UQAC
Université du Québec
à Chicoutimi

SÉCURITÉ DES DONNÉES DE CAPTEURS AMBIANTS DANS LES HABITATS À L'AIDE D'AUTOENCODEURS

PAR BÉGUÉ HADJA RAHINATOU FORGO

MÉMOIRE PRÉSENTÉ À L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI COMME EXIGENCE PARTIELLE EN VUE DE L'OBTENTION DU GRADE DE MAÎTRE ÈS SCIENCES EN INFORMATIQUE

QUÉBEC, CANADA

RÉSUMÉ

Dans ce mémoire, nous visons à implémenter un prototype d'algorithme de cryptographie basé sur les réseaux de neurones. Dans le cadre des travaux sur la reconnaissance d'activités au sein d'habitats intelligents opérés par le LIARA (Laboratoire d'Intelligence Ambiante pour la Reconnaissance d'Activités), une attention est portée de plus en plus sur la protection de la vie privée des individus. En effet pendant longtemps, on a eu affaire à des données de capteurs simples comme les tapis de pressions, les détecteurs électromagnétiques, les détecteurs de mouvement. Comme ces données ne permettent pas d'avoir accès à des informations génériques basiques sur ce qui se déroulait au sein des murs des résidences suivies, l'aspect sécurité occupait une petite place dans les travaux.

Dans le cadre d'un projet de déploiement à grande échelle de maisons intelligentes financé par MEDTEQ (le Consortium pancanadien de recherche industrielle et d'innovation en technologies médicales du Québec) et Age-Well (un réseau canadien visant à élaborer des technologies et des services pour le vieillissement en santé) , notre équipe explore l'utilisation des réseaux de neurones pour sécuriser les données transmises entre les maisons intelligentes et les serveurs de traitement de données.

Afin de bien comprendre nos besoins, il est important de poursuivre cette question cruciale en parallèle avec une étude du potentiel des réseaux de neurones pour remplacer la cryptographie traditionnelle dans l'encodage de données plus complexes. En effet, ces derniers temps, nous avons étudié des capteurs plus informatifs tels que les caméras thermiques, les caméras de profondeurs, les capteurs RFID et les radars à ultra large bande. Ces capteurs ont plus d'informations sur les activités qui se déroulent dans une résidence, mais ils impliquent une plus grande charge de traitement. En conséquence, il semble donc intéressant d'étudier la possibilité d'utiliser une architecture de réseau de neurones artificiels pour encoder localement les données et les décoder à distance afin d'augmenter la sécurité, de réduire les calculs supplémentaires et de diminuer la taille des données à transmettre. En somme, cette étude portera sur l'étude comparative de ces encodages au sein d'habitats intelligents et consistera à mettre en place un système d'encodage de données fonctionnel basé sur des autoencodeurs.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS

AE AutoEncodeur
AES Advanced Encryption Standard
ANN Artifical Neural Neurons
CAE Convolutional Autoencoder
CHSLD Centres d'Hébergement et de Soins de Longue Durée
CIRANO Centre Interuniversitaire de Recherche en ANalyse des Organisations
CNN Convolutional Neural Neurons
DES Data Encryption Standard
FIPS Federal Information Processing Standards
HMAC Hash-based Message Authentication Code
IA Intelligence Artificielle
IoT Internet of Things
IRM Imagerie par Résonance Magnétique
JSON JavaScript Object Notation
LIARA Laboratoire d'Intelligence Ambiante pour la Reconnaissance d'Activités
LoRa Long-Range
LoRaWAN Long-Range Wide-Area Network
MAC Message Authentication Code
MNIST Mixed National Institute of Standards and Technology
MS MilliSecondes
MSE Mean Squared Error
NIST National Institutes of Health Normes et technologie
NPCR Pixel Change Rate
RAM Random Access Memory
RFID Radio Frequency IDentification
SAE Sparse AutoEncoder
SGD Stochastic Gradient Descent
SHA Secure Hash Algorithm
UACI Unified Average Change Intensity
UWB Ultra WideBand
WIFI WIreless FIdelity
XOFs Extendable Output Functions

DÉDICACE

Je dédie cet humble travail,

à mes très chers parents signe de ma gratitude et de mon affection. Merci pour toutes ces fois où vous aviez veillé à mon chevet, pour n'avoir jamais douté de moi, pour m'avoir toujours poussé vers le chemin de l'excellence, pour m'avoir tout donné, pour cette éducation dont je peux me vanter, pour les sacrifices et pour tout cet amour. A toi Papa merci d'être cet homme généreux, sans pareil, juste exceptionnel et merci Maman pour m'avoir non seulement portée pendant 9mois mais aussi tout au long de ces dernières années. Aucun mot sur cette page ne saurait exprimer ce que je vous dois, ni combien je vous aime. Puisse ALLAH me permettre de vous rendre fiers et dignes.

REMERCIEMENTS

De prime abord, je tiens à remercier Kévin Bouchard pour cette opportunité qu'il m'a donné de faire mon projet de recherche sous sa direction. Son expertise, ses capacités humaines, sa patience et surtout ses encouragements m'ont été d'un grand support. Par la même occasion, je remercie mes co-directeurs Sebastien Gaboury et Fehmi Jaafar pour les commentaires constructifs dans le souci de la bonne réalisation de ce mémoire. Je les remercie aussi pour le financement qu'ils m'ont accordé lors de ma recherche.

Ensuite, je remercie toute l'équipe du Laboratoire d'Intelligence Ambiante pour la Reconnaissance d'Activités (LIARA) pour leur accueil chaleureux au sein du laboratoire. J'aimerais remercier particulièrement Virgile Lafontaine pour m'avoir beaucoup aidé dans la réalisation de mes travaux.

Je souhaite aussi exprimer ma reconnaissance à tous les professeurs et personnel de l'Université du Québec A Chicoutimi (UQAC), qui nous ont fourni les outils nécessaires pour la réussite de nos études universitaires.

De plus, j'aimerai remercier Nikiss pour m'avoir beaucoup aidé et avoir toujours répondu présent quand j'en avais besoin.

Je remercie profondement mes amis et spécialement Youka pour sa présence, sa tendresse et ses multiples encouragements tous les jours.

Je ne saurais terminer ces remerciements sans mentionner ma famille. Merci à mes fabuleux parents, à mes frères Aziz, Abdine et à ma soeur Yacine pour leur soutien et affection au quotidien. Je vous souhaite une vie pleine de bonheur et de succès et nous souhaite d'être toujours si complices et unis.

AVANT-PROPOS

Ce mémoire représente le fruit d'une année entière de travail et entre dans le cadre de la préparation d'un projet de fin d'études pour l'obtention du diplôme de maîtrise de recherche en informatique. Désirant travailler sur un sujet portant sur la sécurité et l'intelligence artificielle, Kévin Bouchard et moi avons trouvé l'accord parfait en nous embarquant sur l'implémentation d'un système de sécurisation des données d'habitats intelligents avec les autoencodeurs. Ce document présente le résumé des différentes étapes développées dans ce projet.

CHAPITRE I

INTRODUCTION

1.1 CONTEXTE DE LA RECHERCHE

1.1.1 VIEILLISSEMENT DE LA POPULATION

Dans des pays comme le Canada, l'évolution de la démographie va dans le sens d'une population vieillissante et selon Payeur *et al.* (2019) ce vieillissement de la population s'accélérera d'ici 2066 car les personnes âgées de 65 ans et plus représenteront 28% de la population du Québec. Ce chiffre était de 18% en 2016. Cette situation se comprend vu que de nos jours, le taux d'accroissement annuel moyen est légèrement supérieur à 1% et qu'on a un faible taux de fécondité. Une telle évolution démographique entrainera des conséquences sur certains domaines comme celui de la santé (Slade *et al.*, 2019).

Ce vieillissement vient avec des problèmes de santé affectant ainsi l'autonomie des personnes et les tâches quotidiennes pour prendre soin d'eux et de leur maison que ces individus exerçaient auparavant de manière aisée deviennent alors difficiles. Cette perte d'autonomie est d'autant plus flagrante lors de l'apparition des maladies neuro dégénératives comme la maladie de Parkinson ou celle d'Alzheimer qui touchent le plus souvent les personnes âgées. L'association des malades d'Alzheimer (Association *et al.*, 2018) explique que cette maladie provoque des symptômes comme des pertes de mémoire au stade précoce et entraîne une incapacité importante à interagir avec son environnement. Selon l'ONU (Wimo *et al.*, 2018), les frais déboursés pour la prise en charge des personnes touchées de dégénérescences physiques et/ou mentales atteignent plus de 10 milliards au Canada. Dans les années à venir (notamment en 2050) ils estiment que ce chiffre connaîtrait une augmentation vu qu'il y aurait une croissance d'environ 242% du nombre de personnes concernées. De plus, le Centre

interuniversitaire de recherche en analyse des organisations (CIRANO) estime que les coûts relatifs à la santé passeront de 42,9% à 68,9% des revenus totaux du gouvernement du Québec entre 2013 et 2030, sous l'effet de l'accroissement des coûts structurels de santé et de ce vieillissement de la population (Clavet *et al.*, 2013). En effet, le vieillissement de la population va de pair avec l'intensification des besoins en service de santé (Colloque sur le vieillissement de la population et les contraintes financieres du secteur de la sante. & Conseil économique du Canada., 1987).

Pour faire face à cette perte d'autonomie et prendre soin des aînés dans le besoin, il faut l'aide de la famille ou de personnel médical. Ces aînés sont reçus dans les Centres d'Hébergement et de Soins de Longue Durée (CHSLD). Cependant le coût des soins dans ces centres est généralement plus élevé que le maintien à domicile selon l'étude faite par (Chappell *et al.*, 2004). Aussi d'après le bilan 2021 de l'emploi au Québec, publié par l'Institut du Québec, le vieillissement de la population entraîne un manque de personnel qualifié (soignants, infirmiers).

Cette situation nous amène à revoir le système de santé et à y intégrer l'intelligence ambiante qui pourrait être une belle alternative pour que les personnes âgées reçoivent leur traitement dans leur milieu de vie naturel tout en ayant le confort (Kadowaki *et al.*, 2015) de leur maison. Cela permettrait en outre de garantir le droit à la santé auquel chaque individu a droit et par la même occasion réduire les coûts de santé (Gilmour, 2018).

1.1.2 HABITATS INTELLIGENTS ET RECONNAISSANCE D'ACTIVITÉS

Les maisons intelligentes améliorent de manière non invasive les soins à domicile pour les personnes âgées et les personnes en perte d'autonomie et préviennent la solitude. Il devient important de surveiller l'état sanitaire de ces patients pendant que les coûts des soins de santé

grimpent et que la population vieillit. C'est aussi ce que soutient Hao & Foster (2008). Pour ce faire, plusieurs chercheurs intègrent des capteurs dans les habitats les rendant ainsi intelligents et aptes à faire de la reconnaissance d'activités et par la même occasion de l'assistance et de la surveillance médicale (Chan *et al.*, 2009). Les informations récoltées lors de cette reconnaissance sont utiles en ce sens où elles permettraient de s'assurer que les personnes malades ou ayant une démence ont accompli correctement des tâches importantes et même de détecter des modifications de comportement pouvant être des signes d'une dégradation de la santé, ce qui permettrait une prise en charge immédiate (Alberdi *et al.*, 2018).

Pour Cook & Das (2004), les environnements intelligents sont des environnements qui ont la capacité d'extraire et d'exploiter des informations de l'environnement afin d'améliorer l'expérience d'une personne dans cet environnement. Ils comprennent un système géré à l'aide d'une intelligence ambiante pour fournir des services adaptés au contexte environnemental et faciliter un contrôle des équipements à distance (Alam *et al.*, 2012). Ces maisons produisent beaucoup de données à cause des réseaux de capteurs sans fil (détecteurs de mouvement, de température, de lumière) et actionneurs (écrans, microphones, hauts parleurs) incorporés dans l'habitat intelligent (Hevesi *et al.*, 2014; Cheng *et al.*, 2016). Parmi ces données, les données vitales des résidents et les données environnementales pourront être récoltées. Ces données seront transférées via Internet sur des serveurs distants ainsi qu'aux professionnels de la santé qui pourront les examiner, faire le suivi de leurs patients et intervenir en cas d'alerte ou d'anormalité des mesures effectuées par les capteurs (Noury *et al.*, 2004). Les patients quant à eux, pourront aussi avoir accès aux informations en lien avec leur santé en temps réel. Les actionneurs interfèrent avec le résident en diffusant des messages. Les capteurs, eux, ont pour rôle de communiquer des informations relatives à l'habitat intelligent ainsi qu'aux équipements pour prédire l'activité du résident. Dans la Figure 1.1 nous voyons un schéma fonctionnel d'un système intelligent.

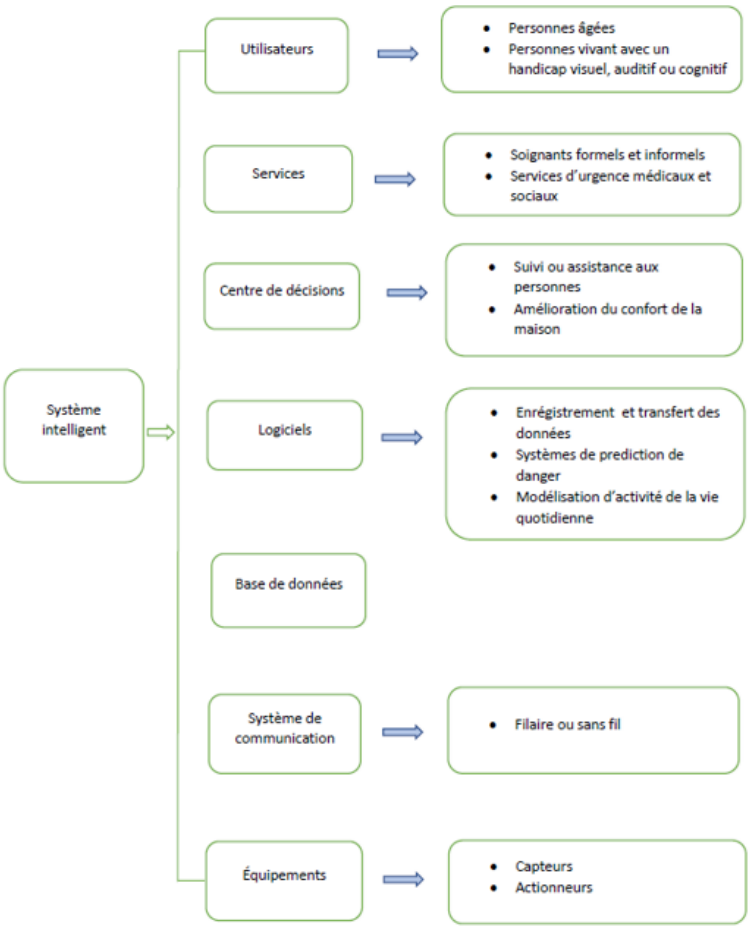


FIGURE 1.1 : Schéma fonctionnel d'un système intelligent

©Bégué Hadja Rahinatou Forgo

1.1.3 IOT ET SÉCURITÉ DES DONNÉES

Pour Zhou *et al.* (2021), il est essentiel pour une maison intelligente de combiner un réseau de capteurs avec Internet et des objets intelligents de la vie réelle. L'intégration de ces capteurs, objets intelligents, appareils et réseau est appelé l'IoT (Internet des Objects). Aussi selon Botterman (2009), l'IoT se définit comme étant «un réseau mondial d'objets interconnectés adressables de manière unique, basé sur des protocoles de communication

standard ». L'objectif principal de l'IoT est d'exploiter la puissance des communications ou de la connectivité Internet et du calcul en utilisant le réseau déjà existant ou un réseau sans fil/câblé personnalisé pour les objets du monde réel préexistants ou utilisés dans la vie de tous les jours. Les objets connectés peuvent inclure tous les objets utilisés quotidiennement, comme un lit, une chaise, une tasse à café, des appareils électroménagers, des appareils de cuisine (Ray & Bagwari, 2020). Ces entités sont capables de communiquer et de s'intégrer entre elles par le biais d'applications et de systèmes de gestion résidant dans des centres de données ou des services cloud pour collecter, générer, traiter et échanger des données (Abdul-Qawy *et al.*, 2015). Ces données sont transmises en local ou à distance depuis ou vers chaque nœud de capteur. Cette technologie de la maison intelligente basée sur l'IoT améliore la vie humaine en offrant une connectivité à tous peu importe l'heure ou le lieu (Gaikwad *et al.*, 2015).

Les applications de l'IoT vont nécessiter des technologies de réseau capable de communiquer sans fil et surtout sur de grandes distances. Parmi ces technologies on retrouve LoRAWAN (réseau étendu à longue portée) qui donne la possibilité de déployer des réseaux privés et de s'intégrer facilement à un certains nombre de plateformes de réseau mondiales comme The Things Network (Haxhibeqiri *et al.*, 2018). C'est d'ailleurs cela qui a fait sa popularité auprès de la communauté des chercheurs (Haxhibeqiri *et al.*, 2018). En plus de l'utilisation générale des réseaux de capteurs sans fil, différentes solutions d'appareils de soins de santé peuvent baser leur communication LoRaWAN. En effet, Catherwood *et al.* (2018) ont utilisé cette technologie comme solution de communication pour les soins à distance. Les auteurs ont démontré que LoRaWAN est utile pour l'Internet des objets médicaux.

Il existe d'autres technologies de transmission de données utilisées par l'IoT comme WIFI, Zigbee, Bluethooth. Les deux dernières technologies de transmission citées sont à courte portée multi-sauts. Certes elles impliquent une très faible consommation d'énergie mais leur

couverture réseau très limitée constitue une grande difficulté surtout s'il faut une couverture urbaine aux différentes applications.

Si l'IoT se développe rapidement, en revanche, il est confronté à des problèmes de sécurité et de protection des données. Il a de nombreux défis parmi lesquels le maintien de la confidentialité, la sécurisation des communications, la protection des informations traitées et/ou stockées, les différents contrôles d'accès et surtout la sensibilisation des utilisateurs. On peut voir ces défis sur la Figure 1.2.

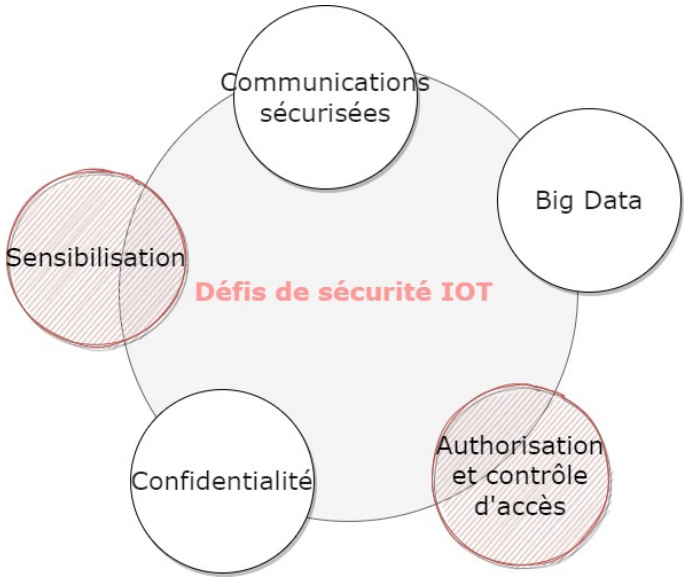


FIGURE 1.2 : Les enjeux de sécurité de l'IOT
©Bégué Hadja Rahinatou Forgo

Dans les habitats intelligents, l'utilisation de certains des capteurs peut parfois être très difficile à accepter, même dans le cadre de surveillance critique aux soins de santé (Offermann-van Heek *et al.*, 2019), puisqu'ils entraînent des répercussions sur la vie privée. En effet, ces maisons produisent beaucoup de données certaines beaucoup plus sensibles. Par exemple, bien que les capteurs de mouvement donnent des informations binaires, on pourrait analyser

et interpréter les données enregistrées et ainsi déduire l'activité d'une personne. L'usage et la consultation de ces données peuvent être effectués par une application ou par un membre du personnel soignant dans le but d'offrir une prise en charge adéquate du patient. Toutefois, ils doivent se faire en respect de la législation en vigueur et du respect de la vie privée du patient [1]. En absence de contrôle, le réseau de capteurs est accessible par tous laissant à quiconque la possibilité d'espionner son occupant.

1.2 PROBLÉMATIQUE DE LA RECHERCHE

Notre étude est faite au Laboratoire d'Intelligence Ambiante pour la Reconnaissance d'Activités (LIARA) de l'Université du Québec à Chicoutimi. Au LIARA l'un des axes des travaux porte sur le développement des habitants intelligents dotés de capteurs ambiants qui permettra au personnel médical ou aux proches de reconnaitre, suivre et de prédire les activités des personnes âgées, en perte d'autonomie ou ayant un handicap particulier.

Dans ces projets, un accent est mis de plus en plus sur l'aspect sécurité. Autrefois l'aspect sécurité n'occupait pas une si grande place vue que les données provenant des capteurs (détecteurs de mouvement, contacts électromagnétiques, tapis de pression, etc.) étaient simples et ne permettait pas d'avoir des informations pouvant porter atteinte à la vie privée des résidents. Aussi, les calculs étaient souvent centralisés sur des machines assez puissantes et branchées sur secteur. Celles-ci pouvaient donc facilement gérer les algorithmes gourmands de cryptographie traditionnelle comme AES, SHA, HMAC-SHA-256 et Blowfish. La majorité des chercheurs et praticiens utilisent les algorithmes de chiffrement comme moyen de sécurisation des données mais ces algorithmes sont gourmands et nécessitent des techniques plus complexes et une plus grande puissance de calcul (Ribordy & Trinkler, 2011).

1. https ://www.legisquebec.gouv.qc.ca/fr/document/lc/P-39.1

En plus de cela, ces dernières années, dans le but d'améliorer la reconnaissance d'activités dans nos habitats intelligents des recherches de plus en plus poussées ont été faites. Par exemple l'étude des capteurs plus riches en information tels que les caméras de profondeurs (Mousse, 2016), les caméras thermiques (Halima, 2021), les radars à ultra large bande (Maitre *et al.*, 2021) et les capteurs RFID (radio-identification) (Fortin-Simard *et al.*, 2015). Ces capteurs fournissent plus d'informations sur ce qui se passe à l'intérieur de la maison, mais en retour, ils impliquent des traitements beaucoup plus lourds et augmentent les risques pour la vie privée des résidents. Au vu de ces réalités et comme la reconnaissance d'activités se fait déjà avec des réseaux de neurones, il semble donc pertinent d'étudier la possibilité d'un encodage local et d'un décodage à distance des données à l'aide d'architectures de neurones artificiels. Les avantages avec cette méthode, c'est qu'elle pourrait assurer un bon niveau de sécurité en termes de confidentialité, réduire la charge de calculs supplémentaires et permettre d'obtenir une certaine compression de la taille des données à transmettre (Liu *et al.*, 2021).

Enfin, la sécurisation des données à ce niveau ne dépendrait pas d'une clé, mais du décodeur qui doit être difficile à reproduire et/ou à obtenir afin de garantir la sécurité.

1.3 APPORTS DE LA RECHERCHE

Technologie et science : Notre système consistant à encoder des données passant entre des nœuds sur un réseau est innovant. Il y a très peu d'équipes de recherche qui ont tenté cette approche et aucune, à notre connaissance, dans le domaine des habitats intelligents. Il existe d'autres moyens de sécurisation des données comme la cryptographie standard mais notre approche innovante est interressante et pourrait apporter un plus-value à la science.

Santé : À terme, un système comme le nôtre viendra augmenter la sécurité des données des résidents dans les habitats intelligents. Il est aussi rapide, robuste, de telle sorte qu'on ne

puisse pas l'attaquer et rendre les données indisponibles. Aussi notre approche est un moyen d'innover et d'apporter de nouveaux horizons à ce domaine important qui est la santé.

1.4 MÉTHODOLOGIE DE LA RECHERCHE

Pour la bonne réalisation de notre projet d'études, nous avons suivi une méthodologie divisée en trois étapes clés, effectuées sur une période de 12 mois. Dans la première étape, nous nous sommes attardés sur la revue littéraire existante afin d'avoir de bonnes informations sur les concepts de notre sujet à savoir les différents modèles de réseaux de neurones, leur architecture et leur mode de fonctionnement. Cet état de l'art s'est aussi intéressé aux différentes technologies utilisées à l'heure actuelle dans le domaine des habitats intelligents de même qu'aux différents types de capteurs ambiants présents dans ces habitats. Cette phase a été très bénéfique car elle nous a permis de recenser dans la littérature scientifique des travaux qui pourraient être utiles pour l'implémentation de notre solution de sécurisation de données.

La deuxième étape, quant à elle, a consisté à l'implémentation de plusieurs algorithmes d'encodage et de décodage de données basés sur les architectures de réseaux de neurones. Cette implémentation s'est faite sur un ordinateur local à l'aide de la librairie KÉRAS et avec le langage de programmation Python. Ces algorithmes ont été testés puis réadaptés selon nos propres besoins. Conjointement, nous avons implémenté des algorithmes de cryptographie traditionnelles puis nous avons effectuer des tests d'encodage et de décodage sur nos données avec ces algorithmes.

Pour la troisième étape, nous avons fait une comparaison de nos résultats d'encodage et de décodage des autoencodeurs (AE) selon des mesures de performance que nous verrons dans une des sections du document avec ceux des algorithmes de cryptographie traditionnelle (HMAC-SHA256, SHA-256, AES-256 et Blowfish).

1.5 ORGANISATION DU DOCUMENT

Ce mémoire est composé de 4 chapitres. Le premier chapitre aborde les les préalables de notre projet de recherche. En effet, nous faisons dans ce chapitre une description du contexte de notre sujet, nous introduisons la problématique et nous expliquons la méthodologie de recherche.

Le deuxième chapitre fait cas de la revue de littérature qui nous a permis d'avoir des connaissances sur notre sujet et de faire un tour d'horizon sur les différentes approches de sécurisation de données utilisées par la communauté scientifique. Dans ce chapitre, nous avons étudier deux methodes de chiffrement de données à savoir, le chiffrement des données à travers la cryptographie traditionnelle et le chiffrement des données avec les autoencodeurs. Cette étude a consisté à expliquer l'architecture de chacune de ces deux méthodes, leur fonctionnement ainsi que leurs applications.

Le troisième chapitre décrit toutes les différentes implémentations et tests réalisés pendant notre recherche. Pour commencer, nous avons fait une description de notre cadre expérimental à savoir le LIARA et nous avons décrit les différents types de données collectées par notre environnement intelligent tout en préparant en parallèle notre jeu de données qui a servi pour nos différentes expérimentations. Par la suite, nous avons implémenter notre solution de sécurisation des données en développant nos modèles d'autoencodeurs. Enfin, nous avons effectué des tests d'encodage et de décodage des données du LIARA avec nos AE en utilisant des mesures de performances comme la similarité cosinus, le temps de calculs des données pour juger de la qualité de notre solution.

Le quatrième et dernier chapitre présente une conclusion des travaux effectués. En effet, il a servi à faire une conclusion sur nos résultats et notre méthode. Il nous a permis de décrire

l’apport de cette étude tout en y expliquant les limites et faisant des propositions de travaux futurs pour parfaire ce travail.

CHAPITRE II
REVUE DE LA LITTÉRATURE

La sécurisation des données existe depuis de nombreuse années (Ghernaouti, 2013), l'un des objectifs de ce chapitre est de faire le tour de certaines méthodes ou techniques qui ont été développées et que nous considérons pertinentes par rapport à notre problématique de recherche. Parmi ces méthodes, il y a le chiffrement des données.

Notre Section 2.1 sera donc dédiée à l'explication du fonctionnement du chiffrement, des différents types de chiffrement et des algorithmes populaires. Dans cette même section, nous verrons des cas appliqués de cette technique et nous verrons quelques limites qui nous conduisent à notre méthode : la sécurisation de données par les autoencodeurs.

Puis, nous ferons le tour sur les différents aspects de cette approche en passant par l'architecture des réseaux de neurones et leur fonctionnement, les différents types d'autoencodeurs.

La Section 2.2 sera consacrée aux applications des autoencodeurs et aux différentes approches de la revue scientifique basée sur le chiffrement des données en utilisant les Réseaux de neurones (NN).

2.1 DÉFINITIONS

Le NIST95 (Computer Security Handbook) (Singh, 2013) définit la sécurité informatique comme les efforts déployés pour maintenir un système d'information en fonctionnement et à l'abri des attaques extérieures. Cela inclut la protection des logiciels, du matériel, des données et des télécommunications. Ces efforts permettront d'atteindre des objectifs applicables

de préservation de l'intégrité, de la disponibilité et de la confidentialité des ressources du système d'information. Selon le domaine d'application de la sécurité informatique, on peut les regrouper en quatre comme nous le montre la Figure 2.1.

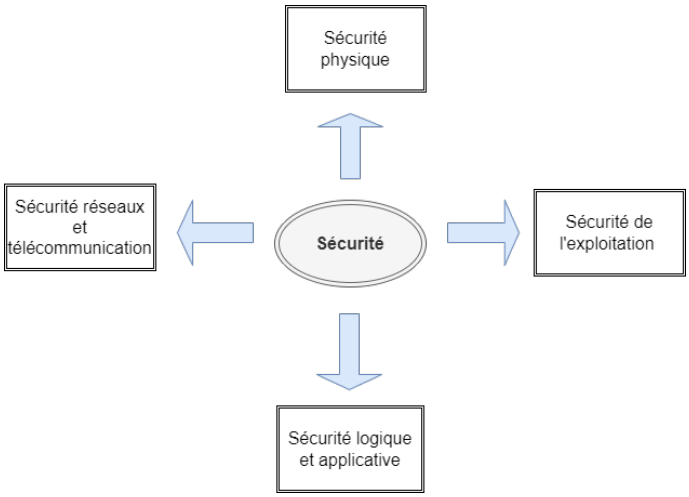


FIGURE 2.1 : Les domaines d'application de la sécurité

©Bégué Hadja Rahinatou Forgo

Dans notre contexte spécifique, bien que toutes les parties de la sécurité informatique soient importantes, nous choisissons de nous concentrer sur la sécurisation des données qui transitent sur le réseau.

Pris à part, un habitat intelligent peut être protégé par isolement. Cependant, dès le moment où l'on doit utiliser des outils pour suivre à distance les résidents de ces habitats intelligents, il faut nécessairement faire transiter les informations d'un point A (la source) vers un point B (la destination). Pendant ce transport, les données peuvent être interceptées, interrompues, modifiées et même fabriquées. Il faut aussi noter que même à l'intérieur d'un réseau local, les données peuvent aussi être modifiées ou transformées à des fins malintentionnées, par exemple, si un utilisateur malveillant parvient à se brancher sur le réseau ou accède à une machine de façon illicite.

Pour éviter une quelconque atteinte à l'intégrité des données on utilise donc des méthodes et techniques pour sécuriser les transitions d'informations. Une des alternatives/méthodes pour offrir la protection nécessaire contre les intrus de données est la cryptographie. La cryptographie est l'une des principales techniques de sécurité informatique qui convertit les informations de leur forme normale en une forme illisible en utilisant des techniques de chiffrement (Hemamalini *et al.*, 2016). Comme l'a mentionné Stallings (2002), « Le chiffrement est de loin l'outil le plus important pour la sécurité des réseaux et des communications ».

2.2 LE CHIFFREMENT DES DONNÉES

Les technologies de chiffrement protègent de nombreux sites Web et applications que les utilisateurs utilisent fréquemment, tels que les courriels, les applications d'achat en ligne, les sites Web bancaires et la messagerie instantanée sécurisée. En effet, ils permettent aux données d'être envoyées, la plupart du temps, en toute sécurité en transit ou au repos. De nombreux nouveaux systèmes d'exploitation, appareils mobiles et services cloud utilisent le chiffrement comme moyens de sécurisation des données. Selon Ghernaouti (2013), le chiffrement est l'opération par laquelle on chiffre un message, une information en les brouillant pour qu'elles deviennent illisibles. C'est une opération de codage. Chiffrer un message permet de la rendre incompréhensible en l'absence d'un décodeur particulier. Autrement dit, le chiffrement des données est une série de transformations utilisant des formules mathématiques et des algorithmes pour convertir le texte brut (les données en langage clair) en texte chiffré à l'aide de la clé de chiffrement en empêchant les personnes non autorisées et ne possédant pas la clé de déchiffrement correcte d'accéder à l'information. Par exemple, Aniisah souhaite envoyer un message à Nouno. Afin d'empêcher quiconque de lire son message privé, Aniisah utilise une clé secrète pour chiffrer les informations. Ce processus rend l'information inaccessible à tout le monde sauf à Nouno qui possède la clé secrète pour déchiffrer le message reçu.

Pendant ce moment, Yacine tente intentionnellement d'analyser le message codé. Cependant, sans accès à la clé, ses efforts seront vains. Même si elle obtient une copie du message, elle ne pourra pas le lire sans entrer en possession de la clé secrète. Tout système de chiffrement pratique se compose de cinq éléments de base qui sont :

— le texte en clair : le message d'origine, celui que l'on veut chiffrer ;

— l'algorithme de chiffrement qui grâce à des moyens mathématiques génèrent des données incompréhensibles appelées texte chiffré ;

— le texte chiffré : ces données peuvent alors transiter sur un réseau non sécurisé ;

— l'algorithme de déchiffrement qui sera la fonction miroir de l'algorithme de chiffrement. Il transforme le texte chiffré avec la clé et redonne le texte en clair ;

— la clé (ou les clés) : utilisée par l'algorithme de chiffrement et de déchiffrement.

En ce qui concerne le chiffrement, si une clé est codée sur n bits (la taille de la clé), elle peut prendre 2^n valeurs. Plus la clé est longue, plus le nombre de clés possibles est élevé. Donc pour ceux qui veulent le comprendre, il faut plus de puissance et de temps de calcul à un tier parti malicieux qui souhaiterait décoder le contenu chiffré en force brute. Ainsi, la clé de chiffrement/déchiffrement doit avoir une taille minimale pour éviter qu'elle ne soit trop facilement déterminée par une entité non autorisée hostile (Ghernaouti, 2013).

Il y a deux classes de systèmes de chiffrement que l'on utilise le plus souvent : les systèmes de chiffrement symétriques ou conventionnels et les systèmes de chiffrement asymétriques ou à clé publique.

2.2.1 CHIFFREMENT SYMÉTRIQUE OU CONVENTIONEL

Le chiffrement symétrique utilise la même clé pour le chiffrement et le déchiffrement. Comme mentionné précédemment, le chiffrement symétrique utilise une clé unique pour

chiffrer et déchiffrer les données. Les clés de chiffrement et de déchiffrement restent secrètes et ne sont connues que de l'expéditeur et du destinataire autorisés qui souhaitent communiquer (Mushtaq *et al.*, 2017). Formellement, le chiffrement symétrique repose sur l'équation (2.1) (Mbuyamba *et al.*, 2023).

$$Y = E[K, X] \tag{2.1}$$

Elle traite les informations de clé de longueur k bits (K) et les informations de texte en clair (X). L'algorithme de chiffrement (E) renvoie le texte chiffré (Y).

Pour le déchiffrement l'opération inverse du chiffrement, elle est obtenue en prenant le texte chiffré et la clé comme entrée pour récupérer le texte en clair initial. L'équation (2.2) du déchiffrement symétrique se traduit par :

$$X = D[K, Y] \tag{2.2}$$

La Figure 2.2 nous décrit le principe du chiffrement symétrique.

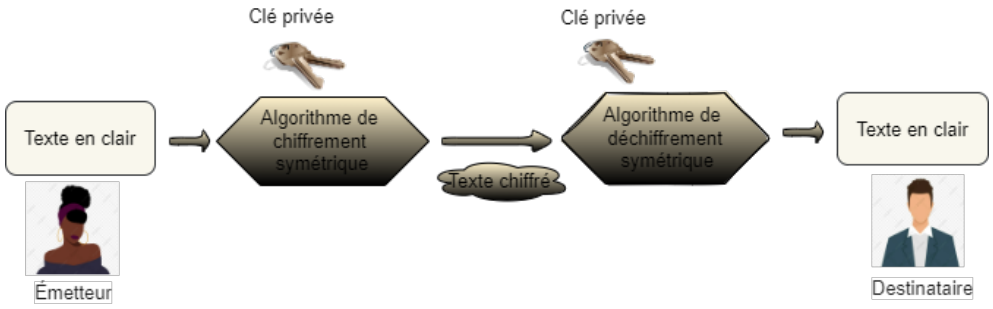


FIGURE 2.2 : Principe du chiffrement symétrique ou conventionel

©Bégué Hadja Rahinatou Forgo

Le chiffrement symétrique est la technique la plus ancienne et la plus connue (Hemamalini *et al.*, 2016). Il peut être classé en chiffrement par bloc et par flux sur la base du regroupement des bits de message (Stallings, 2006; Chandra *et al.*, 2014).

— Dans un chiffrement par bloc, un ensemble de caractères de message de taille fixe (un bloc) est chiffré et envoyé au destinataire en une seule fois. En outre, le chiffrement par bloc peut être classé en chiffrement par bloc binaire et en chiffrement par bloc non binaire en fonction du résultat final des messages, des clés et des textes chiffrés. Les chiffrements par bloc binaire ont des tailles de bits de message de 64, 128, 192 et 256 et le chiffrement par bloc non binaire n'a pas défini la norme qui dépend de l'implémentation du chiffrement. Ce type de chiffrement est implémenté grâce au réseau de Feistel (Maqsood *et al.*, 2017). Le réseau de Feistel est une construction qui s'appuie sur des principes simples d'opérations répetées de permutations et de substitutions des blocs de données.

— Chiffrement de flux (flot) : les chiffrements de flux utilisent une clé symétrique qui utilise du texte en clair combiné à un flux de chiffres de chiffrement pseudo-aléatoire également appelé flux de clés. Il chiffre le texte en clair un à la fois en utilisant le flux de clés correspondant. La sortie sera le flux de texte chiffré correspondant. Il est aussi connu sous le nom de chiffrement d'état car chaque chiffre dépend de l'état actuel du chiffrement. Un chiffre sera un bit et l'opération de combinaison utilisera l'opération XOR. Les flux de clés pseudo-aléatoires sont généralement créés à partir d'une valeur de départ aléatoire qui utilise des registres à décalage numériques. La valeur de départ fonctionnera également comme clé pour déchiffrer le flux de chiffrement. À l'opposé des chiffrements par blocs, les chiffrements par flux représentent une approche différente du chiffrement et du déchiffrement des informations. Un chiffrement de flux nécessite

des graines uniques pour éviter d'être piratés sinon un adversaire pourrait déchiffrer le code.

2.2.2 CHIFFREMENT ASYMÉTRIQUE

En ce qui concerne le chiffrement asymétrique, il utilise deux clés différentes une clé privée et une clé publique pour sécuriser les communications (Naureen *et al.*, 2008). La clé publique pour faire le chiffrement peut être révélée à toute personne qui souhaite chiffrer un message et la clé privée, impossible à dériver de la première, utilisée pour le déchiffrement est détenue individuellement et doit rester secrète. Il faudra noter que la donnée de la clé publique seule ne permet pas de trouver la clé de déchiffrement. La Figure 2.4 nous montre le schéma d'un chiffrement asymétrique :

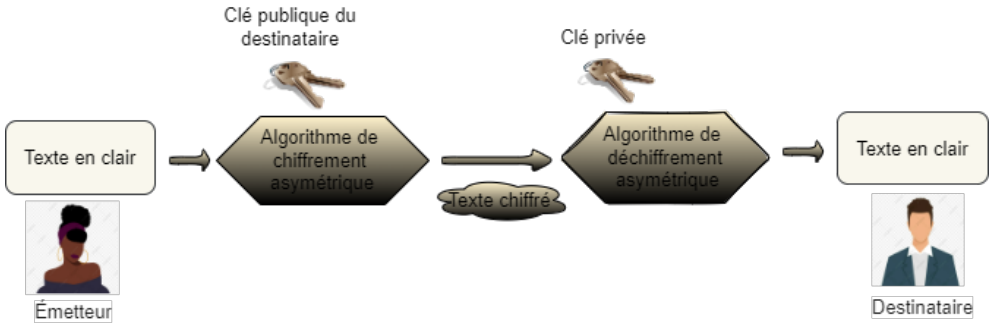


FIGURE 2.3 : Principe du chiffrement asymétrique

©Bégué Hadja Rahinatou Forgo

Dans le chiffrement asymétrique, la taille de clé est plus grande par rapport à la taille des clés dans le chiffrement symétrique ce qui rend le chiffrement symétrique moins sûr pour les données plus sensibles (Padmavathi & Kumari, 2013). De plus, le temps de calcul du chiffrement asymétrique est supérieur à celui du chiffrement symétrique ce qui rend le

chiffrement/ déchiffrement plus complexe pour une grande quantité de données (Singh, 2013; Patil & Goudar, 2013).

La force du chiffrement à clé asymétrique est utilisée avec la signature numérique, puis elle peut être fournie aux utilisateurs via la détection d'authentification de message (Maqsood *et al.*, 2017).

2.3 LES ALGORITHMES DE CHIFFREMENT DE DONNÉES

Les algorithmes ont des conceptions différentes en fonction de leur utilisation de la cryptographie (asymétrique ou symétrique). La cryptographie asymétrique se concentre également sur des problèmes mathématiques difficiles tels que les problèmes de réseau, le logarithme discret sur un champ fini. La cryptographie symétrique quant-à elle repose sur des calculs et des tables mathématiques pour créer un chiffrement sécurisé. Sa sécurité découle des concepts de confusion et de diffusion de Claude Shannon (Shannon, 1949), qui supposent un échange sécurisé entre clés avant utilisation. Dans la cryptographie symétrique, l'arithmétique de base et les tables de recherche sont utilisées pour plus d'efficacité. Bien que cette approche fonctionne en logiciel et en matériel, elle suppose au préalable un échange sécurisé de la clé pour le chiffrement et le déchiffrement.

Dans la suite de notre travail, nous verrons quelques-uns de ces algorithmes de chiffrement à savoir AES, Blowfish , SHA et HMAC-SHA-256.

2.3.1 AES

La norme avancée de chiffrement AES (Advanced Encryption Standard) est une norme de chiffrement utilisée pour la sécurité de l'information. Son développement a été initié par le NIST (National Institutes of Health Normes et Technologie) qui a lancé en 1998 un appel

d'offres public pour remplacer le DES (Data Encryption Standard). À la suite des différents tests, c'est l'algorithme Rijndael qui a été choisi (Dumont, 2006). Rijndael est une combinaison du nom de ses créateurs Joan Daemen et Vincent Rijmen (Daemen & Rijmen, 2002). De par ses origines, AES est gratuit et ne comporte aucune restrictions d'utilisation ni de brevets. AES est basé sur un algorithme de type symétrique, utilisant le chiffrement par bloc. Les données sont traitées par blocs de 128 bits pour le texte clair et celui chiffré. Autrement dit, il prend un bloc d'entrée d'une certaine taille, habituellement 128 bits, et produit un bloc de sortie correspondant de même taille. La transformation exige une deuxième entrée, qui est la clé secrète. Il est important de connaître cette clé secrète. L'algorithme peut être employé avec les trois longueurs de clés différentes 128, 192, 256 et peut être désigné sous le nom de AES-128, de AES-192, et de AES-256 dépendamment de la clé utilisée. Pour chacun d'eux, la taille de bloc (données) d'entrée, de l'état et de sortie est toujours de 128 bits. Elle est représentée ici par 4, qui correspond au nombre de mots de 32 bits (ou nombre de colonnes) dans l'État. AES opère sur des blocs de 128 bits qu'il transforme en blocs chiffrés de 128 bits par une séquence de Nr opérations ou rounds, à partir d'une clé de 128, 192 ou 256 bits. Suivant la taille de celle-ci, le nombre de rounds diffère : respectivement 10, 12 et 14 rounds et qui dépend à la fois de la taille des blocs et de la clé. La longueur de clé, la taille du bloc et le nombre de tours pour les trois cas de l'algorithme sont présenté dans le Tableau 2.1.

TABLEAU 2.1 : Étude comparative des caractéristiques fonctionnelles des variantes de l'algorithme AES.

Algorithme	Longueur de clé	Taille du bloc (32bits)	Nombre de tours
AES-128	128	128	10
AES-192	192	128	12
AES-256	256	128	14

2.3.2 BLOWFISH

Blowfish est un algorithme de chiffrement par bloc symétrique basé sur la fonction Feistel (Alabaichi *et al.*, 2013) et utilisé pour le processus de chiffrement et de déchiffrement. Il a été conçu en 1993 par Bruce Schneier (Schneier, 1994). La plupart des algorithmes de chiffrement ne sont pas accessibles au public et la plupart d'entre eux sont protégés par des brevets. Blowfish est rapide, n'ayant pas de licence, non breveté, librement disponible et une alternative aux algorithmes de chiffrement existants.

Blowfish est un algorithme à usage général et il utilise une clé avec une longueur variant entre 32 bits à 448 bits. Il utilise 16 cycles pour le processus de chiffrement. Cet algorithme a été considérablement analysé et avec le temps, il gagne en popularité en tant que algorithme de chiffrement par blocs robuste (Masram *et al.*, 2014). L'entrée sous forme de texte en clair est une donnée de 64 bits.

L'algorithme Blowfish nécessite plus de temps de traitement car il dépend de la taille de la clé. Aussi la complexité supplémentaire du processus de génération de sous-clés protège contre les attaques par force brute et offre une meilleure sécurité que les techniques de chiffrement existantes.

2.3.3 SHA

SHA(Secure Hash Algorithm) a été introduit par l'American National Institute for Standards and Technology et publié en tant que norme FIPS (Federal Information Processing Standard) en 1993. Il est connu sous le nom de SHA-0 et appartient à la famille des fonctions de hachage MD (Message Digest). Après son introduction, il a subi une modification et amélioration qui l'a valu d'être publié en 1994 en tant que SHA-1 (National Institute of Standards and Technology (NIST) FIPS Publication 180-1 : secure Hash Standard, April 1994).

Par la suite, en 2000, une nouvelle génération de SHA(SHA-2) a été introduite pour augmenter le rendement et changer la portée de certaines conceptions subtiles. Elle a été adopté comme une norme FIPS en 2002. Elle marche avec des tailles de résumé de message plus grands comme 256, 384 et 512 bits. Il sont appelés respectivement SHA-256, SHA-384 et SHA-512. La taille du bloc de message de ces algorithmes de hachage sécurisé dépendrait donc de l'algorithme. Par exemple pour SHA-224 et SHA-256, chaque bloc de message a 512 bits, qui est représenté comme une séquence de seize mots de 32 bits. Pour SHA-384 et SHA-512, chaque bloc de message a 1024 bits, représenté comme une séquence de seize mots de 64 bits. On peut voir cela dans le Tableau 2.2.

TABLEAU 2.2 : Étude comparative des caractéristiques fonctionnelles des variantes de l'algorithme SHA-2.

Algorithme	Taille du message	Taille du bloc (bits)	Complexité de la meilleure attaque	Taille de mots	Taille du résumé du message
SHA-224	2^{64}	512	2^{112}	32	224
SHA-256	2^{64}	512	2^{128}	32	256
SHA-384	2^{128}	1024	2^{192}	64	384
SHA-512	2^{128}	1024	2^{256}	64	512

Par la suite il y a eu SHA-3, également connu sous le nom de KECCAK qui est un algorithme de hachage cryptographique. En effet, KECCAk a remporté le concours de fonctions de hachage du NIST et est proposé comme norme SHA-3 (Dworkin, 2015). Il ne remplace pas SHA-2 qui demeure une méthode sûre mais est conçu pour résister à toutes les attaques cryptographiques connues.SHA-3 comprend quatre fonctions de hachage cryptographique SHA3-224, SHA3-256, SHA3-384 et SHA3-512. Il est aussi composé de deux fonctions à sortie extensible (XOF), appelées SHAKE128 et SHAKE256.

Chaque algorithme peut être décrit en deux étapes : le prétraitement et le calcul du hachage. Le prétraitement comprend le remplissage du message, l'analyse du message rempli en morceaux de m bits et la définition des valeurs d'initialisation à utiliser pour les calculs de hachage. Le calcul de hachage génère un programme de messages à partir du message

rembourré et utilise ce programme avec des fonctions, des constantes et des opérations sur les mots pour générer de manière itérative une série de valeurs de hachage.

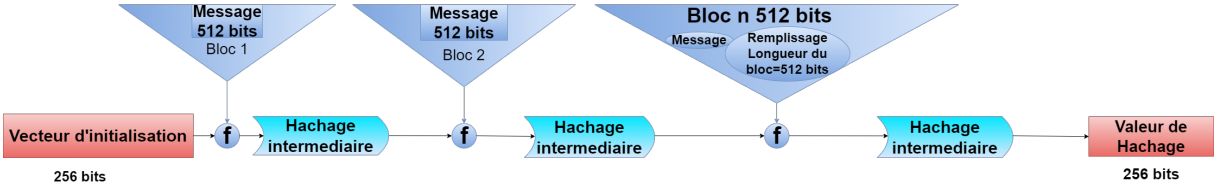


FIGURE 2.4 : Fonctionnement de SHA-256

©Bégué Hadja Rahinatou Forgo

2.3.4 HMAC/SHA-256

L'algorithme HMAC (Hash-Based Message Authentication Code) traite deux données d'entrée qui sont une clé cryptographique et un message dans le but de produire un code d'authentification (MAC). On peut combiner le HMAC avec le SHA-2. Vu qu'on a plusieurs algorithmes regroupés dans le SHA-2, on aurait des combinaisons comme HMAC/SHA-224, HMAC/SHA-256, HMAC/SHA-384, et HMAC/SHA-512 (Turner, 2008).

Le mécanisme d'authentification de message HMAC-SHA est une fonction de hachage cryptographique. C'est une très bonne application pour l'authentification et l'intégrité des données pour une meilleure sécurité du réseau. HMAC/SHA-256 est un algorithme de hachage à clé basé sur la fonction de hachage SHA-256 utilisée comme code d'authentification de messages basé sur le hachage (HMAC). Le processus HMAC mélange la clé avec les données du message, hache le résultat à l'aide d'une fonction de hachage, mélange à nouveau la valeur de hachage avec la clé, puis applique à nouveau la fonction de hachage. Le hachage de sortie a une longueur de 256 bits. En d'autres termes, HMAC/SHA-256 accepte les clés de n'importe quelle taille et produit une séquence de hachage de 256 bits de long.

2.3.5 LIMITES DES ALGORITHMES DE CHIFFREMENT

Plusieurs travaux de recherche se sont concentrés principalement sur l'utilisation d'AES comme algorithme de chiffrement et de déchiffrement (Kumar & Karthikeyan, 2012; Mahajan & Sachdeva, 2013; Saraf *et al.*, 2014). Parmi ces travaux, Zhang *et al.* (2017) ont présenté une méthode de chiffrement rapide basée sur la cryptographie AES pour la sécurité des images. L'algorithme se déroule comme suit, premièrement l'image est découpée en carrés de taille 128 bits. Les carrés de l'image d'origine sont arrangés par un vecteur sous-jacent puis brouillés à l'aide de l'algorithme AES. Les résultats des tests montrent qu'il y a une amélioration de la vitesse et de la robustesse par rapport aux autres.

Saraf *et al.* (2014) implémente le chiffrement et le déchiffrement de textes et d'images à l'aide d'AES. Si les images ont une grande taille de données et ont également un problème de contrainte en temps réel, une méthode similaire ne peut pas être utilisée pour protéger les images ainsi que le texte contre tout accès non autorisé. Cependant, avec quelques variantes de méthode, AES peut être utilisé pour protéger l'image ainsi que le texte. Les différentes approches font appel à AES avec cryptographie visuelle qui offre de bons résultats en chiffrant l'image à l'aide d'AES et la clé d'origine à l'aide de cryptographie visuelle en la convertissant en image. Malgré cela, l'algorithme est toujours susceptible d'être attaqué sur l'image partagée créée pour la clé (Chowdhary *et al.*, 2015; Jin *et al.*, 2020).

Dans la même lancée, Arab *et al.* (2019) ont proposé les techniques AES et Visual Cryptographic pour les images. Bien que l'algorithme AES a l'air de mieux fonctionner que d'autres méthodes, il n'incluait pas de résultats concrets pour la même chose grâce à une variété de mesures de sécurité et de qualité de chiffrement, telles que l'entropie, le NPCR (Pixel Change Rate), l'Unified Average Change Intensity (UACI) (Mortajez *et al.*, 2020). Toujours dans l'optique de sécuriser les données, Hamad *et al.* (2013) sont allés loin en travaillant

pour augmenter la protection du chiffrement des données d'images en utilisant le chiffrement Playfair standard en utilisant une clé modifiée de taille 16 par 16 sur la plage de pixels de 8 bits. En effectuant une opération XOR à l' aide d'un masque aléatoire, les effets sont encore améliorés. Mais la sécurité supplémentaire de l'algorithme à travers la fonction XOR échoue si l'intrus écoute le masque. Un algorithme incorporant le chiffrement XOR avec un processus de rotation a été conçu pour chiffrer efficacement les images. Aussi, Chowdhary *et al.* (2020) ont travaillé sur la sécurité du chiffrement de texte fourni par Double Playfair Cipher en utilisant des matrices de clés 6×6 sur les matrices de clés 5×5 habituelles. Cependant, l'algorithme a échoué en raison de la perte de données sur certains caractères, tels que les espaces et les symboles spéciaux. Une version mise à jour du chiffrement Playfair 5×5 est présentée, ce qui permet à l'utilisateur de chiffrer et de déchiffrer des messages pour n'importe quelle matrice carrée. Le chiffrement fair-play, avec les instructions de codage uniques, est introduit comme premier chiffrement digraphique. Hardi *et al.* (2018) ont combiné l'utilisation du cryptosystème ElGamal et du chiffrement Double Playfair pour protéger les données textuelles en utilisant des clés standard pour la méthode symétrique. Bien que l'algorithme semble fonctionner sur des supports numériques, il ne parvient pas à évaluer les métriques et les mesures de sécurité. D'un autre côté, Hashim & Neamaa (2014) ont utilisé ElGamal comme algorithme de chiffrement asymétrique et ont effectués différents tests avec MATLAB. Les travaux ont conclu qu'il fallait de plus en plus de temps pour le calcul en utilisant un grand nombre premier comme paramètre de chiffrement.

Ces quelques travaux de la littérature scientifique nous montrent que les algorithmes de chiffrement ne sont pas parfaits, ils présentent malheureusement des faiblesses. Si on s'attarde sur les algorithmes de chiffrement symétriques, on voit que le problème de la gestion et de la diffusion des clés secrètes se pose. En effet, dans ce cas de figure, la sécurité du chiffrement dépend du secret de la clé (Stallings, 2002). Si une personne non autorisée entre en possession

de cette clé, elle pourrait aisément lire, modifier les données et ainsi porter atteinte à la vie privée ou professionnelle d'autrui et bien plus. De plus, dans ce type de chiffrement il y a une seule clé pour chiffrer et déchiffrer les données. Pour finir le chiffrement symétrique n'assure que la confidentialité des données, contrairement au chiffrement asymétrique.

Et pour finir, les cryptosystèmes à clé publique (chiffrement asymétrique) sont vulnérables aux attaques à texte en clair choisi et sont lents. Cela s'explique par le fait qu'un algorithme se renforce plus quand l'on allonge la clé de chiffrement qui lui est associée. Ainsi plus sa clé est longue, plus il sera résistant aux attaques par force brute. Les algorithmes à clé publique ont de longues clés mais nécessitent plus de puissance de calcul, ce qui ralentit le chiffrement et le déchiffrement par rapport aux clés courtes.

Dans nos habitats intelligents nous utilisons des données de radars UWB qui requièrent de grandes quantités de calculs. Au vu de cette réalité, on se rend compte que les algorithmes de chiffrement ne sont pas nécessairement la méthode de sécurisation des données qui nous conviendrait le mieux. Ces limitations nous amènent à cette tendance qui est la sécurisation des données basée purement sur les réseaux de neurones (Gupta, 2020) plus précisément avec les autoencodeurs dans notre cas.

2.4 RÉSEAUX DE NEURONES

Au cours des dernières années, les réseaux de neurones ont fait des avancées spectaculaires en ce qui concerne les problèmes complexes comme pour les problèmes de vision telle la reconnaissance d'objets (Voulodimos *et al.*, 2018). Comme pour les problèmes de vision, les réseaux de neurones semblent être une bonne méthode pour l'exploitation des données d'images, de caméras thermiques, de vidéos, de radars UWB puisqu'ils peuvent extraire automatiquement les informations pertinentes d'un ensemble de données.

Les réseaux de neurones artificiels (ANN) désignent généralement des réseaux neuro-mimétiques résultat de l'interconnexion d'un ensemble de neurones artificiels (Solaiman & Richard, 2003). En effet, ils consistent à s'inspirer du fonctionnement du neurone biologique pour construire des modèles capables d'imiter certaines des capacités de l'humain. Il est similaire au cerveau de deux manières : la première du fait que le réseau acquiert la connaissance à travers un processus d'apprentissage et la seconde par le fait que les forces de connexion des interneurones appelées poids synaptiques sont utilisées pour le stockage des informations. Pour Borne *et al.* (2007), les réseaux de neurones sont :

— massivement parallèles ;

— capable de traiter des informations incomplètes ;

— capable d'apprentissage ;

— capable de mémoriser l'information dans les connexions inter-neurones.

Parallèlement au neurone biologique, un neurone artificiel reçoit sous forme vectorielle des entrées x_1, x_2,...x_k ensuite il effectue une combinaison affine de ses entrées moins et y ajoute le seuil d'activation du biais $z = w_1x_1 + w_2x_2 + ... + w_kx_k + b$. Après une fonction d'activation f est appliquée sur cette sortie :

$f(x,w) = b + \sum_{k=1}^{p} w_k x_k$ dans le but de créer une sortie y.

Les paramètres w_1,w_2,...,w_k representent les poids et b représente le biais du neurone. Si un w_i est positif, l'entrée w_i est excitatrice alors que si w_i est négatif, elle est inhibitrice (SOW, 2020). Le neurone artificiel peut être représenté comme suit dans la Figure 2.5.

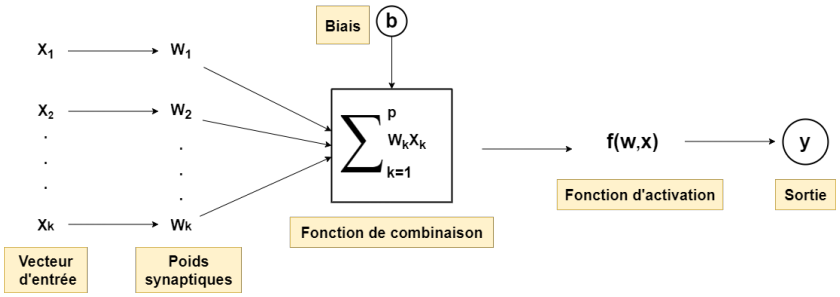


FIGURE 2.5 : Un neurone artificiel

©Bégué Hadja Rahinatou Forgo

Un réseau de neurones est constitué en reliant plusieurs neurones simples, de sorte que la sortie d'un neurone peut être l'entrée d'un autre.

2.4.1 ARCHITECTURE ET FONCTIONNEMENT

Essentiellement, les réseaux de neurones effectuent un apprentissage de représentation, où chaque couche du réseau de neurones apprend une représentation couche par couche. Ils peuvent effectuer des tâches assez étonnantes comme la conception vidéo, la reconnaissance vocale, la reconnaissance des activités, l'encodage et le décodage des données. Le réseau de neurones fait référence à l'interconnexion entre les neurones de différentes couches du système. La première couche a des neurones d'entrée qui envoient les données à la deuxième couche, puis à travers des synapses à la troisième couche de neurones de sortie. Chaque couche essaie de comprendre comment la couche précédente est liée à la couche de sortie. Un système plus complexe aura plus de couches de neurones. Les synapses stockent des paramètres appelés poids qui manipulent les données dans les calculs. Un réseau de neurones est généralement défini par trois types de paramètres (Kinzel & Kanter, 2002) :

- ☐ le Schéma d'interconnexion entre les différentes couches de neurones.
- ☐ le Processus d'apprentissage pour une mise à jour des poids de l'interconnexion.

- la Fonction d'activation qui convertit un neurone d'entrée pondérée à son activation de sortie.

Les types de réseaux de neurones vont de ceux qui n'ont qu'une ou deux couches de logique unidirectionnelle aux boucles et couches de rétroaction directionnelles à plusieurs entrées complexes. Dans l'ensemble, ces systèmes utilisent des algorithmes dans leur programmation pour déterminer le contrôle et l'organisation de leur fonction. La plupart des systèmes utilisent des poids pour modifier les paramètres du débit et les différentes connexions aux neurones. Les réseaux de neurones peuvent être autonomes et apprendre par l'apport d'un enseignant extérieur ou même par auto-apprentissage (Charniya, 2013). Dans les prochaines lignes nous verrons un type de réseau de neurones utilisant l'apprentissage non-supervisé : les autoencodeurs.

2.4.2 AUTOENCODEURS

L'autoencodeur a été développé à la fin des années 1980 et constitue depuis lors une partie importante du paysage historique des réseaux de neurones (Sanger, 1989). L'autoencodeur est un réseau de neurones qui fait partie d'une famille plus large de méthodes d'apprentissage de représentations capables d'apprendre automatiquement des caractéristiques à partir de données non étiquetées. Pour Goodfellow *et al.* (2016) les autoencodeurs consistent en une structure de réseau neuronal entraînée pour générer une copie de sortie des données d'entrée. En effectuant cette opération, ils transforment les données d'entrée au niveau des couches cachées aux représentations de dimension inférieure et assurent la reconstruction sur ces caractéristiques. Autrement dit, un autoencodeur comprime le vecteur d'entrée en un vecteur de dimension inférieure, qui forme la représentation dense des données d'entrée. Cette capacité de l'autoencodeur peut être utilisée pour compresser des images et ensuite les chiffrer. La procédure de chiffrement est appliquée aux données compressées.

L'autoencodeur est une combinaison de trois blocs qui sont :

1. L'encodeur : cette partie du réseau neuronal autoencodeur compresse l'entrée dans une représentation de l'espace latent. Dans la phase d'encodage, le vecteur d'entrée est transformé en représentations cachées. Le rôle de l'encodeur est de prendre une entrée qu'importe le type de données (images, vidéo, capteurs, audio, température, texte) et de la condenser en extrayant du mieux possible les caractéristiques de l'information initiale.
2. L'espace latent : la représentation latente résultante peut ensuite être utilisée pour récupérer des informations pertinentes pour construire des classificateurs ou d'autres prédicteurs, puis régénérer la sortie comme l'entrée.
3. Le décodeur : cette partie de l'autoencodeur rassemble les données générées par l'encodeur qui sont sous forme de représentation d'espace latent.

La Figure 2.6 représente un schéma fonctionnel d'une structure AE standard.

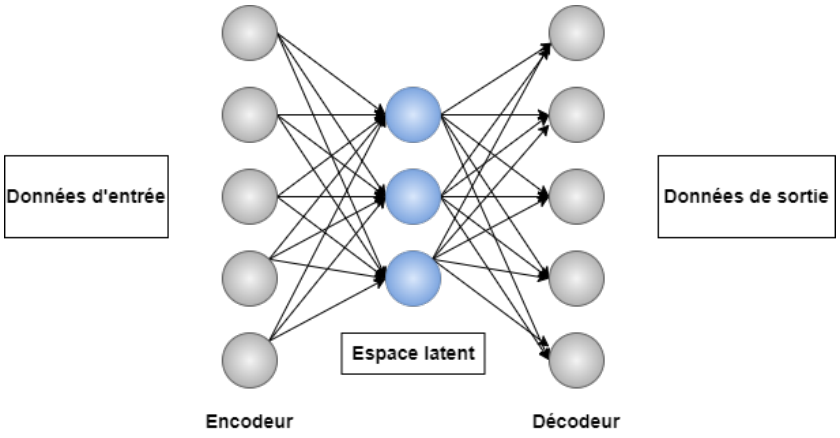


FIGURE 2.6 : La structure générale d'un autoencodeur :la compression des données d'entrée et leur reconstruction à partir de la couche de sortie

©Bégué Hadja Rahinatou Forgo

Dans les autoencodeurs, chaque couche des réseaux neuronaux apprend une représentation des caractéristiques d'origines, et les couches suivantes se basent sur la représentation apprise par les couches précédentes. Couche par couche, l'autoencodeur apprend des représentations de plus en plus compliquées à partir de représentations plus simples, construisant ce que l'on appelle une hiérarchie de concepts de plus en plus abstraits. La couche de sortie est la représentation finale nouvellement apprise des caractéristiques originales. Cette représentation apprise peut ensuite être utilisée comme entrée dans un modèle d'apprentissage supervisé dans le but d'améliorer l'erreur de généralisation. La Figure 2.7 nous montre la structure d'un AE profond.

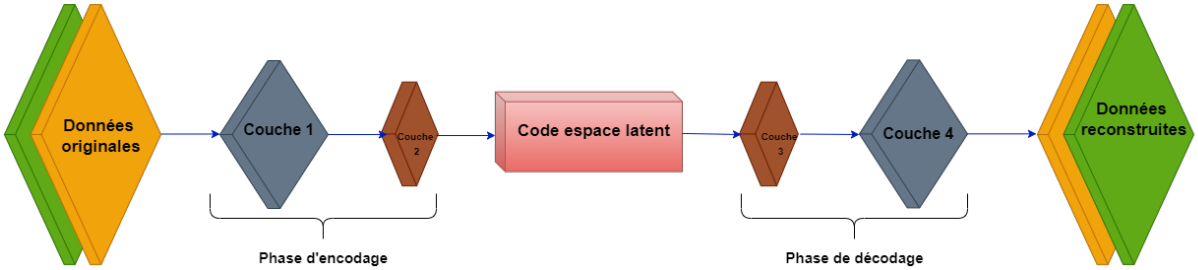


FIGURE 2.7 : Le schéma fonctionnel montrant la hiérarchie de codage et de décodage d'une structure AE profonde simple

©Bégué Hadja Rahinatou Forgo

Les autoencodeurs ont de nombreuses applications : ils sont utilisés pour la réduction de la dimensionnalité (Hinton & Salakhutdinov, 2006), et l'ingénierie/apprentissage automatique des caractéristiques ou encore pour générer ou débruiter des images (Vincent *et al.*, 2008). De plus, ils sont utilisés pour la compression et la recherche d'images, pour la recherche d'anomalies et bien plus. Aujourd'hui, ils sont souvent utilisés pour construire des modèles génératifs tels que les réseaux génératifs adverses (Patel, 2019) et l'encodage et le décodage des données.

2.4.3 AUTOENCODEURS CONVOLUTIONNELS

Un autre type d'AE profond est l'AE convolutif (CAE). Dans les CAE, des couches convolutionnelles sont utilisées dans la section d'encodage au lieu des couches entièrement connectées utilisées dans les AE profonds. Les couches de déconvolution sont utilisées dans la phase de décodage (Turchenko *et al.*, 2017). Les CAE contiennent des couches standard de réseaux neuronaux convolutifs (CNN) ainsi que des couches d'extension telles que le découplage, la déconvolution ou l'échantillonnage ascendant, qui exécutent des fonctions inverses dans la phase de décodage. Ils ont la même configuration que les autoencodeurs simples décrits plus haut. Les étapes d'encodage et de décodage des données sont réalisées à l'aide de réseaux de neurones convolutifs. La structure des deux réseaux est généralement symétrique par rapport à la couche de représentation cachée. La première couche de l'encodeur prend les données brutes en entrée. La Figure 2.8 illustre l'architecture d'un autoencodeur convolutionnel.

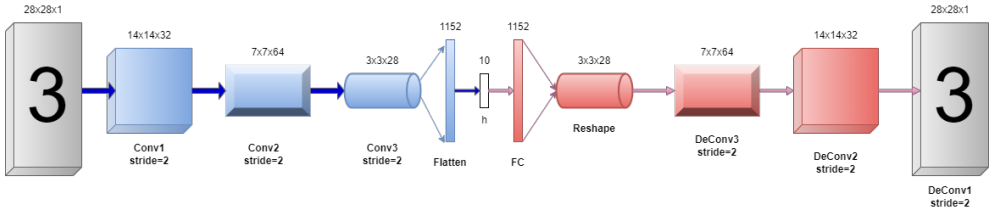


FIGURE 2.8 : Un autoencodeur convolutionnel

©Bégué Hadja Rahinatou Forgo

Les CNN sont particulièrement adaptées aux données sous forme de grille, par exemple, les images. Celles-ci sont d'abord convertis en un tableau de pixels. En termes simples, la convolution est l'acte d'appliquer un filtre mathématique à une image. D'un point de vue plus technique, cela implique de faire glisser une matrice sur l'image. Cette technique nous permet de trouver des parties d'une image qui pourraient nous intéresser. Une couche de

convolution consiste à construire plusieurs cartes de caractéristiques à travers des matrices appelées noyaux. Ces noyaux sont générés par le réseau en tenant compte des caractéristiques comme la forme, la couleur, la texture qu'ils extraient de l'image. Nous choisissons le pas avec lequel nous déplaçons le noyau horizontalement et verticalement à travers les entrants.

La Figure 2.9 nous laisse voir un exemple d'une opération de convolution où un filtre de dimension 3x3 est appliqué à l'image d'entrée. Pour obtenir les valeurs de la carte de caractéristiques on procède comme suit : prenons par exemple le chiffre 0 qui se trouve sur la carte de caractéristique en haut à gauche, elle s'obtient en faisant la somme des produits par correspondance des valeurs entre le filtre et la matrice de l'image et une sous-matrice d'ordre de l'image d'entrée se situant à l'extrémité gauche, en haut de l'image d'entrée. Après avoir défini un pas de longueur 1, nous parcourons l'image d'entrée avec la même série d'actions jusqu'à ce qu'elle soit complètement parcourue. Après cette étape, nous avons ainsi autant de cartes de caractéristiques que de filtres.

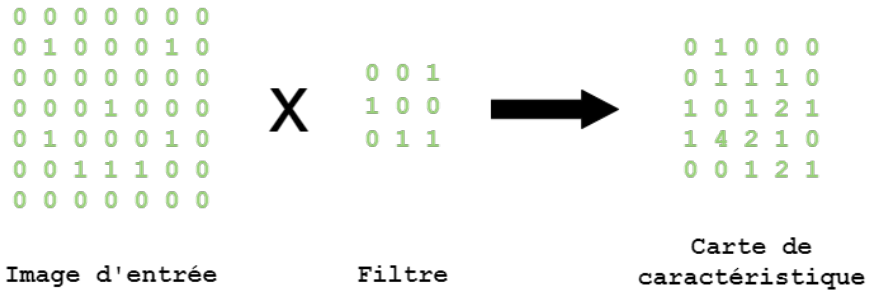


FIGURE 2.9 : Exemple d'opération de convolution

©Bégué Hadja Rahinatou Forgo

2.5 APPROCHES DE CHIFFREMENT BASÉES SUR LES RÉSEAUX DE NEURONES

Les réseaux de neurones profonds ont montré un grand potentiel en tant que méthode de chiffrement et de déchiffrement pour sécuriser les données dans les dernières années. Des techniques telles que les réseaux de neurones de Hopfield, les réseaux de neurones chaotiques retardés (Ameen Suhail & Sankar, 2020), les autoencodeurs et GAN ont montré des résultats encourageants (Gupta, 2020). Malgré cela, il existe encore peu de travaux de recherche sur leur utilisation pour chiffrer les données.

Parmi plusieurs des approches et travaux pour la sécurité des données, on peut mentionner Zhou *et al.* (2014) qui ont proposé un nouvel algorithme hybride pour la compression et le chiffrement, dans lequel la compression et le chiffrement sont effectués en même temps. Dans leur méthode, une image d'entrée a été divisée dabord en quatre blocs puis l'échange des matrices aléatoires de pixels a été introduit pour brouiller les blocs compressés et les chiffrer. À ce niveau, des matrices circulantes ont également été utilisées pour construire la mesure matricielle dans la détection compressive, qui est associée à des matrices aléatoires. La procédure a démontré une performance de chiffrement efficace. Cette méthode permet de surmonter la limitation de la distribution, de la mémorisation et du stockage des grandes clés.

Les algorithmes basés sur le chaos ont été largement adoptés pour chiffrer les images. Dans cette optique, Hu *et al.* (2016) ont proposé un autoencodeur empilé à 5 couches (SAE) et une carte logistique pour chiffrer les images en couleur. Le schéma proposé par les auteurs, utilise deux couches cachées du côté de l'encodeur, ce qui limite le taux de compression. Dans ce schéma, les paramètres initiaux sont calculés à l'aide d'une fonction sigmoïde, ce qui permet de générer une séquence aléatoire qui est mise en XOR avec l'image chiffrée au niveau des bits. Les expériences montrent que cette application est faisable et efficace. Il peut

être utilisé simultanément pour la transmission d'images et la protection d'images sur Internet. Cependant, c'est difficile pour SAE car plusieurs couches cachées consommeraient trop de temps de calcul.

Hu *et al.* (2017) sont allés loin en proposant un schéma de chiffrement d'images par lots tout en y intégrant l'apprentissage profond et le comportement chaotique. Pour ce faire, les auteurs ont construit un autoencodeur avec trois couches cachées et donc deux matrices de poids du réseau d'autoencodeur W0 et W1. Tout d'abord, la matrice chaotique, W0, est utilisée pour générer une matrice de brassage pour la confusion d'image (pour mélanger les positions des pixels sur chaque image individuelle). Ensuite, la matrice W1, est utilisée pour générer un ensemble de séquences chaotiques pour le chiffrement d'images. Ce schéma facilitera la transmission sécurisée d'images distantes en temps réel avec une bande passante minimale. Après cela, l'image est transmise sur le réseau. Ces processus sont à l'opposé de ce qu'ils sont au niveau du récepteur, où l'image reçue est déchiffrée, décompressée et désamalgamée pour récupérer l'image d'origine. Le schéma est efficace en raison des avantages de calcul parallèle de SAE, ce qui se traduit par une réduction significative de la complexité d'exécution. De plus, l'hybridation du brassage et de la confusion augmente l'efficacité du chiffrement. Les résultats expérimentaux et l'analyse démontrent que ce schéma est capable de résister aux attaques par force brute, statistiques et différentielles. Cependant, les auteurs ont mentionné que cette méthode a un inconvénient : toutes les images simples doivent avoir la même taille (pixels dans les expériences). De plus, les autoencodeurs empilés utilisés dans cette méthode pour le chiffrement sont coûteux en calcul pour le codage, ce qui limite leur application et sont moins sécurisés car l'encodeur et le décodeur sont formés en tandem.

Aussi Maniyath & Thanikaiselvan (2020) ont proposé un autoencodeur qui a été utilisé pour générer une clé robuste et a été suivi par l'application de cartes chaotiques pour obtenir l'image chiffrée finale. L'article proposé présente un modèle où l'apprentissage profond-carte

chaotique a été utilisé pour effectuer une meilleure optimisation en vue de renforcer les performances de chiffrement de l'image. Cette technique implique un processus complexe et la distribution uniforme de l'image chiffrée n'est pas prise en compte, ce qui est essentiel pour un excellent algorithme de chiffrement d'image.

Un autre axe des travaux suivants se basent spécialement sur les réseaux d'AE pour chiffrer les données. En effet, Ding *et al.* (2020) ont proposé un nouveau réseau de chiffrement et de déchiffrement d'images médicales basé sur l'apprentissage en profondeur (DeepEDN). En termes plus clairs, le réseau antagoniste génératif de cycle (CycleGAN) sera utilisé pour le transfert des images médicales de leur domaine d'origine vers la cible souhaitée. En effet, DeepEDN se compose principalement de trois sous-réseaux : le réseau de chiffrement G, le réseau de discriminateur D et le réseau de déchiffrement F.

Le réseau de chiffrement G est utilisé pour chiffrer les images d'entrée d'origine. Il commence par une étape de convolution initiale pour sous-échantillonner spatialement et coder les images, et les caractéristiques utiles obtenues à cette étape seront utilisées pour la transformation suivante. Ensuite, le réseau de déchiffrage F, l'opération inverse du chiffrement est responsable de restaurer les images chiffrées à l'original ; il peut être mis en œuvre à la manière d'un débruitage d'image ou d'une reconstruction d'image. Puis, le réseau discriminateur D est principalement conçu pour améliorer les performances du réseau de chiffrement en distinguant les images générées. Dans DeepEDN, les paramètres du réseau de génération sont considérés comme la clé privée pour le chiffrement tandis que les paramètres du réseau de reconstruction sont considérés comme la clé privée pour le déchiffrement. L'inconvénient avec cette technique est que le DeepEDN doit former un réseau de chiffrement différent pour chaque image en clair. Cette méthode est difficile à utiliser car elle nécessite de stocker différentes clés pour différentes images (c'est-à-dire les paramètres des différents réseaux), ce qui requiert une grande capacité de stockage.

Dans cette même lancée d'utilisation d'AE pour chiffrer les données, Yildirim *et al.* (2018) dans leur travail, ont construit un autoencodeur à convolution profonde pour chiffrer les signaux d'électrocardiogramme pour les transférer de manière sécuritaire et en temps réel. Les données vont pouvoir être décodées à distance avec seulement une perte minimale.

Il y a aussi Gupta (2020) qui propose un encodeur (E) simple et peu profond pour chiffrer les données et réduire les coûts de calcul ainsi qu'un décodeur (D) plus complexe qui est un réseau neuronal profond génératif pour le déchiffrement des données. E est un perceptron à une seule couche cachée à action directe. Les encodages de E sont considérés comme des données chiffrées et il n'est pas possible de recréer des images à partir des encodages sans le décodeur. D est un perceptron multicouche et s'exécute sur une machine à haute puissance de calculs. Le décodeur D prend un vecteur codé en entrée et produit un vecteur d'image de dimension n qui est remodelé en une matrice RVB de 8 bits et transformé en une image. Le décodeur D dépend aucunement de l'encodeur E pour l'entraînement, à condition qu'un grand ensemble de données de codage de E soit disponible. Comme les codages proviennent de E qui est un simple réseau de neurones avec un grand nombre de paramètres aléatoires, il serait pratiquement impossible de casser ces codages sans D. Cependant, cette méthode occasionne des perte d'informations car le processus de déchiffrage est avec perte et l'image décodée est de qualité bien inférieure à celle de l'image d'origine.

À la différence de ces travaux, notre étude vise à concevoir un système capable de transmettre de manière efficace et sécuritaire des données de nos résidents des habitats intelligents sous forme chiffrée. C'est une approche qui s'inscrit dans la lignée des travaux de Gupta (2020) et de Ding *et al.* (2020) afin d'encoder les données collectées au sein d'habitats intelligents avec des capteurs complexes (radars à ultra large bande, caméras thermiques, caméras RGB-D, etc.) et de les décoder à distance. Notre cryptosystème prototype avec

l’autoencodeur vise donc à améliorer la sécurité grâce à ses caractéristiques telles que le calcul non linéaire et sa haute tolérance aux pannes.

CHAPITRE III
IMPLÉMENTATION DE L'AUTOENCODEUR POUR CHIFFRER LES DONNÉES

Le chapitre précédent nous a permis de voir qu'il y avait plusieurs approches dans la littérature pour chiffrer les données comme l'utilisation des algorithmes de cryptographie et l'utilisation des réseaux de neurones. Ce présent chapitre va donc consister à présenter une méthodologie et des architectures proposées pour chiffrer les données transitant d'un point A en local vers un serveur distant grâce aux autoencodeurs. Pour ce faire, de principaux travaux ont été réalisés. D'abord, nous avons implémenté des modèles d'autoencodeurs. Ensuite, nous avons effectué des tests d'encodage et de décodage de diverses sources de données (UWB, vidéos, images) provenant de capteurs disponibles au sein du LIARA avec nos autoencodeurs afin de faire une preuve de concept. Nous avons également répété les tests d'encodage et de décodage de données avec les algorithmes de cryptographie classiques AES-258 et Blowfish que nous avons implémentés (via des librairies existantes). Enfin nous avons fait une comparaison de la performance de sécurisation des données avec les deux méthodes de chiffrement de données. Cette comparaison s'est faite au regard de mesures d'évaluation comme la complexité de nos données dans nos habitats intelligents, la similarité cosinus et le temps de calculs lors de l'encodage de nos données. Évidemment, nos tests sont loin d'être exhaustifs, mais ils visaient surtout à établir et mesurer le potentiel de notre méthode comme alternative aux algorithmes de chiffrement.

3.1 DESCRIPTION DU LIARA

Le LIARA est un laboratoire de recherche situé au sein de l'Université du Québec à Chicoutimi. Son objectif est de développer des technologies qui vont permettre d'étendre l'environnement physique dans le but de s'adapter à l'humain et de créer des services et des

dispositifs intelligents qui vont être aptes à répondre aux besoins des individus, des groupes et des sociétés. Dans ce laboratoire est conçu un habitat intelligent pour une personne, de près de 100 m², (Bouchard *et al.*, 2014) regorgeant de nombreux capteurs et effecteurs et permettant de collecter facilement des données. Cet habitat intelligent possède une pièce de vie et une salle de bain. La pièce de vie comprend un espace cuisine, un petit salon ainsi qu'un espace chambre. L'habitat est tout équipé et possède tous les équipements et matériaux dont on a besoin pour vivre décemment. La figure 3.1 donne un aperçu de l'habitat intelligent du LIARA.

FIGURE 3.1 : L'habitat intelligent du LIARA. Réproduit avec la permission de Bouchard *et al.* (2014)

À l'intérieur de l'habitat se trouvent un analyseur de puissance intelligent, des capteurs infrarouges, des tapis de pression, des lumières LED et de nombreux capteurs de température. D'autres effecteurs incluent des haut-parleurs IP, un téléviseur HD à écran plat, un iPad et un cinéma maison. Tous ces capteurs et équipements présents dans le laboratoire permettent de reconnaître les activités et de collecter les données des résidents. Il s'agit donc du parfait environnement test pour vérifier nos hypothèses. Notre présent chapitre utilisera les données

issues d'ensembles de données du LIARA ainsi que d'autres capteurs pour nos différentes expérimentations et tests.

3.2 JEUX DE DONNÉES

Les données sont une des parties importantes dans le processus d'apprentissage, c'est la première étape pour commencer à construire notre modèle. Le LIARA possède plusieurs jeux de données, dont certains disponible publiquement sur le Web. En particulier, un de ces jeux de données a été utilisé pour notre projet. Ce jeu de données comprend des données de nombreux capteurs ambiants (e.g., infrarouges, contact électromagnétiques, etc.) et de radars UWB. Bien qu'au final, seules les données radars ont été utilisées, notre plan initial était d'installer le système de sécurisation par autoencodeur afin de traiter l'entièreté des données disponibles dans le jeu de données et, en fait, même l'entièreté des données pouvant être capturées au LIARA (l'ensemble de données ne comprend qu'un sous-ensemble de capteurs). Les données récoltées dans cet ensemble représentent des activités de la vie quotidienne (AVQ) qu'une personne peut réaliser dans un habitat intelligent. Il a été construit avec une dizaine de participants recrutés au préalable qui ont simulé ces AVQ. Il contient 14 activités distinctes :

- Se brosser les dents ;
- Prendre une douche ;
- Aller aux toilettes ;
- Dormir ;
- S'habiller / se déshabiller ;
- Faire le ménage ;

- Laver la vaisselle ;
- Ranger la vaisselle ;
- Se reposer ;
- Utiliser un un cellulaire(ou un ordinateur) ;
- Ranger le linge ;
- Manger ;
- Boire ;
- Lire.

Ces activités se veulent diverses par leur nature, par leur durée (variant de 15 secondes à 300 secondes) et par le lieu de leur réalisation dans l'habitat intelligent. En somme, au moment de la réalisation de notre projet, le jeu de données du LIARA servant de stockage pour l'ensemble des observations réalisées comprenait des données de radars UWB (fichier JSON), des données de capteurs (fichier JSON) et des données de centrales inertielles (IMU) pour un poids total de 181 Gigaoctets. Le jeu de données se présente avec une arborescence comme l'illustre la figure 3.2. Les données collectées par chaque participant se trouvent dans un dossier possédant un ID unique, puis un autre avec le nom de l'étiquette d'activité réalisée puis les fichiers de données à l'intérieur du fichier.

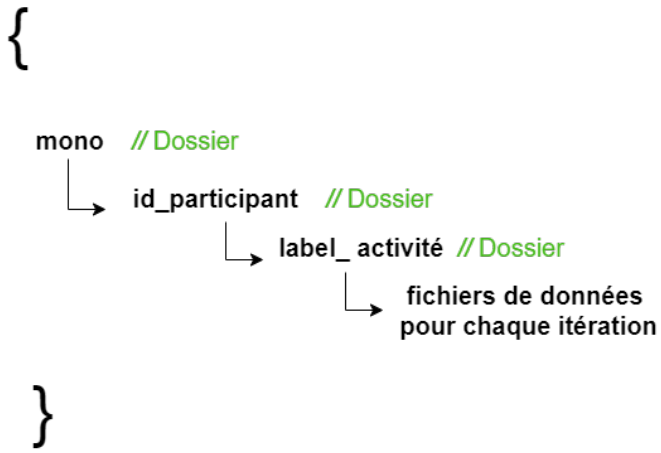


FIGURE 3.2 : L'arborescence des données collectées au LIARA.

©Bégué Hadja Rahinatou Forgo

Il faut noter que le processus de collecte des données au LIARA est rigoureux et respecte les règles éthiques propres aux expérimentations humaines. D'ailleurs, la partie des données issues d'expérimentation avec des êtres humains provient d'une étude approuvée par le comité d'éthique de la recherche de l'UQAC (#2021-487). L'autrice de ce mémoire est d'ailleurs cochercheure de l'étude. Aussi, pendant nos tests, nous avons utilisé le jeu de données de MNIST (Mixed National Institute of Standards and Technology). MNIST est l'une des bases de données les plus populaires dans l'apprentissage automatique. Elle est formée de 60 000 images d'entraînement et 10 000 images de test. Ces images sont carrées et en niveau gris de 28×28 pixels. Elles représentent des chiffres écrits à la main de 0 à 9.

3.3 TYPE DE DONNÉES

Comme mentionné précédemment, l'habitat intelligent comporte plusieurs capteurs qui produisent différents types de données. Dans cette section, nous reviendrons un peu plus en détails sur chaque type de données. Certains de ces différents capteurs utilisés pour la reconnaissance d'activités sont sans fil, ils fonctionnent grâce à Internet (WIFI) et avec des

batteries ou des piles. Les autres sont directement alimenté via le réseau électrique de l'habitat intelligent.

3.3.1 LES RADARS À BANDE ULTRA LARGE

Les radars à bande ultra large ou en anglais Ultra-Wideband (UWB) transmettent des ondes radio. C'est un système d'antennes à impulsions. Ils permettent de détecter la position d'un être humain ou d'un animal ou le mouvement d'objets. Leurs ondes passent à travers les murs et sont utilisés dans le domaine médicale. Les radars UWB présents au laboratoire sont au nombre de 3 localisés respectivement dans la salle à manger, la chambre à coucher et dans la salle de bain. Ils ont une bande de fréquence se situant entre 3.1 GHz et 10.6GHz et retournent des fichiers JSON. Les informations produites par nos radars sont sous forme de nombres complexes et caractérisent les réflexions captées par le radar et renseignant sur l'amplitude et la phase de ces dernières. Chaque radar retourne 184 bins représentant les données Distance-Doppler, la structure combinée des radars a une forme de 750 lectures par 184 bins par 3 radar UWB. La combinaison de l'ensemble des signaux reçus sur l'entièreté des bins constitue une *image* radar, c'est-à-dire la représentation instantanée de son environnement qui s'apparente à une photographie. Il est possible de facilement visualiser la structure combinée en la représentant comme une image RGB. La figure 3.11 nous montre une visualisation de ces données UWB.

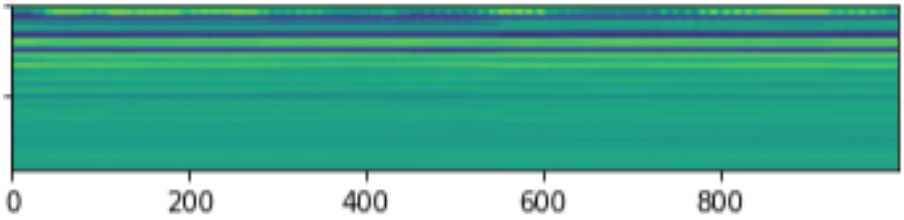


FIGURE 3.3 : Visualisation des données d'un radar sous forme d'image

©Bégué Hadja Rahinatou Forgo

Toutes ces données des résidents seraient, en principe, transmises vers une application de télésurveillance sur un ordinateur distant dans le cadre d'un déploiement réel.

3.3.2 IMAGES ET VIDÉOS

Un autre des types de données capturables au LIARA sont les images et les vidéos produites par les capteurs visuels : les caméras-RGBD. Ces capteurs sont au nombre de deux. Le LIARA ne dispose pas d'ensemble de données utilisable pour des fins de recherche, mais comme l'équipe vise a éventuellement expérimenter avec ces caméras, il semblait intéressant de les inclure dans ce mémoire de recherche. Une des caméras est localisée dans la salle à coucher (RGBD-A) et la seconde dans la salle à manger (RGBD-B). C'est le type de capteur qui offre la plus grande expressivité. Cependant, ils sont très invasifs et le traitement de leurs données est complexe. Par exemple, la reconnaissance de formes simples dans une large gamme de conditions d'éclairage, d'orientations et de couleurs nécessite des algorithmes d'IA assez élaborés. Ces capteurs donnent en sortie des vidéos de 1280 pixels de largeur par 720 pixels de hauteur avec une fréquence de 60 FPS.

En décortiquant un fichier vidéo capturé par une telle caméra, on se retrouve avec 3 composantes. D'abord, les RGB et la profondeur qui fournissent en sortie des fichiers de type

MP4. Ensuite, nous trouvons la combinaison, soit les RGBD, avec pour extension SVO et AVI.

En ce qui concerne les images, elles sont formées de matrices d'amplitude pour chaque canal de couleur RGB (c'est-à-dire rouge, vert, bleu) et sont enregistrées dans différents formats connus (JPEG, GIF, PNG, BMP, WEBP, etc.). Les fichiers d'images sont composés de données numériques qui sont stockées dans l'un de ces formats afin d'être pixélisées et utilisées sur un ordinateur ou une imprimante. On entend par pixellisation, la procédure de conversion des informations d'image en une grille de pixels. Chaque pixel a un nombre de bits qui spécifient sa couleur (et dans certains cas, son opacité). Par exemple, les pixels des images en niveaux de gris sont également représentés dans un format matriciel. Chacune des images pixellisées a un nombre entier correspondant dans l'ensemble 0, . . . , 255. L'intensité maximale est de 255 et la minimale est de 0. Entre 0 et 255, la couleur est intermédiaire entre l'intensité minimale et maximale. La figure 3.4 explique comment une image comme celle-ci peut être encodée sous forme de matrice.

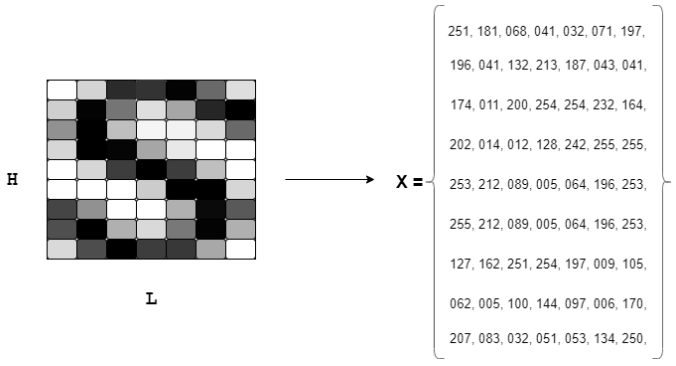


FIGURE 3.4 : Répresentation d'une image en nuance de gris sous la forme d'une matrice d'octets
©Bégué Hadja Rahinatou Forgo

La vidéo peut être définie comme un média numérique qui enregistre, reproduit, lit, diffuse et affiche des images fixes. D'un point de vue informatique, la représentation typique

d'une vidéo est une série d'images espacées d'une durée fixe (fonction du nombre d'images par seconde de la vidéo). Les vidéos ont différents formats standardisés comme AVI, MP4, MKV, WMV et plus encore. Les difficultés associées à l'utilisation des vidéos sont similaires à celles associées aux images. Cependant, il existe au moins un obstacle supplémentaire : les fichiers vidéo peuvent être très volumineux. La taille de ces fichiers implique un temps de traitement plus long et la nécessité d'une grande quantité de stockage si on souhaite utiliser un grand nombre de vidéos. Si de tels vidéos étaient utilisées, nous pourrions voir clairement le résident et tous les détails sur la ou les personnes vivant dans l'habitat intelligent. Nous pourrions aussi analyser ces vidéos en temps réel. Ces médias vidéos pourraient être stockés dans un serveur distant pour porter assistance en cas de besoin [2].

3.3.3 AUTRES TYPES

En plus de ces données, il y a des données de capteurs ambiants et des données provenant d'un bracelet équipé d'une centrale inertielle communicant par Bluetooth. L'ensemble des capteurs ambiants historiquement présents dans le laboratoire :

- Capteurs de pression ;
- Capteurs de température ;
- Capteurs de luminosité ;
- Capteurs d'écoulement d'eau ;
- Capteurs de température d'eau ;
- Capteurs de mouvements ;

2. https://blogue.genium360.ca/article/innovation/les-composants-40-comment-choisir-les-capteurs-de-donnees/

o Contacteurs électromagnétiques.

Ils mettent l'accent sur l'état de l'environnement qui entoure les résidents plutôt que sur eux-mêmes. En termes plus clairs, lorsqu'un événement se produit chaque capteur envoie les informations de l'événement au serveur qui se charge de l'enregistrer dans un fichier JSON. De plus, dans les données du LIARA, on trouve des bracelets Bluetooth qui embarquent une centrale inertielle (IMU) de 6 axes (3 axes accélérométriques et 3 axes gyroscopiques). Ils sont aussi au nombre de 2, retournent des fichiers JSON et captent les données à une fréquence de 50 Hz. Tous les capteurs sont statiques, sauf les bracelets Bluetooth. La figure 3.5 montre quelques uns de ces capteurs et leur disposition au LIARA.

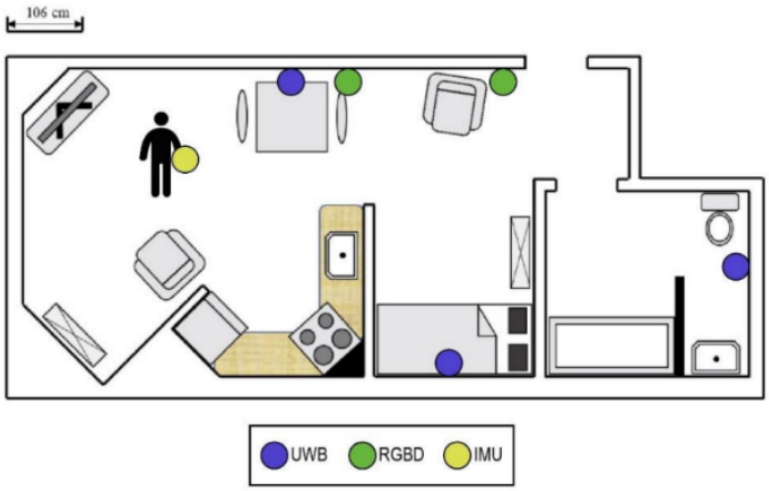


FIGURE 3.5 : Plan du laboratoire indiquant l'emplacement des principaux capteurs. Les petits cercles bleus matérialisent les radars ultra-wideband, les verts montrent les caméras de profondeur et le jaune indique le bracelet Bluetooth. Reproduit avec la permission de Florentin Thullier

3.4 ENCODAGE DES DONNÉES

Après la première étape qui est l'acquisition des données, nous allons importer toutes les librairies nécessaires pour construire notre solution.

TensorFlow est l'un des principaux cadriciel d'apprentissage profond. C'est une bibliothèque logicielle en accès libre, ouverte pour le calcul numérique haute performance et développée par l'équipe Google en 2011 (Abadi *et al.*, 2016). Elle implémente des méthodes d'apprentissage automatique basées sur le principe des réseaux de neurones profonds. C'est d'ailleurs la principale bibliothèque que nous utiliserons pour construire nos réseaux neuronaux.

Tensorflow est disponible sur plusieurs systèmes d'exploitation comme Linux, Windows, Android, macOS et iOS. Elle peut fonctionner sur processeur ou GPU, ce qui rend le logiciel hautement évolutif et adapté à la plupart des utilisateurs sur ordinateur, mobile, Web et cloud. Tensorflow est également capable de paralléliser les calculs, de diviser toute la série d'opérations en plus petits morceaux et de les exécuter simultanément sur plusieurs CPU et GPU. Ce type de fonctionnement est très utile pour les applications d'apprentissage automatique à grande échelle. Il y a aussi la bibliotheque Keras qui fait partie de Tensorflow. Quand on récupère Tensorflow, on récupère également Keras.

Keras est l'API de haut niveau de Tensorflow. En effet, Keras est une bibliothèque de réseaux de neurones en accès libre qui offre une API de plus haut niveau nous permettant d'utiliser les fonctions de plus bas niveau de Tensorflow. Autrement dit, nous utiliserons Keras au-dessus de Tensorflow (le backend) pour disposer d'un ensemble plus intuitif d'appels API pour développer nos modèles d'apprentissage profond.

3.5 ARCHITECTURE DES MODÈLES D'AUTOENCODEURS CHOISIS

Avant de créer un modèle d'encodage de données, il est nécessaire de comparer les architectures des différentes bibliothèques Keras. Ce processus permet de déterminer quelle architecture est la mieux adaptée pour traiter l'ensemble de données. Les modèles présentés dans cette section ont été testés avec les données du LIARA ainsi qu'avec une vidéo collectée par l'autrice de ce mémoire.

Mais d'abord, il semble important de discuter des principaux éléments essentiels. Comme l'indique le tableau 3.1, en général, la création des modèles d'AE nécessite l'utilisation des paramètres suivants : inputs, dense et model.

Le paramètre *keras.input* permet de definir le type de données en entrée par exemple le nombre de lignes, de colonnes que peut recevoir l'autoencodeur. Un des sous paramètres de cette fonction est *shape*. *Shape* permet de definir la taille de la matrice. La matrice peut-être unidimentionnelle, bidimentionnelle ou tridimentionnelle. Par exemple, dans le cas où la matrice est bidimentionnelle, le paramètre definit le nombre de lignes et de colonnes de la matrice. Dans le cas où la matrice est unidimentionnelle, elle définit le nombre de colonnes que doit recevoir l'AE. Ensuite, il y a *batch_size* qui correspond à la taille du lot de données. *Batch_size* est le nombre d'insances d'entraînement utilisées dans une itération. C'est à dire qu'il s'agit de diviser l'ensemble d'entraînement pour chaque *epoch* en plus petits morceaux afin d'accélérer le calcul. Puis le dernier que nous verrons est *name*. *Name* est le nom donné à la couche, il est facultatif. Par défaut un nom est généré automatiquement, ce nom se doit d'être unique dans le modèle afin d'y référer facilement.

Un deuxième paramètre des AE est *keras.Model*. Il prend comme sous-paramètres *inputs*, *outputs* et *name*. *Inputs* étant l'entrée du modèle, *outputs* la sortie provenant de Keras et Name correspondant au nom du modèle. Pour créer les modeles d'AE, il faut enchainer les

TABLEAU 3.1 : Tableau explicatif des différents paramètres pour la construction des modèles d'AE

Paramètres	Sous-paramètres	Par défaut
Input	Shape	None
	Batch_size	None
	Name	None
Dense	Units	-
	Activation	None
	Use bias	True
Model	Inputs	-
	Outputs	-
	Name	-

appels de couches pour indiquer la propagation avant du modèle et par la suite créer le modèle à partir des données d'entrées et des sorties.

Puis on a *layers.Dense*, c'est la couche de base des réseaux de neurones. On parle aussi de couche densément connectée. Les sous-paramètres de dense sont entre autre *units*, *activation*, *kernel_regularizer*. *Units* est sous la forme d'un entier positif et est la dimensionnalité de l'espace de sortie. Activation est la fonction d'activation à utiliser. Au cas où l'on ne choisit pas la fonction d'activation, une fonction d'activation linéaire identité sera appliquée par défaut. *Kernel* est une matrice de poids créée par la couche. *Kernel_initializer* sert donc à initiaiser la *kernel* matrice des poids.

Il existe évidemment beaucoup d'autres options, mais celles-ci sont les principales qui ont été prises en compte dans notre méthodologie de recherche.

3.5.1 AE SIMPLE

Pour ce modèle d'AE, il est formé d'une seule couche neuronale entièrement connectée comme encodeur et comme décodeur. En effet, le réseau possède une couche entrée qui permet d'alimenter le modèle avec les données d'entrée. Puis, une fonction d'activation *Rectified Linear Unit* (ReLU) est appliquée à la couche d'entrée, ce qui permet de générer une couche cachée (la taille de la couche cachée est de 32) qui sera la représentation nouvellement apprise. La fonction ReLU est la fonction d'activation la plus simple et la plus utilisée (Ramachandran *et al.*, 2017). Cette fonction d'activation est utilisée dans la partie encodeur de l'autoencodeur.

Par la suite, une autre fonction d'activation ReLU sera appliquée à la couche cachée pour reconstruire les observations originales. Cette fonction d'activation produit l'entrée de la partie décodeur de l'autoencodeur. La couche de sortie représente donc les observations

nouvellement reconstruites avec les données originales. La figure 3.6 montre un résumé de notre modèle.

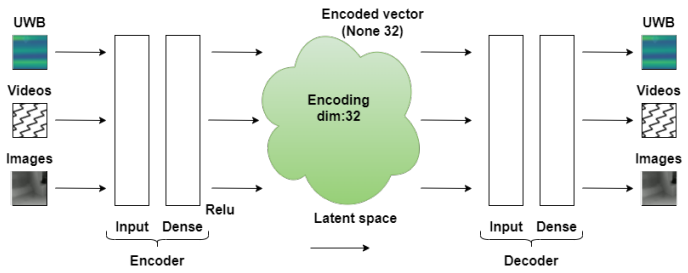


FIGURE 3.6 : Architecture de l'autoencodeur simple dévéloppé

©Bégué Hadja Rahinatou Forgo

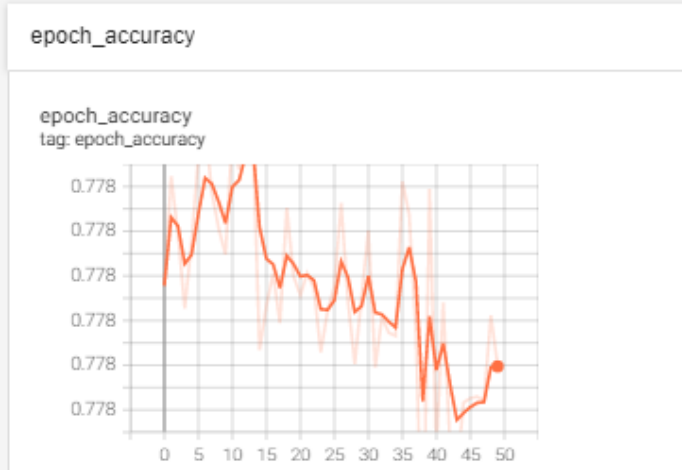


FIGURE 3.7 : Taux d'apprentissage et de précision de l'AE simple

©Bégué Hadja Rahinatou Forgo

Notons que notre réseau utilise la fonction *flatten* qui permet de prendre la sortie multidimensionnelle d'une couche et de la transformer en sortie unidimensionnelle (i.e. en vecteur). Elle n'affecte pas la taille du lot et n'occasionne aucune perte. Il faut préciser que dans notre AE simple, l'encodeur et le décodeur sont formé indépendamment de l'autre.

En résumé, en ce qui concerne notre autoencodeur, on a comme paramètres : la taille de l'espace latent qui a été fixé à 32, la taille, la hauteur de notre instance, ensuite nous avons la profondeur de l'instance.

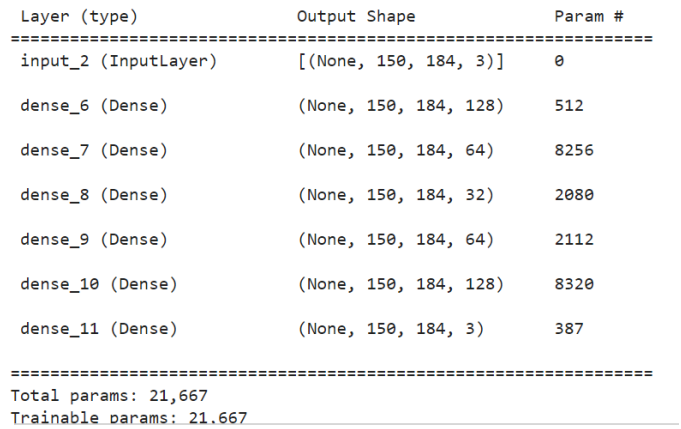
En entrée pour notre AE, il faut adapter à la taille de chaque type de données utilisées. Pour les vidéos, la taille des données d'entrée est de (1280,720,3). Ces chiffres représentent respectivement la largeur, la hauteur et la profondeur. La profondeur est le nombre de canaux, ou matrices dans notre cas spécifique.

Pour les fichiers JSON des UWB, la taille sera de (150,184,3). Le nombre 150 représente les lignes et 184 pour le nombre de colonnes. Le chiffre 3 provient du nombre de radars comme précédemment annoncé. Dans notre implémentation python, nous avons utilisé un *dataframme* où nos données UWB étaient organisées dans un tableau à deux dimensions en lignes et colonnes.

Enfin, pour les images individuelles, nos AE ont la capacité de les encoder et de décoder que ce soit en noir et blanc ou en couleurs. Pour les images en couleurs, l'*input shape* est le même que pour les vidéos, soit de (1280,720,3). Pour les images en noir et blanc, il passe à (1280,720,1). Nous allons ensuite aplatir les matrices des images, afin de réduire leur dimension. Nous allons passer d'une matrice 2D à un vecteur 1D. Cette opération nous permettra d'alimenter les couches du réseau avec ces images.

3.5.2 AE PROFOND

Nous n'avons pas à nous limiter à une seule couche comme encodeur ou décodeur, nous pourrions plutôt utiliser une pile de couches, telles que nous montre la figure 3.8.



```
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 150, 184, 3)]     0

dense_6 (Dense)              (None, 150, 184, 128)     512

dense_7 (Dense)              (None, 150, 184, 64)      8256

dense_8 (Dense)              (None, 150, 184, 32)      2080

dense_9 (Dense)              (None, 150, 184, 64)      2112

dense_10 (Dense)             (None, 150, 184, 128)     8320

dense_11 (Dense)             (None, 150, 184, 3)       387

=================================================================
Total params: 21,667
Trainable params: 21,667
```

FIGURE 3.8 : Les couches d'AE profond

©Bégué Hadja Rahinatou Forgo

Notre AE profond a plus de couches que l'autoencodeur simple et est donc en mesure d'apprendre des fonctionnalités plus complexes. Les données sont alimentées à travers une pile de couches entièrement connectées où chaque neurone est connecté à tous les neurones de la couche suivante. La figure 3.9 nous montre un résumé de l'AE profond avec les différents paramètres utilisés pour la création du modèle.

	Couches	**Paramètres**	**Valeurs**
Encodeur	Dense1	Units	128
		Activation	Relu
	Dense2	Units	64
		Activation	Relu
	Dense3	Units	32
		Activation	Relu
Décodeur	Dense4	Units	64
		Activation	Relu
	Dense5	Units	128
		Activation	Relu

FIGURE 3.9 : Résumé de notre modèle d'AE profond

©Bégué Hadja Rahinatou Forgo

L'entrée de notre autoencodeur est une matrice à une dimension. Pour la vidéo on utilisera une entrée de taille (720,1280,1) pour les vidéos en noir et blanc et (720,1280,3) pour celle en couleur. En ce qui concerne les données UWB, on a utilisé (150,184,3) comme *input shape*.

3.5.3 AE CONVOLUTIONNEL

Nous pouvons aussi ajouter des couches de convolution à un autoencodeur pour faire de lui un CAE (Convolutionnal AutoEncoder). Notre CAE est formé d'un encodeur constitué d'un empilement de couches Conv2D (couche de convolution 2D) et MaxPooling2D (le *pooling* maximum étant utilisé pour le sous-échantillonnage spatial) et d'un décodeur constitué d'un empilement de couches Conv2D et UpSampling2D. En effet, notre CAE en entrée va appliquer des convolutions successives pour extraire des caractéristiques de nos données (JSON, image, vidéo) en entrée et les réduire successivement pour obtenir un vecteur de petite taille qui sera une projection de la donnée en entrée dans notre espace latent. La fontion d'activation ReLU est appliquée juste après la phase de convolution afin de rendre le modèle non linéaire. Les valeurs positives sont maintenues et les valeurs négatives sont ramenées à 0. La taille des cartes de caractéristiques reste toutefois inchangée. Nous avons limité quantité de noyaux des filtres afin que l'encodage permettent de mettre à plat pour constituer un vecteur de faible taille (dim-128).

De l'autre côté de l'espace latent, nous avons décodé ce vecteur, cette projection de l'espace latent, de manière à reconstituer la donnée en entrée. Notre décodeur s'appuie sur des couches de convolution Conv2D. L'objectif de cette opération est d'agrandir notre donnée, la reconstituer à partir des caractéristiques récupérées automatiquement de notre espace latent. La figure 3.10 nous donne l'appercu de notre CAE.

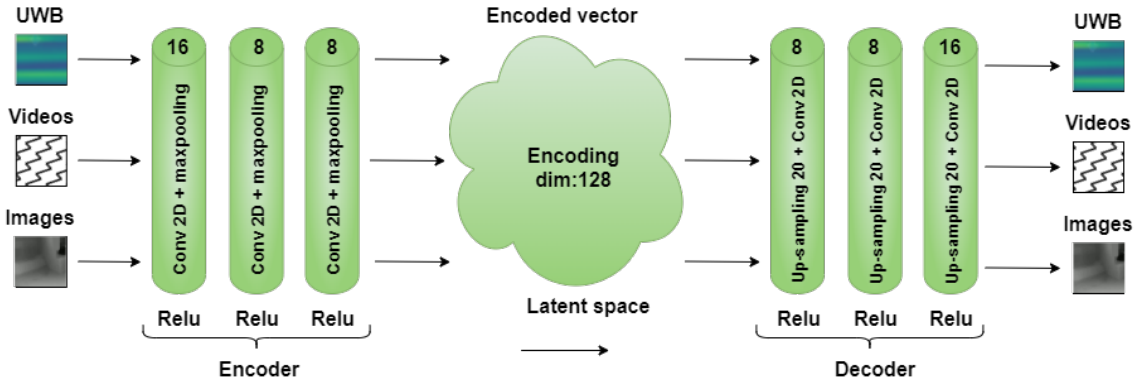


FIGURE 3.10 : Architecture de l'autoencodeur convolutionnel développé

©Bégué Hadja Rahinatou Forgo

3.6 ENTRAINEMENT DES MODÈLES

Les données sont une des parties importantes dans le processus d'apprentissage. Pour l'apprentissage et l'entraînement des modèles d'autoencodeurs, il faut diviser nos différents types de données en 3 groupes. Ces groupes sont :

— Les données d'entraînement qui serviront à entraîner directement le réseau de neurones,

— Les données de test qui seront utilisées pour vérifier la performance de notre réseau.

— Parfois on utilise un dernier groupe de données, les données de validation, utilisées à la toute fin du processus pour optimiser les hyperparamètres.

Techniquement, nous commencerons par créer un ensemble d'apprentissage (X-Train-AE) avec deux tier des données et un ensemble de test (X-Test-AE) avec les un tiers de données restantes. Afin d'y parvenir, nous avons cette fois eu recours à la bibliothèque Scikit-Learn qui offre des fonctions très conviviales pour diviser nos données (Pedregosa *et al.*, 2011).

Tout d'abord, pour les données issues des caméras RGBD, nous avons dû récolter des vidéos de démonstration puisque l'ensemble de données du LIARA ne dispose pas de

vidéos utilisables à des fins de recherche. L'objectif était ici de pouvoir entraîner des modèles types d'AE afin de démontrer la faisabilité de l'approche sur les images et les vidéos dans l'éventualité prochaine où celles-ci seront utilisées. À cette fin, nous avons utilisé 10 courtes vidéos afin de vérifier que les AE pourraient être entraînés. Ces données seront passées aux AE pour permettre à nos differents AE d'apprendre à reconnaitre les vidéos puis, les chiffrer et dechiffrer ulterieurement. Nos différents tests consisterons donc à chiffrer et déchiffrer les données vidéos exemples avec les trois modèles d'autoencodeurs construits. Pour ce faire, il suffira d'abord de faire des traitements sur les données vidéos. La méthode utilisée consiste à extraire toutes les images de la vidéo puis à les envoyer à notre encodeur. Toutes les images de la vidéo ont une taille de (720,1280). L'encodeur à son tour va chiffrer toutes les images extraites et les mettre dans l'espace latent. Ces images chiffrées pourront être envoyées sur le réseau Internet au serveur distant. De l'autre côté, au niveau du centre de surveillance, par exemple, le décodeur quant-à lui va récuperer les images chiffrées puis les déchiffrées. Pour les images, nous avons procédé de la même façon en extrayant directement celles-ci d'un fichier vidéo exemple. Nous avons au total 500 images.

Enfin, pour les données UWB, nous avons pris un total de 60 fichiers JSON directement dans l'ensemble de données du LIARA pour l'entrainement de nos modèles. Ces fichiers représentent des activités réalisées au laboratoire par différents participants. Cependant, les activités n'ont pas réellement d'impact sur nos tests et nous avons donc choisi d'utiliser ces fichiers principalement parce qu'ils étaient disponibles immédiatement en grande quantité.

Une autre partie importante est la préparation de nos données des capteurs du LIARA. En ce qui concerne les données UWB, cette préparation se fera en étapes : la lecture, la fusion, la division et l'affichage des fichiers JSON. En premier lieu, on récupère les fichiers contenant le mot UWB dans leur nom. Pour chaque fichier UWB récupéré, nous le réadaptons au format JSON car le fichier original ne respecte pas la nomenclature des JSON. Pour ce faire, une

virgule a d'abord été mise à la fin de chaque ligne puis un crochet au début du fichier (première ligne) et enfin un autre à la fin de la dernière ligne du fichier JSON. En deuxième lieu, après avoir adapté les fichiers au bon format, il suffira de fusionner tous les fichiers JSON ensemble. Dans notre cas, on prendra 60 fichiers qui correspondent aux données collectées de trois résidents différents. Ensuite, les données de ces résidents sont divisées en trois et chaque partie correspond à un radar. Il y a trois radars comme mentionné précedemment. En dernier lieu, nous affichons chaque radar en utilisant la librairie *matplotlib* comme nous le montre la figure 3.11.

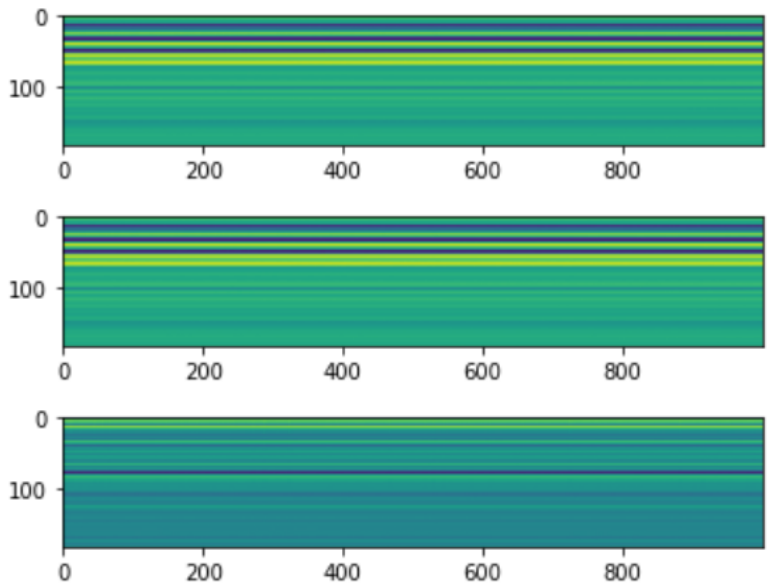


FIGURE 3.11 : Visualisation des données des trois radars sous forme d'image
©Bégué Hadja Rahinatou Forgo

Le schéma 3.12 résume toutes ces étapes.

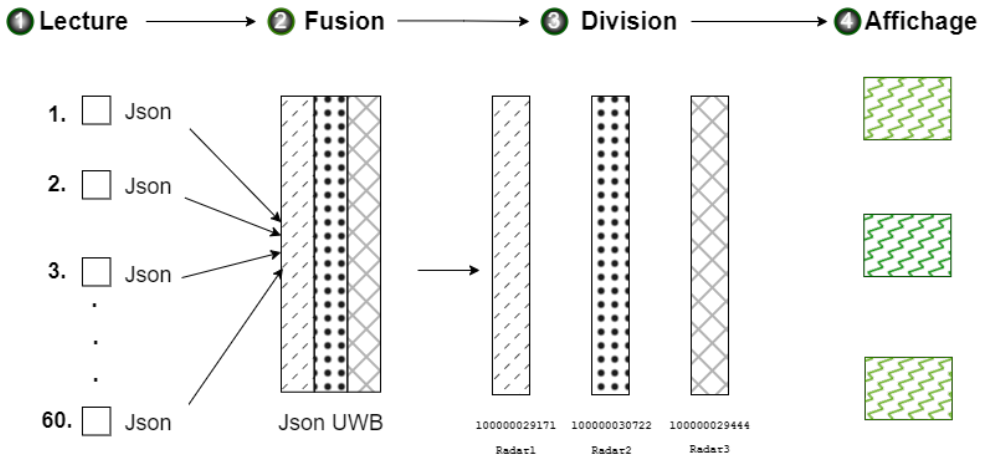


FIGURE 3.12 : Les étapes de la péparation des données UWB

©Bégué Hadja Rahinatou Forgo

Ensuite, nous avons compilé les couches que nous avons conçues pour le réseau de neurones. Pour ce faire, nous avons choisi une fonction de perte MSE (erreur quadratique moyenne) pour guider l'apprentissage des poids, un optimiseur pour définir le processus d'apprentissage des poids et une liste de métriques pour évaluer la qualité du réseau de neurones.

Nous évaluons les modèles d'AE sur la base de l'erreur de reconstruction entre la matrice de caractéristiques nouvellement reconstruite sur la base de l'autoencodeur et la matrice de caractéristiques originale que nous introduisons dans l'autoencodeur.

Les réseaux de neurones sont formés par de nombreux cycles denommés époques. À chaque époque, le réseau de neurones réajuste les poids appris afin de minimiser la perte de l'époque d'avant. La procédure d'apprentissage de ces poids est spécifiée par l'optimiseur. Pour apprendre les poids optimaux, le réseau de neurones doit avoir la capacité de modifier intelligemment son estimation des poids optimaux. Pour ce faire, il faut déplacer les poids dans une direction qui diminue progressivement la fonction de perte ou mieux encore, déplacer les poids dans cette direction de manière stochastique. La descente de gradient stochastique

(ou SGD) est beaucoup utilisée dans l'entraînement des réseaux de neurones mais nous avons choisi d'utiliser l'optimiseur Adam. L'algorithme d'optimisation Adam est une extension de SGD. Dans cet algorithme d'optimisation, les moyennes courantes des gradients et des seconds moments des gradients sont utilisées. Il est bon car il ajuste de manière dynamique le taux d'apprentissage pendant le processus de formation des modèles.

Puis, nous devons sélectionner le nombre d'*epochs* et la taille du lot, puis commencer le processus d'apprentissage en appelant la méthode *fit*. Le nombre d'époques détermine le nombre de fois où l'entraînement a lieu sur l'ensemble des données que nous transmettons au réseau neutre. Nous le fixons à 50 epochs pour commencer puis, par la suite, nous allons varier le nombre d'epochs et voir comment les modèles réagissent face à cette expérimentation. Le lot définit le nombre d'échantillons sur lesquels le réseau neuronal s'entraîne avant d'effectuer la prochaine mise à jour du gradient. Si la taille de lot est égale au nombre total d'observations, le réseau neuronal effectue une mise à jour de gradient à chaque époque. Sinon, il effectuera des mises à jour plusieurs fois par époque. Notre taille du lot a été fixé à 32 lors du début de nos tests. À noter que la littérature ne propose pas de méthodologie claire pour choisir cette valeur.

Enfin, notre modèle est évalué selon la distance cosinus entre les données d'entrées originales puis les données de sorties (déchiffrées), sur le temps d'exécution pour chiffrer et déchiffrer les données et sur l'erreur quadratique moyenne.

3.7 MESURES DE PERFORMANCES

Pour les fins de notre étude, les résultats des tests indiqueront essentiellement trois indices d'évaluation de performance très connus : la similarité cosinus, la perte de reconstruction et le temps de calcul pour chiffrer nos données.

3.7.1 SIMILARITÉ COSINUS

Une analogie similaire peut être établie pour les images structurées telles que les visages, où la similarité basée sur le cosinus peut être utilisée pour modéliser la distribution sous-jacente des valeurs des pixels plutôt que de calculer la distance entre les intensités réelles des pixels.

En particulier, la similarité en cosinus est l'une des méthodes les plus populaires pour analyser les caractéristiques et la tendance des données vectorielles sur la base de la similarité directionnelle. Mathématiquement, elle est calculée en normalisant le produit scalaire de deux vecteurs, le calcul est donc simple et intuitif.

Cette mésure a été employée dans cette étude pour chiffrer et dechiffrer les données. Par exemple, si par la valeur de similarité du cosinus est de 1, l'angle entre deux vecteurs est de 0, ce qui signifie qu'ils ont la même direction.

La similarité cosinus que nous allons utiliser va calculer la distance spatiale de chaque image d'origine avec l'image réconstruite. Et plus la similarité va tendre vers 1, plus les images vont etre identiques.

3.7.2 LA PERTE DE RECONSTRUCTION

En général, il s'agit d'une mesure de distance $dAE(x_i; f(x_i))$ entre l'entrée x_i et sa reconstruction correspondante $f(x_i)$, l'erreur quadratique moyenne des deux variables est souvent utilisée et se traduit par l'équation (3.1) :

$$L = d_{AE}\left(x_i; f\left(x_i\right)\right) = \sum_i \left\|x_i - f\left(x_i\right)\right\|^2 \tag{3.1}$$

L'erreur quadratique moyenne entre l'image d'origine et l'image déchifrée est souvent utilisée pour évaluer la sensibilité à la clé d'un algorithme de chiffrement d'image.

3.7.3 TEMPS DE CALCULS/ CONSOMMATION ÉNERGÉTIQUE

Une autre mesure que nous utiliserons pendant nos tests est le temps d'exécution. Ainsi, pour évaluer le temps d'exécution d'un algorithme, on compte le nombre d'opérations effectuées pour arriver au résultat. Une autre possibilité pour mesurer le temps d'exécution consiste à le faire directement depuis le programme, en Python. Cette façon de faire est même nécessaire si l'on veut mesurer le temps d'exécution d'une partie de programme seulement. La façon la plus simple consiste à utiliser la fonction *time* du module time, un algorithme basé sur des méthodes statistiques a réussi à prédire les temps d'exécutions et la consommation énergétique de programmes de manière plus précise.

3.8 EXPÉRIMENTATION AVEC LES AUTOENCODEURS

Les sections précédentes présentent notre solution de sécurisation des données d'habitats intelligents avec les autoencodeurs. Nous avons présenté les différents modèles d'AE puis étudier leur entrainement avec les différents paramètres d'apprentissage machine. Par ailleurs, nous avons aussi décrit les mesures de performances utilisées pour juger de la qualité de nos différentes implémentations. Les sections qui suivront feront part du chiffrement et du déchiffrement de nos divers types de données avec les AE et les algorithmes de cryptographie traditionnelles AES-256 et Blowfish.

3.8.1 ENCODAGE ET DÉCODAGE DES VIDÉOS

Pour l'AE simple après avoir chiffré et déchiffré les vidéos, on observe une similarité cosinus moyenne de 0.989. Quand au temps de calcul pour encoder les vidéos nos avons en moyenne 977.67 millisecondes (ms) et pour le déchifrement le temps de calcul est de 714.47ms.

55173.44

En ce qui concerne l'autoencodeur profond, on note une similarité cosinus de 0.999. Là, nous voyons que les résultats sont lègerement meilleurs que pour l'AE simple. L'AE profond a fourni des résultats supérieurs car plus l'AE est profond et a des couches plus il apprend et performe mieux avec les données d'images et vidéos. Le chifrement des vidéos s'est fait en 2 652.30ms et le déchifrement 2 772.43ms.

Et enfin pour l'AE convolutionnel, on remarque une faible ressemblance entre les images. Cela est dû aux pertes. La similarité cosinus baisse donc à 0.940. Sans surprise, le temps de chiffrement est beaucoup plus long et atteint 26 375,21ms. De façon similaire, le temps de déchifrement grimpe à 28 798.23ms. Le tableau 3.2 suivant donne un aperçu des différents AE avec leur similarité cosinus ainsi que le temps de calcul pour le chiffrement et le déchiffrement de vidéos.

TABLEAU 3.2 : Étude comparative des différents AE avec leur mesures de performances.

	AE simple	**AE profond**	**AE convolutionnel**
Similarité cosinus	0.998	0.999	0.940
Temps de chiffrement (ms)	977.67	2 652.30	26 375,21
Temps de déchiffrement (ms)	714.47	2 772.43	28 798.23

3.8.2 ENCODAGE ET DÉCODAGE DES IMAGES

Nous allons aplatir les matrices des images afin de réduire leur dimension. En effet, nous allons passer d'une matrice 2D à un vecteur 1D. Cette opération nous permettra d'alimenter les couches du réseau avec ces images.

Dans la figure 3.13, on peut voir par exemple un extrait de 2 images originales sur la première ligne et celles déchiffrées ou reconstruites par notre AE simple sur la deuxième ligne. L'image montre une grande ressemblance détectable à l'oeil nu entre les données originalees et celles déchiffrées.

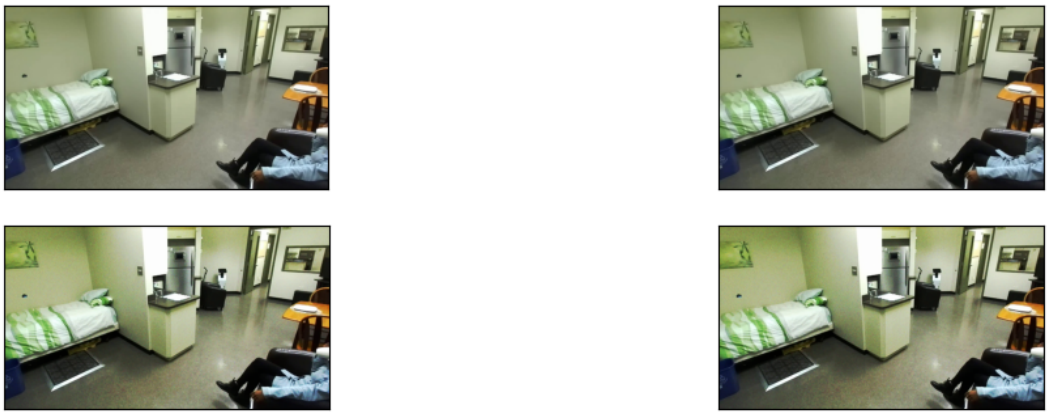


FIGURE 3.13 : Visualisation des images originales, chiffrées et déchiffrées

©Bégué Hadja Rahinatou Forgo

Les tests de chiffrage et de déchiffrage des données d'images ont données les résultats représentés dans le tableau 3.3.

TABLEAU 3.3 : Étude comparative des différents AE avec leur mesures de performances.

	AE simple	**AE profond**	**AE convolutionnel**
Similarité cosinus	0.999	0.999	0.988
Temps de chiffrement (ms)	100.54	144.69	184.69
Temps de déchiffrement (ms)	106.25	156	188.64

On remarque que la similarité cosinus des images avec l'AE simple a donné 0.999. Pour le temps de chiffrement il est de 100.54 ms et enfin on note 106.25 ms comme temps de déchiffrement.

Pour l'AE profond, la similarité cosinus est de 0.999 Le temps de chiffrement des données d'images est de 144.69 ms. Le temps de déchiffrement est de 156 ms.

En ce qui concerne l'AE convolutionnelle, la similarité cosinus est de 0.988. Le temps mis pour encoder les données a éte de 184.69 ms. Et, enfin, le temps de décodage est de 188.64 ms.

3.8.3 ENCODAGE ET DÉCODAGE DES UWB

Les différents tests d'encodage et de décodage des données UWB ont donné les résultats contenus dans le tableau 3.4. Les résultats semblent concorder avec nos observations pour les images et les vidéos.

TABLEAU 3.4 : Étude comparative des différents AE avec leur mesures de performances pour l'encodage et le décodage des UWB.

	AE simple	**AE profond**	**AE convolutionnel**
Similarité cosinus	0.999	0.987	0.963
Temps de chiffrement (ms)	164.36	272	460.50
Temps de déchiffrement (ms)	102.15	169	486.36

3.9 EXPÉRIMENTATION AVEC LES ALGORITHMES DE CRYPTOGRAPHIE

Plusieurs langages de programmation sont fréquemment utilisés pour implémenter les fonctions de cryptographique. Parmi ces langages, Python est un langage largement utilisé.

De nombreuses bibliothèques Python sont dédiées à l'implémentation d'algorithmes de cryptographie tels que M2Crypto, PyCrypto, pyOpenSSL, python-nss et les liaisons Python de Botan. La bibliothèque PyCrypto est la plus utilisée à notre connaissance (Vidhya, 2018).

En effet, la boîte à outils de cryptographie Python (PyCrypto) vise à fournir une solution fiable et une base stable pour écrire des fonctions cryptographiques en langage de programmation Python. Elle prend en charge les chiffrements par blocs, les flux et les hachages ainsi qu'une fonction de calcul pour eux. C'est dailleurs celle que nous utiliserons pour implémenter nos algorithmes de cryptographie AES-256 et Blowfish dans le but de chiffrer et dechiffrer nos différents types de données.

3.9.1 IMPLÉMENTATION DE L'ALGORITHME DE CRYPTOGRAPHIE AES-256

AES a une taille de bloc de 128 bits et cette implémentation d'AES prend en charge une taille de clés de 32 octets de long pour AES-256.

Le chiffrement AES-256 nécessite une clé forte. Plus la clé est forte, plus le chiffrement est fort. Le vecteur d'initialisation doit avoir une longueur de 16 octets. La première étape est le choix de la clé. Il faudra entrer la clé secrète qui peut être composée de n'importe quels caractères alphanumériques. Elle sera utilisée pour déchiffrer le fichier plus tard.

Après avoir eu la clé, il faut passer à la définition de la fonction de chiffrement ENCRYPT. Cette fonction chiffre le fichier fourni par le paramètre. La sortie de la fonction ENCRYPT

est un fichier avec le mot _encrypted qui a été ajouté au nom du fichier initial mais dont les informations ne sont pas lisibles.

Enfin pour le déchiffrement, il faut définir la fonction DECRYPT. Pour ce faire, il faut inverser le processus précédent à l'aide d'AES. Lorsque l'on souhaite déchiffrer un fichier, il faut sélectionner le fichier chiffré précedemment (nom du fichier original concatené avec le mot *_encrypted*) et entrer la clé secrète que l'on a choisi au moment du chiffrement. Le fichier déchiffré portera le nom du fichier chiffré qu'avant avec le mot *_decrypted*. Par exemple on aura *filename_encrypted_decrypted.json* pour un fichier déchiffré de type JSON.

3.9.2 IMPLÉMENTATION DE L'ALGORITHME DE CRYPTOGRAPHIE BLOWFISH

Nous avons implémenté l'algorithme Blowfish en utilisant le module PyCrypto. Le package PyCrypto en Python fournit divers algorithmes de chiffrement. Le package PyCrypto peut être installé à l'aide de la commande suivante *pip install pycrypto*. Le package PyCrypto contient un module Blowfish qui permet de chiffrer et de déchiffrer les données à l'aide de l'algorithme Blowfish. Pour le chiffrement, il faudra :

1. Importer le module blowfish de Crypto.Cipher
2. Créer un chiffrement en utilisant la méthode new() dans le module blowfish qui new génère une clé pour chiffrer et déchiffrer les données.
3. Obtenir les données à chiffrer. Notons que la longueur des données doit être un multiple de 8.
4. Utiliser la méthode encrypt() pour chiffrer les données.
5. Enfin, imprimer les données chiffées.

Pour ce qui est de la partie déchiffrement de Blowfish, il faudra :

1. Utiliser la méthode decrypt() pour déchiffrer les données chiffrées.
2. Enfin, imprimer les données déchiffrées.

3.10 ENCODAGE ET DÉCODAGE DES DONNÉES AVEC AES-256 ET BLOWFISH

La section 3.10 nous montre les différents résultats de chiffrement et de déchiffrement de données de vidéos, d'images et UWB avec les algorithmes de cryptographie AES-256 et Blowfish.

3.10.1 ENCODAGE ET DÉCODAGE DES VIDÉOS

Pour ces algos, nous entrerons directement une vidéo dans chacun des algorithmes. Par exemple nous avons mis la vidéo *test4.mp4* à sécuriser avec AES. La vidéo a été dans un premier temps chiffrée et a donné le fichier *filename_encrypted*. Puis ce fichier a été déchiffré à son tour en donnant le fichier *testvideo.mp4_encrypted_decrypted.mp4*. Par la suite nous avons comparé ces deux fichiers pour trouvé la similarité cosinus entre eux. Le tableau 3.5 nous montre les différents résultats de tests de chiffrement et de déchiffrement.

TABLEAU 3.5 : Étude comparative des différents algos avec leur mesures de performances.

	AES-256	**Blowfish**
Similarité cosinus	1	1
Temps de chiffrement (ms)	0.177	0.167
Temps de déchiffrement (ms)	0.170	0.223

L'objectif du chiffrement de vidéos est d'obtenir une vidéo de même format et de taille égale au maximum à la taille de la vidéo originale. En effet à la sortie on obtient la même taille des fichiers comme chez les AE.

Nous voyons que l'algorithme AES-256 et Blowfish donne tous deux 1 comme similarité cosinus. Concernant le temps de chiffrement de AES pour les vidéo, il est de 0.117 ms. Le temps de dechiffrement est de 0.170 ms.

Le temps de chiffrement de la vidéo par Blowfish est de 0.1382 ms tandis que le temps de dechiffrement de la vidéo est de 0.2365 ms.

3.10.2 ENCODAGE ET DÉCODAGE DES IMAGES AVEC AES-256 ET BLOWFISH

Pour les images nous avons utilisé la même méthode que celle avec les vidéos et avons obtenus les résultats présentés dans le tableau 3.5.

TABLEAU 3.6 : Résultats de codage et de décodage des images

	AES-256	**Blowfish**
Similarité cosinus	1	1
Temps de chiffrement (ms)	0.112	0.138
Temps de déchiffrement (ms)	0.159	0.236

3.10.3 ENCODAGE ET DÉCODAGE DES UWB AVEC AES-256 ET BLOWFISH

À ce niveau, nous trouvons aussi comme similarité cosinus 1. Les fichiers UWB chiffrées et déchiffrées sont exactement les mêmes. Pour le temps de chiffrement et de déchiffrement des UWB on peut le voir dans le tableau 3.7.

TABLEAU 3.7 : Résultats de codage et de décodage des UWB

	AES-256	**Blowfish**
Similarité cosinus	1	1
Temps de chiffrement (ms)	1384.21	2745.47
Temps de déchiffrement (ms)	1107.29	6276.05

3.11 ANALYSE ET DISCUSSION

Avant de conclure ce mémoire, il apparait essentiel d'analyser les résultats que nous avons pu observés dans les sections précédentes. Premièrement, nous pouvons remarqué que l'AE simple performe plus que les AE profond et convolutionnel avec les données images, vidéos, UWB et la similarité cosinus atteint jusqu'à 0.99. En même temps, on remarque que ce modèle d'AE met moins de temps pour encoder et decoder les données comparées aux autres AE.

Puis, contrairement à notre hypothèse de départ, alors que l'AE convolutionnel est très adapté pour le chiffrement et déchiffrement des images et vidéos, nous observons que sa similarité cosinus moindre par rapport à l'AE simple et profond. Cela pourrait être expliqué par la perte d'information dans la reconstruction. En effet, la convolution et le *max pooling* s'inverse avec une perte. Ainsi, on ne peut pas retrouver les valeurs d'origine. De plus, le CAE prend énormément de temps pour chiffrer et déchiffrer les vidéos, soit 22 300 ms pour le chiffrement et 24 523.05 ms pour le déchiffrement. Évidemment, ce temps peut sembler inhabituel, puisque lorsqu'on compare, par exemple, avec l'encodage/décodage des images de la célèbre banque de données MNIST, la tâche est exécutée beaucoup plus rapidement. Cependant, il faut comprendre que nos données sont beaucoup plus volumineuses, nos images ont une taille de (1280,720) alors que celles de MNIST ne sont que de (28,28) pixels.

De l'autre coté, si l'on se penche sur les algorithmes de cryptographie traditionnelle AES-256 et Blowfish, on voit que la similarité cosinus est de 1 pour qu'importe le type de données que nous leurs donnons. Ce n'est pas vraiment surprenant puisque ces algorithmes réalisent le chiffrement sans perte d'information. Puisque c'est une fonction mathématique simple qui permet de faire le chiffrement, elle peut être directement inversée avec la clé pour retrouver les données d'origine. Ce n'est évidemment pas le cas des AE qui compressent

d'abord les données avec perte dans l'espace latent et apprennent à les recréer de toute pièce dans la partie décodeur. Nous nous attendions donc à ce que les algorithmes de cryptographie traditionnelle performent mieux que notre approche sur cet aspect.

C'est au niveau du temps où notre analyse est un peu plus décevante. Les expérimentations n'ont pas donné les résultats espérés. Après plusieurs tests, les résultats ne se sont pas améliorés et nos temps sont très loin de ceux des algorithmes classiques. Nous avons cependant une bonne idée des raisons qui peuvent expliquer cette différence. De notre côté, nous avons développé directement en Python et notre programme fait appel à plusieurs librairies dont le chargement et le démarrage peut avoir un impact imprévisible/incrontrôlable sur le temps d'exécution. De plus, nos données sont probablement transformées initialement par Tensorflow, ajoutant des délais supplémentaires. D'un autre côté, les algorithmes de cryptographie traditionnelle avec lesquels nous nous sommes comparés sont programmés en C/C++ et ont été optimisés au maximum par les développeurs pour gagner chaque millième de seconde possible.

Notre travail est évidemment de nature exploratoire et bien que nos résultats soient peu concluants, ils semblent montré qu'il faut pousser les travaux de recherche dans les prochaines années. En effet, il faudrait impérativement réussir à comparer la vitesse des méthodes sur une base plus juste, mais cela requièrerait une expertise un peu différente de celle que nous avons développée. Un autre aspect intéressant à explorer serait l'éprouvement de la méthode par sa mise en oeuvre et par des tentatives d'attaques sur celle-ci.

Pour terminer, il faut noter que les résultats n'invalident pas les avantages de notre approche. Nous pensons, entres autres, que la possibilité de scinder le réseau en deux morceaux pourra être pratique pour les réseaux distribués qu'on retrouve habituellement dans les habitats intelligents. De plus, notre méthode pourra supporter les pertes dans les données, puisqu'en

réalité, dans les habitats intelligents, nous n'avons généralement pas besoin des données brutes pour appliquer des méthodes d'intelligence artificielle. Par exemple, nous pourrions imaginer un modèle de reconnaissance d'activités sur un serveur distant qui recevrait les données encodées plutôt que sur les données brutes pour analyser ce qui se passe dans l'habitat.

Pour le lecteur, il faut préciser que notre travail est exploratoire et n'a pas encore été testé dans les habitats intelligents réels. C'est un premier travail dans un créneau innovant qui devrait ouvrir la voie pour de nombreuses années de recherche. Il y a donc plusieurs zones d'ombre à éclaircir puis des parties à améliorer mais nous les reservons pour les travaux futurs. Par exemple, nous visons à tester des modèles de réseaux de neurones plus complexes que ceux vu dans le mémoire. Nous pensons aussi qu'il serait intéressant d'implémenter différentes techniques pour tenter de déchiffrer les données encodées avec les réseaux de neurones afin de vérifier empiriquement qu'ils sont sécuritaires. Enfin, les réseaux devront être implémentés de façon plus efficace pour qu'ils puissent être utilisés en temps réel.

CHAPITRE IV
CONCLUSIONS ET PERSPECTIVES

Le Chapitre 4 conclut les différents travaux ayant permis de réaliser ce mémoire. Nous avons vu qu'il existait différents moyens de chiffrer et de déchiffrer les données comme les algorithmes de cryptographie traditionnelle tels que AES-256, Blowfish, SHA-256. Nous les avons testés avec nos données du LIARA et en parallèle nous avons aussi implémenté une solution de sécurisation de données d'habitats intelligents avec des autoencodeurs. Cette méthode plutôt nouvelle et intéressante semble être une belle alternative dans la sécurisation des données surtout celles étant plus complexes. Notre dernier chapitre énumère aussi les contributions apportées par ce document ainsi que des possibles axes d'amélioration à effectuer dans les travaux futurs pour améliorer cette étude.

4.1 REVUE DE CONTRIBUTIONS

Une des contributions est l'implémentation d'un système de chiffrement et de déchiffrement de données (données tant lourdes et complexes que simples) à distance. Les données pourront transiter en toute sécurité et en temps réel sur le réseau. C'est un pas de plus dans la sécurisation d'habitats avec les autoencodeurs, un contexte plutôt nouveau. Les résultats obtenus représenterons une première étape pour les chercheurs afin d'explorer l'encodage des données pour sécuriser les habitats intelligents. Un autre aspect interressant qu'offre ce mémoire est l'étude comparative de la cryptographie traditionnelle et des réseaux de neurones.

Aussi l'ensemble de données collectées au LIARA (des types de données qu'on ne retrouve pas souvent) pourrait eventuellement être utilisé par les chercheurs et scientifiques

pour développer et/ou faire une comparaison des différents types d'algorithmes d'intelligence artificielle et systèmes pour l'assistance à domicile et la sécurisation des données.

4.2 LIMITES

En ce qui concerne notre système de sécurisation des données développé, il n'a pas été déployé dans les conditions réelles d'un habitat intelligent. Même si de nombreuses expérimentations ont été effectuées en laboratoire dans le but de se rapprocher de ces conditions, il reste tout de même une part d'inconnu vis-à-vis d'un déploiement complet. Par exemple, il se peut que certaines interactions entre des humains, des capteurs ou encore l'environnement en lui-même n'aient été pas prévues lors de la conception du système et qu'il en résulte une série de fonctionnement imprévus.

Les réseaux de neurones, comme pour la plupart des algorithmes d'apprentissage machine sont sensibles à la qualité des données. Il est donc important d'avoir suffisamment de données pour extraire les caractéristiques générales d'un problème, mais il est également important de prêter attention à tout biais de données. Outre la qualité, le volume de données disponibles est également primordial pour la capacité d'apprentissage d'un réseau de neurones. Plus le modèle est alimenté en données, plus l'algorithme d'apprentissage est performant et offre des résultats précis.

Aussi pour le fait que nos CAE avaient de moins bon résultats que les autres AE, c'est-à dire AE simple et AE profond, nous avons utilisé le *max pooling* dans notre architecture. Pour être plus clair, le but des couches de *pooling* est de rendre les cartes moins détaillées afin d'obtenir une invariance spatiale. Semblable à la phase de convolution, le *stride* est également choisi pour l'opération de *pooling*. Cette procédure diminue également la possibilité de surapprentissage. Deux types d'opérations de pooling sont utilisés habituellement : le *max*

pooling et l'*average pooling*. Le *max pooling* ne retient que seulement le maximum entre les valeurs sélectionnées dans la carte de caractéristique ce qui entraine des pertes.

Pour bien entraîner les paramètres des réseaux de neurones cela prend un CPU et un GPU de très haute spécification et de traitement. Il a également besoin d'une Random Access Memory (RAM) supérieure à 12 Go pour traiter ces modèles. Pendant nos différents tests, nous avons été confrontés à un problème de RAM insuffisante pour entrainer certains de nos modèles d'AE. Nous avons du sursoir certains entrainement d'AE dû à l'absence d'une machine aussi puissante.

4.3 TRAVAUX FUTURS

Aucun travail n'est ni complet ni parfait. Il y'a toujours des zones à améliorer. À la suite de nos travaux on peut énumerer ou prévoir plusieurs futurs travaux.

Pour commencer, nous pourrions étendre l'étude en faisant ces expérimentations dans un environnemnet réel avec un ou des résidents vivant dans un habitat intelligent. Puis on pourrait chiffrer ces données avec notre encodeur puis de l'autre bord avoir un vrai centre de surveillance avec des cliniciens qui pourront déchiffrer les données avec le décodeur installé a leur niveau. Certes, nous avons utilisé les données du LIARA mais cela aurait être encore plus réaliste.

Ensuite, puisque lors de nos différentes expérimentations nous avons fait face aux problèmes d'insuffisance de la RAM, il serait plus judicieux à l'avenir d'utiliser des machines plus puissantes pour avoir de meilleurs résultats lors des tests et surtout pour mieux entrainer nos algorithmes et modèles d'apprentissage machine.

Puis, l'on pourrait étendre l'analyse en faisant l'étude comparative avec d'autres algorithmes de cryptographie comme SHA-256 ou encore HMAC-SHA-256.

Enfin, il apparaît important de souligner l'importance, dans le futur, d'implémenter les réseaux de neurones de façon plus efficace. En effet, que ce soit au niveau du code en lui-même, des appels aux librairies externes ou bien du chargement des données, si nous souhaitons que notre approche soit utilisée, sa vitesse d'exécution devra minimalement se rapprocher de celle des algorithmes de cryptographie classique.

4.4 CONCLUSION PERSONNELLE

Ce document représente pour moi plusieurs heures de travail, de la volonté, des nuits blanches mais surtout un accomplissement de soi. J'ai toujours adoré chercher et surtout trouver mais j'avoue que ce travail a été tumultueux. Il y a eu des bouts beaucoup plus longs qui m'ont amené à douter de sa réalisation mais je n'ai rien lâché et j'y ai mis tout mon cœur. J'ai développé plusieurs compétences sur les réseaux de neurones et au final je suis fière de moi. J'espère que ce travail pourra aider beaucoup de personnes et j'ai hâte de l'approfondir très prochainement.

BIBLIOGRAPHIE

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. *et al.* (2016). Tensorflow : a system for large-scale machine learning. Dans *Osdi*, Vol. 16, pp. 265–283. Savannah, GA, USA.

Abdul-Qawy, A. S., Pramod, P., Magesh, E. & Srinivasulu, T. (2015). The internet of things (iot) : An overview. *International Journal of Engineering Research and Applications*, *5*(12), 71–82.

Alabaichi, A., Ahmad, F. & Mahmod, R. (2013). Security analysis of blowfish algorithm. Dans *2013 Second International Conference on Informatics & Applications (ICIA)*, pp. 12–18. IEEE.

Alam, M. R., Reaz, M. B. I. & Ali, M. A. M. (2012). A review of smart homes—Past, present, and future. *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, *42*(6), 1190–1203.

Alberdi, A., Weakley, A., Schmitter-Edgecombe, M., Cook, D. J., Aztiria, A., Basarab, A. & Barrenechea, M. (2018). Smart home-based prediction of multidomain symptoms related to Alzheimer's disease. *IEEE journal of biomedical and health informatics*, *22*(6), 1720–1731.

Ameen Suhail, K. & Sankar, S. (2020). Image compression and encryption combining autoencoder and chaotic logistic map. *Iranian Journal of Science and Technology, Transactions A : Science*, *44*(4), 1091–1100.

Arab, A., Rostami, M. J. & Ghavami, B. (2019). An image encryption method based on chaos system and AES algorithm. *The Journal of Supercomputing*, *75*(10), 6663–6682.

Association, A. *et al.* (2018). 2018 Alzheimer's disease facts and figures. *Alzheimer's & Dementia*, *14*(3), 367–429.

Borne, P., Benrejeb, M. & Haggége, J. (2007). *Les réseaux de neurones : présentation et applications*, Vol. 15. Editions OPHRYS.

Botterman, M. (2009). Internet of Things : an early reality of the Future Internet. Dans *Workshop Report, European Commission Information Society and Media*, Vol. 15. sn.

Bouchard, K., Bouchard, B. & Bouzouanea, A. (2014). Practical guidelines to build smart homes : lessons learned. *Opportunistic networking, smart home, smart city, smart systems*, pp. 1–37.

Catherwood, P. A., Steele, D., Little, M., Mccomb, S. & McLaughlin, J. (2018). A community-based IoT personalized wireless healthcare solution trial. *IEEE journal of translational engineering in health and medicine*, *6*, 1–13.

Chan, M., Campo, E., Estève, D. & Fourniols, J.-Y. (2009). Smart homes—current features and future perspectives. *Maturitas*, *64*(2), 90–97.

Chandra, S., Paira, S., Alam, S. S. & Sanyal, G. (2014). A comparative survey of symmetric and asymmetric key cryptography. Dans *2014 international conference on electronics, communication and computational engineering (ICECCE)*, pp. 83–93. IEEE.

Chappell, N. L., Dlitt, B. H., Hollander, M. J., Miller, J. A. & McWilliam, C. (2004). Comparative costs of home care and residential care. *The Gerontologist*, *44*(3), 389–400.

Charniya, N. N. (2013). Design of near-optimal classifier using multi-layer perceptron neural networks for intelligent sensors. *International Journal of Modeling and Optimization*, *3*(1), 56.

Cheng, J., Sundholm, M., Zhou, B., Hirsch, M. & Lukowicz, P. (2016). Smart-surface : Large scale textile pressure sensors arrays for activity recognition. *Pervasive and Mobile Computing*, *30*, 97–112.

Chowdhary, C. L., Mittal, M., Pattanaik, P. & Marszalek, Z. (2020). An efficient segmentation and classification system in medical images using intuitionist possibilistic fuzzy C-mean clustering and fuzzy SVM algorithm. *Sensors*, *20*(14), 3903.

Chowdhary, C. L., Muatjitjeja, K. & Jat, D. S. (2015). Three-dimensional object recognition based intelligence system for identification. Dans *2015 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*, pp. 162–166.

IEEE.

Clavet, N.-J., Duclos, J.-Y., Fortin, B., Marchand, S. & Michaud, P.-C. (2013). Les dépenses en santé du gouvernement du Québec, 2013-2030 : projections et déterminants.

Colloque sur le vieillissement de la population et les contraintes financieres du secteur de la sante. & Conseil économique du Canada. (1987). *Troisieme age et soins de sante : actes*. Le Conseil. Section : vi, 253 pages.

Cook, D. & Das, S. K. (2004). *Smart environments : technology, protocols, and applications*, Vol. 43. John Wiley & Sons.

Daemen, J. & Rijmen, V. (2002). *The design of Rijndael*, Vol. 2. Springer.

Ding, Y., Wu, G., Chen, D., Zhang, N., Gong, L., Cao, M. & Qin, Z. (2020). DeepEDN : a deep-learning-based image encryption and decryption network for internet of medical things. *IEEE Internet of Things Journal*, *8*(3), 1504–1518.

Dumont, R. (2006). Introduction a la cryptographie et à la sécurité informatique. *Notes de cours provisoires*, *2007*.

Dworkin, M. J. (2015). SHA-3 standard : Permutation-based hash and extendable-output functions.

Fortin-Simard, D., Bilodeau, J.-S., Bouchard, K., Gaboury, S., Bouchard, B. & Bouzouane, A. (2015). Exploiting passive RFID technology for activity recognition in smart homes. *IEEE Intelligent systems*, *30*(4), 7–15.

Gaikwad, P. P., Gabhane, J. P. & Golait, S. S. (2015). A survey based on Smart Homes system using Internet-of-Things. Dans *2015 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC)*, pp. 0330–0335. IEEE.

Ghernaouti, S. (2013). *Sécurité informatique et réseaux-4e édition : Cours avec plus de 100 exercices corrigés*. Dunod.

Gilmour, H. (2018). Formal home care use in Canada. *Health Rep*, *29*(9), 3–9.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep learning*. MIT press.

Gupta, C. (2020). Shallow Encoder Deep Decoder (SEDD) Networks for Image Encryption and Decryption. *arXiv preprint arXiv :2001.03017*.

Halima, I. (2021). *Suivi de l'activité d'une personne à partir de capteurs multi-modalités préservant l'anonymat dans un cadre de détection et prévention des chutes chez les personnes âgées*. *Suivi de l'activité d'une personne à partir de capteurs multi-modalités préservant l'anonymat dans un cadre de détection et prévention des chutes chez les personnes âgées*.

Hamad, S., Khalifa, A., Elhadad, A. & Rida, S. (2013). A modified playfair cipher for encrypting digital images. *Mod. Sci*, *3*, 76–81.

Hao, Y. & Foster, R. (2008). Wireless body sensor networks for health-monitoring applications. *Physiological measurement*, *29*(11), R27.

Hardi, S., Tarigan, J. & Safrina, N. (2018). Hybrid cryptosystem for image file using elgamal and double Playfair cipher algorithm. Dans *Journal of Physics : Conference Series*, Vol. 978, p. 012068. IOP Publishing.

Hashim, H. R. & Neamaa, I. A. (2014). Image encryption and decryption in a modification of ElGamal cryptosystem in MATLAB. *arXiv preprint arXiv :1412.8490*.

Haxhibeqiri, J., De Poorter, E., Moerman, I. & Hoebeke, J. (2018). A survey of LoRaWAN for IoT : From technology to application. *Sensors*, *18*(11), 3995.

Hemamalini, V., Zayaraz, G., Susmitha, V., Gayathri, M. & Dhanam, M. (2016). A Survey on Elementary, Symmetric and Asymmetric Key Cryptographic Techniques. *International Journal of Computing Academic Research (IJCAR)*, pp. 11–26.

Hevesi, P., Wille, S., Pirkl, G., Wehn, N. & Lukowicz, P. (2014). Monitoring household activities and user location with a cheap, unobtrusive thermal sensor array. Dans *Proceedings*

of the 2014 ACM international joint conference on pervasive and ubiquitous computing, pp. 141–145.

Hinton, G. E. & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504–507.

Hu, F., Pu, C., Gao, H., Tang, M. & Li, L. (2016). An image compression and encryption scheme based on deep learning. *arXiv preprint arXiv :1608.05001*.

Hu, F., Wang, J., Xu, X., Pu, C. & Peng, T. (2017). Batch image encryption using generated deep features based on stacked autoencoder network. *Mathematical Problems in Engineering*, *2017*.

Jin, X., Duan, X., Jin, H. & Ma, Y. (2020). A novel hybrid secure image encryption based on the shuffle algorithm and the hidden attractor chaos system. *Entropy*, *22*(6), 640.

Kadowaki, L., Wister, A. V. & Chappell, N. L. (2015). Influence of home care on life satisfaction, loneliness, and perceived life stress. *Canadian Journal on Aging/La Revue canadienne du vieillissement*, *34*(1), 75–89.

Kinzel, W. & Kanter, I. (2002). Interacting neural networks and cryptography. Dans *Advances in solid state physics* pp. 383–391. Springer.

Kumar, M. A. & Karthikeyan, S. (2012). Investigating the efficiency of Blowfish and Rejindael (AES) algorithms. *International Journal of Computer Network and Information Security*, *4*(2), 22.

Liu, T., Wang, J., Liu, Q., Alibhai, S., Lu, T. & He, X. (2021). High-ratio lossy compression : Exploring the autoencoder to compress scientific data. *IEEE Transactions on Big Data*.

Mahajan, P. & Sachdeva, A. (2013). A study of encryption algorithms AES, DES and RSA for security. *Global Journal of Computer Science and Technology*.

Maitre, J., Bouchard, K., Bertuglia, C. & Gaboury, S. (2021). Recognizing activities of daily living from UWB radars and deep learning. *Expert Systems with Applications*, *164*,

113994.

Maniyath, S. R. & Thanikaiselvan, V. (2020). An efficient image encryption using deep neural network and chaotic map. *Microprocessors and Microsystems*, *77*, 103134.

Maqsood, F., Ahmed, M., Ali, M. M. & Shah, M. A. (2017). Cryptography : a comparative analysis for modern techniques. *International Journal of Advanced Computer Science and Applications*, *8*(6).

Masram, R., Shahare, V., Abraham, J. & Moona, R. (2014). Analysis and comparison of symmetric key cryptographic algorithms based on various file features. *International Journal of Network Security & Its Applications*, *6*(4), 43.

Mbuyamba, E. I., Kalonji, S. T. *et al.* (2023). Chiffrement et déchiffrement d'images à l'aide de l'algorithme AES. *European Journal of Computer Science and Information Technology*, *11*(1), 1–8.

Mortajez, S., Tahmasbi, M., Zarei, J. & Jamshidnezhad, A. (2020). A novel chaotic encryption scheme based on efficient secret keys and confusion technique for confidential of DICOM images. *Informatics in Medicine Unlocked*, *20*, 100396.

Mousse, A. M. (2016). Reconnaissance d'activités humaines à partir de séquences multi-caméras : application à la détection de chute de personne. Université du Littoral-Côte d'Opale.

Mushtaq, M. F., Jamel, S., Disina, A. H., Pindar, Z. A., Shakir, N. S. A. & Deris, M. M. (2017). A survey on the cryptographic encryption algorithms. *International Journal of Advanced Computer Science and Applications*, *8*(11).

Naureen, A., Akram, A., Maqsood, T., Riaz, R., Kim, K.-H. & Ahmed, H. F. (2008). Performance and security assessment of a PKC based key management scheme for hierarchical sensor networks. Dans *VTC Spring 2008-IEEE Vehicular Technology Conference*, pp. 163–167. IEEE.

Noury, N., Virone, G., Barralon, P., Rialle, V. & Demongeot, J. (2004). Maisons intelligentes pour personnes âgées : technologies de l'information intégrées au service des soins à

domicile. *J3eA*, *3*, 020.

Offermann-van Heek, J., Schomakers, E.-M. & Ziefle, M. (2019). Bare necessities ? How the need for care modulates the acceptance of ambient assisted living technologies. *International journal of medical informatics*, *127*, 147–156.

Padmavathi, B. & Kumari, S. R. (2013). A survey on performance analysis of DES, AES and RSA algorithm along with LSB substitution. *IJSR, India*, *2*, 2319–7064.

Patel, A. A. (2019). *Hands-on unsupervised learning using Python : how to build applied machine learning solutions from unlabeled data*. O'Reilly Media.

Patil, A. & Goudar, R. (2013). A Comparative Survey of Symmetric Encryption Techniques for Wireless Devices. *International journal of scientific & technology research*, *2*(8).

Payeur, F. F., Azeredo Teixeira, A. C., St-Amour, M., André, D., Girard, C., Berthiaume, P. & Bureau de la statistique du Quebec. (2019). *Perspectives démographiques du Québec et des régions, 2016-2066*. Québec: Institut de la statistique du Québec.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. *et al.* (2011). Scikit-learn : Machine learning in Python. *the Journal of machine Learning research*, *12*, 2825–2830.

Ramachandran, P., Zoph, B. & Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv :1710.05941*.

Ray, A. K. & Bagwari, A. (2020). IoT based Smart home : Security Aspects and security architecture. Dans *2020 IEEE 9th international conference on communication systems and network technologies (CSNT)*, pp. 218–222. IEEE.

Ribordy, G. & Trinkler, P. (2011). Physique quantique et cryptographie. *Bulletin des SEV VSE Including Jahresheft*, *102*(7), 27.

Sanger, T. D. (1989). Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural networks*, *2*(6), 459–473.

Saraf, K. R., Jagtap, V. P. & Mishra, A. K. (2014). Text and image encryption decryption using advanced encryption standard. *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, *3*(3), 118–126.

Schneier, B. (1994). The Blowfish encryption algorithm. *Dr Dobb's Journal-Software Tools for the Professional Programmer*, *19*(4), 38–43.

Shannon, C. E. (1949). Communication in the presence of noise. *Proceedings of the IRE*, *37*(1), 10–21.

Singh, G. (2013). A study of encryption algorithms (RSA, DES, 3DES and AES) for information security. *International Journal of Computer Applications*, *67*(19).

Slade, S., Shrichand, A. & Et Dimillo, S. (2019). Une étude sur les soins médicaux prodigués aux personnes âgées au Canada.

Solaiman, B. & Richard, L. (2003). *Les réseaux de neurones artificiels et leurs applications en imagerie et en vision par ordinateur*. Presse Universitaire du Québec.

SOW, A. M. (2020). *CLASSIFICATION, RÉDUCTION DE DIMENSIONNALITÉ ET RÉSEAUX DE NEURONES : DONNÉES MASSIVES ET SCIENCE DES DONNÉES*. (Mémoire de maîtrise). Université du Québec à Trois Rivières. Repéré à https://depot-e.uqtr.ca/id/eprint/9600/

Stallings, W. (2002). *Sécurité des réseaux : applications et standards*. Vuibert.

Stallings, W. (2006). *Cryptography and network security principles and practices 4th edition*.

Turchenko, V., Chalmers, E. & Luczak, A. (2017). A deep convolutional auto-encoder with pooling-unpooling layers in caffe. *arXiv preprint arXiv :1701.04949*.

Turner, J. M. (2008). The keyed-hash message authentication code (hmac). *Federal Information Processing Standards Publication*, *198*(1), 1–13.

Vidhya, S. (2018). Network Security using Python. *2018 IJSRSET*, *4*(4).

Vincent, P., Larochelle, H., Bengio, Y. & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. Dans *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103.

Voulodimos, A., Doulamis, N., Doulamis, A. & Protopapadakis, E. (2018). Deep learning for computer vision : A brief review. *Computational intelligence and neuroscience*, *2018*.

Wimo, A., Gauthier, S., Prince, M. *et al.* (2018). Global estimates of informal care. *World Alzheimer report. London : Alzheimer's disease international (ADI) and Karolinska Institute*.

Yildirim, O., San Tan, R. & Acharya, U. R. (2018). An efficient compression of ECG signals using deep convolutional autoencoders. *Cognitive Systems Research*, *52*, 198–211.

Zhang, Y., Li, X. & Hou, W. (2017). A fast image encryption scheme based on AES. Dans *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, pp. 624–628. IEEE.

Zhou, I., Makhdoom, I., Shariati, N., Raza, M. A., Keshavarz, R., Lipman, J., Abolhasan, M. & Jamalipour, A. (2021). Internet of things 2.0 : Concepts, applications, and future directions. *IEEE Access*, *9*, 70961–71012.

Zhou, N., Zhang, A., Zheng, F. & Gong, L. (2014). Novel image compression–encryption hybrid algorithm based on key-controlled measurement matrix in compressive sensing. *Optics & Laser Technology*, *62*, 152–160.

APPENDICE A

Ce mémoire de maitrîse a fait l'objet d'une certification éthique émis par le comité de la recherche de l'UQAC. Le numéro du certificat éthique est le 2021-487.