

République du Cameroun

Paix-Travail-Patrie

Ministère de l'Enseignement Supérieur

Université de Maroua

Ecole Nationale Supérieure

Polytechnique de Maroua



Republic of Cameroon

Peace-Work-Fatherland

Ministry of Higher Education

The University of Maroua

National Advanced School of

Engineering of Maroua

INFORMATIQUE ET TELECOMMUNICATIONS

CRYPTOGRAPHIE ET SÉCURITÉ INFORMATIQUE

CONCEPTION ET DÉPLOIEMENT D'UN MODULE DE SIGNATURE ÉLECTRONIQUE BASÉ SUR LA PKI

Mémoire présenté et soutenu en vue de l'obtention du Diplôme
D'INGÉNIEUR DE CONCEPTION EN CRYPTOGRAPHIE ET SÉCURITÉ
INFORMATIQUE

par

OUMAR DJIMÉ RATOU

licencié en Mathématique et Informatique, options : Informatique

Matricule : 17Y4s02P

sous la Direction de :

Dr. BOUDJOU TCHAPGNOUO HORTENSE

Chargé de Cours

Devant le jury composé de :

Président : Dr. BOUDJOU TCHAPGNOUO HORTENSE

Examineur : Dr. BOUDJOU TCHAPGNOUO HORTENSE

Encadrereur : Dr. BOUDJOU TCHAPGNOUO HORTENSE

ANNÉE ACADÉMIQUE : 2018-2019

Dédicaces

♥ A MES PARENTS

♥ SPÉCIALEMENT A MA MÈRE ACHTA AHMAT RATOU

♥ A MON ONCLE MATERNELLE RATOU BARKA

♥ A LA FAMILLE RATOU

Remerciements

Le présent travail n'aurait sans doute pas été réalisé sans le soutien de certaines personnes à qui nous tenons à exprimer notre profonde gratitude et nos sincères remerciements. Nous tenons donc ainsi à remercier :

- ⇒ Notre président de jury Dr. BOUDJOU TCHAPGNOUO HORTENSE pour avoir accepté de présider notre jury ;
- ⇒ Notre examinateur Dr. BOUDJOU TCHAPGNOUO HORTENSE pour toutes les remarques et suggestions apportées à notre travail ;
- ⇒ Notre encadreur Dr. BOUDJOU TCHAPGNOUO HORTENSE pour sa disponibilité, sa rigueur dans le travail et les remarques apportés à ce travail ;
- ⇒ Notre Chef de Département Dr. KALADZAVI GUIDEDI pour tous ses conseils, sa disponibilité et les efforts consentis à notre égard ;
- ⇒ Nos enseignants du Département Informatique et Télécommunications pour les enseignements dispensés et les conseils prodigués ;
- ⇒ Notre encadreur professionnel de stage Dr. Bell Georges pour sa disponibilité, son encadrement et pour les remarques apportés à notre travail pendant notre stage ;
- ⇒ les consultants de l'entreprise qui nous ont aidés pendant notre stage ;
- ⇒ Nos parents, M. RATOUBARKA, M. DJIMÉ RATOUBARKA et Mme ACHTA AHMAT RATOUBARKA pour le soutien moral, affectif, les conseils et surtout financier ;
- ⇒ Nos frères et sœurs pour leur soutien moral et affectif ;
- ⇒ Notre Oncle paternel M. BATIKODNE BADA pour le soutien moral et affectif ;
- ⇒ La famille RATOUBARKA pour tout l'amour, le soutien et l'attention qu'elle nous apporte ;
- ⇒ nos amis de l'enfance qui ne m'ont jamais oublié pendant toutes ces années d'étude ;
- ⇒ Nos camarades de promotion pour les conseils, aides et les moments passés ensemble ainsi que l'ambiance conviviale qui a toujours régné entre nous ;
- ⇒ Tous ceux qui ont contribué de près ou de loin à la réalisation de ce projet.

Tables de matière

Dédicaces	i
Remerciements	ii
Tables de matière	v
Résumé	vi
Abstract	vii
Listes des tableaux	viii
Listes des figures	x
Introduction	1
I Présentation générale	2
1 Contexte et problématique	3
Introduction	3
1.1 Présentation de l'entreprise	3
1.1.1 Historique	3
1.1.2 Organisation Administrative	3
1.1.3 Missions	4
1.1.4 Services et Produits	5
1.1.5 Cadre de stage	7
1.2 Contexte	8
1.3 Problématique	9
1.4 Objectifs	9

1.4.1	Objectif général	9
1.4.2	Objectif spécifique	10
1.5	Méthodologie	10
	Conclusion	10
2	Présentation générale de la signature électronique	12
	Introduction	12
2.1	Concept général sur la signature électronique	12
2.1.1	Historique	12
2.1.2	Signature	13
2.1.3	Signature électronique	14
2.2	Cryptographie	17
2.2.1	Terminologie	17
2.2.2	Buts de la cryptographie	19
2.2.3	Cryptographie symétrique	20
2.2.4	Cryptographie asymétrique	21
2.2.5	Hachage	26
2.3	Infrastructure à clé publique	29
2.3.1	La gestion des clefs	30
2.3.2	Les composants d'un PKI	31
2.3.3	Quel est le cadre juridique qui régit l'exercice des activités électroniques ou de la certification au Cameroun	32
2.3.4	Les certificats numériques	33
	Conclusion	34
II	Analyse, Conception et implémentation	35
3	Analyse et Conception	36
	Introduction	36
3.1	Choix du cycle de développement	36
3.2	Orientation et faisabilité	38
3.3	Analyse de besoins	39
3.3.1	Cahier de charge	39
3.4	Budgétisation	41

3.5	Conception architecturale	41
3.6	Conception détaillée	42
3.6.1	Présentation de langage UML	42
3.6.2	Modélisation avec le langage UML	44
	Conclusion	63
4	Implémentation et Tests	64
4.1	Introduction	64
	Introduction	64
4.2	Environnement de développement	64
4.2.1	Environnement matériel	64
4.2.2	Environnement Logiciel	64
4.3	Développement du module	65
4.3.1	La structure générale de notre module	65
4.3.2	Les modules importés	67
4.4	Développement de l'interface utilisateur	67
4.4.1	La structure générale du projet	67
4.5	Tests	68
4.5.1	Test du module en console	68
4.5.2	Test de l'interface graphique du plate-forme	70
	Conclusion	71
5	Résultats	72
	Introduction	72
	Conclusion	72
	Conclusion	73
	Bibliographie	75
	Annexes	76
A	Code source du module python	77
A.1	le contenu de fichier setup.py	77
A.2	Contenu du fichier init	78

Résumé

La sécurité des systèmes de communications et des réseaux repose très largement sur des méthodes de cryptographie. En outre, les algorithmes de chiffrement sont toujours utilisés dans des processus incluant signature électronique, authentification et échanges de clés. Dans ce document nous allons approfondir le concept de la signature électronique en se basant sur l'infrastructure à clé publique et la création d'un module (ou bibliothèque) dans un langage de programmation qu'on précisera par la suite. En suite nous allons déployer dans une plate-forme web de signature électronique.

signature électronique, hachage, vérification

Abstract

The security of communications systems and networks is largely based on cryptography methods. In addition, encryption algorithms are always used in processes including electronic signature, authentication and key exchange. In this document, we will delve deeper into the concept of electronic signature based on the public key infrastructure and the creation of a module (or library) in a programming language that will later be specified. Then we will deploy in a web-based electronic signature platform.

electronic signature, hashing, verification

Liste des tableaux

2.1	Listes de noms par convention utilisé en cryptographie	19
3.1	La budgétisation	41
3.2	Description contextuelle de cas d'utilisation s'authentifier	49
3.3	Description contextuelle de cas d'utilisation générer certificat	50
3.4	Description contextuelle de cas d'utilisation générer paire de clé	51
3.5	Description contextuelle de cas d'utilisation Signer document	52
3.6	Description contextuelle de cas d'utilisation Envoyer Document	53
3.7	Description contextuelle de cas d'utilisation Chiffrer/Déchiffrer	54

Table des figures

1.1	Organigramme de administrative de l'entreprise ITS	4
1.2	Situation géographique du lieu de stage	8
2.1	Représentation graphique d'une signature	13
2.2	Schéma d'utilisation de la signature numérique	14
2.3	L'arbre hiérarchique de la cryptologie	18
2.4	Schéma d'utilisation de la cryptographie symétrique	20
2.5	Schéma d'utilisation de la cryptographie asymétrique	21
2.6	Schéma illustrative d'une fonction de hachage	27
2.7	Schéma qui illustre la fonction de hachage	29
2.8	Exemple de demande d'un certificat pour signer numériquement les e-mails de Pierre	32
3.1	Modèle de cycle en V	37
3.2	Architecture Générale Client Serveur	42
3.3	différentes vues du formalisme UML	43
3.4	Diagramme de cas d'utilisation général	45
3.5	Diagramme de cas d'utilisation d'administrateur	46
3.6	Diagramme de cas d'utilisation d'utilisateur	47
3.7	Diagramme de cas d'utilisation de système externe	48
3.8	Diagramme de classe	55
3.9	Diagramme de packages	56
3.10	Diagramme de séquence de cas d'utilisation s'authentifier	57
3.11	Diagramme de séquence de cas d'utilisation générer key	58
3.12	Diagramme de séquence de cas d'utilisation générer certificat	59
3.13	Diagramme de séquence de cas d'utilisation signer document	60
3.14	Diagramme de séquence de cas d'utilisation signer document	61

3.15	Diagramme de séquence de cas d'utilisation envoyer document	62
3.16	Diagramme de séquence de cas d'utilisation envoyer document	63
4.1	La création de l'environnement virtuelle	68
4.2	L'activation de l'environnement virtuelle	69
4.3	L'installation du module oudjirasign	69
4.4	Test du fonctionnalité générer paire de clef	69
4.5	Lancement du serveur intégré du framework python Flask	70
4.6	page d'accueil de la plate-forme de signature électronique	71

Introduction

« L'intégrité est une composante essentielle de la sécurité. Et pas seulement en informatique ! »

Didier Hallépée

Depuis l'antiquité jusqu'à nos jours, le document papier était notre support privilégié dès lors qu'il nous est nécessaire de conserver le témoignage d'un accord entre plusieurs parties. Traditionnellement, et à défaut de pouvoir en protéger l'intégrité, l'usage de sceaux ou de signatures, permet de garantir l'authenticité de tels documents.

Avec l'utilisation croissante des outils de communication « immatériels », que sont le téléphone, le fax ou encore l'Internet, le problème de la protection de nos échanges est devenu particulièrement critique.

Les progrès conjugués des mathématiques et de l'informatique ont permis, depuis les années 1970, de disposer progressivement d'un panel complet de solutions algorithmiques et de standards adaptés à la certification et signature électronique de nos documents électroniques.

Avec la criticité croissante de nos échanges et la disponibilité de solutions techniques avérées, la mise en place progressive d'un cadre juridique adapté est venue compléter l'ensemble.

Dans cette mémoire on va essayer de trouver les solutions juridiques concernant donner la valeur probante pour la signature électronique en donnant des exemples de la loi Camerounaise et Européenne. Et concevoir un module de signature électronique ensuite on déploiera dans une plate-forme web.

Première partie

Présentation générale

Chapitre 1

Contexte et problématique

Introduction

Dans ce chapitre de la première partie, nous présentons l'entreprise où nous avons effectué notre stage académique à savoir ITS. Nous parlerons ensuite du contexte relatif à notre sujet de stage de fin d'étude, nous terminerons par une mise en évidence de la problématique liée à ce contexte et les objectifs à atteindre.

1.1 Présentation de l'entreprise

1.1.1 Historique

ITS est une entreprise spécialisée dans la protection des systèmes d'informations, la sécurité informatique et la cryptologie de droit Camerounais dont le siège générale se trouve à Yaoundé-Cameroun BP : 8570, plus précisément à Byem-Assi dans le 6ème arrondissement du département de MFOUNDI. Elle commence ces services depuis 2008.

1.1.2 Organisation Administrative

Au sein de l'entreprise ITS on trouve une direction générale où se trouve le Directeur général et ses employés comme le montre l'organigramme suivant :

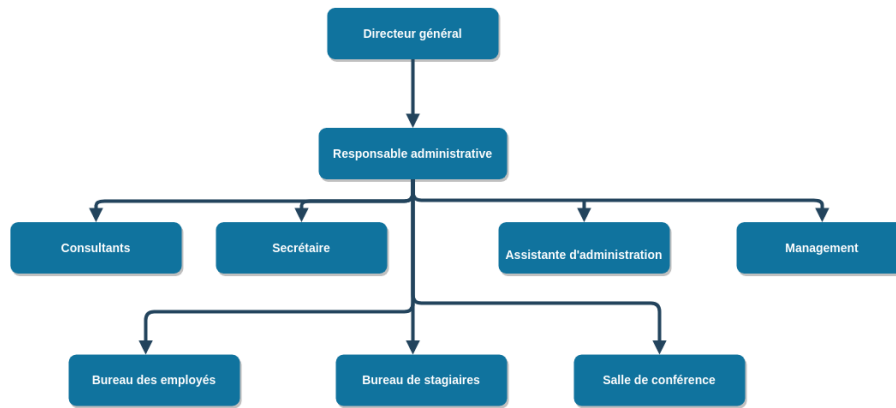


FIGURE 1.1 – Organigramme de administrative de l'entreprise ITS

1.1.3 Missions

ITS est une entreprise résolument tournée vers l'innovation et en constante croissance. La mission d'ITS est de fournir à toute les entreprises et organismes publics, quelle que soit leur taille, les solutions des outils cryptographiques, de sécurité réseaux et de sécurités de systèmes d'informations le plus performant du marché. ITS fondé en 2008, est une entreprise leader dans le domaine de cryptographie et de sécurité des systèmes d'informations au Cameroun. Les solutions apportées par ITS permettent à leurs clients d'avoir l'assurance qu'une faille de sécurité ne menacera jamais leurs activités. Ils peuvent donc consacrer totalement à leur croissance, car ces solutions les protègent efficacement contre les risques et les menaces informatiques. D'ailleurs sa réputation tient à la qualité des solutions de sécurités qu'elle met en place depuis sa création. Celles-ci intègrent les fonctionnalités essentielles suivantes :

- ✓ prévention d'intrusion (IDS),
- ✓ par-feu,
- ✓ protection antivirale et antispware,
- ✓ filtrage antispam et de contenu,
- ✓ PKI (Public Key Infrastructure),
- ✓ mobilité sécurité VPN,
- ✓ outils cryptographiques,
- ✓ etc.

Et enfin elle édifie les personnels des entreprises à travers de conférences, séminaires et webinaires et forme les intéressés dans plusieurs domaine de la sécurité informatique en vue de l'obtention de certificat à la fin de chaque formation.

1.1.4 Services et Produits

Entreprise ITS renferme plusieurs services, E-services et une centre de formations :

1.1.4.1 Sévices

Les services sont :

- ➡ **Sécurité des SI** : Aujourd'hui l'implémentation des technologies de l'information et de la communication engendre les problèmes de types nouveaux de sécurité des informations sensibles, des infrastructures et organisations mises en place. Ainsi le contrôle et la gestion du risque informationnel dans ces nouveaux systèmes deviennent indispensables pour le bon fonctionnement et même l'existence de ces derniers.
- ➡ **Investigation Numériques** : La cybercriminalité gagne du terrain avec la globalisation des systèmes d'information et l'intensification de leur utilisation dans tous les domaines d'activités de l'homme. La coopération internationale ne promet pas de résultats intéressants en même temps que les cybercriminels exploitent de mieux en mieux les technologies d'attaques disponibles, et ceci dans des conditions de partage d'expériences très bonnes. La police et les services de sécurité traînent le pas même si la législation permettant de combattre le fléau prend corps. La loi ne peut être efficace que si la preuve numérique du crime commis est à la disposition de la justice.
- ➡ **Audit des SI** : L'utilisation des systèmes d'information de plus en plus complexes et leur implémentation dans le contrôle et la gestion des processus sensibles imposent une normalisation visant la conformité de tous les systèmes d'information selon l'activité et le métier dans le but de mieux maîtriser le risque qu'engendre le système d'information dans le fonctionnement de toute organisation. Ainsi l'audit des systèmes d'information comme activité visant à mesurer le niveau de conformité d'un système d'information par rapport à des règles bien définies et à examiner le niveau de dérive du système par rapport à ces standards aide à anticiper sur les problèmes et à proposer les solutions pour y remédier avant même l'occurrence d'incidents.
- ➡ **Gouvernance des SI** : Toute activité nécessite un système de gouvernance solide et efficace pour s'assurer de la pérennité de cette dernière, ainsi que de l'atteinte des objectifs fixés au départ. Le système d'information ne s'aurait faire exception à cette règle, au contraire nécessite plus d'attention dans ce sens car complexe et indispensable pour toute entreprise. Les investissements faits pour son système d'information doivent se justifier non pas par un besoin simple, mais au moyen d'une étude préalable présentant le gain retour sur cet investissement par la création et l'exploitation des services liés à l'investissement en question.

1.1.4.2 E-Services

Dans les e-services on a plusieurs catégories :

1. **Web conférences** : rencontre internationale de Yaoundé sur la gestion du secret, l'usage de la cryptographie (Science du secret) dans la protection de l'information stratégique, la maîtrise des méthodes, moyens et systèmes de protection de l'information.
2. **Web séminaires**
 - (a) **webinaire¹ Protection d'informations stratégiques** : Cette formation a pour objectif, la maîtrise des techniques de sécurisation et protection des informations stratégiques, ainsi que l'étude des mécanisme de protection de données sensibles.
 - (b) **webinaire : Audit informatique** : Le but de la formation est la maîtrise des notions et techniques d'audit des systèmes d'information, ainsi que l'étude des cas selon une démarche spécifique.
3. Web consultations
4. E-Catalogue

1.1.4.3 Formations

Le centre de formation ITS est un centre de formation de référence spécialisée dans les modules de formation suivants :

- ➡ sécurité des système d'information ;
- ➡ investigations numériques ;
- ➡ audit des systèmes d'information ;
- ➡ gouvernance des systèmes d'information ;
- ➡ développement informatique ;
- ➡ infographie.

Les formations ont un cycle de douze mois soit neuf mois de cours et trois mois de pratique en entreprise. Le centre de formation CF-ITS est agréé par le MINEFOPE (Ministère de l'Emploi et de la Formation Professionnelle) et donc délivre les certificats de fin de formation conformément à la réglementation en vigueur. En marge des formations dans les salles.

Le centre donne l'accès gratuit et illimité aux forums et discussions via le portail web linkedin (www.linkedin.com) avec un nombre illimité d'experts internationaux du domaine de la cybercriminalité, cybersécurité, et investigation numériques, gouvernance et audit des systèmes d'information.

1. Webinaire est un mot-valise associant les mots web et séminaire, créé pour désigner toutes les formes de réunions interactives de type séminaire faites via internet généralement dans un but de travail collaboratif ou d'enseignement à distance

1.1.4.4 Certifications

ITS offre également des certifications suivantes :

1. Certification Professionnelle Nationale (MINEFOP)
 - ⇒ Sécurité des systèmes d'Informations
 - ⇒ Investigations numériques
 - ⇒ Audit des systèmes d'information
 - ⇒ Gouvernance des systèmes d'information
2. Certification Professionnelle Internationale (ISACA)
 - ⇒ CISM - Certified Information Security Manager
 - ⇒ CISA - certified Information Systems Auditor
 - ⇒ CRISC - Certified in Risk and Information Systems Control
 - ⇒ CGEIT-Certified in Governance of Enterprise IT
3. Certification Professionnelle Internationale (CISCO)

1.1.4.5 Produits

L'entreprise a mis à la disposition de tout le monde des logiciels gratuits, on peut citer entre autres les logiciels suivants :

- ⇒ Utilitaire de désinstallation d'antivirus **AVAST**,
- ⇒ Logiciel de calcul de l'empreinte numérique des fichiers,
- ⇒ Calcul du hash code des fichiers par **MD5**,
- ⇒ Logiciel d'**EBIOS**,
- ⇒ Logiciel de stéganographie,
- ⇒ **Ntop** - Logiciel de **monitoring** du réseau.

Et des logiciels payants :

- ⇒ Logiciel de récupération de mots de passe,
- ⇒ etc.

1.1.5 Cadre de stage

On a effectué notre stage au sein de la direction générale de l'entreprise, elle se trouve dans le quartier **Biyem-Assi** dans la ville de Yaoundé, département de MFOUNDI plus précisément situé en plein cœur du 6^e arrondissement. Leur direction se trouve dans l'**Avenue Biyem-Assi** à côté de **Pharmacie Les Béatitudes**.

1.1.5.1 Situation géographique

La figure ci-dessous présente l'emplacement de notre lieu de stage réalisé grâce au service du géant Google **Google Map**.

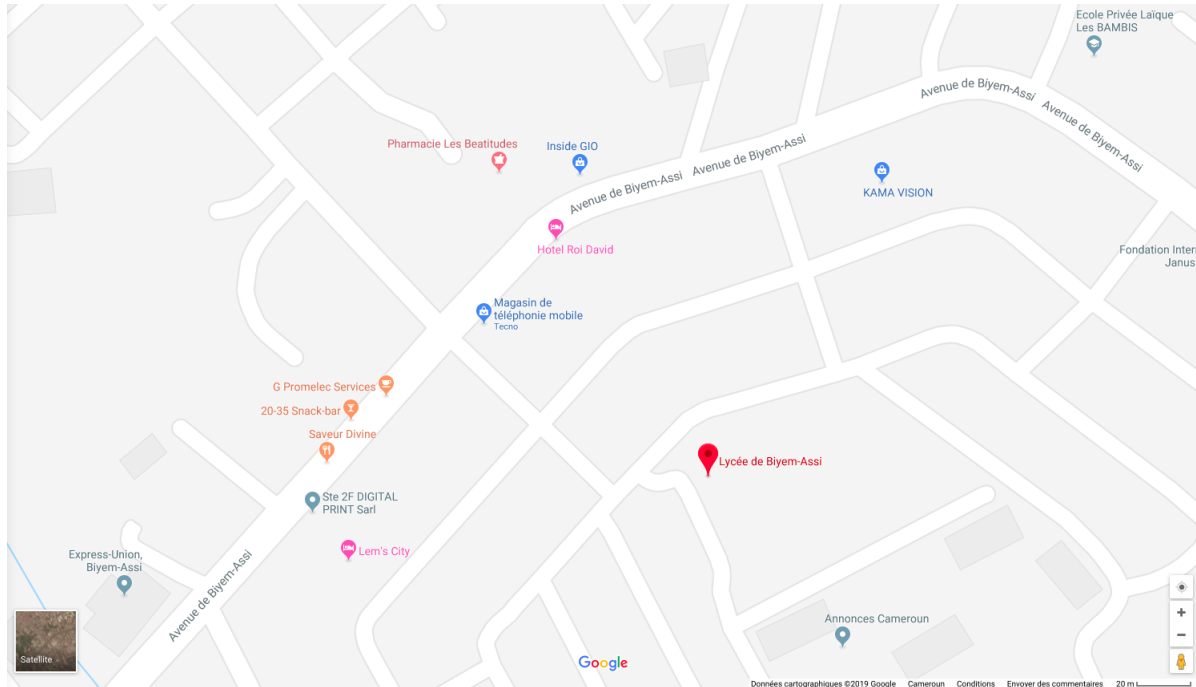


FIGURE 1.2 – Situation géographique du lieu de stage

1.2 Contexte

ITS, étant une entreprise qui s'exerce dans le domaine de la sécurité des systèmes d'informations, elle a plusieurs clients issue de différents secteurs : privé, publique, organisationnelle et gouvernementale. Les échanges se font soit via le réseau peu sûr, soit l'entreprise envoie quelqu'un pour récupérer le contrat, l'entreprise signe l'accord et renvoi, soit on envoie un consultant pour aller chez le client et signer l'accord sur place. **Comment ça marche les trois processus pré-cités ?**

- **via le réseau peu sûr** : lorsqu'il y a un accord entre l'entreprise et le client, généralement le client envoie les documents à signer via un service de messagerie : Yahoo, G-mail, etc. À la réception, l'entreprise imprime le document, signe, scanne et renvoie par le même canal de transmission. Le client fait de même, il télécharge le fichier, imprime et signe à son tour. Enfin, le client fait un scan et renvoie pour la confirmation à l'entreprise.

Cette solution est moins sûre puisqu'elle est exposée à des personnes malveillantes (l'homme du milieu, par exemple) et plusieurs transactions sont émises de part et d'autre pour une seule signature. Par conséquent, ça ne résout pas les urgences.

- **via un coursier** : généralement c'est l'entreprise qui envoie un coursier, il part récupérer le document à signer, l'entreprise signe et renvoie le document par ce même service de coursier. Le client étant conscient, il signe aussi et le coursier rentre avec la confirmation de document signé.

Cette solution est plus sûre que la précédente, mais le fait qu'on loue un service pour la course de ce document, il y'a d'abord le problème de confiance, le document peut se perdre en route ce qui fait un double travail et ça entraîne les dépenses à l'entreprise.

- **via un consultant ou le directeur général** : ici, c'est presque la même démarche que la dernière, mais par contre c'est un consultant ou le directeur général qui fait le déplacement car ils ont des clients partout dans le Cameroun et en Afrique. L'accord est alors signé sur place par les deux parties.

Ici, on constate que c'est le moyen le plus sûr par rapport aux deux autres mais il peut y avoir plusieurs clients d'urgences au même moment, du coup ça entraîne l'indisponibilité de l'entreprise pour ces clients.

Toutes ces démarches de signature d'un d'accord ont l'air de bien marcher mais l'entreprise trouve tout cela fastidieux.

Quelle que soit la solution citée si haut choisie, les ressources humaines, matérielles et surtout financières sont importantes.

1.3 Problématique

Le processus de signature de document a de très graves répercussions sur l'entreprise :

- ⇒ La ressource matérielle est très utilisée pour faire signer un accord avec le client ;
- ⇒ La ressource financière est extrêmement utilisée pour payer les coursiers et autres ;
- ⇒ La ressource humaine est très utilisée pour faire signer un document, puisqu'il faut envoyer un consultant ou le directeur général lui-même qui fait le déplacement ;
- ⇒ La perte de temps ;
- ⇒ la disponibilité de l'entreprise dans les cas des clients d'urgences au même moment ;

1.4 Objectifs

1.4.1 Objectif général

L'objectif principal de notre projet est d'offrir à l'entreprise et toute personne désirant signer un document électronique, un outil de signature électronique de manière flexible et

transparent.

1.4.2 Objectif spécifique

De façon spécifique, il sera question pour nous de concevoir un outil de signature électronique en se base sur l'architecture à clé publique capable de :

- ⇒ créer une signature des documents électronique (texte, son, vidéo, PDF, etc.) ;
- ⇒ automatiser de la création des signatures électronique ;
- ⇒ vérifier la signature de document électronique ;
- ⇒ prouver l'authenticité d'un document ;
- ⇒ prouver l'identité d'un signataire ;
- ⇒ faciliter à l'entreprise lors d'un accord avec le client de signer un document électronique ;
- ⇒ générer une paire de clef RSA ;
- ⇒ rendre la vie facile à tous les utilisateurs désirant signer un document numérique ;

1.5 Méthodologie

Pour atteindre ces objectifs, nous proposons à l'entreprise la conception et le déploiement d'un module de signature électronique basé sur l'infrastructure à clé publique (PKI). Nous allons suivre le cheminement de la conception dont les grandes étapes sont suivantes :

1. L'analyse des besoins du client qui débouchera sur l'élaboration d'un cahier de charges ;
2. La conception du système qui débouchera sur la modélisation des différentes vues du système ;
3. L'implémentation logicielle qui est le codage proprement dit des différents éléments du système ;
4. Et enfin nous procéderons au déploiement, aux tests et validation.

Conclusion

Dans ce chapitre de la première partie nous avons présenté le contexte et la problématique liés à notre thème et les services et produits que proposent ITS. En énumérant les problèmes que rencontre l'entreprise et les clients lors de la signature d'un documents numérique que nous proposons de résoudre par la mise en place de la conception et déploiement d'un module

de signature électronique en se basant sur la infrastructure à clé publique. Dans le chapitre suivant, nous présenterons les généralités liés à la signature électronique en particulier et la cryptographie en générale.

Chapitre 2

Présentation générale de la signature électronique

Introduction

Dans ce chapitre nous allons mettre en évidence les concepts générales sur la signature électronique. Ensuite nous donnerons une idée générale sur la procédure ou la mécanisme de la signature électronique dans sa vue globale et les différentes méthodes peuvent intervenir pour la rendre encore plus sûre. Donc nous allons présentés premièrement les généralités sur la signature numérique en soulignant son rôle ainsi que les différents technologie qui interviennent.

2.1 Concept général sur la signature électronique

2.1.1 Historique

La signature électronique est régit par deux normes[Fro13] : ISO 14533 au niveau international et eiDAS[Cos18] (Electronic Identification And trust Services) au niveau européen. Cette dernière est entré en vigueur en 2014, elle régit des normes juridiques pour l'authentification électronique et entretien un service de confiance à l'égard de toutes les transactions électroniques.

Par ailleurs, la norme ISO 14533 garanti l'interopérabilité des signatures électroniques lorsque les documents qu'elles authentifient sont transformées et traités par des systèmes d'informations différents. Cette norme est composée de deux parties :

- ⇒ ISO 14533-1 : publié en 2012 et révisée en 2014, intitulée : Processus, éléments d'informations et documents dans le commerces, industrie et l'administration - Profils de signature à long terme - Partie 1 : Profils de signature à long terme pour les signature électronique

avancée CMS (CADES) (Cryptographic Message Syntax, Advanced Electronic Signatures),

- ⇒ ISO 14533-2 : publié en 2012, intitulée : Processus, éléments d'informations et documents dans le commerces, industrie et l'administration - Profils de signature à long terme - Partie 2 : Profils de signature à long terme pour les signature électronique avancée XML (XAES) (XML Advanced Electronic Signatures).

On distingue alors trois formats de signatures électroniques[Wik18] :

1. le format PAdES (PDF Advanced Electronic and Services) est une norme de l'ETSI¹ permettant de signer des documents PDF en signature jointe, avec un seul fichier,
2. le format CAdES est un ensemble d'extensions au standard de signature Cryptographic Message Syntax (CMS) le rendant compatible avec la signature électronique avancée, ici on peut signer les documents comme Word, Excel, txt, image et PDF en signature disjointe avec 2 fichiers,
3. et le format XAdES est un ensemble d'extensions à la norme XML-DSig² qui la rendent compatibles avec la signature électronique avancée.

2.1.2 Signature

Pour mieux **savoir ce** que c'est une signature électronique, nous parlerons de ce qu'on entend par signature. **C'est quoi une signature ?**



FIGURE 2.1 – Représentation graphique d'une signature

Définition 2.1.2.1. La signature est le graphisme par lequel une personne s'identifie dans un acte et, par lequel elle exprime son approbation au contenu de ce document. La validité de tout engagement est subordonné à l'existence de cette signature manuscrite qui confère au document sa force probatoire.

1. L'European Telecommunications Standards Institute, c'est-à-dire l'Institut européen des normes de télécommunications, est l'organisme de normalisation européen du domaine des télécommunications.

2. XML Signature (aussi nommé XMLDsig, XML-DSig, XML-Sig) est une recommandation du W3C destinée à permettre l'utilisation de signatures numériques dans les documents XML.

La figure 2.1 est la représentation graphique d'une signature dite manuscrite, son utilisation est faite soit sur les documents physiques soit sur les documents numériques de façon claire, c'est-à-dire visible à l'œil nu, c'est-à-dire qui est exposé au faux, falsification etc, des signatures d'autrui.

2.1.2.1 Faux et usage de faux en signatures.

Imiter la signature d'un individu sur un document, même si la signature imitée ne ressemble en rien à la signature officielle de la personne lésée, peut constituer un délit de faux et usage de faux et d'usurpation d'identité, ainsi que d'autres délits répertoriés dans le code civil ou pénal : arnaque, escroquerie, vol de biens, de droits, vol d'identité, détournement de fonds, d'héritage, etc.

Les procédés utilisés par les faussaires sont nombreux, étant souvent très difficiles à identifier. On peut évoquer l'imitation manuelle, plus ou moins réussie, le faux par décalquage, la photo-composition physique ou numérique, ainsi que l'imitation de fantaisie, ou le résultat ne comporte aucune ressemblance par rapport à la vraie signature de la victime.

On peut constater que les risques de falsification de signature manuscrite sont énormes, pour palier à ces problèmes on a une autre manière de signer les documents numériques de façon totalement transparente. Bien sûr il s'agit de la signature numérique.

2.1.3 Signature électronique

À présent voyons voir ce que c'est une signature numérique (dite aussi électronique).

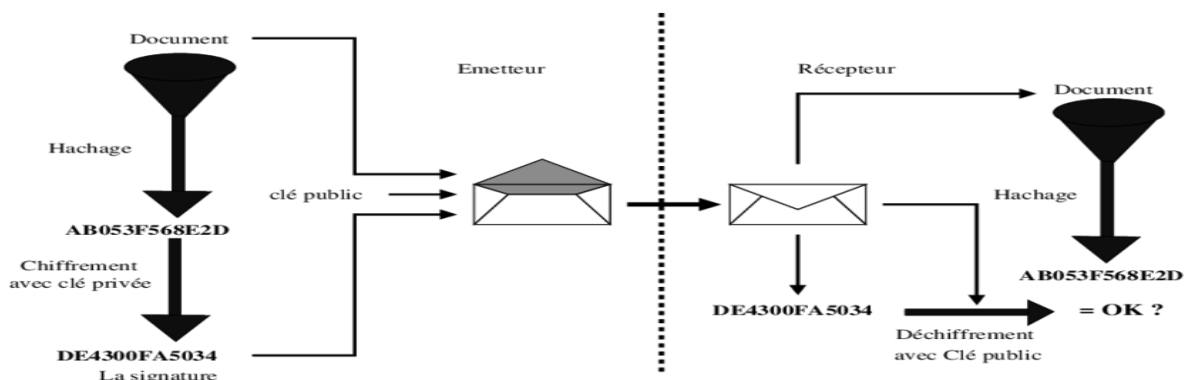


FIGURE 2.2 – Schéma d'utilisation de la signature numérique

Source d'image :³

2.1.3.1 Qu'est-ce que la signature électronique ?

Cette question, qui semble pourtant basique, est au cœur du débat. Car avant de l'adopter, encore faut-il savoir de quoi il s'agit concrètement. Avant toute chose, il est important de

comprendre ce que la signature électronique n'est pas, et surtout de couper court à certaines croyances populaires : en l'occurrence, la signature électronique n'est pas un « scan » de la signature manuscrite[CERin]. Ceci étant précisé, **qu'est-ce que la signature électronique alors ?**

Définition 2.1.3.1. La signature électronique (dite aussi signature numérique) est un processus, utilisant des mécanismes de cryptologie, permettant de garantir l'intégrité d'un document électronique et d'en authentifier l'auteur, par analogie avec la signature manuscrite d'un document papier[Brain].

Définition 2.1.3.2. La signature numérique constitue la forme de signature électronique la plus avancée et sécurisée. Vous pouvez l'utiliser pour vous conformer aux dispositions réglementaires et juridiques les plus strictes, car elle offre les niveaux de garantie les plus élevés sur l'identité des signataires et l'authenticité des documents qu'ils signent.

Les signatures numériques utilisent un identifiant numérique basé sur un certificat délivré par une autorité de certification ou un prestataire de services de confiance accrédité. Ainsi, lorsque vous signez numériquement un document, votre identité vous est associée de manière exclusive, la signature est liée au document par cryptage, et tout peut être vérifié à l'aide d'une technologie sous-jacente appelée l'infrastructure à clé publique[CERin]..

Ce dernier point représente généralement la plus grande difficulté de compréhension au non-initié. Car si l'outil nécessaire à la signature manuscrite n'est ni plus ni moins qu'un stylo, les outils de signature électronique sont multiples, autant que les moyens techniques nécessaires à leur réalisation. Concrètement, il s'agit dans la majorité des cas d'un certificat numérique porté sur différents supports (carte à puce, clé USB, carte d'identité, PC, smart-phone, etc.) et qui a pour fonction d'identifier le signataire d'une part, et de sceller le document pour en garantir l'intégrité d'autre part.

2.1.3.2 La signature numérique a-t-elle la même valeur juridique que la signature manuscrite ?

La signature électronique a été introduite dans le droit français par la loi du 13 mars 2000[Jeain]. Elle dispose des mêmes prérogatives et engage le consentement du signataire de la même façon que la signature manuscrite, sous réserve de « l'usage d'un procédé fiable d'identification garantissant son lien avec l'acte auquel elle s'attache », selon l'article 1367 du Code Civil.

Mais elle va encore plus loin que la signature manuscrite : en scellant l'ensemble du document lors de son apposition, elle en garantit l'intégrité, c'est-à-dire l'état précis, au moment de l'engagement du consentement par le signataire. Un peu comme si l'on paraphait chaque lettre

ou chaque ponctuation d'un document papier ! En d'autres termes, la signature électronique profite non seulement de la même valeur juridique que la signature manuscrite, mais elle est aussi beaucoup plus sécurisante pour toutes les parties.

Par ailleurs, le recours à une signature numérique permet d'éviter que le signataire nie avoir signé (propriété de non-répudiation). Si un signataire nie une signature numérique valide, c'est soit que sa clé privée a été compromise, soit qu'il ment. Dans de nombreux pays, notamment aux Etats-Unis, les signatures numériques ont la même valeur juridique que les signatures classiques.

2.1.3.3 Rôle de signature électronique

Typiquement, dans un monde où n'importe quel cybercriminel⁴ peut usurper l'adresse email d'un contact et envoyer des courriers malveillants et des documents infectés en son nom, la signature électronique joue un rôle clé dans la sécurisation des échanges. Si un contact signe systématiquement tous les courriers qu'il envoie, la réception d'un courrier non signé de sa part est un signe d'usurpation. D'une façon similaire, signer un document permet d'en assurer l'authenticité et l'origine puisque l'opération est datée, que le créateur est authentifié et que la non-répudiation ne permet pas d'en modifier l'origine.

Mais la signature d'un document ne cherche pas uniquement à authentifier son créateur ou son expéditeur. Elle garantit aussi l'intégrité de ce document. Et cette garantie devient de plus en plus nécessaire dans un monde connecté où les malwares⁵ peuvent modifier des documents et où des cybercriminels peuvent être payés, par des concurrents par exemple, pour venir altérer à votre insu des documents et vous mettre en situation délicate.

La signature est présentée comme indispensable pour authentifier un document. Elle permet au lecteur d'identifier la personne et l'organisme qui l'a émis. Elle apporte de la confiance dans l'environnement numérique, la confiance dans le signataire et dans le contenu de l'information[Pré16].

La signature électronique est un mécanisme permettant de garantir l'intégrité d'un document électronique et d'en authentifier l'auteur, par analogie avec la signature manuscrite d'un document papier.

Cependant elle se différencie de la signature écrite par le fait qu'elle n'est pas visuelle : l'identification ne repose sur aucun élément graphique.

4. Homme qui commet un crime à l'aide d'outils informatiques, notamment en piratant des données existantes sur Internet afin d'obtenir illégalement de l'argent ou un quelconque profit.

5. Le malware est la contraction des termes anglais malicious et software. Il désigne un logiciel malveillant s'attaquant aux ordinateurs, terminaux mobiles et objets connectés

2.1.3.4 Comment la signature électronique fonctionne ?

Concrètement suivant la Figure 2.2 , on commence par générer une empreinte grâce au fonction mathématique dite de hachage ou encore cette empreinte est appelée le condensé du message que l'on souhaite envoyer. Ensuite on chiffre le l'empreinte par la clé privée de l'émetteur. Le couple message original et le chiffré de l'empreinte constitue le message signer par l'émetteur. Cette empreinte servira de vérifier l'intégrité de message source.

Du côté du destinataire, une fois qu'il a reçu le message signer c'est à dire le chiffré de empreinte et le message en clair, il procède de la même manière en générant l'empreinte du message clair en utilisant la même fonction mathématique de hachage, ensuite il décrypte l'empreinte reçu et il compare les deux hachés, s'il y'a égalité alors le message n'a pas été altéré sinon c'est dire que les deux hachés ne correspondent pas alors le message à été altéré durant son parcours vers le destinataire.

2.2 Cryptographie

La cryptographie utilise des concepts issus de nombreux domaines (Informatique, Mathématiques, Électronique). Toutefois, les techniques évoluent et trouvent aujourd'hui régulièrement racine dans d'autres branches (Biologie, Physique, etc.)[Dum10].

2.2.1 Terminologie

- ⇒ La **Cryptologie** est la science des messages secrets. Elle se décompose en deux disciplines : la cryptographie et la cryptanalyse[San12]. Dans la figure ci-dessous on voit que la cryptographie est le centre de débat de la cryptologie, ce qui fait que dans cette section on va seulement nous intéressé à la cryptographie.

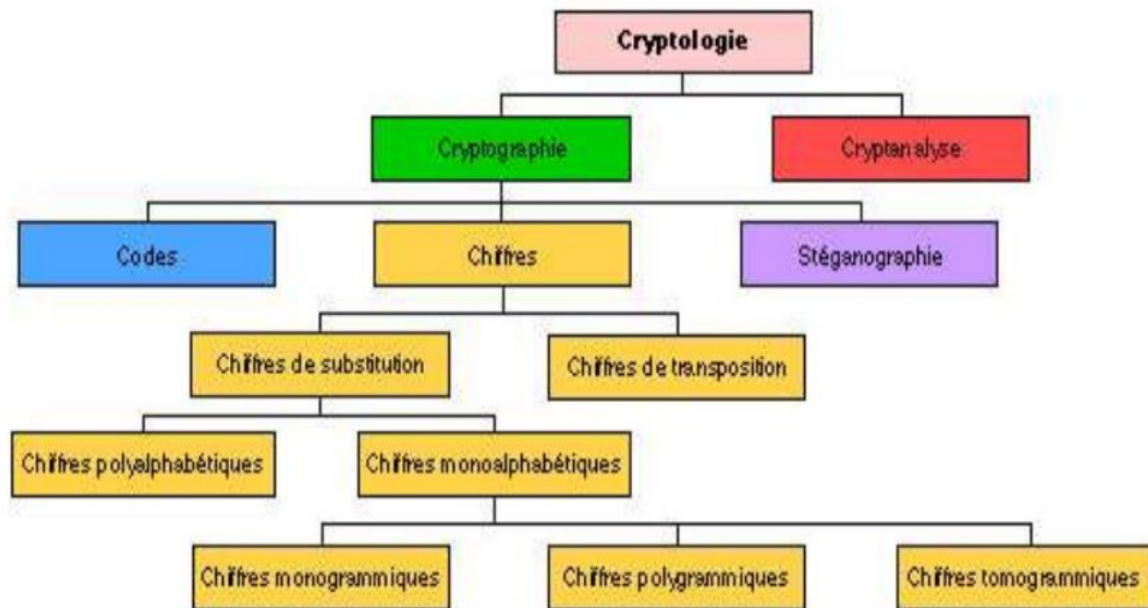


FIGURE 2.3 – L'arbre hiérarchique de la cryptologie

- ⇒ **Cryptographie** : Art de transformer un message clair en un message inintelligible. Cependant, on utilise souvent le cryptographie comme synonyme de cryptographie.
- ⇒ **Cryptanalyse** : Art d'analyser un message chiffré afin de le décrypter. On parle aussi de décryptement.
- ⇒ **Chiffre** (ou chiffrement) : Ensemble de procédés et ensemble de symboles (lettres, nombres, signes, etc.) employés pour remplacer les lettres du message à chiffrer. Bien souvent appelé cryptage, un mot qui provient d'anglicisme "encryption".
- ⇒ **Code** Ensemble de procédés et ensemble des symboles (lettres, nombres, signes, etc.) employés pour remplacer les mots du message à coder.
- ⇒ **Stéganographie** : Branche particulière de la cryptographie qui consiste non pas à rendre le message inintelligible, mais à le camoufler dans un support (texte, image, etc.) de manière à masquer sa présence.
- ⇒ **Déchiffrement** : Opération inverse du chiffrement : obtenir la version originale d'un message qui a été précédemment chiffré en connaissant la méthode de chiffrement et les clefs.
- ⇒ **Décryenunrateptement** (ou décryptage) : Restauration des données qui avaient été chiffrées à leur état premier, sans disposer des clefs théoriquement nécessaires.
- ⇒ **Texte en clair** : c'est le message à protéger [ram09].
- ⇒ **Texte chiffré** : c'est le résultat du chiffrement du texte clair.

2.2.2 Buts de la cryptographie

Globalement, la cryptographie permet de résoudre quatre problèmes différents :

1. **La confidentialité** : Le texte chiffré ne doit être lisible que par les destinataires légitimes. Il ne doit pas pouvoir être lu par un intrus.
2. **L'authentification** : Le destinataire d'un message doit pouvoir s'assurer de son origine. Un intrus ne doit pas être capable de se faire passer pour quelqu'un d'autre.
3. **L'intégrité** : Le destinataire d'un message doit pouvoir vérifier que celui-ci n'a pas été modifié en chemin. Un intrus ne doit pas être capable de faire passer un faux message pour légitime.
4. **La non répudiation** : Un expéditeur ne doit pas pouvoir, par la suite, nier à tort avoir envoyé un message ainsi que le destinataire ne doit pas pouvoir nier à tort avoir reçu un message.

Remarque : Il ne faut pas confondre **cryptographie** et **codage**. En effet, la cryptographie va s'attacher à cacher le sens d'un texte ; Le codage quant à lui s'attache à modifier le texte de façon à utiliser un support particulier. Ainsi par exemple, il existe le code **Morse** qui transforme les chiffres et lettres d'un texte en une succession de traits et de points afin de pouvoir utiliser le support télégraphique. Il existe aussi le code **ASCII** qui transforme les caractères en valeurs codées sur 8 bits (de 0 à 255) pour utiliser les supports informatiques. Il n'y a strictement rien de secret dans un codage.

2.2.2.1 Alice, Bob et autres

Traditionnellement, pour illustrer les protocoles, on parle de communication entre des personnes fictives. Pour la cryptographie, par convention, ces personnes sont :

TABLE 2.1 – Listes de noms par convention utilisé en cryptographie

Français	Anglais	Rôle
Alice	Alice	Alice veut envoyer un message à Bob
Bob	Bob	Bob communique avec Alice
Christine	Carol	S'il faut un 3ème pour communiquer avec Alice et Bob c'est Christine
David	Dave	S'il faut un 4ème pour communiquer avec Alice et Bob c'est David
...

2.2.3 Cryptographie symétrique

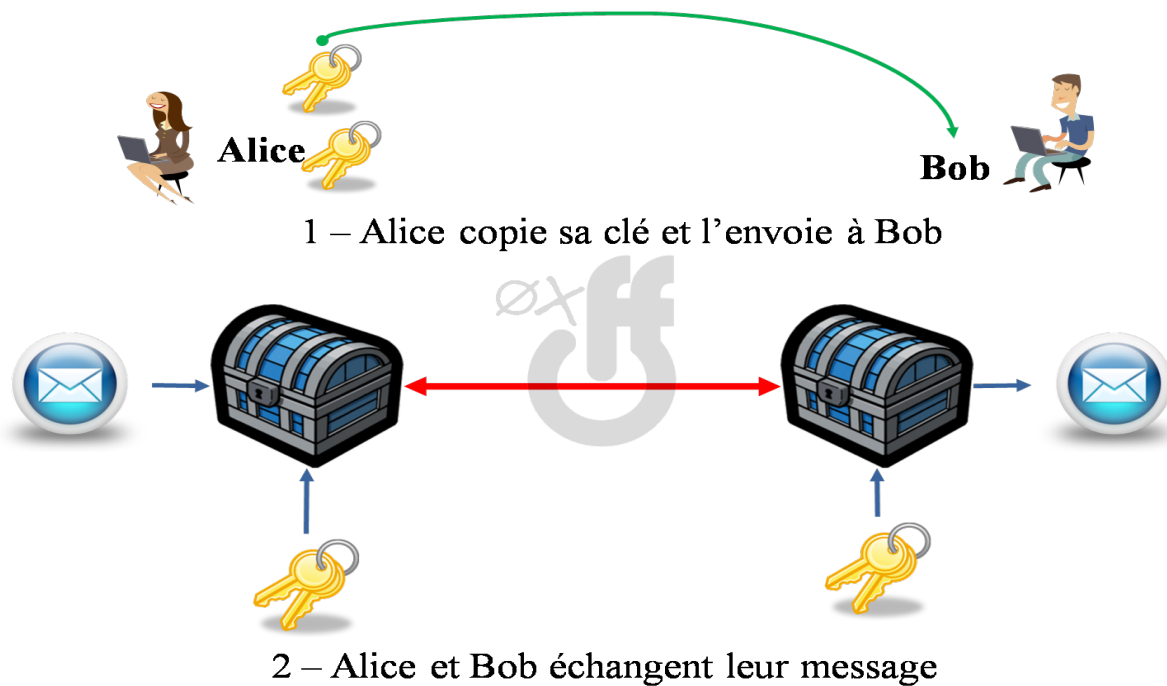


FIGURE 2.4 – Schéma d’utilisation de la cryptographie symétrique

Source d’image :⁶

La cryptographie symétrique, également dite à clé secrète (par opposition à la cryptographie à clé publique), est la plus ancienne forme de chiffrement. On a des traces de son utilisation par les Égyptiens vers 2000 av. J.-C. Plus proche de nous, on peut citer le chiffre de Jules César, dont le ROT13 est une variante.

2.2.3.1 Clé et sécurité

L’un des concepts fondamentaux de la cryptographie symétrique est la clé. Une clé est une donnée qui (traitée par un algorithme) permet de chiffrer et de déchiffrer un message. Toutes les méthodes de cryptage n’utilisent pas de clé. Le ROT13, par exemple, n’a pas de clé. [Wikin]Quiconque découvre qu’un message a été codé avec cet algorithme peut le déchiffrer sans autre information. Une fois l’algorithme découvert, tous les messages chiffrés par lui deviennent lisibles.

Si l’on modifiait le ROT13 en rendant le décalage variable, alors la valeur de ce décalage deviendrait une clé, car il ne serait plus possible de crypter et décrypter sans elle. L’ensemble des clés possibles comporterait alors 26 décalages[Wikin].

Cet exemple montre le rôle et l’importance de la clé dans un algorithme de chiffrement ; et les restrictions qu’elle implique. Auguste Kerckhoffs (La cryptographie militaire, 1883) énonce le principe de Kerckhoffs : pour être sûr, l’algorithme doit pouvoir être divulgué. En outre, il

faut aussi que la clé puisse prendre suffisamment de valeurs pour qu’une attaque exhaustive — essai systématique de toutes les clés — soit beaucoup trop longue pour être menée à bien. Cela s’appelle la sécurité calculatoire[Wikin].

Cette sécurité calculatoire s’altère avec le progrès technique, et la puissance croissante des moyens de calcul la fait reculer constamment. Exemple : le DES, devenu obsolète à cause du trop petit nombre de clés qu’il peut utiliser (pourtant 2^{56}). Actuellement, 2^{80} est un strict minimum. [Wikin]À titre indicatif, l’algorithme AES, dernier standard d’algorithme symétrique choisi par l’institut de standardisation américain NIST en décembre 2001 [Wikin], utilise des clés dont la taille est au moins de 128 bits soit 16 octets, autrement dit il y en a 2^{128} . Pour donner un ordre de grandeur sur ce nombre, cela fait environ $3,4 * 10^{38}$ clés possibles ; l’âge de l’univers étant de 1010 années, si on suppose qu’il est possible de tester 1 000 milliards de clés par seconde (soit $3,2 * 10^{19}$ clés par an), il faudra encore plus d’un milliard de fois l’âge de l’univers. Dans un tel cas, on pourrait raisonnablement penser que notre algorithme est sûr. Toutefois, l’utilisation en parallèle de très nombreux ordinateurs, synchronisés par internet, fragilise la sécurité calculatoire[Wikin].

2.2.4 Cryptographie asymétrique

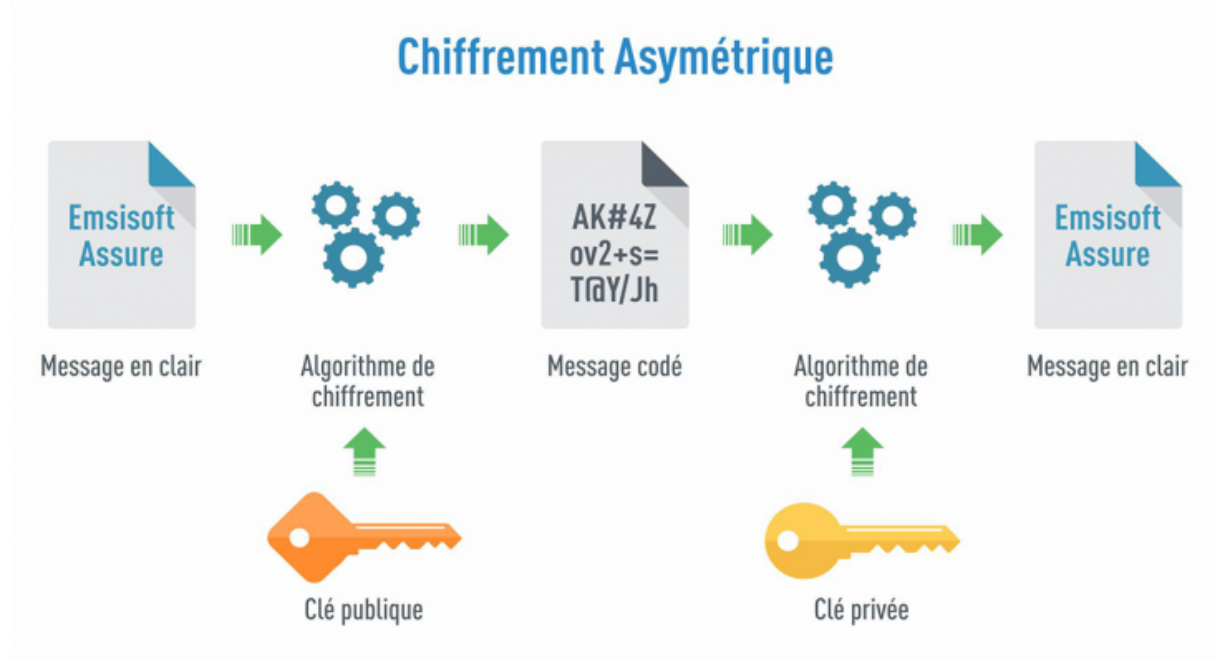


FIGURE 2.5 – Schéma d’utilisation de la cryptographie asymétrique

Source d’image :⁷

La cryptographie asymétrique qui, comme son nom l’indique, est une méthode utilisée

pour transmettre et échanger des messages de façon sécurisée en s'assurant de respecter les principes suivants :

- ⇒ Authentification de l'émetteur,
- ⇒ Garantie d'intégrité,
- ⇒ Garantie de confidentialité

Cette technique repose sur le principe de « paire de clés » (ou bi-clés) composée d'une clé dite « privée » conservée totalement secrète et ne doit être communiquée à personne et d'une clé dite « publique » qui, comme son nom l'indique peut être transmise à tous sans aucune restriction.

Les clés dites asymétriques sont des clés de chiffrement. Le chiffrement étant le nom général donné aux techniques mathématiques de codage ou de décodage des données.

Pour garantir la confidentialité[Rou18], l'intégrité, l'authenticité et la non-répudiabilité du chiffrement asymétrique, les utilisateurs et les systèmes doivent s'assurer qu'une clé publique est authentique, qu'elle appartient bien à la personne ou à l'entité qui la revendique et qu'elle n'a pas été falsifiée ni remplacée par un tiers malveillant. Il n'existe aucune solution idéale à ce problème d'authentification de la clé publique. L'approche la plus courante consiste en une infrastructure à clé publique (PKI, Public Key Infrastructure) que nous verrons dans la section 2.3 , par laquelle des autorités de certification de confiance se portent garantes de la propriété des paires de clés et des certificats, tandis que les produits de chiffrement basés sur le modèle Pretty Good Privacy (PGP), dont OpenPGP, utilisent un système d'authentification décentralisé appelé « Web of Trust » (WOT), qui repose sur des approbations individuelles du lien entre un utilisateur et une clé publique[Rou18].

2.2.4.1 La genèse de la cryptographie asymétrique

Whitfield Diffie et Martin Hellman, chercheurs à l'Université de Stanford, ont été les premiers à proposer publiquement le chiffrement asymétrique dans leur article de 1977 intitulé « New Directions in Cryptography »[Rou18]. Quelques années auparavant, le concept avait déjà été proposé de façon indépendante et en secret par James Ellis, qui travaillait pour le service de renseignement et de sécurité britannique, le Government Communications Headquarters (GCHQ), Clifford Cocks aurait décrit dès 1973 ce qu'on appelle l'algorithme RSA et Malcolm J. Williamson aurait inventé un protocole d'échange de clef très proche de celui de Diffie et de Hellman dès 1974. L'algorithme asymétrique, tel qu'il est exposé dans l'article de Diffie-Hellman, utilise des nombres élevés à des puissances spécifiques pour produire des clés de déchiffrement[Rou18].

Dans leur article de 1976, W. Diffie et M. Hellman n'avaient pas pu donner l'exemple d'un système à clef publique[Wik19], n'en ayant pas trouvé. Il fallut attendre 1978 pour avoir un exemple donné par Ronald Rivest, Adi Shamir et Leonard Adleman, le RSA, abréviation tirée des trois noms de ses auteurs. Les trois hommes fondèrent par la suite la société RSA Security. Le système Merkle-Hellman est généralement considéré comme la première réalisation pratique d'un système de chiffrement à clef publique, il a cependant été prouvé non sûr par Shamir en 1982[Wik19].

2.2.4.2 L'application de la cryptographie à clé publique

De nombreux protocoles[Rou18], tels que SSH, OpenPGP, S/MIME et SSL/TLS reposent sur la cryptographie asymétrique pour leurs fonctions de chiffrement et de signature numérique. Cette technique est également utilisée dans des logiciels, tels que les navigateurs, qui ont besoin d'établir une connexion sécurisée sur un réseau peu sûr comme Internet ou de valider une signature numérique. La puissance du chiffrement est directement liée à la taille de la clé. De ce fait, un doublement de la longueur de la clé renforce le chiffrement de façon exponentielle, au détriment toutefois des performances. Cependant, à mesure que la puissance de traitement augmente et que des algorithmes de factorisation plus efficaces sont découverts, il devient possible de factoriser des nombres de plus en plus élevés[Rou18].

L'algorithme RSA (Rivest-Shamir-Adleman), le plus répandu, est intégré au protocole SSL/TLS qui sert à assurer la sécurité des communications sur un réseau informatique. Si RSA offre autant de sécurité, c'est parce qu'il est très difficile de factoriser de grands entiers qui sont eux-mêmes le produit de deux grands nombres premiers. Multiplier deux grands nombres premiers est facile, mais la sécurité par cryptographie à clé publique repose sur la difficulté qu'il y a à déterminer les nombres d'origine à partir du total (la factorisation). En effet, on considère que le temps nécessaire pour factoriser le produit de deux nombres premiers suffisamment élevés dépasse les capacités de la plupart des pirates, à l'exception des organismes d'Etat, les seuls susceptibles d'avoir accès à une puissance de traitement de grande envergure. Les clés RSA font généralement 1024 ou 2048 bits, mais les experts pensent que les clés de 1024 bits pourraient être décryptées à brève échéance. C'est la raison pour laquelle l'administration comme le secteur privé commencent à adopter des clés d'une longueur minimale de 2048 bits[Rou18].

2.2.4.3 Fonctionnement : Principe général

La cryptographie asymétrique, ou cryptographie à clef publique est fondée sur l'existence des fonctions à sens unique et à brèche secrète.

Les fonctions à sens unique sont des fonctions mathématiques telles qu'une fois appliquées à un message, il est extrêmement difficile de retrouver le message original.

L'existence d'une brèche secrète permet cependant à la personne qui a conçu la fonction à sens unique de décoder facilement le message grâce à un élément d'information qu'elle possède, appelé clef privée.

Supposons qu'Alice souhaite recevoir un message secret de Bob (voir la figure 2.5) sur un canal susceptible d'être écouté par un attaquant passif Eve :

- ⇒ Alice transmet à Bob une fonction à sens unique pour laquelle elle seule connaît la brèche secrète ;
- ⇒ Bob utilise la fonction transmise par Alice pour chiffrer son message secret ;
- ⇒ Alice réceptionne le message chiffré puis le décode grâce à la brèche secrète ;
- ⇒ Si Eve réceptionne également le message alors qu'il circule sur le canal public, elle ne peut le décoder, même si elle a également intercepté l'envoi de la fonction à sens unique, car elle n'a pas connaissance de la brèche secrète.

La terminologie classiquement retenue est :

- ⇒ pour la fonction à sens unique : "clef publique" ;
- ⇒ pour la brèche secrète : "clef privée".

En pratique, sont utilisées des fonctions de chiffrement classiques, les termes "clef publique" et "clef privée" correspondant alors à des paramètres employés pour ces fonctions.

2.2.4.4 Fonctionnement pratique

Alice souhaite pouvoir recevoir des messages chiffrés de n'importe qui.

- ⇒ Elle génère alors une valeur à partir d'une fonction à sens unique et à brèche secrète à l'aide d'un algorithme de chiffrement asymétrique, par exemple RSA.
- ⇒ Alice diffuse à tout le monde la fonction pour coder les messages (notée clef publique) mais garde secrète la fonction de décodage (notée clef privée).
- ⇒ L'un des rôles de la clef publique est de permettre le chiffrement ; c'est donc cette clef qu'utilisera Bob pour envoyer des messages chiffrés à Alice ;
- ⇒ L'autre clef — l'information secrète — sert à déchiffrer. Ainsi, Alice, et elle seule, peut prendre connaissance des messages de Bob. La connaissance d'une clef ne permet pas de déduire l'autre.
- ⇒ D'autre part, l'utilisation par Alice de sa clef privée sur le condensat d'un message, permettra à Bob de vérifier que le message provient bien d'Alice : il appliquera la clef publique d'Alice au condensat fourni (condensat chiffré avec la clef privée d'Alice) et retrouve donc le condensat original du message. Il lui suffira de comparer le condensat ainsi obtenu et

le condensat réel du message pour savoir si Alice est bien l'expéditeur. C'est donc ainsi que Bob sera rassuré sur l'origine du message reçu : il appartient bien à Alice. C'est sur ce mécanisme notamment que fonctionne la signature numérique 2.1.3.4.

2.2.4.5 Mécanismes d'authentification

Un inconvénient majeur de l'utilisation des mécanismes de chiffrement asymétriques est le fait que la clef publique est distribuée à toutes les personnes : Bob, Carole et Alice souhaitant échanger des données de façon confidentielle. De ce fait, lorsque la personne possédant la clef privée, Alice, déchiffre les données chiffrées, elle n'a aucun moyen de vérifier avec certitude la provenance de ces données (Bob ou Carole) : on parle de problèmes d'authentification.

Afin de résoudre ce problème, on utilise des mécanismes d'authentification permettant de garantir la provenance des informations chiffrées. Ces mécanismes sont eux aussi fondés sur le chiffrement asymétrique dont le principe est le suivant : Bob souhaite envoyer des données chiffrées à Alice en lui garantissant qu'il en est l'expéditeur[Wik19].

1. Bob crée une paire de clefs asymétriques : il définit une clef privée et diffuse librement sa clef publique (notamment à Alice)
2. Alice crée une paire de clefs asymétriques : elle définit une clef privée et diffuse librement sa clef publique (notamment à Bob)
3. Bob effectue un condensat de son message "en clair" puis chiffre ce condensat avec sa clef privée
4. Bob chiffre une seconde fois son message déjà chiffré avec la clef publique d'Alice
5. Bob envoie alors le message chiffré à Alice
6. Alice reçoit le message chiffré de Bob (mais qu'un tiers, par exemple Ève, pourrait intercepter)
7. Alice est en mesure de déchiffrer le message avec sa clef privée. Elle obtient alors un message lisible sous forme de condensat. Eve quant à elle ne peut pas déchiffrer le message intercepté de Bob car elle ne connaît pas la clef privée d'Alice. En revanche Alice n'est pas sûre que le message déchiffré (sous forme de condensat) est bien celui de Bob
8. Pour le lire, Alice va alors déchiffrer le condensat (chiffré avec la clef privée de Bob) avec la clef publique de Bob. Par ce moyen, Alice peut avoir la certitude que Bob est l'expéditeur. Dans le cas contraire, le message est indéchiffrable et elle pourra présumer qu'une personne malveillante a tenté de lui envoyer un message en se faisant passer pour Bob

Cette méthode d'authentification utilise la spécificité des paires de clefs asymétriques : si l'on chiffre un message en utilisant la clef publique[Wik19], alors on peut déchiffrer le message

en utilisant la clef privée ; l'inverse est aussi possible : si l'on chiffre en utilisant la clef privée alors on peut déchiffrer en utilisant la clef publique.

2.2.4.6 Sécurité

Un chiffrement symétrique au moyen d'une clef de 128 bits propose 2^{128} (3, 410³⁸) façons de chiffrer un message. Un pirate qui essaierait de déchiffrer le message par la force brute devrait les essayer une par une [Wik19].

Pour les systèmes à clef publique, il en va autrement. Tout d'abord les clefs sont plus longues (par exemple 1 024 bits minimum pour RSA) ; en effet, elles possèdent une structure mathématique très particulière (on ne peut pas choisir une suite de bits aléatoire comme clef secrète, par exemple dans le cas du RSA, seuls les nombres premiers sont utilisés). Certains algorithmes exploitant cette structure sont plus efficaces qu'une recherche exhaustive sur, par exemple [Wik19], 1 024 bits. Ainsi, dans le cas de RSA, le crible général des corps de nombres est une méthode plus efficace que la recherche exhaustive pour la factorisation.

Il faut noter le développement actuel de la cryptographie utilisant les courbes elliptiques, qui permettent (au prix d'une théorie et d'implémentation plus complexes) l'utilisation de clefs nettement plus petites que celles des algorithmes classiques (une taille de 160 bits étant considérée comme très sûre actuellement), pour un niveau de sécurité équivalent.

Dans son édition du 6 septembre 2013, le journal The Guardian affirmait que la NSA était capable de déchiffrer la plupart des données chiffrées circulant sur Internet [Wik19]. De nombreuses sources ont cependant indiqué que la NSA n'avait pas mathématiquement cassé les chiffrements mais s'appuierait sur des faiblesses d'implémentation des protocoles de sécurité.

2.2.5 Hachage

Les fonctions de hachage sont un outil informatique et mathématique devenu incontournable à tel point que vous utilisez plusieurs fonctions de hachage chaque jour sans forcément vous en rendre compte. Nous nous intéresserons ici aux fonctions de hachage cryptographiques.

Les procédures de signature précédentes ont un coût prohibitif pour signer des longs messages car la signature est aussi longue que le message. On double donc la longueur du texte à crypter.

Pour produire la longueur de la signature on peut utiliser une fonction de hachage cryptographique (hash function en anglais), h , ou fonction de condensation. C'est une fonction à sens unique qui sera publique dans les applications.

Mais qu'est-ce qu'une fonction de hachage cryptographique ? Quelles sont les caractéristiques qu'une bonne fonction doit avoir ? Comment peut-on en attaquer

une ?

2.2.5.1 Définition

Une fonction de hachage, c'est en fait tout simple. Cela consiste en une fonction qui transforme une donnée quelconque en une donnée de taille fixée. C'est tout.

Bien entendu, la donnée en question peut avoir plusieurs formes. Ce peut être du texte, une image, ... mais dans tous les cas la donnée sera transformée en un texte binaire avant qu'on lui applique la fonction de hachage.

Plus concrètement, une fonction de hachage est une fonction mathématique qui permet de convertir une valeur numérique d'une certaine taille dans une valeur numérique d'une autre taille[Quéin]. On pourrait comparer la fonction de hachage à une presse dans laquelle est inséré un objet, qui une fois compressé ressort avec une taille plus petite, mais toujours identique, quelque soit la taille de l'objet inséré[Dan10].

Le schéma de calcul d'une signature avec une fonction de hachage est le suivant [Dan10] :

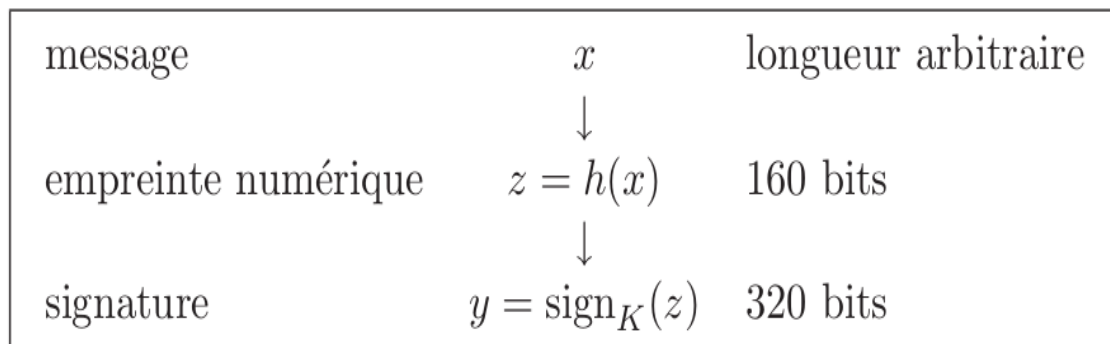


FIGURE 2.6 – Schéma illustrative d'une fonction de hachage

En réalité une fonctions de hachage prend un message x de taille inférieure à N fixé qu'elle transforme en une empreinte de taille 160 bits (ou 256 ou 384 ou 512)[Dan10].

2.2.5.2 Construction des fonctions de hachage

La fonction de hachage doit être construite avec soin pour qu'elle n'affaiblisse pas le protocole de signature. Un opposant comme Martin ne doit pas pouvoir forger la signature d'Alice.

Comme une fonction de hachage n'est évidemment pas injective, il existe des couples de messages x' et x tels que $h(x') = h(x)$. L'attaque la plus évidente consiste pour Martin à

partir d'un message signé (x, y) authentique ($y = \text{sign}_K(x)$) précédemment calculé par Alice à calculer $z = h(x)$ et à chercher $x' \neq x$ tel que $h(x) = h(x')$. Si Martin y parvient (x', y) est un message valide [Dan10].

Donnons tout d'abord quelques définitions de qualités que l'on peut exiger d'une fonction de hachage.

Définition 2.2.5.1. Une fonction de hachage h est à **collisions faibles difficiles** si étant donné un message x , il est calculatoirement difficile d'obtenir un message $x' \neq x$ tel que $h(x) = h(x')$

Définition 2.2.5.2. Une fonction de hachage h est à **collisions fortes difficiles** s'il est calculatoirement difficile d'obtenir deux messages différents x' et x tel que $h(x) = h(x')$

Définition 2.2.5.3. Une fonction de hachage est une **fonction de hachage à sens unique** si étant donnée une empreinte numérique z , il est calculatoirement difficile de trouver un message x tels $h(x) = z$

Si une fonction de hachage est à collisions fortes difficiles elle est bien sûr à collisions faibles difficiles, mais aussi à sens unique [Dan10].

2.2.5.3 A quoi sert une fonction de hachage en cryptographie ?

Si l'on veut garantir l'authenticité de l'émetteur d'un document, on utilise ce que l'on appelle une signature électronique. Par la suite, on appelle H la fonction de hachage. On appellera C l'action de chiffrer et D déchiffrer.

- ⇒ A veut envoyer un message M à B
- ⇒ A crée un haché du document $H(M)$
- ⇒ A chiffre $H(M)$ avec sa clé privée : $C(H(M))$
- ⇒ A envoie M et $C(H(M))$
- ⇒ B récupère tout ça
- ⇒ B crée un haché du document $H(M)$
- ⇒ B utilise la clé publique de A et compare $H(M)$ à $D(C(H(M)))$

Alors, si c'est bien A qui a envoyé les documents, $D(C(H(M))) = H(M)$ et B est certain de l'émetteur.

Résumé sur l'image ci-dessous (source : wikipédia) :

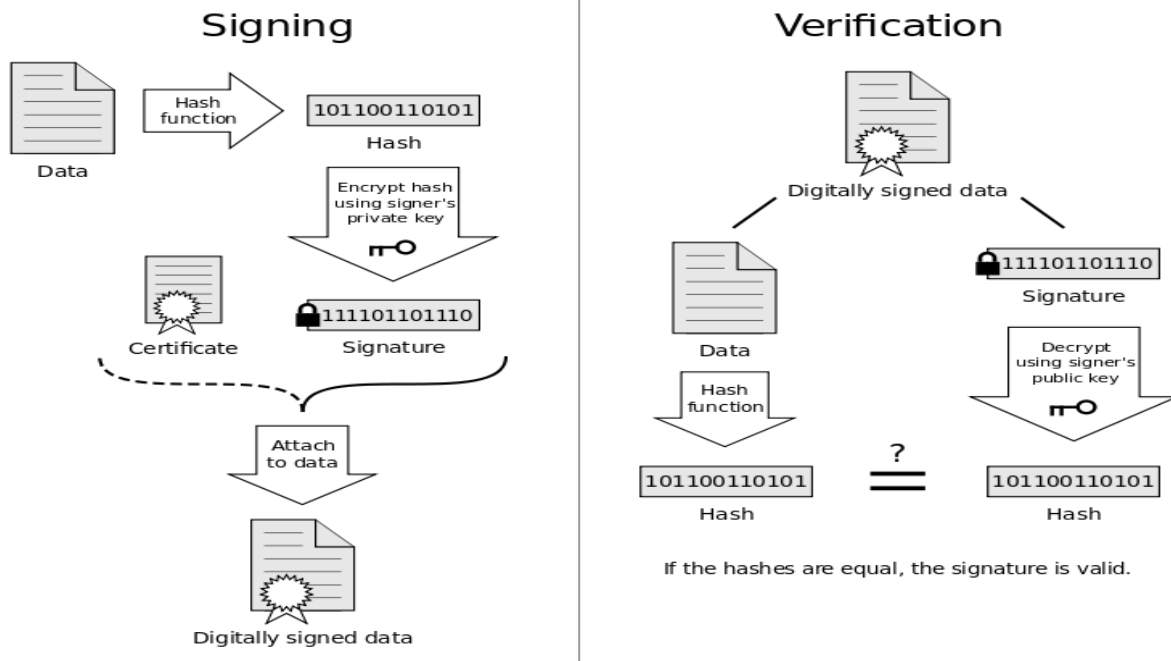


FIGURE 2.7 – Schéma qui illustre la fonction de hachage

2.3 Infrastructure à clé publique

Définition 2.3.0.1. L'utilisation massive de messages électroniques et l'expansion du commerce électronique est devenu un opération de plus en plus courante. En effet, les données qui transitent sur internet, sont sujettes à diverses attaques comme *Man in the middle* lorsque les entités échangent leurs clefs publiques. Dans un petite structure, il pourrait être envisageable de générer sa paire de clefs localement et d'échanger les clefs publiques hors ligne (en main propre par exemple), mais cette solution est inimaginable pour une structure internationale. Dans de cas de figure, une authentification automatique des clefs est indispensable.

C'est dans ce contexte que la NIST (National Institute ans Technology) s'est vu imposer en 1994 la tâche d'étudier et de définir un standard afin de gérer l'authentification dans un environnement international. Ce projet avait pour but de permettre l'interopérabilité des différents systèmes électroniques opérant dans le commerce électronique. L'étude était porté sur la manière de générer les clefs, de les distribuer, d'obtenir les clefs publiques au moyen de certificats, et la publication des certificats obsolètes.

A la même façon qu'un passeport ou d'une carte d'identité, la PKI⁸ va fournir une garantie d'identité numérique aux utilisateurs. Cette pièce d'identité numérique, appelée cer-

8. Public Key Infrastructure

tificat numérique, contient la clef publique de l'utilisateur, mais également des informations personnelles sur l'utilisateur. Comme tout document formel, le certificat numérique est signé par l'autorité de certification et c'est cette signature qui lui donnera toute crédibilité aux yeux des utilisateurs. Le certificat numérique est largement publié, il n'a pas à être tenu secret, il est consultable via un annuaire.

Pour obtenir un certificat numérique, le client doit effectuer une requête auprès d'un organisme reconnu. Il transmet avec sa requête sa clef publique. L'organisme construit un certificat incorporant la clef publique du client, il signe le certificat à l'aide de sa clef privée. L'autorité de certification publiera le certificat signé comportant la clef publique et l'identité précise du propriétaire, quiconque consultera ce certificat aura l'assurance dans l'authenticité de la clef publique contenue dans celui-ci car il a confiance dans l'autorité de certification qui a délivré ce certificat.

2.3.1 La gestion des clefs

La gestion des clefs de l'infrastructure doit être rigoureuse. En effet, il a été démontré dans les faits qu'il est beaucoup plus facile de s'introduire dans un système et de se procurer illicitement les clefs plutôt que de casser un algorithme[Monin]. Et le moment le plus propice pour espérer se procurer les clefs est sans conteste le moment où l'échange des clefs a lieu. C'est pourquoi, l'échange des clefs doit être fait avec la plus grande prudence car il représente le point de vulnérabilité de tout le système.

La gestion des clés proprement dite se compose des opérations suivantes[Monin] :

- ⇒ **Génération** : Les clefs doivent être générées de manière aléatoirement, de manière à ce qu'elle soient non prédictibles.
- ⇒ **Distribution** : La distribution est l'action de déplacer une clef de cryptage. Un exemple de distribution est la clef de session, on va créer une clef qui va permettre le transport d'une autre clef.
- ⇒ **Suppression** : La suppression de clefs intervient quand la clef à atteint sa fin de validité ou lorsqu'un doute subsiste sur sa confidentialité.
- ⇒ **Archivage** : L'archivage des clefs permet de conserver une copie des clefs même si elles ne sont plus utilisées.
- ⇒ **Recouvrement** : Le recouvrement des clefs est une procédure délicate qui permet de retrouver la clef privée d'un client.

Toutes ces étapes doivent être minutieusement effectuées et contrôlées pour que la PKI ne soit pas sujette à diverses attaques.

2.3.2 Les composants d'un PKI

Une PKI[VIL06] contient plusieurs composants principaux essentiels à son bon fonctionnement :

- **Une Autorité d'Enregistrement (Registration Authorities)** : c'est cette autorité qui aura pour mission de **traiter les demandes de certificat** émanant des utilisateurs et de générer les couples de clés nécessaires (clé publique et clé privée). Son rôle peut s'apparenter à la préfecture lors d'une demande de carte d'identité.
- **Une Autorité de Certification (Certification Authorities)** : elle reçoit de l'Autorité d'Enregistrement les demandes de certificats accompagnées de la clé publique à certifier. Elle va **signer à l'aide de sa clé privée** les certificats, un peu à la manière de la signature de l'autorité sur une carte d'identité. Il s'agit du composant le plus critique de cette infrastructure en raison du degré de sécurité requis par sa clé privée.
- **Une Autorité de Dépôt (PKI Repositories)** : il s'agit de l'élément chargé de **diffuser les certificats numériques signés** par la CA sur le réseau (privé, Internet, etc).
- **Les utilisateurs de la PKI** : ce sont les personnes effectuant des demandes de certificat mais aussi ceux qui souhaitent vérifier l'identité d'un certificat qu'ils ont reçu.

On peut résumer tout ceci par le schéma suivant (source [VIL06]) :

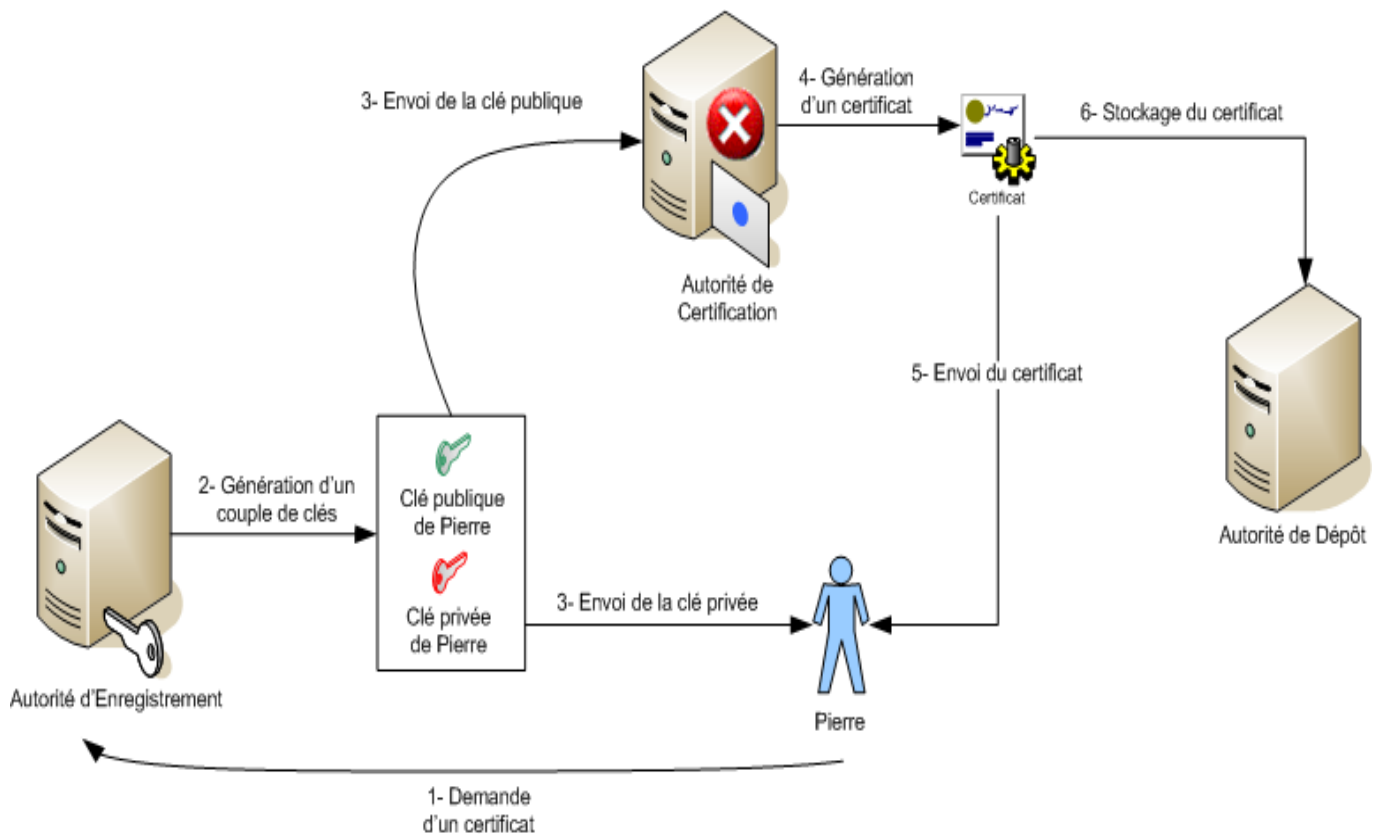


FIGURE 2.8 – Exemple de demande d'un certificat pour signer numériquement les e-mails de Pierre

2.3.3 Quel est le cadre juridique qui régit l'exercice des activités électroniques ou de la certification au Cameroun

Les activités de Certification électroniques sont régies par les textes ci-après :

- ⇒ Loi n°2010/012 relative à la cybercriminalité au Cameroun qui fixe le cadre légal définissant le régime des activités de la certification électronique en ses article 10, 11, 13 et 14 ;
- ⇒ Décret 2012/1318/PM du 22 mai 2012 fixant les conditions et les modalités d'octroi de l'autorisation d'exercice de l'activité de certification électronique ;
- ⇒ Décret 2013/0400/PM du 27 février 2013, fixant les modalités de déclaration et d'autorisation préalables, ainsi que les conditions d'obtention du certificat d'homologation en vue de la fourniture, l'exportation, l'importation ou l'utilisation des moyens ou des prestations de cryptologie ;
- ⇒ Arrêté conjoint n°000 00013/minpostel/minfi du mai 2013 les montants et les modalités

de paiement de frais perçus par l'Agence Nationale des Technologies de l'information et de la Communication.

2.3.3.1 Qui délivre les certificats électroniques

Seule une autorité de certification agréée a qualité de délivrer un certificat électronique qualifié.

Pour l'instant, le secteur public camerounais est desservi par l'ANTIC qui assure le rôle d'autorité de certification de l'Administration Publique.

A l'avenir, les autorités de certification qui seront accréditées par l'ANTIC pourront certifier, elle aussi, les autres personnes(physique et morales) issus du secteur privé et de la société civile. Il convient de relever que, les autorités de certifications étrangères qui seront reconnues comme telles à travers une convention de reconnaissance mutuelle signée du Ministre chargé des Télécommunications pourront également émettre des certificats électroniques valables au Cameroun.

2.3.4 Les certificats numériques

Un certificat numérique (aussi appelé certificat électronique) est un fichier permettant de certifier l'identité du propriétaire d'une clé publique, un peu à la manière d'une carte d'identité. Un certificat est généré dans une infrastructure à clés publiques 2.3 (aussi appelé PKI pour Public Key Infrastructure) par une autorité de certification (Certification Authority , CA) qui a donc la capacité de générer des certificats numériques contenant la clé publique en question.

Actuellement, les certificats numériques sont reconnus à la norme X.509 version 3. Ce format se compose entre autre de [VIL06] :

- ⇒ la version du certificat X.509 (actuellement la V3)
- ⇒ le numéro de série
- ⇒ l'algorithme de signature
- ⇒ le nom de l'émetteur (autorité de certification)
- ⇒ la date de début de fin de validité
- ⇒ l'adresse électronique du propriétaire
- ⇒ la clé publique à transmettre
- ⇒ le type de certificat
- ⇒ l'empreinte du certificat (signature électronique)

La signature électronique est générée par l'autorité de certification à l'aide d'informations personnelles (telles que le nom, le prénom, l'adresse e-mail, le pays du demandeur, etc) en utilisant sa propre clé privée.

2.3.4.1 Les différents sortes des certificats numériques

Il existe de nombreux types de certificats numériques, répondant chacun à un besoin particulier. Les principaux types sont[VIL06] :

- ⇒ Certificat de messagerie (permet de crypter et de signer ses e-mails)
- ⇒ Authentification IPsec pour un accès distant par VPN
- ⇒ Authentification Internet pour les pages Web sécurisées
- ⇒ Cryptage des données avec EFS
- ⇒ Signature de logiciel

2.3.4.2 Pourquoi utiliser un certificat numérique ?

Un certificat numérique intervient dans différents mécanismes permettant de sécuriser l'échange de données sur un réseau[VIL06]. On y retrouve le cryptage asymétrique (cf. section 2.2.4, page 21) ou encore la signature électronique (cf. section 2.1.3, page 14) combinée à un contrôle d'intégrité des données.

Conclusion

Dans ce dernier chapitre de la première partie, on a présenté le concept de la signature numérique et en comparant avec la signature manuscrite. On a aussi vu qu'elle a la même valeur juridique (cf. section 2.1.3.2, page 15) que la signature manuscrite. En suite on a vu comment fonctionne la signature numérique tout en évoquant les éléments qui sont indispensables à la signature numérique tels que le chiffrement symétrique, le chiffrement asymétrique, la fonction de hachage et en terminant avec l'infrastructure à clé publique (cf. section 2.3, page 29). Ainsi s'achève la première partie. Dans le premier chapitre de la deuxième partie on va entrer dans la conception, analyse et implémentation proprement dite et on finira avec le dernier chapitre concernant les tests.

Deuxième partie

Analyse, Conception et implémentation

Chapitre 3

Analyse et Conception

Introduction

Une fois que la présentation des concepts liée à notre projet est faite, nous allons effectuer l'analyse et la conception de notre module de signature électronique. Pour cela, nous effectuerons premièrement le choix de cycle de vie de développement d'un logiciel qui nous convient, deuxièmement nous procéderons à l'analyse de besoins, ensuite nous ferons la conception proprement dite **et enfin nous terminerons par l'implémentation du système.**

3.1 Choix du cycle de développement

Le cycle de vie d'un logiciel indique les étapes par lesquelles doit passer un logiciel de sa conception jusqu'à sa mort. Ce cycle de vie permet de détecter les erreurs tout au long du processus de réalisation et ainsi les corriger pour produire un logiciel de qualité, maîtriser les délais de sa réalisation et les coûts associés. Le cycle de vie d'un logiciel comprend généralement les étapes suivantes :

1. **Pré-étude** : Cette étape permet de définir les objectifs du projet de définir le domaine d'activité. Les questions posées sont : **Quoi ? , Combien ? et Quoi ?**
En entrée, on a les besoins et en sortie on a un cahier de charges.
2. **Analyse** : Cette consiste à recueillir les informations et formaliser les besoins du client, de définir les contraintes et d'estimer la faisabilité de ce besoins. La question à poser est : **que fait le système ?**
En entrée, on a le cahier de charge et en sortie on a le dossier d'analyse.
3. **Conception** : Cette étape permet d'élaborer la structure générale du système et de définir chaque sous-ensemble de logiciel à produire. La question à poser est : **comment faire ce qu'il est demandé de faire ?**

En entrée, on a le dossier d'analyse et en sortie on a un dossier de conception.

4. **Codage** : Cette étape consiste à coder ou à programmer les fonctionnalités définies dans la phase de conception.

En entrée, on a le dossier de conception en sortie on a des programmes.

5. **Tests** : Cette étape permet de tester le logiciel conformément aux spécifications (fonctionnelle ou non fonctionnelle). Il existe quatre types de tests à savoir : le test unitaire, le test d'intégration, le test fonctionnel et le test de validation.

6. **Réception** : Cette étape permet au client de vérifier la conformité de logiciel avec les spécifications initiales.

En entrée on a un logiciel plus un cahier de charge et en sortie on a un procès verbal de réception(acceptation ou refus du livrable).

7. **Maintenance** : Cette étape permet de prendre en charge les actions collectives du système (**Maintenance collective et évolutive**).

En entrée on a un logiciel et en sortie on a un logiciel modifié.

Nous avons vu quelles sont les étapes clés dans le cycle de vie d'une application. Afin d'obtenir un résultat optimal, il est conseillé de suivre cette démarche qui peut subir des améliorations[MONin].

Il existe plusieurs modèles de cycle de vie à savoir : le cycle en cascade, le cycle en V, le cycle en spirale, le cycle semi-itératif mais dans notre cas on a décidé de suivre le modèle du cycle en V.

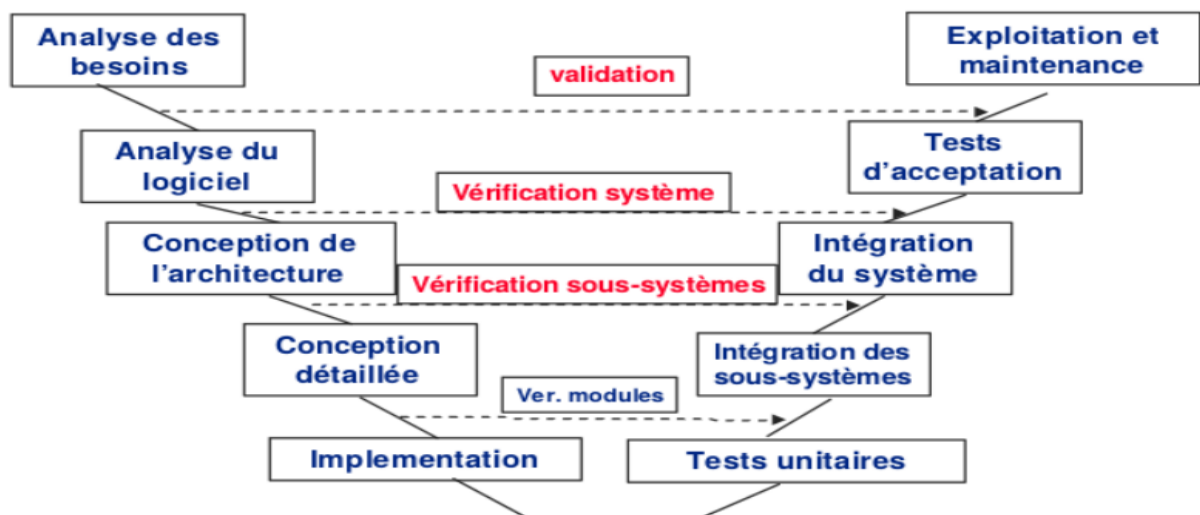


FIGURE 3.1 – Modèle de cycle en V

3.1.0.1 Pourquoi cycle en V ?

Il est introduit en **1997**. La version actuelle est désignée de **V-Modell XT**(depuis février 2005). Cette méthode consiste en un modèle pour la planification et le développement de logiciel[Kol06]. Le modèle V répond à quatre requête importantes de développement : **Who ? What ? When ? How ?**

L'on peut poser la question comme suit : **Who has to do what, when, and how within a project ? Qui doit faire quoi, quand et comment dans un projet ?** Cette question est capital lors de la construction d'un logiciel.

Par ailleurs, dans le modèle V, le fond réalise l'implémentation. La branche gauche définit les différentes spécifications. La branche droite crée des corrélations avec la partie gauche à savoir la validation du système, la vérification système et du logiciel.

L'accent est mis sur la réduction des erreurs. La branche droite permet en général la détection rapide des erreurs et des anomalies présents dans la partie gauche et entreprend des mesures adéquates pour les corriger. Et en plus, la finalité de cycle de vie en V consiste à parvenir sans incident à livrer un logiciel totalement conforme au cahier des charges[Kol06]. Voilà pourquoi nous avons choisi le modèle V.

3.2 Orientation et faisabilité

Le projet intitulé « Conception et déploiement d'un module de signature numérique basé sur l'architecture à clé publique »est né du fait que l'entreprise ITS utilise des moyens de signature d'un document à l'aide des méthodes qui sont les unes sont traditionnelles d'autres sont coûteux, ce qui pénalise l'entreprise. Et aussi, on a constater qu'il n'y a pas un outil de signature électronique disponible et intégra

Par ailleurs, bien qu'ils existent des tels outils tels que Word de Microsoft, Adobe Reader etc, mais ne sont pas accessible de tous, dans les systèmes d'exploitation libre(open source) comme Linux par exemple.

L'environnement dans lequel nous nous trouvons est favorable au projet, bien que des tels outils existent mais sous une licence payante et complexe. Sa mise en œuvre sera une innovation importante dans l'évolution numérique au sein de l'entreprise, dans le monde pourquoi pas et surtout dans notre cher pays puisqu'on peut l'étendre dans beaucoup d'autres préoccupations, comme l'authentification des documents administratives, l'authentification des diplômes (Bac, BTS, DUT, Licence, Master, etc.) etc.

L'utilisation de cette outil, bénéficiera à ITS de :

⇒ permet d'avoir son propre outil de signature électronique ;

- ⇒ Signer désormais ces documents numériques ;
- ⇒ gagner en temps et l'argent ;
- ⇒ servir rapidement ;

Ce projet d'inscrit donc largement dans le cadre de la sécurité de système d'information qui est aujourd'hui très capital tant pour les entreprises que les utilisateurs.

3.3 Analyse de besoins

Dans cette section, nous allons recueillir les besoins du demandeur (le client) et l'ensemble des contraintes liés au système à mettre sur pied.

3.3.1 Cahier de charge

Le cahier de charge a pour but de présenter notre projet su la signature électronique basé sur l'architecture à clé publique des documents numériques de l'entreprise ITS et autres tel que spécifié dans la section précédente.

3.3.1.1 Besoins fonctionnels

Il est question ici de présenter toutes les fonctionnalités du système de signature électronique. Ce sont les besoins spécifiant un comportement d'entrée/sortie du système. Le système doit permettre à :

1. l'Administrateur de :

- ⇒ gérer les comptes (créer, supprimer, modifier) ;
- ⇒ générer une paire de clé(publique et privée) ;
- ⇒ 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 ;
- ⇒ générer un Hash du document numérique ;
- ⇒ générer un certificat (auto-signé, puisque le signé est payant) ;
- ⇒ chiffrer/déchiffrer un document ;

Il faut noter que l'administrateur ne pourra effectuer ces fonctionnalités que s'il est authentifier avec son compte administrateur, question de sécurité.

2. l'Utilisateur de :

- ⇒ générer une paire de clé(publique et privée) ;
- ⇒ signer un document numérique à l'aide de sa clé privée,

- ⇒ vérifier la signature d'un document numérique à l'aide de sa clé publique ;
- ⇒ chiffrer/déchiffrer un document ;
- ⇒ générer le Hash du document numérique.

3. au système externe de :

- ⇒ générer une paire de clé(publique et privée),
- ⇒ signer un document numérique à l'aide de la clé privée,
- ⇒ générer un certificat (auto-signé, puisque le signé est payant),
- ⇒ vérifier la signature d'un document numérique à l'aide de la clé publique,
- ⇒ chiffrer/déchiffrer un document,
- ⇒ générer le Hash du document numérique.

3.3.1.2 Besoins non fonctionnels

A part les besoins fondamentaux, notre système doit répondre aux critères suivants :

- ⇒ la rapidité de traitement : En effet, vu le nombre important des transactions quotidiennes, il est impérativement nécessaire que la durée d'exécution des traitements s'approche le plus possible du temps réel ;
- ⇒ la performance : Un logiciel doit être avant tout performant c'est-à-dire à travers ses fonctionnalités, répond à toutes les exigences des usagers d'une manière optimale ;
- ⇒ La convivialité : Le futur logiciel doit être facile à utiliser. En effet, les interfaces utilisateurs doivent être conviviales c'est-à-dire simples, ergonomiques et adaptées à l'utilisateur.

Elle devra aussi être capable de :

- ⇒ tourner en réseaux pour qu'il soit accessible de tous ;
- ⇒ être compatible avec n'importe quel système d'exploitation, smartphone, tablette et OS.

Il faut aussi souligner que l'application devra être hautement sécurisée car les informations ne devront pas être accessibles de tous, sauf les légitimes.

3.4 Budgétisation

TABLE 3.1 – La budgétisation

Ressources	Noms	Quantité	Prix unitaire	Total
Ressources humaines	Ingénieur en génie logiciel	2	300000 FCFA	900 000 FCFA
	Ingénieur en cryptographie et sécurité informatique	1	500000 FCFA	500 000 FCFA
Ressources matérielles	Serveur	1	1 524 346 FCFA	1 524 346 FCFA
	Ordinateur de contrôle	1	350000 FCFA	350000 FCFA
Total				3 274 346 FCFA

3.5 Conception architecturale

Nous élaborons les spécifications de notre architecture générale de notre système. Nous optons pour une architecture client serveur centralisée. L'environnement client/serveur désigne un mode de communication organisé par l'intermédiaire d'un réseau et d'une interface Web entre plusieurs ordinateurs . « cela signifie que des machines clientes (machines faisant partie du réseau) contactent un serveur, une machine généralement très puissante en terme de capacités d'entrées-sorties , qui leur fournit des services. Lesquels services sont exploités par des programmes ,appelés programmes clients, s'exécutant sur les machines clientes. »

Puisqu'il existe plusieurs environnements client/serveur(Architecture "**Peer to Peer**",Architecture à 2 niveaux,Architecture à 3 niveaux, etc), plus précisément nous optons pour une architecture client/serveur Peer to Peer. Le réseau est dit pair à pair (peer-to-peer en anglais, ou P2P), lorsque chaque terminal connecté au réseau est susceptible de jouer tour à tour le rôle de client et celui de serveur.

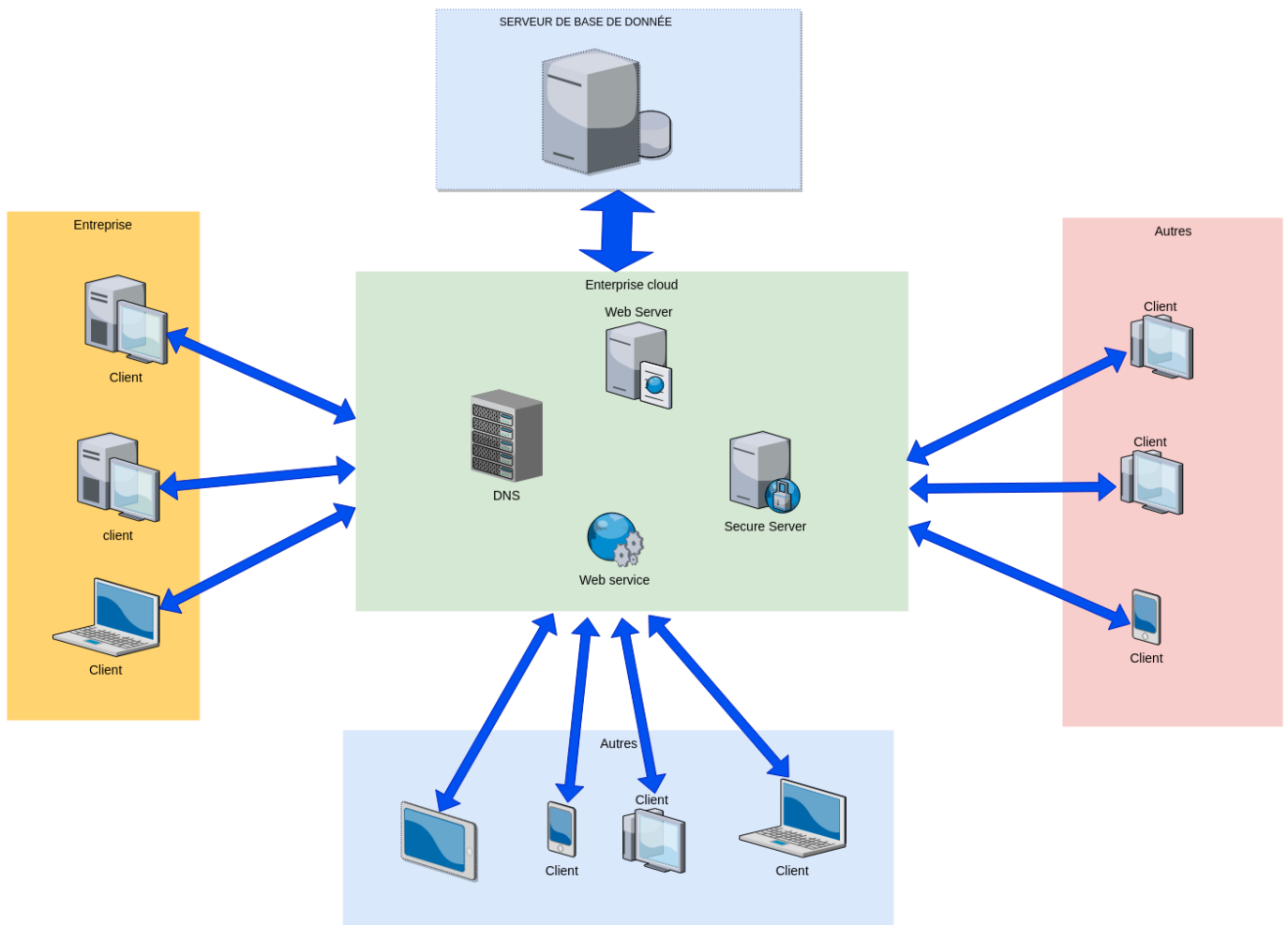


FIGURE 3.2 – Architecture Générale Client Serveur

3.6 Conception détaillée

Pour réaliser la conception détaillée de notre système nous utiliserons le langage de modélisation UML (Unified Modeling Language).

3.6.1 Présentation de langage UML

L'UML (pour Unified Modeling Language, ou "langage de modélisation unifié" en français) est un langage permettant de modéliser nos classes et leurs interactions. Autrement c'est un ensemble de notations graphiques s'appuyant sur des diagrammes et permettant de spécifier, visualiser et de documenter les systèmes logiciels orientés-objet. UML utilise des diagrammes pour modéliser un système. Il ne s'agit pas d'une simple notation graphique car les concepts transmis par un diagramme ont une sémantique[Gui12].

En ce qui concerne la structure du formalisme UML, il peut être vu comme nous montre

la figure suivante :

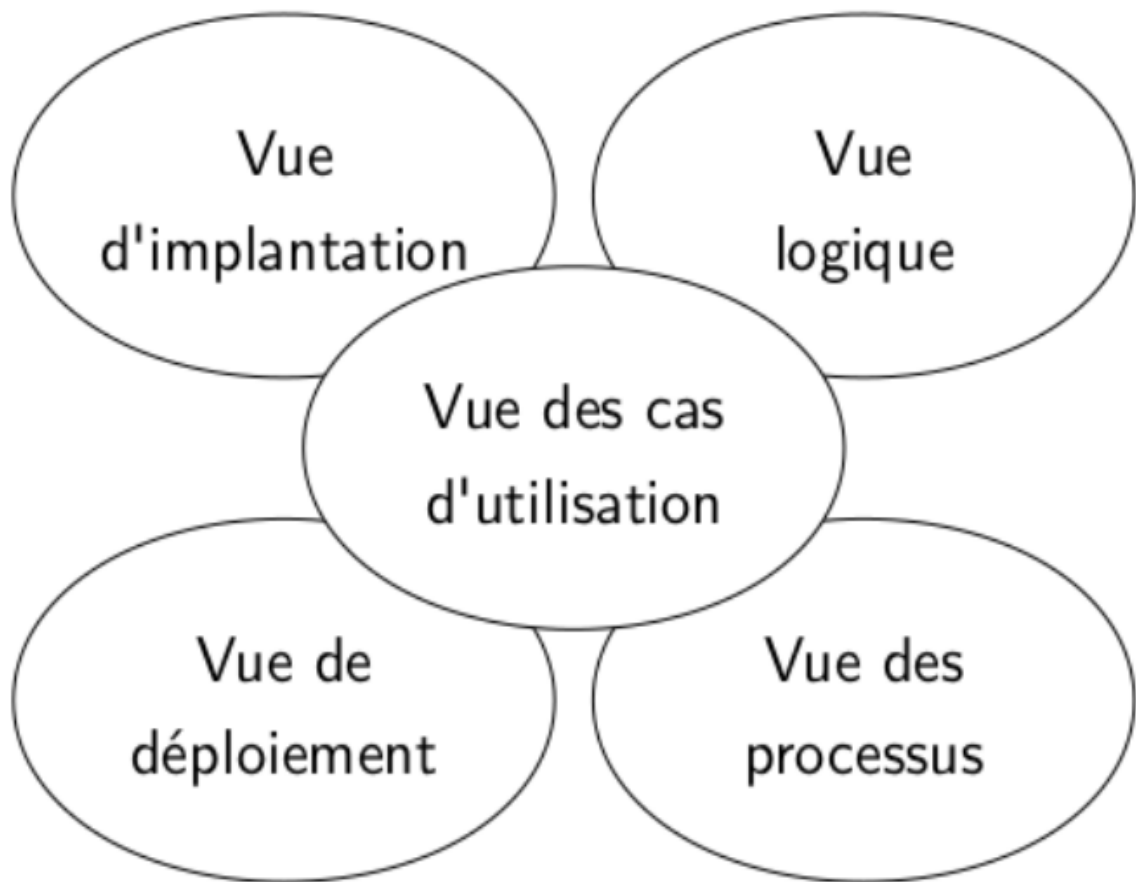


FIGURE 3.3 – différentes vues du formalisme UML

A chaque vue est associée certains diagramme :

- ⇒ **Vue de cas d'utilisation[ROE19]** : S'applique à l'ensemble des cas d'utilisation qui décrivent ensemble le comportement d'un système donné vu par ses acteurs. Cette vue indique les forces internes et externes qui forment l'architecture du système. Elle définit les besoins des clients du système et centre la définition de l'architecture du système sur la réalisation de ces besoins ; elle conduit à la définition d'un modèle d'architecture pertinent et cohérent en se basant sur les scénarios décrits dans les cas d'utilisation.
Elle motive les choix, permet d'identifier les interfaces critiques et force à se concentrer sur les problèmes importants.
- ⇒ **Vue logique** : a pour but d'identifier les éléments du domaine, les relations et interactions entre ces éléments. Cette vue organise les éléments du domaine en « catégories ». Deux diagrammes peuvent être utilisés pour cette vue : diagramme de classes et diagramme des objets.

- ⇒ **Vue des processus** : Démontre la décomposition système en processus et actions, les interactions entre les processus, la synchronisation et la communication des activités parallèles. Elle s'appuie sur plusieurs diagrammes : diagramme de séquence, diagramme d'activité, diagramme de collaboration etc.
- ⇒ **Vue de déploiement** : décrit les ressources matérielles et la répartition des parties du logiciel sur ces éléments. Il contient un diagramme : le diagramme de déploiement.
- ⇒ **Vue d'implémentation** : Décrit l'ensemble des algorithmes utilisés et le code source[ROE19].

3.6.2 Modélisation avec le langage UML

Pour modéliser notre système avec le langage UML, nous allons utiliser les outils suivants :

- ⇒ Astah-pro, version d'évaluation,
- ⇒ Umbrello,
- ⇒ Draw.io,
- ⇒ GIMP 2.10.

3.6.2.1 Diagramme de cas d'utilisation

Le diagramme des cas d'utilisation est une notation très simple et compréhensible par tous et qui permet de structurer les besoins (cahier des charges) et le reste du développement. Un diagramme de cas d'utilisation décrit les acteurs¹, les cas d'utilisation² et le système. Un modèle de cas utilisation peut être formé de plusieurs diagrammes de cas d'utilisation, de descriptions textuelles, de diagrammes de séquences. Un cas d'utilisation CU) décrit une manière d'utiliser le système en une suite d'interactions entre un acteur et le système.

1. Diagramme de cas d'utilisation général :

-
1. Un acteur est un utilisateur, humain ou non, du système qui est doté d'un nom qui correspond à son rôle.
 2. Un cas d'utilisation est une manière spécifique d'utiliser le système.

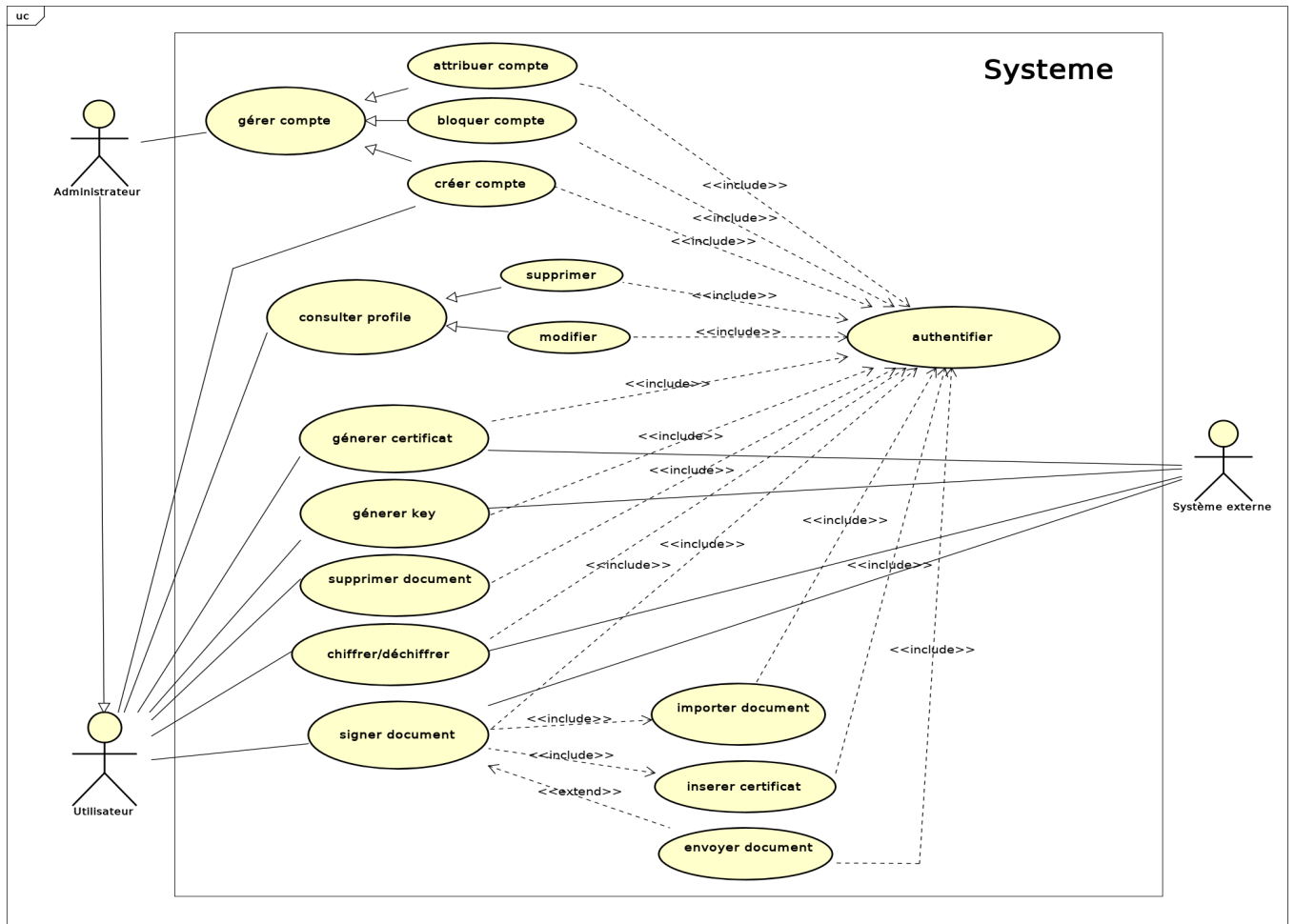


FIGURE 3.4 – Diagramme de cas d'utilisation général

2. Diagramme de cas d'utilisation de l'administrateur :

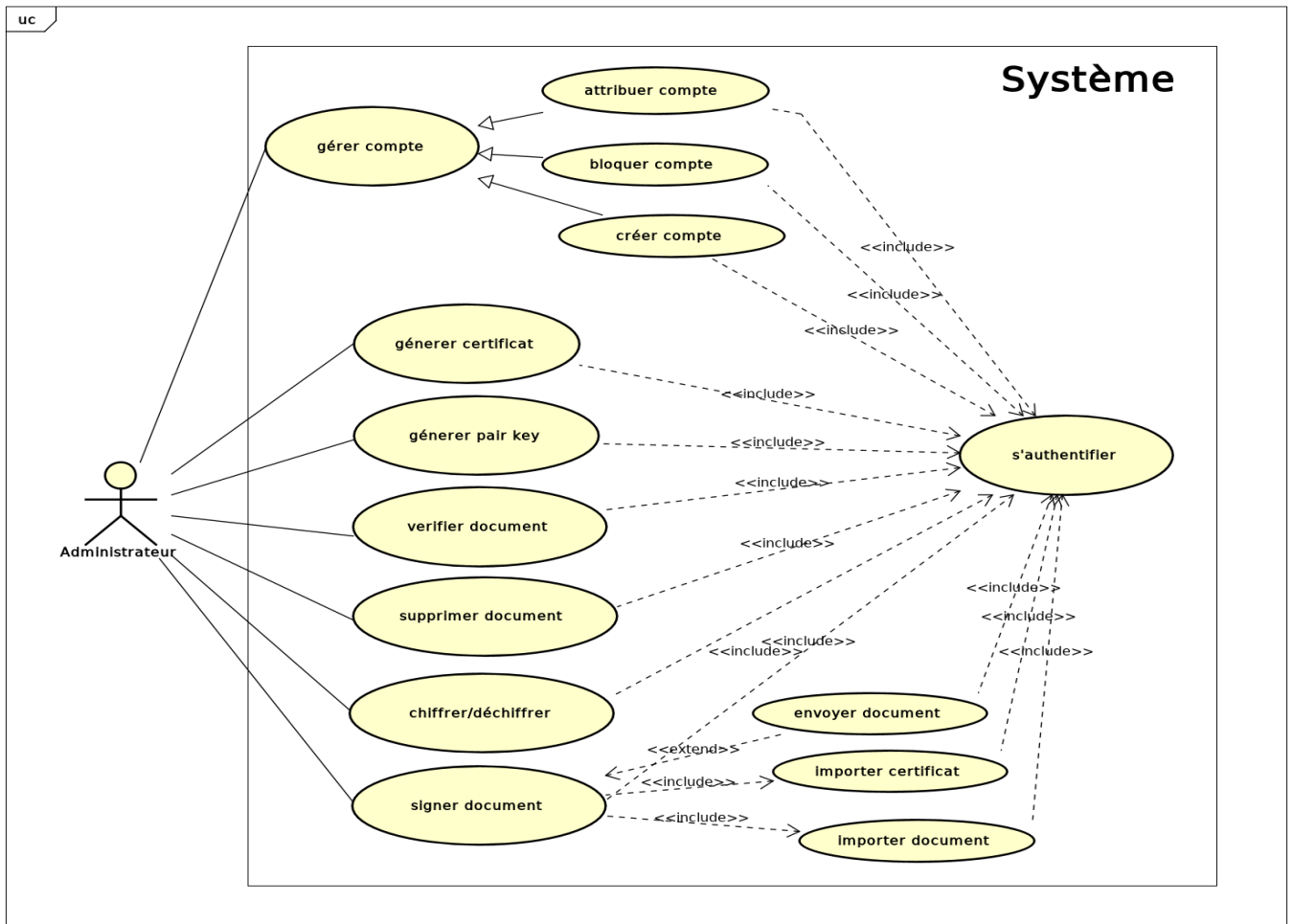


FIGURE 3.5 – Diagramme de cas d'utilisation d'administrateur

3. Diagramme de cas d'utilisation de l'utilisateur :

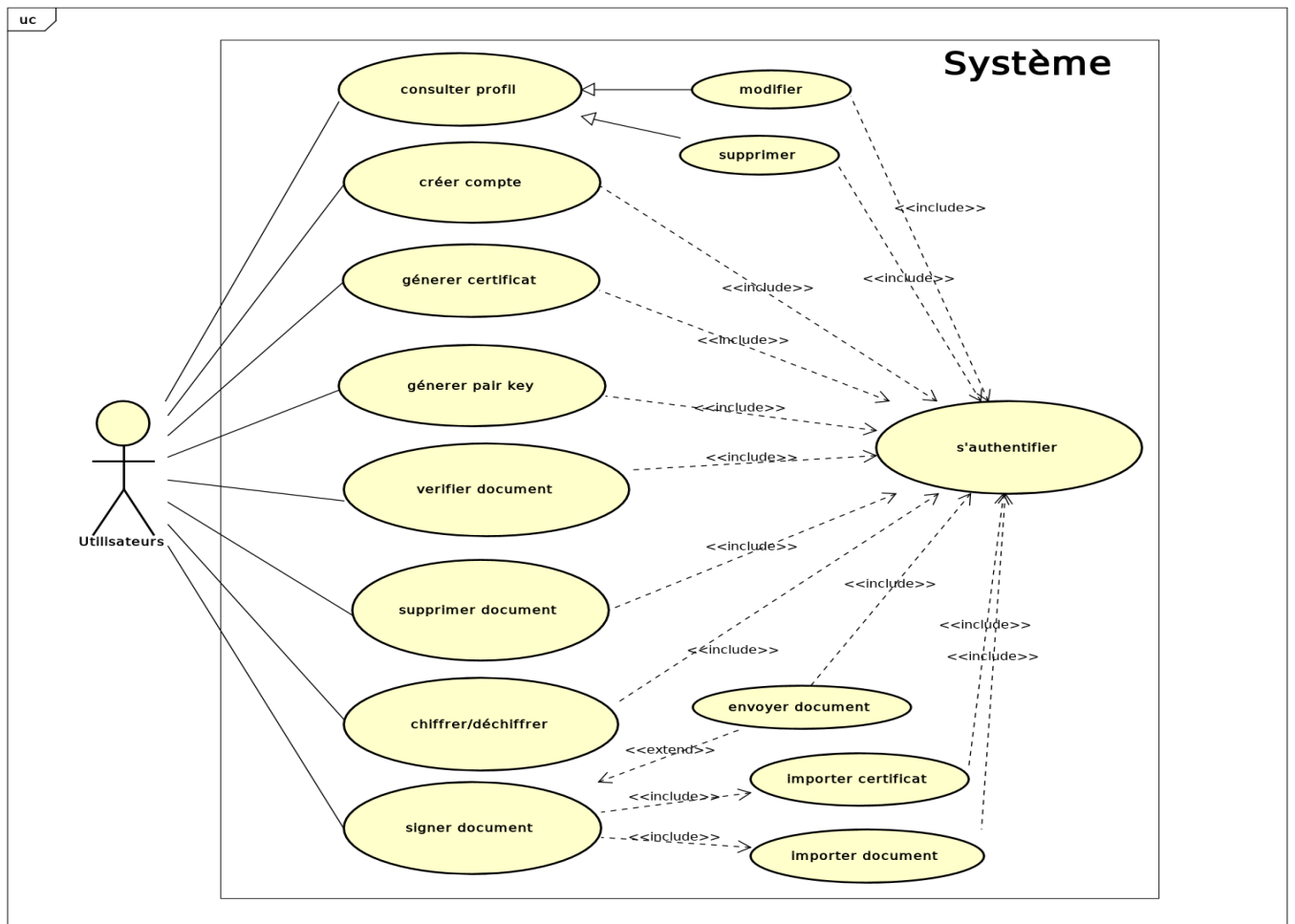


FIGURE 3.6 – Diagramme de cas d'utilisation d'utilisateur

4. Diagramme de cas d'utilisation de système externe :

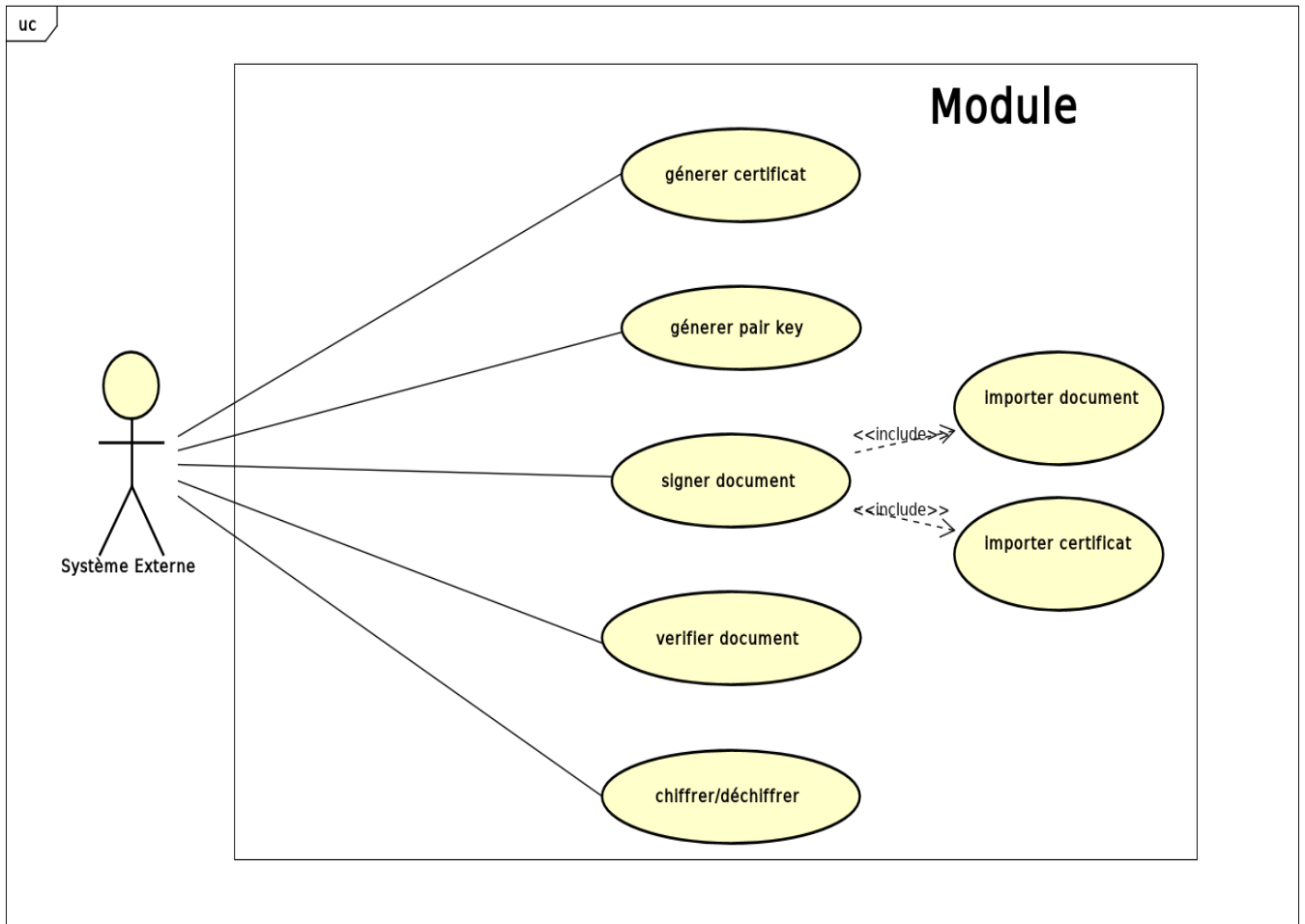


FIGURE 3.7 – Diagramme de cas d'utilisation de système externe

3.6.2.2 Description textuelle

1. Description textuelle de cas d'utilisation : **S'authentifier**

TABLE 3.2 – Description contextuelle de cas d'utilisation s'authentifier

Titre	S'authentifier
Résumé	Ce cas d'utilisation permet d'accéder au tableau de bord de l'utilisateur
Acteur	Administrateur, utilisateur
Pré-condition	l'application doit être lancer(page d'accueil)
Scénario nominal	1) l'utilisateur fait une demande l'accès au système en cliquant sur le bouton connexion, 2) le système lui renvoi le formulaire de connexion 3) l'utilisateur introduit son username et password 4) le système vérifie que sont username et password corrects 5) le système ouvre une session de l'utilisateur
Enchaînement Alternatif	le username et le password sont corrects., l'enchaînement commence au point 3 du scénario nominal. le message affiche un message d'erreur, aller au point 2.
Enchaînement d'Erreur	E1 : Si l'étape 2 de scénario nominal n'est pas vérifié un message d'erreur sera affiché.
Post condition	ouverture d'une session, accès au compte

2. Description textuelle de cas d'utilisation : **Générer certificat**

TABLE 3.3 – Description contextuelle de cas d'utilisation générer certificat

Titre	Générer Certificat
Résumé	Ce cas d'utilisation permet de générer des certificats aux utilisateurs
Acteur	Autorité de certification Administrateur, Système externe
Pré-condition	l'application doit être lancée (page d'accueil)
Scénario nominal	1) un AC peut générer un certificat en cliquant sur le bouton <code>genererCertifcat</code> , 2) le système lui renvoie le formulaire à remplir 3) on introduit la clé publique de l'utilisateur 4) on renseigne le nom et le prénom de l'utilisateur 5) la date de validité (début et fin) 6) numéro de version 7) numéro de série.
Enchaînement Alternatif	la clé publique n'est pas correcte. l'enchaînement commence au point 2 du scénario nominal.
Enchaînement d'Erreur	E1 : Si l'étape 2 de scénario nominal n'est pas vérifiée un message d'erreur sera affiché.
Post condition	création de certificat avec succès.

3. Description textuelle de cas d'utilisation : **Générer Paire de Clé**

TABLE 3.4 – Description contextuelle de cas d'utilisation générer paire de clé

Titre	Générer Paire de clé
Résumé	Ce cas d'utilisation permet de générer des paires de clé aux utilisateurs
Acteur	Autorité de certification, Administrateur, Système externe, Utilisateur
Pré-condition	l'application doit être lancer (page d'accueil) et l'utilisateur est authentifié
Scénario nominal	1) un utilisateur peut générer une paire en cliquant sur le bouton genererPaireKey, 2) le système lui renvoi le formulaire à remplir 3) il introduit le cryptosystème à utiliser (ex. RSA) 4) ensuite il introduit la taille de clé (ex.4096) 5) il valide en cliquant sur le bouton générer
Enchaînement Alternatif	l'algorithme ou la taille de clé est incorrect. l'enchaînement commence au point 2 du scénario nominal.
Enchaînement d'Erreur	E1 : Si l'étape 3 de scenario nominal est incorrect, un message d'erreur sera affiché. Si l'étape 4 de scenario nominal est incorrect, un message d'erreur sera affiché.
Post condition	génération de paire de clé avec succès.

4. Description textuelle de cas d'utilisation : **Signer document**

TABLE 3.5 – Description contextuelle de cas d'utilisation Signer document

Titre	Signer document
Résumé	Ce cas d'utilisation permet aux utilisateurs de signer leurs documents
Acteur	Autorité de certification, Administrateur, Système externe, Utilisateur
Pré-condition	l'application doit être lancée (page d'accueil) et l'utilisateur est authentifié
Scénario nominal	1) un utilisateur peut signer son document en cliquant simplement sur le bouton Signer document, 2) le système lui renvoie le formulaire à remplir 3) il téléverse son fichier(txt, docs, pdf,etc) 4) ensuite il téléverse sa clé publique 5) il sélectionne la fonction de hachage (ex. SHA1, SHA256 ...) 6) il valide en cliquant sur le bouton Signer document
Enchaînement Alternatif	l'algorithme est incorrect. l'enchaînement commence au point 2 du scénario nominal.
Enchaînement d'Erreur	E1 : Si l'étape 3 de scénario nominal est incorrect, un message d'erreur sera affiché. Si l'étape 4 de scénario nominal est incorrect, un message d'erreur sera affiché.
Post condition	La signature est effectuée avec succès.

5. Description textuelle de cas d'utilisation : **Envoyer Document**

TABLE 3.6 – Description contextuelle de cas d'utilisation Envoyer Document

Titre	Envoyer Document
Résumé	Ce cas d'utilisation permet aux utilisateurs d'envoyer leurs document signer
Acteur	Administrateur, Utilisateur
Pré-condition	l'application doit être lancer (page d'accueil) et l'utilisateur est authentifié
Scénario nominal	1) un utilisateur peut envoyer un document en cliquant sur l'onglet message Envoyer document 2) le système lui renvoi le formulaire à remplir 3) il téléverse son document 4) il téléverse son certificat 5) ensuite il choisit un destinataire 6) enfin il valide en cliquant sur le bouton Envoyer .
Enchaînement Alternatif	l'adresse E-mail est incorrect. l'enchaînement commence au point 2 du scénario nominal.
Enchaînement d'Erreur	E1 : Si l'étape 5 de scénario nominal est incorrect, un message d'erreur sera affiché.
Post condition	L'envoi est effectué avec succès.

6. Description textuelle de cas d'utilisation : **Chiffrer/Déchiffrer**

TABLE 3.7 – Description contextuelle de cas d'utilisation Chiffrer/Déchiffrer

Titre	Chiffrer/Déchiffrer
Résumé	Ce cas d'utilisation permet aux utilisateurs chiffrer/déchiffrer leurs documents
Acteur	Administrateur, Utilisateur et système externe
Pré-condition	l'application doit être lancée (page d'accueil) et l'utilisateur est authentifié
Scénario nominal	1) un utilisateur peut chiffrer/déchiffrer un document en cliquant sur le bouton chiffrer/déchiffrer 2) le système lui renvoie le formulaire à remplir 3) il téléverse son document 4) il téléverse sa clé publique 5) il choisit l'algorithme et la taille 6) enfin il valide en cliquant sur le bouton chiffrer/déchiffrer .
Enchaînement Alternatif	l'algorithme est incorrect. l'enchaînement commence au point 2 du scénario nominal.
Enchaînement d'Erreur	E1 : Si l'étape 5 de scénario nominal est incorrect, un message d'erreur sera affiché.
Post condition	Le chiffrement/déchiffrement est effectué avec succès.

3.6.2.3 Diagramme de classes

Diagramme de classe est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation. Il permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation. Le diagramme de classe modélise les concepts du domaine d'application ainsi que les concepts internes créés de toutes pièces dans le cadre de l'implémentation d'une application. Les principaux éléments de cette vue statique sont les classes et leurs relations : associations, généralisations et plusieurs types de dépendances, telles que la relation et l'utilisation.

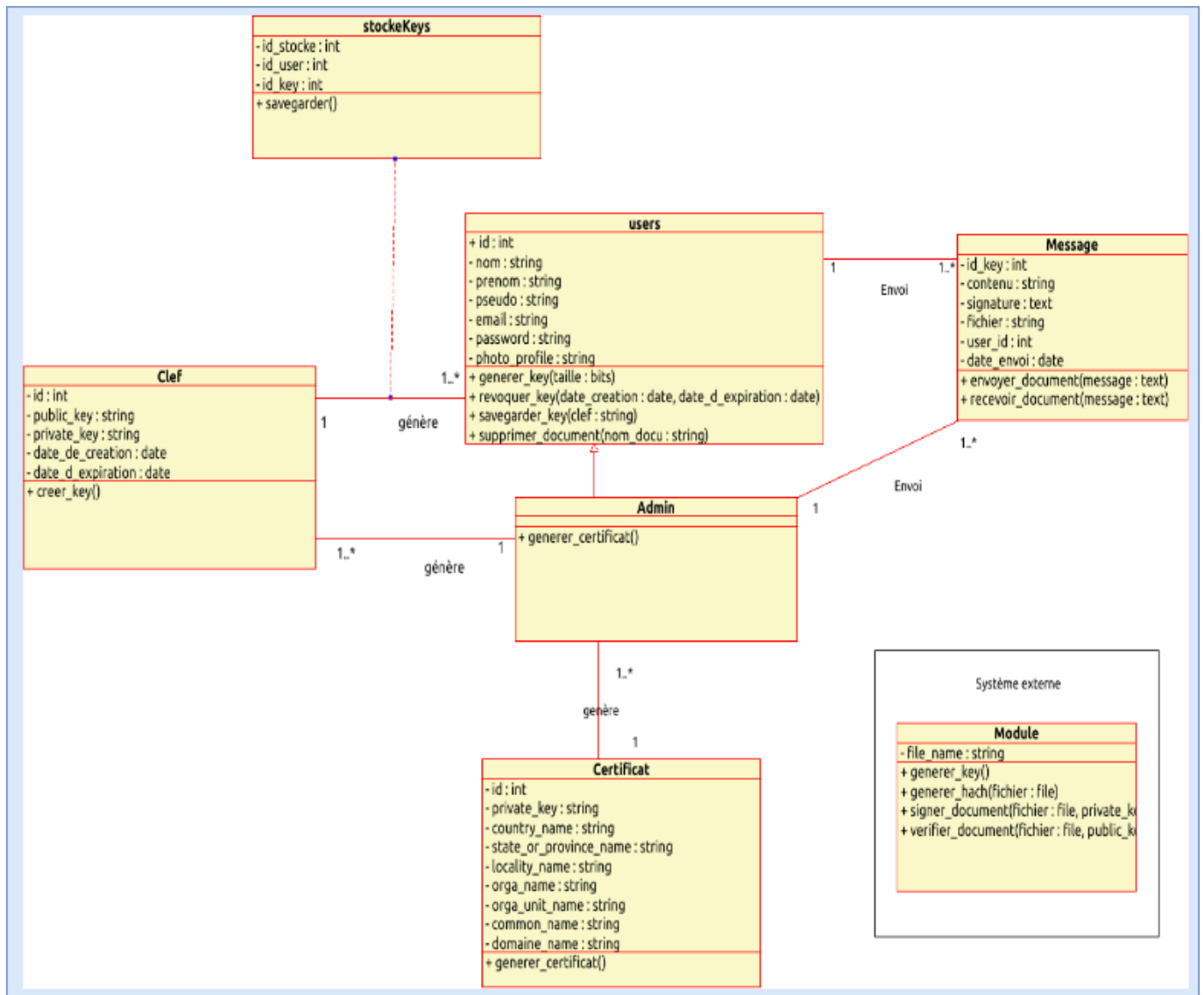


FIGURE 3.8 – Diagramme de classe

3.6.2.4 Diagramme de packages

Lorsque nous sommes en présence d'un système de grande taille, il peut être intéressant de le décomposer en plusieurs parties (appelées paquetage). Un paquetage est donc un regroupement de différents éléments d'un système (regroupement de classes, diagrammes, fonctions, interfaces...). Cela permet de clarifier le modèle en l'organisant. Il est représenté par un dossier avec son nom à l'intérieur.

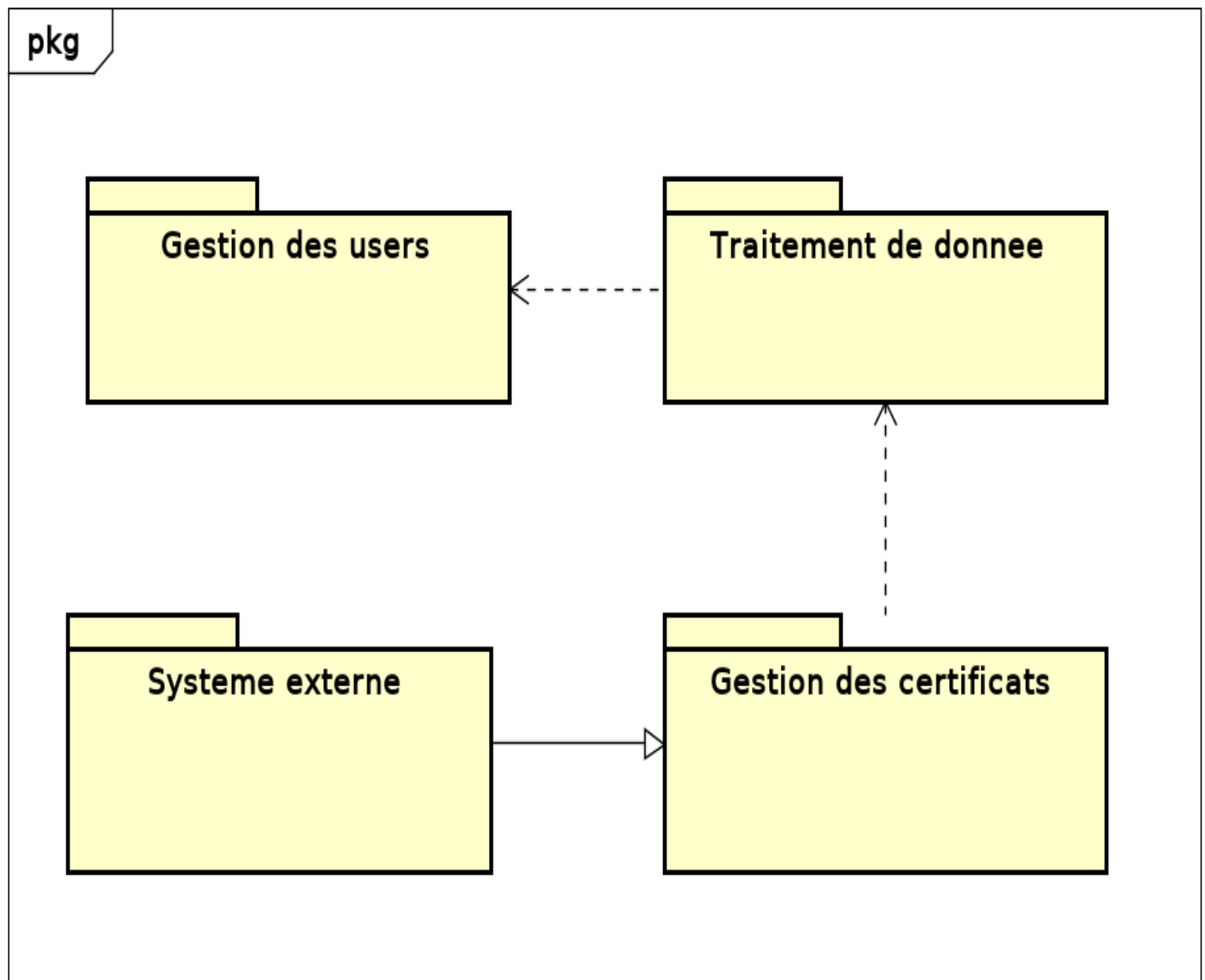


FIGURE 3.9 – Diagramme de packages

3.6.2.5 Diagramme de séquences

Pour décrire un scénario, UML propose un diagramme de séquences qui permet de décrire une séquence des messages échangés entre différents objets. Les diagrammes de séquences permettent de décrire comment les éléments du système interagissent entre eux et avec les acteurs. Les objets d'un système interagissent en s'échangeant des messages. Les acteurs interagissent avec le système au moyen d'interfaces homme-machine[MER18] .

1. Diagramme de séquence de cas d'utilisation s'authentifier :

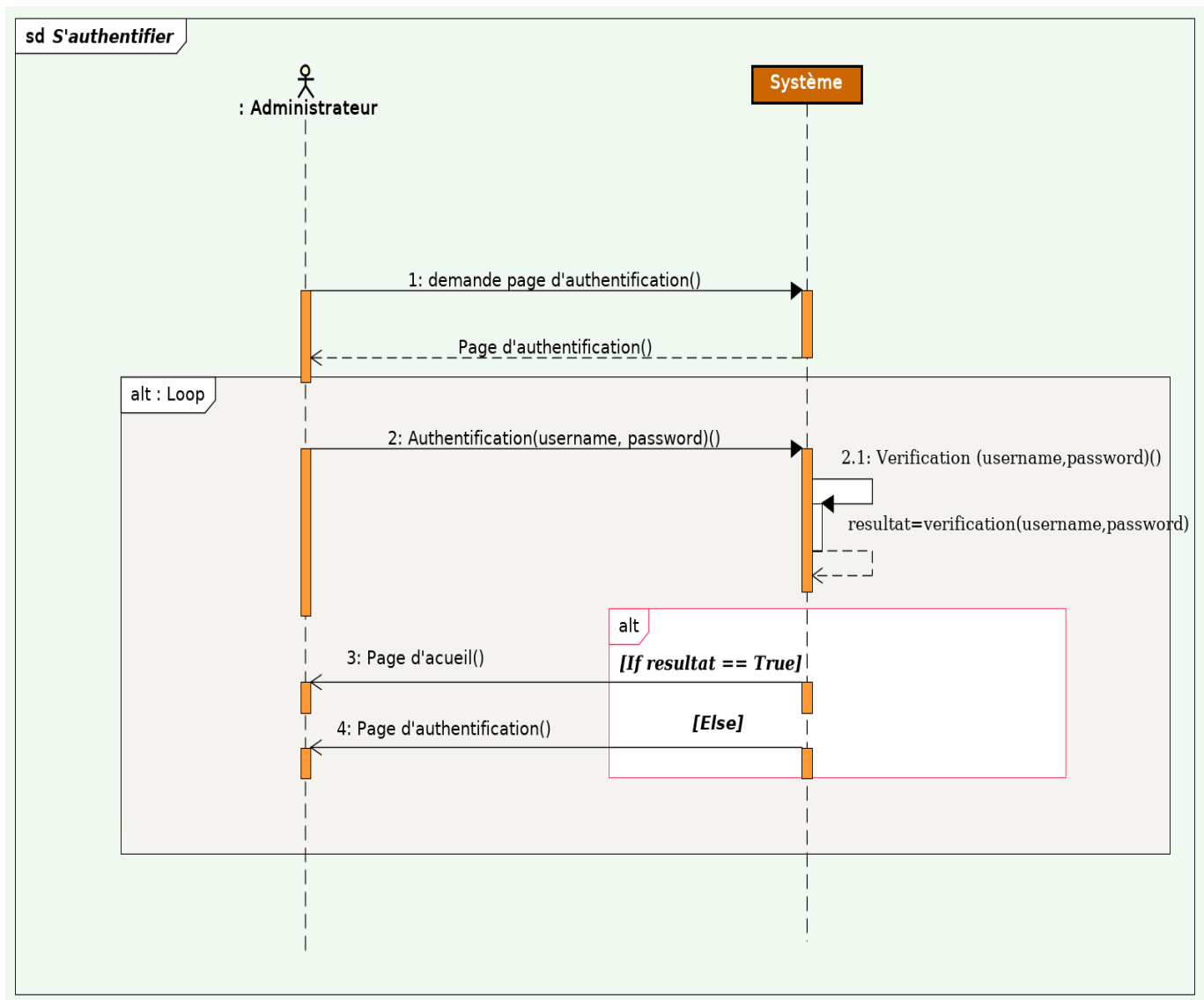


FIGURE 3.10 – Diagramme de séquence de cas d'utilisation s'authentifier

2. Diagramme de séquence de cas d'utilisation générer key :

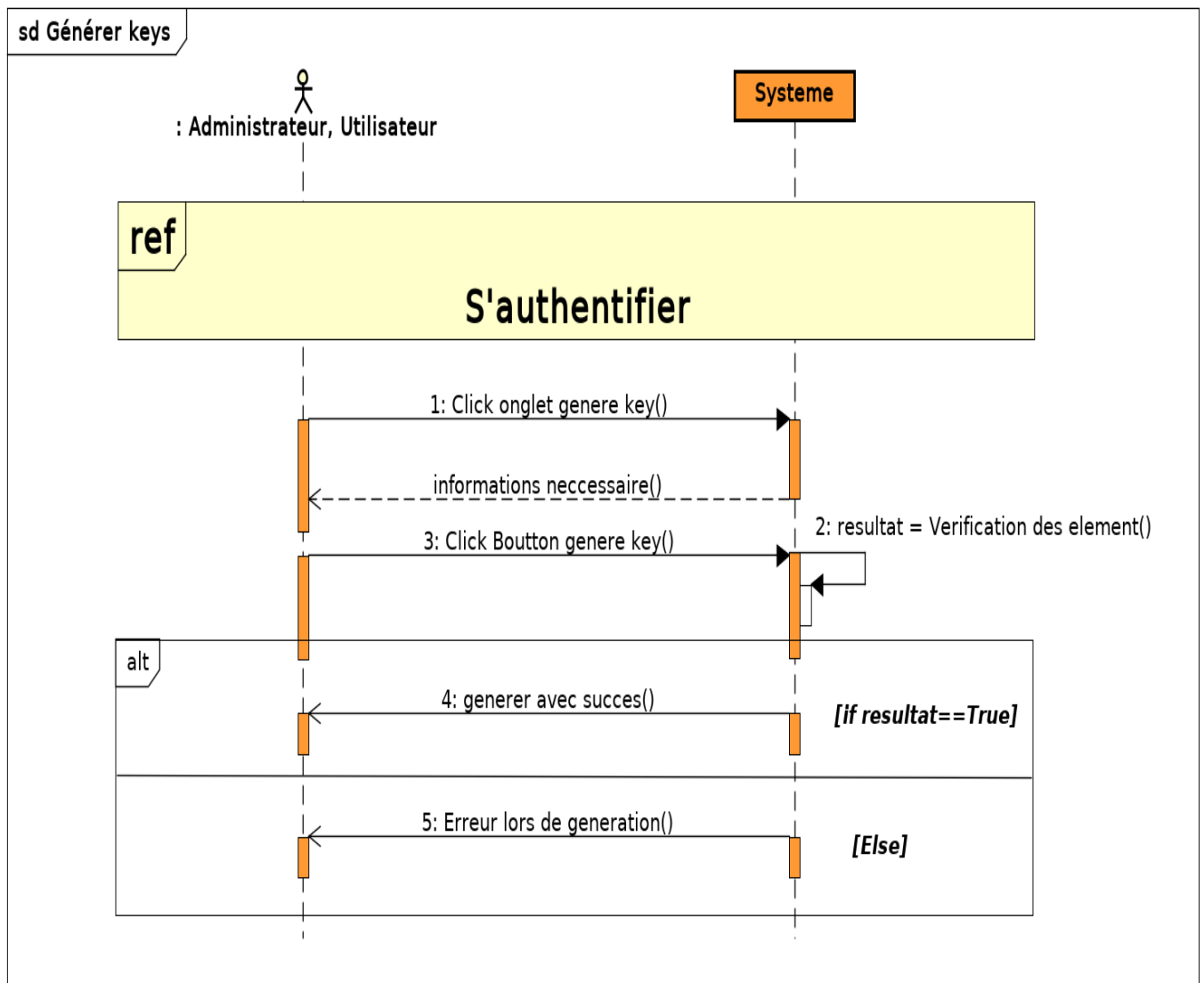


FIGURE 3.11 – Diagramme de séquence de cas d'utilisation générer key

3. Diagramme de séquence de cas d'utilisation certificat

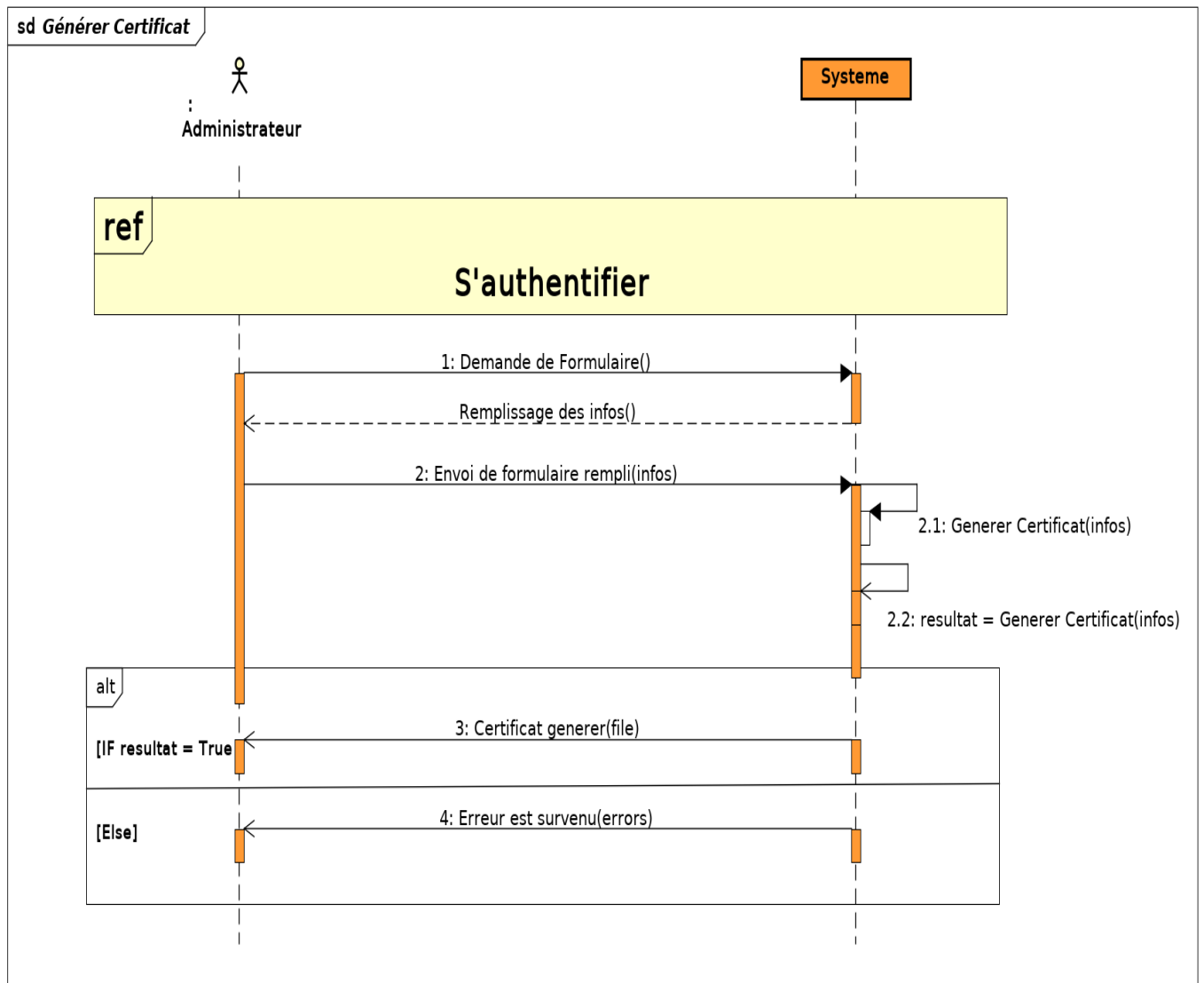


FIGURE 3.12 – Diagramme de séquence de cas d'utilisation générer certificat

4. Diagramme de séquence de cas d'utilisation signer document :

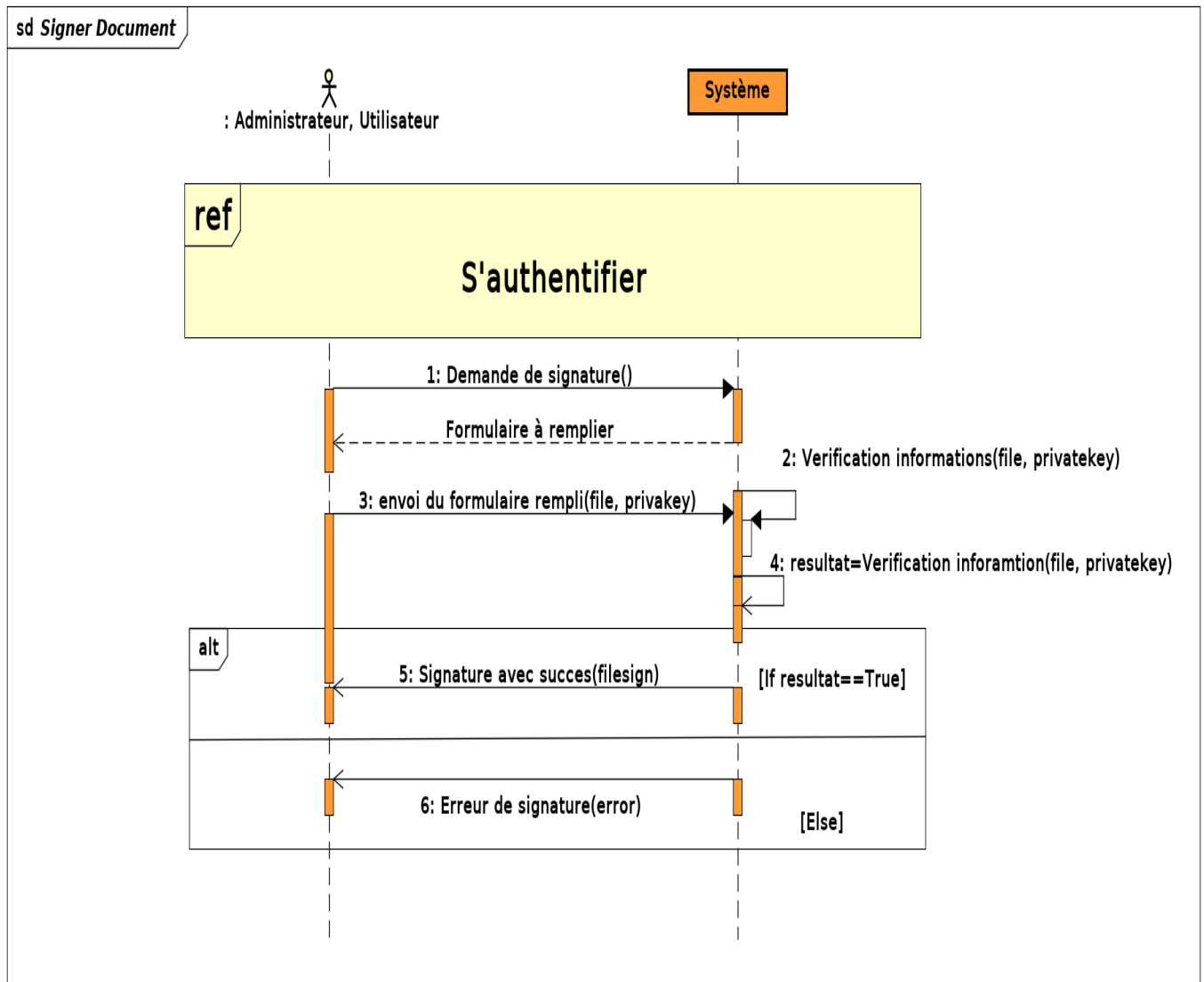


FIGURE 3.13 – Diagramme de séquence de cas d'utilisation signer document

5. Diagramme de séquence de cas d'utilisation chiffrer/déchiffrer :

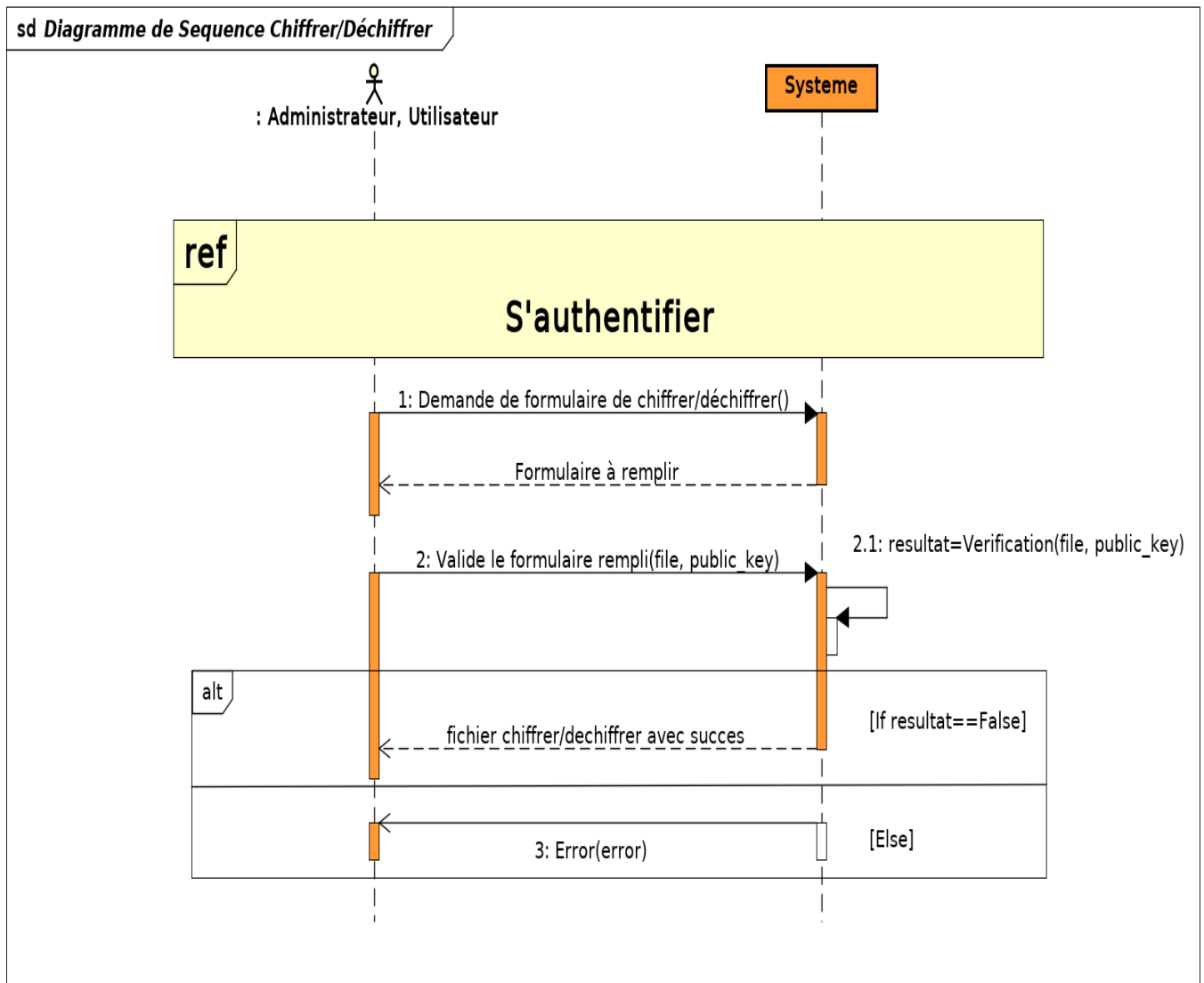


FIGURE 3.14 – Diagramme de séquence de cas d'utilisation signer document

6. Diagramme de séquence de cas d'utilisation envoyer document :

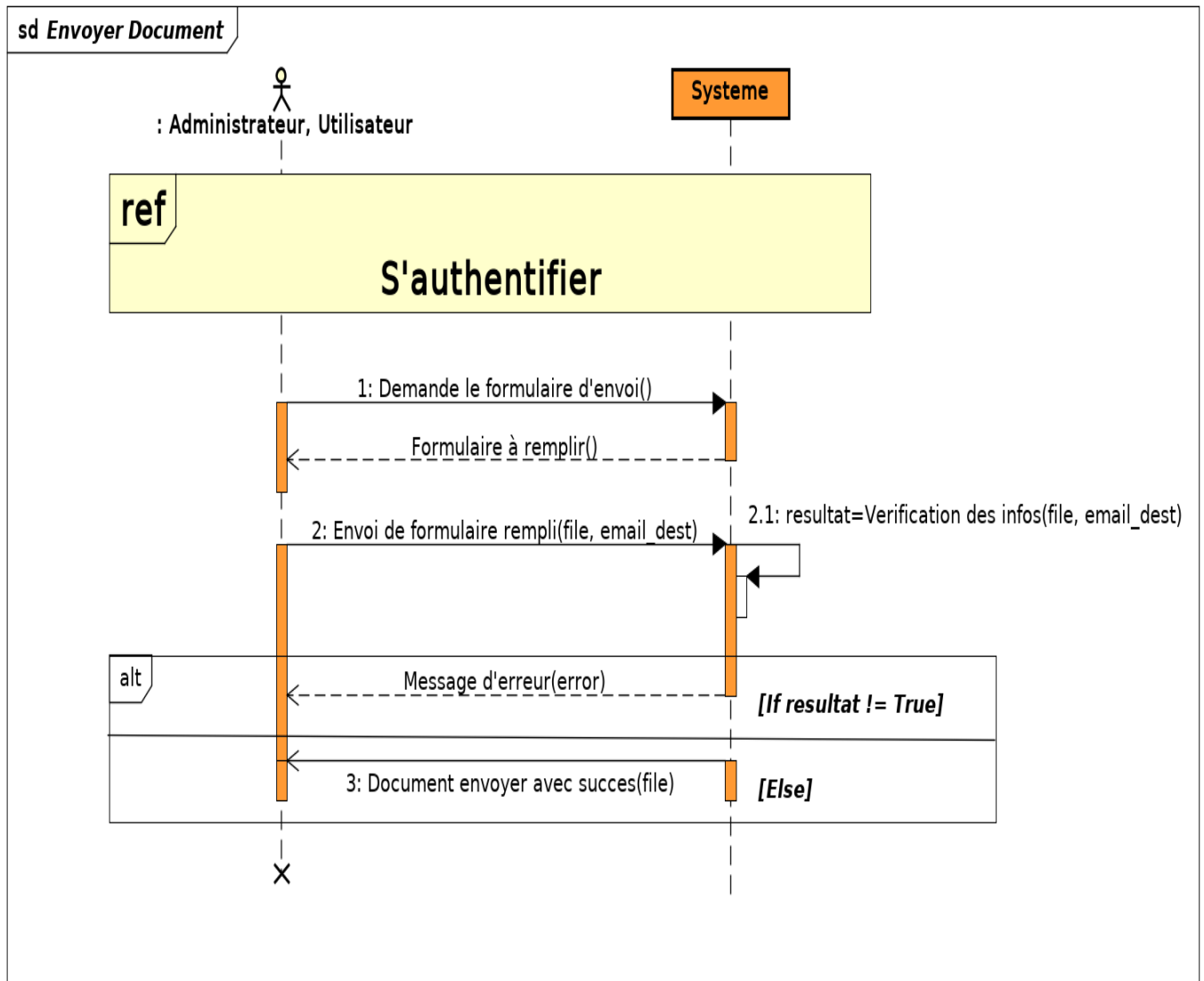


FIGURE 3.15 – Diagramme de séquence de cas d'utilisation envoyer document

3.6.2.6 Diagramme de déploiement

Dans le contexte du langage de modélisation unifié (UML), un diagramme de déploiement fait partie de la catégorie des diagrammes structurels, car il décrit un aspect du système même. Dans le cas présent, le diagramme de déploiement décrit le déploiement physique des informations générées par le logiciel sur des composants matériels. On appelle artefact l'information qui est générée par le logiciel[Lucin].

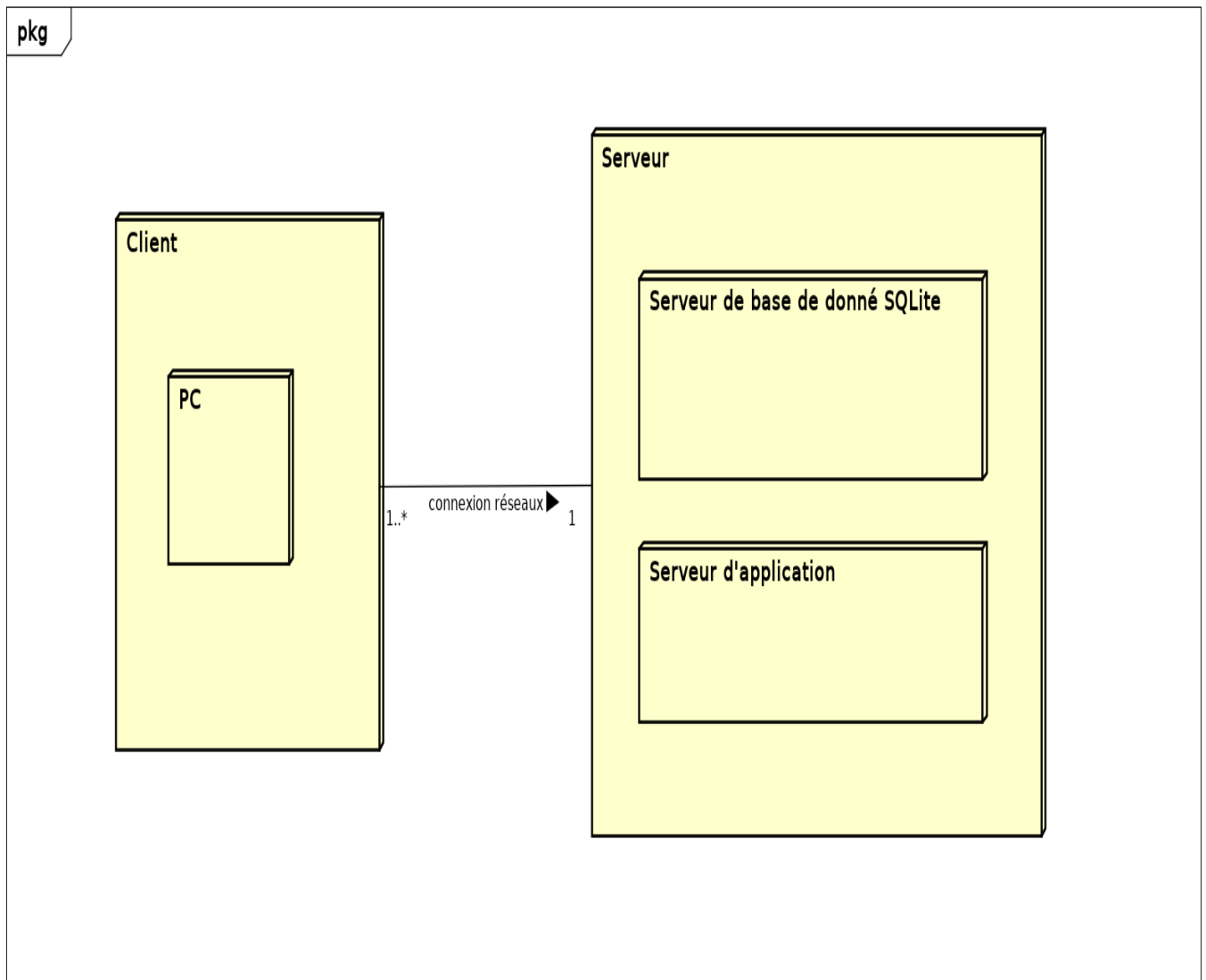


FIGURE 3.16 – Diagramme de séquence de cas d'utilisation envoyer document

Conclusion

En conclusion, il était question dans ce chapitre de présenter l'analyse et la conception de notre plate-forme de signature numérique. Nous avons recueillis les besoins nécessaires, ensuite nous avons effectué la modélisation avec le langage de modélisation UML.

Chapitre 4

Implémentation et Tests

4.1 Introduction

La conception et l'analyse étant terminée, dans ce chapitre nous implémenterons les fonctionnalités du module et nous terminerons avec les tests.

4.2 Environnement de développement

4.2.1 Environnement matériel

Nous avons utilisé côté matériel les éléments suivants :

- ⇒ Un PC marque ACER Aspire 5 A515-51-527A,
- ⇒ Un système d'exploitation Linux (Ubuntu 18.10LTS),
- ⇒ Type du système d'exploitation : 64 bits,
- ⇒ Processeur Intel Core i5-7200U CPU @ $2.50GHz \times 4$,
- ⇒ Carte Graphique : Intel® HD Graphics 620 (Kaby Lake GT2),
- ⇒ Disque dure 1000 Go.

4.2.2 Environnement Logiciel

Concernant les logiciels, nous avons utilisé :

1. **LAMP** : est un acronyme désignant un ensemble de logiciels libres permettant de construire des serveurs de sites web. L'acronyme original se réfère aux logiciels suivants :
 - ⇒ « Linux », le système d'exploitation (GNU/Linux) ;
 - ⇒ « Apache », le serveur Web ;

- ⇒ « MySQL ou MariaDB », le serveur de base de données ;
- ⇒ À l'origine « PHP », « Perl » ou « Python », les langages de script.

Les rôles de ces quatre composants sont les suivants :

- ⇒ Linux assure l'attribution des ressources aux autres composants (Rôle d'un Système d'exploitation ou OS pour Operating System) ;
 - ⇒ Apache est le serveur web « frontal » : il est « devant » tous les autres et répond directement aux requêtes du client web (navigateur) ;
 - ⇒ MySQL est un système de gestion de bases de données (SGBD). Il permet de stocker et d'organiser des données ;
 - ⇒ le langage de script PHP permet la génération de pages web dynamiques et la communication avec le serveur MySQL.
2. **Sublime Text 3** : Sublime Text est un éditeur de texte générique codé en C++ et Python sous licence End User License Agreement(propriétaire) mais gratuit d'utilisation, disponible sur Windows, Mac et Linux. Le logiciel a été conçu tout d'abord comme une extension pour Vim, riche en fonctionnalités. Depuis la version 2.0, sortie le 26 juin 2012, l'éditeur prend en charge 44 langages de programmation majeurs, tandis que des plugins sont souvent disponibles pour les langages plus rares. Il a été développé par Jon Skinner ancien développeur de Google. Actuellement on est à la version 3.
3. **Le framework python Flask** : est un microframework pour Python basé sur Werkzeug¹, Jinja² et de bonnes intentions. Il s'agit d'un ensemble de modules qui permettent de développer plus rapidement en fournissant des fonctionnalités courantes. Lorsque vous concevez une application web[San19], vous avez toujours besoin de gérer les requêtes HTTP, d'un serveur web, d'afficher des pages web dynamiques, de gérer les cookies... Pourquoi coder ces fonctionnalités à chaque fois ? **Un framework web vous fournit ces fonctionnalités pour commencer un projet sur des bases solides.**

4.3 Développement du module

4.3.1 La structure générale de notre module

Le module est nommé **oudjirasign**, il composé des dossiers et des fichiers selon la hiérarchie conventionnelle de création d'un module python suivant :

-
1. WSGI est l'interface de passerelle Web Server. Il s'agit d'une spécification qui décrit comment un serveur Web communique avec des applications Web et comment les applications Web peuvent être chaînées pour traiter une demande.
 2. Jinja2 est un moteur de template complet pour Python. Il offre une prise en charge complète de l'unicode, un environnement d'exécution intégré en option, largement utilisé et sous licence BSD.

```
oudjirasign/  
├── .gitignore  
├── LICENSE  
├── MANIFEST.in  
├── oudjirasign  
└── ├── __init__.py  
    ├── README.md  
    ├── setup.py  
    └── venv/
```

Cette arborescence est affichée par ordre alphabétique selon commande qu'on a utilisé pour cette affichage et bah on va suivre ce même ordre pour expliquer chaque dossier et fichiers qui s'y trouve dans notre module oudjirasign.

- ⇒ **oudjirasign/** : est le dossier qui contient notre module, celui le dossier racine,
- ⇒ **.gitignore** : est fichier caché qui contient quelques configurations git³, il nous permet d'exclure certains dossier ou fichier qu'on ne veut pas uploader vers github⁴,
- ⇒ **LICENSE** : est une permission officielle ou un permis de faire, d'utiliser ou de posséder quelque chose. Il existe plusieurs license (Libre, propriétaire, etc).
- ⇒ **MANIFEST.in** est un fichier peut être ajouté dans un projet pour définir la liste des fichiers à inclure dans la distribution générée par la commande **sdist** que nous verrons un peu devant,
- ⇒ **oudjirasign** : est un sous dossier dont le nom est le nom de notre module ou en l'occurrence le package puisqu'il contiendra un fichier magique avec beaucoup de underscore,
- ⇒ **__init__.py** : est un fichier python un peu magique puisque lorsque qu'on exécute le projet oudjirasign c'est le premier fichier qui va être exécuter automatiquement avant tous les autres fichiers, s'il contient du code ou pas. C'est dans cette fichier qu'on a mis notre code source du module.
- ⇒ **README.md** : est un fichier qui contient la description ainsi le manuel de notre projet,
- ⇒ **setup.p** : est un fichier python le plus important d'un module python, il contient plusieurs spécifications tels que : la version, le nom l'auteur, une courte description, les dépendances etc.

3. Git est un système de contrôle de version distribué permettant de suivre les modifications du code source au cours du développement du logiciel

4. GitHub est une société américaine qui fournit un hébergement pour le contrôle de version de développement de logiciel à l'aide de Git. C'est une filiale de Microsoft, qui a acquis la société en 2018 pour 7,5 milliards de dollars

- ⇒ **venv** : ce dossier est l'environnement virtuelle. Un environnement virtuel est une arborescence de dossiers qui contiennent les fichiers exécutables Python et autres fichiers qui indiquent que c'est un environnement virtuel. Ce n'est pas obligatoire que ce dossier soit dans le même package, l'essentiel c'est de l'activer. Une fois activé l'installation de toutes les dépendances sont installées dans cet environnement virtuel et non dans la machine hôte. Pour plus d'information consulter le fichier README.md. Vous pouvez donner le nom que vous voulez, ce n'est pas obligatoire non plus que le nom soit **venv**.

4.3.2 Les modules importés

Notre module de signature est un programme Python qui utilise d'autres modules tel que **pycryptodome** pour ne pas recréer la roue, et dans ce module on a importé les fonctions suivantes :

- ⇒ **Crypto.PublicKey import RSA** : Pour générer la paire de clef RSA,
- ⇒ **from Crypto import Random** : Pour générer un grand nombre aléatoire,
- ⇒ **from Crypto.Cipher import PKCS1_OAEP**⁵ : Protocole de cryptage RSA, dans ce protocole on a la fonction cryptage et décryptage RSA.
- ⇒ **from Crypto.Hash import SHA256** : Cette fonction nous permet de générer le hachage d'un texte ou tout autre document électronique,
- ⇒ **from Crypto.Signature import PKCS1_v1_5** : Un schéma de signature numérique ancien mais toujours solide basé sur RSA.

4.4 Développement de l'interface utilisateur

4.4.1 La structure générale du projet

Pour réaliser ce projet on a utilisé le Framework python Flask comme expliqué dans la section 4.2.2. Cette Framework est structuré comme suit[FLAin] :

```
signs_docs/  
├── main.py  
├── README.md  
├── signs_docs/  
├── __init__.py  
├── models.py  
└── routes.py
```

5. Optimal Asymmetric Encryption Padding

```
|— static/  
|— templates/
```

où,

- ⇒ **signs_docs/** : est le dossier la racine du projet,
- ⇒ **main.py** : est un fichier python contenant le fonction principale du projet, c'est grâce à ce fichier qu'on peut exécuter le projet. Exemple **python main.py** et ceci lancera un mini-serveur web en locale avec le port par défaut 5000 (**http ://127.0.0.1 :5000**),
- ⇒ **README.md** : c'est déjà expliqué ici 4.3.1,
- ⇒ **signs_docs/** : un package Python contenant votre code d'application et vos fichiers (Python, HTML, CSS, JS, etc).
- ⇒ **__init__.py** : déjà expliqué dans la section 4.3.1
- ⇒ **models.py** : comme son nom l'indique ce fichier contient le model de notre application. En fait il s'agit tout simplement les tables de notre base de donnée.
- ⇒ **routes.py** : ce fichier contient les routes de notre application. C'est à dire il décrit les liens entre les différentes pages.
- ⇒ **static/** : est un dossier qui contient de fichier statique tels que des fichiers CSS, JS, etc.
- ⇒ **templates/** : est dossier qui contient uniquement des fichiers HTML, c'est à dire le corps de notre projet.

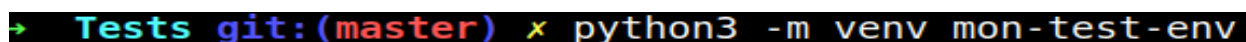
NB : Tous les dossiers et le fichiers peuvent être renommés comme bon vous semble sauf les deux dossiers (**static/** et **templates/**) qui sont par convention inchangeable et **__init__.py** s'il s'agit d'un package sinon on peut le renommé comme on veut.

4.5 Tests

4.5.1 Test du module en console

Pour installer et tester le module nous devons assuré que python et l'environnement virtuelle sont installés dans notre machine et que l'environnement virtuelle activé4.3.1.

La création de l'environnement virtuelle se fait par la commande suivante comme le montre la figure.



```
➔ Tests git:(master) ✕ python3 -m venv mon-test-env
```

FIGURE 4.1 – La création de l'environnement virtuelle

Cette commande permet de créer un dossier appelé mon-test-env, en d'autre terme c'est le nom de l'environnement virtuelle. Pour l'activer c'est très simple il suffit taper la commande suivante :

```
→ Tests git:(master) x source mon-test-env/bin/activate
(mon-test-env) → Tests git:(master) x
```

FIGURE 4.2 – L'activation de l'environnement virtuelle

Nous remarquons qu'il y'a eu un changement au niveau de la deuxième ligne, ce changement indique l'environnement virtuelle est activé.

Nous passons maintenant à l'installation du module **oudjirasign**, pour ce faire on tape la commande suivante comme le montre la figure suivante :

```
(mon-test-env) → Tests git:(master) x pip install oudjirasign
Collecting oudjirasign
  Using cached https://files.pythonhosted.org/packages/00/41/fecfe91543c21c0687c95
606c30976f4dedcea160895b1b1765359fa38e5/oudjirasign-0.0.0.1-py3-none-any.whl
Collecting pycrypto (from oudjirasign)
Installing collected packages: pycrypto, oudjirasign
Successfully installed oudjirasign-0.0.0.1 pycrypto-2.6.1
(mon-test-env) → Tests git:(master) x
```

FIGURE 4.3 – L'installation du module oudjirasign

Nous pouvons voir que le module et ses dépendances sont installés avec succès. Pour tester les fonctionnalités veuillez vous rendre dans le fichier README, nous avons mis un lien vers la documentation dans laquelle où nous avons testé toutes les fonctionnalités.

Exemple 4.5.1.1. Dans cet exemple nous testons la fonctionnalité générer paire clef RSA. Pour ce faire nous tapons la commande dans la figure ci-dessous :

```
(mon-test-env) → Tests git:(master) x oudjirasign -t 1024
génération de paire de clefs... |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@| 100%

Les clefs sont générées avec succès et sont stockées dans les fichiers ci-dessous.
privatekey.pem, publickey.pem

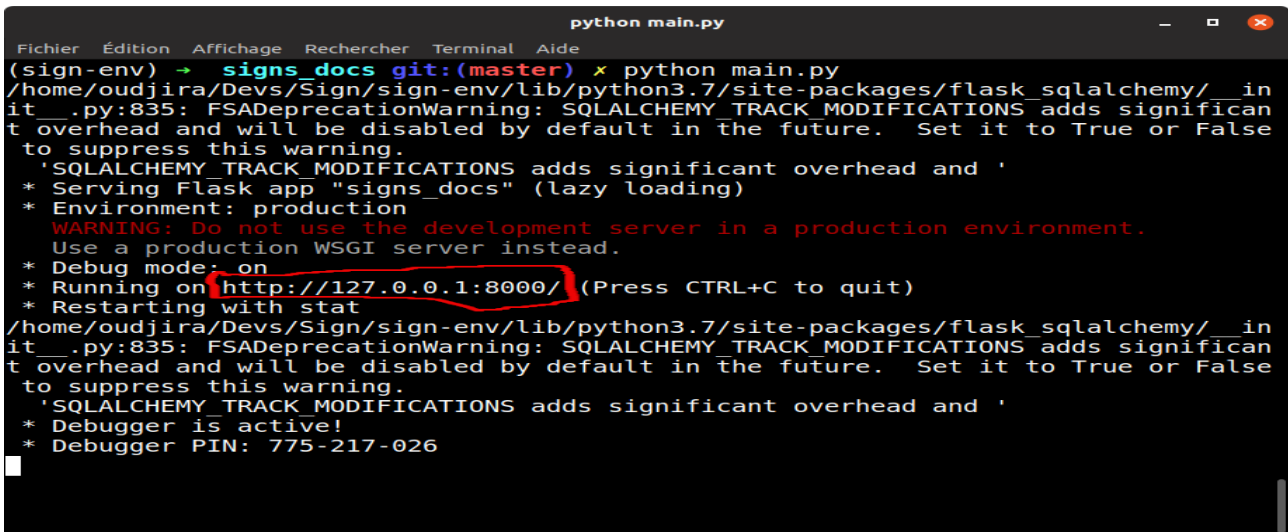
Voulez-vous générer un certificat associé à votre paire de clef(Y/n) : n
Bye.....:)
(mon-test-env) → Tests git:(master) x ls
mon-test-env privatekey.pem publickey.pem
(mon-test-env) → Tests git:(master) x
```

FIGURE 4.4 – Test du fonctionnalité générer paire de clef

Et en faisant un `ls` pour lister le contenu du dossier courant, nous pouvons constater que belle et bien notre paire de clef a été générée et stockée dans les fichiers.

4.5.2 Test de l'interface graphique du plate-forme

Pour lancer la plate-forme nous devons faire les étapes précédentes jusqu'à l'activation de l'environnement. Ensuite on démarre le serveur intégré de framework Flask avec la commande suivante :



```
python main.py
(sign-env) → signs_docs git:(master) × python main.py
/home/oudjira/Devs/Sign/sign-env/lib/python3.7/site-packages/flask_sqlalchemy/__init__.py:835: FSADeprecationWarning: SQLAlchemy TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future. Set it to True or False to suppress this warning.
'SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and '
* Serving Flask app "signs_docs" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:8000/ (Press CTRL+C to quit)
* Restarting with stat
/home/oudjira/Devs/Sign/sign-env/lib/python3.7/site-packages/flask_sqlalchemy/__init__.py:835: FSADeprecationWarning: SQLAlchemy TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future. Set it to True or False to suppress this warning.
'SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and '
* Debugger is active!
* Debugger PIN: 775-217-026
```

FIGURE 4.5 – Lancement du serveur intégré du framework python Flask

Nous pouvons juste entrer l'adresse locale `http://127.0.0.1:8000` dans la barre d'adresse d'un navigateur quelconques pour accéder au plate-forme web comme le montre la figure suivante.

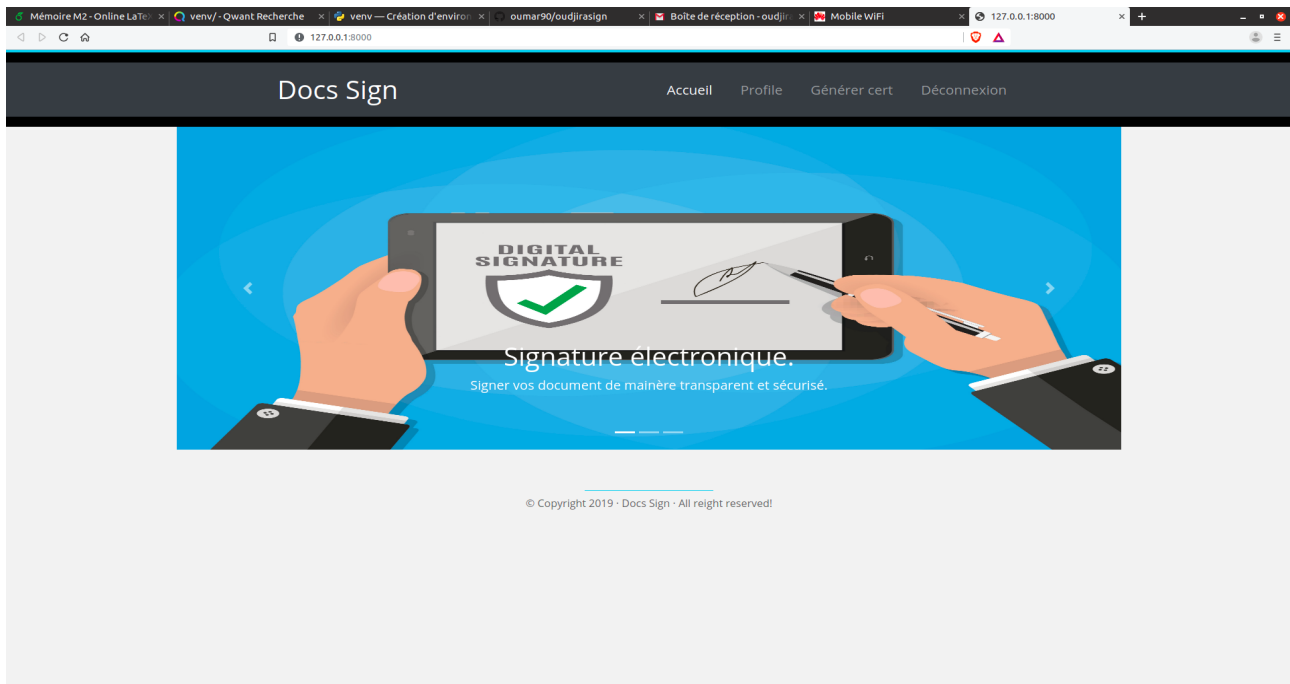


FIGURE 4.6 – page d'accueil de la plate-forme de signature électronique

Conclusion

Dans ce chapitre, il était question d'implémenter la plate-forme de signature électronique précédemment modélisée. De ce fait, nous avons d'abord créé un module python qui contient toutes fonctionnalités attendues et ensuite nous avons utilisé le **framework Python Flask** pour coder la dite plate-forme en faisant appel à ce module. Et nous avons fini par quelques tests de fonctionnalités attendues.

Chapitre 5

Résultats

Introduction

resultats ...

Conclusion

Conclusion

Bibliographie

- [San19] Céline Martinet SANCHEZ. *Concevez un site avec Flask*. 1/02/2019.
- [ROE19] CARINA ROELS. « Débutez l'analyse logicielle avec UML : Les différents types de diagrammes ». In : *OpenClassrooms* (24/04/2019).
- [Dum10] Renaud DUMONT. « Cryptographie et Sécurité informatique ». In : *Université de Liège Faculté des Sciences Appliquées* (2009 - 2010).
- [Gui12] Lagny Blanche GUILLAUME JERE. *Rapport du TER : SMS sécurisé Smart-Phones*. UNIVERSITÉ DE VERSAILLES. 23 Mai 2012.
- [VIL06] Alexandre VILLOING. « Implémenter une infrastructure à clés publiques dans un environnement Windows Server 2003 ». In : *SUPNIFO* (2mai 2006).
- [Rou18] Margaret ROUSE. « cryptographie asymétrique (cryptographie à clé publique) ». In : *LeMagIT* (20 mai 2018).
- [Dan10] Ghislain Dartois DANIEL BARSKY. *Cryptographie Paris 13*. 1 octobre 2010.
- [Pré16] place de la PRÉFECTURE À TOURS. « La signature électronique ». In : *Les Ateliers de la démat* (Jeudi 24 novembre 2016).
- [MONin] Lambert LOYE MONDJO. « CYCLE DE VIE D'UN LOGICIEL ». In : *SUPINFO* (Le 24 avril 2019, 09h19min).
- [Fro13] Roger FROST. « Des signatures électroniques authentifiées pour longtemps avec une norme ISO ». In : *ISO* (20 février 2013).
- [Cos18] Grégory COSTE. « Comment faire une signature électronique de document légale ? » In : *Appvizer* (21 février 2018).
- [Kol06] Dr.-Ing. KOLYANG. *Introduction au génie logiciel*. Sous la dir. de Kolyang et KA'ARANG : EDITIONS ET MÉDIA. Université de Ngaoundéré, Cameroun, 2006.
- [San12] Fernand Lone SANG. « Protection des systèmes informatiques contre les attaques par entrées-sorties ». In : *INSA de Toulouse* (2012).

- [MER18] GADJOU SOH TATIANA MERVEILLE. « CONCEPTION ET REALISATION D'UN SYSTEME DE CONTROLE D'ACCES PAR AUTHENTIFICATION FORTE (CARTE A PUCE + MOT DE PASSE) ». Mém.de mast. Ecole Nationale Supérieure Polytechnique De Maroua, 2018.
- [Wik18] WIKIPÉDIA. « CAdES — Wikipédia, l'encyclopédie libre ». In : (2018). [En ligne ; Page disponible le 28-mars-2018].
- [Wik19] WIKIPÉDIA. *Cryptographie asymétrique — Wikipédia, l'encyclopédie libre*. [En ligne ; Page disponible le 16-mai-2019]. 2019.
- [Brain] Serge BRAUDO. « Définition de Signature ». In : *DICTIONNAIRE DU DROIT PRIVÉ* (Vu le 19/05/2019 à 04h05min).
- [Lucin] LUCIDCHART. « Tutoriel sur les diagrammes de déploiement ». In : *LUCIDCHART* (Vue le 09/05/2019 à 11h01min).
- [FLAin] FLASK. *Mise en page du projet*. Vue le 07/07/2019 à 01h03min.
- [Quéin] Stanislas de QUÉNETAIN. *Hachage Cryptographique - Le guide pour tout comprendre*. Vue le 16/07/2019 à 21h32min.
- [CERin] CERTEUROPE. « Tout savoir sur la signature électronique ». In : *certeurope* (Vue le 26/05/2019 à 17h48min).
- [Jeain] Dominique Vaufreydaz JEAN-LUC PAROUTY Roland Dirlewanger. « La signature électronique, contexte, applications et mise en œuvre. » In : *INRIA/Direction des Réseaux et des Systèmes d'Information (DRSI)* (Vue le 27/mai/2019 à 20h11min).
- [Monin] Gaspard MONGE. « Public Key Infrastructure (PKI) ». In : *Institut d'électronique et d'informatique Gaspard-Monge* (Vu le 17 juin 2019 à 13h35min).
- [Wikin] WIKIPEDIA. « Cryptographie symétrique : Historique des versions ». In : *Le Parisien* (Vue le 16 juin 2019 à 03h59min).
- [ram09] RAM-0000. « Introduction à la Cryptographie ». In : *DEVELOPPEZ.COM* (Date de publication : 8 janvier 2009).

Annexes

Annexe A

Code source du module python

A.1 le contenu de fichier setup.py

Listing A.1 – Code python du fichier setup

```
1
2     from setuptools import setup
3
4     def readme():
5         with open('README.md') as f:
6             README = f.read()
7         return README
8
9     setup(
10         name="oudjirasign",
11         version="0.0.0.2",
12         description="Module_python_de_signature_electronique.",
13         long_description=readme(),
14         long_description_content_type="text/markdown",
15         url="https://github.com/oumar90/oudjirasign",
16         author="Oumar_Djime_Ratou",
17         author_email="oudjira90@gmail.com",
18         license="MIT",
19         classifiers=[
20             'Development_Status :: 4 - Beta',
21             'License :: OSI Approved :: MIT License',
22             'Programming_Language :: Python :: 3',
```



```
23         "Programming_Language::_Python::_3.7",
24     ],
25     packages=["oudjirasign"],
26     include_package_data=True,
27     install_requires=["pycryptodome", "progress", "cryptography", "arg
28     entry_points={
29         "console_scripts": [
30             "oudjirasign=oudjirasign:main",
31         ]
32     },
33 )
```

A.2 Contenu du fichier init

Listing A.2 – Code source python du fichier init

```
1  #!/usr/bin/env python3
2
3  import time
4  import datetime
5  import argparse
6  import codecs
7
8  from time import sleep
9  from progress.bar import Bar
10
11 from Crypto.PublicKey import RSA
12 from Crypto import Random
13 from Crypto.Cipher import PKCS1_OAEP
14 from Crypto.Hash import SHA256
15 from Crypto.Signature import PKCS1_v1_5
16
17 # importations des modules neccessaires pour le certificat
18 from cryptography import x509
19 from cryptography.x509.oid import NameOID
20 from cryptography.hazmat.primitives import hashes
21 from cryptography.hazmat.backends import default_backend
```

```
22 from cryptography.hazmat.primitives import serialization
23 from cryptography.hazmat.primitives.asymmetric import rsa
24
25
26 """
27 DESCRIPTION:
28
29     Comme son nom l'indique, oudirasign est un module de signature electronique
30     elle permet de signer numeriquement les documents electronique. En plus e
31     egalement de chiffrer/chiffre de message et bien d'autre.
32
33     """# importations des modules neccessaires pour le certificat
34
35
36
37 __author__="Oumar_Djime_Ratou"
38 __copyright__="Copy_Right_2019,_ITS"
39
40
41 # Generate rsa keys
42 def generatorsakeys(length=2048):
43     """ Fonction rsakeys(bits) permet generer une paire de cle RSA
44     elle prend en parametre la taille de cle et reourne un tuple (privatekey,
45
46     =====
47
48     Exemple :
49         (privatekey, publickey) = generatorsakeys(taille) // par default taille
50     """
51     generate_random_number = Random.new().read
52     key=RSA.generate(length, generate_random_number)
53     privatekey = key.exportKey()
54     publickey=key.publickey().exportKey()
55     return privatekey, publickey
56
57 # Importation de cle privee
58 def importPrivateKey(privatekey):
```

```
59     """ Cette fonction permet de importer la cle prive ,
60     elle prend en parametre use cle privee """
61     return RSA.importKey(privatekey)
62
63 # Importation de cle public
64 def importPublicKey(publickey):
65     """ Cette fonction permet de exporter la cle public ,
66     elle prend en parametre use cle public """
67     return RSA.importKey(publickey)
68
69 # Chiffrement un message
70 def chiffre(message, pubkey):
71     """ Cette fonction permet de chiffrer un message ,
72     elle prend en parametre le message et la cle public et retourne le messa
73
74     =====
75
76     Exemple :
77         message_chiffre = chiffre(message_clair, publickey)
78         """
79     #key = RSA.importKey(open(pubkey).read()) # Si la cle est stocker sur un
80     cipher = PKCS1_OAEP.new(pubkey)
81     ciphertext = cipher.encrypt(message.encode("utf-8"))
82
83     return ciphertext
84
85 # Dechiffrement d'un message
86 def dechiffre(ciphertext, privbkey):
87     """ Cette fonction permet de dechiffrer un message ,
88     elle prend en parametre le message chiffre et la cle privee
89     et retourne le message en claire .
90
91     =====
92
93     Exemple :
94         dechiffre = dechiffre(message_chiffre, privatekey)
95         """
```

```
96     #key = RSA.importKey(open(privbkey).read()) # Si la cle est stocker sur u
97     cipher = PKCS1_OAEP.new(privbkey)
98     message = cipher.decrypt(ciphertext).decode("utf-8")
99
100     return message
101
102 # Fonction de hachage
103 def hacher(message):
104     """ Cette fonction permet de hacher un message,
105         elle prend en parametre le message en claire .
106         Elle retourne le hache d'un message.
107
108         =====
109
110         Exemple :
111             hache = hacher(message_clair)
112     """
113
114     return SHA256.new(message.encode("utf-8"))
115
116 def hacherdocs(nomdoc):
117     """
118     Cette fonction permet de hacher un fichier(txt,pdf, docx etc),
119     elle prend en parametre le fichier en claire .
120     Elle retourne le hache d'un message.
121
122     =====
123
124     Exemple :
125         hache = hacherdocs(nomdoc)
126
127     """
128     f = open(nomdoc, "rb")
129     doc_read = f.read()
130     hach = SHA256.new(doc_read)
131     # hach = hach.hexdigest()
132     return hach
```

```
133
134 def signerdocs(hach,privatekey):
135     """ Cette fonction permet de signer un fichier (txt,pdf,docx,etc),
136     elle prend en parametre 02 arguments,
137     le hache et la cle privree et retourne la signature.
138
139     =====
140
141     Exemple :
142         signature = signerdocs(hach,privatekey)
143     """
144     sig = PKCS1_v1_5.new(privatekey)
145     return sig.sign(hach)
146
147
148 def verifierdocs(hach,publickey , signature):
149     """ Cette fonction permet de verifier un fichier (txt,pdf,docx,etc),
150     elle prend en parametre 03 arguments,
151     le message, la cle public et la signature. Retourne un boolean (True or F
152
153     =====
154
155     Exemple :
156         verif = verifierdocs(hach,publickey , signature)
157     """
158     sig = PKCS1_v1_5.new(publickey)
159     return sig.verify(hach , signature)
160
161
162 # Fonction de Signature
163 def signer(message,privatekey):
164     """ Cette fonction permet de signer un message,
165     elle prend en parametre 02 arguments,
166     le hache et la cle privree et retourne la signature.
167
168     =====
169
```

170 Exemple :

171 signature = signer(message_claire, privatekey)

172 """

173 hache = SHA256.new(message.encode("utf-8"))

174 hache.hexdigest()

175 sig = PKCS1_v1_5.new(privatekey)

176 signature = sig.sign(hache)

177 hexfy = codecs.getencoder('hex')

178 ms = hexfy(signature)[0]

179

180 # return signature

181 return ms.decode("utf-8")

182

183

184 # Fonction de Verification

185 def verifier(message, publickey, signature):

186 """ Cette fonction permet de verifier la signature d'un message,

187 elle prend en parametre 03 arguments,

188 le message, la cle public et la signature. Retourne un boolean (True or False)

189

190 =====

191

192 Exemple :

193 verifier = verifier(message, publickey, signature)

194 """

195 hache = SHA256.new(message.encode("utf-8"))

196 # hache.hexdigest()

197 signer = PKCS1_v1_5.new(publickey)

198

199 hexfy = codecs.getdecoder('hex')

200 ms = hexfy(signature)[0]

201

202 return signer.verify(hache, ms)

203

204 def generate_keys_rsa(taille=2048, mypass=b'password'):

205 """ Fonction generate_keys_rsa(taille=2048, mypass=b'password') permet
generer une paire de cle RSA

```
206     elle prend en parametre la taille de cle et le mot de passe pour proteger
207     et reourne un tri-tuple (key, privatekey, publickey).Par default le mot de
208
209     =====
210
211     Exemple :
212         (key, privatekey, publickey) = generatorsakeys(taille) // par default
213     """
214     # Generation de nos cles
215     key = rsa.generate_private_key(
216         public_exponent=65537,
217         key_size=taille,
218         backend=default_backend()
219     )
220     private = key.private_bytes(
221         encoding=serialization.Encoding.PEM,
222         format=serialization.PrivateFormat.PKCS8,
223         encryption_algorithm=serialization.BestAvailableEncryption(mypass
224     )
225     public_key = key.public_key()
226     public = public_key.public_bytes(
227         encoding=serialization.Encoding.PEM,
228         format=serialization.PublicFormat.SubjectPublicKeyInfo
229     )
230     return key, private, public
231
232
233 def generate_certificat_auto_sign():
234     """ Cette fonction permet de generer un certificat auto-signe, il ne pren
235     elle retourne un certificat.
236
237     =====
238
239     Exemple :
240         certificat = generate_certificat_auto_sign()
241     """
242     contry_name = input("Entrer le nom de votre pays : [Ex. CM, pour Cameroun
```

```
243 city_name = input("Entrer le nom de l'état ou de la province : [Ex. Centre] ")
244 locality_name = input("Entrer le nom de votre ville : [Ex. Yaounde] : ")
245 organi_name = input("Entrer le nom de votre organisation : [Ex. ITS] : ")
246 common_name = input("Entrer le nom de votre section : [Ex. SECURITE] : ")
247 nom_domaine = input("Entrer le nom de domaine : [Ex. groupits.cm] : ")
248
249 key, private, public = generate_keys_rsa()
250
251 # Generation de CSR (Certificate Signing Request)
252 # Divers details sur qui nous sommes. Pour un certificat auto-signé, le s
253 subject = issuer = x509.Name([
254     x509.NameAttribute(NameOID.COUNTRY_NAME, contry_name),
255     x509.NameAttribute(NameOID.STATE_OR_PROVINCE_NAME, city_name),
256     x509.NameAttribute(NameOID.LOCALITY_NAME, locality_name),
257     x509.NameAttribute(NameOID.ORGANIZATION_NAME, organi_name),
258     x509.NameAttribute(NameOID.COMMON_NAME, common_name),
259 ])
260
261 cert = x509.CertificateBuilder().subject_name(
262     subject
263 ).issuer_name(
264     issuer
265 ).public_key(
266     key.public_key()
267 ).serial_number(
268     x509.random_serial_number()
269 ).not_valid_before(
270     datetime.datetime.utcnow()
271 ).not_valid_after(
272     # notre certificate sera valide pour 10 jours
273     datetime.datetime.utcnow() + datetime.timedelta(days=30)
274 ).add_extension(
275     x509.SubjectAlternativeName([x509.DNSName(nom_domaine)]),
276     critical=False,
277 # Signer notre certificat avec notre cle prive
278 ).sign(key, hashes.SHA256(), default_backend())
279
```



```
280
281    ertif = cert.public_bytes(serialization.Encoding.PEM)
282
283     return ertif
284
285 def savekeys(privakeyname, publickeyname):
286     """ Fonction qui permet de sauvegarder les clefs dans les fichiers
287     privatekey.pem respectivement publickey.pem """
288
289     with open("privatekey.pem", "wb") as f_private:
290         f_private.write(privakeyname)
291         f_private.close()
292
293     with open("publickey.pem", "wb") as f_public:
294         f_public.write(publickeyname)
295         f_public.close()
296
297 def main():
298     """ Fonction principale """
299     # create argument parser object
300     parser = argparse.ArgumentParser(description = "Comme_son_nom_l'indique, ")
301
302     parser.add_argument("-t", "--taille", type = int, nargs = 1,
303                         metavar="taille", default = None, help = "genere_une_")
304
305
306     # parse the arguments from standard input
307     args = parser.parse_args()
308
309
310     bar = Bar('generation_de_paire_de_clefs ...', fill='@', suffix='%(percent)')
311     for i in range(100):
312         sleep(0.001)
313         key, private, public = generate_keys_rsa(args.taille[0])
314         savekeys(private, public)
315         bar.next()
316     print()
```

```
317     print("\nLes_clefs_sont_generees_avec_succes_et_sont_stockees_dans_les_fi
318     print("privatekey.pem,_publickey.pem")
319
320     print()
321     reponse = input('Voulez-vous_generer_un_certificat_associe_a_votre_votre
322     if reponse == 'Y' or reponse == 'y' or reponse == '':
323         print("generation_du_certificat_en_cours...")
324         for i in range(100):
325             sleep(0.001)
326             cert = generate_certificat_auto_sign()
327             with open("cert_auto_signe.pem", "wb") as ct:
328                 ct.write(cert)
329                 ct.close()
330             print("Le_certificat_est_genere_avec_succes.")
331     elif reponse == "N" or reponse=='n':
332
333         print("Bye.....:")
334
335     else:
336         print("Bye.....:")
337
338 if __name__ == "__main__":
339     main()
```
