d'échange entre les autorités. Le modèle en pont ressemble fortement au modèle Peer-to-Peer sauf qu'il permet de limiter les échanges entres les autorités. Le nombre d'échange entre les autorités est réduit car il ne faut plus échanger la clef publique avec toutes les autres autorités mais uniquement avec l'autorité pont .

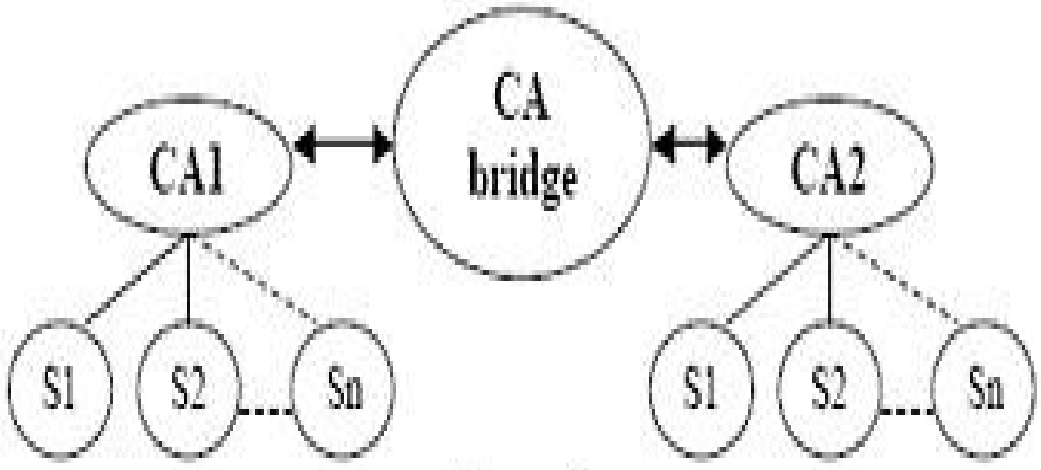



**Figure 12** – Modele bridge

## 2.4 Cycle de vie des clés et des certificats

Sans une gestion de leur cycle de vie, les certificats peuvent se perdre, expirer et entraîner toutes sortes de perturbations imprévues. Les certificats constituent le socle de la sécurité réseau et jouent un rôle clé sur la confiance en ligne et sur les réseaux internes. Il semblerait par conséquent logique de les gérer efficacement, non ?

Pour une bonne gestion de l'ensemble de vos certificats et de leur cycle de vie, le mieux est d'éviter les contrôles manuels. L'utilisation d'un service de gestion du cycle de vie des certificats permet aux administrateurs de surveiller en continu leurs systèmes et certificats numériques, et de pouvoir lancer des audits permettant d'anticiper les expirations et les renouvellements afin d'éviter toute interruption de service.

De nombreuses entreprises font aujourd'hui appel aux fournisseurs de solutions d'infrastructures à clés publiques (PKI) leaders pour gérer leurs certificats tout au long de leur cycle de vie.

## 2.5 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 ensemble de règles, qui fournit un renseignement sur la possibilité d'utiliser un certificat pour une communauté particulière ou des applications ayant des besoins de sécurité communs. Elle spécifie entre autre les conditions et les caractéristiques de délivrance du certificat. Dans le cas général, un certificat est utilisable par n'importe quelle application pour autant que ces conditions et ces caractéristiques sont jugées satisfaisantes. Cependant, une politique de certification peut éventuellement restreindre l'usage du certificat à un ensemble données d'applications, voir même à une seule application.

— **La politique de sécurité** :

La politique de sécurité est la partie juridique de la PKI. En effet, lorsqu'on met en place une PKI, il faut fournir trois documents :

Rapport pratique de certification : qui spécifie 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 permet de responsabiliser les utilisateurs en cas de perte ou de fraude à l'intérieur même de la PKI.

## 2.6 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.

### 2.6.1 CRL

Une liste de révocation de certificats (CRL) est une liste de certificats numériques qui ont été révoqués par l' autorité de certification (CA) émettrice avant leur date d'expiration prévue et qui ne devraient plus être approuvés. Les CRL sont un type de liste noire et sont utilisées par divers points de terminaison, y compris les navigateurs Web , pour vérifier si un certificat est valide et digne de confiance. Un certificat est révoqué de manière irréversible si, par exemple, il est découvert que l'autorité de certification (CA) a émis un certificat de manière incorrecte, ou si une clé privée est supposée avoir été compromise. Les certificats peuvent également être révoqués en cas de non-respect par l'entité identifiée des exigences de la politique, telles que la publication de faux documents, la fausse déclaration du comportement du logiciel ou la violation de toute autre politique spécifiée par l'opérateur CA ou son client. La raison la plus courante de révocation est que l'utilisateur n'est plus en possession exclusive de la clé privée (par exemple, le jeton contenant la clé privée a été perdu ou volé).

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

### 2.6.2 OCSP

Online Certificate Status Protocol (OCSP, en français « protocole de vérification de certificat en ligne ») est un protocole Internet utilisé pour valider un certificat numérique X.509. OCSP est standardisé par l'IETF dans la RFC 69601.

Ce protocole est une alternative réglant certains des problèmes posés par les listes de révocation de certificats (CRL) dans une infrastructure à clés publiques (PKI). Les messages OCSP sont codés en ASN.1 et peuvent être transportés par différents

protocoles applicatifs (SMTP, LDAP, HTTP, etc.). Les communications OCSP étant de la forme « requête/réponse », les serveurs OCSP sont appelés répondeurs OCSP.

— **Avantage par rapport aux CRL :** Plusieurs raisons peuvent amener à préférer le protocole OCSP aux traditionnelles CRL :

OCSP fournit des informations sur le statut du certificat plus à jour ;

avec OCSP, le client n'a plus besoin de récupérer lui-même la CRL. Vu la taille parfois importante de cette CRL, cela allège le trafic réseau ;

le client n'a plus à traiter lui-même la CRL. Cela permet l'économie d'un traitement relativement complexe ; le répondeur OCSP permet de proposer des mécanismes de facturation au vendeur, et non pas à l'acheteur ;

les CRL peuvent être comparées à une « liste de mauvais clients » d'une banque. Cela constitue une fuite d'information non souhaitable.

— **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 suivante illustre comment la requête OCSP est formée et envoyée vers le serveur OCSP.

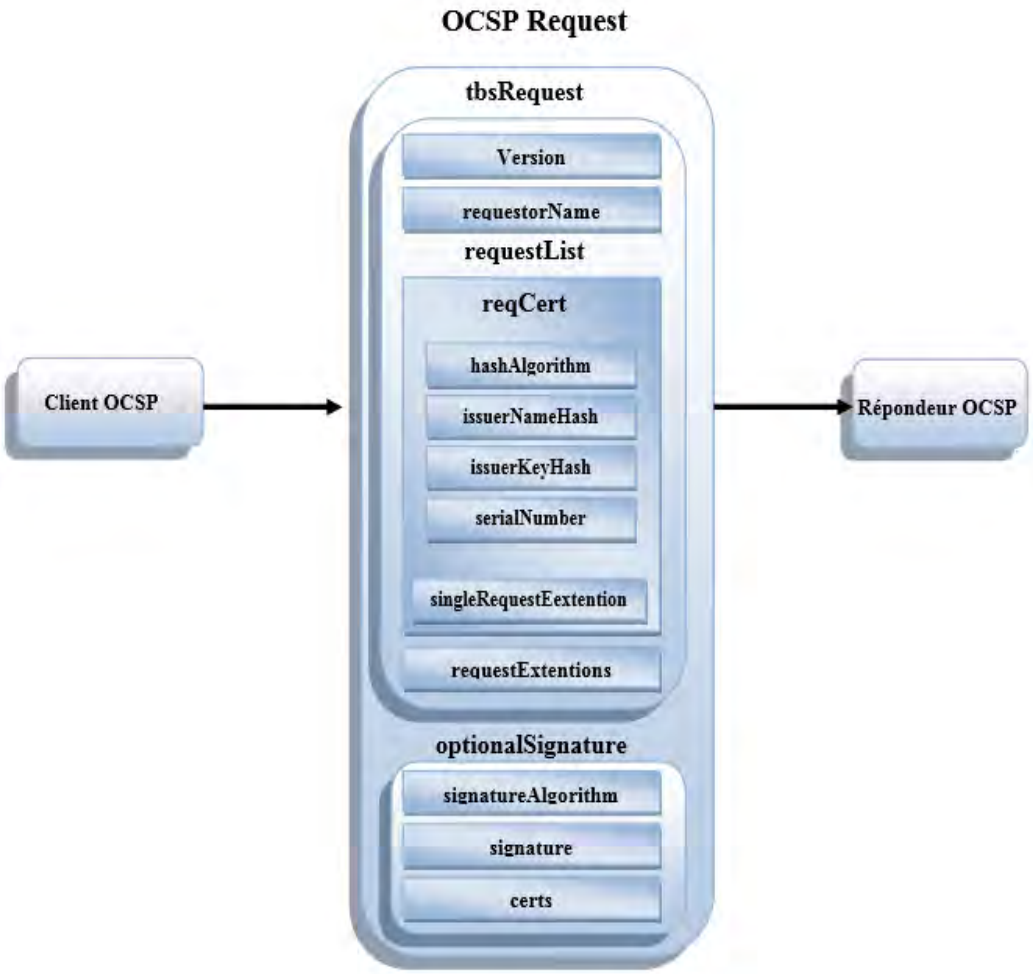



**Figure 13** – Composition d'une requête OCSP

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.

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-àdire 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.

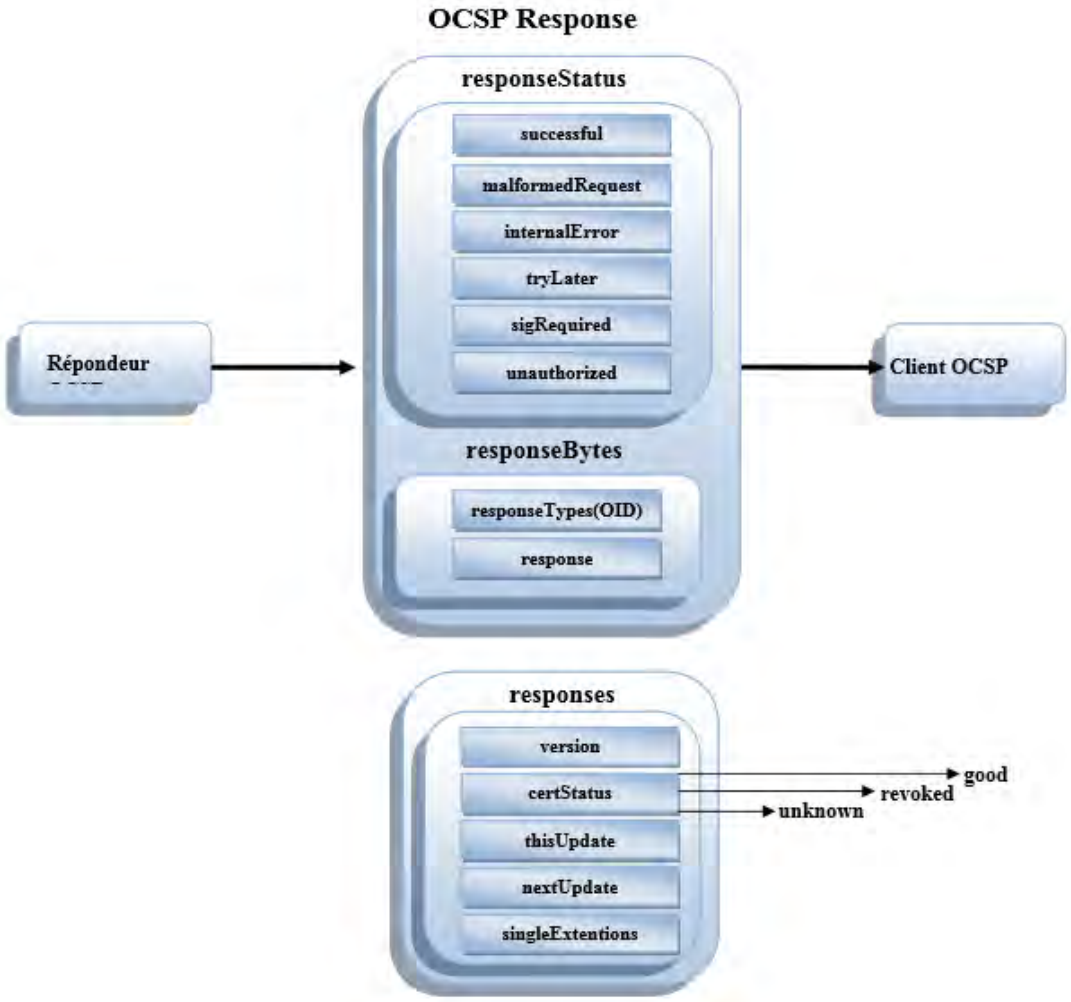



**Figure 14** – Composition d'une réponse OCSP

— **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 [RFC 2560].

### 2.6.3 CMP

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 [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.

## 2.7 Annuaire

### 2.7.1 Définition

Les certificats émis par une ICP doivent être rendus publiques afin que les différents partenaires qui les utilisent puissent s'échanger leur clé publique. Pour cela, les certificats sont publiés dans un annuaire d'accès libre.

Cet annuaire peut également contenir le certificat de l'AC et les CRLs. Des annuaires LDAP sont généralement utilisés pour cette fonction.

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.

### 2.7.2 Annuaire et PKI

L'annuaire est indépendant de la PKI cependant la PKI en a besoin. Les seules contraintes de l'annuaire sont qu'il doit accepter le protocole X.509 pour le stockage des certificats révoqués et le protocole LDAP .

Son rôle est comme dit précédemment de stocker les certificats révoqués et par la même occasion, les certificats en cours de validité afin d'avoir un accès rapide à ces certificats. De plus, l'annuaire peut stocker les clefs privées des utilisateurs dans le cadre du recouvrement de clef. Sachant que les certificats sont largement distribués, l'annuaire est une solution pour les mettre à disposition.

### 2.7.3 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.

1. X.500, Dans le cadre de l'authentification avec les annuaires, la norme X.509 fait partie du standard OSI X.500 pour les annuaires. Il définit le modèle des données d'un certificat (c'est-à-dire les attributs userCertificate, caCertificate, crossCertificatePair, certificateRevocationList...). Il définit des mécanismes pour l'authentification. Le certificat inclut le DN de l'utilisateur et le DN du CA qui a signé le certificat. Les évolution de la norme X.509 pour satisfaire les exigences de PKIX sont les suivantes : X.509v3(1997)

   Nouveau mécanisme d'extension

   Extensions prédéfinies :

   Des informations sur les clés : ID, utilisation...

   Des informations de politique : certificate policy,...

   Des extensions sur l'utilisateur et la CA : nom alternatif,...

   Des contraintes de chemin de certification X.509v4(2000)

   Vérification des chaînes de certificats avec des CAs de différents domaines et hiérarchies.

   Amélioration dans le domaine des révocations de certificat

   Nouvelles caractéristiques dans les certificats en attribut

   Définition de l'usage des certificats en attribut pour le contrôle d'accès et l'autorisation d'accès

2. LDAP (Lightweight Directory Access Protocol) Lightweight Directory Access Protocol (LDAP) est à l'origine un protocole permettant l'interrogation et la modification des services d'annuaire (il est une évolution du protocole DAP). Ce protocole repose sur TCP/IP. Il a cependant évolué pour représenter une norme pour les systèmes d'annuaires, incluant un modèle de données, un modèle de nommage, un modèle fonctionnel basé sur le protocole LDAP, un modèle de sécurité et un modèle de réplication. C'est une structure arborescente dont chacun des nœuds est constitué d'attributs associés à leurs valeurs. LDAP est moins complexe que le modèle X.500 édicté par l'UIT-T.

LDAP a été initialement conçu pour accéder de manière légère aux annuaires X.500. Ces annuaires étaient traditionnellement interrogés à travers le protocole X.500 Directory Access Protocol (DAP) qui nécessitait l'utilisation de la pile de protocoles du modèle OSI. L'utilisation d'une passerelle LDAP/DAP permettait d'accéder à un serveur DAP en étant sur un réseau TCP/IP. Ce modèle d'annuaire est dérivé de DIXIE et de Directory Assistance Service. 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.

La communication LDAP s'appuie sur le protocole TCP, et de manière optionnelle sur TLS

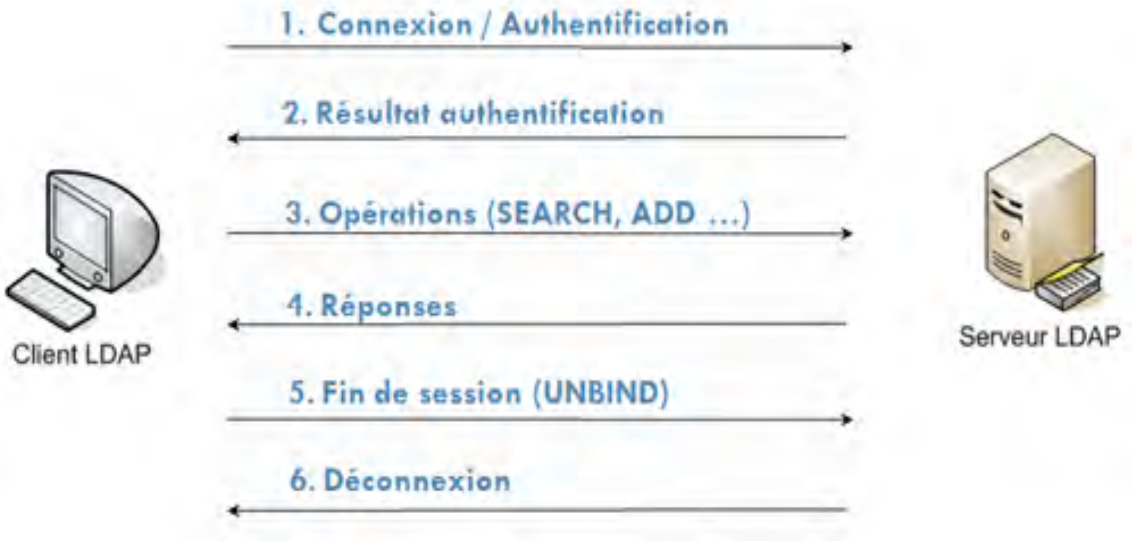



**Figure 15** – Fonctionnement du protocole LDAP

**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.

# 3 Chapitre III : Mise en œuvre d'une PKI(EJBCA)

**Introduction** :

Toutes structures informatisés,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 recherche, de réaliser une infrastructure de gestion de clés publiques qui s'inscrit dans le cadre de la mise en place d'une configuration qui s'occupe de la création des certificats, la validation et la publication de ces derniers.

## 3.1 Présentation du projet

Notre projet consiste en la mise en oeuvre d'une infrastructure de gestion de clés publique (PKI) .

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

**OpenCA** est aujourd'hui la composante d'un vaste projet communautaire, visant à définir les standards de développement d'un logiciel de PKI. OpenCA PKI en est la partie dédiée à la gestion des certificats.

**EasyCA** permet de gérer très rapidement et sans fioriture une PKI de petite taille (moins d'une cinquantaine de certificats).

Développé par Ferry Kemps en 2005, easyCA permet de s'abstraire quasi-totalement de la complexité relative d'OpenSSL en permettant de créer très vite ses autorités de certification ainsi que ses certificats Client. Il permet en outre la gestion des révocations et propose des options d'export pour sauvegarde.

**OpenSSL** est la librairie open source, quasiment élevée au rang de standard sous UNIX, pour ce qui concerne les fonctions cryptographiques et les fonctions de hachage. En particulier, elle implémente quasi-complètement le standard des PKI, i.e la norme X509. Le développement d'OpenSSL est ancien et a débuté avant 1998. Il est sous double licence Apache et BSD.

**EJBCA** est une solution open source de gestion d'une PKI (infrastructure de clés publiques), parmi les plus complètes qui soient. Elle est actuellement portée et

maintenue activement par la société suédoise Primekey. A l'instar d'autres solutions de PKI, EJBCA permet non seulement de gérer tous les aspects de la certification courante X509 (émission de certificats, révocations avec CRL, chaînes de certifications) mais fait partie des seuls produits, et c'est là son grand avantage, à implémenter une grande partie des standards liés à la spécification X509 (répondeur OCSP, CMS...) et gère correctement les matériels spécifiques tels que les HSM. Il propose également une interface d'administration complète avec restrictions des droits ainsi qu'un portail client.[**W7**]

### 3.1.1 Tableau comparatif des PKI open source

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

**Table 3** – Tableau comparatif d'EJBCA avec les autres PKI [**W4**]

### 3.1.2 Présentation du produit EJBCA

1. Définition EJBCA est une PKI (Public Key infrastructure) ou IGC (Infrastructure de gestion de clés) sous licence open source (LGPL) développée en Java

JEE.EJBCA bien plus qu'une PKI EJBCA fournit à la fois les fonctions classiques que l'on retrouve dans la plupart des PKI du marché. Mais elle fournit également un serveur OCSP, un serveur d'horodatage et un serveur de signature. Chaque composant peut être déployé de manière autonome ou intégrée dans EJBCA. EJBCA un générateur de vos besoins. Il a été construit comme un générateur, permettant de répondre rapidement et sans développement complémentaire aux contraintes liées aux certificats ou au SI de l'entreprise. Le générateur permet de créer des profils de certificat, des formats de requête, de personnaliser des cartes à puce ou de tokens USB.

2. Architecture d'EJBCA

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

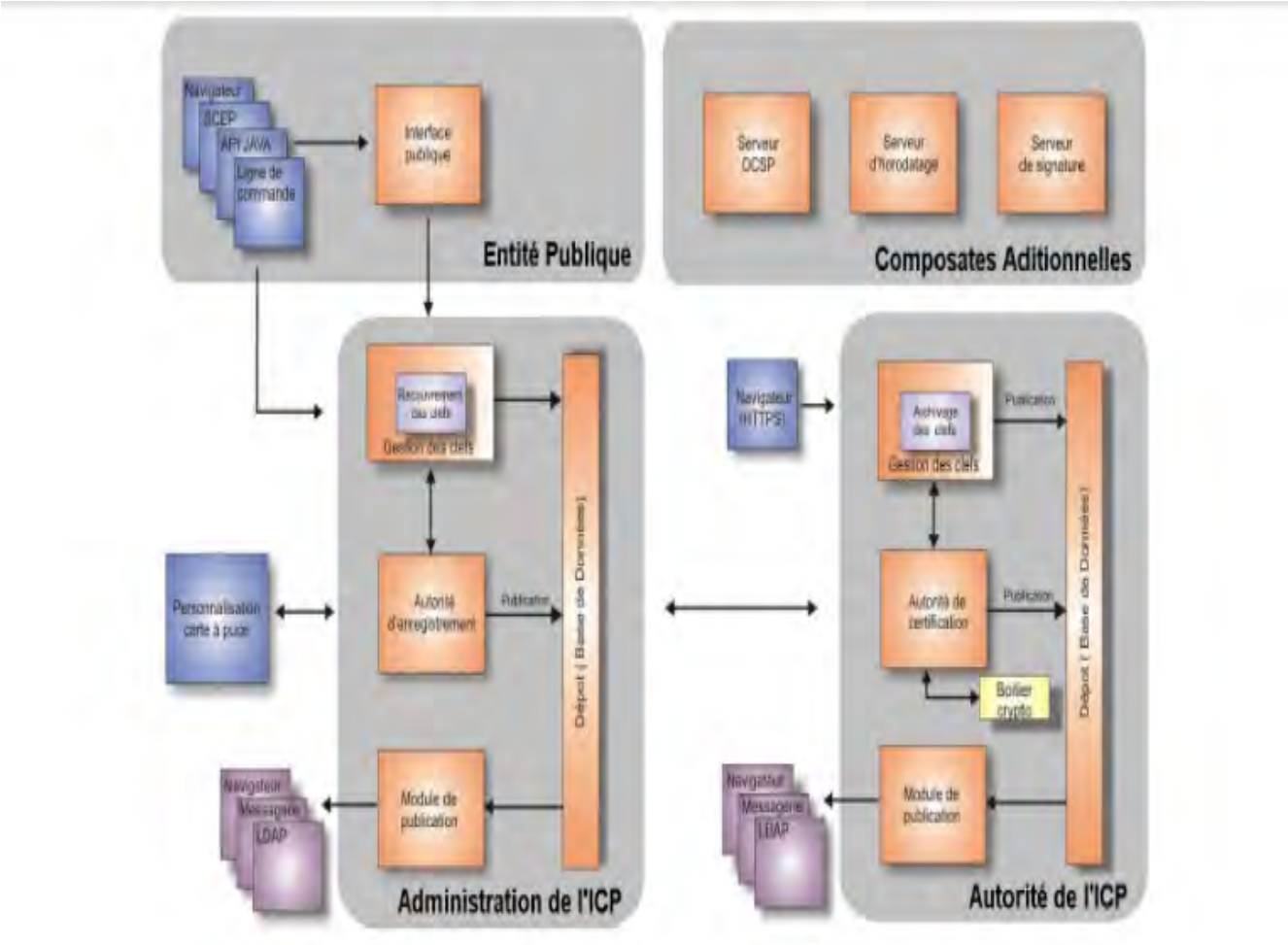



**Figure 16** – Architecture fonctionnelle d'EJBCA

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.

3. Caractéristiques techniques ;
   Les caractéristiques techniques d' EJBCA sont illustrées dans le tableau suivant :

| Caracteristiques technique | Description |
|---|---|
| Éditeur | PrimeKey Solutions et contributeurs externes |
| Solution | EJBCA |
| Version utilisé | 4.0.10 |
| Référence | https ://www.ejbca.org/ |
| Plateformes supportées | Plateformes Java™ J2EE .<br>Serveurs d'applications :<br>JBoss ; Weblogic ; Glassfish ; WildFly ; OC4J ; Websphere |
| Système d'exploitation supporté | Unix, Linux, Microsoft |
| Fonctionalité | • Multiple AC et niveau d'AC par instance ;<br>• Génération de certificats de manière individuelle ou pour lot ;<br>• Génération de certificats en mode centralisé et décentralisé ;<br>• Administration par ligne de commande ou interface Web ;<br>• Gestion des profils de certificats ;<br>• Gestion des profils d'entités |
| Protocoles et standards | • Certificat x509v3 et CRL v2 (RFC3280) ;<br>OCSP (protocole de validation - RFC2560) ;<br>CMP (protocole de gestion – RFC4210) ;<br>Horodatage (RFC3261) ;<br>LDAP v3. |
| Base de données | Hypersonic, Mysql, PostGresql, Oracle, MSSQL, Sysbase,MariaDb, informix. |
| Entité | • Autorité de certification<br>• Autorité d'enregistrement<br>• Gestionnaire de clé (Séquestre et renouvellement)<br>• Autorité d'enregistrement publique<br>• Serveur OCSP interne ou externe<br>• Gestionnaire de carte à puce et dongle USB<br>• Service de journalisation<br>• Service de publication LDAP et courriel |
| Support cryptographique | RSA (jusqu'à 4096-bits)<br>Elliptic Curve DSA signing<br>MD5, SHA-1, SHA-256 |
| Caractéristiques de certificats | Certificats X.509 v3<br>Longueur de clés allant jusqu'à 4096-bit pour authentification.<br>Configuration flexible des DN de certificats |
| Annuaire LDAP | Sun directory, OpenLDAP, Active directory, LDAPv3 |

**Table 4** – Caractéristiques technique du produit EJBCA

## 3.2 Mise en oeuvre d'EJBCA

Le Produit EJBCA est offert sous licence LGPL et n'existe que sous une seule édition, celui-ci offre donc toutes les libertés fondamentaux associées au logiciel libre. Un support commercial est contractable auprès de la société éditrice Primekey. 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.

### 3.2.1 Description de l'environnement

Pour mettre en œuvre notre PKI, nous avons eu le besoin de mettre en place deux machines : une machine d'administration EJBCA 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 Table 5.

### 3.2.2 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.

| LOGICIELS | VERSION | DESCRIPTION |
|---|---|---|
| EJBCA | 4.0.10 | autorité de certification |
| JBOSS | 5.1.0-GA | serveur d'applications Java EE Libre écrit en Java fournissant les services nécessaires au fonctionnement d'EJBCA |
| Ant | 1.8.2 | Bibliothèque Java qui aide la compilation d'application java |
| Java développement kit | Jdk6 | Ensemble d'outils avec lesquels le code Java peut être compilé |

**Table 5** – les logiciels requis pour la mise en place d'une PKI avec EJBCA

— **Linux**

Linux ou GNU/Linux est une famille de systèmes d'exploitation open source de type Unix fondé sur le noyau Linux, créé en 1991 par Linus Torvalds. 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.

Nous avons choisie Ubuntu, un système d'exploitation1 libre, gratuit, sécurisé et convivial, qui peut facilement remplacer ou cohabiter avec votre système actuel (Windows, macOS, autre GNU/Linux, ...). 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 3.1.2. 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.

— **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.

Ant pourrait être comparé au célèbre outil make sous Unix. Il a été développé pour fournir un outil de construction indépendant de toute plate-forme. Ceci est particulièrement utile pour des projets développés sur et pour plusieurs systèmes ou, pour migrer des projets d'un système sur un autre. Il est aussi très efficace pour de petits développements.

— **OpenJDK :**

Le JDK (Java Development Kit) permet de développer et d'exécuter des applica-

tions 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.

— **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.

### 3.2.3 Installation

Dans cette partie nous allons citer les étapes essentielles pour l'installation de l'autorité de certification d'EJBCA .

**NB** :Les sorties terminales sont représentées ainsi :

$ : commande1

# ou (sudo devant $) : commande2

Les commandes à exécuter avec des droits utilisateurs sont précédées du caractère « $ » tandis que celles à exécuter les droits root sont précédées du caractère «# » ou sudo devant $

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

1. Installation des programmes ant, jdk, unzip et ntp : Après avoir démarré votre machine ayant pour système d'exploitation Ubuntu 14.04, ouvrez un nouveau terminal. Dans ce terminal exécutez les commandes :

   **sudo apt-get install openjdk-6-jdk**

```
fama@ubuntu:~$ sudo apt-get install openjdk-6-jdk
```

   L'outil Ant du projet Apache est nécessaire à la compilation d'EJBCA :

   **sudo apt-get install ant-optional**

```
fama@ubuntu:~$ sudo apt-get install ant ant-optional
```

   Unzip qui sera utilisé ultérieurement pour désarchiver et ntp pour synchroniser, via un réseau informatique, l'horloge locale d'ordinateurs sur une référence d'heure. Il trouve son importance lorsque l'on veut à distance accéder au serveur où EJBCA est installé.

```
fama@ubuntu:~$ sudo apt-get install unzip
```

```
fama@ubuntu:~$ sudo apt-get install ntp
```

2. Téléchargement d'EJBCA et de JBOSS Toujours dans le nouveau terminal exécutez les commandes suivantes dans l'ordre :

   cd : Cette commande vous permettra de vous déplacer dans le répertoire de l'utilisateur connecté,

   wget `http://sourceforge.net/projects/jboss/files/JBoss/JBoss-5.1.0.GA/jboss-5.1.0.GA-jdk6.zip` : Cette commande vous permettra de télécharger le logiciel JBOSS décris dans les prérequis,

   unzip jboss-5.1.0.GA-jdk6.zip : Cette commande désarchivera le fichier zip JBOSS que vous venez de télécharger, wget `https://sourceforge.net/projects/ejbca/files/ejbca4/ejbca_4_0_10/ejbca_4_0_10.zip` : Cette commande vous permettra de télécharger le logiciel EJBCA décris dans les prérequis,

   unzip ejbca_4_0_10.zip : Cette commande désarchivera le fichier zip EJBCA que vous venez de télécharger,

3. Configuration d'EJBCA : Dans le même terminal, vous allez maintenant configurer EJBCA pour qu'il puisse trouver l'emplacement du serveur d'application JBOSS. Pour ce faire exécutez la commande :

   **echo "appserver.home=/home/(le nom de l'utilisateur connecté)/jboss-5.1.0.GA" » ejbca_4_0_10/conf/ejbca.properties**.

```
fama@ubuntu:~$ echo "appserver.home=/home/fama/jboss.5.1.0.GA
" >> ejbca_4_0_10/conf/ejbca.properties
```

   Toutefois, pour vérifier que la configuration s'est bien déroulée, exécutez la commande :

   nano ejbca_4_0_10/conf/ejbca.properties. Un fichier s'ouvrira dans lequel vous aurez appserver.home=/home/(nom de l'utilisateur connecté)/jboss-5.1.0.GA

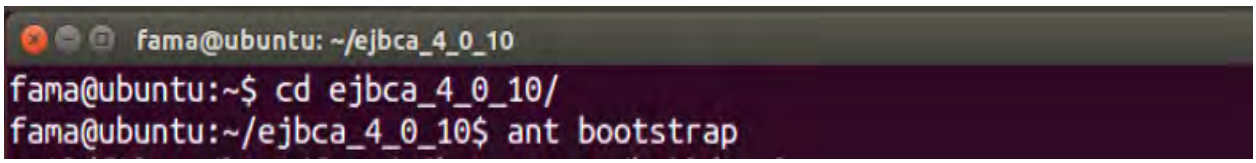
```
  GNU nano 2.2.6    File: conf/ejbca.properties

appserver.home=/home/fama/jboss-5.1.0.GA
```

4. Compilation du programme EJBCA :

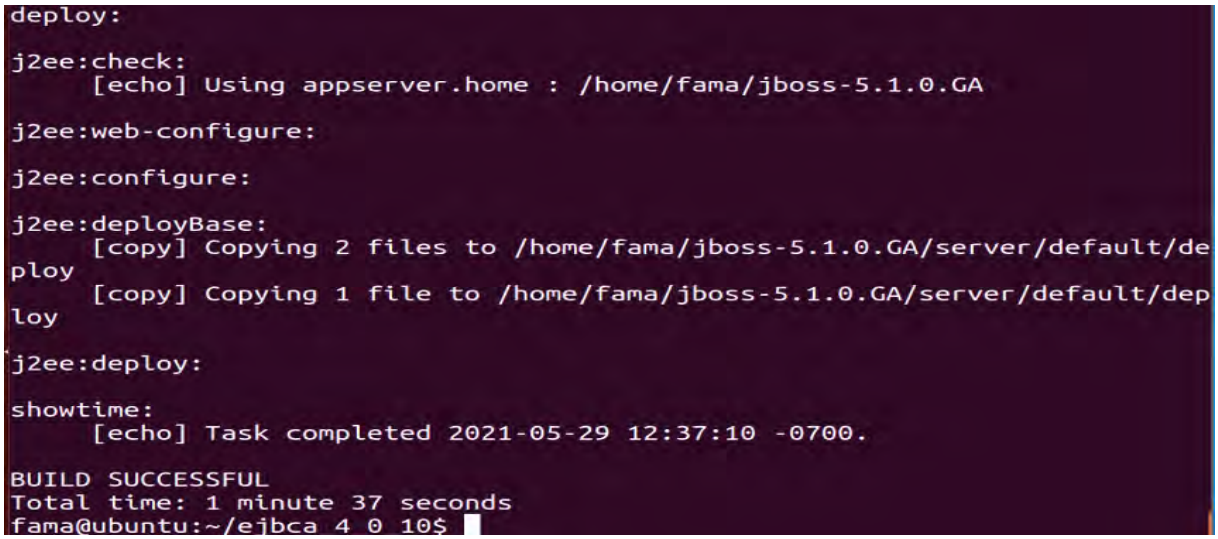
   Dans le même terminal, vous allez compiler le programme EJBCA en exécutant les commandes :

   cd ejbca_4_0_10, pour se déplacer dans le répertoire ejbca_4_0_10,

   ant bootstrap pour compiler,




   Durant la compilation, il vous sera demandé d'entrer le mot de passe pour la base de donnée Appuyer sur la touche entrée. A la fin de la compilation, si tout se passe bien vous verrez normalement le message **BUILD SUCCESSFUL** (voir capture ci-dessous)




5. Démarrage du serveur d'application JBOSS : Pour démarrer JBOSS, ouvrer un nouveau terminal et exécuter la commande : **jboss-5.1.0.GA/bin/run.sh** Patienter un peu jusqu'à voir le message Started in **mn : ss : ms** .

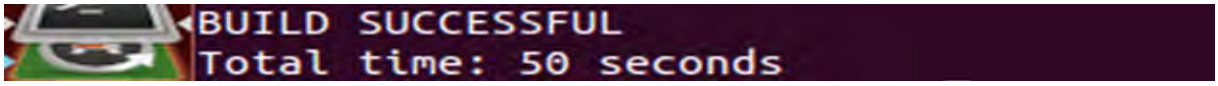

6. Création de l'autorité de certification

```
23:20:59,257 INFO  [Http11Protocol] Starting Coyote HTTP/1.1 on http-0.0.0.0-844
2
23:20:59,353 INFO  [Http11Protocol] Starting Coyote HTTP/1.1 on http-0.0.0.0-844
3
23:20:59,444 INFO  [AjpProtocol] Starting Coyote AJP/1.3 on ajp-127.0.0.1-8009
23:20:59,542 INFO  [ServerImpl] JBoss (Microcontainer) [5.1.0.GA (build: SVNTag=
JBoss_5_1_0_GA date=200905221634)] Started in 5m:29s:687ms
```

Retournez dans le terminal où vous avez compilé le programme EJBCA et exécutez la commande **ant install**. Cette commande permet de générer tous les certificats, clés, etc. nécessaires pour fonctionner avec une autorité de certification initiale. Au cours de la création une série de question vous sera posé. Appuyer sur la touche ENTRER pour choisir les valeurs par défauts. A la fin de la création, si tout se passe bien vous verrez le message **BUILD SUCCESSFUL**.

```
fama@ubuntu:~/ejbca_4_0_10$ ant install
```

et si tout ce passe bien, vous aurez :




7. Déploiement d'EJBCA :

   Dans le même terminal, exécutez la commande **ant deploy** pour déployer et configurer le serveur avec le fichier de magasin de clés. Au cours du déploiement une série de questions vous sera posée Appuyer sur la touche ENTRER pour choisir les valeurs par défauts. A la fin de déploiement, si tout se passe bien vous verrez le message **BUILD SUCCESSFUL**.

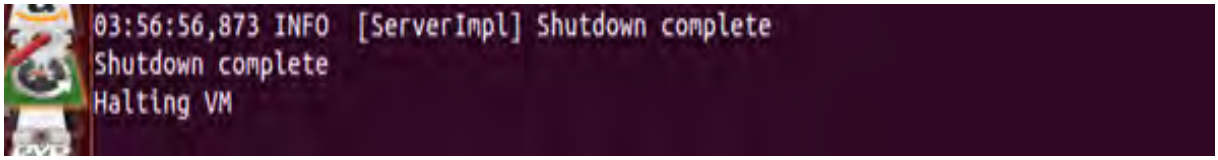
```
fama@ubuntu:~/ejbca_4_0_10$ ant deploy
```

   et si tout ce passe bien, vous aurez :

8. Redémarrage de JBOSS

   Retourner dans le terminal ou vous avez démarré le serveur JBOSS et exécuter les commandes :

   ctrl-c pour arrêter le serveur,


jboss-5.1.0.GA/bin/run.sh pour démarrer le serveur

```
23:20:57,544 INFO  [UniqueSernoHelper] Custom certificate serial number not allo
wed since there is no unique index on (issuerDN,serialNumber) on the 'Certificat
eData' table.
23:20:57,658 INFO  [TomcatDeployment] deploy, ctxPath=/ejbca/clearcache
23:20:57,830 INFO  [TomcatDeployment] deploy, ctxPath=/ejbca/publicweb
23:20:57,953 INFO  [TomcatDeployment] deploy, ctxPath=/ejbca/doc
23:20:58,061 INFO  [TomcatDeployment] deploy, ctxPath=/ejbca/publicweb/healthche
ck
23:20:58,180 INFO  [TomcatDeployment] deploy, ctxPath=/ejbca
23:20:58,323 INFO  [TomcatDeployment] deploy, ctxPath=/ejbca/publicweb/apply
23:20:58,437 INFO  [TomcatDeployment] deploy, ctxPath=/ejbca/publicweb/status
23:20:58,610 INFO  [OCSPServletBase] ExtensionOids not defined.
23:20:58,611 INFO  [OCSPServletBase] ExtensionClass not defined.
23:20:58,785 INFO  [TomcatDeployment] deploy, ctxPath=/ejbca/publicweb/webdist
23:20:59,097 INFO  [Http11Protocol] Starting Coyote HTTP/1.1 on http-0.0.0.0-808
0
23:20:59,257 INFO  [Http11Protocol] Starting Coyote HTTP/1.1 on http-0.0.0.0-844
2
23:20:59,353 INFO  [Http11Protocol] Starting Coyote HTTP/1.1 on http-0.0.0.0-844
3
23:20:59,444 INFO  [AjpProtocol] Starting Coyote AJP/1.3 on ajp-127.0.0.1-8009
23:20:59,542 INFO  [ServerImpl] JBoss (Microcontainer) [5.1.0.GA (build: SVNTag=
JBoss_5_1_0_GA date=200905221634)] Started in 5m:29s:687ms
```

9. Importation du certificat administrateur dans le navigateur

   Fermer tous les terminaux ouverts en excluant celui ou vous avez démarré le serveur JBOSS et ouvrez un nouveau terminal. Exécutez la commande **cp ejbca__4__0__10/p12/superadmin.p12** bureau/ Le certificat administrateur créé durant l'étape 7 sera copié sur le bureau.

   Ouvrez votre navigateur (pour nous mozilla Firefox) et allez dans préférences. Cliquez sur **Private and Security**, puis allez jusqu'en bas. Cliquez sur **view certificates** (voir certificat) Dans l'onglet **your certificate** (votre certificat) cliquer sur **import** puis bureau et sélectionnez **superadmin.p12**

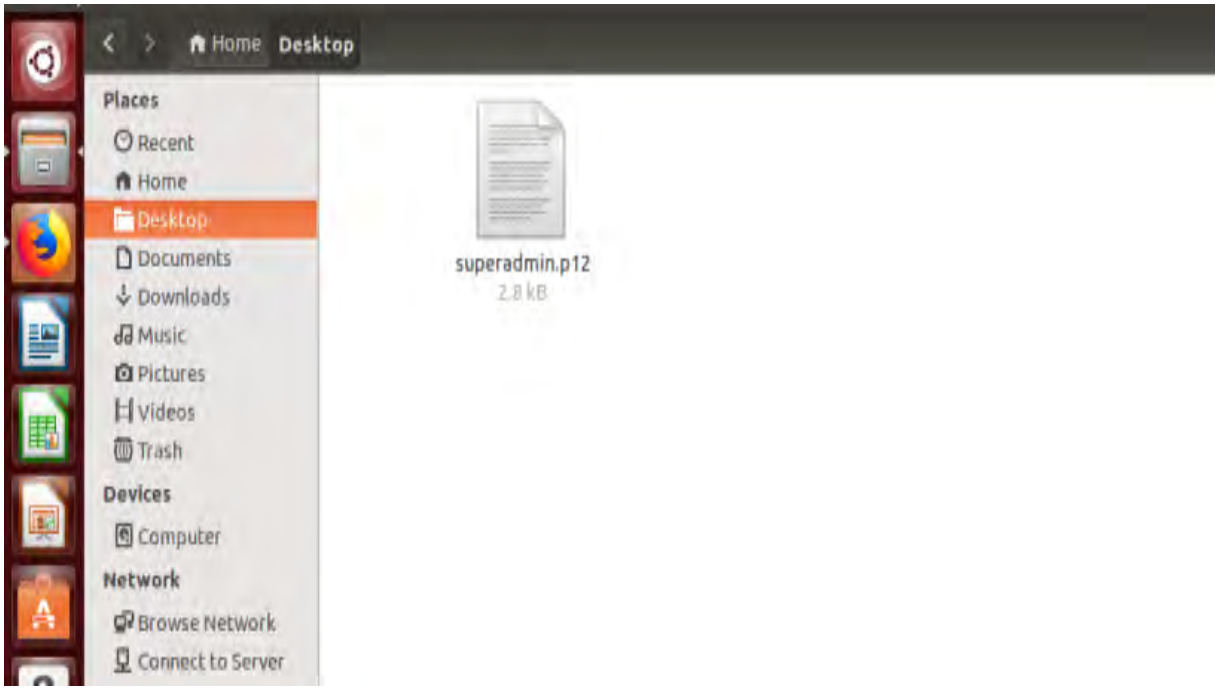
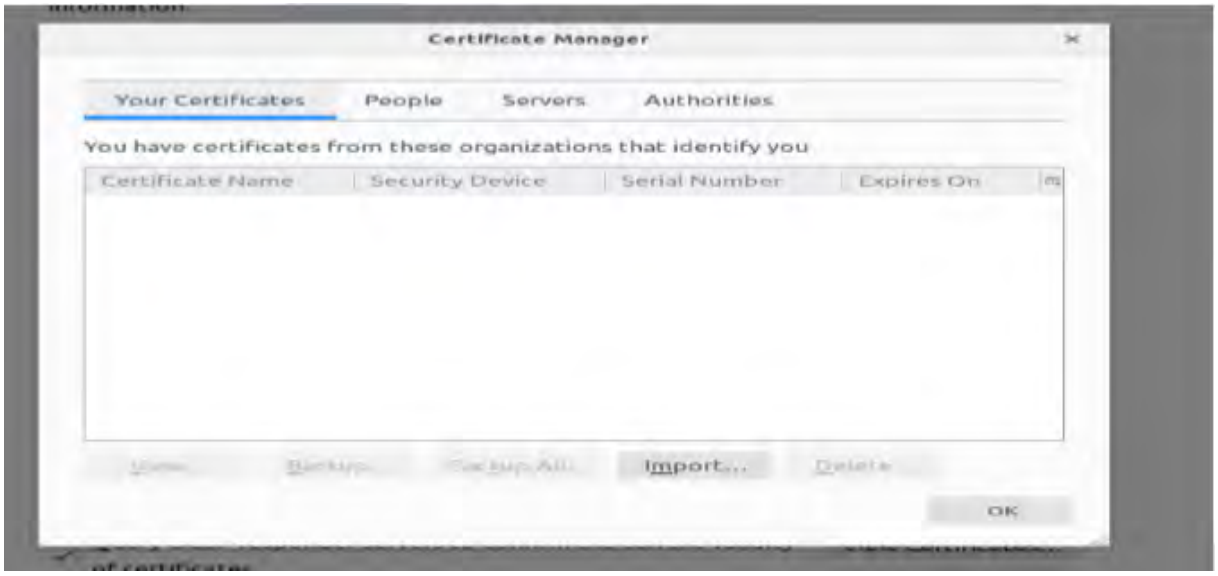



**Figure 17** – Copie du certificat administrateur sur le bureau




10. Importation du certificat dans le navigateur

    Un mot de passe vous sera demandé. Saisissez le mot de passe de votre choix (root pour nous) Ensuite le mot de passe utilisée pour crypter le certificat vous sera demandé. Saisissez ejbca Cliquez sur OK. Dans l'onglet Your certificates vous verrez l'apparition du certificat SuperAdmin

    Cliquez encore sur OK et fermer la fenêtre des préférences.


**Figure 18** – Importation du certificat dans le navigateur

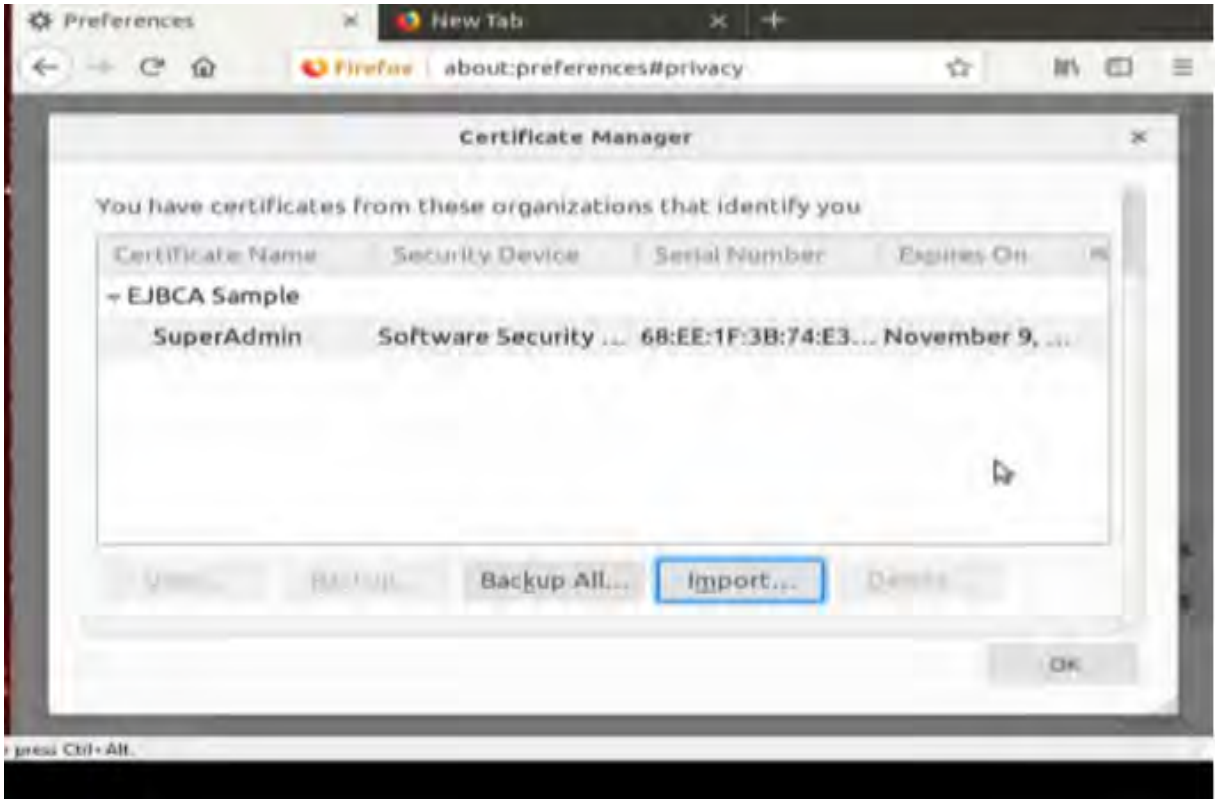



**Figure 19** – Apparition du nouveau certificat

Vous pouvez maintenant accéder sur votre page d'administation d'EJBCA via l'adresse

**https ://adresseipduserveur :8443/ejbca**

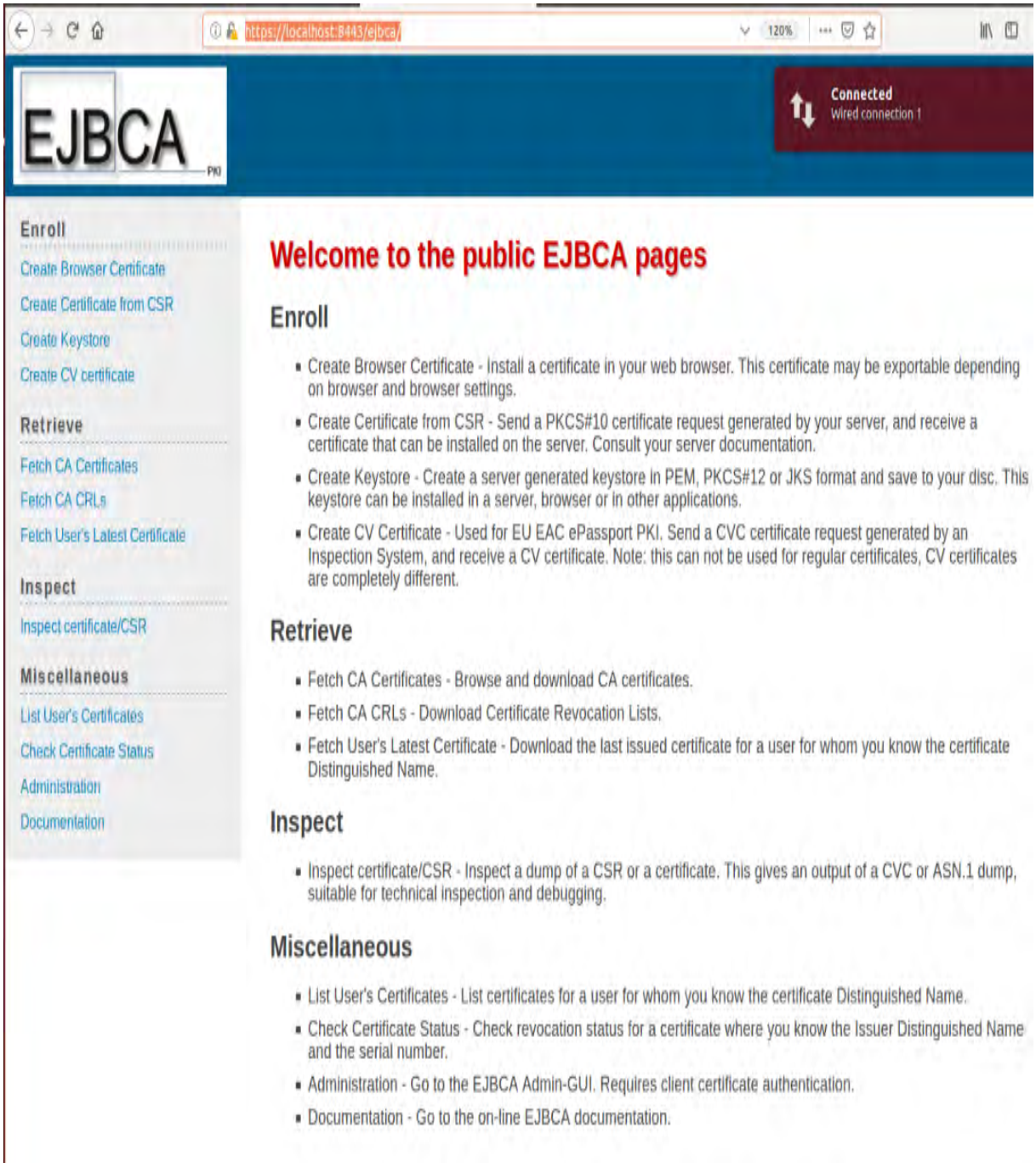



**Figure 20** – Page d'accueil ejbca

### 3.2.4 Configuration

#### Configuration D'EJBCA

Suivant le principe d'une PKI vue précédemment, nous allons configurer EJBCA en créant :

— Une interface de publication **CRL Publisher** (certificate revocation list) utili-

sée et paramétrée par la CA émettrice.

— Une interface de publication **AD Publisher** pour les certificats, utilisée également par la CA émettrice mais paramétrée dans le modèle de certificat.
— Une autorité de certification racine **ROOTCA**. Cette autorité est signée par elle-même (autosignée). Aucune autre autorité ne se porte garante de sa validité.
— Une autorité de certification émettrice **SousCAEmet** signée par ROOTCA.
— Un modèle de certificat **User cert Profile**.
— Un profil utilisateur **End Entity UUSER1**.
— Un utilisateur nommé **test** avec comme profil End Entity UUSER1 pour la phase test.

**Phase 1 : connexion au module administration**

Veuillez-vous connecter sur le serveur JBOSS à l'adresse :

https ://localhost :8443/ejbca/. Sur la page d'accueil d'EJBCA cliquez sur administration. Sur la page administration, sélectionnez system configuration pour modifier les paramètres du système.

Les paramètres suivants peuvent être modifiés : (voir capture ci-dessous)

— Titre de l'application.
— Bannières haute et basse.
— Limitation des autorisations sur les entités finales.
— Utilisation du séquestre des clés.
— L'utilisation de supports physiques.
— Demander la confirmation pour l'envoi des mails de validation.
— La langue d'affichage.
— Le nombre de lignes par page.

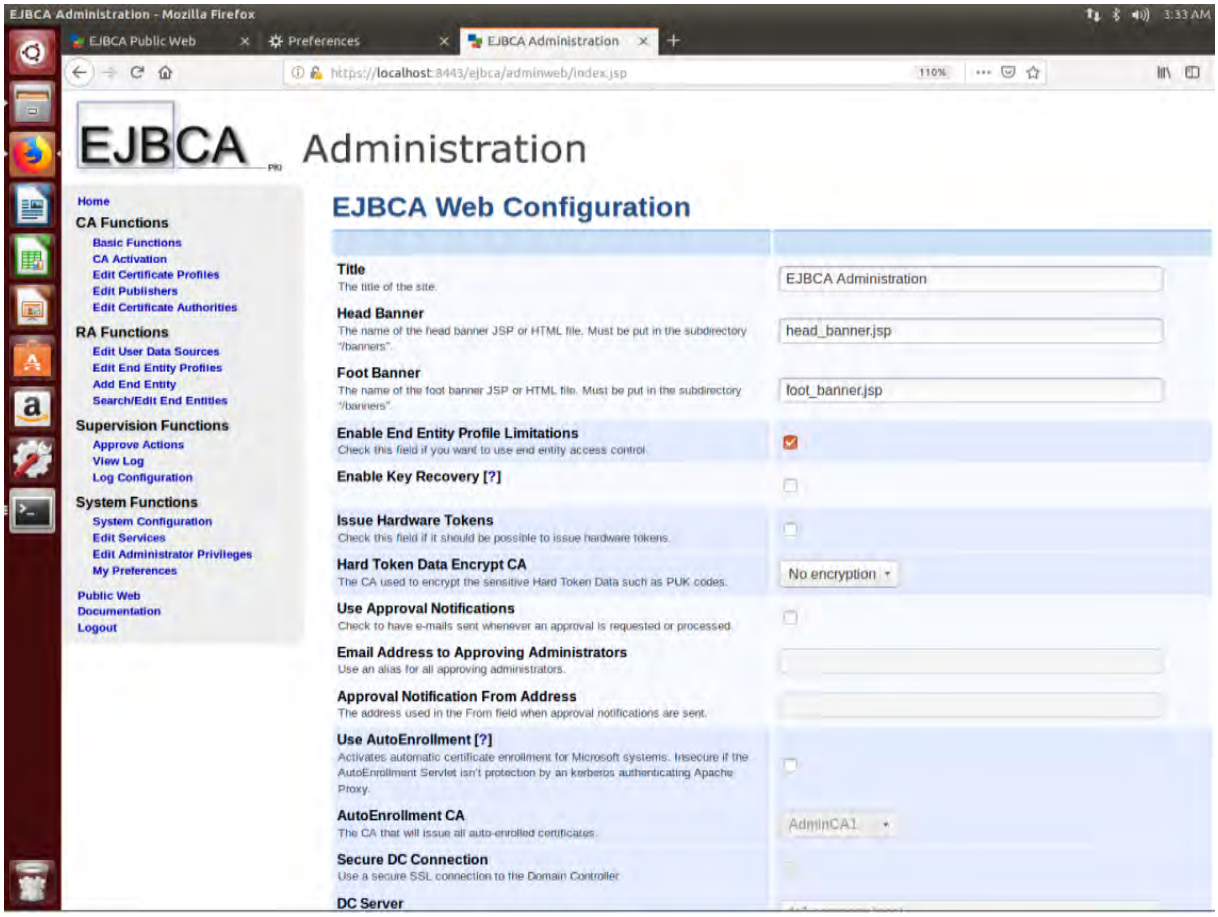



**Figure 21** – connexion au module administration

### Phase 2 : Création des interfaces de publication

Il est conseillé de créer les interfaces de publication en premier, car elles peuvent être utilisées par les CA. Nous allons créer deux interfaces de publication, une pour la CRL , utilisée et paramétrée par la CA émettrice et l'autre pour les certificats, utilisée également par la CA émettrice mais paramétrée dans le modèle de certificat. Elles porteront respectivement les noms de **CRL Publisher** et **AD Publisher**. Procédure de création d'une interface de publication : Pour créer une interface de publication, sélectionnez le choix **« Edit Publishers »** Entrez un nouveau nom **(CRL Publisher)** pour l'interface et sélectionnez le bouton **« Add »**.

Sélectionnez ensuite le nom (CRL Publisher) dans **Current Publishers** puis cliquez sur le bouton **« Edit Publisher »**.

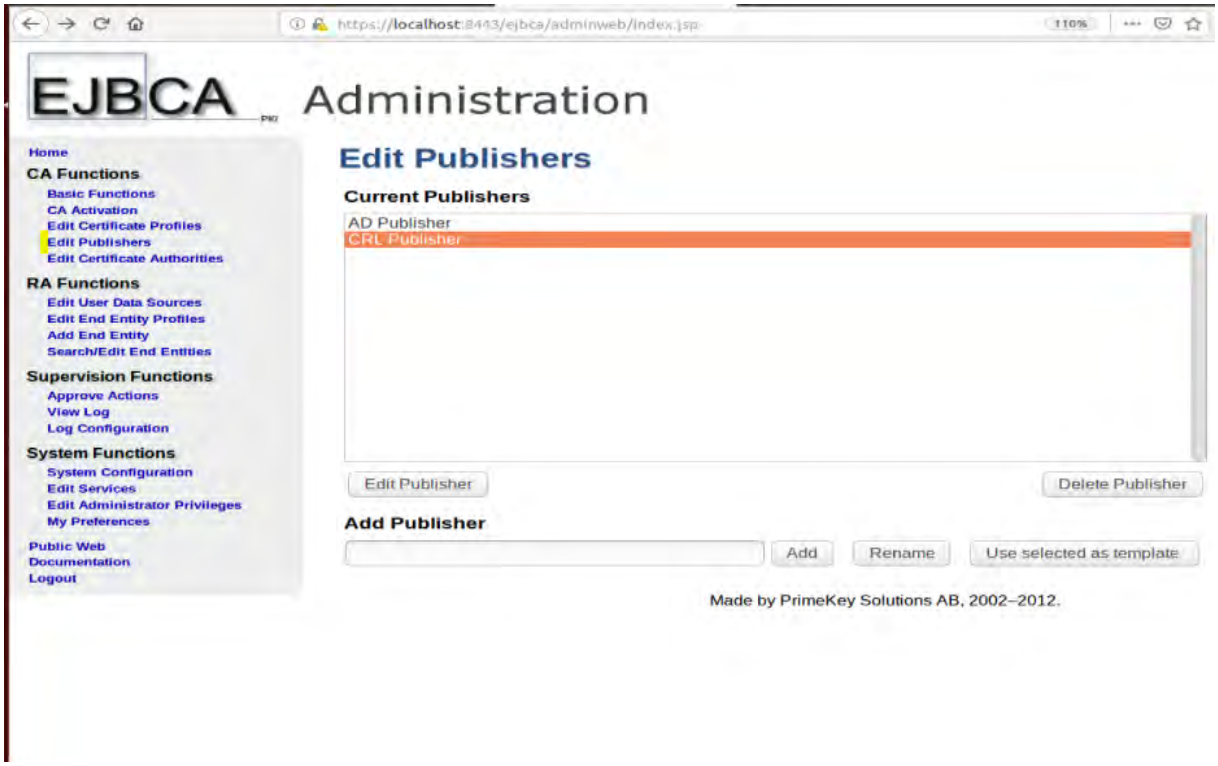



**Figure 22** – Ouverture Edit Publishers et création de l'interface CRL Publisher

Enfin, Renseignez les paramètres par exemple le type de publication, le port, la base DN, et cliquez sur le bouton save pour sauvegarder.

1. Création de CRL publisher et d'AD publisher

   En suivant la procédure ci-dessus, nous allons créer CRL publisher et AD publisher avec les paramètres suivants :

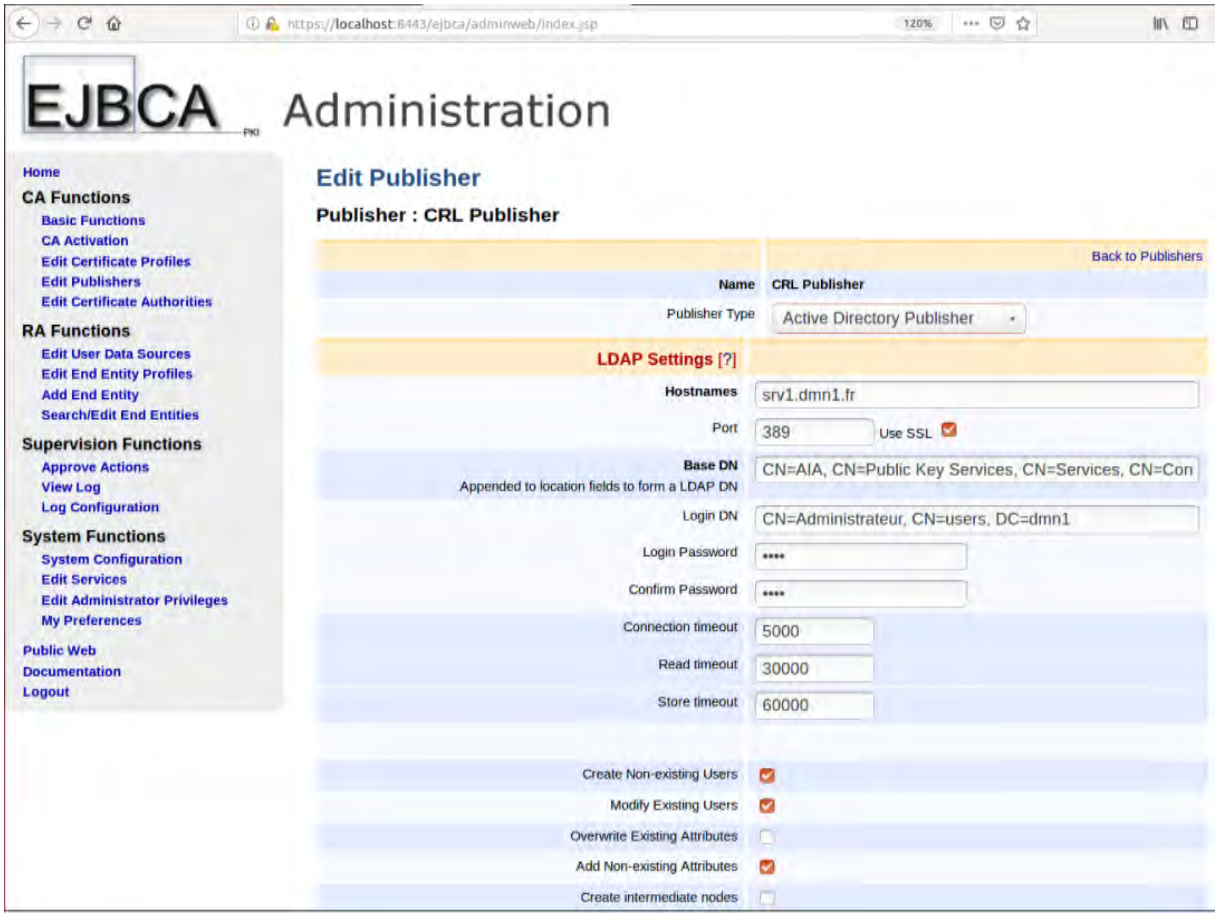
| | |
|---|---|
| Hostname | srv1.dmn1.fr |
| Port | 389 (cochez use SSL) |
| Base DN | CN=AIA, CN= Public Key Services, CN=Services,CN=Configuration, DC=dmn1, DC= fr |
| Login | CN=Administrateur, CN=users, DC=dmn1 |
| Mot de passe | root |

**Table 6** – Paramètres pour la création de CRL

| | |
|---|---|
| Publisher TYPE | Active Directory Publisher |
| Hostname | srv1.dmn1.fr |
| Port | 389 (cochez use SSL) |
| Base DN | cn=user,dc=dmn1,dc=fr |
| Login DN | cn=administrateur,cn=users,dc=dmn1 |
| Mot de passe | root |

**Table 7** – Paramètres pour AD Publisher

**fichier d'exemple de Création de CRL publisher et d'AD publisher**




2. Création de CA ROOT et CA émettrice

Nous utiliserons la CA émettrice pour la génération des certificats d'utilisateurs Nous allons choisir l'onglet **Edit Certificate Authorities** dans la barre de navigation. Nous allons ensuite renseigner le nom **Root CA** et cliquer sur **Create** Ici nous allons donc renseigner les informations nécessaires pour la création de notre autorité de certification. Par le même procéder nous allons créer **SousCAEmet**. Les informations des certificats sont les suivants :

| | |
|---|---|
| Type de CA | X509 |
| Signing Algorithm | SHA1WithRSA |
| RSA key size | 2048 |
| Validity (*y *mo *d) or end date of the certificate [?] | 3650d (3650 JOURS) |
| Subject DN | CN=RootCA,O=Lion Certicate,C=fr |
| Signed By | Self Signed |
| Certificate Profile | ROOTCA |

**Table 8** – Paramètres pour la création de RootCA

| | |
|---|---|
| Type de CA | X509 |
| Signing Algorithm | SHA1WithRSA |
| RSA key size | 2048 |
| Validity (*y *mo *d) or end date of the certificate [?] | 3650d (3650 JOURS) |
| Subject DN | CN=SousCAEmet,O=test,C=fr |
| Issuer DN | CN=RootCA,O=Lion Certicate,C=fr |
| Signed By | ROOTCA |
| Certificate Profile | SUBCA |
| Publishers | CRL Publisher |

**Table 9** – Paramètres pour la création du SousCAEmet

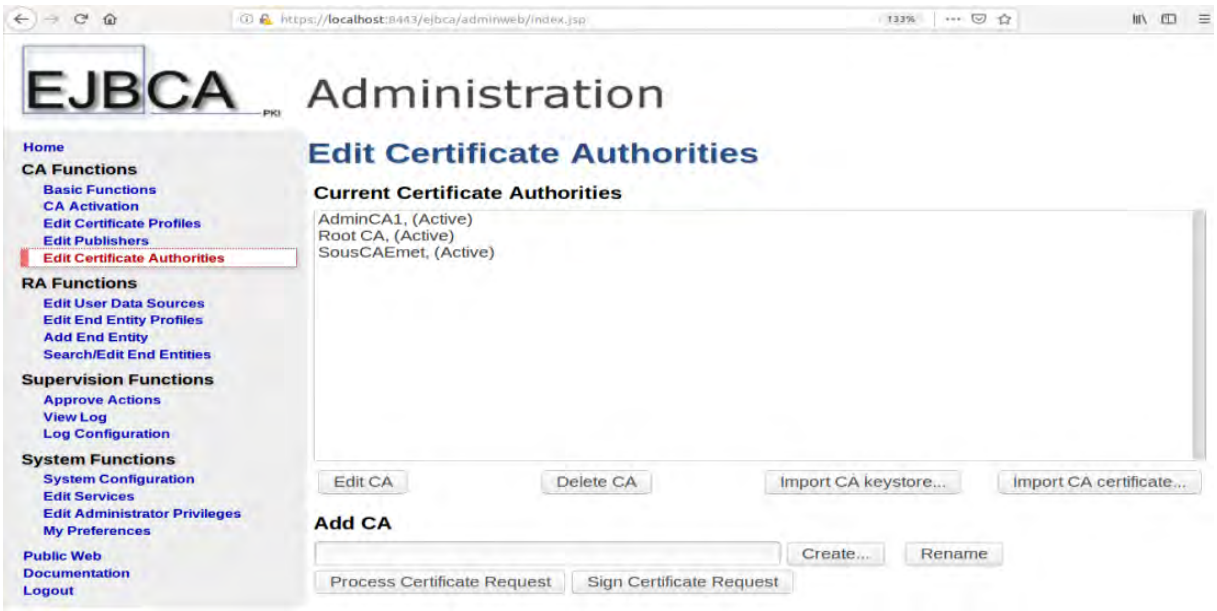



**Figure 23** – Création de Root CA et SousCAEmet

3. Création d'un modèle de certificat

   Pour se faire nous allons choisir l'onglet Edit certificate profiles dans la barre de navigation

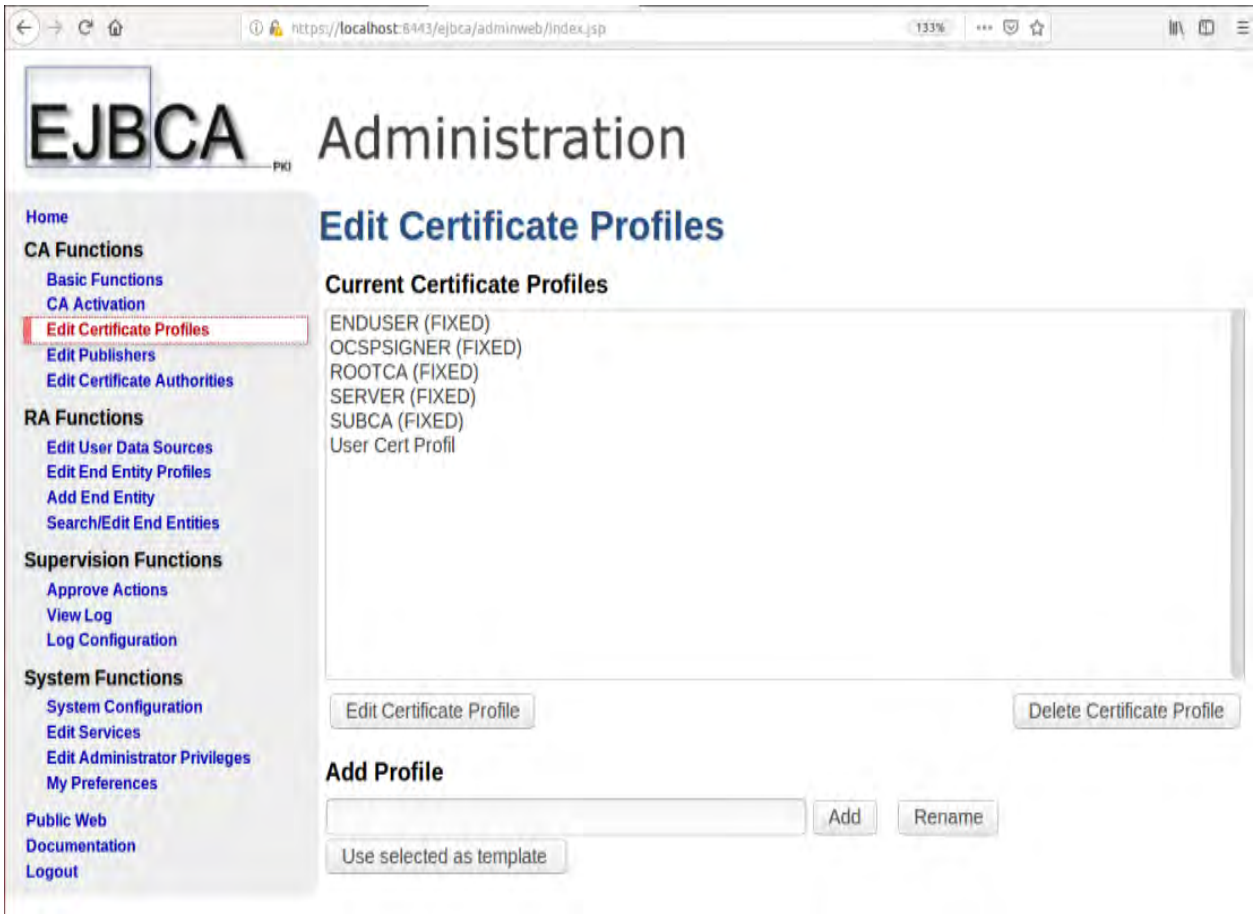



**Figure 24** – Ouverture onglet Edit Certificate Profil

Nous allons ensuite renseigner le nom **User Cert Profil** dans **Add Profile** et cliquer sur **Add** Ensuite sélectionner le nouveau profile et cliquer sur **Edit Certificate Profile** Enfin nous allons renseigner les paramètres. Pour notre exemple nous allons prendre ce qui suit :

| | |
|---|---|
| Available CAs | SousCAEmet |
| Publishers | AD publisher |

**Table 10** – Paramètre pour création d'User Cert Profil

4. Création des profils

   Pour se faire nous allons choisir l'onglet **Edit End Entity Profile** dans la barre de navigation. Nous allons ensuite renseigner le nom **End Entity UUSER1** dans **Add Profile** et cliquer sur **Add** Puis sélectionner le nouveau profile et cliquer sur **Edit End Entity Profile**

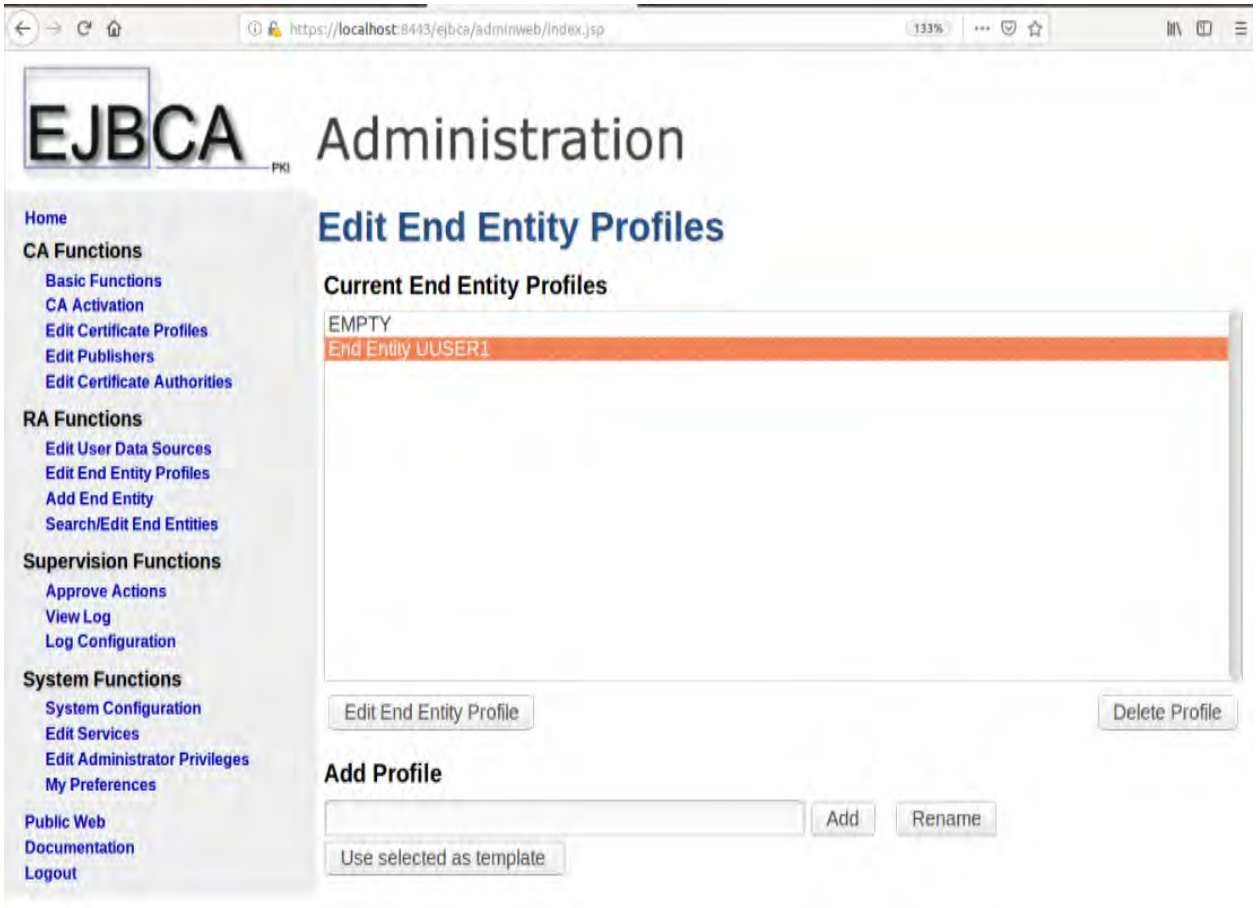



**Figure 25** – Création et configuration d'End Entity UUSER1

Ici nous allons donc renseigner les paramètres. Pour notre exemple nous prendrons les informations suivantes :

| | |
|---|---|
| Username | test (nom d'utilisateur du profile) |
| Password | root (mot de passe du profile) |
| Available Certificate Profile | User cert Profile (modèle créé précédemment) |
| Default CA | SousCAEmet (Autorité de certification émettrice qui permettra de gérer les certificats) |
| Default Tokens | P12 file (format du certificat générer) |

**Table 11** – Paramètre pour End Entity UUSER1

### 3.2.5 Applications

1. Création d'un utilisateur Nous allons créer un utilisateur **(test)** pour pouvoir procéder au test de l'installation. Pour se faire nous allons choisir l'onglet **Add End Entity** dans la barre de navigation. Il suffit de renseigner les informations et de valider la saisie. Les cases **required** qui sont cochées sont les champs obligatoires pour la création de l'utilisateur.

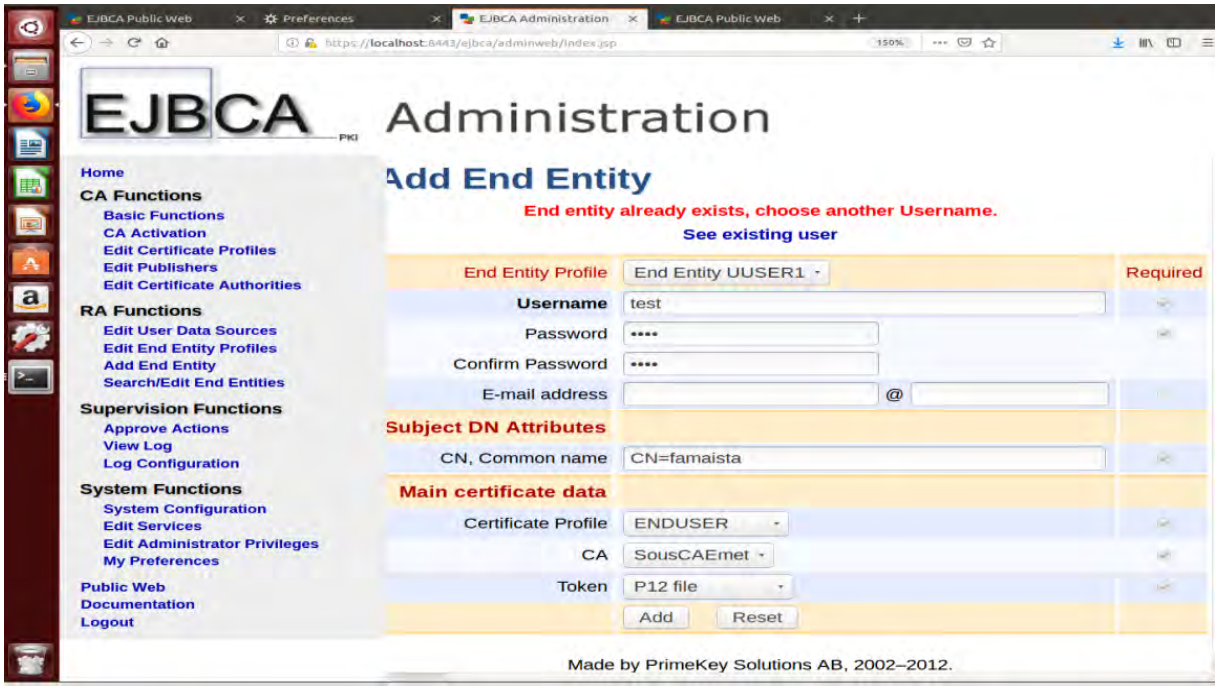



**Figure 26** – Création de l'utilisateur test

### Phase de test

Nous allons nous rendre à l'url suivant : **https ://localhost :8080/ejbca/** sur la barre de navigation à gauche, nous irons à l'onglet **Create browser certificate** pour créer un certificat navigateur. Se connecter avec les informations utilisateurs créé précédemment (test et test)

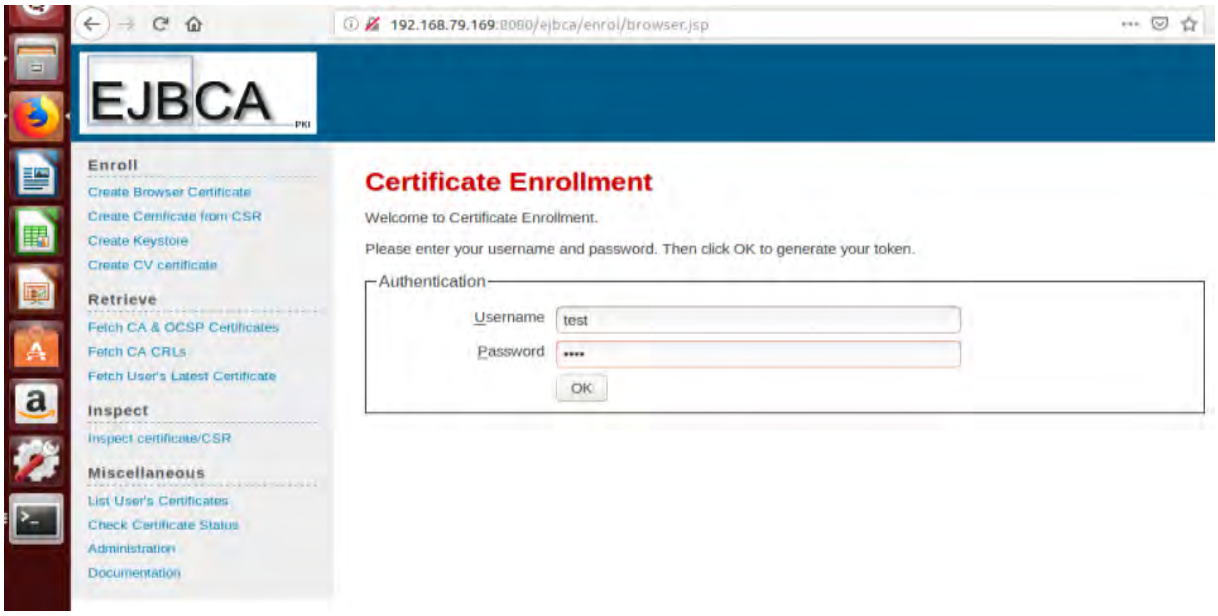



**Figure 27** – Création certificat navigateur

Choisir la taille de la clé (2048bits) et le profil du certificat (User Cert Profil).

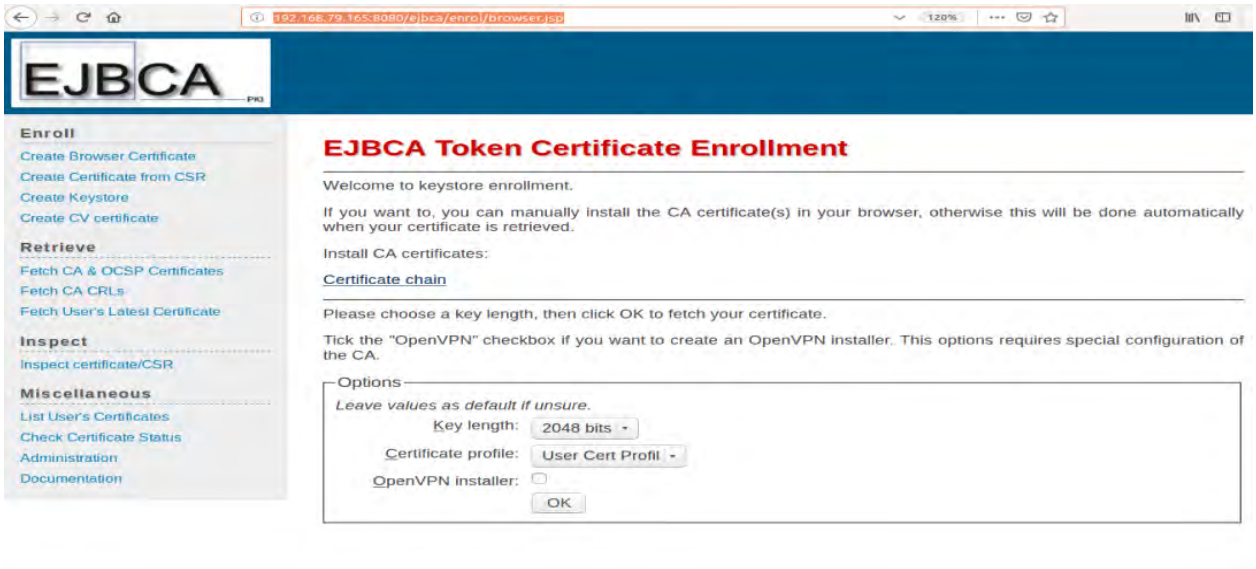



**Figure 28** – Choix de la taille clé et du profil certificat

Ce dernier sera téléchargé automatiquement après validation Après importation sur le navigateur, nous pourrons constater que le certificat a bien été généré


**Figure 29** – Vérification création certificat

### 3.2.6 Implémentation d'EBCA sur un site web

1. Création site web

   Nous avons créé un site web www.testcert.com . On a créé une machine client Windows 10 pour heberger notre site web ainsi avec notre certificat (test.p12) créé précédemment on va passer du HTTP vers le HTTPS. Pour la suite de l'application nous utiliserons IIS (Internet Information Services)

2. Service IIS

   IIS (Internet Information Services) est un serveur Web développé par Microsoft et disponible sur les versions client et serveur de Microsoft Windows. Il supporte de nombreux protocoles réseaux dont HTTP, HTTPS, FTP et SMTP.

3. Activation du service IIS

   IIS est préinstallé sur la plupart des versions de Microsoft Windows. Dans le présent document, nous utiliserons Microsoft Windows 10. L'activation de ce service se fait au niveau du panneau de configuration. Pour y accéder, il faut se rendre dans le menu démarrer et saisir « Panneau de configuration » Il faut cliquer sur « Programmes », puis sur « Activer ou désactiver des fonctionnalités Windows ». Il faut ensuite, cliquer cocher la case « Internet Information Service

», puis cliquer sur « OK ». L'activation d'IIS est maintenant terminée. Sur les versions serveur de Microsoft Windows, IIS s'active en tant que rôle.

4. Hébergement d'un site Web non sécurisé

   L'hébergement d'un site web, se fait en spécifiant le chemin du site et en faisant le choix entre le protocole HTTP et HTTPS. Pour le moment, nous déploierons ce site en HTTP, c'est-à-dire de façon non sécurisée. Tout d'abord dans c :// on a créé le fichier « «testCert » et à l'intérieur on a mis le fichier « index.html » de notre site web. On ouvre le « Gestionnaire des services internet » Ensuite faire un clic droit sur « Sites », puis cliquer sur « Ajouter un site Web... » Dans la fenêtre intitulée « Ajouter un site Web », il faut nommer le site à ajouter, et spécifier son chemin d'accès. Par défaut, le type de liaison est http.

   **NB :** Ici le port par défaut est changé par **81** (pas obligatoirement de le changer)

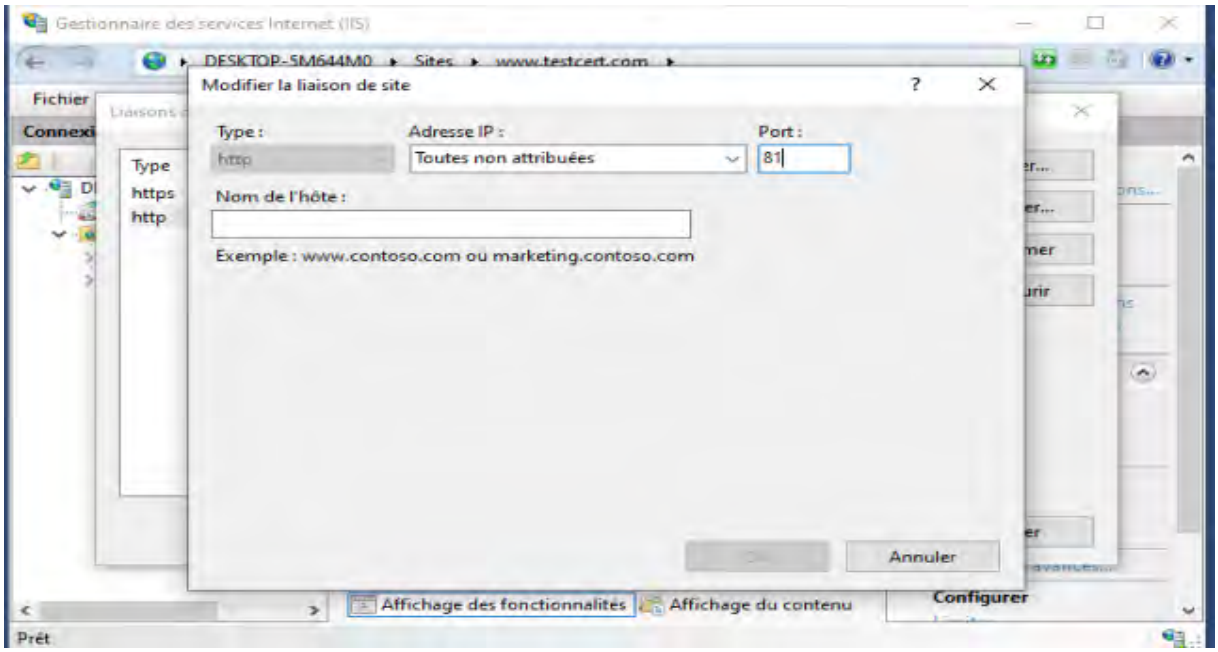



**Figure 30** – Nom et chemin d'accès du site

Après la création du site, on fait clic droit sur le site, on choisit « Gérer le site web » puis « Démarrer »

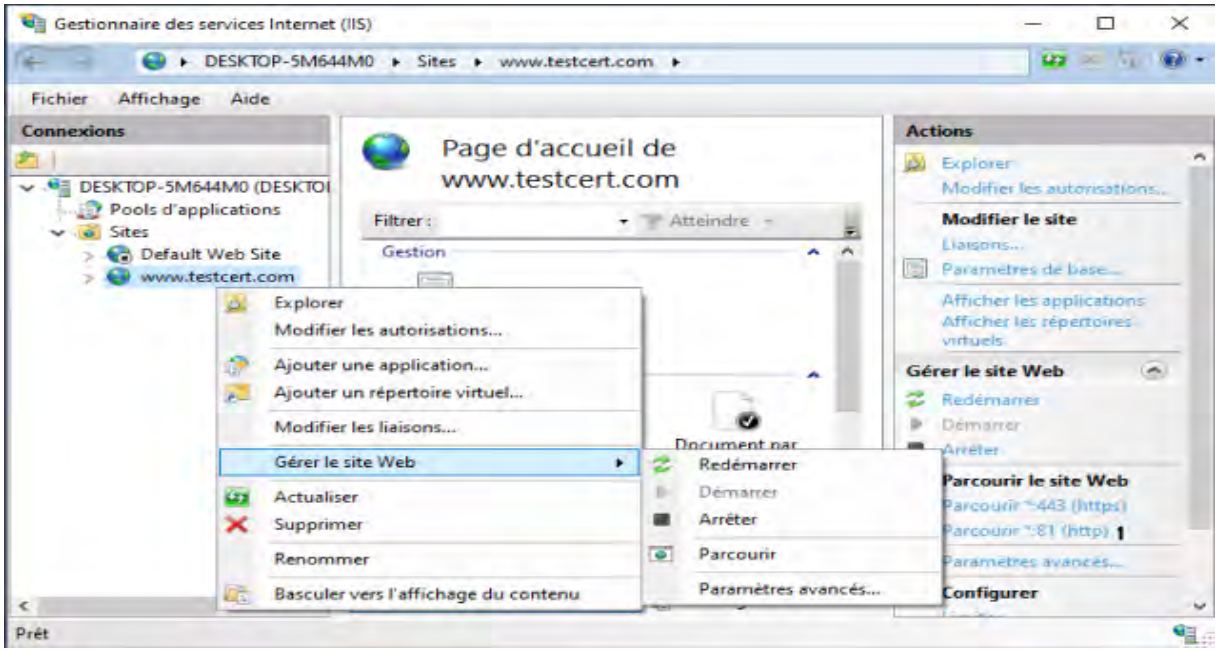



**Figure 31** – Démarrage du site

En ouvrant un navigateur Web, et en saisissant dans la barre d'adresse, « localhost », nous sommes redirigés vers la page d'accueil du site que nous venons d'héberger. En regardant de plus près l'icône précédent l'adresse du site Web, nous constatons que la liaison entre le client et le serveur n'est pas sécurisée. Cela s'explique par le fait que nous avons utilisé le protocole http.

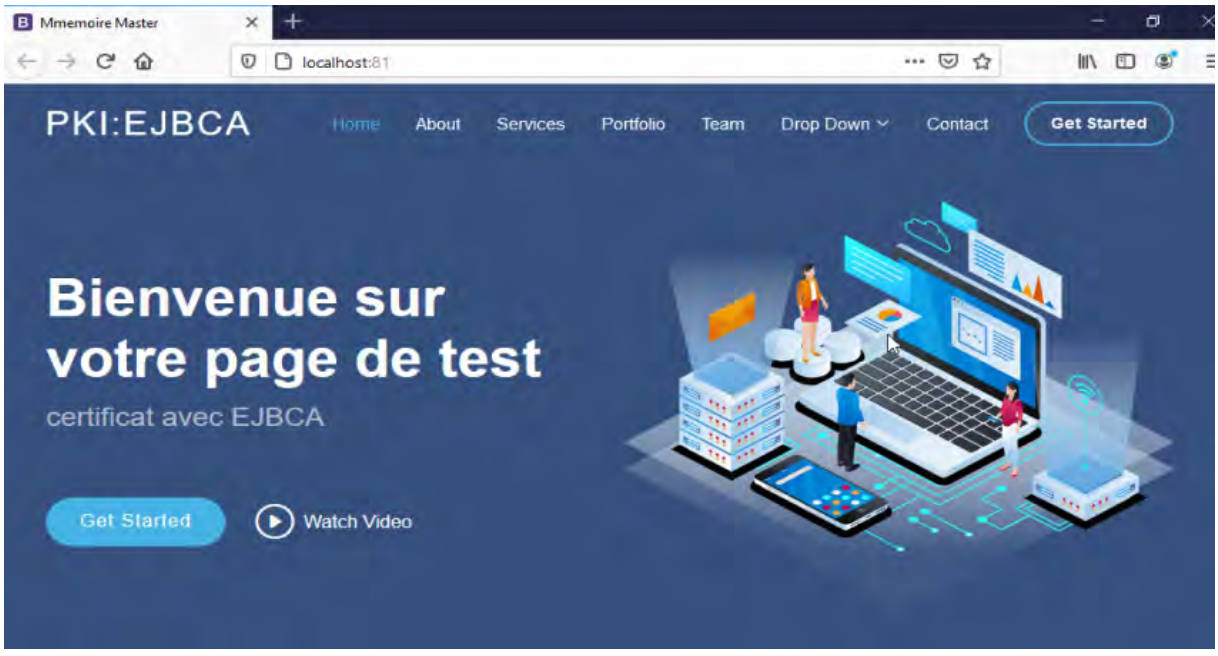



**Figure 32** – Liaison non sécurisée entre client/serveur

Dans la suite du document, nous verrons comment sécuriser ce site Web. Cette sécurisation passe par un certificat.

5. Importation d'un certificat

   A partir de la machine d'EJBCA **(Ubuntu)** on va copier le fichier **« test.p12 »** et le coller dans notre machine **Windows 10**. L'importation d'un certificat se fait sur l'interface d'administration d'IIS. Il faut d'abord double-cliquer sur **« Certificats de serveur »**, ensuite sur « importer » et on choisira le chemin du fichier **« test.p12 »** et son mot de passe et enfin on clique sur **« OK »**

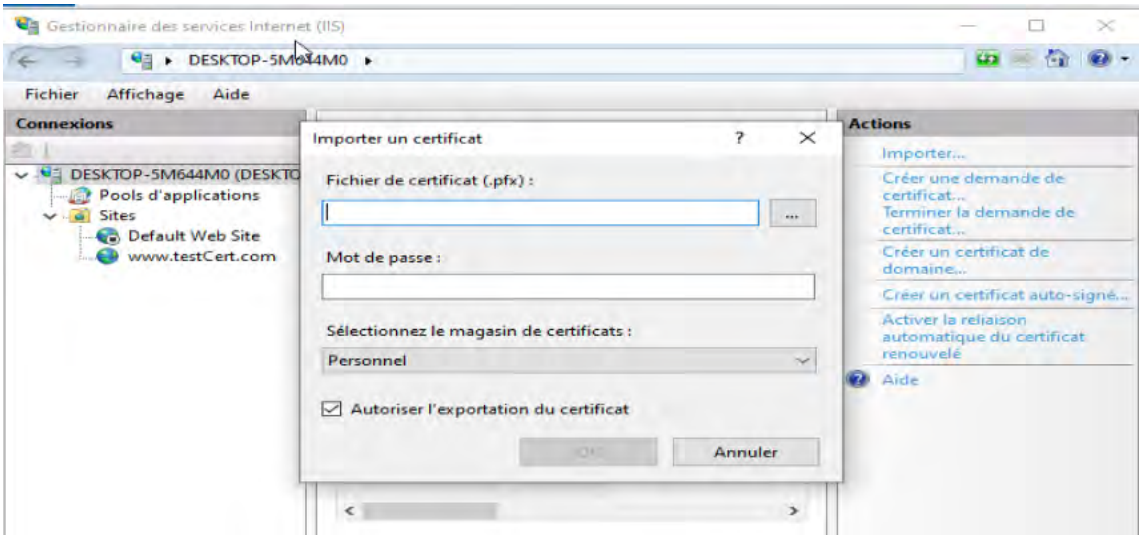



**Figure 33** – Importation certificat

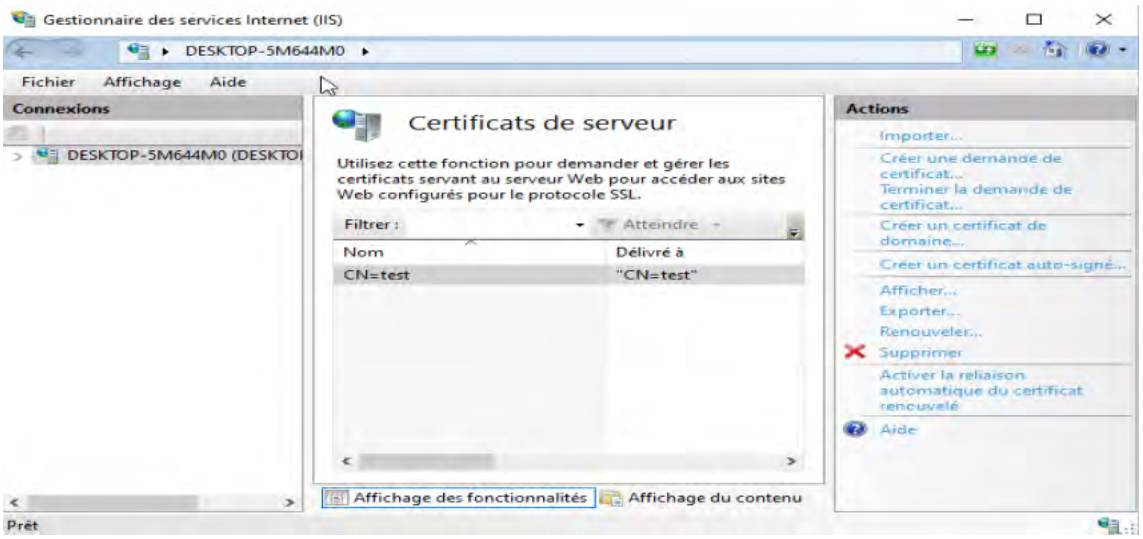



**Figure 34** – Réussite de l'importation

6. Sécurisation du site Web : migration vers le protocole HTTPS Tout d'abord on clique sur le site Web, dans la section « **Modifier les liaisons** », Dans celle ci, il faudra créer une nouvelle liaison de type HTTPS, en cliquant sur le bouton « **Ajouter** ». Ensuite il faudra choisir le protocole HTTPS, sans oublier de sélectionner un certificat SSL. Dans notre cas, nous sélectionnerons celui que nous avons importé précédemment.

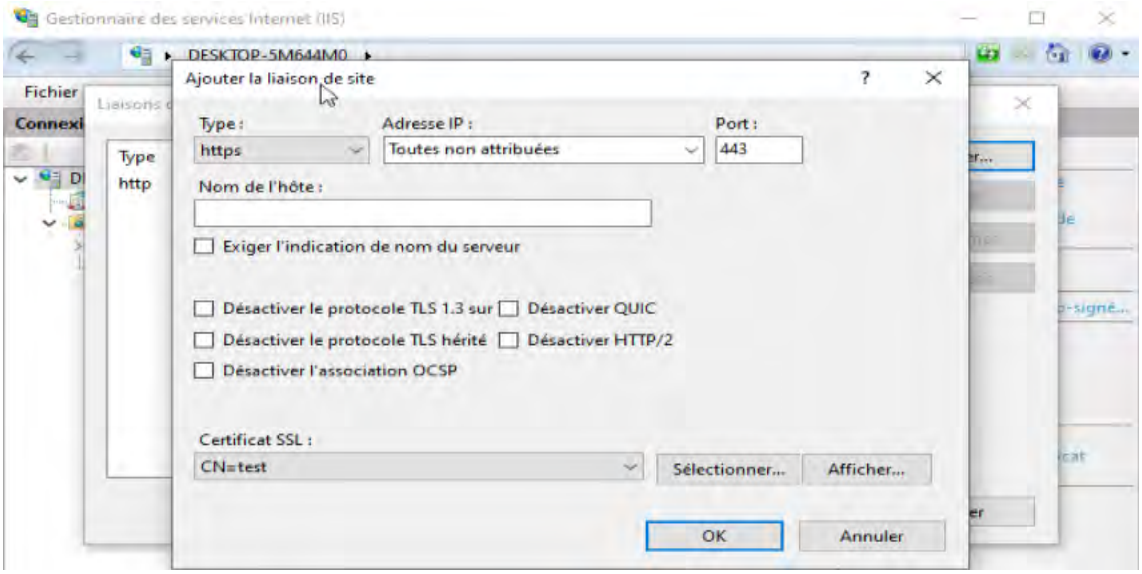



**Figure 35** – Ajout nouvelle liaison HTTPS

Notre site Web, est maintenant sécurisé et utilise le protocole HTTPS

7. Test sur un navigateur

   En saisissant « **localhost** » dans la barre de recherche d'un navigateur, on reçoit un avertissement. Cela est tout à fait normal, car le certificat utilisé par notre site Web, n'est pas connu du navigateur. Il faut donc passer outre cet avertissement, afin d'accéder au site Web.

   Lorsque le site Web s'affiche, on peut cliquer sur le label « Non sécurisé », juste à gauche de l'URL dans la barre d'adresse, pour vérifier le certificat qui intervient dans cette communication client-serveur

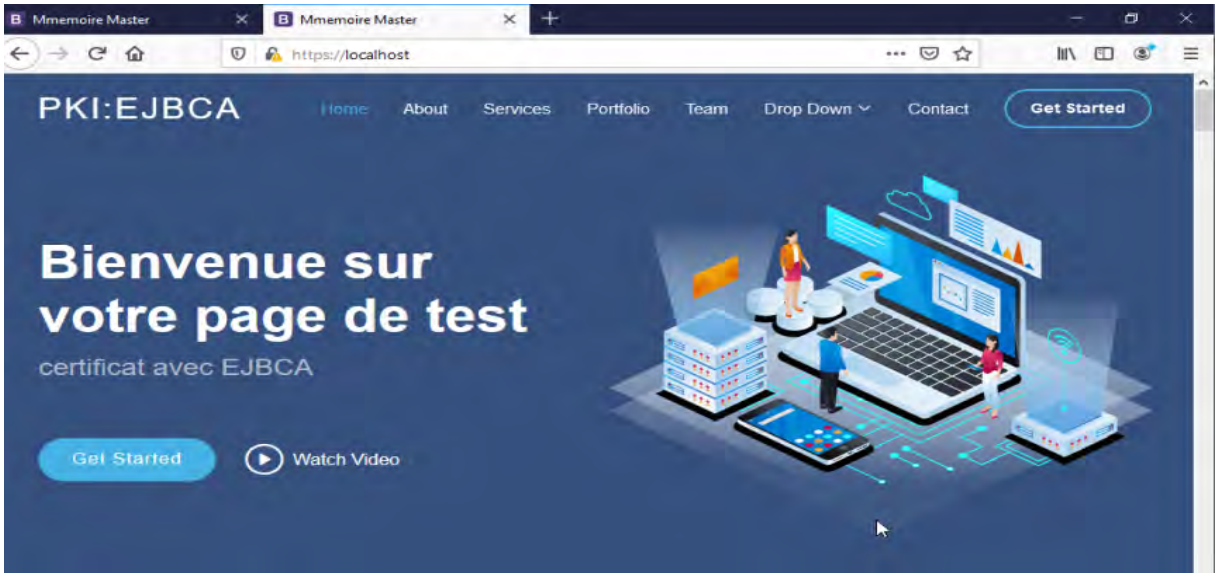



**Figure 36** – Liaison sécurisée entre client/serveur

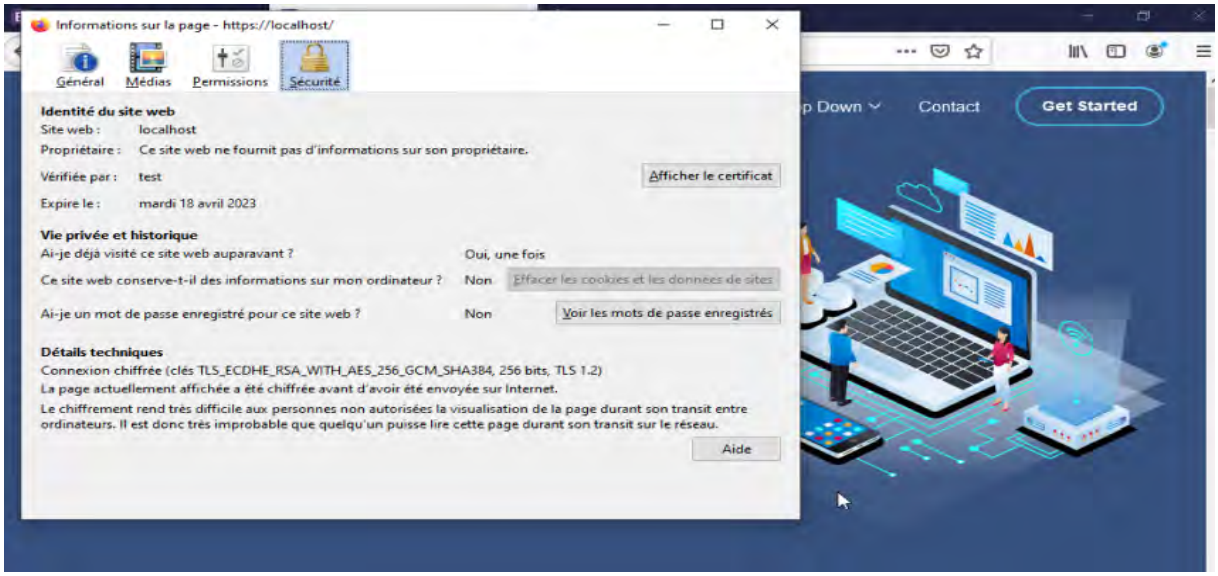



**Figure 37** – Vérification du certificat sur le site web

**Conclusion** :

Dans ce chapitre, Nous avons vu comment mettre en place une infrastructure de gestion de clés publique avec EJBCA.

Bien que normalement sur des serveurs dédiés, à partir du moment où on a un système d'exploitation on peut l'y adapter. Il existe des alternatives à l'EJBCA mais elle reste pratique C'est ainsi qu'on va l'appliquer à un site web et sécurisé ce dernier en passant du HTTP vers le HTTPS

# Conclusion générale et perspectives

Durant notre projet, nous avons pu mettre en œuvre une infrastructure de gestion de clés publiques avec EJBCA pour la garantie des services de base de la cryptographie moderne à savoir la confidentialité, l'authentification, l'intégrité et la non répudiation.

La mise en place de cette PKI a permis de faciliter la gestion des clés publiques et la sécurisation des sites web par le moyen des certificats numériques et avec les travaux pratiques que nous avons eu à effectuer, nous avons eu une idée sur le fonctionnement de la PKI et son importance.

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.

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.

Néanmoins, notre projet pourra être amélioré par l'ajout d'autres fonctionnalités comme la sécurisation de :

La messagerie : S/MIME

La plateforme de monitoring réseau Nagios

Du serveur FTP

# Bibliographie


[1] Cours cryptographie de Docteur Demba SOW Enseignant chercheur au Département de Mathématiques/Informatique Email : demba1.sow@ucad.edu.sn, sowdembis@yahoo.fr

[2] Saidou DIOP, PKI pour sécuriser les messages dans un réseau V2G, maitrise en mathématique et informatique appliquées, année MARS 2018,présenté à l'université du QUÉBEC À TROIS-RIVIÈRES :

[3] AbdelKader Fatima Zahra et Faetan Soumia,
memoire de master en informatique année universitaire 2016/2017, Universite Belhadji Bouchaid d'Ain-Temouchent
disponible sur : `https://fr.scribd.com/document/465632145/MEMOIRE-Openssl3-pdf`

[4] Etude de cryptosystèmes à clé publique basés sur les codes MDPC quasi-cycliques par Julia Chaulet
Consulté le 01/04/2021 , disponible sur :
`https://tel.archives-ouvertes.fr/tel-01599347/file/2017PA066064.pdf`

[5] LIVRE BLANC GLOBALSIGN John B Harris, Expert en sécurité Pour GMO GlobalSign Ltd.

[6] LIVRE BLANC SoluCom,sur :
Les PKI : Vers une Infrastructure Globale de Sécurité ?
disponible sur :
`http://babacar.ndaw.free.fr/Infrastructures`

[7] Spécialistes en cybersécurité au service de votre entreprise,
disponible sur :
`https://www.securiteinfo.com/cryptographie/pki.shtml`

[8] David Kohel & Igor E. Shparlinski
THÉORIE DES NOMBRES ET CRYPTOGRAPHIE disponible sur :
`http://iml.univ-mrs.fr/~kohel/pub/NT-Crypto.pdf`

[9] Cryptographie et procédés de chiffrement,
disponible sur :
https://lipn.univ-paris13.fr/~poinsot/Articles/SRenAction-Crypto.pdf

[10] Les objectifs de la sécurité informatique
disponible sur :
http://helios.mi.parisdescartes.fr/~mea/cours/M1/securite_crypto_principes.pdf

[11] Daniel Barsky & Ghislain Dartois
Cours sur la cryptographie, publié le 1 octobre 2010 à Paris 13, consulté P.120-127
disponible sur :
https://www.math.univ-paris13.fr/~boyer/enseignement/PolyCrypto2010.pdf

[12] Resumé sur les structures algébriques des ensembles avec opérations groupe,anneaux,corps
Université Claude Bernard Lyon 1,Licence Math-Info 1 ère année
disponible sur :http://math.univ-lyon1.fr/~frabetti/AnalyseI/CM1-groupes.pdf

[13] Licence de Mathématiques Troisième année, U.E. 35MATF2
Université Blaise Pascal U.F.R. Sciences et Technologies Département de Mathématiques et Informatique
disponible sur :http://math.univ-bpclermont.fr/~fdumas/fichiers/GpAnn1cours.pdf

[14] cours du jeudi 19 octobre 2017 sur les corps

disponible sur :http://math.univ-lyon1.fr/~tchoudjem/ENSEIGNEMENT/PREPA_AGREG/corps_19oct.pdf

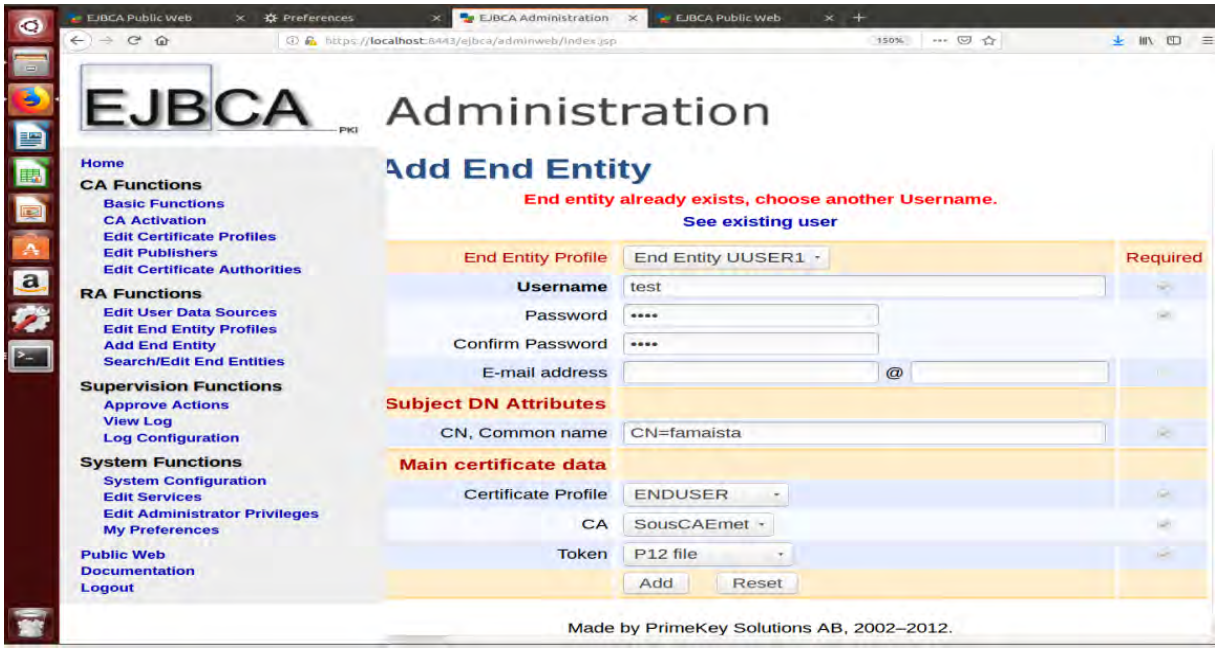
# Webographie

[**W1**] methode de chiffrement
Consulté le 21/02/2021 , disponible sur :
https://www.journaldunet.fr/patrimoine/guide-des-finances-personnelles/1209336-cryp
[**W2**] PKI et Sécurité par David CARELLA's Web Site
Consulté le 20/03/2021 , disponible sur :
http://david.carella.free.fr/fr/cryptographie/pki-et-certificats.html
[**W3**] Une PKI open source en route vers la certifcation Critères communs :
Consulté le 21/02/2021 , disponible sur :
https://fr.slideshare.net/linagora/linpki-ejbca-une-pki-open-source-en-route-vers-l
[**W4**] Les différentes types de PKI Consulté le 11/02/2021 , disponible sur :
https://open-source-guide.com/Solutions/Developpement-et-couches-intermediaires/
[**W5**] Clé privée et publique
Consulté le 01/02/2021 , disponible sur :
https://www.certificat.fr/fr/support/faq/quest-ce-que-est-cle-privee-et-cle-publiqu
[**W6**] Cryptographie asymétrique : tout sur la méthode de chiffrement
Consulté le 15/01/2021 , disponible sur :
https://www.journaldunet.fr/patrimoine/guide-des-finances-personnelles/1209336-cryp
[**W7**] https://download.primekey.com/ : (consulté le 04/12/2021) [**W8**] Public
Key Infrastructure (PKI)
Consulté le 22/04/2021 , disponible sur :
http://igm.univ-mlv.fr/~dr/XPOSE2007/vma_PKI/pki.html

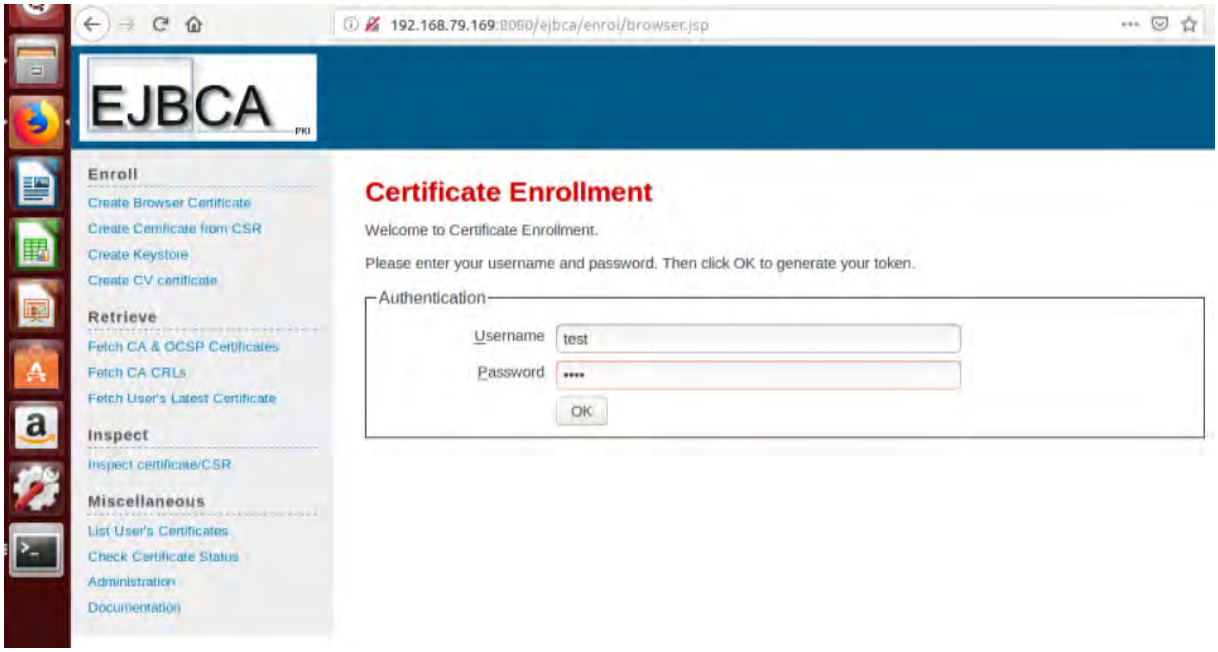
# Annexe

1. creation de certificat :

création d'utilisateur avec EJBCA



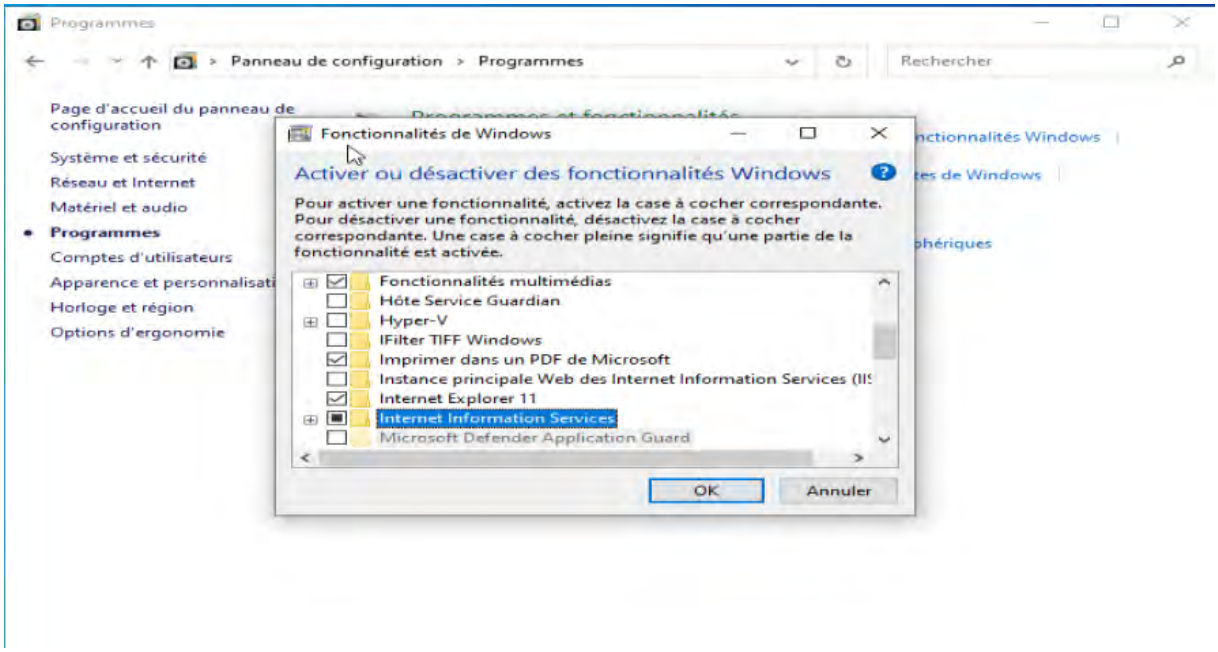

Création certificat navigateur avec l'utilisateur précédent



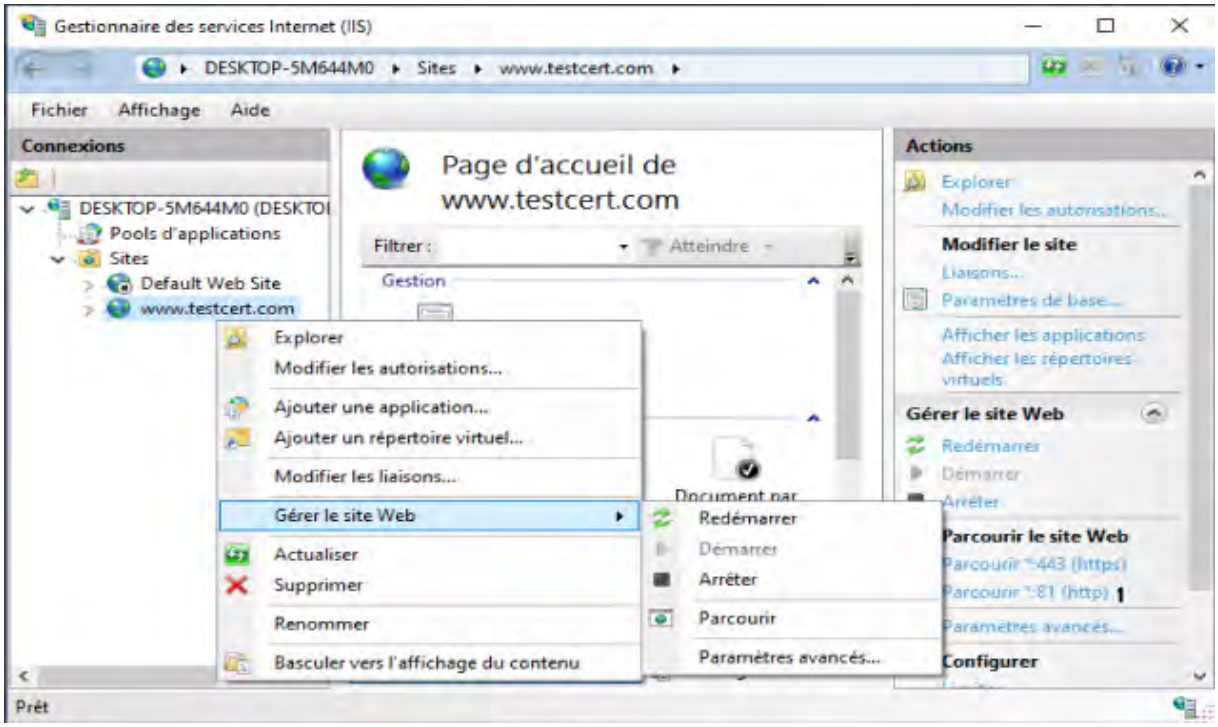
Notre certificat en place


2. Test avec notre client windows :

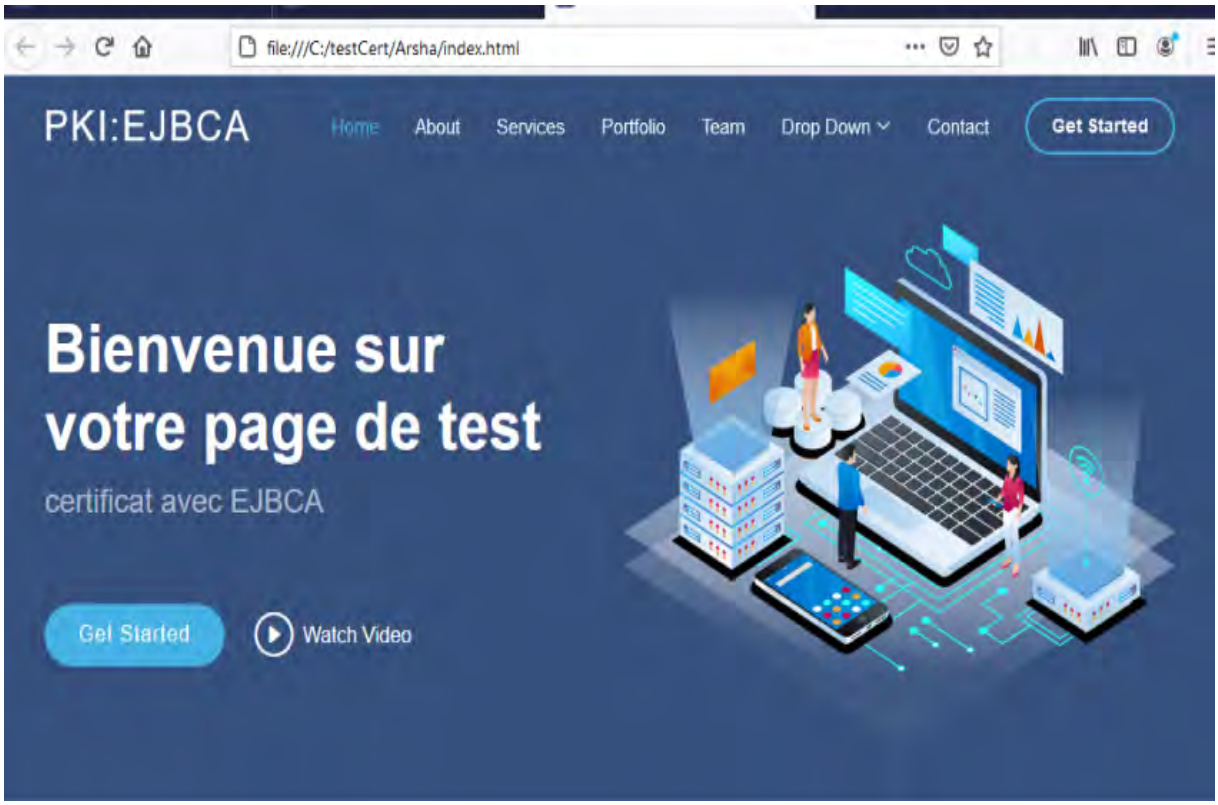
Activation de IIS



Ajout du site web www.testcert.com

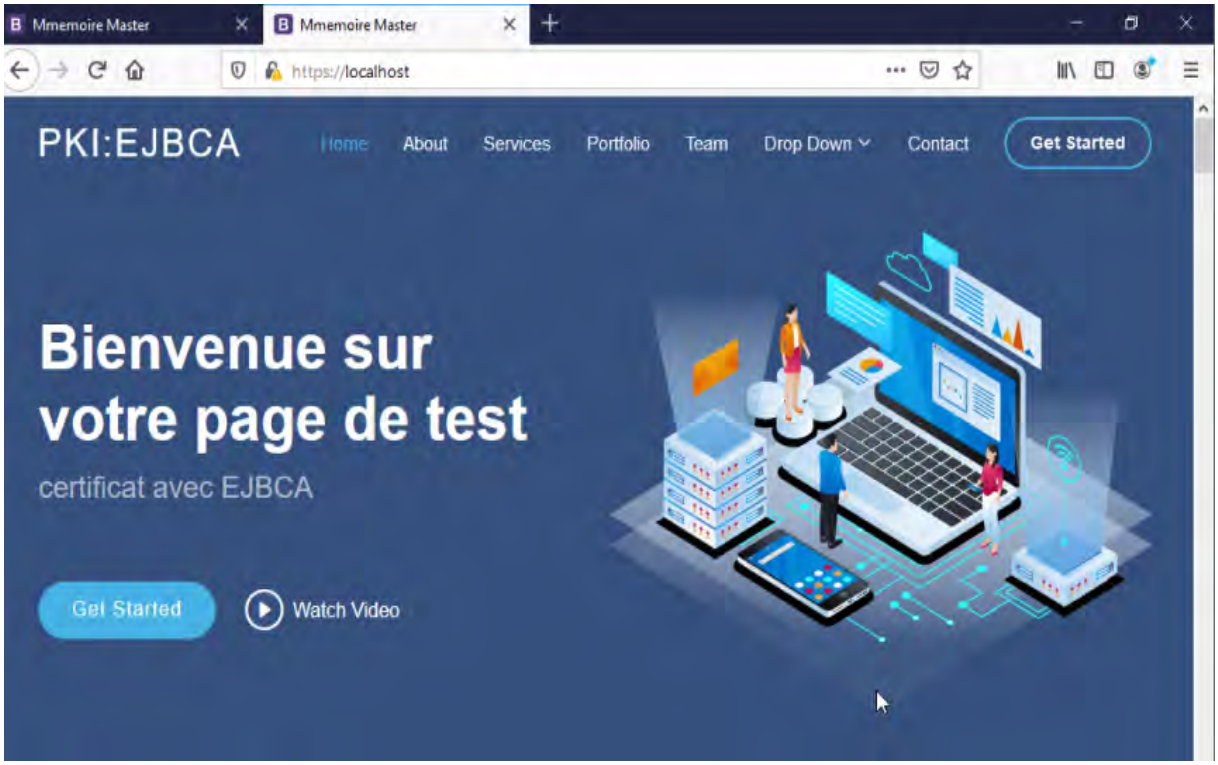



site web



connexion avec HTTP


connexion avec HTTPS



Information sur le certificat