REPUBLIQUE DU BENIN

****

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

****

UNIVERSITE D'ABOMEY-CALAVI (UAC)

****

ECOLE NATIONALE D'ECONOMIE APPLIQUEE ET DE MANAGEMENT (ENEAM)

****

DEPARTEMENT DE L'INFORMATIQUE

Mémoire de fin de formation de master professionnel

Mention : Informatique de Gestion

Spécialité : Génie Logiciel et Audit des Systèmes Informatiques

THEME :
Etude et conception d'un système d'authentification forte avec clé USB sécurisée pour la connexion à une application web

Etudiant

Cyrille Ahmed MIDINGOYI

Directeur de Mémoire

Dr Pélagie HOUNGUE

Année académique : 2012-2013

Dédicaces

A

Mes parents

Remerciements

Je tiens à exprimer toute ma gratitude à tous ceux qui ont œuvré directement ou indirectement à la réalisation de ce travail. L'élaboration du présent mémoire nécessite beaucoup d'efforts notamment axés sur des démarches multiples comprenant documentations, observations, discussions, interprétations et analyses.

Je souhaite tout d'abord remercier chaleureusement, Dr Pélagie HOUNGUE, Enseignante - chercheur à l'Institut de Mathématiques et de Sciences Physiques, pour avoir accepté la supervision de ce mémoire. Je la remercie également pour sa disponibilité, sa patience et ses pertinentes remarques tout au long de cette période de recherche en dépit de son agenda très chargé.

Je tiens à remercier Dr Théophile DAGBA, Chef du Département de l'Informatique à l'ENEAM-UAC, qui a pris une part active dans la réussite de cette formation.

Mes remerciements vont également à l'endroit de l'administration de l'ENEAM et tous les professeurs qui m'ont encadré tout au long de cette formation.

Je ne saurais terminer sans adresser ma reconnaissance à l'endroit de M. Lucien ATTINDOGBE, mon maître de stage et M. Razack ICHOLA pour tout leur encadrement durant le stage.

En dernier lieu, j'adresse mes sincères remerciements à ma famille et mes amis, qui m'ont toujours soutenu et qui ont su m'accompagner et m'encourager durant tout mon cursus universitaire. Je pense ainsi à mon père Soulé MIDINGOYI et à ma mère Thérèse HOMONHEDO qui ont toujours prié Allah de voir leur fils obtenir le Doctorat. Chers parents, qu'Allah exauce vos vœux.

A ma femme Amanda DJAMBA et mes enfants Safwane et Asmah pour l'envie de réussir qu'ils éveillent en moi.

Résumé

La violation de gestion d'authentification est l'un des risques de sécurité informatique qui regroupe toutes les vulnérabilités pouvant mener à une usurpation d'identité. Ainsi, les applications web qui ne garantissent pas une bonne gestion d'authentification permettent aux attaquants d'accéder à des fonctionnalités auxquelles ils n'ont pas droit. En conséquence, la protection des accès aux applications des entreprises repose sur la mise en place d'un système d'authentification. Etant donné les faiblesses constatées des systèmes d'authentification courants, nous avons décidé d'étudier et d'implémenter un système d'authentification à double facteur (mot de passe et clé USB) répondant au protocole standard U2F (Universal Second Factor) afin de renforcer la protection des accès à un exemple d'application web sur un serveur local.

Pour atteindre cet objectif, nous avons suivi dans un premier temps un stage dans une entreprise afin d'appréhender entre autres la sécurité de leur système d'information, notamment en matière d'authentification et par la suite faire des recommandations pour son amélioration. Ensuite, nous avons fait une synthèse non seulement des concepts et mécanismes d'authentification les plus communs mais aussi des attaques d'usurpation d'identité courantes. Enfin, nous avons étudié le protocole U2F, ce qui nous a permis la mise en œuvre d'un système d'authentification forte avec l'authentificateur USB de Yubico.

Pour ce faire, nous avons développé le module de connexion de l'application web afin qu'il puisse répondre aux exigences du protocole U2F ; ceci constitue l'originalité de notre travail.

Ce type d'authentification mis en place permet de pallier à de nombreuses failles dont les attaques par force brute, les keyloggers, l'attaque de l'intercepteur, le phishing et bien d'autres.

Mots-clés : authentification, protocole U2F, application web, sécurité informatique, attaques

Abstract

Broken Authentication management is one of the IT security risk which includes all the vulnerabilities that could lead to identity theft. Thus, the web applications that don't provide secure authentication management allow attackers to access some functionalities to which they are not entitled. Consequently, protecting access to business application based on the establishment of an authentication system. Given the weaknesses of the current authentication systems, we decided to study and implement a two-factor authentication system (password and USB key) responsive to standard U2F protocol to strengthen the protection of access an example of local web application.

To achieve this goal, we followed initially an internship in a company in order to understand the security of their information system especially in terms of authentication and thereafter make recommendations for improvements. Then we have summarized the concepts and authentication mechanisms but also the most common attacks relative to authentication. Finally we studied the U2F protocol, which allowed us to implement a strong authentication system with USB Authenticator Yubico.

We have developed the connection module of the web application in order to meet the requirements of U2F protocol; this is the originality of our work.

This type of implemented authentication system overcomes many flaws including brute force attacks, the keyloggers, the middle-man attacks, phishing and others.

Key words: authentication, U2F protocole, web application, IT security, attacks

SOMMAIRE

Liste des figures

Liste des abréviations

ASECNA : Agence pour la Sécurité de la Navigation Aérienne en Afrique et à Madagascar

CA : Certificate Authority (Autorité de certification)

DHCP : Dynamic Host Configuration Protocol (Protocole de configuration dynamique des hôtes)

DNS : Domain Name Service (Service de noms de domaine)

DSA : Digital Signature Algorithm (Algorithme de Signature Numérique)

ECDSA : Elliptic curve digital signature algorithm (Algorithme de Signature Numérique sur les courbes elliptiques)

FIDO : Federal Identity On Line (Fédération d'identité en ligne)

FREDA : logiciel de Facturation et de Redevances Aéronautiques

ID : Identity (identité)

IP : Internet Protocol (Protocole Internet)

ISACA: Information Systems Audit and Control Association

ISF : Information Security Forum

ISO : Organisation Internationale de Normalisation

KDC : Key Distribution Center (Centre de Distribution de clés)

LDAP: Lightweight Directory Access Protocole (Protocole d'accès au repertoire allege)

MD5: Message Digest 5

MITM: Man-In-The-Middle (homme du milieu)

MOL2P : Multiplexeur Optimisant la liaison avec Priorité à la parole

NIP: Numéro d'Identification Personnel

NIST: National Institute of Standards and Technology

OTP : One-Time Password (mot de passe unique)

OWASP: Open Web Application Security Project

PAP : Password Authentication Protocol (protocole d'authentification basé sur le mot de passe)

PDA : Personal Digital Assistant (Assistant numérique personnel)

RFID : Identification par Radio Fréquence

RSI-Météo : Réseaux et Systèmes Informatiques

SADIS : Satellite de Distribution

SHA : Secure hash Algorithm

SIOMA : Système intégré d'Observation Météorologique d'Aérodrome

SSL: Secure Socket Layer

TLS: Transport Layer Security

TOTP: Time-based One Time Password

TUP : Test User Presence (Test de presence de l'utilisateur)

U2F: Universal Second Factor

URL : Uniform Resource Locator (localisateur Uniforme de Ressource)

USB : Universal Series Bus

Introduction générale

Les systèmes d'information constituent un élément stratégique permettant d'améliorer la productivité des entreprises en optimisant les processus et en réduisant les coûts. Pour être compétitives sur le marché, les entreprises mettent en place des applications en vue de remplacer les tâches récurrentes par des traitements automatisés. Ces applications sont souvent conçues pour être utilisées dans des réseaux intranet ou internet. Elles permettent d'accéder à des données confidentielles qu'il est primordial de protéger. Au Bénin, la loi N°2009-09 portant protection des données à caractère personnel délibérée le 27 Avril 2009 révèle toute la sensibilité des données et comporte des dispositions qui régissent tout mécanisme d'identification des personnes.

Aussi bien qu'il est nécessaire de sécuriser les réseaux locaux ou internet, il est fondamental de garantir la sécurité des applications web. Plusieurs techniques de sécurisation des applications web existent telles que la mise en place d'un pare-feu applicatif, le chiffrage des flux, la limitation des protocoles à risque… et sans oublier les techniques pour renforcer la sécurité d'accès aux applications web. Pour cela, des mécanismes d'authentification sont mis en place. Il est courant de remarquer que la plupart de ces mécanismes sont de type **mot de passe et nom d'utilisateur** et présentent de nombreuses failles telles que les attaques par force brute, les keyloggers, les injections SQL etc. Les pirates possèdent des systèmes de plus en plus perfectionnés et peuvent retrouver facilement un identifiant et un mot de passe. Afin de lutter contre les « pirates » les entreprises changent régulièrement les mots de passe de leurs employés. Cependant, cette mesure, pour sécuriser le système d'information, fait que les employés auront du mal à mémoriser leur mot de passe. Ils sont parfois contraints à noter leur mot de passe sur un bout de papier auquel tout le monde peut accéder. Aussi, plus de quarante années de recherche ont démontré que les mots de passe sont en proie à des problèmes de sécurité (Morris et Thompson, 1979).

Les récentes prises de contrôle des comptes utilisateurs ont mis en évidence le défi de la sécurisation des données utilisateur en ligne car les comptes ne sont souvent protégés que par un mot de passe faible. Selon Kindervag et *al.* (2015), le coût global - direct et indirect - de la cybercriminalité dans le monde en 2014 se serait élevé à la somme colossale de 575 milliards de dollars.

La bonne démarche consisterait donc à renforcer l'authentification afin d'empêcher des intrus de pénétrer le réseau, mais surtout à verrouiller les données par des solutions d'encryptage, car il faut admettre qu'il n'y a pas de « risque zéro » en matière de cyber-attaques. La recherche académique a produit de nombreuses propositions pour se départir des mots de passe, mais dans la pratique de tels efforts ont largement échoué (Bonneau et *al.*, 2012).

De nos jours, de nombreuses entreprises, pour sécuriser leur système d'information, mettent en place des systèmes d'authentification forte qui requièrent la combinaison de deux facteurs d'authentification. En raison du coût de l'authentification forte, son usage reste aujourd'hui réservé aux grandes structures, ou plus généralement aux secteurs critiques de l'industrie et des services (banque, énergie, défense, aéronautique, automobile, recherche scientifique).

De nombreux fournisseurs de services augmentent l'authentification par mot de passe avec un second facteur sous la forme par exemple d'un mot de passe unique (OTP) (Railton et Kleemola, 2015). Malheureusement, les jetons OTP comme deuxième facteur sont vulnérables à des attaques relativement courantes telles que le phishing (Railton et Kleemola, 2015), ce qui limite le succès et le déploiement des jetons OTP comme un deuxième facteur fiable et sécurisé. Aussi, les cartes nationales d'identité et les cartes à puce nécessitent des matériels personnalisés ou des pilotes de logiciels avant leur utilisation. En fonction de la mise en œuvre, ces systèmes permettent difficilement aux utilisateurs de protéger leur vie privée (Harbach, Fahl, Rieger, et Smith, 2013).

Pour remédier à toutes ces faiblesses, les géants du web notamment Google, Lenovo, PayPal, LG, BlackBerry, Intel, Microsoft ont décidé de créer en Juillet 2012 une alliance dénommée FIDO (Federal Identity On Line) pour élaborer un standard en matière d'authentification forte appelée protocole U2F (Universal Second Factor) mais aussi en réponse au manque d'interopérabilité des solutions d'authentification forte. Ce protocole permet l'utilisation d'une clé de sécurité physique pour la validation en deux étapes.

Au vu de toutes ces observations et pour mieux s'imprégner des mesures de sécurité des systèmes informatiques notamment dans le domaine de l'authentification dans les entreprises nous avons eu l'honneur d'être accueilli à l'Agence pour la Sécurité de la Navigation Aérienne en Afrique et à Madagascar (ASECNA). Cette agence nous a permis de mettre en pratique les connaissances théoriques acquises au cours de la formation et d'acquérir ainsi une expérience en contexte professionnel, en capitalisant des savoirs pouvant nous aider à la

rédaction du projet personnel de fin d'études portant sur le thème « Etude et conception d'un système d'authentification forte avec clé USB pour la connexion à une application web ».

Ainsi ce mémoire a pour objectif de comprendre les techniques d'authentification existantes et d'implémenter un système d'authentification à deux facteurs dont l'un des facteurs est le mot de passe et le second une clé de sécurité USB en vue de son intégration dans une application web pour assurer une connexion sécurisée.

Dans ce travail, après une introduction générale, nous présenterons dans le premier chapitre la structure d'accueil et les différents travaux effectués. Ensuite, le second chapitre portera sur la revue de littérature relative à la sécurité informatique, en particulier les concepts et techniques d'authentification existantes, les failles liées à une faible authentification. Enfin, le troisième chapitre sera dédié à l'implémentation du module d'authentification forte avec une clé USB sécurisée pour une connexion sécurisée de l'application web. Dans ce dernier chapitre, nous tenterons donc au prime abord de comprendre le protocole standard U2F afin de faciliter sa mise en œuvre dans une application web existante.

Nous achèverons ce mémoire par une conclusion générale dans laquelle nous essayerons de dresser le bilan de notre travail et ouvrir des perspectives sur nos futurs travaux.

CHAPITRE 1

Chapitre 1 : Présentation du stage

1- Présentation de la structure d'accueil

1-1- Historique

Le 12 décembre 1959 à Saint Louis au Sénégal, les chefs d'Etat et de gouvernements des Etats autonomes issus des ex fédérations de l'Afrique Equatoriale Française (AEF), de l'Afrique Occidentale Française (AOF) et de Madagascar signent la convention qui va donner naissance à l'Agence pour la Sécurité de la Navigation Aérienne en Afrique et à Madagascar (ASECNA). Cette Agence dont le siège est à Dakar au Sénégal, comprend aujourd'hui 17 Etats membres africains et la France.

Le motif de cette création est non seulement d'éviter de morceler l'espace aérien mais aussi d'unir les moyens financiers, les capacités matérielles et humaines afin d'œuvrer en commun pour mieux assurer la sécurité de la navigation aérienne.

Aux fins d'une africanisation nécessaire et souhaitable du personnel, l'ASECNA s'est transformée pour s'adapter au nouveau contexte politique et économique. Elle devient ainsi un modèle en matière de coopération inter- États africains et l'un des leaders du développement en Afrique des technologies de navigation par satellite.

Aujourd'hui, à l'heure de la formation des agents aux technologies innovantes, l'esprit qui a présidé à la création de l'ASECNA en 1959 reste le même : placé sous le signe de l'efficacité, de la solidarité et de la coopération dans le but d'assurer une entente cordiale avec ses usagers et leur sécurité optimale.

Forte d'une expérience profondément enracinée dans l'histoire, l'ASECNA a donc tous les atouts en main pour aborder les meilleures conditions de l'aviation civile du XXIème siècle avec toujours une exigence constante de qualité au service de la sécurité aérienne.

1-2- Missions de la Représentation

Dans chaque Etat membre, les missions de l'ASECNA sont assurées par une Représentation gérée par un Représentant. Ce dernier est nommé par le Directeur Général en accord avec le Ministre de tutelle concerné. Il est responsable des activités de l'Agence dans son Etat d'affectation. Il a pour mission essentielle d'assurer la sécurisation des biens et des personnes dans le domaine de la navigation aérienne.

1-3- Organisation

La Représentation de l'ASECNA auprès du Bénin est structurée suivant l'organigramme présenté en annexe 4.

Le stage s'est déroulé dans l'Unité Réseaux et Systèmes informatiques (RSI-Météo) au sein du service de Maintenance Infrastructures Radio Electriques et informatique (MIRE)

Pour mener à bien sa mission, La MIRE est divisée en 5 unités :

- Unité Réseaux et Systèmes Informatiques
- Unité Communication Navigation Surveillance
- Unité Energie et Balisage
- Unité Transit et Gestion des Stocks
- Unité Qualification et Intégration du Personnel

Notre stage a été exclusivement réalisé dans l'unité RSI-Météo qui est chargée, entre autres, de l'installation, du suivi et de la maintenance :

- des équipements météorologiques
- des systèmes informatiques opérationnels (commutateurs de messages, serveurs, bases de données opérationnelles …)
- du parc des matériels et logiciels informatiques de gestion

2- Déroulement du stage

Au cours de notre stage, nous avons mené des activités qui peuvent être résumées en ces points :

- Description du système informatique opérationnel et du réseau local
- Tâches pratiques réalisées
- Consultations de documents
- Entretiens

Ces entretiens et consultations nous ont permis de dégager les aspects de sécurité du système informatique de la Représentation. Ensuite, nous avons essayé de faire des propositions en vue de son amélioration.

Dans cette partie, nous avons brièvement décrit les activités menées sans perdre de vue l'objectif principal qu'est d'identifier le niveau de sécurité des systèmes en matière d'authentification.

2-1- Description des équipements

2-1-1- Equipements météorologiques

Comme nous l'avons évoqué précédemment, cette unité a en charge le maintien en bon fonctionnement des équipements météorologiques. Il s'agit entre autres de :

- ✓ SADIS (Satellite de Distribution) :

Il est constitué d'un émetteur, d'un récepteur et d'un ordinateur servant d'interface homme-machine. Il fait l'acquisition et la transmission des données météorologiques telles que les messages d'observations aéronautiques, les prévisions d'aéroport (TAF), les avertissements d'intempéries dangereuses (SIGMET). Ces données permettent aux prévisionnistes de constituer le dossier de vol à l'usage des pilotes avant le décollage.

- ✓ SIOMA (Système intégré d'Observation Météorologique d'Aérodrome)

Il est composé de plusieurs capteurs intelligents reliés à une baie de traitement appelée MIRIA par l'intermédiaire du réseau de terrain CIBUS (bus de capteur intelligent). Les informations traitées par la baie MIRIA sont exploitées à partir des PC, et visualisées à la tour de contrôle. Les différents capteurs sont : le capteur de température, d'humidité, de vent, de luminancemètre, de pression et de balisage.

- ✓ Le système de radiosondage :

Il est constitué d'une radiosonde, d'un ballon en latex et d'un ordinateur. Un ensemble de capteurs mesurant entre autres les données de pression, de température, d'humidité et de vent, intégrés dans un boîtier appelé radiosonde, s'élève dans l'atmosphère grâce à un ballon en latex et permet de tracer un profil vertical des données mesurées (une coupe de l'atmosphère).

2-1-2- Equipements de communication

La maintenance des équipements de communication est aussi assurée en partie par l'unité RSI.

- ✓ La chaîne radio : pour assurer les échanges vocaux entre contrôleurs de la circulation aérienne

✓ MOL2P (Multiplexeur Optimisant la liaison avec Priorité à la parole) :

C'est un équipement qui assure la transmission des données asynchrones, synchrones, de la phonie, du fax.

✓ Système opérationnel TopSky-ATC : Composé de postes opérationnels et techniques :

- Les postes opérationnels comprennent :
  Exécutive Controller (EC) : Gère le trafic en-route et approche ;
  Planning Controller (PLC) : Assiste l'EC
  Tower Controller (TWR) : Gère les décollage & atterrissages
  Supervisor (OPSUP)/ Team Manager: Gestion de l'équipe
  Playback Operator (REPLAY)
- Postes techniques
  Database Management (DBM) : post-traitement des données,
  Préparation facturation,
  Technical Supervisor (TKSUP)

✓ AMS1500 (Automatic Message system 1500)

C'est un serveur de messagerie automatique assurant la commutation automatique des messages d'un correspondant à un autre.

2-1-3- Système informatique de la Représentation

La représentation de l'ASECNA au Bénin dispose de deux types de systèmes informatiques bien distincts :

- Un système informatique opérationnel pour le traitement des informations techniques liées à la navigation aérienne.
- Un système informatique pour assurer la gestion administrative

➢ **Description du système informatique opérationnel**

Le système informatique opérationnel constitué de sous-systèmes dont certains sont décrits plus hauts (SADIS, TopSky…) est dédié à l'échange des données aéronautiques. Ce système est représenté par la figure 1.

Il est constitué d'une antenne VSAT qui assure la réception et l'émission des données aéronautiques, la téléphonie et l'accès à Internet. Il permet de relier les sites des représentations avec le site de la Direction Générale. Cette antenne est reliée à des modems pour assurer la

communication avec les autres sous-systèmes de par sa fonction de modulation/démodulation du signal.

- **Description du système informatique pour la gestion administrative de la représentation**

D'après les visites et les documents qui nous ont été fournis, cette partie du Système informatique peut être répartie comme suit :

- Des Micro-ordinateurs de type PC
- Des Serveurs (facturation, messagerie, contrôleur de domaine, antivirus, DHCP et DNS)
- Système d'Exploitation (Windows et linux)
- Les applications : PGI (Progiciel de Gestion Intégrée), le logiciel de Facturation FREDA
- Des équipements réseaux : switch (type DLINK), routeur (CISCO)
- Des imprimantes monoposte et réseau

Les équipements du réseau utilisent des adresses IP privée de classe A avec un masque réseau de classe B. Ces adresses sont non routables sur internet.

Tous les postes de travail connectés au réseau sont placés sur le même segment. Des Switch en cascade sont utilisés et les différents postes de travail leurs sont connectés.
Un réseau privé virtuel est mis en place pour la liaison avec la Direction Générale et pour une exploitation plus sécurisée des applications informatiques et échanges de données.
Le réseau local est connecté à un routeur qui reçoit deux types de données : les données issues des applications informatiques de la structure et les données internet (mails, téléchargements, …). Il est ensuite relié à deux modems de marque « iDirect» pour assurer la modulation et la démodulation des deux types de signaux avec des débits différents, et qui parviennent enfin au satellite VSAT pointé vers Dakar.

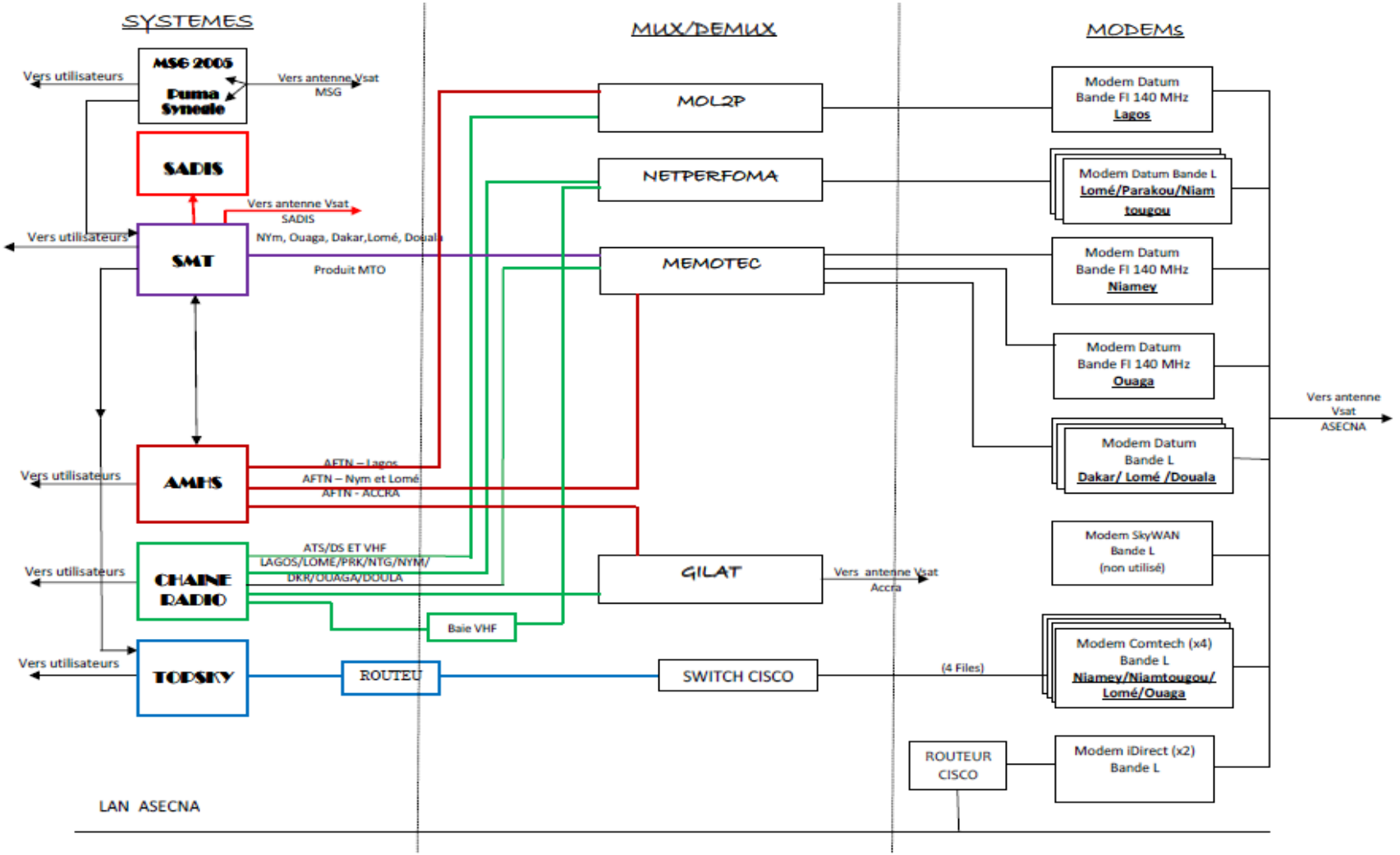


Figure 1: Système informatique opérationnel de l'ASECNA

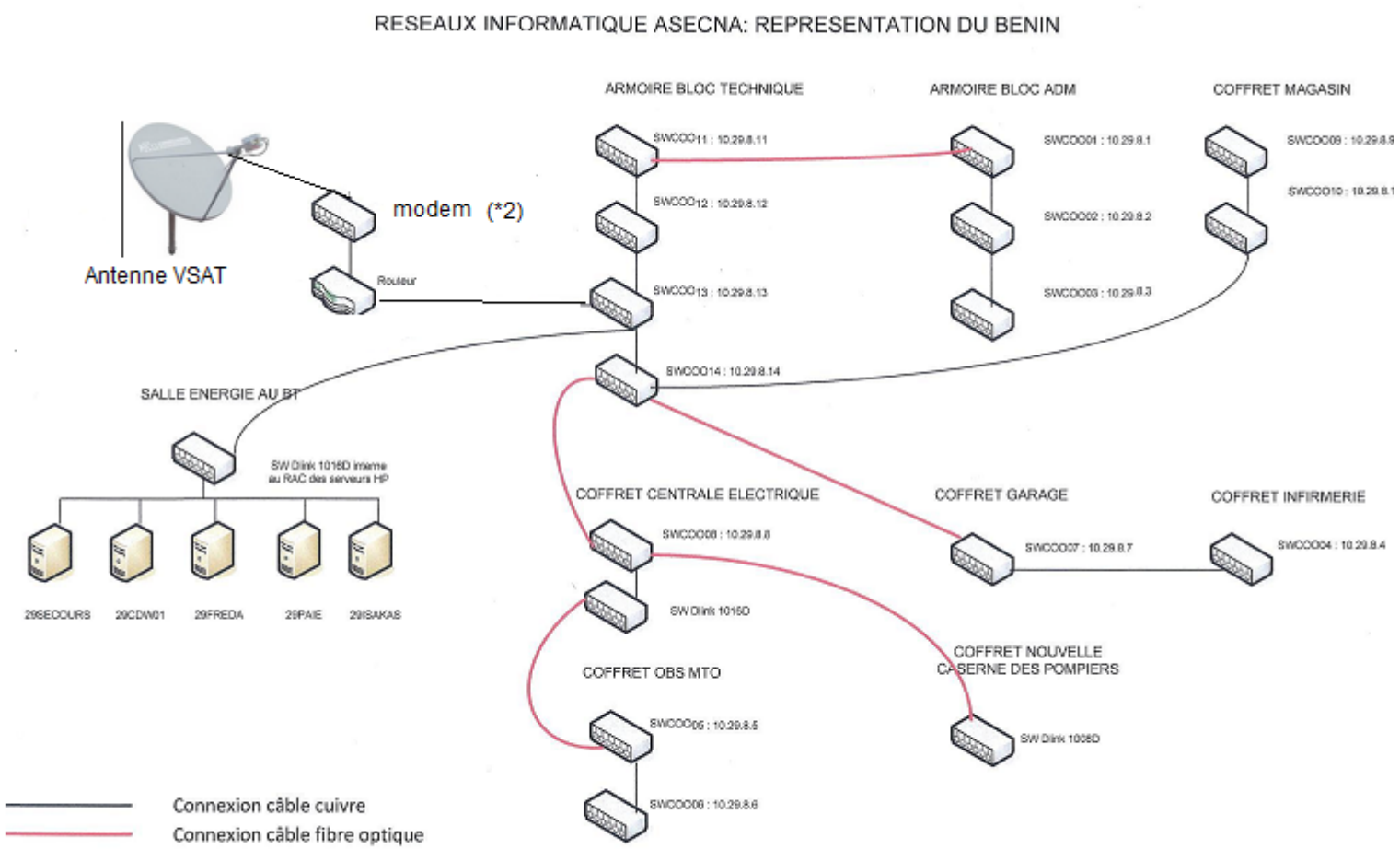


Figure 2: Système informatique de la gestion administrative de la Représentation de l'ASECNA

2-2- Tâches pratiques

Durant cette période de stage, nous avons pu effectuer des tâches quotidiennes sous l'instruction des techniciens de l'Unité RSI-Météo. Il s'agit entre autres de :

- Recueil de l'état de fonctionnement des équipements à travers la consultation des messages d'incidents et certaines mesures afin de vérifier si elles se situent dans la plage normale requise.
- Dépoussiérage et Nettoyage des équipements
- Création d'un groupe d'utilisateurs de la Direction Générale grâce au contrôleur de domaine pour accès à distance à la base de données FREDA
- Planification de la sauvegarde, de la mise à jour du serveur FREDA et de sa synchronisation avec le serveur central à Dakar
- Intégration des données du serveur de Paie dans le serveur Facturation

2-3- Analyse du système

Après les visites, échanges et consultation de documents, nous avons pu noter des mesures de sécurité qui ont été prises par la Représentation pour assurer au mieux la sécurité des infrastructures et des utilisateurs du réseau.

2-1-1- Aspect de sécurité existante

- **Sécurité physique**

- Entretiens réguliers des locaux
- Contrôle d'accès du site par des agents de sécurité
- La salle des serveurs est interdite d'accès aux étrangers
- L'alimentation est secourue par des groupes électrogènes pour assurer la continuité de service des ressources critiques
- Les extincteurs d'incendie sont installés dans les couloirs

- **Sécurité logique**

Pour la sécurité logique la Représentation a mis en place les moyens suivants :

- Mise en place d'un serveur de sauvegarde de données et d'un serveur de messagerie
- Planification automatique de la sauvegarde et de la synchronisation avec le serveur de la Direction Générale

- Redondance matérielle dans les secteurs critiques comme la tour de contrôle pour assurer la continuité des services

➢ **Sécurité réseau**

- Mise en place d'un contrôleur de domaines pour gérer des ressources liées à la gestion du réseau (domaines, comptes utilisateurs, stratégies de sécurité,...)
- Pour l'accès au réseau internet, le mécanisme de NAT est assuré par un informaticien de la Direction Générale
- Le filtrage des accès est assuré par la Direction Générale
- Présence des alarmes pour notifier l'état de fonctionnement des équipements
- Réseau VPN pour la liaison avec la Direction Générale qui assure le chiffrage du flux de l'ensemble du trafic.

➢ **Sécurité des systèmes**

- Système antiviral : un serveur d'antivirus Kaspersky
- Surveillance de l'état de fonctionnement des systèmes et services

Il est important de souligner qu'il existe au sein de l'Agence, une politique du système de management qui inclut la sécurité du système d'information aéronautique nommée Système de management intégré (SMI). La mise en œuvre de cette démarche intégrée passe par :

- l'identification des risques en matière de sécurité ;
- la mise en œuvre des mesures correctives nécessaires au maintien de performances de sécurités convenues ;
- la surveillance continuelle et l'évaluation régulière des performances de sécurité ;
- l'amélioration continue des performances globales du système de gestion de la sécurité ;

2-2- Propositions d'amélioration du système informatique de gestion

Etant donné qu'un système parfait n'existe pas, nous avons formulé des essais de propositions pour l'amélioration du système informatique assurant la gestion administrative de la Représentation. Ces propositions se présentent comme suit :

- Mettre en place au sein de la Représentation une politique de sécurité du réseau local.
- Appliquer une séparation physique du réseau local en utilisant les commutateurs entre les équipements interconnectés selon le degré de confidentialité.

- Décentraliser l'administration du réseau local.
- Mettre en place une structure dédiée à la planification et au suivi des travaux informatiques pour une bonne exploitation et gestion du parc informatique et du réseau local. Cette structure sera indépendante de celle qui s'occupe du réseau synoptique opérationnel.
- Configurer les postes utilisateurs en deux sessions. Il est important d'avoir au moins deux sessions pour chaque poste, une pour l'utilisateur avec privilège restreint de préférence pour ne pas modifier la configuration initiale et la deuxième pour l'administrateur qui est le seul à pouvoir modifier les paramètres de base.
- Former les utilisateurs à la sécurité informatique. Il faut s'assurer que les utilisateurs soient sensibilisés aux risques informatiques et qu'ils adhèrent aux exigences de sécurité des systèmes d'information.
- Implanter un système de détection d'intrusion même si la configuration actuelle ne laisse pas envisager une attaque externe.
- Rendre plus opérationnel le serveur d'antivirus Kaspersky afin qu'il permette réellement un déploiement et une mise à jour automatique sur les postes clients.
- Mettre en place un système d'identification centralisée pour éviter les saisies multiples de l'identité pour accéder à plusieurs applications.
- Mettre en place un système d'authentification forte pour la connexion aux applications de gestion afin de renforcer leur sécurité. Cette dernière proposition fait l'objet du présent projet professionnel.

3- Conclusion et Perspectives

La Représentation de l'ASECNA auprès du Bénin dispose d'un Système Informatique opérationnel très sécurisé dont la gestion est basée sur la mise en œuvre de son système de management intégré. Cependant, certains points faibles relevés sont relatifs à la gestion du réseau local et peuvent impacter sur le rendement de la structure.

Nous avons remarqué que la connexion à toutes les applications mêmes les plus sensibles repose sur le protocole d'authentification basé sur les mots de passe (PAP). L'administrateur se trouve dans l'obligation de disposer d'un mot de passe pour chaque application. Etant donné les nombreuses failles que comporte l'authentification simple avec mot de passe et la difficulté de garder en mémoire plusieurs mots de passe pour des applications différentes, nous nous

sommes intéressé à étudier les systèmes d'authentification qui existent afin de proposer une solution plus conviviale et sécuritaire pour la connexion aux applications sensibles.

CHAPITRE 2

Chapitre 2 : Etat de l'art sur l'authentification

Les applications informatiques sont conçues par les développeurs de logiciels en tenant compte des exigences de la partie demandeur. Ces développeurs sont tenus d'intégrer et d'adopter les techniques de sécurisation de code existantes. A tous les niveaux du développement d'une application web, d'interface utilisateur, de la logique métier, du contrôleur, du code de base de données, etc., il faut garder à l'esprit que la sécurité est importante. Cela représente une tâche qui peut être très difficile et conduit souvent les développeurs à des situations d'échec.

C'est pourquoi ce chapitre a pour but de décrire le concept de sécurité des applications web en particulier la notion d'authentification, les failles de sécurité existantes et les éléments de base dans la mise en place d'une solution d'authentification.

1- Définition des objectifs de sécurité

Parmi les objectifs de la sécurité informatique, on peut citer :

- L'intégrité, c'est-à-dire garantir que les données sont bien celles que l'on croit être ;
- La confidentialité, consistant à assurer que seules les personnes autorisées aient accès aux ressources échangées ;
- La disponibilité, permettant de maintenir le bon fonctionnement du système d'information ;
- La non-répudiation, permettant de garantir qu'une transaction ne peut être niée ;
- L'authentification, consistant à assurer que seules les personnes autorisées aient accès aux ressources.

Avec la popularité grandissante des réseaux et des échanges de données, de nombreux risques menacent l'atteinte de ces cinq objectifs de sécurité :

- Interruption : risque lié à la disponibilité des données
- Interception : risque lié à la confidentialité des données
- Modification : risque lié à l'intégration des données
- Fabrication risque lié à l'authenticité des données

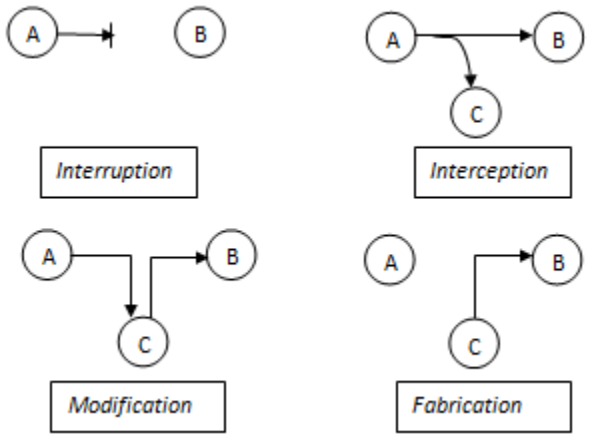


Figure 3: Types de menaces actives

Ainsi l'authentification constitue un élément majeur et fondamental de la sécurité informatique car une faille dans l'authentification ne garantit plus la fiabilité d'aucun des quatre autres principaux objectifs de la sécurité informatique.

Pour sécuriser l'accès aux données d'une entreprise par un réseau local ou internet, un système d'authentification doit être mis en place. Il peut s'agir :

- d'authentifier l'utilisateur qui accède à la ressource
- d'authentifier l'ordinateur qui accède à la ressource (réseau local)
- ou authentifier le processus d'accès à la ressource

Le terme authentification ne concerne donc pas uniquement la personne accédant à une ressource et est différente de la notion d'identification.

2- Le concept d'authentification

2-1- Définition

Pour mieux cerner la notion d'authentification, plusieurs définitions ont été proposées par certains standards souvent référencés comme sources de bons conseils sur le concept d'authentification.

ISO (2014) définit l'authentification comme un moyen pour une entité d'assurer la légitimité d'une caractéristique revendiquée.

NIST (2013) définit l'authentification comme la vérification de l'identité d'un utilisateur, processus, ou appareil, et une condition préalable pour autoriser l'accès aux ressources d'un système d'information.

ISF (2014) se réfère à l'authentification comme un élément du contrôle de l'identité et de la gestion du contrôle des accès, comme une composante de la gestion des accès.

ISACA (2012) définit l'authentification comme l'acte de vérification de l'identité d'un utilisateur et l'admissibilité de l'utilisateur d'accéder à l'information (d'autorisation).

Selon la communauté OWASP, La définition donnée par NIST semble être la plus précise et complète les autres. Mais il faut toutefois noter que, lorsque l'authentification de l'expéditeur d'un processus ou d'un ensemble de données est effectuée, l'authentification a pour but de vérifier aussi l'intégrité des données, par exemple, lorsqu'il s'agit d'authentifier la signature numérique d'un fichier.

Sur la base de ces normes et l'usage courant dans le domaine, la définition suivante est proposée :

L'authentification est la fonction de sécurité qui consiste à apporter et à contrôler la preuve de l'identité d'une personne, de l'émetteur d'un message, d'un logiciel, d'un serveur logique ou d'un équipement.

2-2- Les facteurs d'authentification

L'authentification peut se faire de multiples manières, et notamment par la vérification de :

- ce que l'entité connaît (un mot de passe, un code NIP, une phrase secrète, etc.)
- ce que l'entité détient (une carte magnétique, RFID, une clé USB, un PDA, une carte à puce, un smartphone, etc.). Soit un élément physique appelé jeton d'authentification, authentificateur ou token.
- ce que l'entité est, soit une personne physique (empreinte digitale, empreinte rétinienne, structure de la main, structure osseuse du visage ou tout autre élément biométrique)
- ce que l'entité sait faire ou fait, soit une personne physique (biométrie comportementale tel que signature manuscrite, reconnaissance de la voix, un type de calcul connu de lui seul, un comportement, etc.)

- où l'entité est, soit un endroit d'où, suite à une identification et authentification réussie, elle est autorisée (accéder à un système logique d'un endroit prescrit)

L'authentification d'un utilisateur est qualifiée de forte lorsqu'elle a recours à une combinaison d'au moins deux de ces méthodes.

2-3- Les jetons d'authentification

2-3-1- Définition

Un jeton d'authentification est un élément que l'utilisateur connaît, possède ou contrôle et qu'il peut utiliser pour corroborer sa prétention d'être la personne désignée par son nom.

- Mot de passe statique : c'est un mot ou groupe de mots mémorisé ou conservé en secret par un utilisateur, qui ne devrait être connu que de celui-ci ;
- Jeton secret préenregistré : c'est une série de questions et réponses définie par l'utilisateur pendant le processus d'inscription ;
- Jeton secret matriciel : matrice (grille ou « carte-bingo ») électronique ou imprimée permettant de générer des mots de passe au moyen d'un mécanisme de questions-réponses chaque fois qu'une authentification est nécessaire ;
- Jeton secret hors bande : Solution combinant un dispositif physique (p. ex., téléavertisseur, téléphone cellulaire, assistant numérique ou téléphone filaire) et un secret transmis au dispositif par un vérificateur chaque fois qu'une authentification est nécessaire.
- Jeton à mot de passe ponctuel : Dispositif matériel qui génère un mot de passe ponctuel (c'est-à-dire à usage unique) partagé entre l'utilisateur et le vérificateur chaque fois qu'une authentification est nécessaire ;
- Jeton biométrique : Représentation d'un attribut de l'utilisateur qui peut être mesurée quantitativement et comparée à une valeur conservée en mémoire (par exemple : empreinte digitale, empreinte rétinienne, image faciale, empreinte vocale ou signature) ;
- Jeton cryptographique logiciel : Clé cryptographique normalement conservée sur disque ou sur un autre support de stockage, qui peut être déverrouillée seulement à l'aide de données d'activation (par exemple : mot de passe) ;
- Jeton cryptographique matériel : Dispositif matériel contenant une clé cryptographique protégée, qui peut être déverrouillée seulement à l'aide de données d'activation (par exemple : mot de passe, attribut biométrique).

2-3-2- Avantages et inconvénients de quelques jetons d'authentification

Aucun de ces facteurs d'authentification n'est parfait et tous présentent des inconvénients et des faiblesses. Le tableau suivant recense les principaux avantages et inconvénients de quelques jetons d'authentification (Villacres, 2003).

Tableau1 : Avantages et inconvénients de quelques jetons

Jeton	Avantages	Inconvénients
Mot de passe statique	- peu coûteux - facile à mettre en œuvre - facile à utiliser	- vol du mot de passe - oubli du mot de passe - peu robuste (facilement devinable ou «craquable ») - partageable, et trop souvent partagé - mot de passe rejouable par une personne malveillante
Mot de passe statique stocké dans une carte magnétique activée par code PIN	- robustesse du mot de passe (possibilité de choisir un mot de passe aléatoire et comprenant des caractères spéciaux) - pas de nécessité de mémoriser le mot de passe	- vol, perte ou oubli de la carte - carte partageable - durée de vie du mot de passe souvent trop longue : nécessité de renouveler régulièrement l'enregistrement dans la carte - mot de passe rejouable
Mot de passe dynamique généré par un outil logiciel	- robustesse du mot de passe (mot de passe souvent aléatoire et comprenant des caractères spéciaux) - confort d'utilisation pour l'utilisateur (absence de mémorisation du mot de passe)	- peu de confort d'utilisation (nécessité d'utiliser un logiciel à Chaque nouvelle connexion) - protection de l'utilisation du logiciel par une personne non autorisée
Mot de passe dynamique généré par un outil matériel	- confort d'utilisation pour l'utilisateur (absence de mémorisation du mot de passe) - robustesse du mot de passe (usage unique)	- vol, perte ou oubli du générateur de mot de passe - le générateur peut se désynchroniser avec le serveur qui contrôle la vérification du mot de passe
Certificat X.509 dans le navigateur de l'ordinateur	- multi-usage - robustesse de la méthode d'authentification - confort d'utilisation pour l'utilisateur	- vol ou utilisation frauduleuse de l'ordinateur et copie de la clé privée associée au certificat.
Certificat X.509 dans un token USB	- multi-usage - robustesse de la méthode d'authentification	- vol, perte ou oubli du matériel

	- attitude similaire à la possession de clés (maison, voiture)	
Biométrie et caractéristiques de référence dans une base de données en réseau	- pas d'oubli ou de vol possible	- technologie encore immature - facilement falsifiable - Problème moins bien connu (Zimmer, 2008)
Biométrie associée à une carte magnétique	- Robustesse de la méthode d'authentification	- technologie encore immature - vol, perte ou oubli de la carte magnétique - coût élevé
Jeton secret matriciel	- confort d'utilisation pour l'utilisateur	- vol, oubli

La qualité d'une méthode d'authentification ne se mesure pas uniquement à ses avantages et inconvénients généraux ou à sa robustesse théorique face aux attaques mais avant tout à sa capacité à répondre aux besoins de sécurité de l'entreprise tout en prenant en considération le secteur d'activité dans lequel il est mis en place (son environnement) i.e. en tenant compte des risques mais aussi des contraintes techniques, organisationnelles, temporelles de même que financières et légales (Villacres, 2003).

2-4- Les menaces liées à l'authentification

Les applications Web sont exposées à une très grande variété de menaces et de scénarios d'attaque. Les paramètres d'authentification et de contrôle d'accès soumis par l'utilisateur sont souvent incorrectement pris en compte, gérés ou contrôlés. Cette situation peut entraîner des risques d'usurpation d'identité et d'accès à des fonctionnalités ou données illégitimes, et donc d'atteinte à la confidentialité ou à l'intégrité des données.

Une solution d'authentification doit être en mesure d'atténuer un ensemble de menaces à l'authentification. La présente section décrit brièvement différents types de menace visant les processus d'authentification.

2-4-1- Tentative de deviner en ligne

Une personne non autorisée se connecte au vérificateur en ligne et tente de deviner un jeton secret (par exemple : un mot de passe) avec l'intention d'usurper l'identité de l'utilisateur légitime. Dans cette rubrique on peut distinguer aussi deux types :

- **Attaque par force brute**

L'attaque par force brute est une méthode utilisée en cryptanalyse pour trouver un mot de passe ou une clé. Cette méthode est en général considérée comme la plus simple concevable. Elle permet de casser tout mot de passe en un temps fini indépendamment de la protection utilisée, mais le temps augmente avec la longueur du mot de passe.

Le principe général de l'attaque par force brute reste toujours de tester l'ensemble des mots de passe possibles, cependant l'ordre de test peut être optimisé afin d'obtenir de meilleurs rendements qu'une attaque par ordre alphabétique. Certains systèmes de mots de passe étant capables de reconnaître les tentatives d'attaque par force brute suivant les algorithmes les plus courants et de les bloquer, l'introduction d'éléments aléatoires peut masquer l'attaque.

- **Attaque par dictionnaire**

Plutôt que d'utiliser des chaînes de caractères aléatoires comme mot de passe, les utilisateurs ont tendance à utiliser des mots courants plus faciles à retenir. Dans une attaque par dictionnaire, on va tenter tous les mots présents dans de simples dictionnaires de la langue de l'utilisateur attaqué ainsi que les mots de passe faibles habituels (mots de passe par défaut des équipements, dates, immatriculations de véhicules, dictionnaires de noms d'animaux, dictionnaires des patronymes d'un pays ou d'une culture, dictionnaire des prénoms etc.).

2-4-2- Réinsertion

Forme particulière d'interception dans laquelle un attaquant prend note des informations d'une transaction d'authentification réussie, puis réinsère ces informations dans une nouvelle transaction d'authentification en vue d'obtenir des renseignements utilisateur sensibles.

2-4-3- Détournement de session

Un attaquant cherche à prendre le contrôle d'une session d'application auparavant ouverte par un utilisateur autorisé. L'attaquant tire parti du fait que les communications peuvent

être protégées au moment de l'établissement initial de la transaction d'authentification, mais non par la suite.

2-4-4- Usurpation de l'identité du vérificateur(Hameçonnage)

Un attaquant se fait passer pour un vérificateur avec l'intention d'inciter un utilisateur à divulguer des jetons secrets. L'hameçonnage, le phishing ou le filoutage est une technique utilisée par des fraudeurs pour obtenir des renseignements personnels dans le but de perpétrer une usurpation d'identité. La technique consiste à faire croire à la victime qu'elle s'adresse à un tiers de confiance (banque, administration, etc.) afin de lui soutirer des renseignements personnels : mot de passe, numéro de carte de crédit, date de naissance, etc. C'est une forme d'attaque informatique reposant sur l'ingénierie sociale. Elle peut se faire par courrier électronique, par des sites web falsifiés ou autres moyens électroniques.

2-4-5- Attaque de l'intercepteur (Man-in-the-Middle)

Un attaquant se place dans la voie de communication entre un utilisateur et un vérificateur, de manière à ce que toutes les communications passent par lui. L'attaquant peut être passif (en prenant note de l'information tout en la transmettant comme il est supposé faire) ou jouer un rôle actif (en communiquant avec l'utilisateur et avec le vérificateur tout en se faisant passer pour l'autre) en vue d'obtenir des renseignements sensibles sur l'utilisateur.

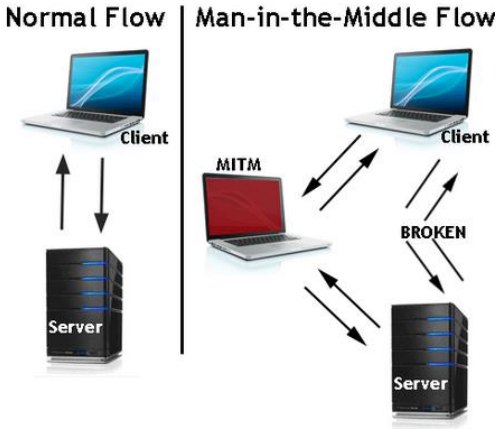


Figure 4: Attaque de l'intercepteur

L'image ci-dessus montre que l'attaquant s'insère lui-même dans le flux de trafic entre le client et le serveur. Maintenant que l'attaquant a fait intrusion dans la communication entre les deux extrémités, il peut injecter de fausses informations et intercepter les données transférées entre eux.

2-5- Les modules cryptographiques

Les jetons sélectionnés pour un système d'authentification peuvent nécessiter l'emploi de modules cryptographiques logiciels ou matériels afin de garantir la sécurité de transaction des informations et une meilleure authentification de l'utilisateur.

2-5-1- La cryptographie

Le mot cryptographie est un terme générique désignant l'ensemble des techniques permettant de chiffrer des messages, c'est-à-dire permettant de les rendre inintelligibles sans une action spécifique.

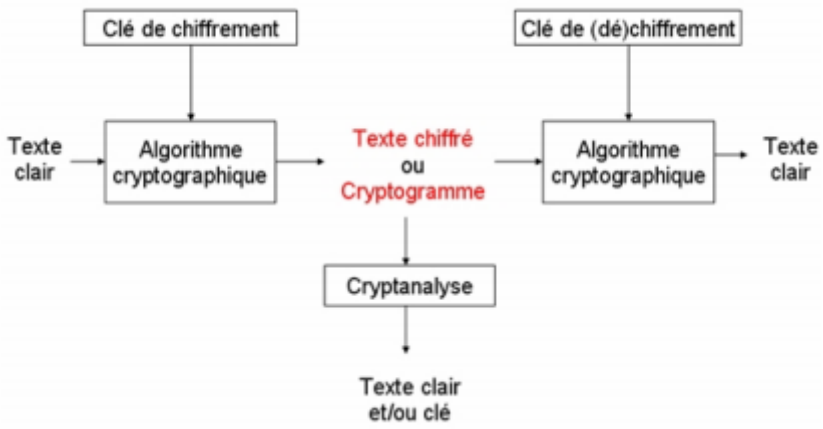


Figure 5: Protocole de chiffrement

Le chiffrement des messages est une opération qui consiste à appliquer une fonction avec des caractéristiques très particulières sur ces messages. Cette fonction dépend d'une variable appelée clé de chiffrement, qui est une suite de bits quelconques. Pour pouvoir lire le texte chiffré, il faut aussi appliquer une fonction inverse déchiffrement qui dépend aussi d'une clé de déchiffrement. Ces deux fonctions mathématiques sont appelées algorithmes cryptographiques ou algorithmes de chiffrement. La valeur de la clé de déchiffrement dépend évidemment de la valeur de la clé de chiffrement et uniquement le possesseur de la clé de déchiffrement peut déchiffrer le texte. On distingue les algorithmes de chiffrement symétrique et les algorithmes de chiffrement asymétriques.

- **Chiffrements symétriques**

Le chiffrement symétrique (aussi appelé *chiffrement à clé privée* ou *chiffrement à clé secrète*) consiste à utiliser la même clé pour le chiffrement et le déchiffrement. La valeur de cette clé doit être un secret partagé uniquement entre l'émetteur et le récepteur.

L’avantage est que les opérations de chiffrement et de déchiffrement sont rapides à exécuter sur des ordinateurs classiques

Dans les années 40, Claude Shannon démontra que pour être totalement sûr, les systèmes à clefs privées doivent utiliser des clefs d'une longueur au moins égale à celle du message à chiffrer. De plus, le chiffrement symétrique impose un canal sécurisé pour l'échange de la clé, ce qui dégrade sérieusement l'intérêt d'un tel système de chiffrement. Le principal inconvénient d'un algorithme à clefs secrètes provient de l'échange des clés.

Figure 6: chiffrement symétrique

❖ Chiffrements asymétriques

Dans le cas des systèmes symétriques, on utilise une même clé pour le chiffrement et le déchiffrement.

Le problème repose dans la transmission de la clé : il faut une clé par destinataire. Dans le cas des systèmes asymétriques, chaque personne possède 2 clés distinctes (une privée, une publique) avec impossibilité de déduire la clé privée à partir de la clé publique. De ce fait, il est possible de distribuer librement cette dernière.

Figure 7: chiffrement asymétrique

Le problème consistant à se communiquer la clé de déchiffrement n'existe plus, dans la mesure où les clés publiques peuvent être envoyées librement. Le chiffrement par clés publiques

permet donc à des personnes d'échanger des messages chiffrés sans pour autant posséder de secret en commun.

En contrepartie, tout le challenge consiste à s'assurer que la clé publique que l'on récupère, est bien celle de la personne à qui l'on souhaite faire parvenir l'information chiffrée !

❖ **Clé de session**

La distribution de clés publiques bien que simple et permettant la confidentialité et l'authentification, reste lente. L'objectif restant de protéger le contenu d'un message, une solution plus rapide est d'utiliser un système hybride permettant l'échange final d'une clé de session.

Cette solution fut proposée par Merkle en 1979. Le principe de la clé de session est simple : il consiste à générer aléatoirement une clé de session de taille raisonnable, et de chiffrer celle-ci à l'aide d'un algorithme de chiffrement à clef publique (plus exactement à l'aide de la clé publique du destinataire). Le destinataire est en mesure de déchiffrer la clé de session à l'aide de sa clé privée. Ainsi, expéditeur et destinataires sont en possession d'une clé commune dont ils sont seuls connaisseurs. Il leur est alors possible de s'envoyer des documents chiffrés à l'aide d'un algorithme de chiffrement symétrique.

Figure 8: clé de session

2-5-2- Le hachage

Il s'agit de la troisième grande famille d'algorithmes utilisés en cryptographie. Le principe est qu'un message clair de longueur quelconque doit être transformé en un message de longueur fixe inférieure à celle de départ. Le message réduit portera le nom de "Haché" ou de "Condensé" (digest en anglais). L'intérêt est d'utiliser ce condensé comme empreinte digitale

du message original afin que ce dernier soit identifié de manière univoque. Cette fonction est donc utilisée pour signer le message envoyé. Elle assure l'intégrité du message.

MD5(MD pour Message Digest) est une fonction de hachage très répandue, elle crée une empreinte de 128 bits [MD5]. SHA (Secure Hash Algorithm), autre fonction, crée des empreintes de 160 bits [SHA]. Avant d'envoyer le message, l'outil logiciel émetteur calcule l'empreinte du message, résultat d'une fonction de hachage appliquée au message. Il chiffre ensuite cette empreinte par un algorithme asymétrique avec sa clé privée. Ce résultat est appelé signature électronique. Avant l'envoi, cette signature est ajoutée au message, qui devient un message signé.

Le logiciel du destinataire qui reçoit l'ensemble déchiffre cette empreinte chiffrée avec la clé publique de l'émetteur. Puis il recalcule la fonction de hachage sur le message reçu et compare le résultat avec l'empreinte déchiffrée. Si les deux sont égaux, cela veut dire que le message n'a pas été modifié durant le transfert et que l'émetteur est authentifié.

En effet, si le message a été modifié, les 2 empreintes seront différentes. De plus, être capable de déchiffrer, avec la clé publique d'une personne, une empreinte chiffrée, prouve que cette empreinte a obligatoirement été chiffrée avec la clé privée de la personne, clé que seul possède l'émetteur. Cela authentifie donc l'émetteur. On peut rappeler qu'une des propriétés du couple clé privée - clé publique est que tout ce qui est chiffré avec une des clés peut être déchiffré avec l'autre clé et uniquement avec celle-ci.

2-5-3- Certificat

Le problème de clés publiques est qu'un pirate peut arriver à remplacer votre clé publique par la sienne, par exemple sur un annuaire. Et toutes les personnes croyant encrypter pour vous, encrypterons pour le pirate. Les systèmes à clés publiques ne garantissent donc pas que la clé est bien celle de l'utilisateur à qui elle est censée appartenir.

Les certificats servent à cela : ils garantissent que la clé appartient bien à la personne qui l'a générée. Pour cela, des organismes de certification appelés CA (Certification Authority) fournissent un certificat (moyennant finances) garantissant cela. Elles sont en charge de délivrer des certificats, de leur donner une date de validité et de révoquer les certificats en cas de problèmes de confiance.

Un certificat est l'équivalent d'une carte d'identité ou d'un passeport. Un passeport contient des informations concernant son propriétaire (nom, prénom, adresse, …), la signature manuscrite, la date de validité, ainsi qu'un tampon et une présentation (forme, couleur, papier) qui permettent de reconnaître que ce passeport n'est pas un faux, qu'il a été délivré par une autorité bien connue.

Les certificats résolvent le problème du canal sécurisé grâce à la signature de tiers de confiance.

Un certificat électronique est un ensemble de données contenant :

- au moins une clé publique
- des informations d'identification
- au moins une signature ; de ce fait, l'entité signataire est la seule autorité permettant de prêter confiance (ou non) à l'exactitude des informations du certificat.

Les certificats électroniques sont utilisés dans différentes applications informatiques dans le cadre de la sécurité des systèmes d'information pour garantir :

- la non-répudiation et l'intégrité des données avec la signature numérique
- la confidentialité des données grâce au chiffrement des données
- l'authentification d'un individu ou d'une identité non physique

3- Les protocoles utilisant des mécanismes d'authentification

Cette partie a pour but de faire le point des protocoles d'authentification existants. Nous allons nous limiter aux protocoles d'authentification d'un utilisateur par un serveur pour accéder aux ressources d'une entreprise.

3-1- Protocole d'authentification basée sur les mots de passe (PAP)

Le protocole PAP (Password Authentication Protocol) est, comme son nom l'indique, un protocole d'authentification simple par mot de passe Il permet de vérifier l'identité de l'utilisateur lors de la connexion au serveur web. Le principe du protocole PAP consiste à envoyer l'identifiant et le mot de passe en clair à travers le réseau. Si le mot de passe correspond, alors l'accès est autorisé. Une table de noms d'utilisateur et de mots de passe est stockée sur un serveur, et, lorsqu'un utilisateur s'identifie, son nom et son mot de passe sont transmis au serveur pour vérification. Toutefois, comme avec ce protocole, le mot de passe circule en clair, la sécurité des données n'est aucunement garantie.

Ainsi, le protocole PAP n'est utilisé en pratique qu'à travers un réseau sécurisé.

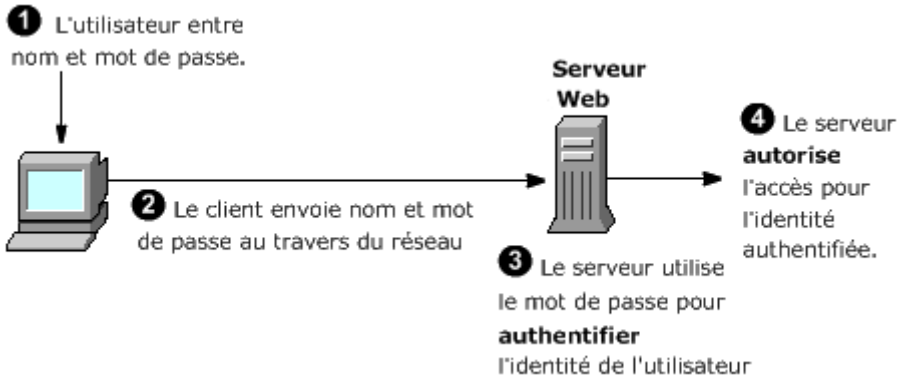


Figure 9: Principe de fonctionnement PAP

Pour ce type de mécanisme, le client (navigateur web) a déjà décidé de faire confiance au serveur, sans authentification ou sur la base d'une authentification de serveur via SSL.
Une implémentation propre ne mémorise pas les mots de passe en texte simple. À la place, il concatène le mot de passe avec une valeur aléatoire propre à chaque utilisateur (également appelée « salt ») et mémorise la valeur hachée du résultat avec le « salt ». Le salage d'un mot de passe n'augmente pas la difficulté de recherche par une attaque exhaustive sur un seul mot de passe puisque le "sel" est conservé en clair dans le fichier de mots de passe ; par contre, elle élève le coût d'une attaque simultanée sur plusieurs mots de passe, puisqu'il faut prévoir 2^t variations de chaque mot de passe (Molva & Roudier, 2011).

3-2- Protocoles cryptographiques question-réponse

Ce type de protocole a été étudié par Molva et Roudier (2011). Lorsqu'une entité (utilisateur « A ») effectue une authentification forte auprès d'une autre entité (serveur « B »), elle prouve qu'elle connaît un secret associé à son identité déclarée. Cette preuve s'appuie sur l'utilisation de techniques telles que les fonctions de hachage, ou la cryptographie symétrique ou asymétrique : on parle de protocole cryptographique question-réponse.

Un protocole question-réponse fonctionne d'après le principe suivant :

- L'entité B qui joue le rôle de vérificateur choisit de manière aléatoire une donnée, appelée question, qui est envoyée à l'entité A qui doit prouver son identité ;
- L'entité A applique à son tour à la question une opération cryptographique basée sur un secret qu'elle détient ;

Le résultat de cette opération, appelé réponse, est renvoyé à B pour fournir la preuve de l'identité de A.

Le but principal des protocoles question-réponse est d'empêcher une famille d'attaques connue sous le nom de rejeu. Le rejeu décrit la retransmission par un intrus, dans le but d'usurper l'identité du demandeur, d'une réponse qui a déjà été utilisée entre deux entités légitimes comme réponse à une nouvelle question.

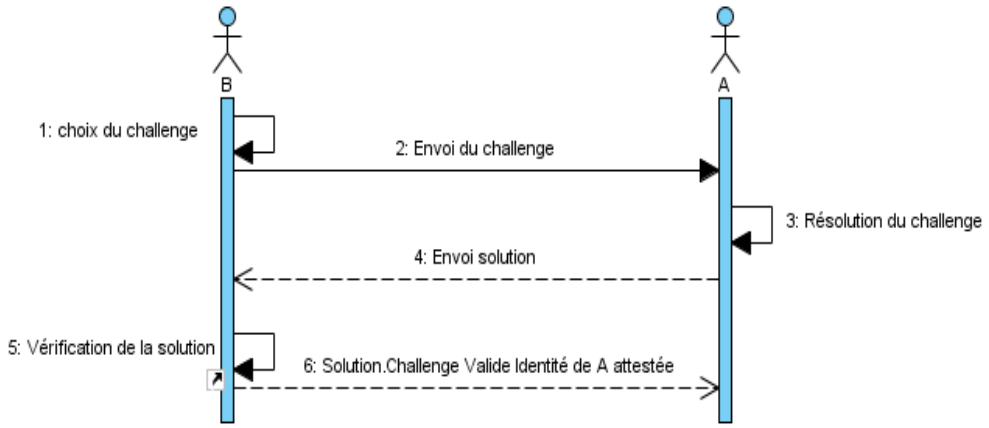


Figure 10: principe de fonctionnement protocole question-réponse

Afin d'assurer la protection contre le rejeu, la réponse calculée par le demandeur A doit être différente à chaque exécution du protocole d'authentification. La technique permettant d'obtenir un paramètre qui varie dans le temps constitue un des éléments essentiels d'un protocole d'authentification. Le second élément important d'un protocole d'authentification concerne la manière dont le demandeur A calcule la réponse correspondant à la question posée par le vérificateur B. Plusieurs variantes de calcul existent en fonction de la méthode cryptographique utilisée. Nous présentons les protocoles correspondant aux cas les plus significatifs :

3-2-1- Protocoles avec secret partagé

Dans ce type de protocoles le secret partagé, entre deux entités, demandeur et vérificateur, est au cœur des opérations cryptographiques intervenant dans le processus d'authentification. Ainsi une entité B, pour attester de l'authenticité de l'identité d'une entité A, va-t-elle appliquer une opération cryptographique basée sur un secret qu'elle partage avec A.

L'authenticité de A est prouvée par comparaison du résultat obtenu par B avec une donnée dont la connaissance est antérieure au calcul effectué.

3-2-2- Protocoles à base de clés publiques

Deux méthodes se trouvent à la base des protocoles d'authentification utilisant les algorithmes à clés publiques :

- le demandeur chiffre (ou signe) la question avec sa clé privée et la réponse résultante est déchiffrée par le vérificateur en utilisant la clé publique du demandeur ;
- le demandeur déchiffre avec sa clé privée une question qui a été chiffrée par le vérificateur en utilisant la clé publique du demandeur.

Les protocoles d'authentification à base de clés publiques sont de plus en plus répandus dans la mise en œuvre des nouveaux protocoles de communication en raison de l'absence de besoin de distribution de secret partagé et grâce au développement des infrastructures de certification qui permettent une utilisation sécurisée des clés publiques.

3-3- Protocoles basé sur des certificats

L'authentification basée sur les certificats est une étape du protocole SSL. Le serveur utilise les techniques de cryptographie à clef publique pour valider la signature et confirmer la validité du certificat.

La figure 11 décrit le fonctionnement d'une authentification client à l'aide des certificats et du protocole SSL. Pour authentifier un utilisateur auprès d'un serveur, Le client signe numériquement des données générées aléatoirement et envoie à la fois ces données signées et le certificat sur le réseau. La signature numérique associée aux données signées peut être considérée comme une preuve fournie par le client au serveur. Le serveur authentifie l'identité de l'utilisateur en se basant sur la force de cette preuve.

Contrairement au processus décrit à la Figure 9, celui de la Figure 11 nécessite d'utiliser SSL. La Figure 11 suppose également que le client possède un certificat valide qui peut être utilisé pour l'identifier auprès du serveur. L'authentification par certificat est généralement considérée comme préférable à l'authentification par mot de passe car elle est basée sur ce que l'utilisateur a (*la clef privée*) aussi bien que sur ce que l'utilisateur sait (le mot de passe qui protège cette clef privée)

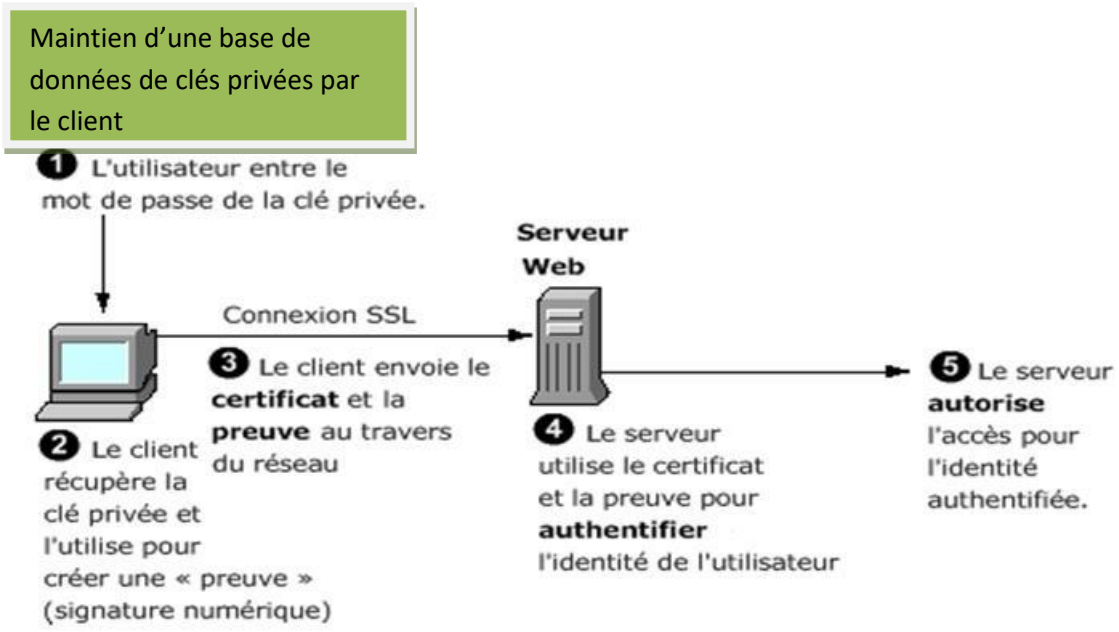


Figure 11: Protocole basé sur des certificats

Le processus peut être décrit comme suit :

1. Le logiciel client, tel que le navigateur, maintient une base de données des clefs privées correspondantes aux clefs publiques publiées avec tous les certificats émis pour ce client. Le client demande le mot de passe de cette base de données la première fois qu'il a besoin d'y accéder lors d'une session donnée, par exemple, la première fois que l'utilisateur essaie d'accéder à un serveur SSL qui requiert une authentification par certificat. Après avoir renseigné une première fois ce mot de passe, l'utilisateur n'en a plus besoin pour la durée de la session, même en accédant à d'autres serveurs SSL.
2. Le client débloque la base de données des clefs privées, récupère la clef privée du certificat de l'utilisateur et utilise cette clef privée pour signer numériquement des données générées aléatoirement dans ce but en se basant sur des entrées du client et du serveur. Ces données et la signature numérique constituent une « preuve » de la validité de la clef privée. La signature numérique peut uniquement être créée avec la clef privée et peut être validée par la clef privée associée aux données signées, ce qui est réservé à la session SSL.
3. Le client envoie le certificat de l'utilisateur et la preuve (les données générées aléatoirement signées numériquement) par le réseau.
4. Le serveur utilise le certificat et la preuve pour authentifier l'identité de l'utilisateur

5. À ce moment, le serveur peut éventuellement exécuter des tâches d'authentification supplémentaires, comme vérifier si le certificat présenté par le client est stocké dans l'entrée de l'utilisateur d'un annuaire LDAP.

A ce niveau, le processus d'authentification prend fin.

Le serveur vérifie si l'utilisateur identifié est autorisé ou non à accéder à la ressource demandée. Ce processus d'évaluation peut employer une variété de mécanismes standards d'autorisation, en utilisant éventuellement des informations présentes dans un annuaire LDAP, des bases de données d'entreprises, etc. Si le résultat de l'évaluation est positif, le serveur autorise le client à accéder à la ressource demandée.

Comme on peut le voir en comparant les Figure 9 et Figure 11, les certificats remplacent la portion de l'authentification correspondant à l'interaction entre le client et le serveur. Plutôt que de demander à l'utilisateur d'envoyer des mots de passe par le réseau à longueur de journée, l'ouverture de session unique demande une seule fois à l'utilisateur de saisir son mot de passe de base de données de clefs privée, sans l'envoyer par le réseau. Pour la suite de la session, le client présente le certificat de l'utilisateur pour authentifier l'utilisateur auprès de chaque serveur auquel il se connecte.

3-4- Protocoles utilisant un serveur d'authentification

Le concept de serveur d'authentification a pour vocation d'introduire un gestionnaire de processus d'authentification dans les différents protocoles cryptographiques question-réponse. Ce concept pallie le problème de distribution du secret partagé qui subsiste dans le cas des protocoles à base de secret partagé. Le premier protocole d'authentification utilisant un serveur a été introduit par Needham et Schroeder (1978).

Son principe de fonctionnement, tel qu'illustré sur la Figure 12, veut qu'une entité (A) voulant s'authentifier auprès d'une entité (B), demande au préalable à un centre de distribution de clé (KDC) de lui générer un secret à partager avec B. Une fois le secret généré pour la durée de cette session du protocole, le centre de distribution de clé le fait parvenir à A, avec un ticket qui est une enveloppe à l'intention de B, chiffré par sa clé Kb et contenant, en plus de l'identité de A, la nouvelle clé partagée Kab. L'entité B après avoir pris connaissance du ticket, par déchiffrage de ce dernier, envoie une question à A qui devra prouver sa connaissance de Kab pour être authentifié par B.

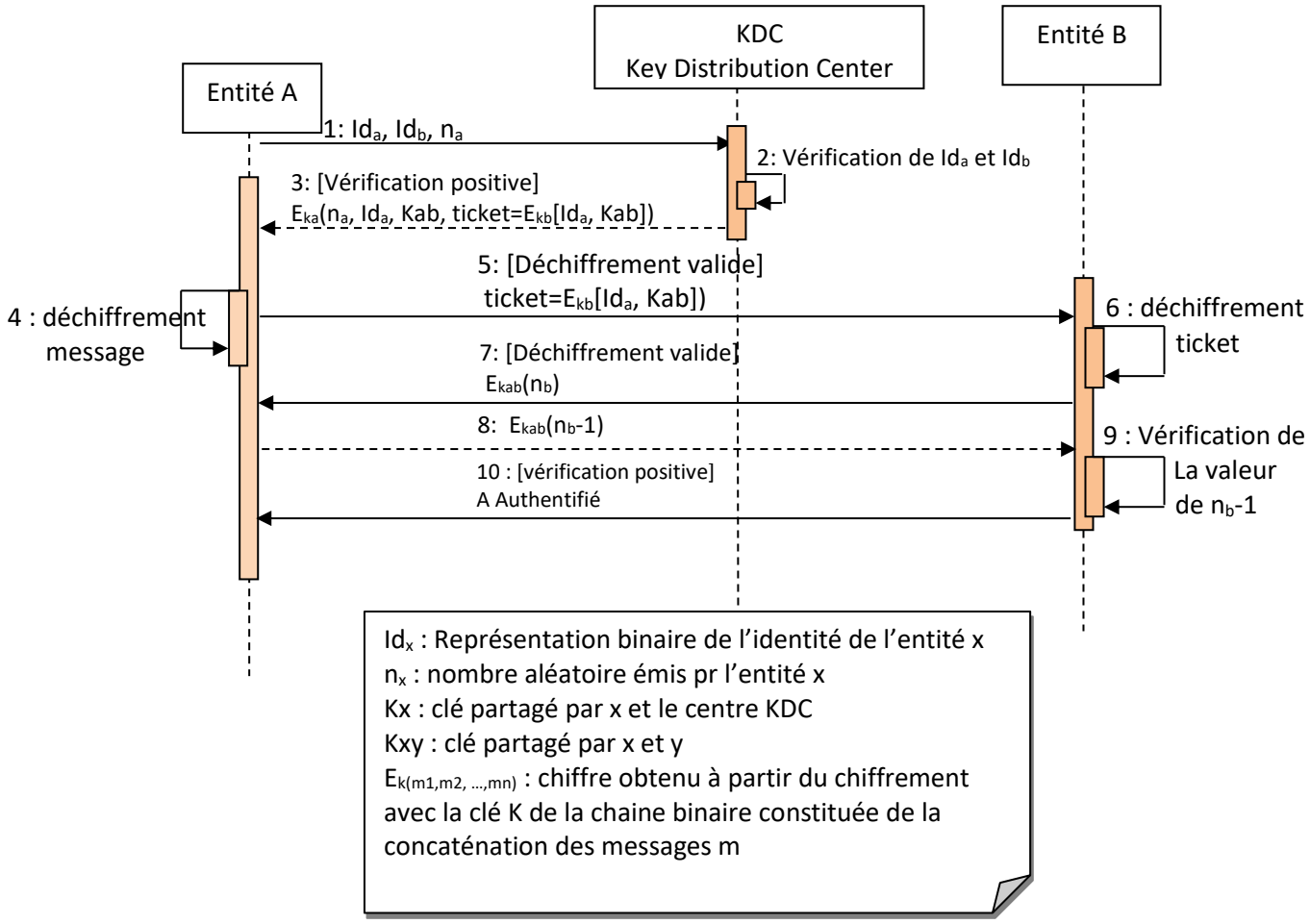


Figure 12: Protocole basé sur un serveur d'authentification

Le protocole de Nedham/Schroeder présente cependant une faiblesse : aucun élément de ce protocole ne permet de garantir que la clé Kab est nouvelle. Ainsi, dans le cas où la valeur de la clé Kab, correspondant à une exécution antérieure du protocole, aurait été interceptée par un hacker, celui-ci peut parfaitement réussir à passer pour A auprès de B en effectuant un rejeu du message 5 et en exécutant le reste du protocole en utilisant la valeur de Kab. Cette faiblesse a été corrigée par les auteurs du protocole en mettant en œuvre le système Kerberos. La principale amélioration du protocole de Kerberos consiste à inclure un cachet d'horodatage dans le ticket afin de limiter la durée de vie de la clé Kab. Plusieurs variantes de protocoles d'authentification avec un serveur ont été proposées par Otway & Rees (1987).

3-5- Dispositifs personnels

L'authentification forte repose sur la connaissance par le demandeur d'un secret qui résiste à diverses attaques concernant la méthode cryptographique utilisée par le protocole d'authentification. L'exigence concernant la taille du secret pose un problème dans le cas où l'entité qui exécute le protocole d'authentification forte est un système public qui agit au nom d'un utilisateur humain (terminal public, distributeur de billets, etc.) :

- d'une part, le secret ne peut pas être mémorisé par l'utilisateur humain puisque contrairement aux informations redondantes qui sont faciles à mémoriser pour l'homme, un secret sûr est une donnée aléatoire ou pseudo-aléatoire qui ne présente pas de redondance,
- d'autre part, un secret ne peut être stocké dans la mémoire d'un système public qui est à la portée de tous.

La solution à ce problème consiste à doter chaque utilisateur d'un dispositif personnel qui lui permet de mettre en œuvre un protocole d'authentification forte à base de secrets sûrs sans toutefois nécessiter leur mémorisation par l'utilisateur.

Deux types de dispositifs permettent de répondre à ce besoin :

- les dispositifs passifs ont la seule fonction de mémoire et
- les dispositifs actifs qui possèdent les fonctions de mémoire et de calcul qui leur permettent d'agir comme une entité indépendante dans le processus d'authentification.

4- Conclusion partielle

L'authentification est l'un des objectifs fondamentaux de sécurité informatique. Elle doit être rendue forte afin de garantir un niveau de sécurité optimum au système ou à l'application à mettre en place. Il existe plusieurs techniques d'authentification forte avec de nombreux protocoles d'authentification pour permettre aux développeurs leur mise en œuvre. Ni l'authentification par mot de passe, ni l'authentification par certificat ne répondent aux questions de sécurité soulevées par l'accès physique à l'ordinateur d'un individu ou à ses mots de passe. La cryptographie à clef publique peut uniquement vérifier qu'une clef privée utilisée pour signer des données, correspond à la clef publique présente dans un certificat. Il est de la responsabilité de l'utilisateur de protéger physiquement son ordinateur et de conserver secret le mot de passe de sa clef privée.

CHAPITRE 3

Chapitre 3 : Choix de la solution, conception et mise en œuvre

Dans le cadre de cette étude, nous nous intéressons à l'implémentation d'un système d'authentification forte basé sur le protocole standard U2F avec une clé USB de sécurité répondant au protocole. La clé de sécurité correspond ainsi au second facteur d'authentification forte.

1- Bref aperçu du protocole U2F

L'authentification à deux facteurs est devenue une norme en matière de sécurité. U2F est une norme d'authentification ouverte qui permet aux appareils mobiles, des téléphones portables et d'autres dispositifs pour accéder en toute sécurité à un certain nombre de services basés sur le Web - instantanément et avec aucun pilote ou logiciel client nécessaires. Ce protocole a été créé par Google et Yubico, avec la contribution de NXP, et est aujourd'hui hébergé par l'industrie open-authentification consortium FIDO Alliance, une organisation avec plus de 250 entreprises membres. Les spécifications techniques ont été lancé à la fin de 2014, y compris le support natif dans les comptes Google et Chrome, et ont depuis donné lieu à un écosystème florissant des fournisseurs de matériel, de logiciels et de services (FIDO, 2015) .

Ainsi, plusieurs membres ont conçu des clés de sécurité qui répondent au protocole U2F. Cela constitue un énorme gain pour les développeurs dont la tâche ne serait que de développer des applications web répondant au protocole U2F qui nécessiteront l'utilisation de la clé de sécurité dans le module de connexion.

Le protocole U2F permet aux fournisseurs de service d'offrir une forte option d'authentification à deux facteurs aux utilisateurs. La dépendance des mots de passe est réduite. Les utilisateurs doivent disposer d'une clé de sécurité U2F qui fonctionne avec une application ou site web supportant le protocole.

Les clés de sécurité sont conçues pour être utilisées dans le cadre d'une application Web, dans lequel le serveur souhaite vérifier l'identité de l'utilisateur. Il existe deux fonctions principales dans le protocole U2F : les fonctions Enregistrement et Authentification.

- Enregistrement : Au cours de cette phase, une paire de clés asymétriques est générée par l'authentificateur U2F et est seulement utilisable par l'application. L'authentificateur retourne la clé publique au serveur. La clé publique est associée au compte de l'utilisateur.

- Authentification : lors de l'authentification, la clé de sécurité teste la présence de l'utilisateur et utilise sa clé privée pour fournir une réponse. Le serveur peut vérifier que la réponse est valide, et ensuite authentifier l'utilisateur.

La Figure 13 montre deux différentes clés de sécurité fabriquées par Yubico, l'un des nombreux fournisseurs qui produisent des clés de sécurité. Chaque appareil communique via une interface USB et dispose d'un capteur tactile capacitif qui doit être touché par l'utilisateur afin d'autoriser toute opération (enregistrer ou authentifier). Les deux appareils contiennent un élément sécurisé inviolable.

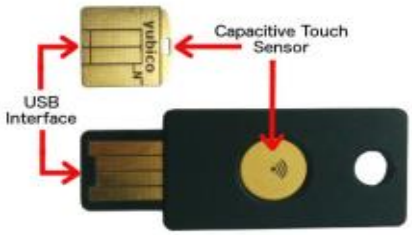


Figure 13: clés de sécurité yubico

Dans la suite du développement, nous détaillerons ces deux grandes phases du protocole qui ont servi à l'intégration du système d'authentification forte dans une application web locale. Pour l'implémentation du protocole, des spécifications ont été fournies par l'Alliance afin de mieux comprendre, d'uniformiser et de simplifier le développement du système.

2- Choix de la solution

Afin de soutenir le choix de la mise en œuvre du protocole U2F dans le cadre d'une authentification forte, une étude comparative a été sommairement faite entre la technologie U2F basée sur la clé de sécurité U2F et les autres moyens d'authentification.

2-1- OTP

Les mots de passe OTP sont des codes courts (généralement 6 à 8 chiffres) qui sont utilisés pendant un temps donné et sont envoyés à l'utilisateur par SMS ou sont générés par un dongle physique (clé électronique). Bien qu'ils offrent plus de sécurité que les mots de passe,

les jetons OTP ont un certain nombre d'inconvénients. D'abord, ils sont vulnérables aux attaques phishing et aux attaques de l'intercepteur (Railton & Kleemola, 2015). Deuxièmement, les jetons OTP qui sont livrés par les téléphones sont soumis à la disponibilité des données et du téléphone, tandis que ceux qui sont générés par les dongles (clés électroniques de protection) forcent l'utilisateur à avoir un dongle par site web. Enfin, OTP offrent une expérience utilisateur sous-optimal car l'utilisateur est souvent contraint à copier manuellement les codes d'un appareil à un autre. Les clés de sécurité U2F sont résistantes à l'hameçonnage et à l'attaque de l'intercepteur de par leur conception (FIDO, 2015).

2-2- Smartphone comme deuxième facteur

Un certain nombre d'efforts ont tenté de tirer profit du téléphone de l'utilisateur comme un deuxième facteur cryptographique, à la fois dans le milieu universitaire (Parno, Kuo, & Perrig, 2006) et dans l'industrie (Toopher, 2012). Bien que prometteur, ils font face à un certain nombre de défis : par exemple, la protection de la logique applicative des logiciels malveillants est difficile sur une plate-forme informatique à usage général (Lang, 2016). En outre, le téléphone d'un utilisateur ne peut pas toujours être accessible : le téléphone peut ne pas avoir une connexion de données ou la batterie peut être épuisée. Les clés de sécurité ne nécessitent pas de piles et ont généralement un élément sécurisé inviolable dédié (FIDO, 2015).

2-3- Les cartes à puces

Les clés de sécurité disposent d'un schéma d'authentification semblable à celui des cartes à puces. Cependant, ces derniers exigent un lecteur spécifique contrairement aux clés de sécurité et/ou l'installation préalable d'un logiciel pilote.

2-4- Les certificats de client SSL

Ils sont utilisés pour identifier des clients auprès de serveurs via SSL (authentification client). Pour certaines applications sensibles, on préfère utiliser un système d'authentification forte par certificats, plutôt que des couples login/mot de passe. C'est de plus en plus le cas avec certaines applications Web ou encore des Web Services devenant des points d'entrées critiques dans le SI des clients. Le client à la connexion va vérifier le certificat présenté par le serveur (notamment l'autorité de certification qui signe les certificats) et inversement le serveur va vérifier que le certificat du client est valide et autorisé. Malheureusement, les implémentations

actuelles de certificats clients SSL ont une mauvaise expérience utilisateur (Lang, 2016). Selon l'auteur, en règle générale, lorsque les serveurs web demandent que les navigateurs génèrent un certificat client SSL, les navigateurs affichent une boîte de dialogue où l'utilisateur doit choisir le certificat de chiffrement (méthode de chiffrement) et la longueur de la clé, un détail cryptographique qui est inconnu et déroutant pour la plupart des utilisateurs. Lorsque les serveurs web demandent au navigateur de fournir un certificat, l'utilisateur est invité à sélectionner le certificat client à utiliser ; accidentellement le choix d'un mauvais certificat peut entraîner la divulgation de l'identité de l'utilisateur à travers les sites. Cependant, certaines approches d'authentification peuvent combiner plusieurs éléments par exemple l'utilisation d'un certificat client SSL et d'une carte à puce.

2-5- Avantages du dispositif de sécurité U2F

Selon FIDO (2015), le dispositif de sécurité U2F présente plusieurs avantages dans le système d'authentification forte. On peut citer entre autres :

Sécurité renforcée : le dispositif garantit une authentification forte à deux facteurs, en utilisant une clé publique cryptographique et un support natif dans le navigateur (en commençant par Chrome). Il protège contre le phishing, le détournement de session, l'attaque par intercepteur, et les attaques de logiciels malveillants.

Facile à utiliser : le dispositif permet l'authentification instantanée à un certain nombre de services sans la nécessité d'installer un pilote.

Forte protection de la vie privée : le dispositif permet aux utilisateurs de choisir, de posséder et de contrôler leur identité en ligne sécurisée. Chaque utilisateur peut également choisir d'avoir des identités multiples, y compris anonymes (pas d'informations personnelles associées à l'identité). Le dispositif U2F génère une nouvelle paire de clés pour chaque service, la clé publique est stockée uniquement sur le service spécifique où il se connecte. Avec cette approche, aucun secret n'est partagé entre les fournisseurs de services, et même des appareils U2F à faible coût peuvent supporter plusieurs services.

Choix multiples : le dispositif est conçu pour les téléphones et les ordinateurs avec de nombreuses modalités d'authentification (téléphone portable, lecteur d'empreintes digitales, etc.) et présente différentes méthodes de communication (USB, NFC, Bluetooth).

Interopérable : le protocole U2F est une norme ouverte soutenue par des services Internet et financiers, y compris Google, Bank of America et 170 entreprises de l'Alliance FIDO. U2F permet à chaque fournisseur de services d'être son propre fournisseur d'identité, ou éventuellement, aux utilisateurs de s'authentifier par un fournisseur de services fédéré.

Rentable : les utilisateurs peuvent aujourd'hui choisir parmi une gamme d'appareils U2F à faible coût provenant de plusieurs fournisseurs, disponibles sur Amazon et d'autres magasins de détails à travers le monde.

Facile pour les développeurs : le système d'authentification forte utilisant le protocole U2F est facile à implémenter par les développeurs dans leur site Web grâce à des API simples ou des librairies.

Malgré ses énormes atouts, les clés de sécurité U2F restent très récentes dans le monde informatique et ne sont pas encore vendues dans la sphère africaine. Contrairement aux autres technologies qui ne sont pas normalisées et restent une propriété industrielle de leurs concepteurs, le protocole u2f est une norme d'authentification ouverte et permet d'assurer une sécurité plus efficace.

3- Conception du protocole U2F

Dans cette partie, nous détaillons les deux grandes phases du protocole U2F et quelques concepts qui nous ont servi à implémenter le module d'authentification forte. De plus amples informations sur le protocole peuvent être obtenues grâce à FIDO (2015).

3-1- La phase d'enregistrement

Pendant cette phase, après avoir validé le nom de l'utilisateur et son mot de passe attribués par l'administrateur, le serveur web génère un challenge aléatoire. Le navigateur de l'utilisateur construit une donnée nommée « ClientData » en concaténant le challenge, ID du canal SSL, le nom du serveur, le port, le type de requête (authentification ou enregistrement). Le navigateur envoie le paramètre de l'application (dans notre cas, l'URL de l'application) et le hash de la donnée ClientData à l'authentificateur USB. En réponse, l'authentificateur USB génère une nouvelle paire de clés ainsi qu'un conteneur de clés « key handle ». Le conteneur de clé stocke les clés cryptographiques asymétriques et est encrypté par une clé connue seulement de l'authentificateur. L'authentificateur USB associe la paire de clé à l'URL de

l'application web et retourne la clé publique générée, le conteneur de clé, un certificat et une signature de l'ensemble : l'URL de l'application web, le hash du ClientData, la clé publique et le conteneur de clé. Le navigateur web fait suivre cette donnée ainsi que le ClientData à l'application web. Cette dernière vérifie la signature et associe la clé publique et le conteneur de clé au compte de l'utilisateur. Cette phase est représentée par la Figure 14.

3-2- La phase d'authentification

Durant l'authentification, l'application web demande que l'authentificateur U2F se serve d'un authentificateur particulier qui a été précédemment enregistré par un utilisateur. Spécifiquement, l'application envoie le conteneur de clé et le challenge au navigateur. Ce dernier génère le ClientData et envoie son hash ainsi que le conteneur de clés et l'url de l'application à l'authentificateur U2F. Si l'authentificateur ne reconnaît pas le conteneur de clés ou n'est pas d'accord qu'il soit associé à l'application web qui a demandé la signature, la demande est rejetée. Autrement il signe le ClientData. La clé de sécurité signe deux attributs supplémentaires : si un test de la présence de l'utilisateur (TUP) est réussi, et une valeur de compteur. La valeur du compteur est un compteur 32 bits qui est incrémenté à chaque signature que l'authentificateur effectue. Le compteur permet au serveur de détecter le clonage potentiel d'une clé de sécurité, par exemple, lorsque la valeur du compteur semble diminuer d'une signature à l'autre.

Le navigateur transmet la signature, ainsi que la valeur de compteur au serveur. Le serveur vérifie ensuite la signature avec la clé publique qu'il a enregistrée et authentifie l'utilisateur si la signature correspond. Cette phase est représentée par la Figure 15.

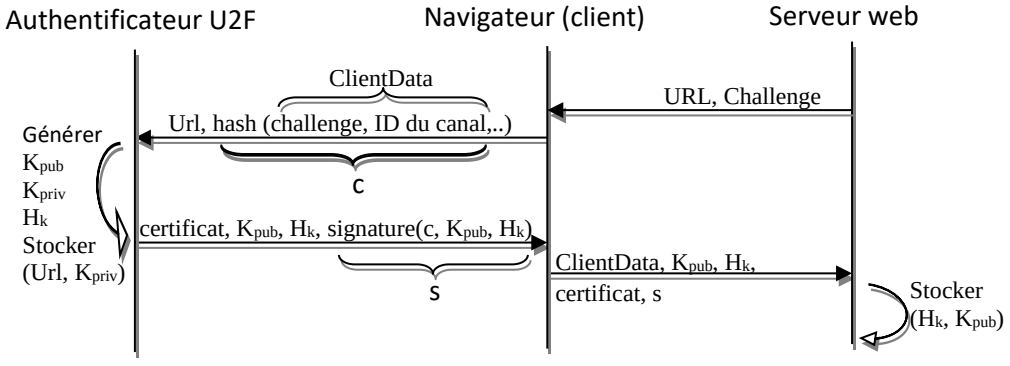


Figure 14: Processus d'enregistrement

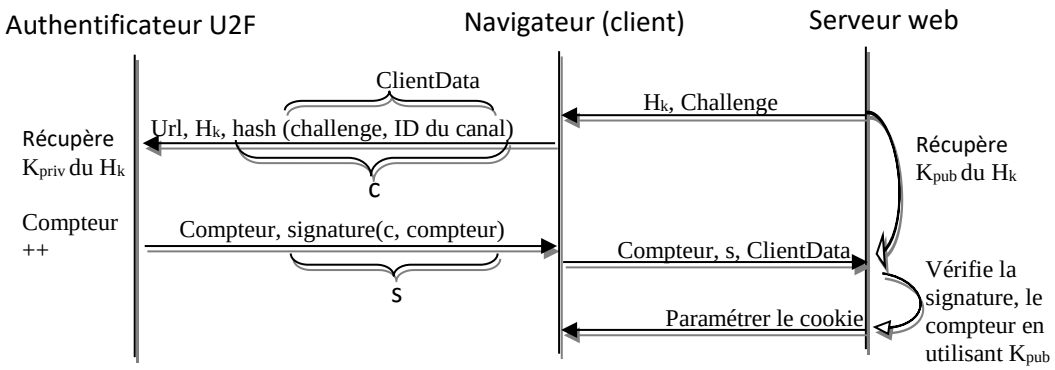


Figure 15: Processus d'authentification

3-3- Attestation du certificat d'authentificateur

Chaque authentificateur doit fournir un certificat lors de l'inscription. Cela permet à des serveurs d'autoriser l'utilisation d'un authentificateur particulier (par exemple, si les serveurs font confiance qu'à certains fournisseurs de clés de sécurité). Ainsi, toutes les clés de sécurité fabriquées par un fournisseur possèdent le même certificat. Ce qui permet de pouvoir révoquer tous les authentificateurs d'un fournisseur et d'autoriser ceux des fournisseurs voulus.

3-4- ClientData

C'est une donnée qui contient le challenge fourni par le serveur associé à aperçu de la connexion du navigateur au serveur. Spécifiquement, elle inclut le type de requête, le challenge et si possible l'ID du canal TLS de la connexion (Popov, 2015). Associer l'ID du canal SSL permet au serveur de détecter la présence de l'homme de milieu TLS. Quand le serveur reçoit un ID du canal TLS signé, il le compare avec l'ID du canal TLS qu'il observe dans la couche de transport TLS. S'ils diffèrent le serveur atteste la présence de l'intercepteur TLS et interrompt la connexion

3-5- Le test de la présence de l'utilisateur (TUP)

Il permet de vérifier si un être humain est présent lors de l'exécution de la commande. Cela sert à deux fins : en premier, il fournit un mécanisme pour confirmation des commandes de l'utilisateur humaine. Deuxio, il permet aux applications Web de mettre en œuvre une politique basée sur ce contrôle, par exemple les transactions pour un montant supérieur à 1000$

nécessitent confirmation". La mise en œuvre du TUP est du ressort du fabricant de l'authentificateur.

4- Mise en œuvre du système d'authentification forte basé sur le protocole U2F

4-1- Fonctionnalités du système

Nous avons développé et intégré un module d'authentification forte avec la clé de sécurité U2F en se servant des librairies open source disponibles. Après avoir étudié les fonctions existantes dans le protocole U2F à savoir les fonctions d'enregistrement et d'authentification de l'utilisateur, nous les avons mises en œuvre en vue de les intégrer dans une application web.

Dans notre cas d'étude, il s'agit d'une application web professionnel dont la connexion doit nécessiter non seulement un nom d'utilisateur et un mot de passe fourni initialement par l'administrateur mais aussi la possession d'un authentificateur U2F.

Le système d'authentification forte doit tenir compte des spécifications fonctionnelles suivantes :

- **Etapes de l'enregistrement**

La première connexion de l'utilisateur se déroule comme suit :

- Création d'un nom d'utilisateur et d'un mot de passe par l'administrateur ;
- Saisie du nom d'utilisateur et du mot de passe par l'utilisateur ;
- Appui sur le bouton « Connexion » ;
- Invitation à appuyer sur l'authentificateur U2F (la clé se met à clignoter) ;
- Appui sur la clé (la clé envoie les informations nécessaires pour l'enregistrement) ;
- Message de confirmation de l'enregistrement.

- **Etapes d'authentification**
  - Saisie du nom d'utilisateur et du mot de passe
  - Appui sur le bouton « Connexion » ;
  - Invitation à appuyer sur l'authentificateur U2F (la clé se met à clignoter) ;
  - Appui sur la clé (la clé envoie les informations nécessaire pour l'authentification) ;
  - Message de confirmation de la réussite de l'authentification (la page d'accueil de l'application test s'affiche) ;
  - Changement du mot de passe par défaut de l'utilisateur (au choix).

Ainsi, aucune connexion à l'application web ne peut se faire sans l'enregistrement au préalable de l'utilisateur par l'administrateur et sans la possession d'un authentificateur U2F enregistré.

La perte de l'authentificateur USB ou sa révocation est gérée par l'administrateur. Ce dernier désactive la clé perdue de l'utilisateur pour lui permettre de se réenregistrer avec une nouvelle clé U2F tout en récupérant son ancien profil.

Comme besoins non fonctionnels du module d'authentification, nous pouvons énumérer les suivants :

- Le code du module d'authentification doit être clair pour permettre des futures évolutions et améliorations ;
- Ce module doit offrir une interface conviviale et facile à utiliser ;
- Ce module doit véritablement respecter la confidentialité, l'intégrité et l'authenticité des données.

4-2- Matériels et outils de développement du système

4-2-1- Authentificateur U2F utilisé

La connexion à l'application web nécessite l'utilisation d'une clé de sécurité USB répondant au protocole U2F. Nous avons opté pour l'achat d'une clé de sécurité du fournisseur Yubico dont les offres sont moins chères (FIDO, 2015). Nous avons donc commandé cette clé qui ne coûte que 18 euros TTC soit environ 12000FCFA. Cette clé est utilisable pour tous les services qui supportent le protocole U2F. Ces clés ne sont pas reconnues comme un support de stockage de masse mais comme un clavier. Elles font appel aux spécifications et aux pilotes génériques HID (Human Interface Device class) et fonctionnent ainsi sur presque n'importe quel ordinateur.

4-2-2- Méthodes cryptographiques

Pour toutes les opérations de signature, l'algorithme ECDSA (***Elliptic Curve Digital Signature Algorithm)*** sur la courbe NIST P-256 est utilisé (FIDO, 2015). Pour toutes les opérations de hachage, nous avons choisi SHA-256. Le choix de l'algorithme de hachage et la courbe a été faite en raison de la préférence qui lui est accordée par l'alliance FIDO.

L’algorithme ECDSA est un algorithme de signature numérique à clé publique standardisé par NIST. Il permet donc la génération de la paire de clés, la signature des données et la vérification. Cet algorithme fait appel à la cryptographie sur les courbes elliptiques.

Les avantages d’ECDSA sur DSA et RSA sont des longueurs de clés plus courtes et des opérations de signature et de chiffrement plus rapides. En pratique l'ECDSA repose souvent sur des courbes recommandées par des organisations comme la NIST ou Certicom. Pour le cas des clés de sécurité Yubico, ce sont les paramètres de la courbe NIST P-256 qui sont utilisés.

Pour sécuriser les communications entre le serveur et le client, nous avons utilisé la librairie de chiffrement openSSL version 1.0.0 pour implémenter le protocole TLS. C’est cette librairie qui a permis la mise en œuvre de la fonction de hachage et de chiffrement.

4-2-3- Le type de navigateur

Nous avons utilisé les extensions du navigateur Chrome qui est le premier navigateur à prendre en charge l’U2F. Seules les versions supérieures ou égales à 40 comportent l’extension U2F.

4-2-4- Serveur web

Nous avons installé la plate-forme de développement Wamp Server incluant donc le serveur Apache, le langage PHP et le système de gestion de base de données MySQL. Nous avons choisi Wamp Server car nous voulons tester le système d’authentification sur un serveur web local, le serveur de base de données n’étant pas hébergé.

4-2-5- Environnement de développement

L’environnement de développement intégré utilisé est NETBEANS sous le système d’exploitation Windows 8.

4-2-6- Langage de développement

Le langage de développement sera principalement du côté serveur du PHP et du côté client du JavaScript.

Nous avons choisi JavaScript car il est très courant. Il est renforcé avec la mise en œuvre des feuilles de style (CSS) et des méthodes AJAX permettant d’améliorer l’aspect et le confort d’utilisation.

PHP est très populaire et permet un développement plus rapide.

4-2-7- Les librairies utilisées

Le site de Yubico propose plusieurs librairies côté serveur et côté client développées en plusieurs langages. Nous avons exploitées celles écrites en PHP et JavaScript pour mettre en œuvre les procédures d'enregistrement et d'authentification.

4-3- Procédure d'implémentation

4-3-1- Installation d'un certificat serveur SSL

Pour assurer la protection des échanges entre le serveur et l'utilisateur nous avons jugé nécessaire de sécuriser le serveur web en mettant en œuvre un certificat SSL du serveur WAMP en utilisant OpenSSL. Les étapes dans la mise en place sont détaillées dans (Chincoun, 2014). La connexion sécurisée de l'application par le protocole U2F nécessite donc l'activation de l'extension OpenSSL de PHP.

4-3-2- Mise en œuvre des fonctions

4-3-2-1- Côté serveur

Quatre fonctions de base sont clairement identifiées du côté du serveur web dans le développement du module et ont été codées en langage PHP. Il s'agit de :

- Pour la phase d'enregistrement : *getRegisterData( ), doRegister( )*
- Pour la phase d'authentification : *getAuthenticateData( ), doAuthenticate( ).*

✓ **getRegisterData( )**

Cette fonction est appelée pour obtenir une demande d'enregistrement à envoyer à l'utilisateur. Elle retourne un tableau d'une demande d'enregistrement et de demande de signature. Elle prend en paramètres la liste des enregistrements en cours pour empêcher l'utilisateur de s'enregistrer plusieurs fois avec le même authentificateur U2F.

Le tableau de demande d'enregistrement est composé de la version du protocole, du challenge et de l'url de l'application web.

Le tableau de demande de signature est composé des éléments renvoyés par getAuthenticateData afin d'identifier les authentificateurs U2F déjà enregistrés.

```
public function getRegisterData(array $registrations = array())
   {
      $challenge = $this->createChallenge();
      $request = new RegisterRequest($challenge, $this->appId);
      $signs = $this->getAuthenticateData($registrations);
      return array($request, $signs);
   }
```

Afin de vérifier s'il existe déjà une ligne d'enregistrement correspondant à l'utilisateur qui veut s'enregistrer, on insère donc en paramètre *getEnreg($user->id).*

Pendant cette phase, le navigateur contacte l'application web qui génère un challenge aléatoire par la fonction *createChallenge( ).* Le challenge est une chaine de caractères de 32 bits qui est obtenu aléatoirement en utilisant la fonction *openssl_random_pseudo_bytes (32, $crypto_strong)* de PHP.

- ✓ **doRegister( )**

Cette fonction prend en entrées un tableau de demande d'enregistrement, la réponse de l'utilisateur, le certificat et retourne une ligne d'enregistrement effectué. Elle est appelée pour vérifier et décompresser la réponse envoyée par l'utilisateur à travers le navigateur.

La réponse est un message provenant du navigateur incluant le message de réponse de l'authentificateur U2F après génération de la paire de clés suite au message de demande d'enregistrement. Le message de réponse de l'authentificateur U2F, après activation de l'authentificateur U2F, est une enveloppe de l'ensemble clé publique, conteneur de clé (key handle H_k) et certificat.

Cette réponse peut donc être un échec si l'authentificateur U2F n'est pas activé.

Plusieurs tests de vérifications sont effectués parmi lesquels on peut citer :

- la comparaison entre le challenge issu de la demande d'enregistrement et le challenge contenu dans le clientData de la réponse

- la vérification de la signature par la fonction *openssl_verify( ).*

```
function doRegister($request, $response, $includeCert = true){
…
$registration->publicKey
$registration->keyHandle
$registration->certificate
$dataToVerify  = chr(0);
      $dataToVerify .= hash('sha256', $request->appId, true);
      $dataToVerify .= hash('sha256', $clientData, true);
      $dataToVerify .= $kh;
      $dataToVerify .= $pubKey;

      if(openssl_verify($dataToVerify, $signature, $pemCert, 'sha256') === 1) {
         return $registration; // La clé publique étant extraite du certificat $pemCert
      }                              // la signature provient de la réponse de l'authentificateur
…
}
```

Le serveur vérifie donc la signature d'enregistrement et que les données du client correspond à son point de vue à la demande d'enregistrement. Il vérifie aussi si le certificat est conforme à ses exigences. En supposant que tous les paramètres en jeu sont jugés acceptables, le serveur stocke le conteneur de clé H_k et la clé publique Kpub pour le compte de l'utilisateur.

Cet enregistrement est associé au compte de l'utilisateur par une fonction définie par « ajoutEnreg » pour le stockage dans la base de données.

ajoutEnreg ($user->id, $reg); // $reg étant la sortie de la fonction doRegister( )

- ✓ **getAuthenticateData( )**

Cette fonction est appelée pour obtenir une demande d'authentification. Elle prend en paramètre un tableau d'enregistrement et crée un tableau de demande de signature contenant l'url de l'application web, le conteneur de clé et un challenge créé.

```
public function getAuthenticateData(array $registrations)
   {
      $sigs = array(); // demande de signature
      foreach ($registrations as $reg) { // pour chaque ligne d'enregistrement

         $sig = new SignRequest();
         $sig->appId = $this->appId;
         $sig->keyHandle = $reg->keyHandle;
         $sig->challenge = $this->createChallenge();
         $sigs[ ] = $sig;
      }
      return $sigs;
   }
```

Cette fonction est appelée pour vérifier la signature de la réponse d'authentification avec la clé publique obtenue durant l'enregistrement. Elle prend en paramètres un tableau de demande d'authentification, un tableau d'enregistrement en cours, et la réponse d'authentification du client. Elle retourne un tableau d'enregistrement contenant un compteur qui se met à jour à chaque authentification.

```
public function doAuthenticate(array $requests, array $registrations, $response){
        $signData = $this->base64u_decode($response->signatureData);
        $dataToVerify  = hash('sha256', $req->appId, true);
        $dataToVerify .= substr($signData, 0, 5);
        $dataToVerify .= hash('sha256', $clientData, true);
        $signature = substr($signData, 5);

        if(openssl_verify($dataToVerify, $signature, $pemKey, 'sha256') === 1) {
           $ctr = unpack("Nctr", substr($signData, 1, 4));
           $counter = $ctr['ctr'];
                    if($counter > $reg->counter){ //le serveur vérifie que le compteur a
augmenté
                       $reg->counter = $counter;
                         return $reg;
                     }
          }
}
```

La réponse d'authentification du client comporte le message de réponse de l'authentificateur (SignatureData) ainsi que le clientData et le conteneur de clé. La réponse SignatureData inclut le compteur et la signature de l'ensemble clientData et compteur.

Le serveur vérifie la signature avec la clé publique Kpub qu'il a stockée dans H_k, et si la valeur du compteur a augmenté. Le serveur vérifie si les données clientData correspondent à celles de la demande d'authentification. Il vérifie aussi si le certificat est conforme à ses exigences. En supposant que tous les paramètres en jeu sont jugés acceptables, l'utilisateur est authentifié.

4-3-2-2- Côté client

Le rôle principal du client (navigateur) est d'être l'interface entre le serveur et l'authentificateur U2F. Deux fonctions principales sont utilisées côté client :

✓ **u2f.Register( )** : Phase d'enregistrement

Elle prend en paramètres la demande d'enregistrement incluant le challenge et une liste de conteneur de clé ($H_{k)}$ déjà enregistrées. La liste des H_k déjà enregistrées permet au navigateur d'éviter un double enregistrement de la même clé de sécurité. Si le navigateur détecte un authentificateur U2F admissible, il envoie un message d'enregistrement à la clé de sécurité.

Cette dernière génère une paire de clés (kpub, kpriv) associée à l'url de l'application. Ensuite, le navigateur envoie le message de réponse de la clé de sécurité, ainsi que la donnée ClientData au serveur. Cette dernière contient l'identité du canal de connexion entre le serveur et le client à travers une fonction spécifique au navigateur chrome « chrome.runtime.connect () ».

Cette identité permet de s'assurer que les points d'extrémités du canal sécurisé entre le serveur et le client de la couche réseau sont les mêmes que ceux de la couche transport par authentification des liaisons.

✓ **u2f.Sign( )** : Phase d'authentification

Le serveur web demande une signature à partir de la clé de sécurité à travers le navigateur par la fonction *u2f.Sign( ).* Les paramètres de la signature sont le challenge, et tout conteneur de clés H_k enregistré de l'utilisateur. Le navigateur recherche alors les authentificateurs U2F disponibles. Pour chaque appareil détecté, le navigateur envoie un message signé, décrit plus haut. Après avoir reçu la réponse d'authentification de la clé de sécurité, le navigateur fournit la signature au serveur, ainsi que les données clientData et le conteneur de clé H_k.

En résumé, ces différentes fonctions implémentées décrivent les principaux rôles joués par le serveur et le client dans le système d'authentification forte mis en place. Le programme principal de connexion à l'application web est un ordonnancement des différentes fonctions évoquées plus haut tout en suivant les spécifications du protocole établies en mai 2015. La seule modification effectuée dans la base de données existante est la création d'une nouvelle table dénommée « enregistrement » qui est reliée à la table « Utilisateurs » existante. La table des enregistrements a un attribut « compteur » qui correspond à -1 lors d'une opération d'enregistrement et après authentification la valeur du compteur correspond au nombre d'authentifications effectuées par la clé de sécurité (voir code principal en annexe).

5- Présentation des résultats

La page de connexion à l'application test se présente comme suit :

Figure 16: Page de connexion de l'application test

L'administrateur décide de créer un utilisateur à partir de l'application web. Il s'authentifie et accède à la page suivante (Figure 17) qu'il remplit et l'enregistre.

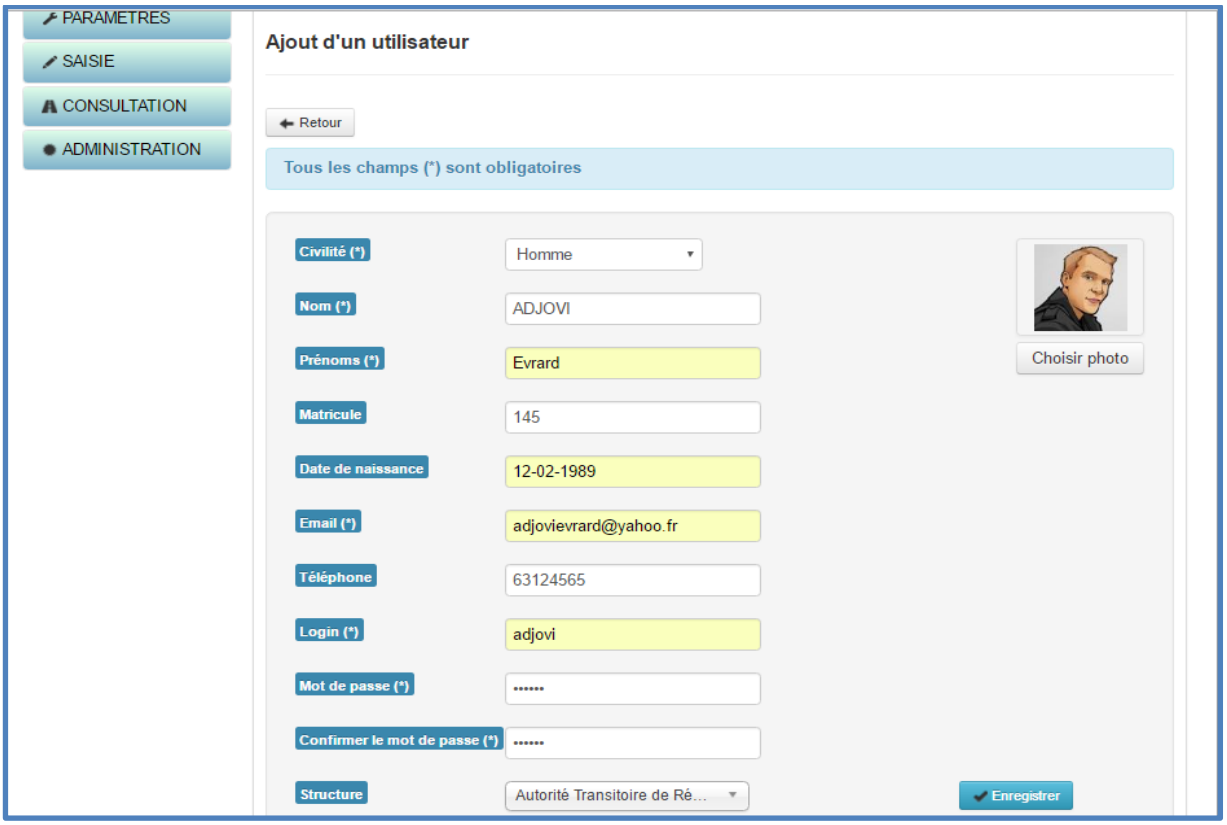


Figure 17: Ajout d'un utilisateur

Après l'enregistrement primaire de l'utilisateur, ce dernier utilise le nom d'utilisateur et le mot de passe par défaut par lequel l'administrateur l'a enregistré pour véritablement s'enregistrer dans la base avec la clé de sécurité afin de pouvoir accéder aux ressources auxquelles il a droit. Si le serveur reconnaît ses paramètres, la clé de sécurité clignotera et reste en attente d'activation (Figure 18).

Figure 18: Tentative d'enregistrement de l'utilisateur avec la clé Yubico

Après avoir appuyé le bouton de la clé qui clignote, l'utilisateur est finalement enregistré avec la clé dans la table « enregistrements » de la base de données (Figure 19 et Figure 24). Ainsi, le

serveur stocke dans la base la clé publique et le key handle associés à l'appliation test et à l'utilisateur qui s'est enregistré.

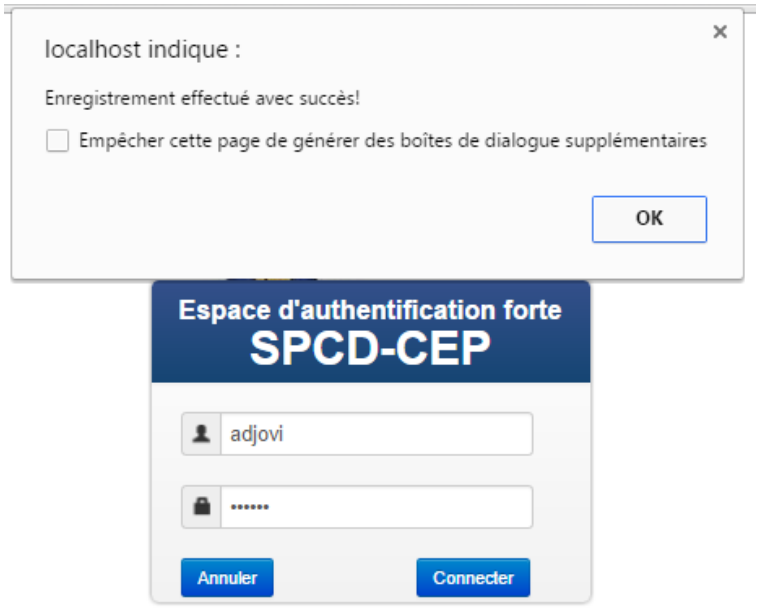


Figure 19: Enregistrement effectué avec succès

Après le succès de l'enregistrement, il est autorisé à s'authentifier avec la clé pour accéder aux ressources (Figure 20, Figure 21 et Figure 22).

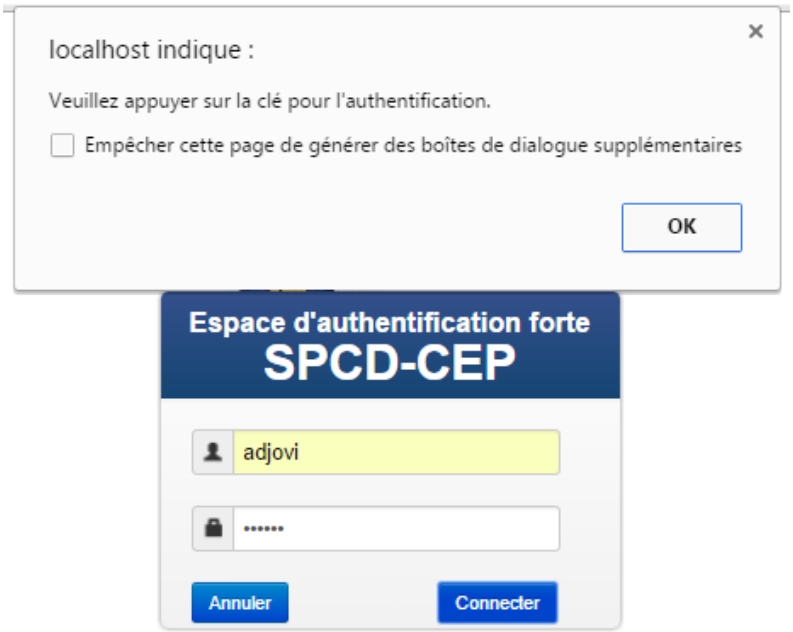


Figure 20: Tentative d'authentification de l'utilisateur

Le serveur vérifie la signature du clientData et du numéro du compteur avec la clé publique qu'il a enregistrée et authentifie l'utilisateur si la signature correspond.

Figure 21: Authentification réussie avec la clé

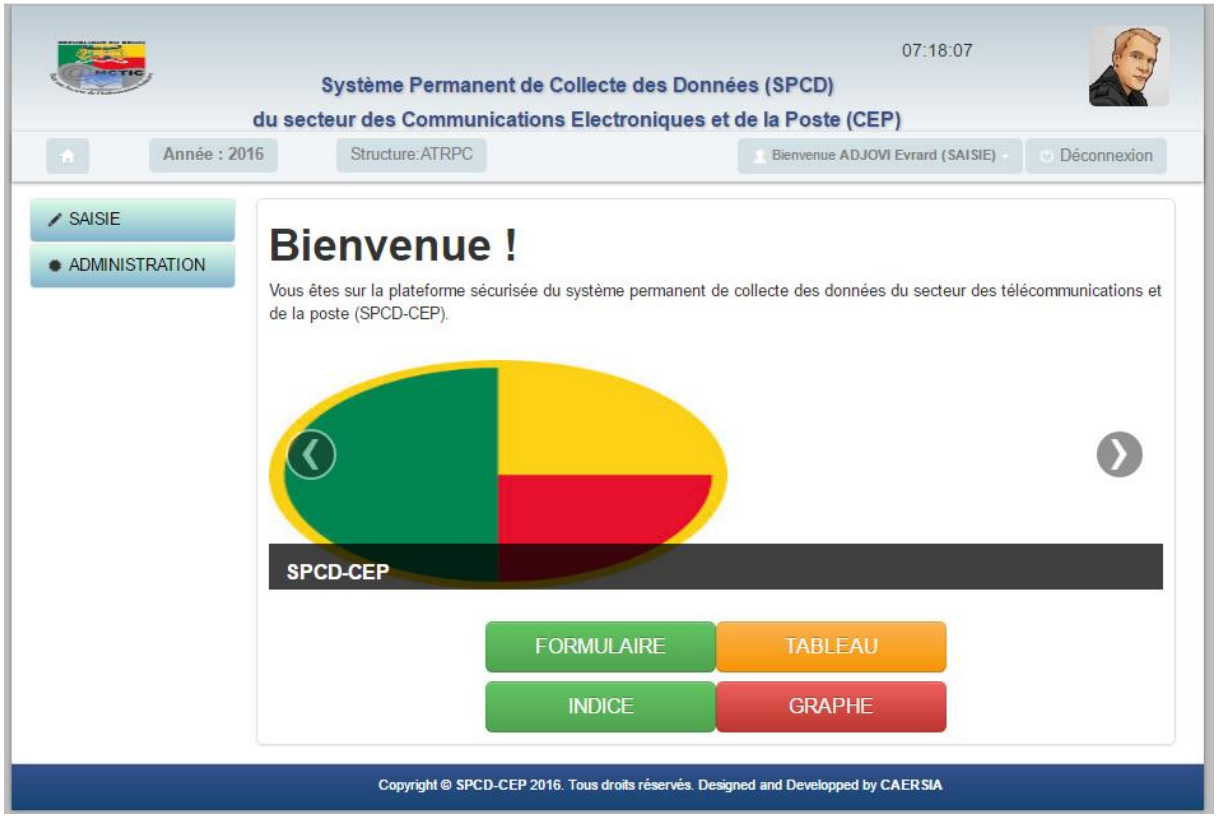


Figure 22: session de l'utilisateur authentifié

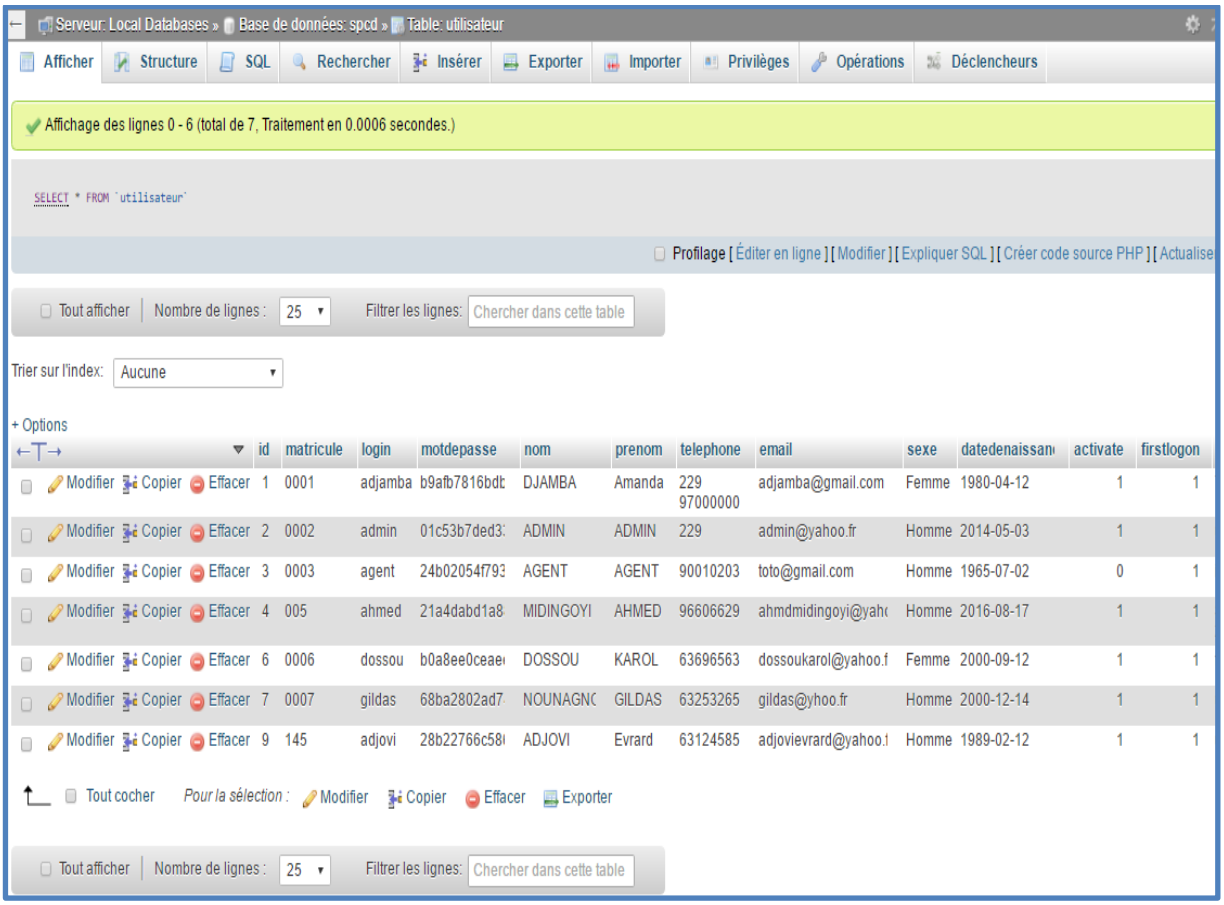


Figure 23: Table utilisateurs dans la base de données

La figure 23 présente la structure de la table contenant les informations des utilisateurs de l'application test. Cette table est constituée du champ « id » qui est le numéro d'incrémentation automatique, des informations standard de l'utilisateur (le numéro de matricule, le nom, le prénom, le numéro de téléphone, l'adresse électronique, le sexe et la date de naissance) et des champs « activate » et « firstlogon » qui permettent de gérer respectivement l'activation ou la désactivation d'un utilisateur et la première connexion d'un utilisateur.

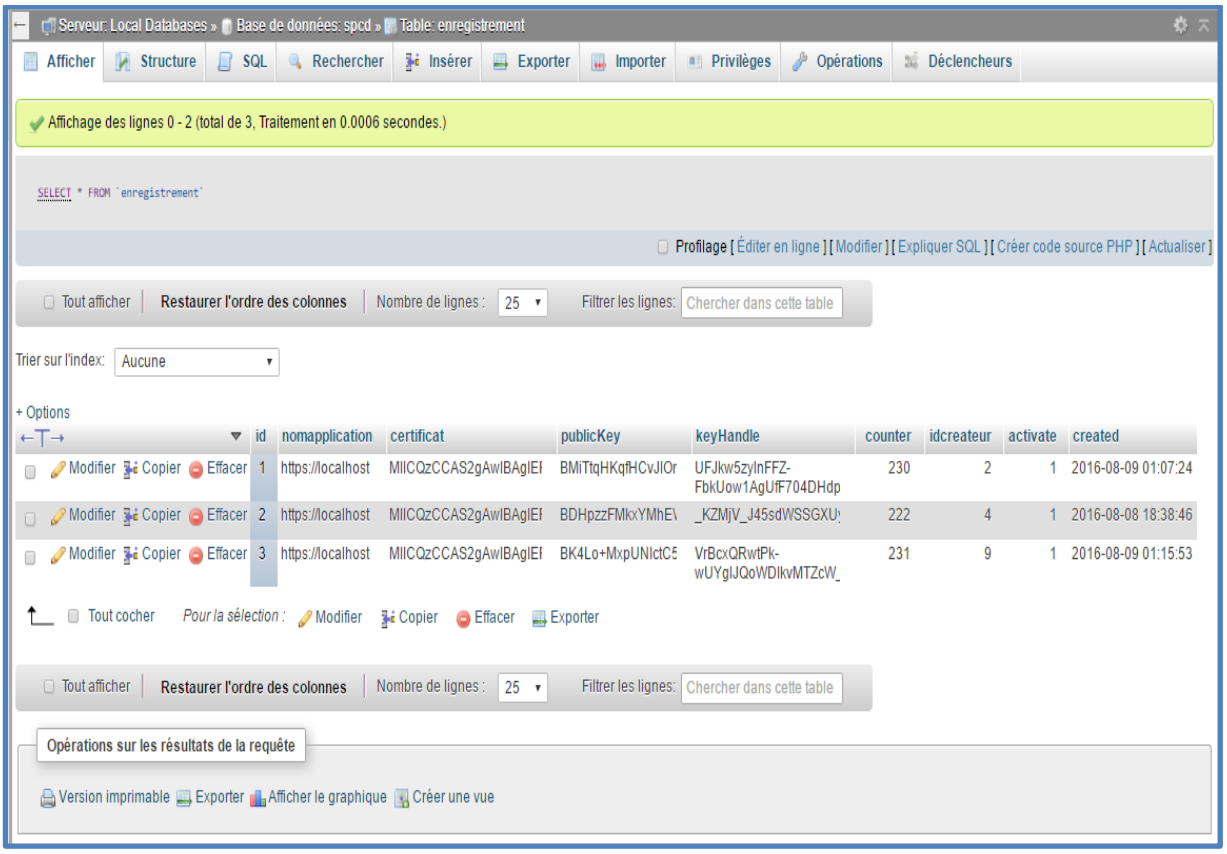


Figure 24: Table "Enregistrements" dans la base de données

La figure 24 nous présente la structure de la table contenant les informations de gestion de la clé de sécurité pour chaque utilisateur. La table est constituée du numéro d'incrémentation automatique « id », de l'url de l'application test «nomapplication», du certificat de la clé de sécurité «certificat», de la clé publique «publicKey», du conteneur de clé «keyHandle», du compteur d'authentification de la clé «counter», du numéro identifiant l'utilisateur «idcreateur» et du champ «activate» qui renseigne sur l'état de la clé (activée=1, désactivée=0).

Grâce au formulaire de gestion des utilisateurs présenté dans la figure 25, l'administrateur a la capacité de désactiver l'utilisateur ou de désactiver sa clé de sécurité à la suite d'une éventuelle perte.

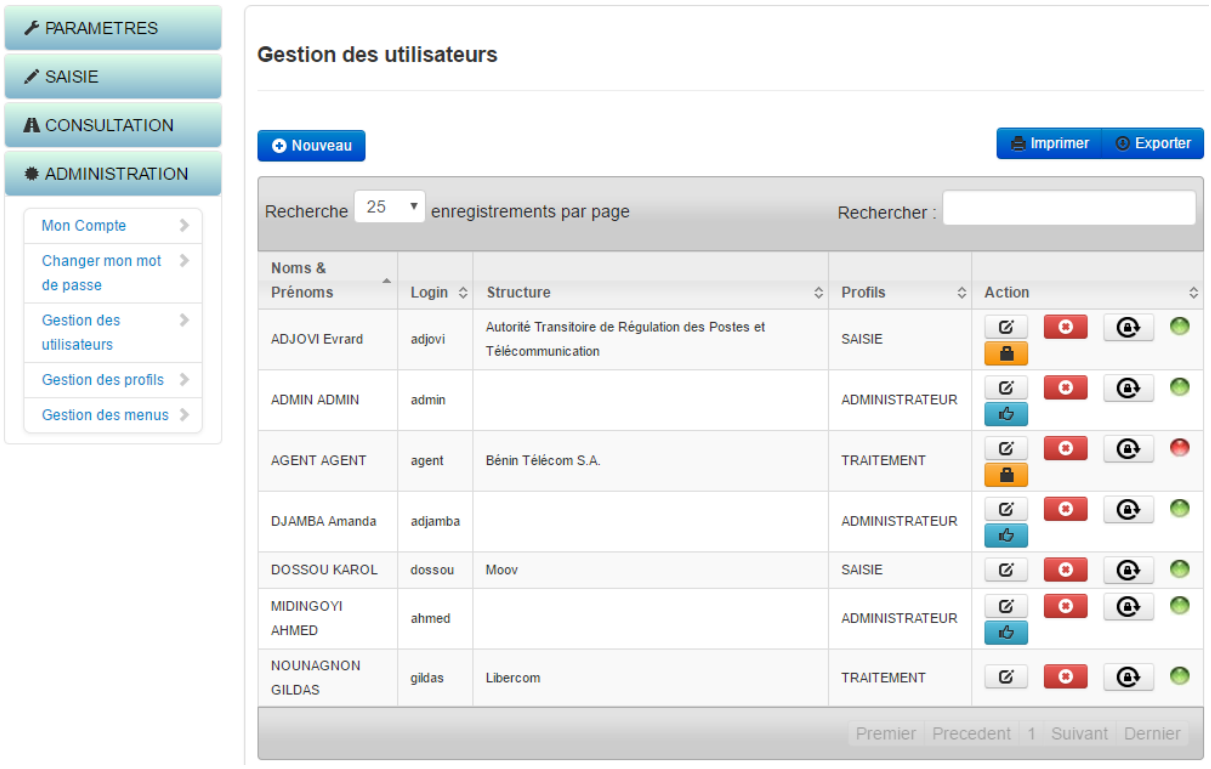


Figure 25: Gestion des utilisateurs

6- Discussion

Les spécifications du protocole U2F établies par l'alliance FIDO ont permis d'implémenter un système d'authentification forte et de l'intégrer dans une application web existante. L'objectif de ce système étant de palier aux différentes menaces des applications web dans le cas d'une authentification basée sur mot de passe, nous allons dans cette partie montrer ce qui fait la force et la faiblesse de ce système d'authentification.

6-1- Protection contre les vulnérabilités web

Il faut en premier lieu noter que certaines failles de sécurité web identifiées par OWASP ont pu être gérées grâce à ce système :

6-1-1- Protection contre l'injection SQL

Dans le cadre de notre étude qui s'est basée sur une application web professionnel où ce n'est pas l'utilisateur qui crée ses paramètres de connexion (mot de passe, nom d'utilisateur), ce problème ne se pose pas car il faut détenir une clé de sécurité pour se connecter. Cependant malgré cela, nous avons pris le soin de mettre en œuvre la solution à ce type d'attaque qui peut avoir des conséquences au cas où l'utilisateur perd sa clé de connexion sans l'avoir signalée à

l'administrateur. De ce fait, il s'agit de ne pas faire confiance aux requêtes de l'utilisateur en utilisant des requêtes préparées ou paramétrées à chaque fois qu'on interroge la base de données.

6-1-2- Protection contre la violation de la gestion d'authentification et de session

Elle est la deuxième faille des applications web identifiée par OWASP. Plusieurs méthodes existent pour résoudre ce type d'attaque. L'utilisation d'un second facteur d'authentification rend plus difficile le détournement de session ou la violation de la gestion d'authentification du fait du test de la présence de l'utilisateur. Cependant nous n'avons pas mis en œuvre la possibilité de tester la présence de l'utilisateur pour certaines fonctionnalités délicates de l'application après avoir été connecté. Mais il faut souligner que les études de Lang (2016) et de Bergem et Maury (2016) montrent que la confirmation des transactions n'était pas possible avec ce protocole, ce qui ne permet pas pour l'instant de remplacer les protocoles de validation par SMS par le protocole U2F pour ces types de fonctionnalités.

Les Spécifications de U2F excluent aussi le vol de session par XSS en supposant que le navigateur Web agit comme un agent de confiance de l'utilisateur ; le principe de l'attaque XSS reposant sur la confiance que le client manifeste à l'égard du serveur, et la possibilité de mélanger des scripts JavaScript avec la description HTML d'informations dans une page web.

6-1-3- Protection contre le hameçonnage et l'attaque de l'intercepteur (Man-In-The-Middle) pendant l'authentification

Le protocole U2F permet de lutter contre le hameçonnage puisqu'à chaque authentification, l'authentificateur U2F détermine la clé privée correspondant à l'URL afin de signer les données qui lui sont fournies, or l'URL de l'hameçonnage est différent de l'URL réel. Par conséquent la demande de signature sera rejetée. Cependant, la méthode la plus simple de lutter contre cette attaque est de ne pas cliquer directement sur les liens mais d'ouvrir le navigateur et saisir soi-même l'URL d'accès au service.

Durant le processus d'enregistrement, une paire de clés cryptographiques U2F est associée à une application web spécifique. Ainsi, l'authentificateur U2F doit refuser de signer avec une paire de clés délivrée pour une autre application web. Ce refus de signature est illustré par la Figure 26

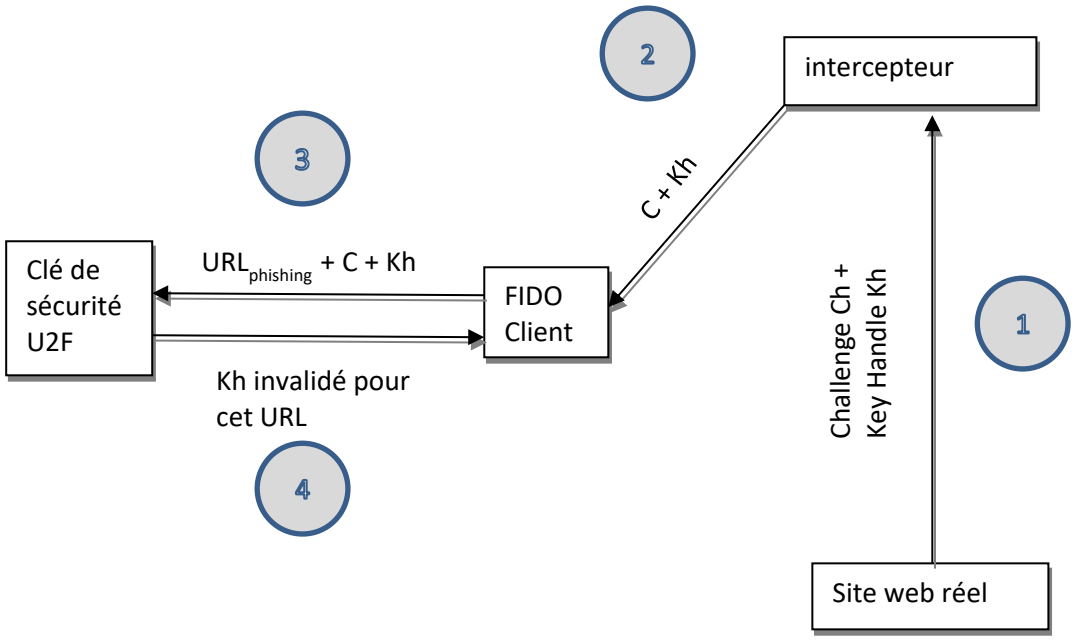


Figure 26: Défense contre l'intercepteur durant l'authentification

Dans ce cas l'intercepteur a pu obtenir le challenge et le conteneur de clé envoyés par le site légitime. La vérification de l'URL de l'application web durant l'authentification par la clé de sécurité U2F permet de s'assurer que la clé publique et le conteneur de clés délivrés à une application web donnée ne peuvent pas être exploités pour une autre application notamment le site web de l'intercepteur. Ainsi, si un intercepteur (MITM) tente de s'insérer entre l'utilisateur et le serveur durant la phase d'authentification, le système d'authentification basé sur le protocole u2f peut le détecter dans le cas où leurs URLs sont différents.

Par contre, il existe aussi des cas où, l'intercepteur étant directement sur le même chemin réseau a le même URL que le site légitime ou bien il présente un certificat validé par le navigateur ou l'utilisateur. Ce type d'attaque est dénommé "attaque de l'intercepteur TLS" (Man-In-the-middle TLS). U2F intègre également une protection contre les attaques Man-In-The-Middle TLS. Pour cela, U2F repose actuellement sur une extension TLS nommée « TLS Channel ID », qui n'est implémentée que sur les serveurs de Google (dans leur librairie BoringSSL). En revanche, la vérification de cette attaque est volontairement désactivée, pour éviter de bloquer les clients utilisant un proxy TLS, le proxy agissant alors comme un MITM TLS.

Concernant les données envoyées à l'authentificateur U2F pour demande de signature, le navigateur retourne un objet contenant des informations relatives à l'aperçu de l'application web dénommé ClientData décrit plus haut. Il contient entre autres, l'identifiant du canal TLS (channelID TLS). Selon les spécifications, l'utilisation de cet identifiant est optionnelle. En

utilisant cette option, lors de la vérification de la signature par le serveur, ce dernier peut remarquer deux différentes connexions TLS lorsqu'il y a présence d'un intercepteur TLS. Cependant, si cet identifiant n'est pas exploité, cela n'a pas d'impact sur la sécurité de U2F (Bergem et Maury, 2016). Son utilisation renforce la sécurité et améliore les solutions contre les attaques, notamment l'attaque de l'intercepteur TLS.

6-2- Recul sur le protocole U2F

Le système mis en place s'est basé sur un protocole dont les spécifications techniques ont été publiées en 2015. C'est un nouveau protocole avec peu de documentations et d'études dont celles relatives à l'expérimentation des attaques sur les systèmes répondant à ce protocole. Sa fiabilité n'a donc pas encore été approuvée contrairement aux autres schémas d'authentification forte tels que les TOTP (Time-based One-Time Password) ou encore les certificats client. Aussi depuis longtemps seul le navigateur Chrome (version 40 et plus) avait implémenté une extension du protocole U2F. Le navigateur Firefox l'a intégrée cette année grâce à son extension add-ons.

La clé de sécurité ne stocke aucune information personnelle d'un utilisateur, ce qui constitue une force mais aussi une faiblesse. Une force, dans la mesure où, lorsque la clé est dérobée, le possesseur ne pourra lier la clé à un utilisateur. Une faiblesse, dans la mesure où, la clé peut être utilisée par un autre utilisateur.

Conclusion générale

La sécurité des applications web est fondamentale pour assurer la fiabilité, l'intégrité et la confidentialité des échanges de données. Ainsi, pour atteindre ces objectifs, ces applications doivent intégrer un système d'authentification sécurisé afin d'éviter que des intrus n'accèdent aux ressources auxquelles ils ne sont pas autorisés. Il apparaît que les systèmes d'authentification courants sont basés sur les mots de passe ou sur un second facteur qui n'assurent pas la sécurité et la convivialité voulues. Ainsi, le sujet de ce mémoire a été choisi dans le but de contribuer au renforcement des mesures de sécurité dans le développement des applications web.

L'objectif principal de notre travail est d'étudier le nouveau protocole standard U2F en vue de l'implémenter pour assurer une authentification forte à deux facteurs pour la connexion à une application web. Le premier facteur est le mot de passe et le second facteur un authentificateur U2F. L'intérêt de notre étude est de compenser non seulement la faiblesse des mots de passe mais aussi des seconds facteurs d'authentification (comme les codes de sécurité envoyés par SMS) face aux attaques de phishing, en utilisant un second facteur sécurisé reposant sur une solution matérielle telle que la clé de sécurité USB de Yubico. U2F est un protocole d'authentification forte créé par la FIDO Alliance qui repose sur un token physique et communiquant en USB, en Bluetooth ou en NFC (pour les smartphones par exemple).

Après avoir effectué un stage à l'ASECNA où nous avions pu mettre en pratique les connaissances acquises au cours de notre formation, nous avons profité des fructueux échanges avec les informaticiens de la structure pour bien cerner le contour de notre projet d'étude. Ainsi grâce aux spécifications du protocole standard, nous avons pu modifier le code d'une application web existant pour garantir une authentification forte avec la clé Yubikey pour la connexion à l'application. Ce système apporte des garanties de sécurité et une facilité d'utilisation. En tenant compte de l'implémentation du protocole, nous pouvons déduire que le système permet notamment de se protéger contre les attaques par phishing, les injections SQL, les vols de session. U2F intègre également une protection contre les attaques Man-In-The-Middle TLS en se reposant sur l'extension TLS nommée « TLS Channel ID » qui a été aussi en partie implémentée dans les librairies U2F existantes sur le site de Yubico du côté client. La solution de cette attaque n'a pas pu être implémentée dans notre système car nous n'avions pas pu obtenir de documentation nécessaire pour sa mise en œuvre. Cette protection avancée n'étant donc pas pris en compte dans le système, ce dernier reste néanmoins une solution simple et

efficace comme système d'authentification à double facteur non seulement à cause de l'utilisation de la clé de sécurité mais aussi des méthodes cryptographiques utilisées pour garantir l'intégrité et la fiabilité des échanges de données.

Les résultats de ce travail permettront aux développeurs de mieux appréhender le nouveau protocole ouvert qui représente un standard en matière d'authentification qu'ils pourront aussi intégrer dans le développement de leurs applications à travers les fonctions essentielles décrites dans le mémoire. Aussi, les entreprises peuvent exiger que leurs applications répondent au protocole u2f qui assure une authentification plus sécurisée que l'utilisation simple des mots de passe.

Les perspectives de notre étude se résument en deux points. De prime abord, nous envisageons mettre en œuvre les attaques sur le système pour vérifier de manière pratique la résistance des systèmes d'authentification basés sur le protocole U2F face aux différentes attaques web. Secundo, nous comptons étudier l'extension Channel ID de TLS et voir la faisabilité de l'intégrer côté serveur pour assurer une défense contre l'intercepteur TLS.

Bibliographie

Adams, A., & Sasse, M. (1999). Users Are Not The Enemy. *Commun. ACM , 42* (12), 41-46.

Bergem, M., & Maury, F. (2016). *A first glance at the U2F protocol Mickaël.* Consulté le 06 12, 2016, sur https://www.sstic.org/2016/presentation/a_first_glance_at_the_u2f_protocol/

Bonneau, J., Herley, C., Van Oorschot, P., & Stajano, F. (2012). The Quest to Replace Passwords: A framework for Comparative Evaluation of Web Authentication Schemes. *In: 2012 IEEE Symposium on Security and Privacy* .

Chincoun, V. (2014). *Mise en place d'un tiers de confiance pour la délivrance des certificats électroniques.* Master, ENEAM.

FIDO. (2015). Consulté le 02 12, 2016, sur Fast IDentity Online: https://fidoalliance.org/

Google Inc. (2015). *Google 2-Step Verification.* Consulté le 07 10, 2016, sur Google: https://support.google.com/accounts/answer/180744

Harbach, M., Fahl, S., Rieger, M., & Smith, M. (2013). On the Acceptance PrivacyPreserving Authentication Technology: The Curious Case of National Identity Cards. (Springer, Éd.) *In: Privacy Enhancing Technologies* , 245-264.

ISACA. (2012). *DSS06 within Cobit 5: Enabling Processes*. Consulté le 01 02, 2016, sur http://www.isaca.org/COBIT/Pages/COBIT-5-Enabling-Processes-product-page.aspx

ISF. (2014). *The Standard of Good Practice for Information Security*. Consulté le 01 02, 2016, sur Information Security Forum: https://www.securityforum.org/shop/p-71-173

ISO. (2014). Consulté le 01 20, 2016, sur ISO: http://standards.iso.org/ittf/PubliclyAvailableStandards/c063411_ISO_IEC_27000_2014.zip

Kindervag, J., Shey, H., & Mak, K. (2015). Undersyand the business impact and cost of a breach. *Forrester Research* , 16.

Lang. (2016). *Security Keys: Practical Cryptographic SecondFactors for the Modern Web*. Consulté le 05 2, 2016, sur http://fc16.ifca.ai/preproceedings/25_Lang.pdf

Molva, R., & Roudier, Y. (2011). *Protocoles d`Authentification.* Consulté le 01 09, 2016, sur www.eurecom.fr/fr/publication/836/download/ce-roudyv-010501.pdf

Morris, R., & Thompson, K. (1979). Password security: a case history. *ACM , 22* (11), 594-597.

Needham, R., & Schroeder, M. (1978). Using encryption for authentication in large networks of computers. *Communications of the ACM* (21), 993-999.

NIST. (2013). *Security and Privacy Controls for Federal Information Systems and Organizations. Special Publication 800-53 revision 4*. Récupéré sur Joint Task Force

Transformation Initiative. U.S. National Institute of Standards and Technology.: http://dx.doi.org/10.6028/NIS

Otway, D., et Rees, O. (1987). Efficient and timely mutual authentication. *Operating Systems review , 21* (1), 8-10.

Parno, B., Kuo, C., et Perrig, A. (2006). Phoolproof phishing prevention. (Springer, Éd.) *Lecture Notes in Computer Science , 4107*, 1-19.

Railton, J., et Kleemola, K. (2015). *London Calling: Two-Factor Authentication Phishing From Iran.* Consulté le 06 12, 2016, sur https://citizenlab.org/2015/08/iran_two_factor_phishing

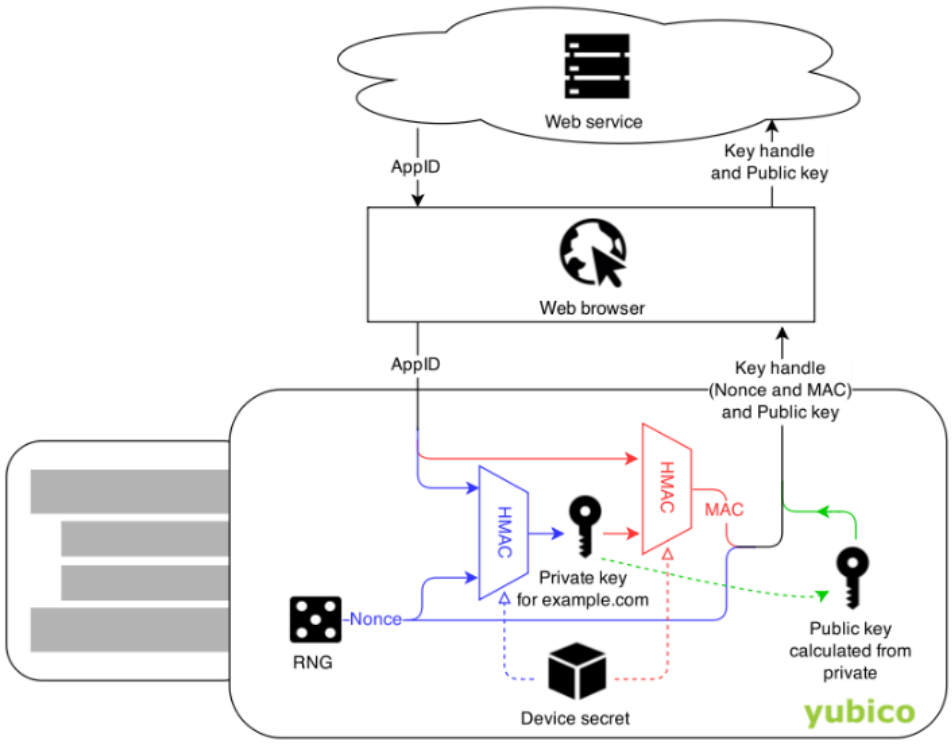
Toopher. (2012). *2 Factor Authentication*. Consulté le 4 23, 2016, sur http://toopher.com

Villacres, C. (2003). *Authentification de A à Z.* Récupéré sur http://docplayer.fr/553646-1-l-authentification-de-a-a-z.html

Table des matières

Annexe

Annexe 1- Structure de la clé de sécurité Yubico



Annexe 2- Protocole SSL /TLS

Le protocole SSL (Secured Socket Layer) a été conçu pour assurer une communication confidentielle et fiable entre deux applications et pour identifier le serveur et parfois le client. Ce protocole a été repris et légèrement modifié par l'IETF (RFC 2246) sous le nom de TLS (Transport Layer Security). Le terme SSL désigne communément les protocoles SSL version 3 ou TLS. Le protocole SSL nécessite un protocole de transport sûr (par exemple TCP) pour la transmission et la réception de données.

Le protocole est composé de deux couches. Au niveau le plus bas, juste au-dessus d'un protocole de transport sûr, se trouve le SSL Record Protocol. Celui-ci est utilisé pour encapsuler d'autres protocoles de plus haut niveau tel le SSL Handshake Protocol qui permet au serveur et au client de s'authentifier et de négocier un algorithme de chiffrement et des clés cryptographiques communes avant que le protocole d'application ne reçoive son premier octet d'information.

Annexe 3- Code principal

```
<?php
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 * Description of ConnectClass
 *
 */
class ConnectAdmin {

//Déclaration des variables de la classe
    var $login;
    var $pass;
    var $register;
    var $authenticate;
    var $operation;

//   constructeur----------------------
    function __construct($login, $pass, $register, $authenticate, $operation) {
        $this->login = $login;
        $this->pass = $pass;
        $this->register = $register;
        $this->authenticate = $authenticate;
        $this->operation = $operation;
    }

//récupération des valeurs des parametres
    function set_param($login, $pass, $register, $authenticate, $operation) {
        $this->login = $login;
```

```
        $this->pass = $pass;
        $this->register = $register;
        $this->authenticate = $authenticate;
        $this->operation = $operation;
    }

    //Controle d'authentifiacation
    function control_access() {
        global $dbh; //variable globale de la connexion
        //récupération des informations entrées au clavier(login et mot de passe)
        $send_login = $this->login; //recuperation du login
        $send_pass = Functions::crypter($this->pass); //recuperation et cryptage du mot de passe saisi
        //recuperation des valeurs des zones de texte cachées pour le traitement U2F
        $send_op = $this->operation; //zone de texte pour identifier l'opération encours (enregistrement/authentification)
        $send_register = $this->register; //zone de texte qui recupere les informations provenant de la clé pour l'enregistrement
        $send_authenticate = $this->authenticate; //zone de texte qui recupere les informations provenant de la clé pour l'authenification

        $login_retourne = ""; //initialisation de la variable login qui sera selectioné dans la base
        $pass_retourne = ""; //initialisation de la variable mot de passe qui sera selectioné dans la base
        $return = array(); //mise à vide du tableau de retour
        $return['redirectNo'] = 0; //Initialisation de la variable redirectNo pour controler les cas de redirection
        if (!empty($send_login)) {//Si le login n'est pas vide
            if (!empty($send_pass)) {//Si le mot de passe n'est pas vide
                $search_login_sql = 'SELECT login FROM utilisateur WHERE motdepasse = ?'; //requete de sélection du login du user en fonction du mot de passe saisi
                $search_login = $dbh->prepare($search_login_sql); //preparation de la requete
                $search_login->bindParam(1, $send_pass); //insertion du parametre (login)
                $search_login->Execute(); //execution de la requete
                while ($data_login = $search_login->fetch()) {//mis a diposition du champ selectionné ds la requete
                    if ($data_login['login'] == $send_login) {//compraison du login sélectionné et de celui saisi par le user
                        $login_retourne = $data_login['login']; //recuperation du login ds la variable $login_retourne (lorsqu'ils st identique)
                    }
                }
```

```
$search_pass_sql = 'SELECT  motdepasse FROM utilisateur WHERE  login = ?'; //requete de sélection du mot de passe du user en fonction du login saisi

$search_pass = $dbh->prepare($search_pass_sql); //preparation de la requete

$search_pass->bindParam(1, $send_login); //insertion du parametre

$search_pass->Execute(); //execution de la requete

while ($data_pass = $search_pass->fetch()) {//mis a diposition du champ selectionné ds la requete

  if ($data_pass['motdepasse'] == $send_pass) {//compraison du login sélectionné et de celui saisi par le user

    $pass_retourne = $data_pass['motdepasse']; //recuperation du mot de passe ds la variable $pass_retourne (lorsqu'ils st identique)

  }

}


//Si les entrées(login et mot de passe saisi) concordent aux données dans la base de données (login et mot de passe sélectionnés)

if (($login_retourne == $send_login) AND ($pass_retourne == $send_pass)) {

//Récupération des informations du connecté

  $search_infos_sql = "SELECT utilisateur.*, idProfil, libelleprofil FROM profil, utilisateurprofil,utilisateur

      WHERE utilisateurprofil.idProfil=profil.id AND

      utilisateurprofil.idUtilisateur=utilisateur.id AND

      login=? AND motdepasse = ?"; //requete de sélection des infomations du user et de so profil

  $search_infos = $dbh->prepare($search_infos_sql); ////preparation de la requete

  $search_infos->bindParam(1, $send_login); //insertion du 1er parametre

  $search_infos->bindParam(2, $send_pass); //insertion du 2eme parametre

  $search_infos->Execute(); //execution de la requete

  $data_infos = $search_infos->fetch(); //mis a diposition des champs selectionnés ds la requete


  if ($data_infos["activate"]) {//si le compte du user est activé ou si un profil lui a été attribué

    //afecion des données sélections dans le tableau $return (précedement initialisé)

    $return["user"]["id"] = $data_infos['id'];

    $return["user"]["login"] = $data_infos['login'];

    $return["user"]["nom"] = $data_infos['nom'];

    $return["user"]["idprofil"] = $data_infos['idProfil'];

    $return["user"]["libprofil"] = $data_infos['libelleprofil'];

    $return["user"]["prenom"] = $data_infos['prenom'];

    $return["user"]["nomprenom"] = $data_infos['nom'] . ' ' . $data_infos['prenom'];
```

```
                if ((!empty($return["user"]['nomprenom'])) && (!empty($return["user"]['login']))) {//Si les informations du connecté existent

    //debut traitement U2F==========================================================

                    $userId = enregistrement::getUserId($send_login, $send_pass); //recuperation de l'id du user en fonction du login et du mot de passe

                    $result = 0; //initialisation de la variable $result;

                    if ($userId != 0) {//si l'id du user est différent de zéro(0)

                        if (enregistrement::verifEnreg($userId)) {//verification si l'utilisateur s'est deja enregitré

                            //si oui :demarrage du traitement pour l'authentification

                            if ($send_op == "") {

                                try {

                                    $u2fVar = enregistrement::initU2F(); //instanciation de la classe U2F:creation de l'objet $u2fVar

                                    $resultTab = enregistrement::getEnreg($userId); //recuperation d'un tableau d'objet des informations de l'utilisateur

                                    $reqs = json_encode($u2fVar->getAuthenticateData($resultTab)); //recuperation etdes informations du Keyhandle et du challenge et encodage au format json

                                    $_SESSION['authReq'] = $reqs; //création et initialisation de la variable session authReq avec le resultat de getAuthenticateData()

                                    Functions::afficheBoiteMsg("Veuillez appuyer sur la clé pour l'authentification."); //message d'orientation

                                    ?>

                                    <script>

                                        setTimeout(function() {

                                        //                                        console.log("sign: ", <?php echo $reqs; ?>);

                                            u2f.sign(<?php echo $reqs; ?>, function(data) {

                                                var frm = document.getElementById('frmconn');//indexation du fomulaire de connexion

                                                var auth = document.getElementById('authenticate');//indexation de la zone de texte authenticate

                                                var login = document.getElementById('login');//indexation de la zone de texte login

                                                var mdp = document.getElementById('pass');//indexation de la zone de texte pass

                                                var action = document.getElementById('operation');//indexation de la zone de texte operation

                                        //                                            console.log("Authenticate callback", data);

                                                auth.value = JSON.stringify(data);//recuperation et encodage des informations de la clé pr l'authentification

                                                login.value = '<?php echo $send_login; ?>';//affectation du login saisi
```

```
                    action.value = 'auth';//initialisation de l'opération:auth=authentification
                    mdp.value = '<?php echo $this->pass; ?>';//affectation du mot de passe saisi
                    frm.action = "commitConnect.php";//definition de l'action du formulaire
                    frm.method = "post";//definition de la methode du formulaire
                    frm.submit();//revalidation du formulaire
                });
            }, 1000);
        </script>
        <?php
    } catch (Exception $e) {
        Functions::afficheBoiteMsg("Erreur getAuthenticate:" . $e->getMessage()); //message d'erreur en cas d'exception
    }
} else {
    try {
        if ($send_authenticate) {//si la zone de texte authenticate est renseignée
            $u2fVar = enregistrement::initU2F(); //instanciation de la classe U2F:creation de l'objet $u2fVar
            //authentification:comparaison des informations du client et de la clé
            $reg = $u2fVar->doAuthenticate(json_decode($_SESSION['authReq']), enregistrement::getEnreg($userId), json_decode($send_authenticate));
            $lAuthen = new enregistrement($reg->id); //instanciation de la classe enregistrement
            $lAuthen->setCounter($reg->counter); //affectation de la valeur du compteur
            $result = $lAuthen->modifEnreg(); //mis à jour du compteur
            $_SESSION['authReq'] = null; //mis à null de la variable session authReq
            $send_authenticate = ""; //mise à vide de la variable $send_authenticate
            if ($result == 1) {//en cass de succes de la mis à jour du compteur
                Functions::afficheBoiteMsg("Authentification effectué avec succès!");
            } else {//en cas d'echec de la mise à jour du compteur
                Functions::afficheBoiteMsg("Une erreur est survenue!");
            }
        }
    } catch (Exception $e) {
        $send_authenticate = ""; //mise à vide de la variable $send_authenticate
        Functions::afficheBoiteMsg("Erreur dotAuthenticate:" . $e->getMessage()); //message d'erreur en cas d'exception
```

```
            }
        }
    } else {// l'utilisateur ne s'est pas encore enregistré: demarrage du traitement de l'enregistrement du user
        if ($send_op == "") {
            try {
                $u2fVar = enregistrement::initU2F(); //instanciation de la classe U2F:creation de l'objet $u2fVar
                $data = $u2fVar->getRegisterData(enregistrement::getEnreg($userId)); //
                list($req, $sigs) = $data; //création de deux objets a partir du esultat obtenu de getRegisterData
                $_SESSION['regReq'] = json_encode($req); //création de la variable session regReq (constitué de challenge, appId)
                Functions::afficheBoiteMsg("Veuillez appuyer sur la clé pour l'enregistrement.");
                ?>
                <script>
                    setTimeout(function() {
                    //                                      console.log("Register: ", <?php echo json_encode($req); ?>);
                        u2f.register([<?php echo json_encode($req); ?>], <?php echo json_encode($sigs); ?>, function(data) {
                            var frm = document.getElementById('frmconn');//indexation du fomulaire de connexion
                            var reg = document.getElementById('register');//indexation de la zone de texte register
                            var login = document.getElementById('login');//indexation de la zone de texte login
                            var mdp = document.getElementById('pass');//indexation de la zone de texte pass
                            var action = document.getElementById('operation');//indexation de la zone de texte operation
                    //                                      console.log("Register callback", data);
                            if (data.errorCode) {//i erreur
                                alert("Enregistrement non effectué! Code=" + data.errorCode);//message d'erreur
                                return;
                            }
                            reg.value = JSON.stringify(data);//recuperation des informations de la clé pr l'enregistrement
                            login.value = '<?php echo $send_login; ?>';//affectation du login saisi
                            action.value = 'enreg';//initialisation de l'opération:enreg=enreistrement
                            mdp.value = '<?php echo $this->pass; ?>';//affectation du mot de passe saisi
                            frm.action = "commitConnect.php";//definition de l'action du formulaire
```

```
                frm.method = "post";//definition de la methode du formulaire
                frm.submit();//revalidation du formulaire
            });
        }, 1000);
    </script>
    <?php
} catch (Exception $e) {
    Functions::afficheBoiteMsg("Erreur getRegister:" . $e->getMessage()); //message d'erreur en cas d'exception
}
} else {
    try {
        if ($send_register) {//si la zone de texte register est renseignée
            $u2fVar = enregistrement::initU2F(); //instanciation de la classe U2F:creation de l'objet $u2fVar
            //enregistrement :generation de la clé publique, du certificat,...)
            $reg = $u2fVar->doRegister(json_decode($_SESSION['regReq']), json_decode($send_register));
            if (!enregistrement::verifdoublonKey($userId, $reg->publicKey)) {//verifier si c'est pas la mm clé(en cas de désactivation de la précedente clé du user)
                $lAuthen = new enregistrement(0); //instanciation de la classe enregistrement
                $lAuthen->setKeyHandle($reg->keyHandle); //affectation de la valeur du KeyHandle
                $lAuthen->setPublickey($reg->publicKey); //affectation de la valeur de la clé publique
                $lAuthen->setCertificat($reg->certificate); //affectation de la valeur du certificat
                $lAuthen->setNomapplication("https://localhost"); //affectation de la valeur de l'appId
                $lAuthen->setCounter($reg->counter); //affectation de la valeur du compteur
                $lAuthen->setIdcreateur($userId); //affectation de l'id du user
                $result = $lAuthen->ajoutEnreg(); //sauvegarde des informations ds la table enregistrement
                $_SESSION['regReq'] = null; //mis à null de la variable session regReq
                $send_register = ""; //mise à vide de la variable $send_register
                if ($result == 4) {//en cas de succes de la sauvegarde des infos de l'enrgistrement
                    Functions::afficheBoiteMsg("Enregistrement effectué avec succès!");
    //                $return['msg'] = '<div class="alert alert-success">Enregistrement effectué avec succès!</div>';
                }
                if ($result == 0) {//en cas d'echec du à une ereur coté serveur (scrit ou autres)
```

```
                                    Functions::afficheBoiteMsg("Une erreur est survenue!");
                                }
                                if ($result == 2) {//en cas d'erreur d'un autre cas
                                    Functions::afficheBoiteMsg("Cet utilisateur s'est déja enregistré pour cet application!");
                                }
                            } else {//si c'est la clé desactivé que l'utilisateur veut utilisé
                                if ($result == 2) {//en cas d'erreur d'un autre cas
                                    Functions::afficheBoiteMsg("Cet utilisateur s'est déja enregistré avec cette clé, Veuillez contacter l'Administrateur!");
                                }
                            }
                        }
                    } catch (Exception $e) {
                        $send_register = ""; //mise à vide de la variable $send_register
                        Functions::afficheBoiteMsg("Erreur doRegister:" . $e->getMessage()); //message d'erreur en cas d'exception
                    }
                }
            }
        //fin traitement U2F============================================================
                if ($result == 1) {//si l'authentification s'est bien déroulée
                    $return['msg'] = "";
                    $return['rep'] = true; //autorise la redirection sur la page d'acceuil de lapplication
                } else {
                    if ($result != 4) {
                        $return['msg'] = "";
                    } else {//si l'enregistremet s'est bien deroulé
                        $return['redirectNo'] = 1; //autorise la ridirection sur la page de connexion
                    }
                    $return['rep'] = false; //n'autorise pas la redirection sur la page d'acceuil de l'appliaton
                }
            } else { //les informations du connecté n'existent pas
                $return['msg'] = "<div class=\"alert alert-error\">Erreur de connection Code #4411 (Contactez &agrave; l'administrateur)</div>"; //message d'erreur
```

```
                    $return['rep'] = false;

                }

            } else {//le compte du user n'est pas activé ou n'a pas de profil

                $return['msg'] = '<div class="alert alert-error">Compte de l\'utilisateur n\'est pas activé ou n\'a pas de profil! Contactez l\'administrateur.</div>'; //message d'erreur

                $return['rep'] = false;

            }

        } else {//les entrées (login et mode de passa saisi) ne concordent pas aux données dans la base de données (login et mot de passe sélectionés)

            $return['msg'] = '<div class="alert alert-error">Le login et le mot de passe ne concordent pas!</div>'; //message d'erreur

            $return['rep'] = false;

        }

    } else {//le mot de passe est vide

        $return['msg'] = '<div class="alert">Le champ du mot de passe est vide!</div>'; //message d'erreur

        $return['rep'] = false;

    }

} else {//le login est vide

    $return['msg'] = '<div class="alert">Le champ du login est vide!</div>'; //message d'erreur

    $return['rep'] = false;

}

return $return; //renvoie le tableau contenant les information du user connecté

}


function connection() {

    return $this->control_access(); //execute la fonction

}


}

?>
```

Annexe 4- Organigramme de l'ASECNA Bénin

ORGANIGRAMME NOMINATIF DE LA REPRESENTATION DE L'ASECNA AU BENIN
(Référence : décision N°2014/01388/ASECNA/DGDD du 01/07/2014)