REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mouloud MAMMERI de Tizi-Ouzou

Faculté de Génie Electrique et d'Informatique

Département d'informatique

MEMOIRE

DE FIN D'ETUDE

En vue de l'obtention d'un diplôme de Master Académique en Informatique.

Option : Réseaux Mobilité et Systèmes Embarqués.

Conception et Réalisation d'un Module d'Authentification par cartes à puce pour applications web

---

Encadré par :

- *Mr H.ACHOUR.*

Réalisé par :

- *Mr RAHMOUNI Abdelfateh.*
- *Mr SI HADJ MOHAND Salim.*

Promotion 2017-2018

Chapitre II : Les cartes à puce

Chapitre III : Analyse

Chapitre IV : Conception

Chapitre V : Réalisation

Introduction

Depuis l'apparition de l'informatique, l'être humain a toujours essayé d'exploiter cette science pour automatiser ses tâches quotidiennes de gestion, de communication, de vente en ligne, de stockage de donnée et d'accès aux différents services d'une manière sécurisé.

Depuis quelques années, l'accès aux différents services est devenu un enjeu primordial surtout dans la sécurité, cela est dû aux besoins du monde actuel.

Avec l'avancement de la technologie, les mots de passe sont devenus facilement falsifiables et franchissables. C'est pour ça, que le besoin c'est fait ressentir pour trouver des nouvelles manières d'identification et d'authentification impossibles à falsifier, sûrs et efficaces. La possession d'une carte personnelle est l'une des technologies les plus adaptées car elle apporte un bon nombre de solutions aux problèmes de haute sécurité et de contrôle.

La richesse et l'intérêt de cette technologie est qu'elle joue le rôle d'une clé de serrure que seule celui qui la possède peut ouvrir une porte avec cette clé.

De plus le grand besoin d'accéder aux différents services peut être englobé dans une seule carte qui évite d'avoir pour chaque service son propre moyen d'identification prenant l'exemple d'une université, c'est le moyen le plus adapté pour accéder aux différents services (bibliothèque, restaurant, logement…etc.) par le biais d'une seule carte qui regroupe le tout et d'une manière sécurisé.

Au grand désir d'améliorer la qualité d'accès aux différents services d'un enivrement et d'une manière sécurisé, que nous allons aborder dans ce mémoire une problématique qui a pour thème la **conception et la réalisation d'un Module d'Authentification pour applications web** qui va permettre aux utilisateurs d'accéder aux différents services d'une manière sécurisé et avec une seule carte, et aussi un service de personnalisation qui va permettre la personnalisation et la mise à jour des cartes.

Notre tâche est donc de créer un système d'authentification à base de cartes à puce pour des applications Client/Serveur dans le but de renforcer la sécurité et de permettre l'accès à plusieurs services avec cette carte.

Pour cernés les objectifs fixés, nous avons adopté la structure suivante :

Introduction

- Le premier chapitre est « **La Sécurité dans les Systèmes d'information** »

  Ce chapitre expliquera les méthodes de sécurisation des systèmes.

- Le deuxième chapitre s'intitule « **Les cartes à puces** » pour comprendre le fonctionnement des cartes à puce.

- Le troisième chapitre est consacré pour « **Analyse »**
  Qui comprend l'analyse de notre projet et son étude détaillé.

- Le quatrième chapitre s'intitule « **Conception** » qui est consacré pour la conception de notre application ou nous avons opté pour le langage UML étant le mieux adapté pour démonter le fonctionnement de notre système.

- Le cinquième et le dernier chapitre est « **La réalisation »** de notre application et l'illustration de quelques interfaces.

Chapitre I :

La Sécurité dans les Systèmes d'information

Chapitre I la Sécurité dans les Systèmes d'information

Préambule :

Avec le développement de l'utilisation d'internet, de plus en plus d'entreprises et d'établissements ouvrent leur systèmes d'information à leurs partenaires ou leurs fournisseurs ou tout simplement à leur utilisateurs, il est donc essentiel connaître les ressources de l'entreprise à protéger et de maîtriser le contrôle d'accès et les droits des utilisateurs du système d'information. Il en va de même lors de l'ouverture de l'accès de l'entreprise sur internet.

Par ailleurs, avec le nomadisme, consistant à permettre aux personnels de se connecter au système d'information à partir de n'importe quel endroit, les personnels sont amenés à transporter une partie du système d'information hors de l'infrastructure sécurisé de l'entreprise.

Comme les techniques de piratages des systèmes d'information et les failles que peuvent rencontrer les systèmes de sécurités sont de plus en plus en développement, le partage éventuel des données nécessite une importante politique de protection.

1. Objectif de la sécurité :

Le système d'information est généralement défini par l'ensemble des données et des ressources matérielles et logicielles de l'entreprise permettant de les stocker ou de les faire circuler. Le système d'information représente un patrimoine essentiel de l'entreprise, qu'il convient de protéger.

Son objectif d'une manière générale, consiste à assurer que les ressources matérielles ou logicielles d'une organisation sont uniquement utilisées dans le cadre prévu :

- Empêcher des personnes non autorisées d'agir sur le système de façon malveillante.
- Empêcher les utilisateurs d'effectuer des opérations volontaires ou involontaires capables de nuire au système.
- Garantir la non-interruption d'un servi.

2. Services principaux de la sécurité réseau :

Pour remédier aux failles et pour contrer les attaques, la sécurité informatique se base sur un certain nombre de services qui permettent de mettre en place une réponse appropriée à chaque menace. Les principaux services sont :

- **La confidentialité** : consiste a protéger les données transmises contre les attaques passives, et protéger les flux de données contre l'analyse, et préserver le secret des données transmises. Seulement les entités communicantes sont capables d'observer les données.
- **L'intégrité de données** : Vérifier l'intégrité des données consiste à déterminer si les données n'ont pas été altérées durant la communication (de manière fortuite ou intentionnelle).
- **La disponibilité** : C'est la propriété qui permet à un système ou une ressource du système pour qu'il soit accessible et utilisable suite à la demande d'une entité autorisée.
- **La non répudiation** : Empêche l'émetteur ou le receveur de nier avoir transmis ou reçu un message.
- **L'authentification** : L'authentification consiste à assurer l'identité d'un utilisateur, c'est-à- dire de garantir à chacun des correspondants que son partenaire est bien celui qu'il croit être. [6]

Les causes de l'insécurité

On distingue généralement deux types d'insécurité :

- **L'état actif d'insécurité**, c'est-à-dire la non-connaissance par l'utilisateur des fonctionnalités du système, dont certaines pouvant lui être nuisibles (par exemple la non-désactivation de services réseaux non nécessaires à l'utilisateur)
- **L'état passif d'insécurité**, c'est-à-dire lorsque l'administrateur (ou l'utilisateur) d'un système ne connaît pas les dispositifs de sécurité dont il dispose

3. Les mécanismes de sécurité :

Cryptographie, Signature électronique et Certificat

3.1. La Cryptographie :

Les réseaux publics, tels qu'Internet, n'offrent aucun moyen de sécuriser les communications entre les entités. Les communications qui transitent par ces réseaux sont susceptibles d'être lues voire modifiées par des tiers non autorisés. Le chiffrement permet de prévenir la consultation des données, offre des moyens de détecter si les données ont été modifiées et fournit un mode de communication sécurisé via des canaux qui autrement ne sont pas sécurisés. Par exemple, les données peuvent être chiffrées à l'aide d'un algorithme de chiffrement, transmises dans un état chiffré et par la suite déchiffrées par le destinataire prévu. Si un tiers intercepte les données chiffrées, il lui sera difficile de les déchiffrer.

3.1.1. La cryptologie:

L'origine de la cryptologie mot réside dans la Grèce antique. La cryptologie est un mot composé de deux éléments : «cryptos », qui signifie caché et « logos » qui signifie mot. La cryptologie est aussi vieille que l'écriture elle-même, et a été utilisé depuis des milliers d'années pour assurer les communications militaires et diplomatiques. Par exemple, le célèbre empereur romain Jule César utilisait un algorithme de chiffrement pour protéger les messages a ses troupes. Dans le domaine de l'un de cryptologie peut voir deux visions : la cryptographie et la cryptanalyse. Le cryptographe cherche des méthodes pour assurer la sûreté et la sécurité des conversations alors que le Crypto analyse tente de défaire le travail ancien en brisant ses systèmes.

La cryptographie traditionnelle est l'étude des méthodes permettant de transmettre des données de manière confidentielle et la cryptanalyse, à l'inverse est l'étude des procédés cryptographiques, qui dépendent d'un paramètre appelé clé.

3.1.2. La Cryptanalyse :

La cryptanalyse s'oppose en quelque sorte à la cryptographie, c'est l'étude des faiblesses des systèmes cryptographiques, elle est effectuée généralement par un intrus qui met en œuvre des méthodes afin de retrouver des informations secrètes tel que la clé, message en clair à partir d'informations considérées comme publique (cryptogramme, algorithmes), la cryptanalyse est une des disciplines de la cryptologie.

Dans la cryptanalyse on part du principe que l'homme est faible et facilement soudoya le, ainsi la force d'un système doit reposer sur la force du principe utilisé.

Chapitre I la Sécurité dans les Systèmes d'information

Si le but de la cryptographie est d'élaborer des méthodes de protection, le but de la cryptanalyse est au contraire de casser ces protections.une tentative de cryptanalyse d'un système est appelé une attaque, et elle peut conduire à différents résultats :

- ***Cassage complet*** : le cryptanalyse retrouve la clef de déchiffrement.
- ***Obtention globale*** : le cryptanalyse trouve un algorithme équivalent à l'algorithme de déchiffrement, mais qui ne nécessite pas la connaissance de la clef de déchiffrement.
- ***Obtention locale*** : le cryptanalyse retrouve le message en clair correspondant à un message chiffrer.
- ***Obtention d'information*** : le cryptanalyse obtient quelque indication sur le messageen clair ou la clef (certains bits de la clef, un renseignement sur la forme du message en clair).

D'une manière générale, on suppose toujours que le cryptanalyse connait le détail des algorithmes, fonctions mathématiques ou protocoles employés. Même si ce n'est pas toujours le cas en pratique, il serait risqué de se baser sur le secret des mécanismes utilisés pour assurer la sécurité d'un système, d'autant plus que l'usage grandissant de l'informatique rend de plus en plus facile la reconstitution de l'algorithme à partir du programme.

3.2.Définition de la cryptographie

La cryptographie est l'art de chiffrer, coder les messages est devenue aujourd'hui une science à part entière. Au croisement des mathématiques, de l'informatique, et parfois même de la physique, elle permet ce dont les civilisations ont besoin depuis qu'elles existent : le maintien du secret. Pour éviter une guerre, protéger un peuple, il est parfois nécessaire de cacher des choses.

La cryptographie est une science mathématique qui comporte deux branches : la cryptographie et la cryptanalyse.

Le mot cryptographie est un terme générique désignant l'ensemble des techniques permettant de chiffrer des messages, c'est-à-d permettant de les rendre inintelligibles sans une action spécifique. Le verbe crypter est parfois utilisé mais on lui préférera le verbe chiffré.

La cryptanalyse, à l'inverse, est l'étude des procédés cryptographiques dans le but de trouver des faiblesses, en particulier, de pouvoir décrypter des messages chiffrés. Le décryptement est l'action consistant à trouver le message en clair sans connaitre la clef de déchiffrement. [7]

Chapitre I la Sécurité dans les Systèmes d'information

Vocabulaire de base :

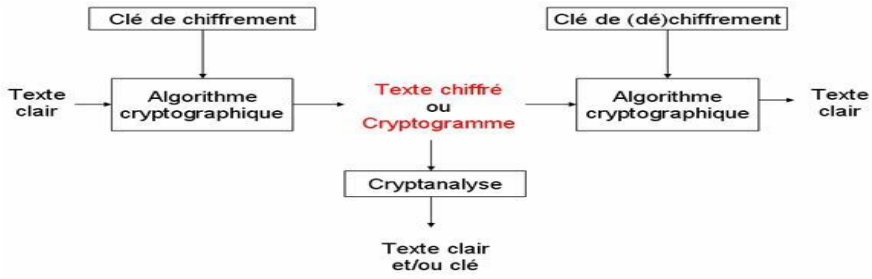


Figure I.1 Protocole de chiffrement

3.3. L'usage de la cryptographie :

La cryptographie est traditionnellement utilisée pour dissimuler des messages aux yeux de certains utilisateurs. Cette utilisation a aujourd'hui un intérêt d'autant plus grand que les communications via internet circulent dans des infrastructures dont on ne peut garantir la fiabilité et la confidentialité. Désormais, la cryptographie sert non seulement à préserver la confidentialité des données mais aussi à garantir leur intégrité et leur authenticité.

- **La confidentialité** : consiste à rendre l'information intelligible à d'autres personnes que les acteurs de la transaction.
- **L'intégrité** : vérifier l'intégrité des données consiste à déterminer si les données n'ont pas été altérées durant la communication.
- **L'authentification** : consiste à assurer l'identité d'un utilisateur, c.-à-d de garantir à chacun des correspondants que son partenaire est bien celui qu'il croit être un contrôle d'accès peut permettre (par exemple par le moyen d'un mot de passe qui devra être crypté) l'accès à des ressources uniquement aux personnes autorisées.
- **La non répudiation** : de l'information est la garantie qu'aucun des correspondants ne pourra nier la transaction.

3.4.Mécanisme de la cryptographie

Un algorithme de cryptographie ou un chiffrement est une fonction mathématique utilisée lors du processus de cryptage et de décryptage. Cet algorithme est associé à une clé(un mot, un nombre ou une phrase), afin de crypter une donnée. Avec des clés différentes, le résultat du cryptage variera également. La sécurité des données cryptées repose entièrement sur deux éléments : l'invulnérabilité de l'algorithme de cryptographie et la confidentialité de la clé. [5]

Qu'entend-on par clé ?

On appelle clé une valeur utilisée dans un algorithme de cryptographie, afin de chiffrer une donnée. Il s'agit en fait d'un nombre complexe dont la taille se mesure en bits. On peut imaginer que la valeur correspondant à 1024 bits est absolument gigantesque. Voir aussi bits bytes. Plus la clé est grande, plus elle contribue à élever la sécurité à la solution. Toutefois, c'est la combinaison d'algorithme complexe et de clés importantes qui seront la garantie d'une solution bien sécurisée.

Les clés doivent être stockées de manière sécurisée et de manière à ce que seul leur propriétaire soit en mesure de les atteindre et de les utiliser

3.5.Cryptographie moderne :

La cryptographie moderne s'intéresse généralement aux problèmes de sécurité des communications, Le but est d'offrir certain nombre de services de sécurité comme la confidentialité, l'intégrité, l'authentification des données transmises.

La cryptographie moderne ce compose de deux grandes parties : La cryptographie symétrique et la cryptographie asymétrique à base de clés.

Avant de les aborder, nous allons d'abord définir la notion de clé qui nous sera utile tout au long de ce chapitre :

Une clé : Paramètre constitué d'une séquence de symbole et utilisé par un algorithme cryptographique, pour transformer, valider, authentifier, chiffrer ou déchiffrer des données.

On distingue généralement deux types de clés :

Les clés symétriques: il s'agit de clés utilisées pour le chiffrement ainsi que pour le déchiffrement. On parle alors de chiffrement symétrique ou de chiffrement à clé secrète.

Les clés asymétriques: il s'agit de clés utilisées dans le cas du chiffrement asymétrique (aussi appelé chiffrement à clé publique). Dans ce cas, une clé différente est utilisée pour le chiffrement et pour le déchiffrement. [4]

3.5.1. Confidentialité et algorithmes de chiffrement

La confidentialité est le premier problème posé à la cryptographie. Il se résout par la notion de chiffrement. Il existe deux grandes familles d'algorithmes cryptographiques à base de clef :

3.5.2. Les algorithmes à clef secrète ou algorithmes symétriques :

La cryptographie à clé symétrique a très longtemps été utilisée pour le chiffrement de messages confidentiels. Son usage a été progressivement réduit depuis l'apparition de la cryptographie à clé publique (cryptographie asymétrique) même si les deux techniques sont encore parfois utilisées conjointement. Dans les algorithmes de chiffrement à clé symétrique ou clé secrète, c'est la même clé qui sert à la fois à chiffrer et à déchiffrer un message. C'est exactement le même principe qu'une clé de porte : c'est la même qui sert à ouvrir et à fermer une serrure.

Les algorithmes symétriques sont de deux types :

Les algorithmes de chiffrement en continu, qui agissent sur le message en clair un bit à la fois.

Les algorithmes de chiffrement par bloc, qui opèrent sur le message en clair par groupes de bits appelés blocs.

Caractéristiques :

- Les clés sont identiques : KE = KD = K
- La clé doit rester secrète
- Les algorithmes les plus répandus sont le DES, AES, 3DES,Blowfish, IDEA, …
- Au niveau de la génération des clés, elle est choisie aléatoirement dans l'espace des clés
- Ces algorithmes sont basés sur des opérations de transposition et de substitution des bits du texte clair en fonction de la clé
- La taille des clés est souvent de l'ordre de 128 bits. Le DES en utilise 56, mais l'AES peut aller jusque 256
- L'avantage principal de ce mode de chiffrement est sa rapidité
- Le principal désavantage réside dans la distribution des clés : pour une meilleure sécurité, on préfèrera l'échange manuel.

Malheureusement, pour de grands systèmes, le nombre de clés peut devenir conséquent. C'est pourquoi on utilisera souvent des échanges sécurisés pour transmettre les clés. En effet, pour un système à N utilisateurs, il y aura $N.(N-1)/2$ paires de clés.

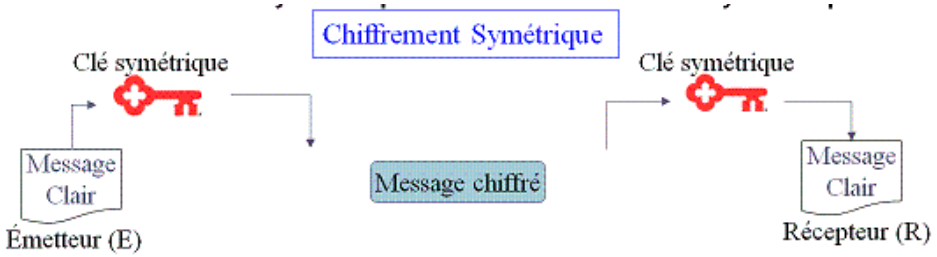


Figure I.2 Chiffrement symétrique

3.6.DES :

Le 15 mai 1973 le NBS (*National Bureau of Standards*, aujourd'hui appelé *NIST - National Institute of Standards and Technology*) a lancé un appel dans le *FederalRegister* (l'équivalent aux Etats-Unix du *Journal Officiel* en France) pour la création d'un algorithme de chiffrement répondant aux critères suivants :

- Posséder un haut niveau de sécurité lié à une clé de petite taille servant au chiffrement et au déchiffrement
- Être compréhensible
- Ne pas dépendre de la confidentialité de l'algorithme
- Être adaptable et économique
- Être efficace et exportable

Fin 1974, IBM propose « Lucifer », qui, grâce à la NSA (National Security Agency), est modifié le 23 novembre 1976 pour donner le DES (*Data Encryption Standard*). Le DES a finalement été approuvé en 1978 par le NBS. Le DES fut normalisé par l'*ANSI* (*American National Standard Institute*) sous le nom de *ANSI X3.92*, plus connu sous la dénomination *DEA* (*Data EncryptionAlgorithm*).

3.6.1. Principe du DES

Il s'agit d'un système de chiffrement symétrique par blocs de 64 bits, dont 8 bits (un octet) servent de test de parité (pour vérifier l'intégrité de la clé). Chaque bit de parité de la clé (1 tous les 8 bits) sert à tester un des octets de la clé par parité impaire, c'est-à-dire que chacun des bits de parité est ajusté de façon à avoir un nombre impair de '1' dans l'octet à qui il appartient. La clé possède donc une longueur « utile » de 56 bits, ce qui signifie que seuls 56 bits servent réellement dans l'algorithme. [2]

L'algorithme consiste à effectuer des combinaisons, des substitutions et des permutations entre le texte à chiffrer et la clé, en faisant en sorte que les opérations puissent se faire dans les deux sens (pour le déchiffrement). La combinaison entre substitutions et permutations est appelée code produit.

La clé est codée sur 64 bits et formée de 16 blocs de 4 bits, généralement notés *k1* à *k16*. Etant donné que « seuls » 56 bits servent effectivement à chiffrer, il peut exister 256 (soit 7.2*1016) clés différentes !

3.6.2. L'algorithme du DES

Les grandes lignes de l'algorithme sont les suivantes :

Fractionnement du texte en blocs de 64 bits (8 octets) ;

Permutation initiale des blocs ;

Découpage des blocs en deux parties: gauche et droite, nommées G et D ;

Etapes de permutation et de substitution répétées 16 fois (appelées rondes) ;

Recollement des parties gauche et droite puis permutation initiale inverse.

Inconvénient :

Aujourd'hui, le D.E.S. est fortement menacé par les puissances de calcul des ordinateurs. Il n'est en effet pas impossible de balayer la plupart des clés pour casser le code. Un nouveau système, le A.E.S. (Advanced Encryption Standard) est prévu pour le remplacer. [2]

3.7.AES [4]

Avec le temps, et les progrès de l'informatique, les 2^{56} clés possibles du DES n'ont plus représenté une barrière infranchissable. Il est désormais possible, même avec des moyens modestes, de percer les messages chiffrés par DES en un temps raisonnable. En janvier 1997, le NIST (*National Institute of Standards and Technologies*) des Etats-Unis lance un appel d'offres pour élaborer l'AES, *Advanced EncryptionSystem*. Le cahier des charges comportait les points suivants :

- évidemment, une grande sécurité.

- une large portabilité : l'algorithme devant remplacer le DES, il est destiné à servir aussi bien dans les cartes à puces, aux processeurs 8 bits peu puissants, que dans des processeurs spécialisés pour chiffrer des milliers de télécommunications à la volée.
- la rapidité.
- une lecture facile de l'algorithme, puisqu'il est destiné à être rendu public.
- techniquement, le chiffrement doit se faire par blocs de 128 bits, les clés comportant 128,192 ou 256 bits.

Au 15 juin 1998, date de la fin des candidatures, 21 projets ont été déposés. Certains sont l'œuvre d'entreprises (IBM,...), d'autres regroupent des universitaires (CNRS,...), les derniers sont écrits par à peine quelques personnes. Pendant deux ans, les algorithmes ont été évalués par des experts, avec forum de discussion sur Internet, et organisation de conférences. Le 2 octobre 2000, le NIST donne sa réponse : c'est le Rijndael qui est choisi, un algorithme mis au point par 2 belges, Joan Daemen et Vincent Rijmen. Depuis, le Rijndael, devenu AES, a été largement déployé et a remplacé progressivement le DES.

Principe de fonctionnement de l'AES

Le Rijndael procède par blocs de 128 bits, avec une clé de 128 bits également. Chaque bloc subit une séquence de 5 transformations répétées 10 fois :

- **Addition de la clé secrète** (par un ou exclusif).
- **Transformation non linéaire d'octets** : les 128 bits sont répartis en 16 blocs de 8 bits (8 bits=un octet), eux-même dispatchés dans un tableau 4×4. Chaque octet est transformé par une fonction non linéaire S. S peut être simplement vu comme une substitution sur les entiers compris entre 1 et 256. En particulier, elle peut être implantée sur ordinateur par un simple tableau.
- **Décalage de lignes** : les 3 dernières lignes sont décalées cycliquement vers la gauche : la 2ème ligne est décalée d'une colonne, la 3ème ligne de 2 colonnes, et la 4ème ligne de 3 colonnes.
- **Brouillage des colonnes** : Chaque colonne est transformée par combinaisons linéaires des différents éléments de la colonne (ce qui revient à multiplier la matrice 4×4 par une autre matrice 4×4). Les calculs sur les octets de 8 bits sont réalisés dans le corps à 2^8 éléments.

- **Addition de la clé de tour** : A chaque tour, une clé de tour est générée à partir de la clé secrète par un sous-algorithme (dit de cadencement). Cette clé de tour est ajoutée par un ou exclusif au dernier bloc obtenu.

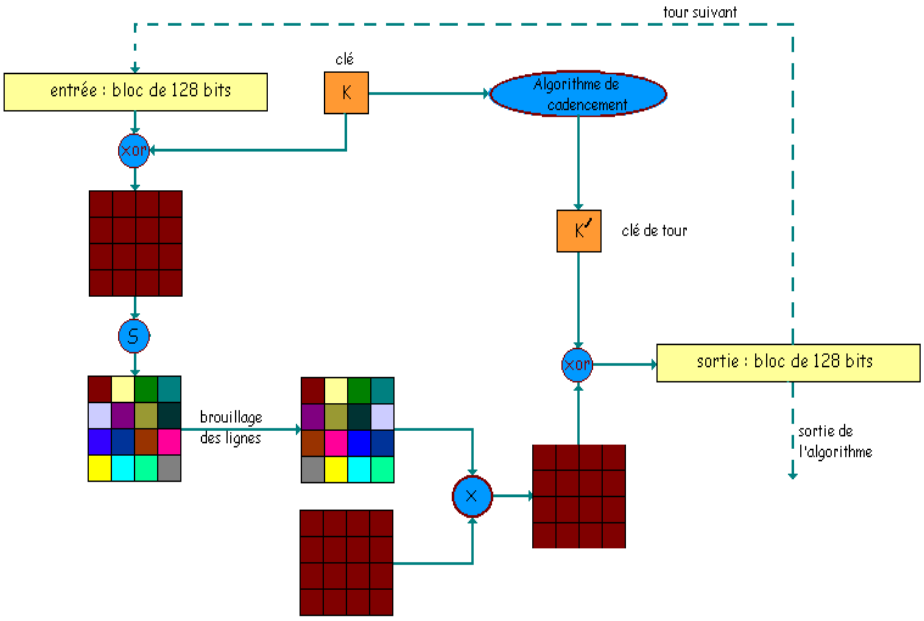


Figure I.3 Diagramme de fonctionnement AES

En conclusion, l'AES est plus sûr que le DES car il présente, entre autres, une plus grande résistance aux attaques par dictionnaires de clés. Les autres attaques ne sont pas applicables dans son cas. [4]

3.8.Les algorithmes à clef publique ou algorithmes asymétriques :

La cryptographie asymétrique 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 est 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.

Les principes généraux de la cryptographie à clé publique:

Un message codé avec une clé privée ne peut être décodé que par la clé publique associée.

Un message codé avec une clé publique ne peut être décodé que par la clé privée associée.

Une clé publique donnée ne peut être associée qu'à une seule clé privée.

Plusieurs clés privées différentes ne peuvent pas avoir la même clé publique comme clé complémentaire.

Une clé privée donnée ne peut être associée qu'à une seule clé publique.

Plusieurs clés publiques différentes ne peuvent pas avoir la même clé privée comme clé complémentaire.

Les principaux algorithmes à clé publique sont :

- protocole de Deffie-Hellman
- chiffrement de RSA
- chiffrement de Rabin
- chiffrement Elgamal
- chiffrement DSA

L'algorithme RSA :

Définition :

L'algorithme RSA (du nom de ses inventeurs Ron Rivest, Adi Shamir et Len Aldeman, qui ont imaginé le principe en 1978) est utilisé pour la cryptographie clé publique et est basé sur le fait qu'il est facile de multiplier deux grands nombres premiers mais difficile de factoriser le produit. C'est l'exemple le plus courant de cryptographie asymétrique, toujours considéré comme sûr, avec la technologie actuelle, pour des clés suffisamment grosses (1024, 2048 voire 4096 bits). [3]

Usages :

RSA, du nom de ces inventeurs, est un algorithme de chiffrement appartenant à la grande famille, "Cryptographie asymétrique".

RSA peut être utilisé pour assurer :

- **la confidentialité :** seul le propriétaire de la clé privée pourra lire le message chiffré avec la clé publique correspondante.
- **la non-altération et la non-répudiation** : seul le propriétaire de la clé privée peut signer un message (avec la clé privée). Une signature déchiffrée avec la clé publique prouvera donc l'authenticité du message.
  Sa robustesse réside dans la difficulté à factoriser un grand nombre.

3.8.1. Génération des clés :

Soient deux grands nombres premiers « aléatoirement » choisis : *p* et *q*.

Notons : $n = p*q$ et $\varphi = (p-1)*(q-1)$

Soient *d* un grand entier « aléatoirement » choisi, premier avec φ. Et *e* l'inverse de *d* modulo φ.

La clé publique de chiffrement est le couple (n,e), la clé privée de déchiffrement le couple (n,d).

Chiffrement

Avant d'être chiffré, le message original doit être décomposé en une série d'entiers *M* de valeurs comprises entre 0 et *n-1*.

Pour chaque entier *M* il faut calculer $C \equiv M^e [n]$.

Le message chiffré est constitué de la succession des entiers *C*.

Déchiffrement

Conformément à la manière dont il a été chiffré, le message reçu doit être composé d'une succession d'entiers *C* de valeurs comprises entre 0 et *n-1*.

Pour chaque entier *C* il faut calculer $M \equiv C^d [n]$.

Le message original peut alors être reconstitué à partir de la série d'entiers *M*.

Fiabilité :

La sécurité de l'algorithme RSA repose sur la difficulté à factoriser n. Pour décrypter le message, il est nécessaire de trouver d connaissant e, ce qui nécessite de recalculer φ, et donc de connaître p et q, les deux facteurs premiers de n. Or, la factorisation d'un entier (de très grande taille) en facteurs premiers est extrêmement difficile, cette opération nécessitant une capacité de calcul très importante. Pour exemple : en 2010, l'INRIA et ses partenaires ont réussi à factoriser une clé de 768 bits. Il leur a fallu deux ans et demi de recherche, et plus de 10^20 calculs. C'est à ce jour le meilleur résultat connu de factorisation.

Afin de se prémunir contre les puissances de calculs grandissantes, il est régulièrement recommandé d'utiliser des tailles de clés de plus en plus grandes (actuellement de 2048 bits). [3]

Les avantages et inconvénients de la cryptographie à clé publique comparé à la cryptographie à clé privée

Le premier avantage de la cryptographie à clé publique est d'améliorer la sécurité elle permet d'échanger des messages de manière sécurisée sans aucun dispositif de sécurité. L'expéditeur et le destinataire n'ont plus besoin de partager des clés secrètes via une voie de transmission sécurisée. Les communications impliquent uniquement l'utilisation de clés publiques et plus aucune clé privée n'est transmise ou partagée.

Avec un système à clé secrète, au contraire, il existe toujours le risque de voir la clé récupérée par une personne tierce quand elle est transmise d'un correspondant à l'autre. Toute personne interceptant la clé lors d'un transfert peut ensuite lire, modifier et falsifier toutes les informations cryptées ou authentifiées avec cette clé. De la norme de cryptage de données DES au code secret de *Jules César*, la distribution des clés reste le problème majeur du cryptage conventionnel. (Autrement dit, comment faire parvenir la clé à son destinataire sans qu'aucune personne ne l'intercepte ?) les moyens à déployer pour garantir la distribution sécurisée des clés correspondants sont très onéreux, ce qui constitue un inconvénient supplémentaire.

Le cryptage à clé publique représente une révolution technologique qui offert à tout citoyen la possibilité d'utiliser une cryptographie robuste. En effet, la cryptographie conventionnelle était auparavant la seule méthode pour transmettre des informations secrètes. Les couts d'institutions disposants de moyens suffisants, telles que gouvernements et banque.

Un autre avantage majeur des systèmes à clé publique est qu'ils permettent l'authentification des messages par signature électronique, ce qui peut aussi servir devant un juge, par exemple.

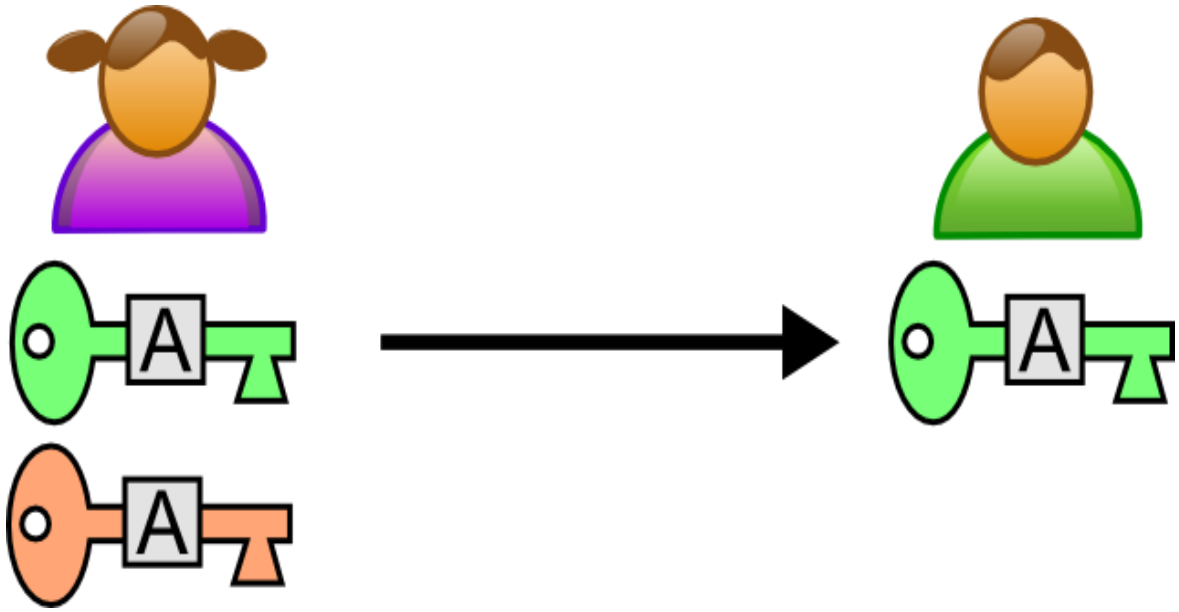
L'inconvénient des systèmes à clé publique est leur vitesse contrairement aux méthodes à clé secrète qui sont plus rapide. Ils sont particulièrement adaptés à la transmission de grandes quantités de données. Mais les deux méthodes peuvent être combinées de manière à obtenir le meilleur de leurs systèmes. Pour le cryptage, la meilleure solution est d'utiliser un système à clé publique pour crypter une clé secrète qui sera alors utilisée pour crypter fichiers et message.

Différence entre chiffrement et codage

Les opérations de chiffrement et du codage font partie de la théorie de l'information. La différence essentielle réside dans la volonté de protéger les informations et d'empêcher des tierces personnes d'accéder aux données dans le cas du chiffrement. Le codage consiste à transformer de l'information (des données) vers un ensemble de mots. Chacun de ces mots est constitué de symboles. La compression est un codage : on transforme les données vers un ensemble de mots adéquats destinés à réduire la taille mais il n'y a pas de volonté de dissimuler (bien que cela se fasse implicitement en rendant plus difficile d'accès le contenu).

Le « code » dans le sens cryptographique du terme travaille au niveau de la sémantique (les mots ou les phrases). Par exemple, un code pourra remplacer le mot « avion » par un numéro. Le chiffrement travaille sur des composants plus élémentaires du message, les lettres ou les bits, sans s'intéresser à la signification du contenu. Un code nécessite une table de conversion, aussi appelée « dictionnaire » (code book en anglais). Ceci étant, « code » et « chiffrement » sont souvent employés de manière synonyme malgré cette différence.

On peut aussi considérer que le chiffrement doit résister à un adversaire « intelligent » qui peut attaquer de plusieurs manières alors que le codage est destiné à une transmission sur un canal qui peut être potentiellement bruité. Ce bruit est un phénomène aléatoire qui n'a pas «d'intelligence » intrinsèque mais peut toutefois être décrit mathématiquement.



3.9.La signature électronique :

Définition :

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

Elle se différencie de la signature écrite par le fait qu'elle n'est pas visuelle, mais correspond à une suite de caractères. [1]

3.9.1. Fonctionnement de la signature numérique :

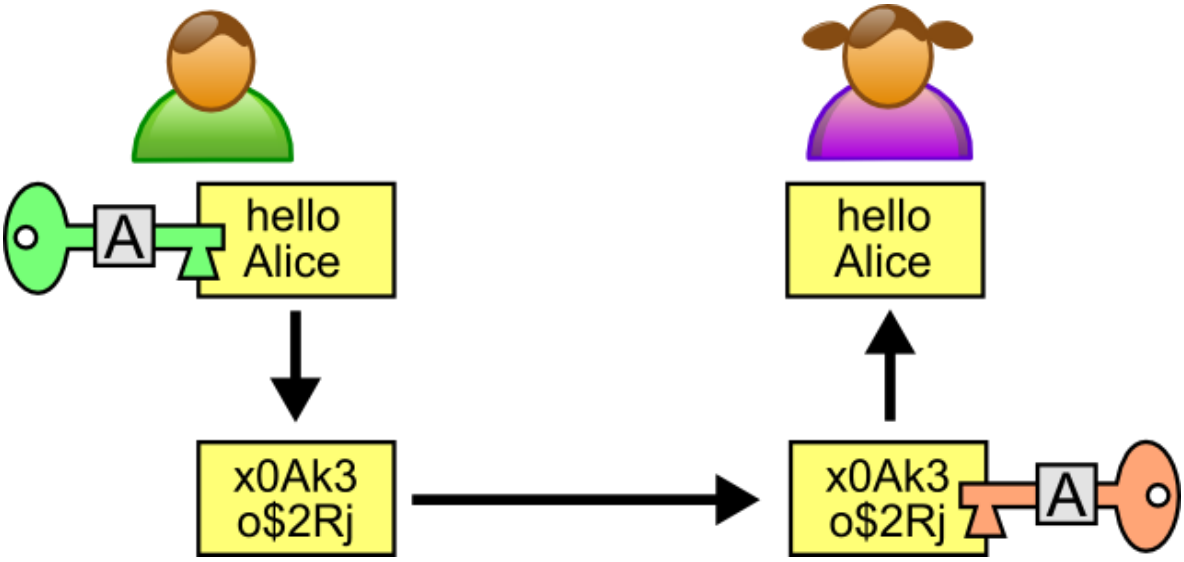
Les concepts de signature numérique sont principalement basés sur la cryptographie asymétrique. Cette technique permet de chiffrer avec un mot de passe et de déchiffrer avec un autre, les deux étant indépendants.

Par exemple, imaginons que Bob souhaite envoyer des messages secrets à Alice. Ils vont pour cela utiliser la cryptographie symétrique.

Alice génère tout d'abord un couple de clés. Une clé privée (en rouge) et une clé publique (en vert). Ces clés ont des propriétés particulières vis à vis des algorithmes utilisés. En effet, un message chiffré avec une clé ne peut être déchiffré qu'avec l'autre clé. Il s'agit de fonctions à sens unique. [1]

Alice transmet ensuite la clé publique (en vert) à Bob.

Grâce à cette clé, Bob peut chiffrer un texte et l'envoyer à Alice.



En utilisant la clé publique d'Alice, Bob est certain de deux choses :

- Personne ne peut lire le message, puisqu'il est crypté
- Seule Alice peut déchiffrer le message, car elle est la seule à posséder la clé privée.

Nous venons de répondre au besoin de confidentialité des données.

Mais la cryptographie asymétrique peut être utilisée d'une autre façon. En effet, on peut également utiliser la clé privée pour chiffrer, la clé publique servant alors à déchiffrer.

Le message ainsi chiffré est lisible par toute personne disposant de la clé publique. Ceci n'est pas très utile si l'on cherche la confidentialité. En revanche, une seule personne est susceptible d'avoir chiffré ce message : Alice. Ainsi, si l'on peut déchiffrer un message avec la clé publique d'Alice, c'est forcément la personne à avoir chiffré ce message.

Fonction de hachage :

Une fonction de hachage est un procédé à sens unique permettant d'obtenir une suite d'octets (une empreinte) caractérisant un ensemble de données. Pour tout ensemble de données de départ, l'empreinte obtenue est toujours la même.

Dans le cadre de la signature numérique, nous nous intéresseront tout particulièrement aux fonctions de hachage cryptographiques. Celles-ci assurent qu'il est impossible de créer un ensemble de données de départ donnant la même empreinte qu'un autre ensemble. Nous pouvons donc utiliser ces fonctions pour nous assurer de l'intégrité d'un document.

Les deux algorithmes les plus utilisées sont MD5 et SHA. A noter que MD5 n'est plus considéré comme sûr par les spécialistes. En effet, une équipe chinoise aurait réussi à trouver une collision complète, c'est à dire deux jeux de données donnant la même empreinte, sans utiliser de méthode de force brute.

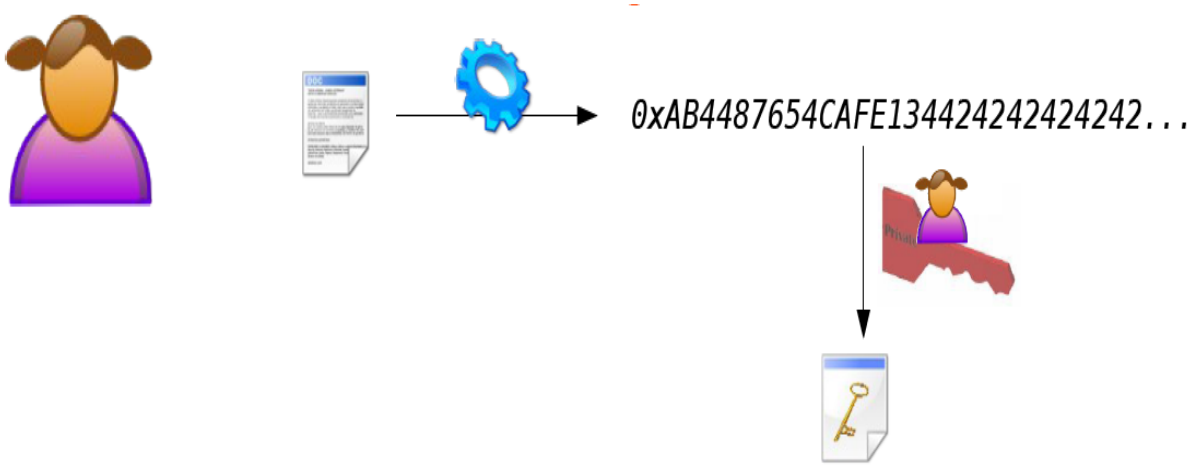
Singer un document :

La signature d'un document utilise à la fois la cryptographie asymétrique et les fonctions de hachage. C'est en effet par l'association de ces deux techniques que nous pouvons obtenir les 5 caractéristiques d'une signature (authentique, infalsifiable, non réutilisable, inaltérable, irrévocable).
Imaginons qu'Alice souhaite envoyer un document signé à Bob.

Tout d'abord, elle génère l'empreinte du document au moyen d'une fonction de hachage.

Puis, elle crypte cette empreinte avec sa clé privée.

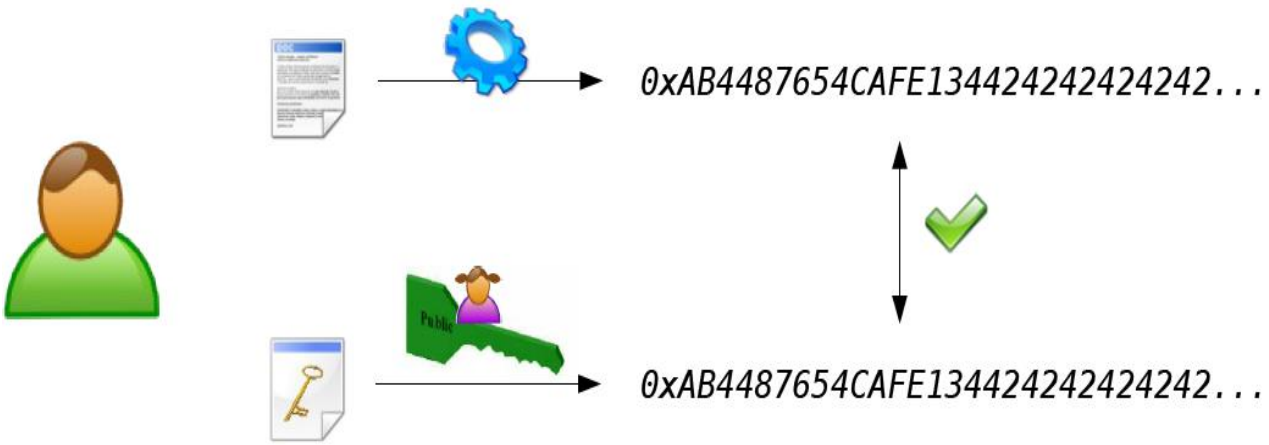
Elle obtient ainsi la signature de son document. Elle envoie donc ces deux éléments à Bob



Pour vérifier la validité du document, Bob doit tout d'abord déchiffrer la signature en utilisant la clé publique d'Alice. Si cela ne fonctionne pas, c'est que le document n'a pas été envoyé par Alice.

Ensuite, Bob génère l'empreinte du document qu'il a reçu, en utilisant la même fonction de hachage qu'Alice (On supposera qu'ils suivent un protocole établi au préalable).

Puis, il compare l'empreinte générée et celle issue de la signature.



Si les deux empreintes sont identiques, la signature est validée. Nous sommes donc sûr que :

C'est Alice qui a envoyé le document,

Le document n'a pas été modifié depuis qu'Alice l'a signé.

Dans le cas contraire, cela peut signifier que :

Le document a été modifié depuis sa signature par Alice,

Ce n'est pas ce document qu'Alice a signé.

Un mécanisme de signature numérique doit présenter les propriétés suivantes :

- Il doit permettre au lecteur d'un document d'identifier la personne ou l'organisme qui a apposé sa signature (propriété d'identification).
- Il doit garantir que le document n'a pas été altéré entre l'instant où l'auteur l'a signé et le moment où le lecteur le consulte (propriété d'intégrité).

Pour cela, les conditions suivantes doivent être réunies :

- **Authentique** : l'identité du signataire doit pouvoir être retrouvée de manière certaine.
- **Infalsifiable** : la signature ne peut pas être falsifiée. Quelqu'un ne peut se faire passer pour un autre.
- **Non réutilisable** : la signature n'est pas réutilisable. Elle fait partie du document signé et ne peut être déplacée sur un autre document.
- **Inaltérable** : un document signé est inaltérable. Une fois qu'il est signé, on ne peut plus le modifier.
- **Irrévocable** : la personne qui a signé ne peut le nier. [3]

Chapitre II :
Les cartes à puce

Préambule :

La carte à puce ne date pas d'hier contrairement à la pensée générale, les cartes à puce n'ont pas été inventées par un français. En effet, une légende urbaine attribut cette invention au français Roland Moreno en 1974. Le premier à avoir eu l'idée de la carte à puce ou à cette époque à une mémoire portative, est l'ingénieur britannique Geoffrey Dummer. Cette idée à été exploité et développée par les allemands Jürgen Dethloff et Helmut Gröttrup qui débouche sur un premier brevet en 1974.

1. Définition d'une carte à puce :

Une **carte à puce** est une carte en matière plastique, voire en papier ou en carton, de quelques centimètres de côté et moins d'un millimètre d'épaisseur[1], portant au moins un circuit intégré capable de contenir de l'information. Le circuit intégré (la *puce*) peut contenir un microprocesseur capable de traiter cette information, ou être limité à des circuits de mémoire non volatile et, éventuellement, un composant de sécurité (*carte mémoire*).

Les cartes à puce sont principalement utilisées comme moyens d'identification personnelle (carte d'identité, badge d'accès aux bâtiments, carte d'assurance maladie, carte SIM) ou de paiement (carte bancaire, porte-monnaie électronique) ou preuve d'abonnement à des services prépayés (carte de téléphone, titre de transport) . La carte peut comporter un hologramme de sécurité pour éviter la contrefaçon. La lecture (l'écriture) des données est réalisée par des équipements spécialisés, certaines puces nécessitant un contact physique (électrique), d'autres pouvant fonctionner à distance (communication par ondes radio). [1]

2. Fonctionnement :

La carte à puce succède :

Aux cartes embossées.

Aux cartes à codes barres.

Aux cartes plastiques à pistes magnétiques.

Quatre catégories de carte à puce sont référencées par le Conservatoire National des Arts & Métiers.

Elles se différencient par les moyens de contrôle d'accès et/ou par le mode de communication.

Communication par contacts et/ou radiofréquences

Contrôle d'accès par microprocesseur ou par logique câblée, celle-ci pouvant être élémentaire (moins de 50 portes) ou complexe.

La logique à haute intégration est mise en œuvre dans la TV payante, ainsi que dans certaines cartes RFID (multi-application, cryptographie DES, triple DES et RSA).

Les cartes à microprocesseurs, largement les plus répandues de nos jours, sont :

Mono-applicatives, comme les cartes bancaires B0' ou les cartes cryptographiques pour la sécurité informatique exploitant la technologie PKI

Multi-applicatives, comme les cartes bancaires Europay Mastercard Visa, ou les cartes SIM des téléphones mobiles.

3. L'architecture interne d'une carte à puce :

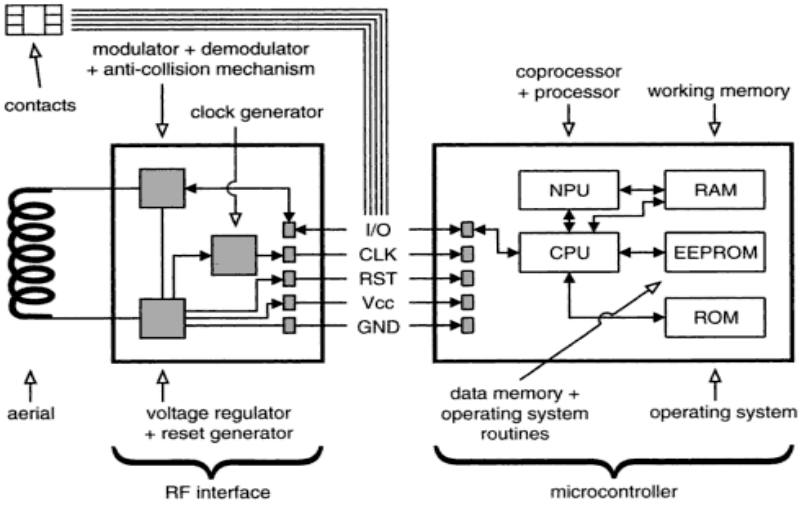


Figure II.3. Architecture interne d'une carte à puce

Après avoir vu l'architecture des carte a puces, ces dernières se distingues par deux principale famille qui cohabitent sur le marché selon leurs mode d'utilisation et leur possibilité qui différent de manière très importante que nous verrons ci-dessous.

3.1.Les cartes synchrones (carte à mémoire) :

La première famille, qui est aussi la plus ancienne, est la carte à mémoire appelée aussi carte synchrone en raison de son protocole de dialogue. Bien que ce soit une carte à puce, puisqu'une mémoire est bel et bien une puce de circuit intégré ; cette famille de carte est cantonnée à des applications relativement simples. Elle ne contient en effet que de la mémoire. En fait, cette famille peut aujourd'hui être scindée à son tour en deux sous-familles avec les cartes à mémoire "simples" appelées cartes à mémoire "tout court" et les cartes dites à mémoire protégée.

Les cartes à mémoire (sous-entendu simples) ne contiennent qu'une zone mémoire et le minimum de logique nécessaire pour pouvoir y accéder via les signaux définis à la partie **(Dimensions et emplacements des contacts ISO7816-2)** qu'on verra un peut plus bas

Ce sont des cartes de ce type qui étaient utilisées pour réaliser les cartes téléphoniques françaises de première génération appelées aussi T1G. Le niveau de sécurité offert par ces cartes est très faible et ne peut reposer que sur des artifices relativement simples qui ne résistent pas plus de quelques minutes en face d'une personne compétente et armée de mauvaises intentions. France Télécom en a d'ailleurs fait les frais dans ses cabines publiques pendant des années.

Les cartes à mémoire protégées font un pas en avant en matière de sécurité en associant de la mémoire, dont certaines zones sont accessibles seulement en lecture ou seulement en phase de personnalisation de la carte, et de la logique permettant l'exécution d'automates simples allant jusqu'à la présentation de mots de passe ou autres succédanés de codes PIN.

Ces cartes sont plus sûres que les cartes précédentes mais elles ne permettent pas la mise en place des applications les plus complexes que sont les cartes bancaires, les cartes SIM ou bien encore les cartes de décryptage TV. Il faut en effet pour cela que la carte dispose d'une "intelligence" locale que n'ont pas les cartes à mémoire.

3.2.Les cartes asynchrones (à microcontrôleur) :

L'intelligence locale qui fait défaut aux cartes à mémoire existe seulement dans les cartes à puce à microcontrôleur, appelées aussi cartes asynchrones en raison de leur protocole de dialogue.

Ces cartes sont désignées aujourd'hui sous le vocable unique de cartes à puce, ou smart cards en anglais, alors que les précédentes se voient appelées cartes à mémoire ou memory cards. Cela peut parfois créer une légère confusion qui se dissipe bien vite lorsque l'on considère l'application supportée par la carte.

Les cartes à puce "intelligentes" renferment un microcontrôleur complet ; c'est à dire l'association en un seul circuit d'une unité centrale de microprocesseur, de mémoire morte, de mémoire vive, de mémoire EEPROM, d'une interface d'entrée/sortie série et de toute la logique nécessaire pour faire fonctionner tout cela.

C'est donc bien un véritable petit micro-ordinateur complet qui est contenu dans ces cartes ; micro-ordinateur dénué de clavier et d'afficheur et dont le seul moyen de communication avec le monde extérieur est sa ligne d'entrée/sortie unique et bidirectionnelle I/O.

L'unité centrale utilisée varie selon la fonction ou la vocation de la carte. On peut trouver de simples microcontrôleurs 8 bits des familles PIC de Microchip ou AVR d'Atmel mais aussi des processeurs beaucoup plus puissants, associés parfois à un crypto processeur comme dans certaines carte de décryptage télévision récentes.

4. Les types de cartes à puces

Bien que l'architecture interne de ces cartes soit similaire à celle des cartes classiques ou cartes a contacts, il existe deux types de carte à puce :

- **Les cartes à puces avec contacte.**
- **Les cartes à puces sans contacte.**

4.1.Les cartes à puces sans contacte :

Au lieu de la surface a contacte, la carte a puce sans contacte utilise un couplage électrique pour réaliser la connexion pour fonctionner la carte doit généralement être placé a proximité <3cm du lecteur.

Pour alimenter la carte, on utilise un couplage inductif ou capacitif .l'horloge peut être interne et les entres/sorties sont réaliser pas modulation de l'alimentation.

Avec les avancées technologiques, on voit cependant apparaitre des puces à consommation très faible et des batteries ultraplates.

Avantage :

Absence de contacts susceptible d'être salis ou endommagés

Elle ne doit pas être insérer dans un lecteur :

Moins de vandalisme sur les lecteurs

Plus rapide à manipuler, d'ou un débit d'utilisateur potentiellement plus élevé

Inconvénients :

Transfert assez lent.

Cout de fabrication plus élevée.

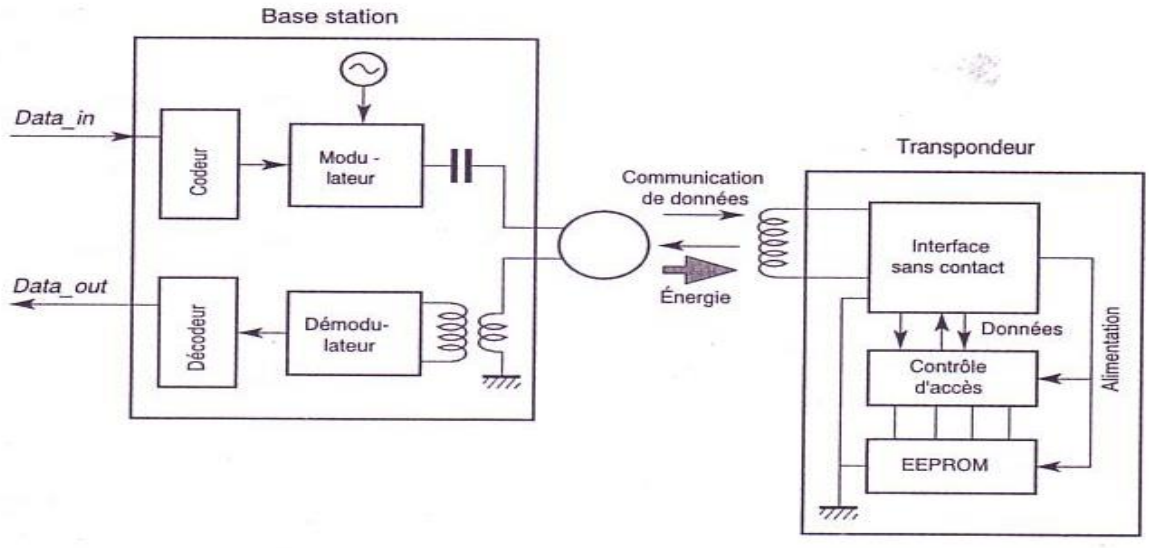
Les cartes peuvent être endommagées quand elles sont pliées.

La sécurité de transmission n'est pas assurées (susceptible d'être entendu).

Caractéristiques physique et électroniques :

La carte a puce sans contacte appartient a une famille de produit beaucoup plus vaste que l'on désigne par le sigle **RFID** (radio frequency identification devices) soit composants pour identification radio fréquence.

Et on voit comment s'effectuent la communication RFID dans la figure ci-dessous :



-

Figure II.4.1 Communication RFID d'une carte à puce sans contacte

4.2.Les cartes à puce avec contact :

Une carte à puce à contact nécessite un contact physique avec le lecteur pour échanger les informations.

Pour que le lecteur et la carte puissent communiquer, ISO (**International Organization for Standardization, Organisation Internationale de normalisation**) a définit une norme pour les cartes à puce à contact, qui est la norme ISO7816.

Caractéristiques physique (la norme ISO7816-1) :

Il existe trois formats pour les cartes à puce ID1, ID00et ID000, représenté dans les schémas qui suit :

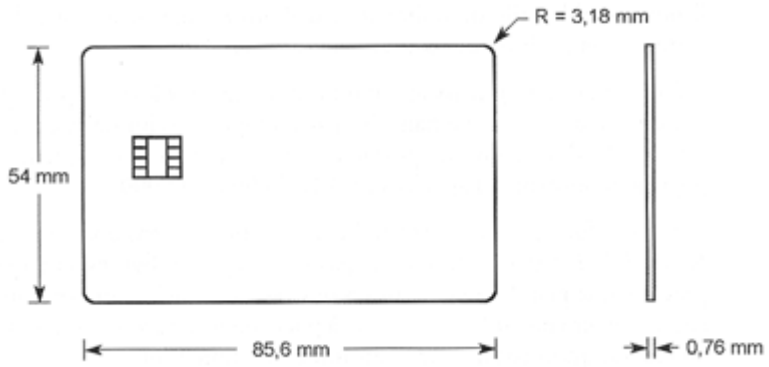


Figure II.4.2: format de carte à puce ID1

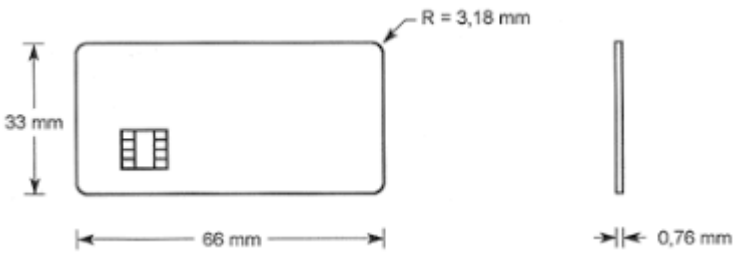


Figure II.4.3 : Format d'une carte à puce ID00

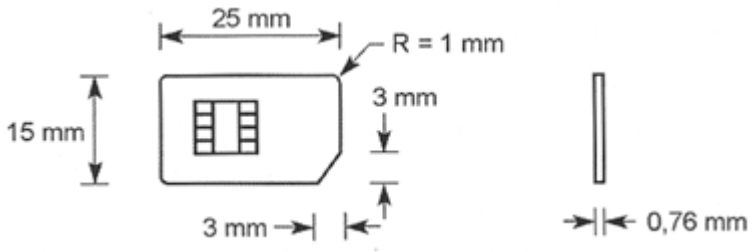


Figure II.4.4: Format d'une carte à puce ID000

Il faut aussi que la carte ne soit pas influencée par les rayons ultras violets, elle doit aussi résister aux chocs extérieurs et elle ne doit pas être endommagée par un champ magnétique statique de 79500 A.tr/m.

5. La technologie des cartes à puce

5.1.Les cartes à mémoire

Elles comportent un bloc de sécurité (optionnel) qui contrôle les accès à des mémoires de type ROM ou E2 PROM. Les télécartes de 1ière génération (ou TG1) n'étaient pas sécurisées; les télécartes de 2ième génération (ou TG2) comportent un bloc de sécurité

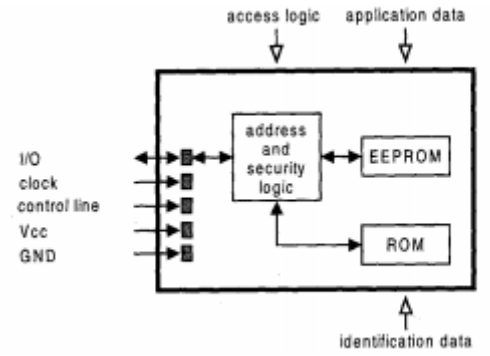


Figure II.5.1.bloc de sécurité de la carte à mémoire

5.2.Les cartes à microprocesseurs

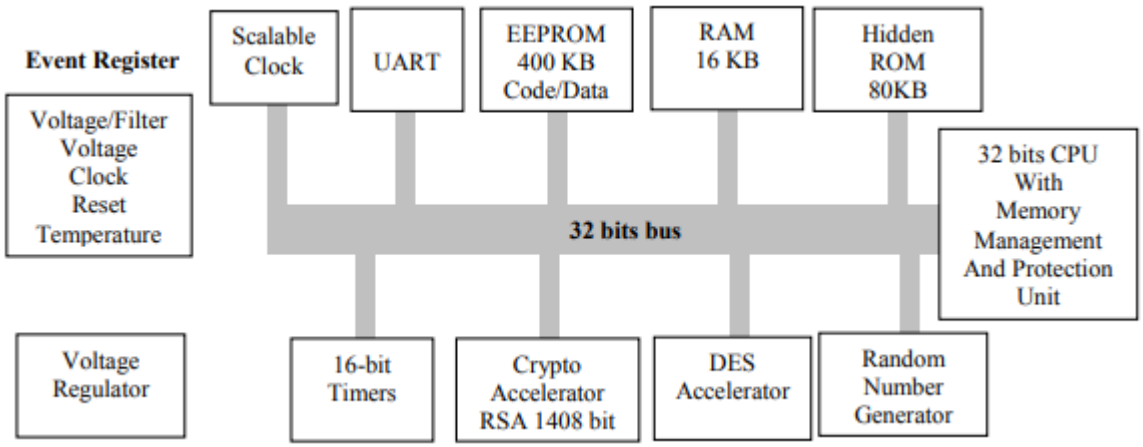


Figure II.5.2. Microcontrôleur 88CFX4000P

Un microcontrôleur se présente typiquement sous la forme d'un rectangle de silicium dont la surface est inférieure à 25 mm2. D'une part cette taille est imposée par les contraintes de flexion induite par le support en PVC, et d'autre part cette dimension limitée réalise un compromis entre sécurité physique et complexité du composant.

Les capacités mémoires sont comprises entre 128 et 256 Ko pour la ROM (surface relative 1), 64 et 128 ko pour l'E2 PROM (surface relative 4), 4 et 8 Ko pour la RAM (surface relative 16). En raison de ces contraintes technologiques la taille de RAM est modeste; l'E2 PROM occupe une portion importante du CHIP. Les écritures en E2 PROM sont relativement lentes (de l'ordre de 1 ms par mot mémoire de 32 à 64 octets), et le nombre de ces opérations est limité (de 104 à 106). L'introduction des mémoires FeRAM devrait amoindrir ces contraintes (109 opérations d'écriture, capacités mémoire de l'ordre du Mo, temps d'écriture inférieur à 200ns).

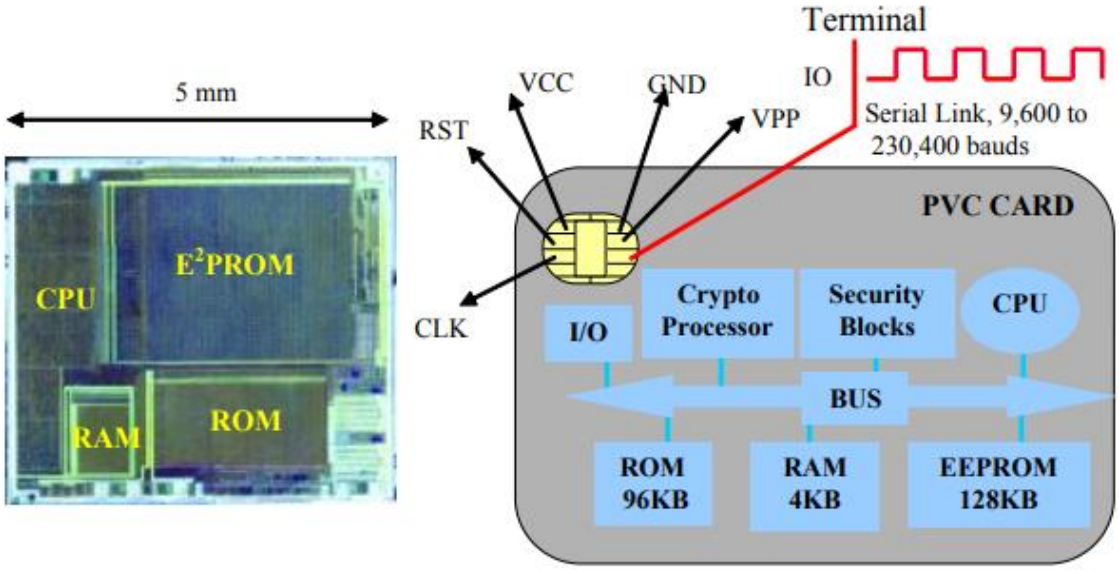


Figure II.5.3 composant de la carte à microcontrôleur

Les classiques processeurs 8 bits ont des puissances de traitements comprises entre 1 et 3 MIPS, ce paramètre est supérieur à 33 MIPS pour les nouvelles architectures à bases de processeurs RISC 32 bits. En termes de puissance de calcul cryptographique, les systèmes 32 bits réalisent un algorithme DES à un Mbit/s et un calcul RSA 1024 bits (clé privé) en moins de 300mS. A l'horizon 2004 l'introduction des technologies de type mémoires FLASH devrait conduire à des capacités de l'ordre de 1 Mo. Les puissances de calculs estimées son de l'ordre de 100 à 200 mips.

6. Couches de communications ISO 7816.

Requête.	Réponse.
CLA INS P1 P2 Lc [Lc octets]	sw1 sw2
CLA INS P1 P2 Le	[Le octets] sw1 sw2
CLA INS P1 P2 Lc [Lc octets] CLA C0 00 00 Le	61 Le [Le octets] sw1 sw2

Figure II.6.1. Commandes APDU définie par la norme ISO 7816

La norme ISO 7816 décrit l'interface de communication entre la puce et son terminal associé. Ce dernier fournit l'alimentation en énergie et une horloge dont la fréquence est typiquement 3.5 Mhz. L'échange de données entre puce et terminal est assuré par une liaison série dont le débit est compris entre 9600 et 230,400 bauds. La norme 7812-12 définir cependant une interface USB à 12 Mbit/s. Le terminal produit une requête (APDU) qui comporte conformément au protocole de transport T=0 au moins 5 octets (CLA INS P1 P2 P3) et des octets optionnels (dont la longueur Lc est précisée par la valeur de l'octet P3). La carte délivre un message de réponse qui comprend des octets d'information (dont la longueur Le est spécifiée par l'octet P3) et un mot de status (sw1 sw2, 9000 notifiant le succès d'une opération) large de deux octets. Lorsque la longueur de la réponse n'est pas connue à priori un mot de statu «61 Le» indique la longueur du message de réponse. Une fois ce paramètre connu le terminal obtient l'information au moyen de la commande GET RESPONSE (CLA C0 00 00 Le).

Les opérations de lecture et d'écriture, l'invocation des fonctions cryptographiques sont associées à des APDUs spécifiques. L'information stockée dans la puce sécurisée est organisée selon un système de fichiers qui comporte un répertoire racine (MF Master File), des sous répertoires (DF Dedicated File) et des fichiers (EF Elementary File). Chaque composant est identifié par un nombre de deux octets; la navigation à travers ce système s'effectue à l'aide d'APDUs particulières (SELECT FILE, READ BINARY, WRITE BINARY). La sécurité est assurée par des protocoles de simple ou mutuelle authentification (transportés par des APDUs), qui en cas de succès autorisent l'accès aux fichiers. La mise en œuvre d'une carte utilise donc un paradigme d'appel de procédure, transporté par des APDUs (généralement définies pour un système d'exploitation spécifique); l'information embarquée est connue à priori et classée par un système de fichier 7816.

7. Intégration des cartes à puce aux technologies de l'information.

L'intégration des puces sécurisées aux technologies de l'information implique l'adaptation des logiciels applicatifs de telle sorte qu'ils génèrent les APDUs nécessaires à l'utilisation des ressources embarquées. Schématiquement le middleware classique consiste à définir les éléments protocolaires (PE) requis par un service (Application Process) et exécutés dans la puce; chaque élément est associé à une suite d'APDUs (Application Protocol) variable selon le type de carte utilisée. L'application localisée sur le terminal utilise la puce aux moyens d'APIs (Application Programmatic Interface) plus ou moins normalisées (par exemple PC/SC pour les environnements win32), qui offre une interface de niveau APDUs ou plus élevé (PE).

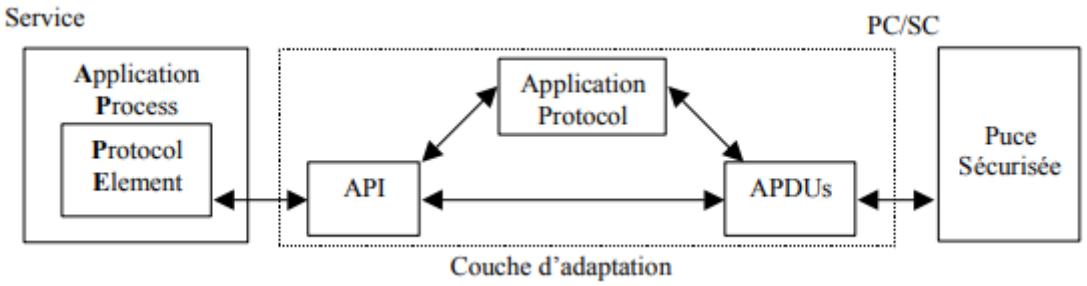


Figure II.7.1 Middleware classique d'une carte à puce.

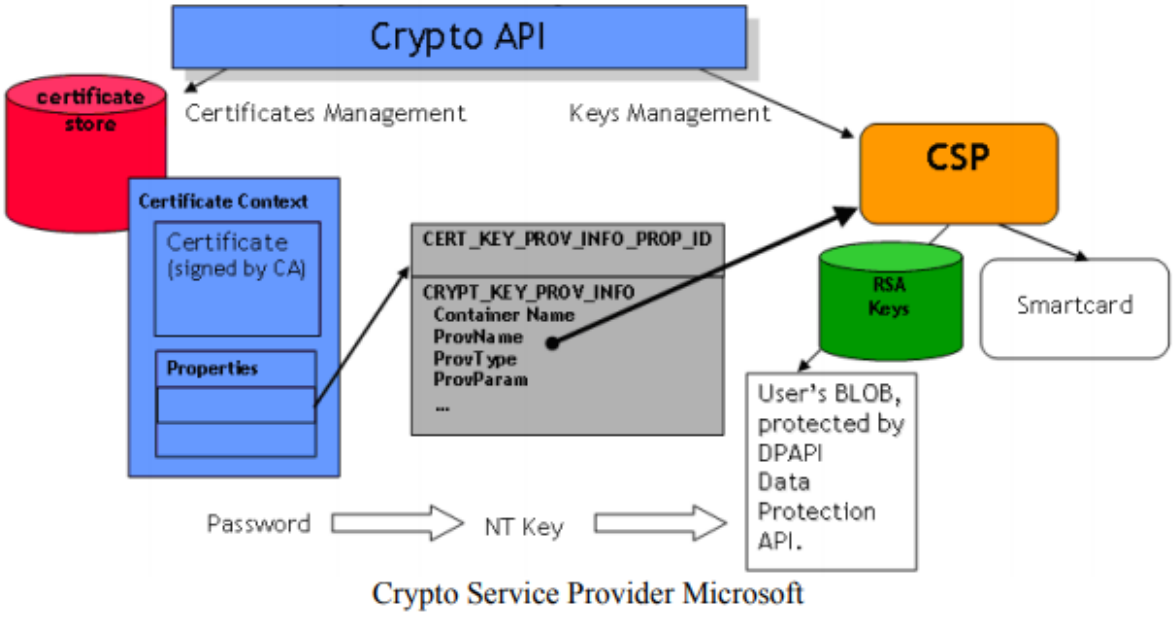


Figure II.7.2. Crypto Service Provider Microsoft

Système d'exploitation d'une carte à puce.

Schématiquement un système d'exploitation d'une carte à puce comporte les éléments suivants :

- Un bloc de gestion des ordres (APDUs) transportés par la liaison série.
- Une bibliothèque de fonctions cryptographiques, dont le code est réalisé de telle manière qu'il soit résistant aux attaques logiques connues (Timing attack, DPA, SPA, …).
- Un module de gestion de la RAM
- Un module de gestion de la mémoire non volatile (E2 PROM…), qui stocke les clés des algorithmes cryptographiques et les secrets partagés).
- Un module de gestion de la RAM, qui est une ressource critique en raison de sa faible quantité et de son partage entre procédures et applications.
- Un bloc de gestion d'un système de fichiers localisé dans la mémoire non volatile.
- Un module de gestion des événements indiquant une attaque probable de la puce sécurisée, comme par exemple :
    - Une variation anormale de la tension d'alimentation (glitch)
    - Une variation anormale de l'horloge externe de la puce sécurisée.
    - Une variation anormale de température.
    - La détection d'une perte d'intégrité physique du système. La surface d'une puce est généralement recouverte par un treillis métallique qui réalise une sorte de couvercle dont le système teste la présence.

Le système d'exploitation est contenu dans la ROM dont le contenu n'est pas chiffré. La connaissance de son code, bien que difficile

ne doit pas rendre possible des attaques autorisant la lecture de la mémoire non volatile.

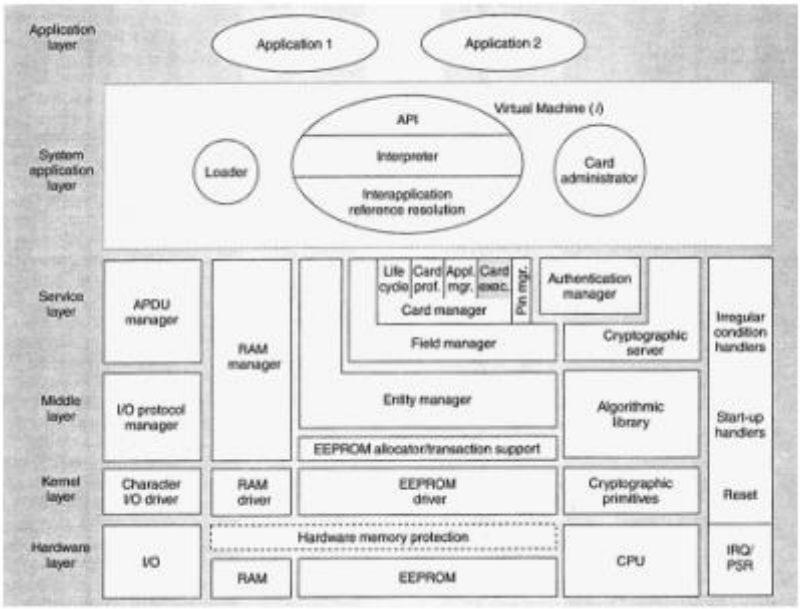


Figure II.7.3. Un exemple de système d'exploitation

8. Les cartes à puces Mifare DESFire :

Les cartes Mifare DesFire EV1 sont des cartes à puces sans contacte fabriquées par NXP, certifier EAL4+ , compatible avec les quatre première couches de ISO/IEC 14443A , et utilise optionnellement les commandes ISO/IEC 7816-4 .

Interface RF «Radio Frequency » :

La transmission avec le lecteur PC/SC [réf] est sans contact avec une portée qui atteint les 100mm, avec un système d'anticollision, la carte est alimentée par les champs RF.

La mémoire NON-volatile :

Selon le modèle de la DesFire EV1, la capacité mémoire est de 2K octets, 4K octets et 8K octets, avec 10 ans de conservation de données, et jusqu'à 500 000 cycles d'écriture.

Organisation mémoire :

Système de gestion de fichier flexible.

Jusqu'à 28 applications.

La taille de fichier est déterminée lors de sa création.

Plusieurs types de fichiers (voir plus bas).

Sécurité :

Numéro de série unique de 7 octets.

Authentification mutuelle (three-pass [réf], ISO/IEC 7816-4).

Une clé principale (master key) pour la carte et 14 clés par application.

Cryptage 3DES ou AES.

Application :

Chaque application est identifiée par un AID « Application Identifier » de 3 octets.

Chaque application peut avoir jusqu'à 32 fichiers.

Chaque application peut avoir jusqu'à 14 clés.

Plusieurs types de clés (DES, 2K3DES, 3K3DES, AES), dans une même application toutes les clés ont le même type.

Les types de fichiers

Standard Data File :

Permet de stocker des données brutes, les opérations possibles sont l'écriture et la lecture.

Backup Data File :

Idem que le Standard Data File mais il prend 2 fois plus de taille, mais il garantit le contenu du fichier si une coupure se produit lors de la transaction.

Value File :

Permet de stocker des données numériques, lors de sa création les valeurs limites inférieure et supérieure sont définit, les opérations possibles sont crédit de débit.

LinearReccord File :

Permet de stocker des multiples enregistrements de même taille, la taille et le nombre d'enregistrements sont définit lors de sa création.

Les opérations possibles sont l'ajout d'un enregistrement, la lecture et la suppression de tous les enregistrements.

Si le nombre d'enregistrements maximum est atteint un effacement total des enregistrements est nécessaire.

CyclicReccord File :

Idem que LinearReccord File, sauf lors de l'atteint de maximum d'enregistrements, le premier enregistrement qui sera réécrit. [13]

Structure de la carte :

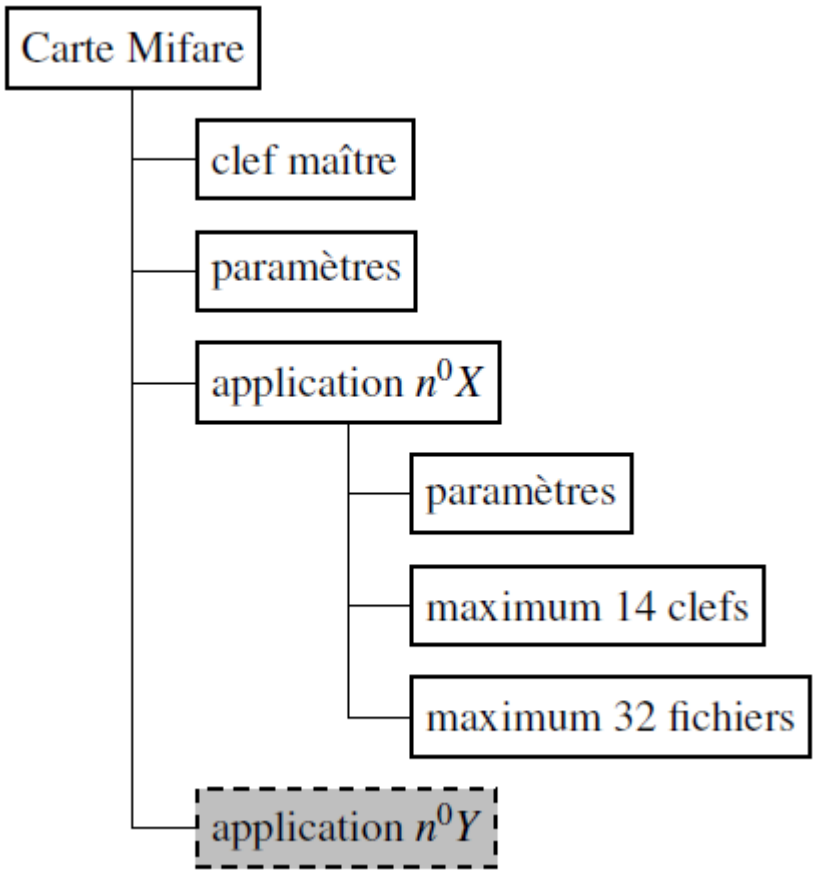


Figure II.8.: Structure générale de la carte [12].

9. La carte SAM (Secure Access Module)

Une SAM est basé sur les cartes à puces ou sur les circuits intégrés, pour traiter les fonctions de sécurités critiques et la sauvegarde des données sensibles, afin de renforcer la sécurité et la performance cryptographique d'un système.

9.1.Les fonctions fournies par la SAM :

Génération de clé principale « master key ».

Génération de clés des applications basées sur la clé principale.

Sauvegarde des clés.

Effectuer les fonctions cryptographiques.

Effectuer l'authentification mutuelle.

Génération des clés de session.

9.2.Les Types de SAM :

Host SAM :

Elle est utilisée pour vérifier les signatures numériques, ainsi que les données importantes (ex : le montant restant à payer des clients, …etc.), elle est comme une boite noire inviolable, elle doit être placée dans un environnement sécurisé.

SAM de terminal :

Elle est placée dans le lecteur de carte à puce, elle sauvegarde toutes les données sensibles pour établir la transaction, ainsi que les fonctions cryptographiques nécessaires, elle traite toutes les opérations critiques plutôt de lecteur.

SAM de personnalisation :

Elle est utilisée pour personnaliser d'autres cartes à puces, elle peut créer les clés principales basées sur l'UID « Unique IDentification », ainsi que les clés des applications, elle doit gérer des gros fichiers et avoir une bonne capacité de traitement.

9.3.Avantage des cartes SAM :

Les traitements critiques sont réalisés par la SAM plutôt de lecteur qui est considérer comme non fiable.

Les systèmes à base de SAM n'ont pas besoin de réaliser les opérations cryptographiques, car cette dernière les effectue. [14]

10. Java Card :

Définition :

Java Card est une norme ouverte de Sun Microsystems pour une plate-forme de développement de cartes à puce. Les cartes à puce créées à l'aide de la plate-forme Java Card sont dotées d'applets Java. Les applets peuvent être ajoutés ou modifiés après l'émission de la carte. [15]

Architecture

La carte est basée sur un interpréteur de bytecode java

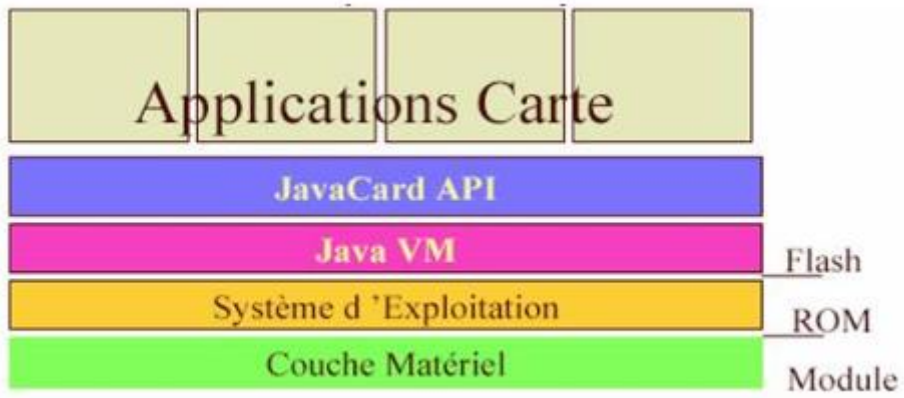


Figure II.10.1. Architecture java card

On retrouve donc la machine virtuelle java (JVM) au dessus du système d'exploitation. Comme pour les ordinateurs, chaque carte d'architecture devra avoir sa propre machine virtuelle. La machine virtuelle JavaCard est identique à celle de Java mais est découpée en deux parties : une sur la carte, l'autre hors carte ("Java Card Converter").[15]

Il est donc nécessaire de précompiler les applications avant de les charger sur la carte :

La figure ci dessous présente le cycle de développement d'une application JavaCard.

.

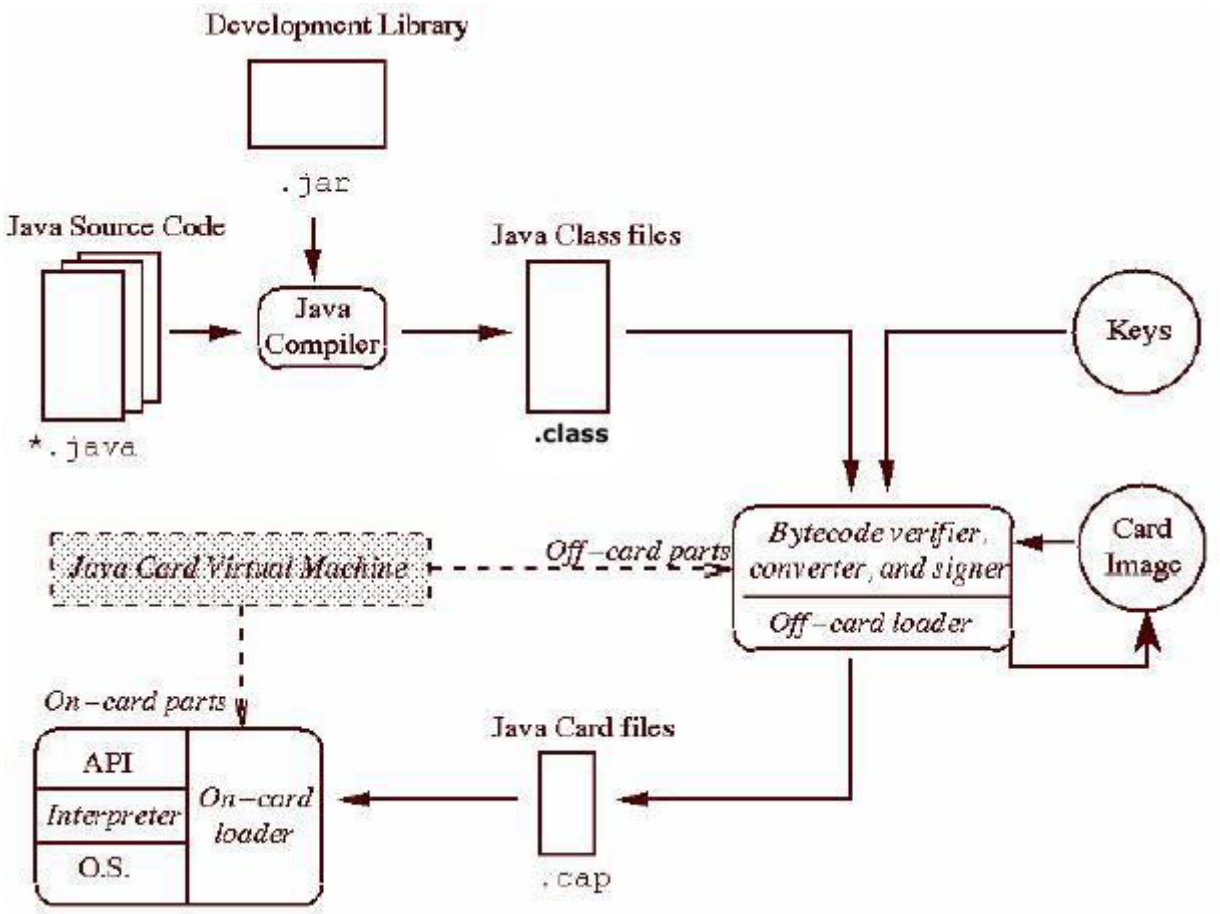


Figure II.10.2. Le cycle de développement d'une application JavaCard.

La programmation s'effectue sur n'importe quel environnement de programmation dans lequel on constitue des fichiers avec pour extension "java". On les compile ensuite afin de fournir le bytecode. Ce bytecode passe ensuite dans la machine virtuelle "hors carte" qui est découpée en 3 modules ayant chacun un rôle précis :

Le Vérifieur de ByteCode

Il utilise le vérifieur de Bytecode Java "classique"

Il contrôle le sous-ensemble JavaCard (langage + API)

Le Convertisseur en trois phases

Préparation : initialise les structures de données de la JCVM

Optimisation : ajuste l'espace mémoire, remplace certains InvokeVirtual par des InvokeStatic, ...

Edition de liens : résout les références symboliques à des classes déjà présentes dans la carte (via "image" de la carte).

Le Signeur

Il valide le passage par le vérificateur et le convertisseur par une signature vérifiée par la carte au moment du chargement.

Le programme fournit alors des fichiers ".cap" qui peuvent être chargés dans la carte. [16]

Chapitre III :

Analyse

Préambule:

Nous avons comme objectif de créer un système d'authentification en utilisant la technologie des cartes à puces, ainsi qu'un système de gestion et de personnalisation de ces cartes, qui pourrait être implémenté dans un environnement multiservices.

Dans ce chapitre nous verrons l'architecture globale de notre système, ainsi que la partie analyse dans laquelle nous détaillerons les étapes suivis pour la réalisation du projet

1. Problématique et objectif attendu :

Dans la plupart des systèmes l'accès aux données se fait en utilisant un mot de passe et un login, et ça parait sécurisé, ce qui n'est pas le cas car on voie que la plupart des systèmes sont faciles à y accéder par des personnes mal intentionnés.

Pour remédier à ce problème nous avons pris comme objectif la réalisation d'un système d'authentification basé sur la technologie des cartes à puces, ainsi qu'un environnement qui permet l'utilisation des cartes à puces sur n'importe quel système, aussi nous ferons une analyse détaillée sur l'environnement de test du système que nous allons réaliser et nous parlerons des modifications apportées au site proposé lors de l'implémentation de notre système conçu.

2. Architecture globale :

La figure suivante montre deux architectures différentes qui sont :

- L'architecture globale d'un système sans carte à puces
- L'architecture globale d'un système après l'intégration de notre solution basé sur les cartes à puces

2.1.L'architecture globale d'un système sans cartes à puces :

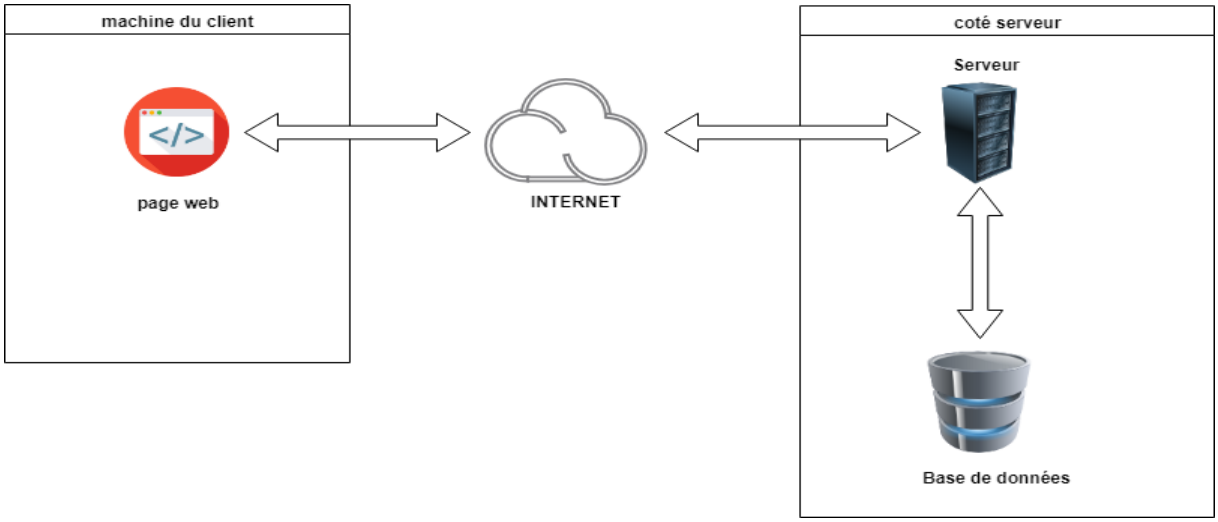


Figure III.1 : Architecture globale d'un système sans carte à puces

Dans un tel système l'identification ce fait par le bais d'un login et d'un mot de passe pour accéder à un espace privilégié.

Avantage :

- Simple à mettre en œuvre.
- Facilité d'accès.

Inconvénient :

- Facilité d'usurpation d'identité : il suffit de connaitre le login et le mot de passe pour accéder aux ressources.
- Pas fiable
- Il ne répond pas à la plupart des règles de la sécurité

2.2.Schéma globale d'un système avec carte à puce :

La figure suivante montre l'architecture globale d'un système utilisant notre solution devisée en trois parties, nous verrons l'essentiel dans ce chapitre et nous détaillerons les différentes parties dans le chapitre suivant qui est la conception.

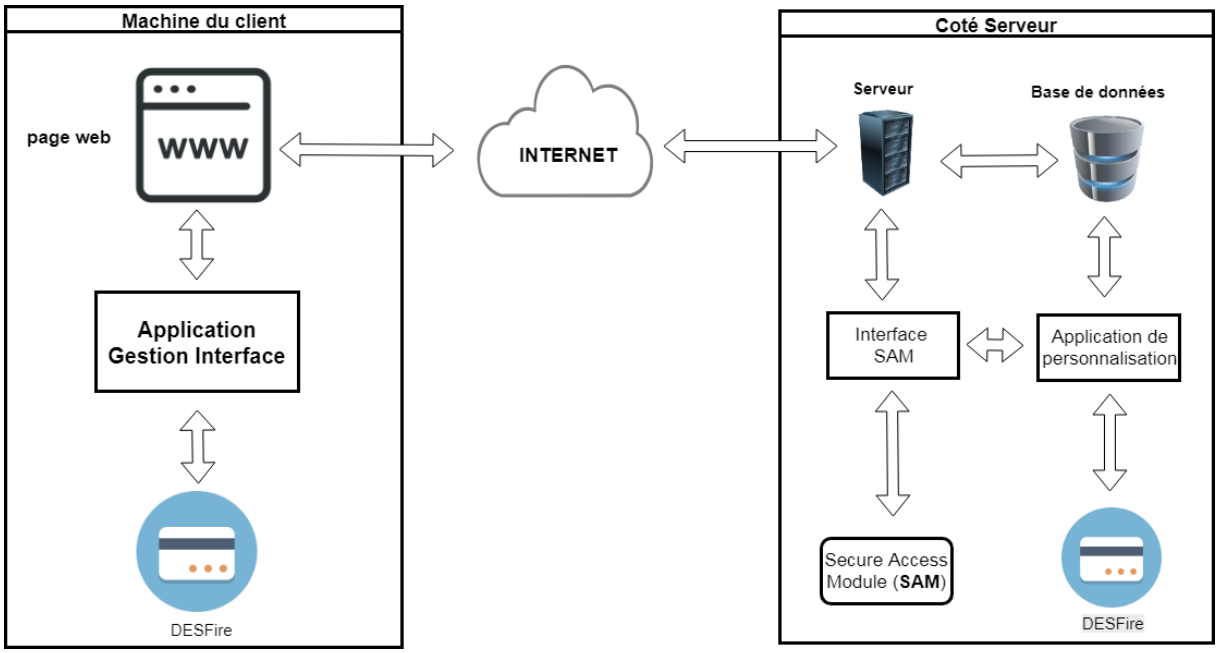


Figure III.2 : Architecture globale d'un système utilisant des cartes à puces

Dans un tel système l'accès se fait par une authentification via des cartes à puces et un mot de passe à fin d'accéder à un espace privilégié.

Avantage :

- Difficulté d'usurpation d'identité (presque impossible) : il faut la présence d'une carte et la possession du mot de passe pour s'authentifier.
- offre un environnement (outils, module et interface) qui permet d'utiliser d'intégrer des cartes à puces dans n'importe quel système.
- Offre un système de personnalisation qui permet de configurer des nouvelles cartes, ou mettre à jour des cartes déjà configurées.
- Toutes les clefs sont générées automatiquement par la SAM (Secure Access Module) : les clés ne sont pas sauvegardées dans la base de données ce qui offre une meilleure sécurité.

Inconvénient :

- Une solution plus coûteuse.

Notre application se compose de trois parties distinctes, partie utilisateur, partie personnalisation et partie serveur.

Chaque partie est composée de modules et d'interfaces, le module représente une entité logique « software » (programme ou partie logique), chaque modules et indépendant des autre modules, quant à une interface c'est une entité physique «hardware»

2.3.Les interfaces et les modules de notre système :

Notre système a besoin des interfaces et des modules suivants :

- **La SAM (Secure Accès Module) :** elle a pour rôle de générer des clés uniques et Pour ce faire on utilisera une JavaCard.

- **Le module « interface SAM » :**
  - Il a pour tâche de récupérer l'APDU de l'instruction à exécuter (la plus part des instructions)
  - De communiquer avec la SAM (il gère la file d'attente quand plusieurs utilisateur veulent se connecter en même temps).
  - En l'utilisant, chaque serveur avec n'importe quel langage de programmation peut l'interroger et exécuter des commandes.
  - Il peut s'exécuter sur la plupart des systèmes d'exploitation (Windows, Linux) essentiellement.

- **Le module « personnalisation » :** a pour fonction de
  - Mettre à jour les cartes DESFire.
  - Lire les données contenues dans les DESFire.
  - Configurer une nouvelle carte DESFire pour un utilisateur.
  - L'application doit communiquer avec la base de données afin d'écrire les données des utilisateurs.
  - L'application peut s'exécuter sur la plupart des systèmes d'exploitation.

- L'application doit être modulaire pour la configurer selon les données dans la base de données (évolution), pour cela nous utiliserons le design paterne MVC (Model View Controler) .

➢ **Le module « gestion d'interface » :**

- a pour rôle de Communiquer la page Web avec la couche PC/SC du système d'exploitation.
- Il doit être multi plateforme (doit pouvoir marcher sur WINDOWS, LINUX).
- Une solution souple qui fonctionne sur n'importe quel navigateur (Google Chrome, Safari …).

3. Analyse :

L'analyse est une étape est une étape essentielle de tout cycle de développement logiciel ou conceptuel qui consiste à effectuer une étude préalable. Le but de cette phase est de comprendre le contexte du système.

Il s'agit d'éclaircir au mieux les besoins fonctionnels et non fonctionnels, faire apparaitre les acteurs et identifier les cas d'utilisation.

3.1.Spécification des besoins :

La spécification des besoins constitue la phase de départ de toute application à développer dans laquelle nous allons identifier les besoins de notre application. Nous distinguons des besoins fonctionnels qui présentent les fonctionnalités attendus de notre application et les besoins non fonctionnels, pour éviter le développement d'une application non satisfaisante ainsi de trouver un accord commun entre fonctionnement et convivialité et de permettre son utilisation par le plu grand public.

Spécification des besoins fonctionnels :

Après une étude détaillée nous avons pu cerner différents besoins fonctionnels pour les futurs utilisateurs de notre application à savoir l'agent de personnalisation, l'utilisateur, et le serveur. Cette partie est réservée à la description des exigences fonctionnelles a savoir :

- La Fourniture d'une page d'authentification qui sera intégré dans un système.

- Fourniture d'une application permettant l'interaction entre la couche PC/SC du système d'exploitation et notre page d'authentification
- Fourniture d'une SAM [ref] qui permettra la génération automatique des clés
- Fourniture d'une interface qui permet l'interaction entre le serveur du système et la SAM, ainsi que la génération des commandes APDU qui seront exécuté par les cartes à puces des utilisateurs.
- Personnalisation des cartes à puces des utilisateurs :
    - Personnalisation manuelle et mise à jour.
    - Création/configuration des fichiers de personnalisation des cartes à puces.
    - Personnalisation automatique des cartes à puces via les fichiers de personnalisation.
- Authentification des utilisateurs via leurs cartes à puces et leur mot de passe.

Spécification des besoins non fonctionnels :

Les besoins non fonctionnels décrivent toutes les contraintes techniques, ergonomiques et esthétiques auquel est soumis notre système pour sa réalisation et son bon fonctionnement.

Vu que notre système est compatible avec n'importe quel service, donc notre système doit répondre à des besoins fonctionnels au système mais aussi à des besoins non fonctionnels.

Et en ce qui concerne notre système nous dégageons les besoins suivants :

- **La compatibilité** : notre système doit être multiplateforme (Linux, Windows), et il doit s'interfacer avec tous les langages coté serveur.
- **La disponibilité** : notre système doit être disponible pour être utilisé à n'importe quel moment.
- **La sécurité d'accès aux informations critiques :** les clés secrètes des cartes à puces sont accessibles seulement via la SAM.
- **La Souplesse** : chaque développeur peut utiliser notre système en fonction des besoins de son application.

3.2.Le choix des cartes :

Le choix des cartes est une étape très importante car un bon choix se fait en fonction de plusieurs facteurs qui sont :

- Le budget du projet : est un facteur essentiel qui a pour but de minimiser les achats de réalisation du système (matérielles, logicielle)
- Niveau de sécurité.
- Nombre d'applications supporté.
- Type d'application.
- Type de carte : choisir les cartes qui répondent à deux conditions nécessaires qui sont le cout de la carte et sa capacité.
- Solidité physique.

- **La Carte de L'utilisateur**

Suivant les critères cités ci-dessus nous avons opté pour les cartes à puces sans contacte **DESFire** qui répondes a la majorité des conditions et qui permettes de :

- Mettre en œuvre un système qui répond aux exigences de l'intéressé et de bénéficier d'une seule carte pouvant supporter plus de 20 application.
- Concevoir un système qui peut être modifié et implémenté sur n'importe quel système d'exploitation.
- Remédier aux problèmes liés à la sécurité

- **Secure Access Module (SAM) :**

On va réaliser une carte à partir d'une JavaCard celle-ci permettra d'augmenter le niveau de sécurité de n'importe quel système utilisant le nôtre car les clés seront stockées dans la **SAM**

3.3.Méthodologie et approche adoptée :

Avant de se lancer dans la programmation et l'écriture du code de notre système il faut tout d'abord organiser les idées, les documents, puis organiser la réalisation en définissant les modules et les étapes de la réalisation. Cette démarche que l'on appelle modélisation finira par produire un module.

La modélisation consiste à créer une représentation virtuelle d'une réalité de telle façon à faire ressortir les points auxquels on s'intéresse. Dans le cadre de notre projet on a utilisé la méthodologie UML pour la modélisation des différents diagrammes.

Présentation d'UML :

En regardant les objectifs fixés pour la réalisation du projet, nous remarquons que nous sommes face à un système orienté objet et qui devra rester ouvert pour les améliorations futures. De ce fait, il est très important d'utiliser un langage universel pour la modélisation afin de clarifier la conception et de facilité les échanges. Notre choix est porté sur le langage UML puisqu'il convient pour toutes les méthodes objet et se prête bien à la représentation de l'architecture du système.

3.4.Identification des acteurs :

Dans ce qui suit on va donner la définition d'un acteur et nous passeront à la présentation des acteurs de notre système suivi de leur rôle.

3.4.1. Définition d'un acteur :

Un acteur est toute entité externe (utilisateur, dispositif matériel ou autre système) qui interagit directement avec le système étudié, il peut consulter et/ou modifier directement l'état du système, en émettant et/ou recevant des messages susceptible d'être porteur de données, autrement dit un objet actif qui utilise les fonctions du système.

3.4.2. Extraction des acteurs et leurs tâches :

Après l'étude de l'existant, on a pu identifier (extraire) les acteurs qui seront les futurs utilisateurs de notre application. Dans notre cas, nous recensons les acteurs suivants :

L'utilisateur : C'est l'utilisateur principal de l'application, il représente toutes personnes ayant une carte **DESFire.** Ainsi notre système lui permet de :

- S'authentifier auprès du serveur d'une manière sécurisé.
- Accéder à son compte et satisfaire ses besoins avec toute confidentialité.
- Faire tous ce que permet la carte.

L'agent de personnalisation : C'est le second acteur de notre système, il représente une autorité qui va s'en charger des tâches suivantes :

- La mise à jour des cartes DESFire.
- La configuration de nouvelles cartes.
- La configuration du logiciel de personnalisation selon les besoins (cas d'une université, d'une banque …).

Le serveur : C'est le troisième acteur de notre système, il est l'intermédiaire entre la SAM et le client, il est doté d'une base de donnée, il permet :

- Subvenir aux besoins du site selon le contexte (vente en ligne, recherche, rencontre …)
- L'interaction entre les carte DESFire et l'interface de la SAM pour exécuter une tâche.
- Vérifier auprès de la SAM l'authenticité du client.

3.5.Analyse détaillé sur l'environnement de test de notre système :

Pour tester notre travail nous allons mettre en place notre solution sur un site web déjà existant.

Sur ce point nous allons expliquer brièvement ce que nous avons à faire pour ce site et prendre les points essentiel qui sont en rapport avec notre système.

Sachant que ce site fait la gestion de stock pour des produits électroniques (les LEDs, les portes NOR…etc.) du département d'électronique

Il est devisé en deux espaces distincts, le premier est pour le gestionnaire de stock et le deuxième est pour un simple consultant de produit :

- **Le gestionnaire de stock :**
  - Il a pour tâches de mettre à jour (ajouter un produit).
  - Lecture des produits (voire les produits contenus dans le stock).

Ce dernier pourra alors accéder à deux interfaces différentes qui sont :

- L'interface ajoute_produit.
- L'interface consulter_produit.

1. L'interface ajout produits

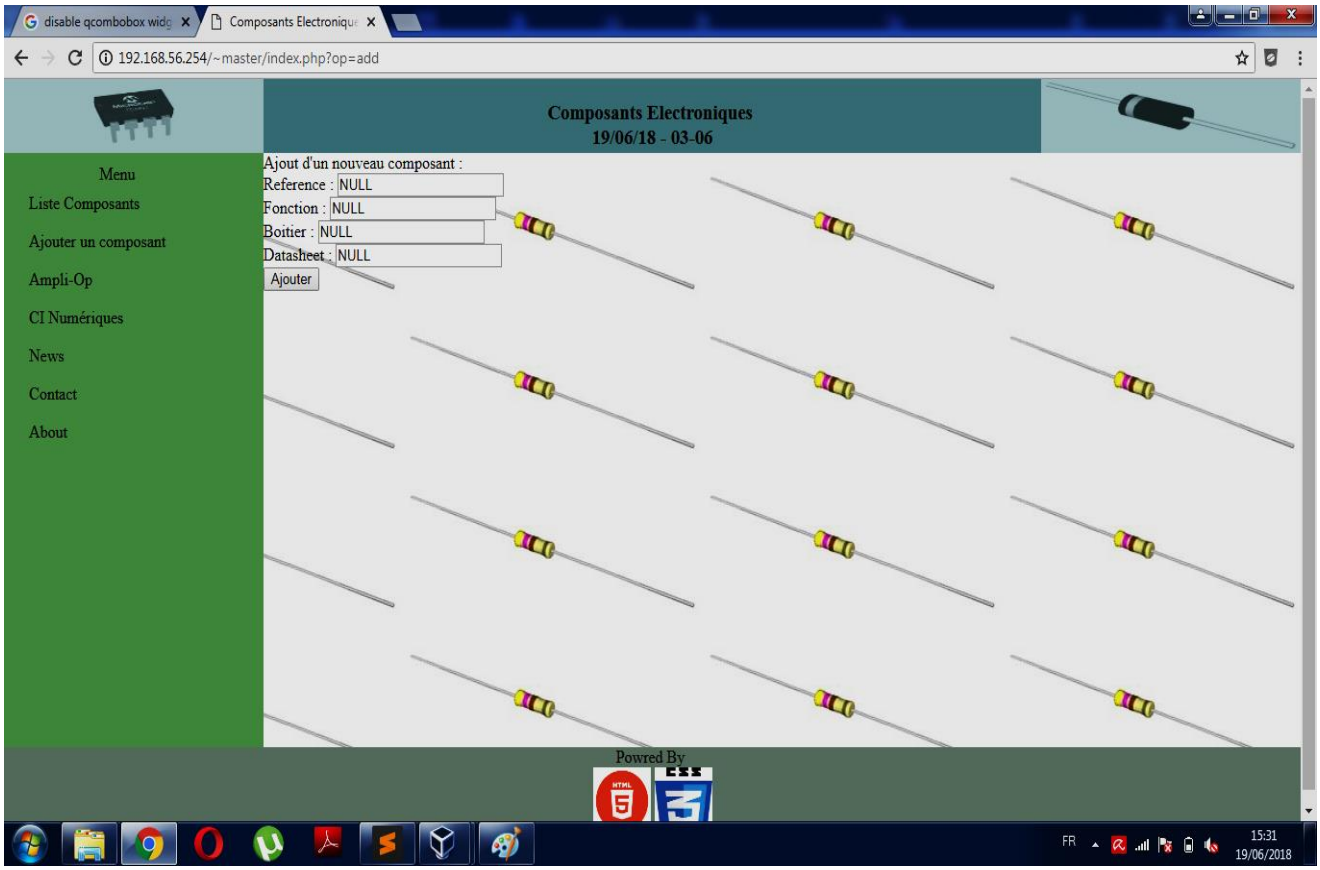
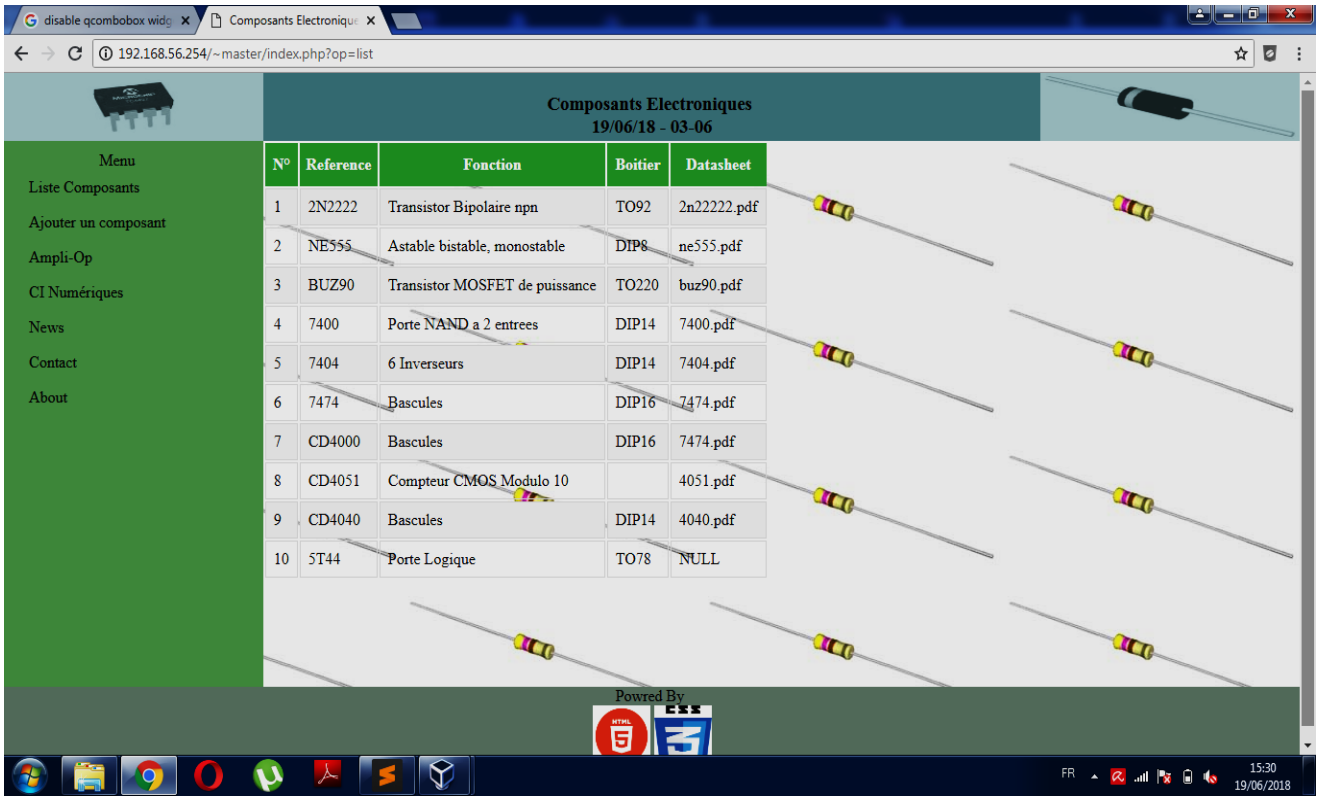


Figure III.3: interface ajout de produit

2. L'interface consulter_produits



N°	Reference	Fonction	Boitier	Datasheet
1	2N2222	Transistor Bipolaire npn	TO92	2n22222.pdf
2	NE555	Astable bistable, monostable	DIP8	ne555.pdf
3	BUZ90	Transistor MOSFET de puissance	TO220	buz90.pdf
4	7400	Porte NAND a 2 entrees	DIP14	7400.pdf
5	7404	6 Inverseurs	DIP14	7404.pdf
6	7474	Bascules	DIP16	7474.pdf
7	CD4000	Bascules	DIP16	7474.pdf
8	CD4051	Compteur CMOS Modulo 10		4051.pdf
9	CD4040	Bascules	DIP14	4040.pdf
10	5T44	Porte Logique	TO78	NULL

Figure III.4 : interface Consulter_produit

- **Simple consultant :**
  - Il peut lire les produits (voire les produits contenu dans le stock).

Il pourra donc accéder à l'interface Consulter_produits.

(Voir)Figure III.4 : interface consulter_produit

Sachant que notre travail est d'implémenter le système que nous allons réaliser (système d'authentification) pour ce site, ainsi que son environnement il faudrait donc faire une interface d'authentification qui sera à la portée des utilisateurs qui possèdent les cartes DESFire qui seront configurées et personnalisées selon les besoins au sein du système de personnalisation là où elle subira les traitements nécessaire.

Pour différentier les privilèges d'accès juste aux interfaces autorisé, c'est les cartes DESFire qui fera la distinction entre les deux lors de l'authentification et vérifiera sa auprès du serveur.

3.6.Les diagrammes de cas d'utilisation :

Les cas d'utilisation décrivent le comportement du système du point de vu de l'utilisateur.

Ils permettent de définir les limites du système et les relations entre le système et son environnement. Un cas d'utilisation est une manière spécifique d'utiliser le système. C'est l'image d'une fonctionnalité en réponse à la stimulation d'un acteur externe.

3.6.1. Diagramme de cas d'utilisation :

Pour bien comprendre notre système, nous allons présenter trois diagrammes globaux pour les différentes parties de notre système ainsi que le système sur le site web sur lequel nous testerons notre système.

- Diagramme de cas d'utilisation pour l'utilisateur.
- Diagramme de cas d'utilisation pour le serveur.
- Diagramme de cas d'utilisation pour la personnalisation des cartes.

Chaque cas d'utilisation sera décrit textuellement, et en fin on va mettre en évidence les différents scénarios nominaux et alternatifs.

3.6.2. Diagramme de cas d'utilisation pour l'utilisateur :

Dans ce qui suit nous allons donner le diagramme de cas d'utilisation global pour la partie utilisateur du système que nous avons nommé **« DESFire »**

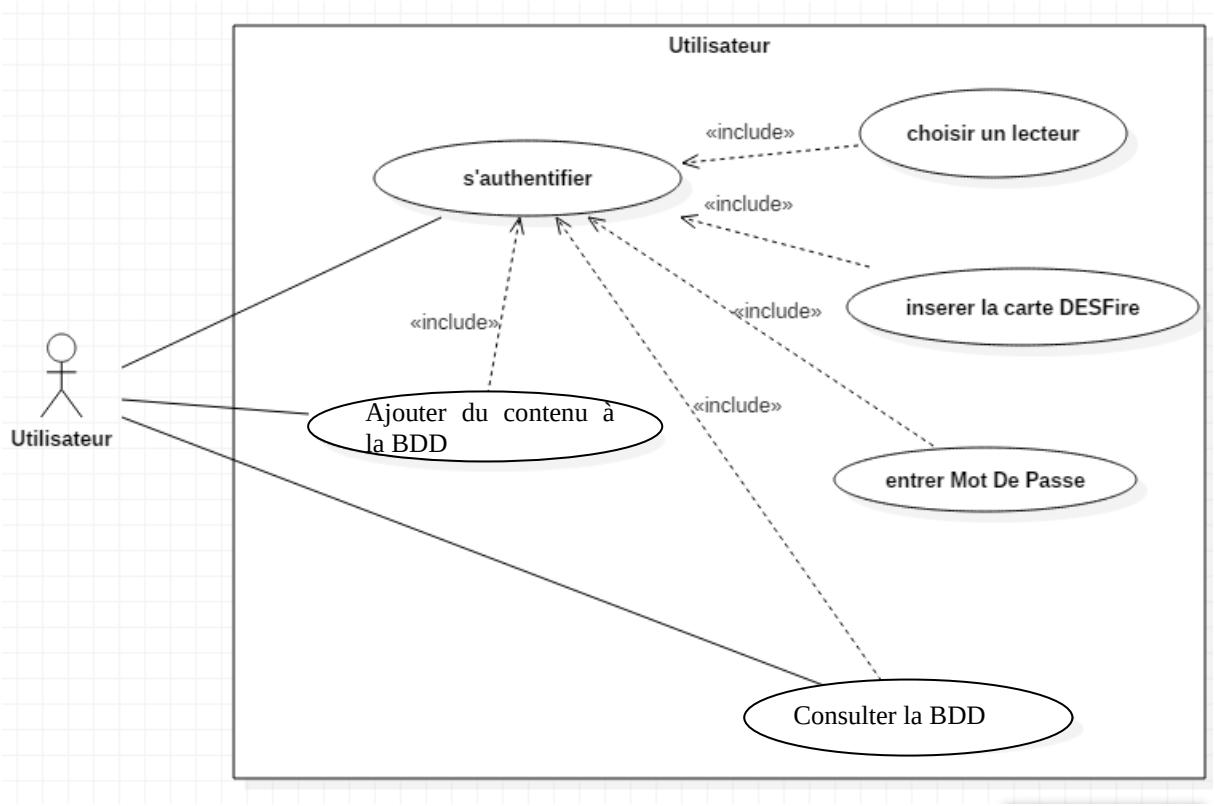


Figure III.5 : Diagramme de cas d'utilisation pour utilisateur

La figure ci-dessus représente le diagramme de cas d'utilisation global pour l'acteur « utilisateur », après qu'il est installé l'exécutable de gestion d'interface dans le navigateur, il doit accéder au site web là où il sera directement rediriger vers la page d'authentification et pour accéder aux différentes fonctionnalités du site à savoir :

1) L'authentification : qui comporte les étapes fondamentales suivantes
   - Insérer la carte (poser la carte sur le lecteur vue que c'est une carte sans contacte)
   - Le serveur lui retourne une page pour saisir le mot de passe
   - L'utilisateur saisit alors le mot de passe
2) Après authentification le serveur le dirige vers l'espace réservé aux utilisateurs du système là où il pourra atteindre les différentes fonctionnalités suivantes :
   - Consulter les produits contenus dans le stock.
   - Ajouter de nouveaux produits s'il est autorisé.

3.6.3. Diagramme de cas d'utilisation pour le serveur :

Dans ce qui suit nous allons donner le diagramme de cas d'utilisation global pour la partie serveur du système que nous avons nommé **« Serveur »**

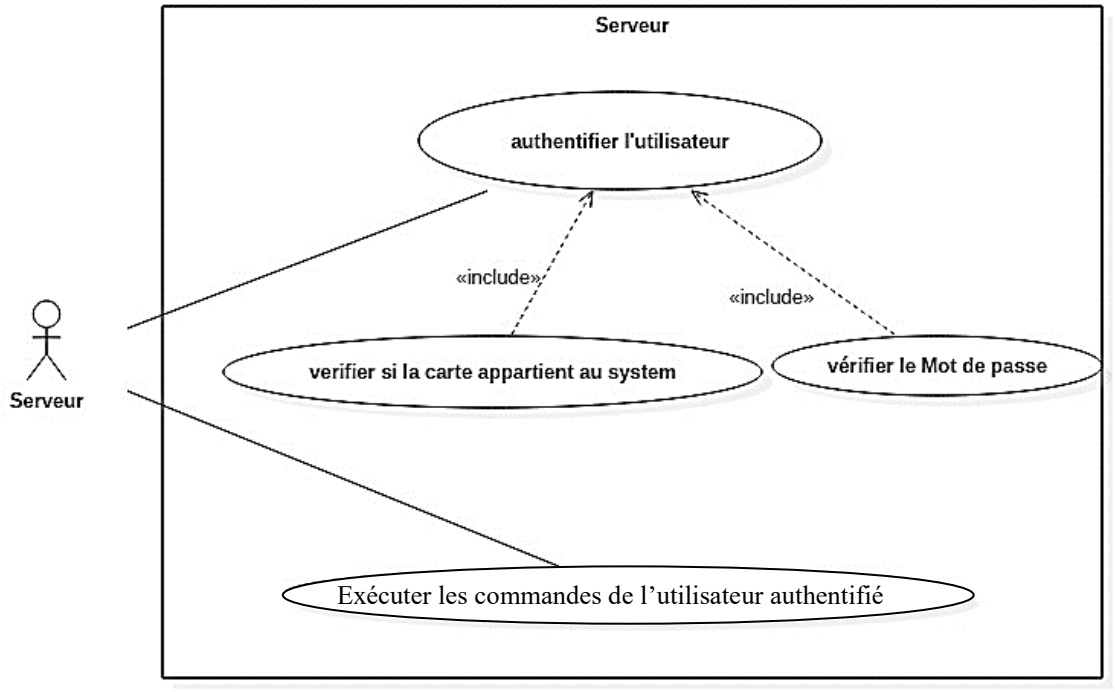


Figure III.6 : Diagramme de cas d'utilisation pour le serveur

La figure ci-dessus représente le diagramme de cas d'utilisation global pour l'acteur **« serveur »**, c'est lui qui est chargé de transmettre et de recevoir les données, les commandes et les requêtes à exécuter. Il fait les tâches importantes qui sont :

1) L'exécution d'une commande :

Exécuter les commandes des utilisateurs dans notre exemple (ajoute et/ou consulter produit)

2) L'authentification d'un utilisateur : qui se présente comme suit
   - L'authentification mutuelle entre l'application souhaitée dans la DESFire et le module gestion d'interface.
   - Lire les données de l'application dans la DESFire.
   - Récupérer le mot de passe de l'utilisateur.
   - Vérifier la validité des données et du mot de passe.
   - Si validé alors l'utilisateur est authentifié, sinon il n'est pas authentifié.

En plus de ces deux tches il faudra que le serveur puisse communiques avec les cartes DESFire en faisant appel à l'interface SAM, nous allons détailler dans le prochain chapitre le fonctionnement de ce processus :

- Envoyer une requête vers l'interface SAM pour récupérer une commande APDU.
- Récupérer la commande APDU.
- Envoyer la commande APDU à la page web.
- Récupérer la réponse APDU.

3.6.4. Diagramme de cas d'utilisation pour la personnalisation :

Dans ce qui suit nous allons donner le diagramme de cas d'utilisation global pour la partie personnalisation du système que nous avons nommé **« Personnalisation »**

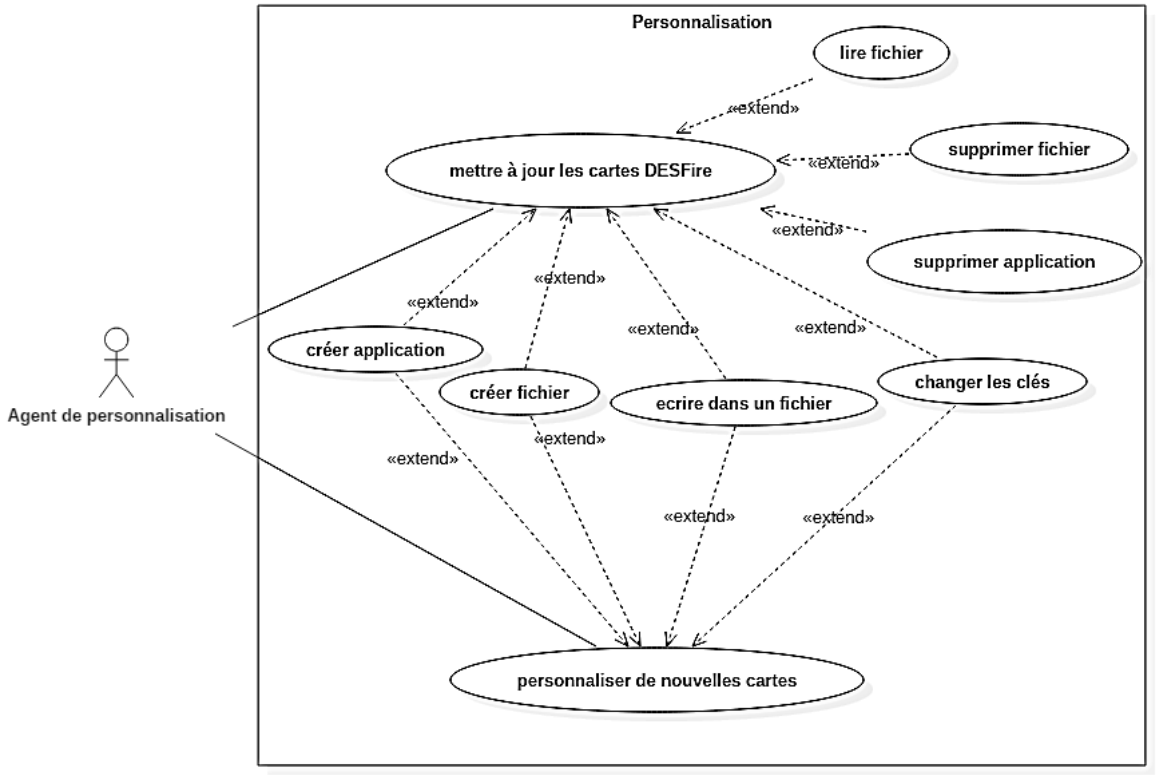


Figure III.7 : Diagramme de cas d'utilisation pour la personnalisation

Comme cité ci-dessus l'agent peut soit :

1) Mettre à jour une carte DESFire pour un utilisateur, il peut alors :

- Lire un fichier, Supprimer un fichier ou une application, ainsi que toutes les fonctions de la personnalisation d'une carte DESFire que nous citons ci-dessous.

2) Personnaliser une nouvelle carte DESFire :
   - Créer un fichier, créer une application, écrire dans un fichier, changer les clés.

Remarque : il peut aussi faire d'autre fonctions à par celles dont nous avons parlé et que nous verrons dans la partie réalisation.

Conclusion :

Au cours de ce chapitre nous avons parlé sur le système que nous allons concevoir et réaliser et comment il peut s'appliquer sur les sites web des développeurs, et cela en l'appliquant sur le site proposé.

Nous l'avons détaillé par l'analyse des besoins par l'ensemble des diagrammes et parler sur les problèmes des sites et des applications de nos jours et comment procéder à leurs sécurisations en utilisant la technologie des cartes à puce, et vu par la suite que notre système offre un environnement standard et facilite la tâches aux intéressés.

Chapitre IV:
Conception

Préambule :

L'objectif de ce chapitre est de fournir une étude conceptuelle suffisamment détaillée de notre travail. Pour cela, nous avons opté pour la modélisation des différents concepts du domaine d'application, afin de faciliter la compréhension, et la conception de notre système.

Au cours de ce chapitre on présentera la conception de notre système sous différents angles en faisant :

- La modélisation des cas d'utilisation vu dans le chapitre Analyse.
- détailler la conception de chaque module de notre système en utilisant des diagrammes et des schémas explicatif.

1. Définition de la conception :

La conception est l'étape qui suit l'analyse. C'est une étape très essentielle car elle permet de comprendre le fonctionnement du système conçu et elle consiste à modéliser et à détailler tous les éléments de modélisation issus de la phase analyse.

Après avoir décrit textuellement les différent cas d'utilisation de notre système nous allons les présenter formellement à l'aide des diagrammes de séquences.

2. Les diagrammes de séquence :

Définition :

Le diagramme de séquence représente des échanges de messages entre objets. Il permet de représenter un processus de façon simplifiée, en se centrant sur les échanges entre les acteurs ou entre acteurs et le système d'information.

Ils illustrent la dynamique d'enchainement des traitements d'une application effectuée par le système. Ces traitements sont donnés dans

Les objets utilisés sont :

❖ **Les objets d'interfaces :**

Un objet d'interface représente l'interface entre l'acteur et le système tels que les pages web ou les écrans de saisie.

L'icône : 

❖ **Les objets entité** :

Sont des objets décrit dans un cas d'utilisation et qui se trouvent dans d'autres cas d'utilisation tels que l'utilisateur.

L'icône : 

❖ **Les objets de contrôle** :

Représentent les activités des processus du système, ils dirigent les activités des entités et interfaces.

L'icône : 

Dans la description des cas d'utilisation du chapitre précédant nous avons pu identifier quelques scénarios de cas d'utilisation. Dans ce qui suit nous allons traduire quelques-uns en diagrammes de séquences et voir d'autre.

2.1. Diagramme de séquence pour le cas d'utilisation « Authentification »

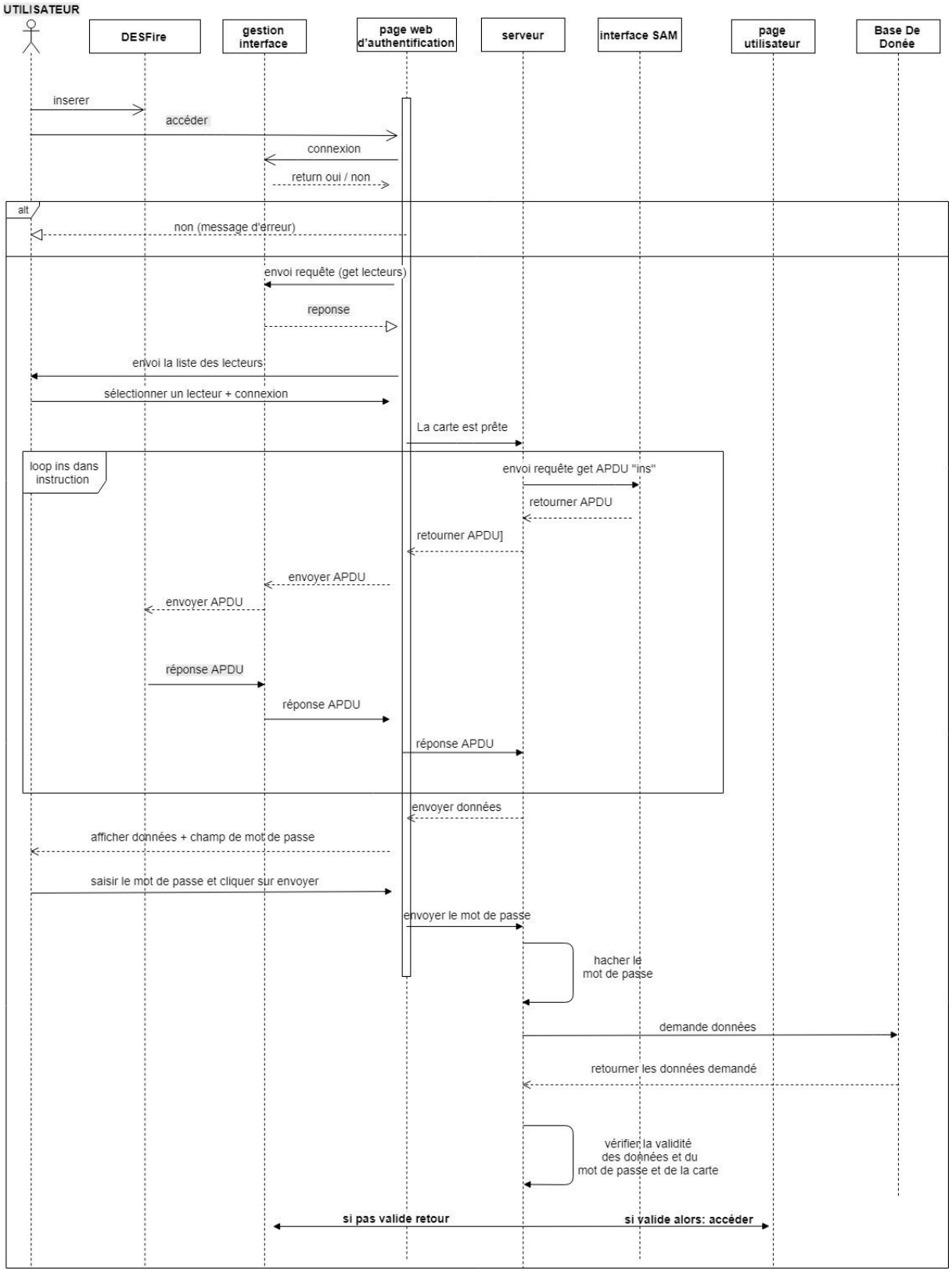


Figure IV.1 : diagramme de séquence pour le cas d'utilisation « s'authentifier »

Dans ce diagramme nous expliquant comment la phase d'authentification d'un utilisateur ce passe avec différentes interaction entre : l'utilisateur, la carte DesFire, gestion d'interface, la page web, le serveur et l'interface SAM comme suit :

1. L'utilisateur insert la carte DESFire.
2. Puis il accède à la page web.
3. Une connexion entre la page web et gestion d'interface.
4. Si la connexion n'est pas établie alors il envoie un message d'erreur.
5. Sinon la connexion est établie alors il passe aux instructions.
6. La page web envoie la requête (get lecteur) pour voir les lecteurs disponible.
7. La gestion d'interface alors envoie une réponse.
8. La page web affiche alors à son tour les lecteurs disponible pour que l'utilisateur puisse en choisir un.
9. L'utilisateur choisit un lecteur et envoie une requête de connexion vers la carte.
10. La page web alors confirme la connexion à la carte.
11. Le serveur envoi une requête pour récupérer la commande APDU de l'instruction voulu « ins » ex : get version, authenticate …ect
12. L'APDU est envoyé jusqu'à arriver à la DesFire.
13. La DesFire exécute la commande APDU et envoie la réponse APDU au serveur
14. Après une série de tests l'utilisateur sera authentifier ou non.

La « loop » représente une boucle qui permet d'exécuter toutes les instructions dons nous avons besoin pour récupérer les données nécessaires présentes dans la carte :

- Getcard version.
- Select application [12,12 ,12]
- Authenticate clé (0)
- Internalauthenticate
- Externalauthenticate
- Read data (file 0…4)

On verra la fonction des instructions dans la partie conception de l'interface SAM

2.2.Diagramme de séquence pour le cas d'utilisation « Ajouter un produit »

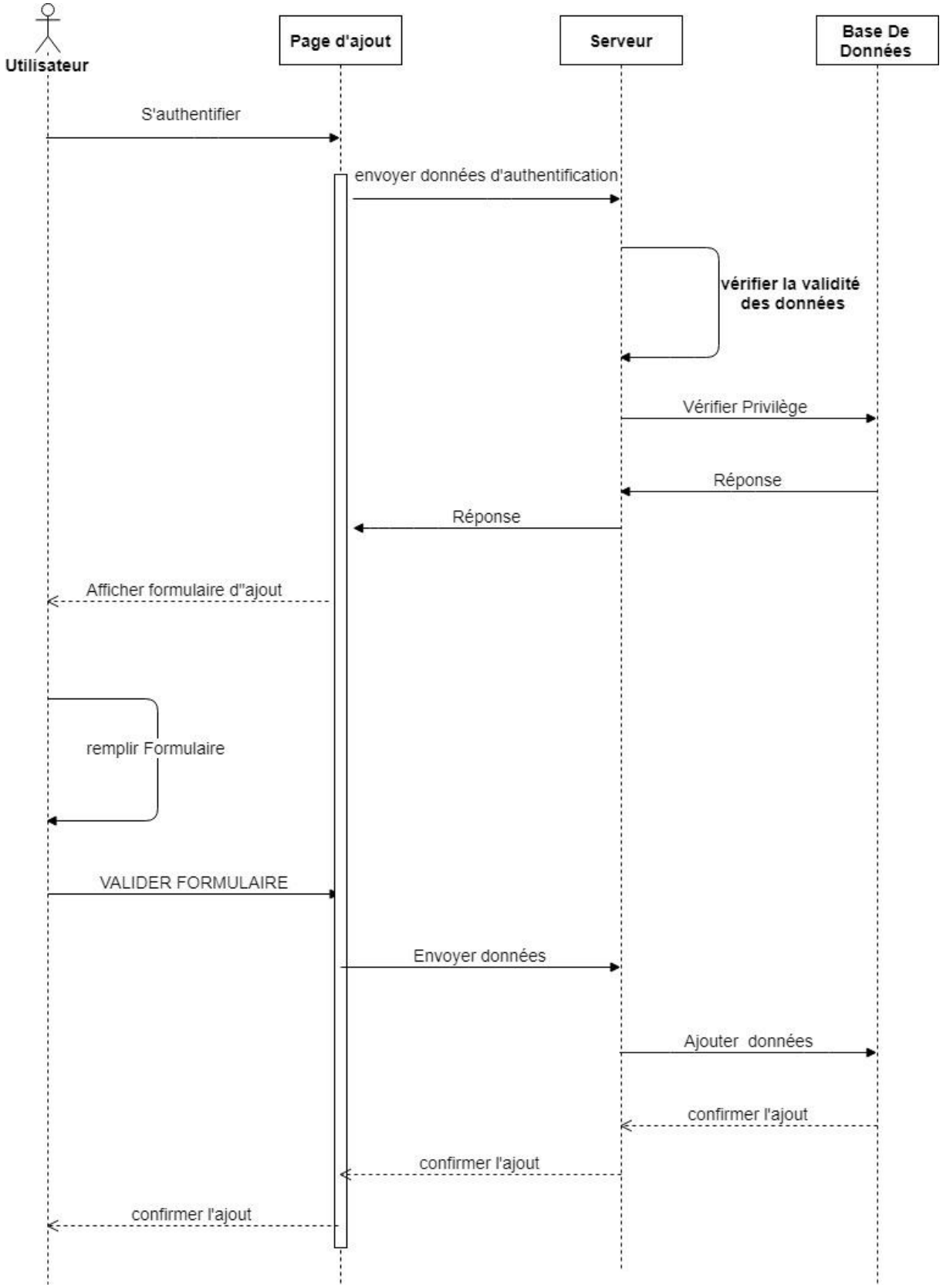


Figure IV.2 : Diagramme de séquence pour le cas d'utilisation « Ajouter un produit »

2.3.Diagramme de séquence pour le cas d'utilisation « Mettre à jour une carte »

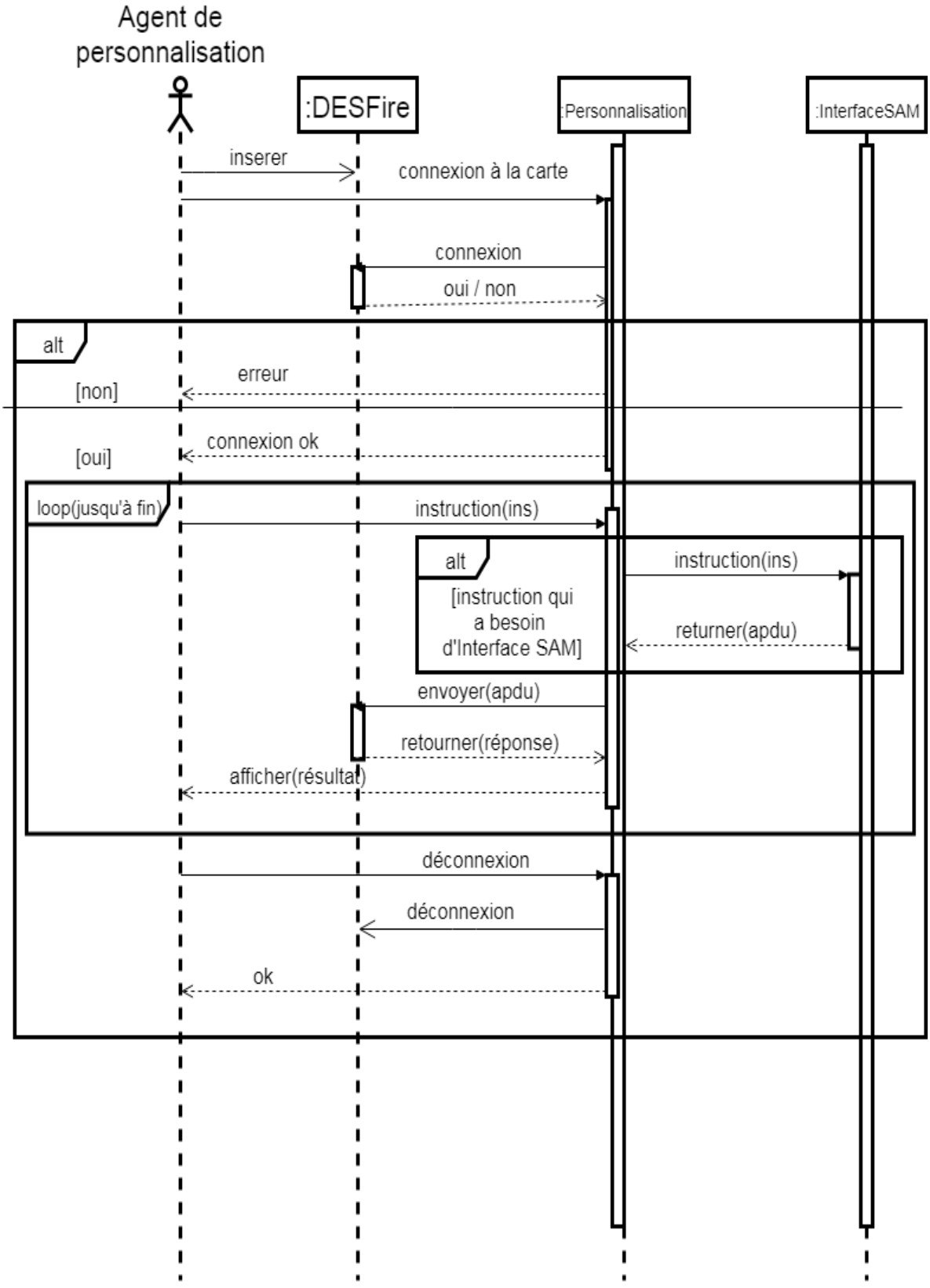


Figure IV.3 : Diagramme de séquence pour le cas d'utilisation « Mettre à jour une carte »

2.4.Diagramme de séquence pour le cas d'utilisation « Personnaliser une nouvelle carte »

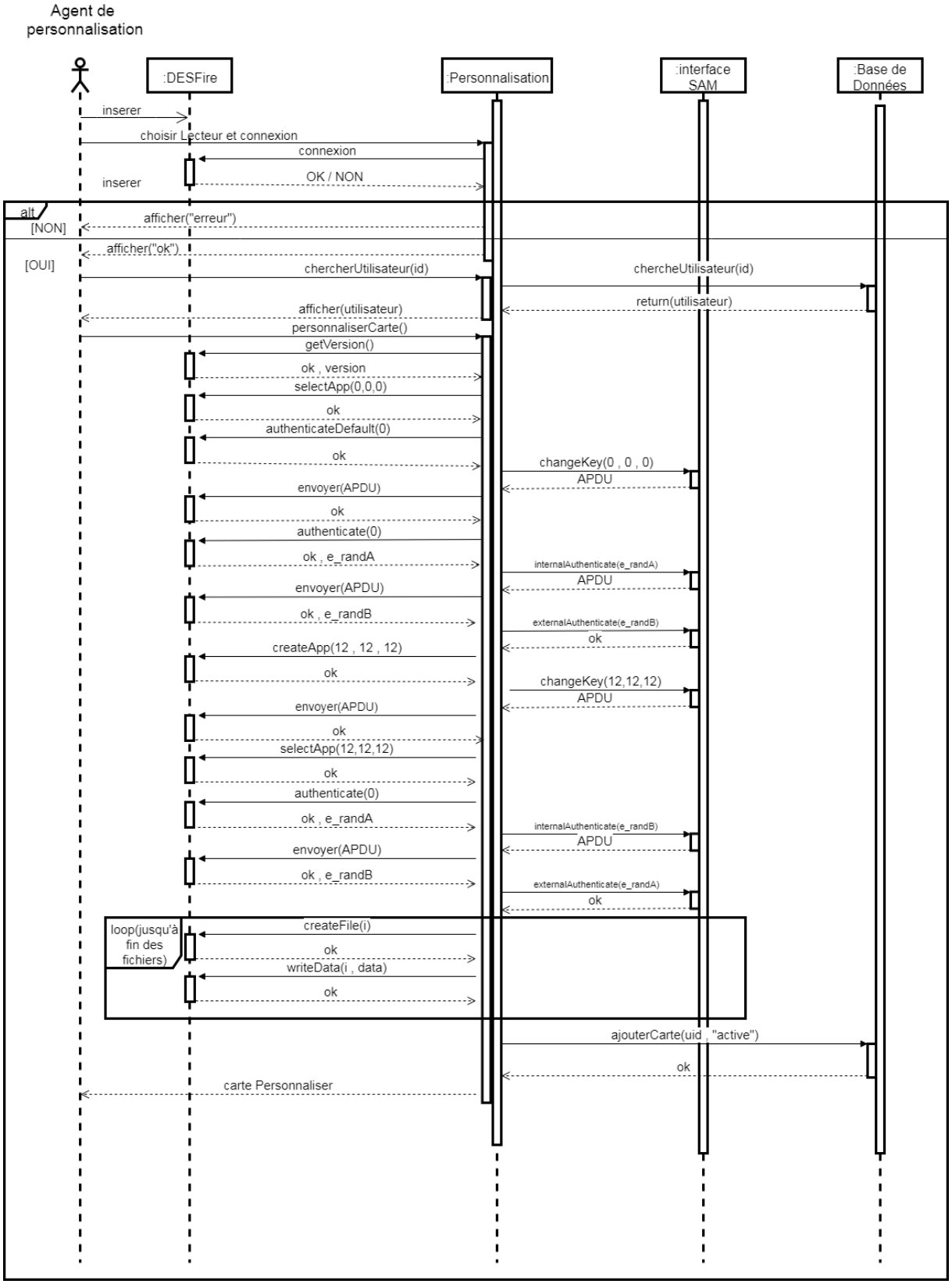


Figure IV.4 : Diagramme de séquence pour le cas d'utilisation « Personnaliser une carte»

3. Conception des différents modules de notre application :

3.1.Conception du module «gestion d'interface» :

Problématique :

Une page web (un code javascript) ne peut pas accéder à la machine de client pour des raisons de sécurité, donc on ne peut pas accéder à la spécification **PS/SC [..]** du système d'exploitation.

Solution :

Pour résoudre ce problème nous allons réaliser une application qu'on appellera « **gestion d'interface** », qui sera installée dans la machine du client, et qui permettra de communiquer avec le code javascript de la page web.

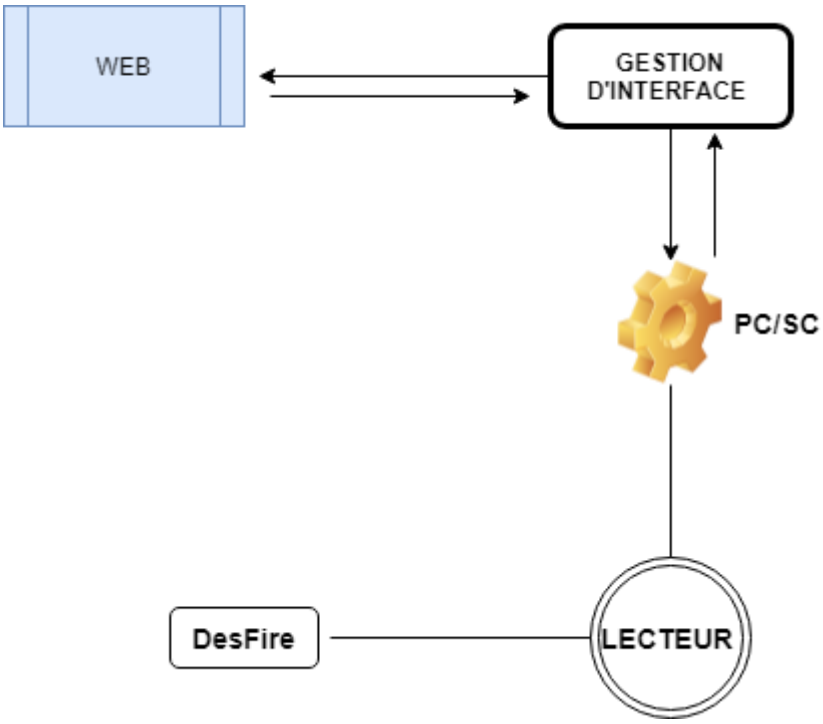


Figure IV.5 : le rôle de gestion d'interface

Résultats :

- Une solution portable qui fonctionne sur la plupart des systèmes d'exploitation (Linux, Windows, etc.).
- Tous les navigateurs pourront l'utiliser, sans ajouter un plugin.
- L'application ce lance automatiquement à chaque démarrage du système.

Diagramme de classe du module gestion d'interface :

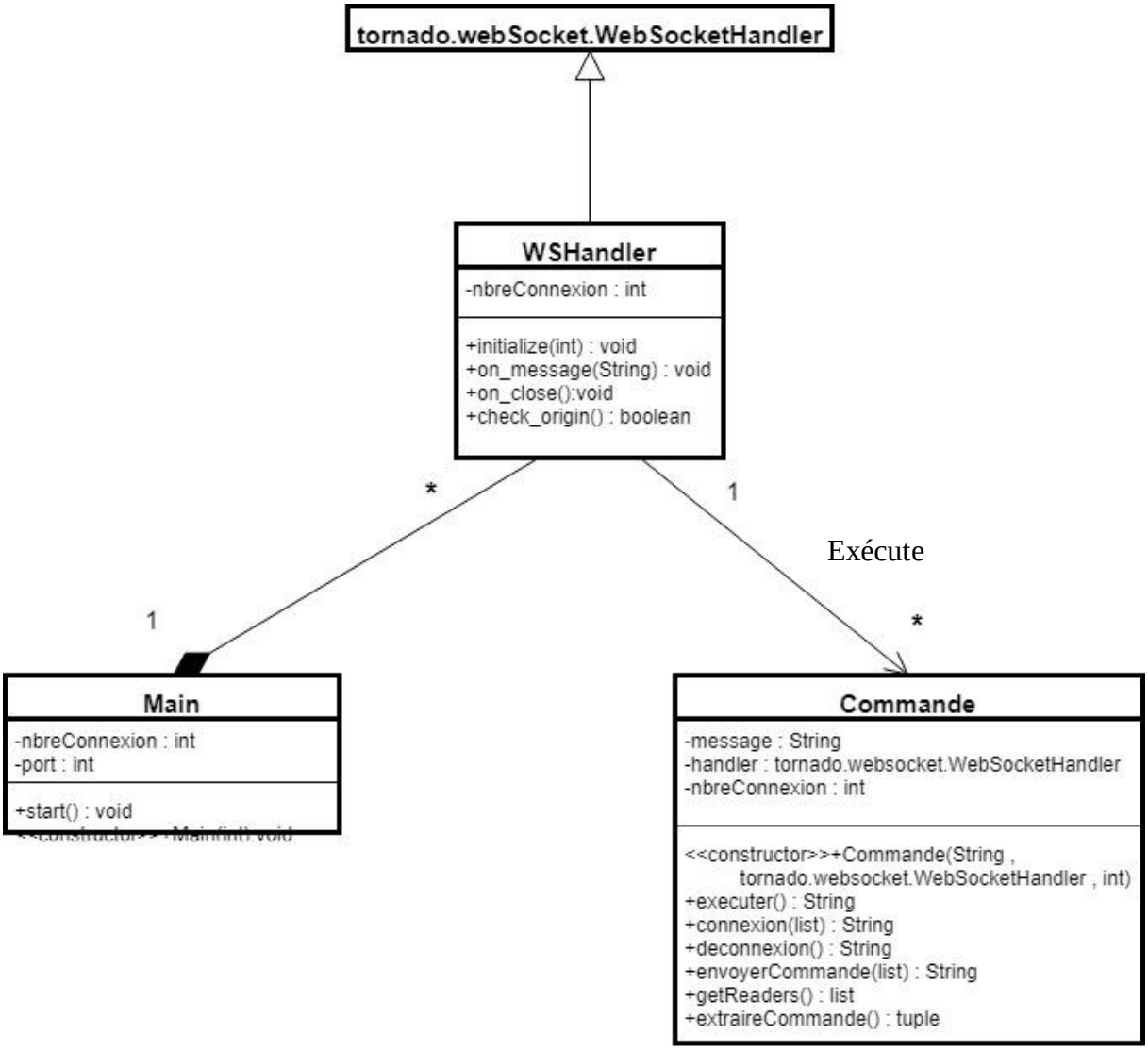


Figure IV.6 : Diagramme de classe de gestion d'interface

Explications sur le diagramme de classe :

Main : c'est la classe qui est le point d'entrée de notre application, elle permet d'initialiser les données, et de lancer le programme.

WSHandler : c'est une classe qui hérite de tornado.websocket.WebSocketHandler, qui permet de gérer les connexions des utilisateurs, et l'échange des données.

Commande : est une classe qui permet de traiter les commandes envoyées par la page web, les instructions possibles sont connexion à une carte, récupérer les lecteurs disponibles dans la machine de l'utilisateur, déconnexion d'une carte et envoyer les instructions à exécuter par la carte à puce.

Remarque :

- Le programme sera réalisé avec python, tornado et pyscard pour réaliser une solution portable
- L'application crée un serveur web qui exécute les requêtes webSocket, donc une solution indépendante du navigateur web, la page web utilise du code javascript pour l'échange des requêtes.
- Le choix de webSocket est dû à l'aspect bidirectionnel des communications entre le serveur et le client dans cette technologie.

3.2.Conception du package DESFiretools :

Les cartes DESFires utilise un ensemble d'instructions **APDU** structurées qui sont définit dans la documentation de la carte.

Afin d'éviter le recoure à cette dernière nous allons réaliser un package qui permettra de faire abstraction de ces instructions.

Rôles :

- Récupérer les instructions DesFire.
- Exécuter les instructions DesFire.

Objectifs :

Programmation orienté objet, pour être évolutif, simple d'usage et facile à mettre à jour.

Séparation entre la récupération d'une commande et l'exécution de cette dernière.

Diagramme de classe de DESFiretools :

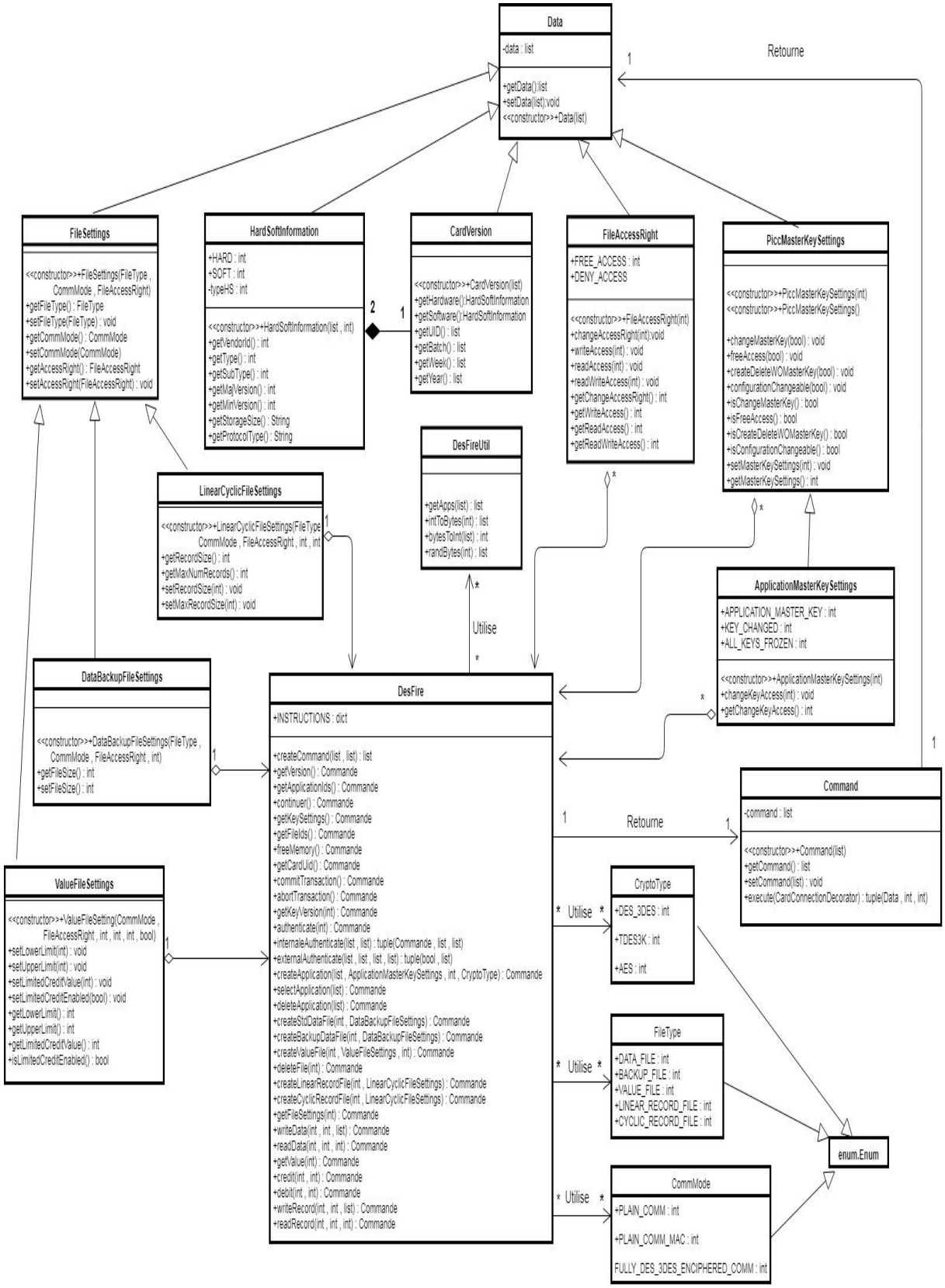


Figure IV.7 : Diagramme de classe de DESFiretools

Explications :

- DesFire : c'est une classe qui permet de retourner les commandes APDU, qui sera exécuté par les Cartes DesFires, elle retourne un objet de type Commande, elle permet aussi d'authentifier une application DesFire.
- Commande : c'est une classe qui permet d'encapsuler une commande APDU, la classe permet de récupérer la commande sous forme d'une liste, ou la possibilité d'envoyer l'APDU à une carte connectée et de retourner la réponse de la carte sous format d'une tuple, le tuple contient un objet Data et l'état de la réponse.
- Data : une classe qui contient une liste de Bytes.
- FileAccessRight : une classe qui hérite de Data qui facilite la création et la récupération des droits d'accès aux fichiers.
- PiccMasterKeySettings : une classe qui hérite de Data, qui permet de facilité la création et la récupération des configurationsde la clé principale de la carte DesFire.
- ApplicationMasterKeySettings : une classe qui hérite de PiccMasterKeySettings, qui permet de facilité la création et la récupération des configurations de la clé principale des applications contenus dans la DesFire.
- HardSoftInformation : est une classe qui hérite de Data, qui permet de récupérer les informations Hardware et software des cartes.
- CardVersion : est une classe qui hérite de Data : qui permet de récupérer les informations utile sur la carte, parmi elles : les informations sur le hardware, software, l'unique Identifier, …etc.
- FileSettings : est une classe qui hérite de Data, elle contient les informationssur la configuration des Fichiers.
- ValueFileSettings : est une classe qui hérite de FileSettings, elle permet de créer et de récupérer les configurations des fichiers de type value file.
- DataBackupFileSettings : est une classe qui hérite de FileSettings, elle permet de créer et de récupérer les configurations des fichiers de type Standard Data File, Backup Data File.
- LinearCyclicFileSettings : est une classe qui hérite de FileSettings, elle permet de créer et de récupérer les configurations des fichiers de type Linear Record Files et Cyclic Record Files.
- DesFireUtil : est une classe qui contient des méthodes qui facilite la manipulation des données.

- CryptoType : est une classe qui hérite d'Enum « énumération », qui permet de récupérer les numéros d'algorithmes de cryptage.
- FileType : est une classe qui hérite d'Enum, qui permet de récupérer les types de Fichiers.
- CommMode : est une classe qui hérite d'Enum, qui permet de récupérer le mode de communication.

3.3.Conception de la SAM (Secure Accès Module) :

Le rôle de la SAM est la génération des clés pour les cartes des utilisateurs« DESFire », et les envoyer à l'interface SAM.

Différentes instructions :

On distingue deux types d'instructions :

1) Instructions de configurations :

Elles permettent de configurer la SAM, pour pouvoir l'utiliser.

Les instructions de configuration de la SAM sont :

- Changer la clé de chiffrement « clé principale ».
- Changer le paramètre iv « Initial Value ».
- Changer le vide restant pour atteindre 16 octets « padding ».

2) Instruction de récupération de la clé :

Elle permet de calculer et d'envoyer la clé demandée en fonction de l'UID de la carte, l'identifient de l'application et de numéro de la clé, l'algorithme que nous allons implémenter sera définit ci-dessous.

Algorithme de génération des clés :

Pour générer les clés automatiquement, nous proposant l'algorithme suivant, qui permet de calculer chaque clé en fonction.

- Unique Identifier « UID » : qu'est un tableau de sept octets unique donné par le fabriquant de la carte.
- Application Identifier « AID »: qu'est un tableau de 3 octets, donné lors de la création de la carte.
- Numéro de la clé : entre zéro et treize qui correspond au numéro de la clé dans l'application.

Algorithme de la fonction qui génère la clé :

1- Récupérer l'UID de la carte.
2- Récupérer l'AID de l'application.
3- Récupérer le numéro de la clé.
4- Concaténer l'UID et l'AID et le numéro de la clé.
5- Ajouter le padding pour compléter les éléments restant, pour atteindre 16 octets, et soit le « tab » le tableau résultant.
6- Crypter « tab » avec l'algorithme de chiffrement 3DES en mode « ede », et « cbc », avec la clé de chiffrement « clé principale ».
7- Le résultat obtenu est la clé.

Remarque :

- La SAM génère les clés automatiquement, donc on n'a pas besoin de les sauvegarder.
- Le choix du padding, la clé de chiffrement et le paramètre « iv », sont décisif pour la génération des clés, chaque modification affecte les clés générées.
- Pour un choix de padding, clé de chiffrement et de paramètre « iv », chaque clé générée est unique.

3.4.Conception du Serveur :

Le serveur a pour rôle d'afficher les produits, et d'ajouter un produit, nous allons ajouter une fonction d'authentification des utilisateurs appartenant au système.

Ainsi attribué à chaque utilisateur des privilèges, nous distinguons deux types d'utilisateurs.

- Utilisateur 1 : qui a les privilèges d'ajout et la consultation.
- Utilisateur 2 : qui a le privilège de consultation seulement.
  Toute autre personne qui n'est pas authentifié n'a pas accès à ce système.

3.5.Conception de l'interface SAM :

Rôles :

- Générer la commande APDU à exécuter par la DESFire.
- Authentification mutuelle avec les DESFire.

Objectifs :

- Un système multiplateforme: il s'exécute sur la plupart des systèmes d'exploitation (Windows, Linux, …etc.).
- Chaque langage Serveur pourra communiquer avec.
- Gérer la file d'attente pour l'accès à la SAM, avec le principe de « FIFO » premier arrivé premier servi.

Solution pour les objectifs :

Pour atteindre les objectifs que nous avons fixés nous proposant les solutions suivantes :

- Nous utilisons python pour réaliser l'interface SAM, car python est un langage multiplateforme.
- Pour permettre à n'importe quel langage d'utiliser l'interface SAM, nous créerons un serveur web Restful, pour permettre aux systèmes d'accéder aux ressources « commande apdu » via des requêtes http.
- Nous utiliserons tornado comme serveur web.

Diagramme de classe de l'interface SAM :

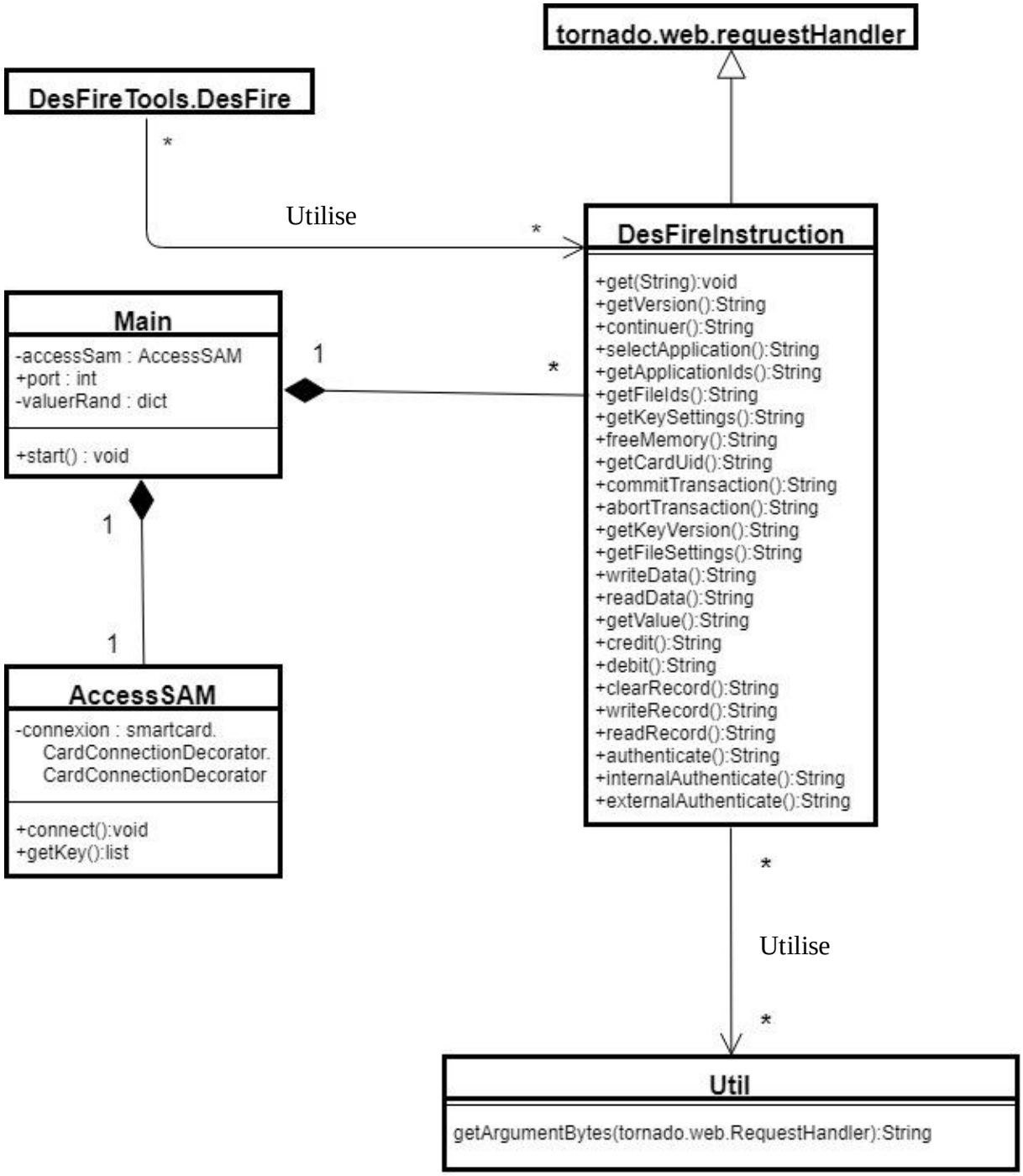


Figure IV.8. Diagramme de classe de l'interface SAM

Explication :

- la classe Main :
  C'est la classe principale de notre application, lors de démarrage, elle initialise le port, et elle crée une instance d'une application tornado.
- La classe Util :

Elle fournit une méthode qui permet de découper les paramètres reçus comme chaine de caractères, et les faire sortir comme une liste de bytes.

- Access SAM :
  Elle permet de communiquer avec la SAM, elle a deux méthodes :
  - Connect : pour gérer la connexion à la SAM.
  - getKey : pour récupérer la clé souhaitée.
- La classe DesFireInstruction :
  C'est la classe qui gère les requêtes, elle permet de traiter la requête et de retourner la réponse sous format JSON, qui contient l'apdu de l'instruction demandée et l'erreur si elle existe.
- Pour traiter les requêtes nous utiliserons DesFireTools.

Remarque :

Pour ce qui concerne l'accès à l'interface SAM, nous supposant qu'il y a un pare-feu qui bloque l'accès aux entités qui n'ont pas le droit d'accès.

3.6.Conception de la partie personnalisation :

Rôle :

L'application permet à l'agent de personnaliser une nouvelle carte, au de faire des mises à jour pour des cartes déjà personnalisées.

Objectif :

- Application multiplateforme.
- Utilisation de l'interface SAM pour la récupération des commandes qui nécessite la connaissance des clés (ex : authentification, changer la clé, …)

Hypothèse :

Nous supposant que l'accès à cette application est restreint à l'agent de la personnalisation, après un contrôle d'accès à la salle de personnalisation.

La Mise à jour d'une carte :

L'étape de la mise à jour, permet à l'agent de personnalisation de supprimer, ajouter ou modifier une carte DesFire déjà personnalisée.

La Personnalisation d'une nouvelle carte :

C'est une étape qui dépend de la base de données, pour notre cas nous allons réaliser cette étape en se basons sur le site web proposé.

La table utilisateur :

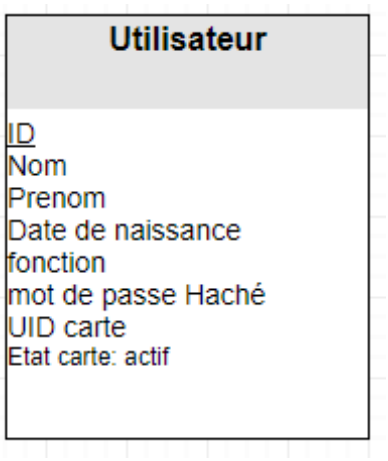


Figure IV.9. La table utilisateur

Utilisateur (id, nom, prénom, date de naissance, fonction, mot de passe haché, UID carte, Etat carte).

Les cartes DesFire :

Pour les cartes des utilisateur nous allons créer une application avec AID = [12 , 12 , 12], cette application contiendra les fichiers suivants :

Id du fichier	Type de fichier	Données du fichier
0	Standard data file	ID
1	Standard data file	Nome
2	Standard data file	Prénom
3	Linear Record file	Date de naissance
4	Standard data file	MDP de l'utilisateur haché

Figure IV.10 : Les fichiers de l'application [12, 12, 12]

Le tableau précédent montre les fichiers contenu dans l'application [12, 12, 12]

Conclusion :

Dans ce chapitre, nous nous sommes concentrés sur l'aspect analytique conceptuel de notre système. Nous avons utilisé un langage de conception qui est UML, on peut dire que c'est l'étape la plus sensible car toute la réalisation repose sur cette dernière.

Comme nous avons montré les différents modules qui composent notre système et détaillé le fonctionnement de chacun d'eux.

Chapitre V :
Réalisation

1. Préambule :

Après avoir présenté les détaille de l'étape analyse et conception du système dans le chapitre précédant, nous passant maintenant à la phase réalisation qui consiste à implémenter les différentes fonctionnalités de notre système.

Dans ce chapitre nous commencerons tout d'abord à la description de notre environnement de travail, puis en deuxième partie nous présenterons les multiples fonctionnalités qu'offres le système à travers différentes interfaces.

2. Description des outils de développement :

L'environnement de travail est le suivant :

2.1. Les langages utilisés :

- **JAVA pour JavaCard :**

La java Card Technologie permettant de faire fonctionner des applications écrites en langage Java pour :

☞ carte à puce

☞ Autres périphériques à mémoire limitée

La technologie Java Card définit une plate-forme sécurisée pour cartes à puce, portable et multi-application qui incorpore beaucoup des avantages du langage Java.

- **Python 3.5 :**

Python est un langage puissant, à la fois facile à apprendre et riche en possibilités. Dès l'instant où vous l'installez sur votre ordinateur, vous disposez de nombreuses fonctionnalités intégrées au langage que nous allons découvrir tout au long de ce livre.

Il est, en outre, très facile d'étendre les fonctionnalités existantes, comme nous allons le voir. Ainsi, il existe ce qu'on appelle des **bibliothèques** qui aident le développeur à travailler sur des projets particuliers. Plusieurs bibliothèques peuvent ainsi être installées pour, par exemple, développer des interfaces graphiques en Python.

Concrètement, voilà ce qu'on peut faire avec Python :

- de petits programmes très simples, appelés **scripts**, chargés d'une mission très précise sur votre ordinateur ;
- des programmes complets, comme des jeux, des suites bureautiques, des logiciels multimédias, des clients de messagerie…
- des projets très complexes, comme des progiciels (ensemble de plusieurs logiciels pouvant fonctionner ensemble, principalement utilisés dans le monde professionnel).

- **PHP7 :**

Le PHP est un langage informatique utilisé sur l'internet. Le terme PHP est un acronyme récursif de "*PHP: HypertextPreprocessor*".

Ce langage est principalement utilisé pour produire un site web dynamique. Il est courant que ce langage soit associé à une base de données, tel que MySQL.
Exécuté du côté serveur (l'endroit où est hébergé le site) il n'y a pas besoin aux visiteurs d'avoir des logiciels ou plugins particulier. Néanmoins, les webmasters qui souhaitent développer un site en PHP doivent s'assurer que l'hébergeur prend en compte ce langage.

Lorsqu'une page PHP est exécutée par le serveur, alors celui-ci renvois généralement au client (aux visiteurs du site) une page web qui peut contenir du HTML,
XHTML, CSS, JavaScript

- **JavaScript**
- **Html5**
- **Css 3**
- **Json**

2.2. Les bibliothèques utilisées :

- **PyQt5**
- **Tornado**
- **Pyscard**
- **WebSocket**
- **Ajax**

2.3. Les outils utilisés :

- **JCID**
- **PyApduTool**

- **SublimText**
- **Lamp**
- **Ubuntu 16.04**
- **phpmyAdmin**
- **Google Chrome**
- **Oracle VM**
- **IDLE (python 3.5)**

3. La démonstration des différentes interfaces de notre Système :

Notre système se compose de deux parties indépendantes l'une de l'autre, partageant une même base de données qui

3.1. Partie utilisateur

L'utilisateur accède à la page principale du site, ou il sera redirigé à la page d'authentification

Le résultat si l'application « gestion d'interface » n'est pas installée

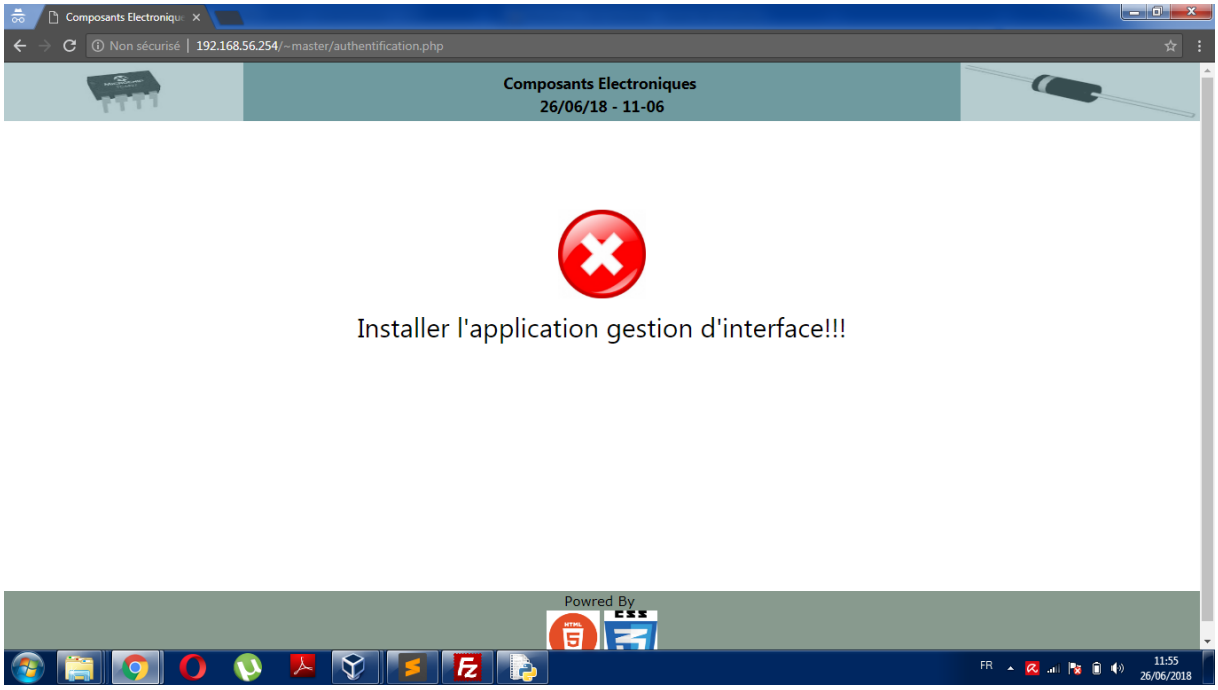


Figure V.1 Résultat si l'application « gestion d'interface » n'est pas installé

Le résultat si l'application « gestion d'interface » est installé :

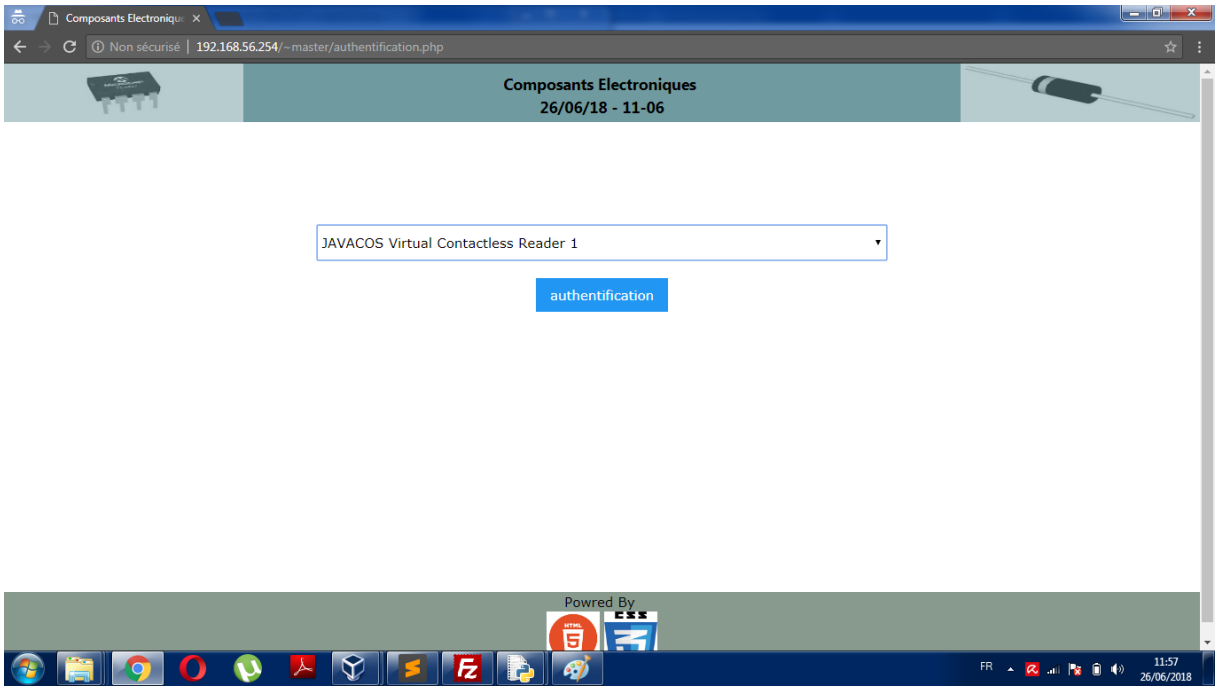


Figure V.2 Interface choisir un lecteur

L'utilisateur aura la possibilité de choisir un des lecteurs disponible.

Ou bien si y a un problème technique dans la carte ou dans le lecteur on aura le résultat suivant :

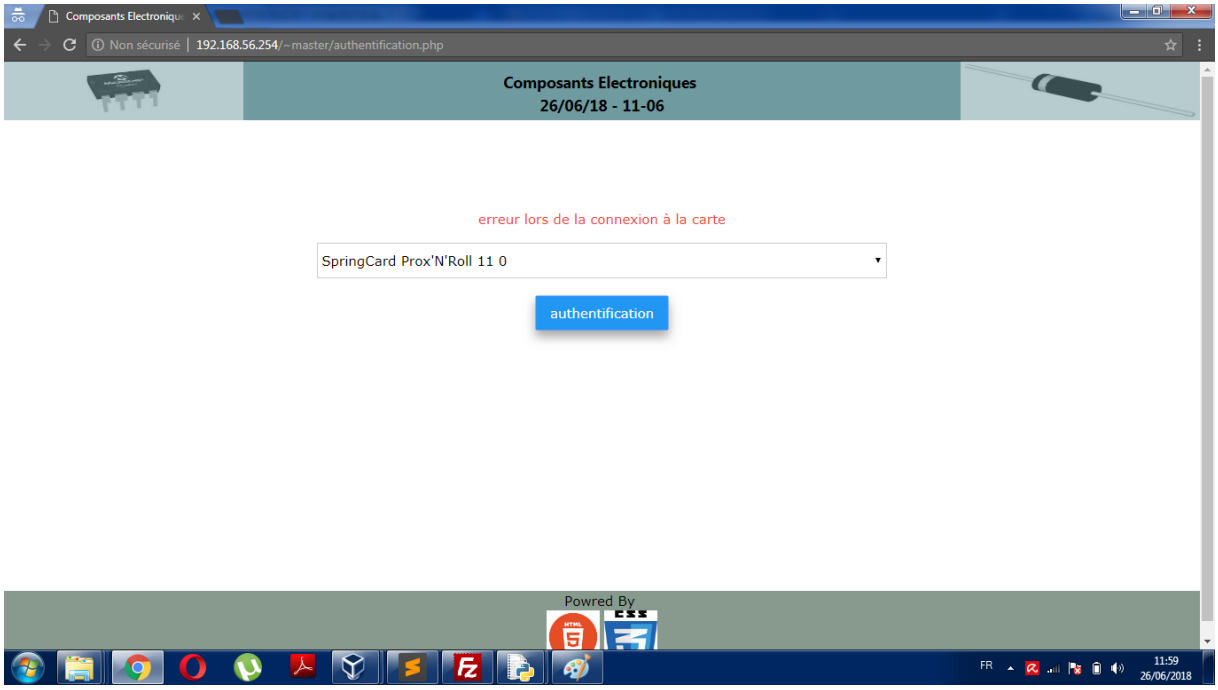


Figure V.3 erreur de connexion à la carte

Une fois la connexion établie avec la carte, le système fait des traitements et affiche les résultats suivant :

Affichage des données de la carte :

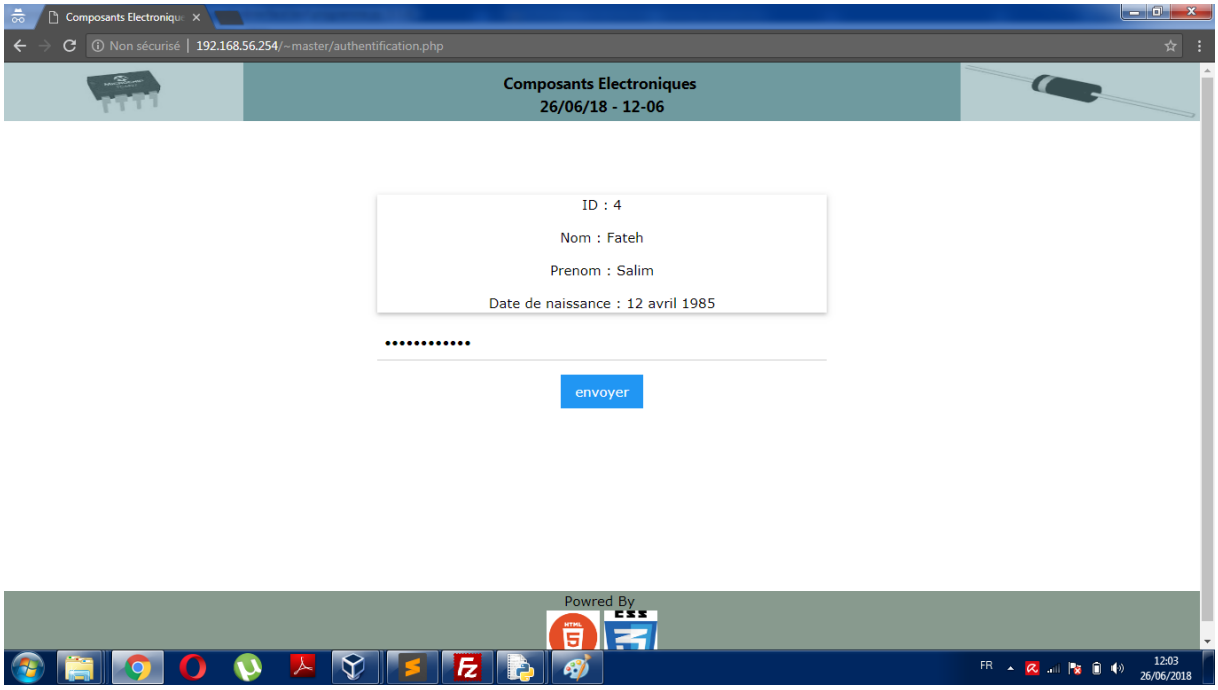


Figure V.4 Interface saisir mot de passe

L'utilisateur saisit le mot de passe.

En cas d'erreur de mot de passe on affiche :

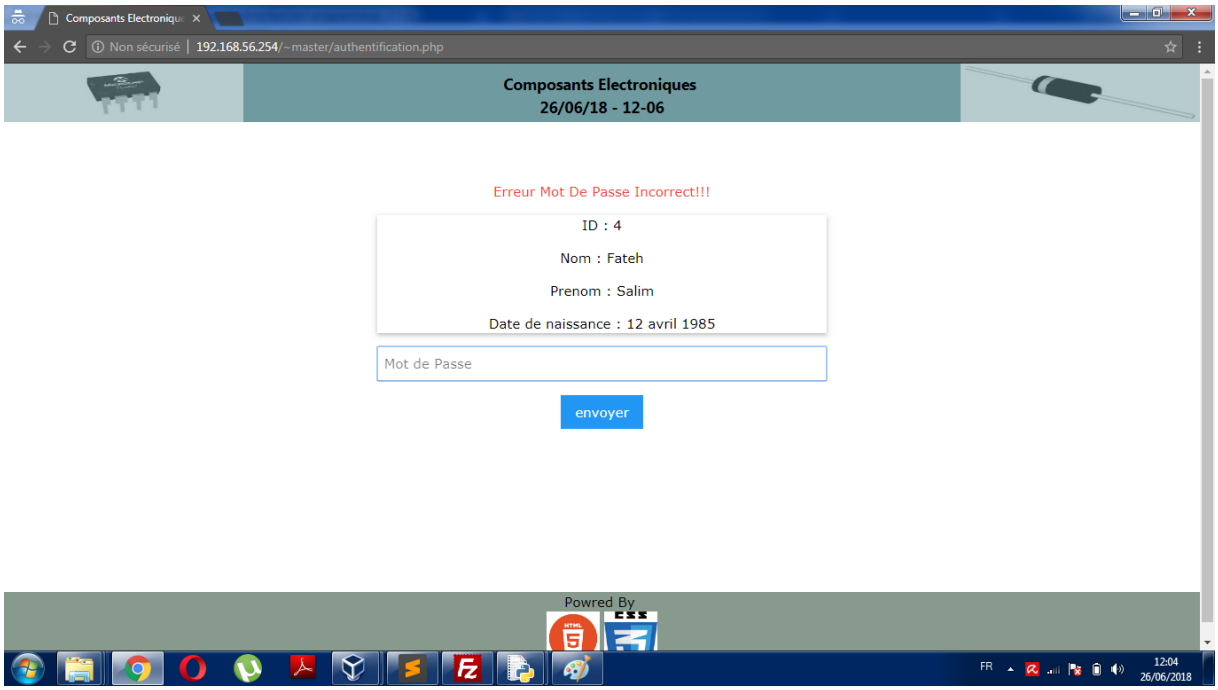


Figure V.5 erreur mot de passe

Après authentification l'utilisateur sera redirigé vers la page d'accueil du site, là où nous distinguons deux types d'utilisateurs :

Cas utilisateur avec droit d'ajout et de consultation :

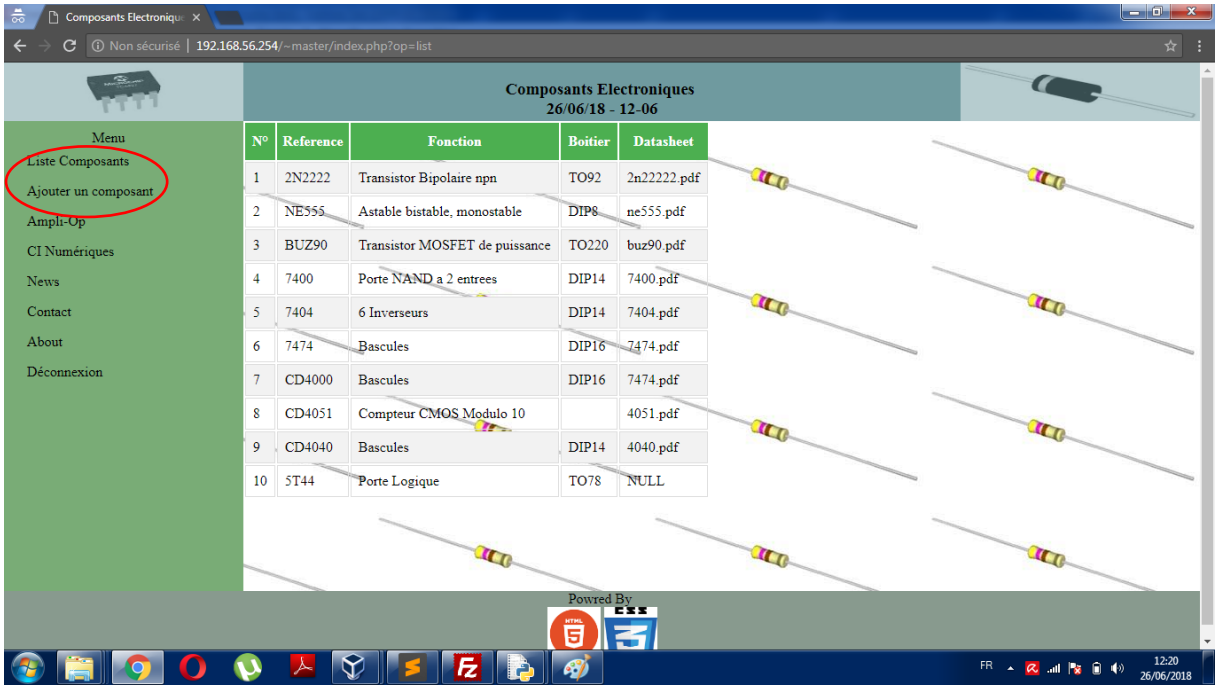
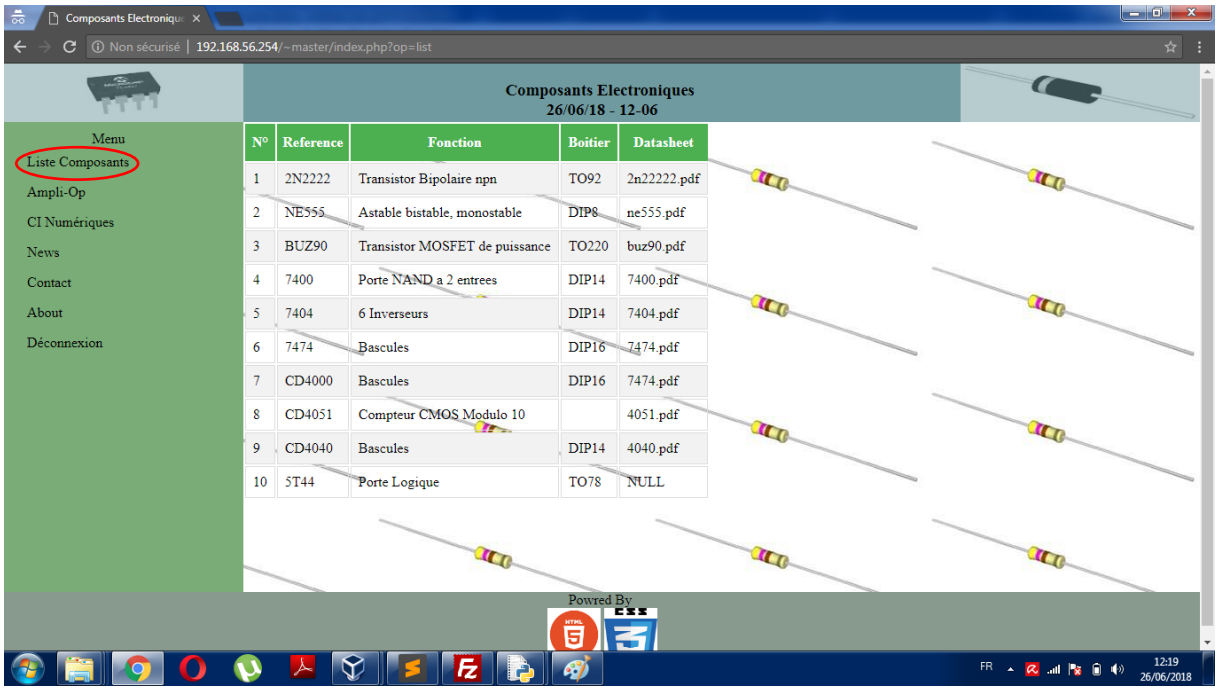


Figure V.6 Interface ajout produit

Cas utilisateur avec juste droit de consulter :



N°	Reference	Fonction	Boitier	Datasheet
1	2N2222	Transistor Bipolaire npn	TO92	2n22222.pdf
2	NE555	Astable bistable, monostable	DIP8	ne555.pdf
3	BUZ90	Transistor MOSFET de puissance	TO220	buz90.pdf
4	7400	Porte NAND a 2 entrees	DIP14	7400.pdf
5	7404	6 Inverseurs	DIP14	7404.pdf
6	7474	Bascules	DIP16	7474.pdf
7	CD4000	Bascules	DIP16	7474.pdf
8	CD4051	Compteur CMOS Modulo 10		4051.pdf
9	CD4040	Bascules	DIP14	4040.pdf
10	5T44	Porte Logique	TO78	NULL

Figure V.7 Interface consulter produit

3.2. Partie personnalisation des cartes

Dans cette partie nous allons faire l'illustration de quelques fonctionnalités du système

Interface de démarrage

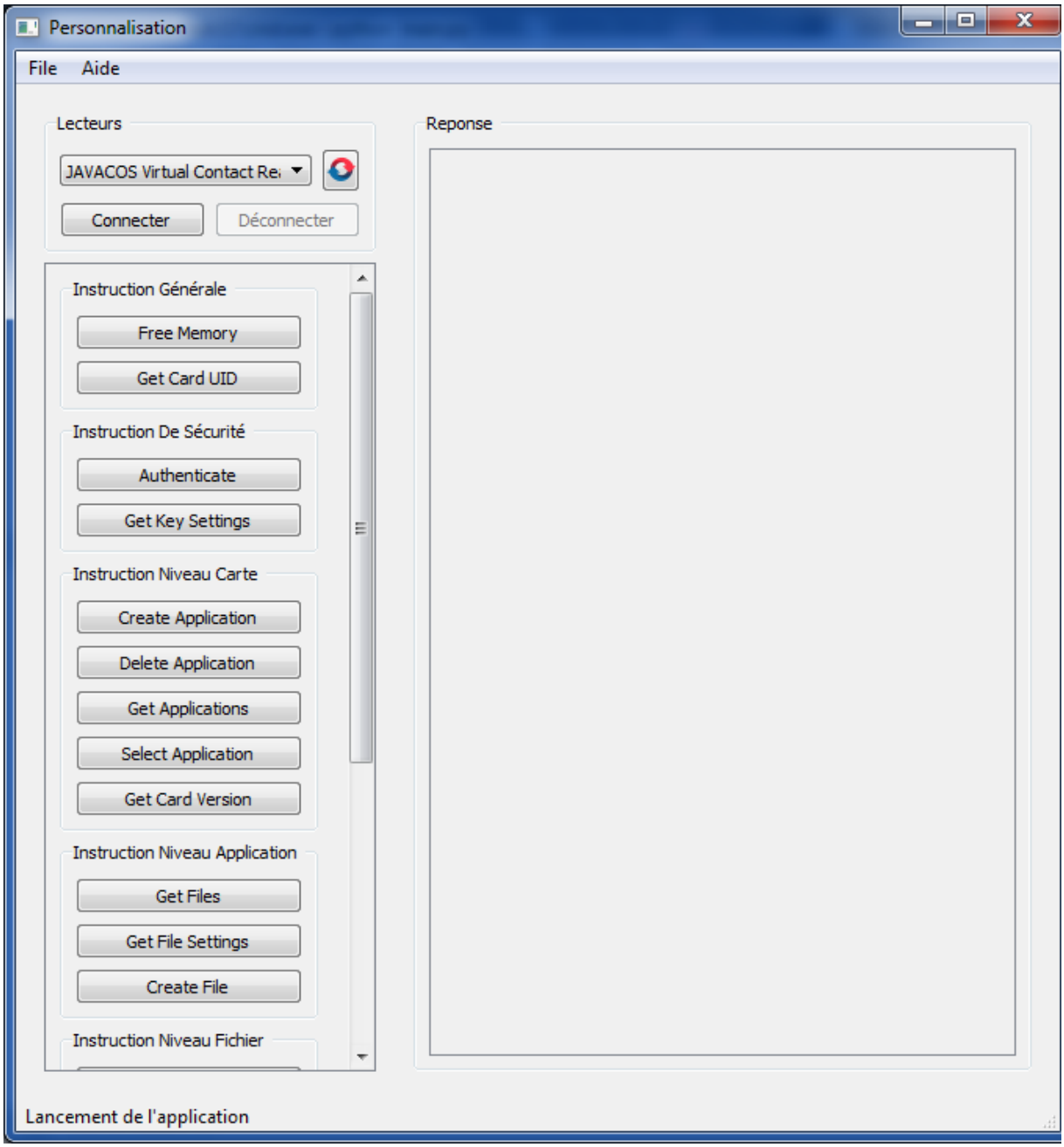


Figure V.8 Interface de personnalisation

L'instruction « Authentifier » :

L'agent de personnalisation Click sur le bouton authenticate, la fenêtre de dialogue suivante luis apparait :

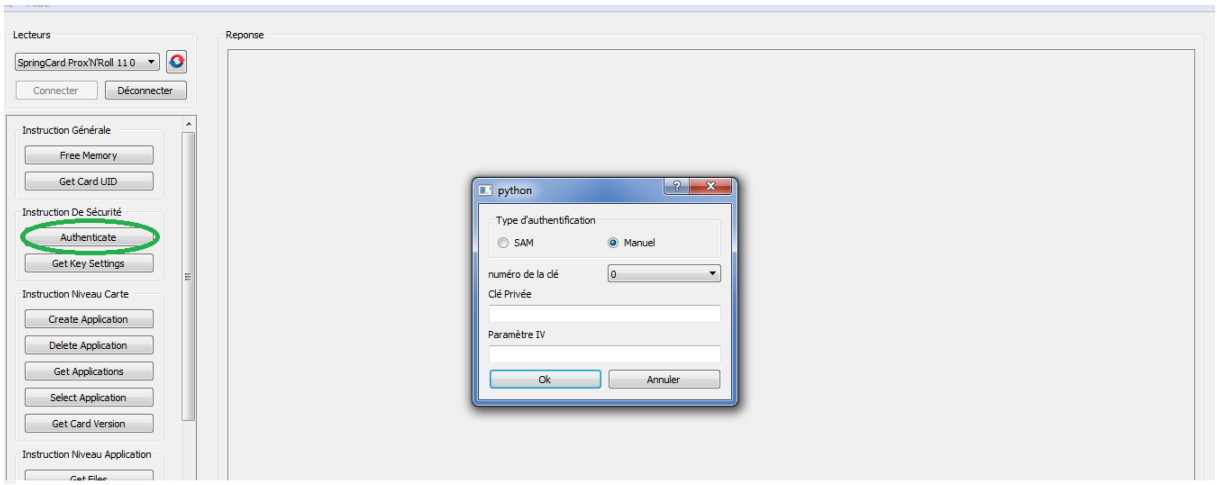


Figure V.9. Interface authenticate

Après remplissage on obtient le résultat suivant :

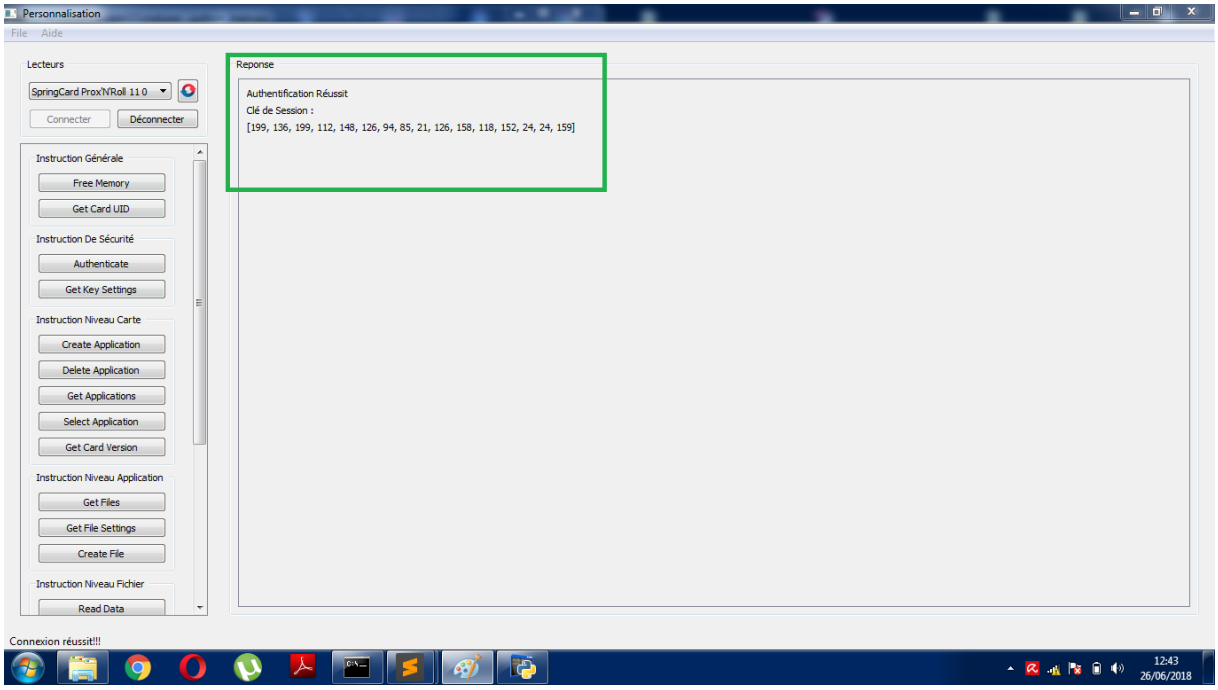


Figure V.10 résultat authenticate

L'instruction « get applications » :

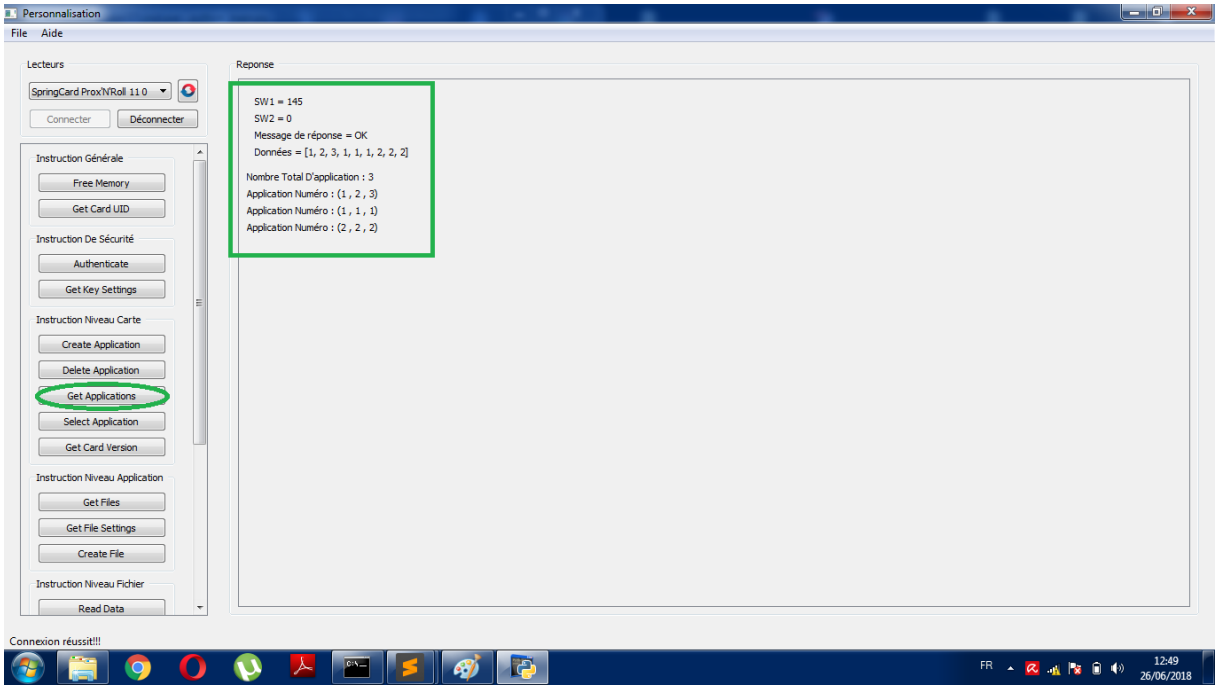


Figure V.11 Résultat get application

L'instruction « get UID » :

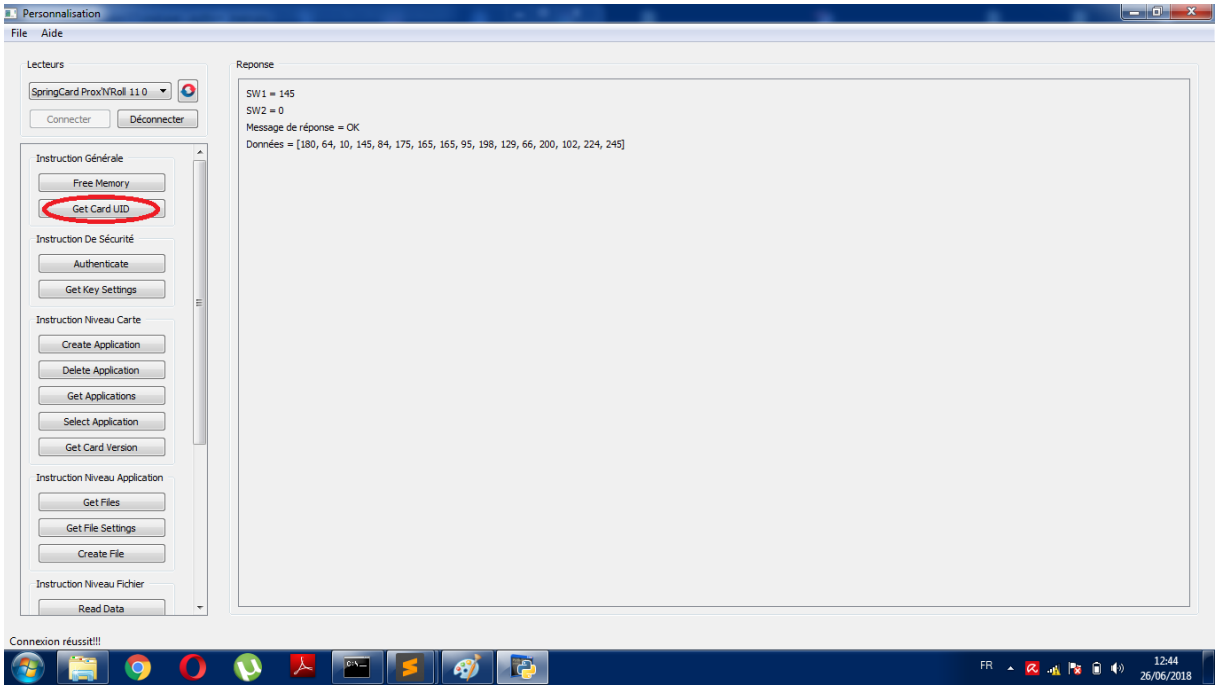


Figure V.12 Résultat get card UID

L'instruction « getcard version » :

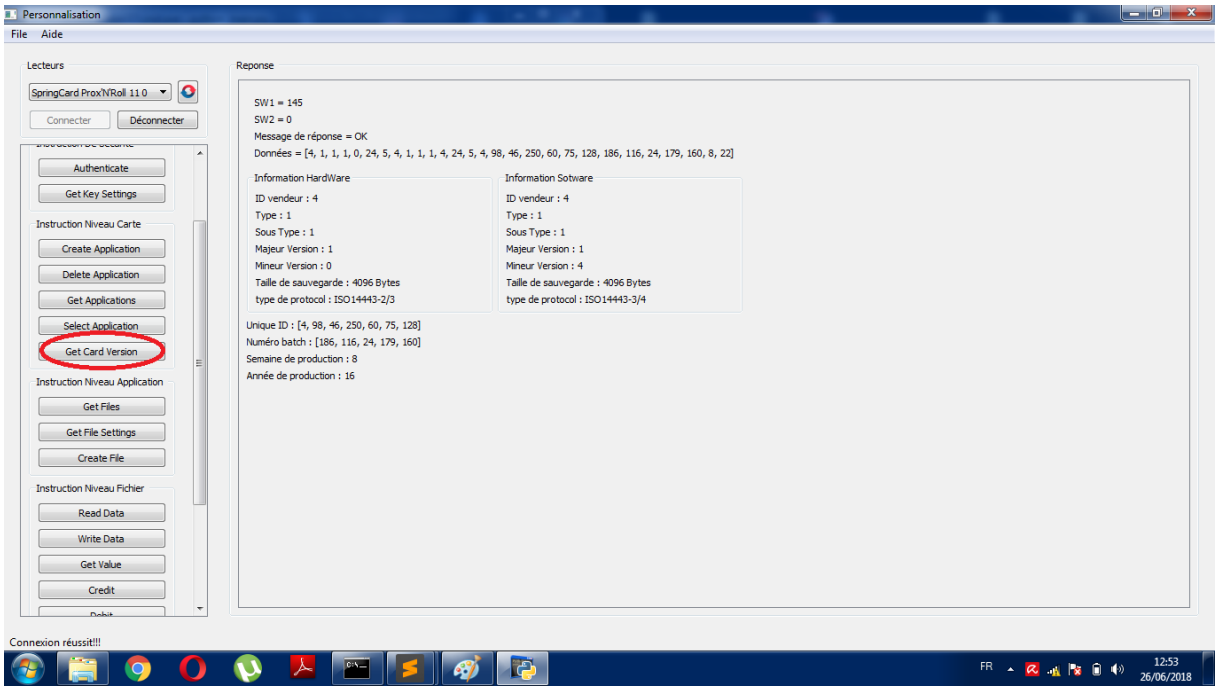


Figure V.13 Résultat getcard version

L'instruction « select application »

Après le clic sur le bouton select applications, la boite de dialogue suivant apparait :

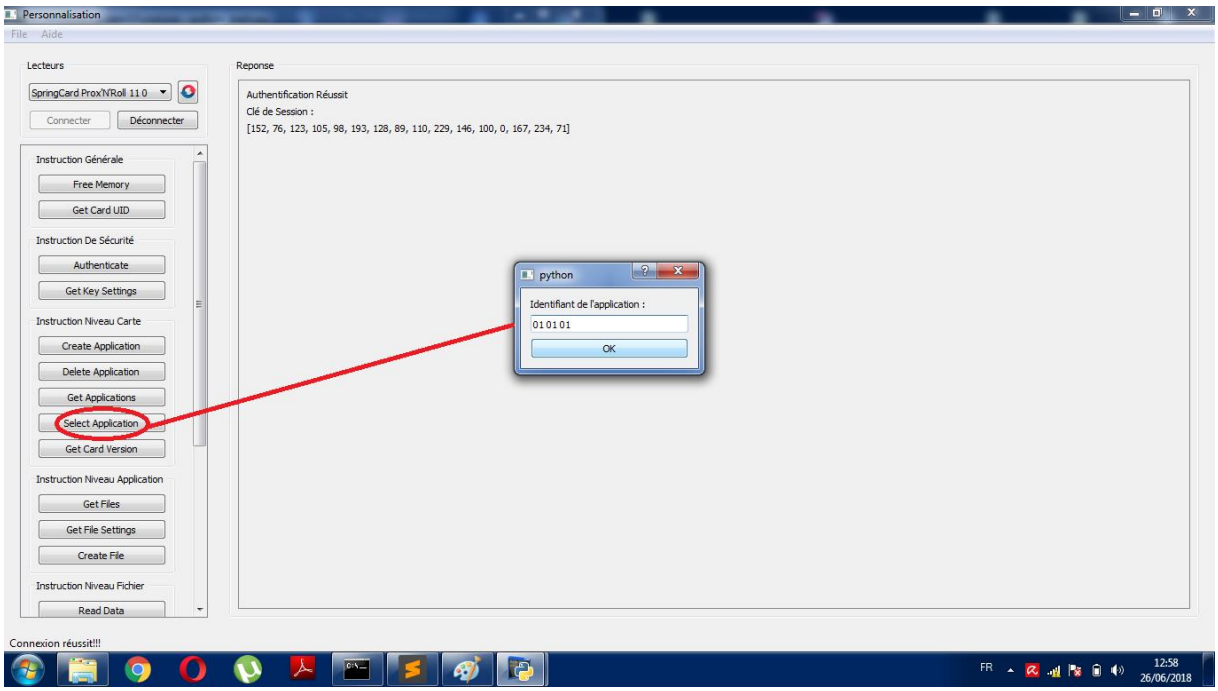


Figure V.14 Interface select Application

On obtient la réponse suivante :

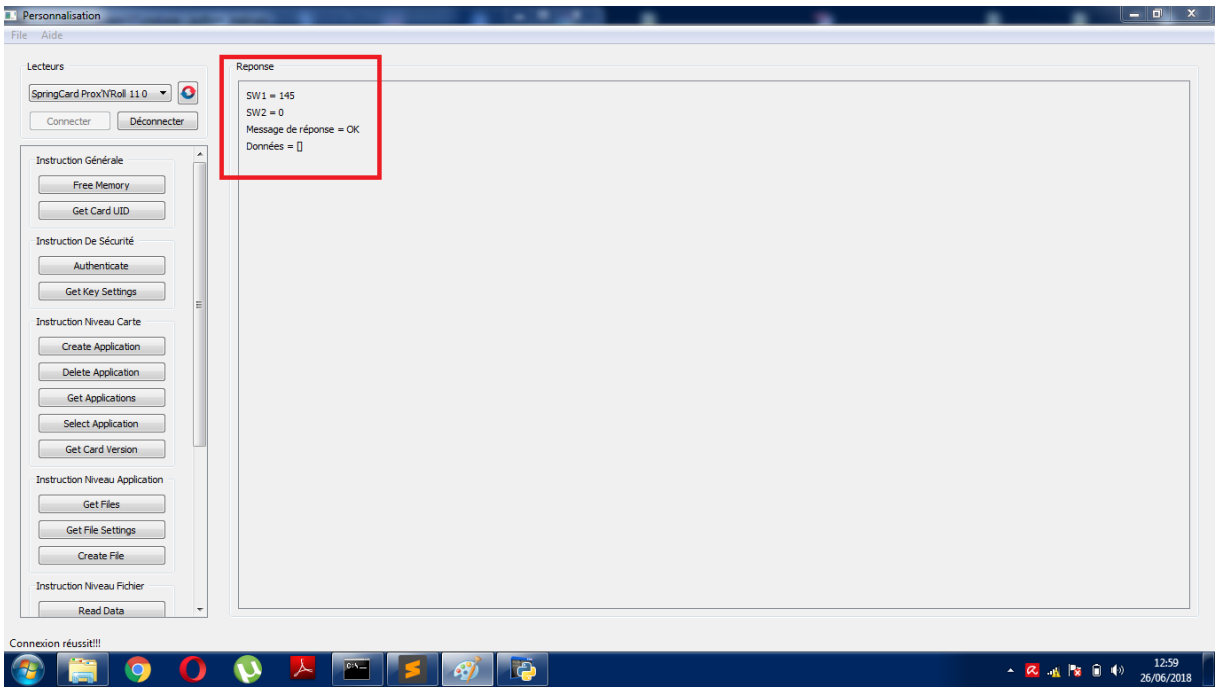


Figure V.15 Résultat select Application

4. Conclusion :

Nous avons présenté dans ce chapitre l'environnement de travail, des outils, de langages de programmation et de bibliothèque utilisé qui nous ont permis de réaliser notre projet.

Nous avons fait la démonstration du fonctionnement réel de notre système qui contient deux parties distinctes que nous avons analysé et conçu dans les chapitres précédents, et nous avons vu son déroulement en quelques captures.

Conclusion générale

Dans le cadre de notre travail, nous avons appris les différents aspects sécuritaires, et son rôle dans les systèmes d'informations et les applications Client/Serveur en utilisant la technologie des cartes à puces.

Le choix des cartes à puces c'est basé sur des critères de sécurités, ainsi que le rapport qualité prix.

Nous avons pu faire une analyse détaillée sur le fonctionnement de notre Système grâce à des schémas de modélisation et de représentation des différentes activités.

En se basant sur l'analyse nous avons fait une conception organisée pour ce thème qui a compris deux parties distinctes, la première pour l'architecture du module d'authentification qui a pour but d'authentifier les utilisateurs grâce à une carte personnelle et un mot de passe, pour la seconde partie elle est pour l'application de personnalisation des cartes utilisateurs.

En fin nous avons pu mettre en œuvre le module d'authentification, ainsi qu'une partie du système de personnalisation.

Comme perspectives à ce travail, l'ajout des certificats numériques pour la SAM permettra une meilleure sécurité. Et nous voulons aboutir à ce que toutes les fonctionnalités de la partie personnalisation soit fonctionnelles.

Table des figures :

Table des références

[1]https://www.commentcamarche.com/contents/212-signature-electronique

[2]https://www.commentcamarche.com/contents/204-introduction-au-chiffrement-avec-des

[3]https://www.commentcamarche.com/contents/208-algorithme-de-chiffrement-rsa

[4]http://www.bibmath.net/crypto/index.php?action=affiche&quoi=moderne/aes

[5]https://www.commentcamarche.com/contents/1033-introduction-a-la-securite-informatique

[6]https://www.commentcamarche.com/contents/995-protection-introduction-a-la-securite-des-reseaux

[7]https://codes-sources.commentcamarche.net/source/list/java-9/284-chiffrement-cryptographie/last?page=2

[8]https://www.securiteinfo.com/conseils/introsecu.shtml

[9]https://www.gralon.net/articles/materiel-et-consommables/materiels-industriels/article-la-carte-a-puce---histoire-d-une-invention-1140.htm

[10] https://cedric.cnam.fr/~bouzefra/cours/Cartes_Intro.pdf

[11] https://perso.telecom-paristech.fr/urien/intro_carte_2012.pdf

[12] https://conf-ng.jres.org/2015/document_revision_1632.html?download

[13] https://www.nxp.com/docs/en/data-sheet/MF3ICDX21_41_81_SDS.pdf

[14] http://acikerisim.deu.edu.tr/xmlui/bitstream/handle/12345/8682/202535.pdf

[15] https://www.theserverside.com/definition/Java-Card

[16] http://www-igm.univ-mlv.fr/~dr/XPOSE2002/puverel/javacard.html