CAHIER DE CHARGE

Développement d'un module de signature électronique (PKI)

Oumar Djimé RATOU

10 avril 2019

Table des matières

1 Description et compréhension du projet

Le projet soumis à mon analyse consiste à mettre sur pied un module de signature électronique (dite **numérique** est un moyen permettant d'identifier son propriétaire et garantir l'authenticité et l'intégrité du contenu) basant sur l'infrastructure à clés publique(ICP) ou la Public key infrastructure (PKI). Il va consister dans un premier temps :

— générer la paire de clés (publiques et privées) :
— générer un certificat auto-signé.

Et ensuite :

— calculer une valeur de hachage du document(empreinte numérique),
— chiffrer l'empreinte générée avec la clé privée,
— associer l'empreinte au document,
— vérifier la signature avec la clé publique.

Ainsi le module permettra à d'autres développeurs d'intégrer facilement dans leur plateforme de même technologie afin que les utilisateurs finaux puissent l'utiliser aisément.

Ou encore on peut facilement l'importer dans un programme quelconque de même technologie (langage de programmation) et utiliser les fonctionnalités, à l'exemple du module **math** sous python : on fait juste **import math** pour utilisé tous les fonctionnalités de la bibliothèque **math** ou on fait **from math import sqrt** pour importer seulement la fonction racine carré. Notre module comportera exactement de la même façon.

2 Étude de la faisabilité technique

2.1 Contexte et problématique

La création des signatures électronique basant sur les infrastructures à clé publiques que sa soit avec OpenSSL ou d'autres se font soit en console, soit d'utiliser des outils propriétaires tels que :
— Word pour Microsoft,
— Adobe Reader de l'entreprise Adobe,
— DocuSign,
— Eversign,
— Yousign,

soit d'aller chez une Autorité de Certification (AC) pour générer un certificat, qui sont complexes et coûteux pour les utilisateurs et surtout à ceux qui débutent en développement des applications et en sécurité informatique. Et encore malheureusement ils sont déjà intégrés dans leurs applications complètes, donc pas des moyens de les réutiliser dans d'autres logiciels comme des modules.

Par ailleurs d'autres ne sont pas dans les systèmes d'exploitation libre(open source) comme Linux, par exemple Word de Microsoft et Adobe Reader ne fonctionnent pas sous Linux.

Les problèmes qui surviennent souvent dans les entreprises en particulier et chez les développeurs en général c'est la disposition des programmes modulaires pour intégrer facilement dans leurs plates-formes en fin de gagner en temps et en l'argent (surtout pour les entreprises). Ces nécessités nous amènent à nous poser les questions suivantes :

— Comment peut-on rendre cette difficulté de signer un document de manière transparente ?
— Comment faciliter le développement d'un outil informatique au sein de l'entreprise ?
— Comment développer un module multi-plateforme ?

2.2 Objectifs

2.2.1 Objectif global

Il sera question pour moi de développer un module des gestions des signatures basant sur les infrastructures à clé publique et le rendre modulaire.

2.2.2 Objectif spécifique

De façon spécifique, il sera question pour moi de gérer les problèmes spécifiques liés aux :
— Création d'une signature des documents numériques (texte, son, vidéo, PDF, etc.) en se basant sur les infrastructures à clé publique ou PKI,
— automatisation de la création des la signature à la main,
— vérification de la signature de document numérique,
— prouver l'authenticité d'un signataire,
— faciliter à l'entreprise lors d'un besoin d'un module de signature électronique,
— rendre la vie facile au développeur qui ne maîtrise pas forcément la notion de cryptographie qui se cache derrière la signature numérique, d'intégrer ce module dans sa plate-forme.

Ainsi les utilisateur avertis pourront juste l'importer dans leurs programmes et l'utilisé facilement. Tout ce qu'un utilisateur doit connaître c'est le nom de la méthode qu'il veut appeler avec sa signature (les paramètres de la méthode) et la méthode retournera le résultat voulu. Il y'aura une commande d'aide si l'utilisateur ne connais pas le nom de la méthode avec sa signature(paramètre) et une manuelle bien documenté avec des exemples d'utilisations.

Environnement

L'environnement dans lequel nous nous trouvons est favorable au projet puisqu'un tel système existe certes mais sous une licence payante et non modulable. Sa mise en œuvre sera une innovation importante dans l'évolution numérique au sein de l'entreprise.

3 Description des besoins

3.1 Spécifications fonctionnelles

Notre module de signature électronique aura comme spécification fonctionnelle :
— générer des paires de clés(publique et privée) ,
— générer un certificat (auto-signé, puisque le signé est payant)
— signer un document numérique à l'aide de la clé privée,
— vérifier la signature d'un document numérique à l'aide de la clé publique,
— chiffrer/déchiffrer un document,
— générer un Hash d'un document.

Explication de chaque fonctionnalité : au préalable on import d'abord le module dans notre programme :

1. pour générer une paire de clé rien de plus compliqué on suppose que le nom de la fonction s'appelle genererPaireKey(algo, taille) et prend en paramètre l'algorithme (RSA par exemple) et la taille de clé (2048 bits par exemple), la commande finale sera **genererPaireKey(RSA, 2048)**, ainsi la méthode nous retournera une paire de clé RSA de taille 2048 bits (ex. OpenSSL). On pourra aller plus loin en protégeant la clé avec un algorithme symétrique, ça veut dire que les paramètres augmenteront et deviendra **genererPaireKey(algoAsym, algoSym, taille)** dont la commande finale sera **genererPaireKey(RSA, 3DES, 2048)**, du coup après la validation on demandera d'entré un mot de passe deux fois de suite. Ainsi on se retrouvera avec un système complet et sécurisé.
2. la génération du certificat par une autorité de certification(AC) doit respecter la spécification de la norme x.509, on aura les paramètres obligatoires suivants :
   — Country Name (ex. CM),
   — Locality Name (ex. YAOUNDÉ),
   — Organizational Unit Name (ex. ITS),
   — Common Name (ex. CRYPTO),
   — Email Address (ex. its@gmail.com)
3. la signature d'un document à besoin de plusieurs paramètres(fichier ou document ,clé privée et clé publique ,certificat, le Hash du fichier) dans deux scénarios :

   Expéditeur : Signature

   — Condensation du contenu : on hache le fichier avec une fonction de hachage (ex. SHA256) et on obtiendra un condensé de 63 bits quelque soit la taille du document,
   — On crypte le résultat précédente avec un algorithme asymétrique en utilisant la clé secrète,
   — En suite on associe un certificat délivré par une autorité de certification,
   — En fin on attache au contenu et on envoi au destinateur.

   Destinateur : Vérification

   — Condensation du contenu : on hache le fichier avec une fonction de hachage (ex. SHA256) et on obtiendra un condensé de 63 bits quelque soit la taille du document. ***Attention*** on utilisera la même fonction de hachage(SHA256) sinon la vérification est sera fichu à l'avance,

— Décryptage de la signature en utilisant la clé publique de l'expéditeur. ***Attention*** : si le décryptage est échoué la vérification n'aura lieu puisqu'il y'aurai un problème d'intégrité,
— Comparaison des deux Hash : Si les deux Hash sont identiques, la signature est valide, sinon l'intégrité est bafoué.

3.2 Spécifications non-fonctionnelles

Comme spécification non-fonctionnelle, nous aurons :
— authentification : le fait de s'assurer que l'expéditeur est bien celui qu'il prétend être,
— intégrité : le fait de s'assurer que l'information ne subisse aucune altération ou destruction volontaire ou accidentelle, et conserve le format initial,
— prouver l'identité de l'expéditeur grâce au certificat délivré à l'expéditeur par une AC,
— la confidentialité est le faite que seul l'expéditeur et destinataire ont connaissance du secret partagé entre eux,
— fonctionne 24 heure sur 24, 7 jours sur 7.

4 Enveloppe budgétaire

Pour réaliser ce projet on a besoin des outils suivants :
— d'un ordinateur avec un système d'exploitation (Linux, Microsoft, ou Mac OS) mais dans ce cas on privilégie l'open source(Linux),
— un environnement de développement, PyCharm (libre),
— une connexion illimité,
— lieu de travail ou une poste adapter à un développeur,

5 Délai

J'estime que ce module pourra me prendre 1 à 2 mois.