ANNEE 2021/2022

UNIVERSITÉ
CAEN
NORMANDIE

RAPPORT DE STAGE

présenté par

"Jarossay Max"

Mission effectuée du 09/03/2022 au 31/08/2022 au :

GREYC

Sujet de la mission :

"Analyse de la dynamique de frappe au clavier sonore pour l'identification, le profilage et l'extraction du texte saisi"

Directeur de stage : **"Christophe Rosenberger"**

Tuteur école : **"Marc Spaniol"**

Table des matières

Remerciements **2**

Résumé **3**

Abstract **3**

1 Le laboratoire de recherche GREYC **4**
1.1 Présentation du GREYC 4
1.2 Présentation de l'équipe SAFE 5
1.3 Espace de travail et matériel 5

2 Introduction **6**
2.1 Explication du sujet / Contexte 6
2.2 État de l'art 7
2.3 Voie à suivre / Chemin choisi 10

3 Création d'un corpus de mots **11**

4 Identification d'un individu **13**
4.1 Procédé 13
4.2 Résultats 13

5 Détection du texte saisi : "Détection mot par mot" **16**
5.1 Données diverses 16
5.2 Explication du fonctionnement 16
5.2.1 Préparation 16
5.2.2 Reconnaissance sans connaître la position des touches sur le fichier audio 17
5.3 Résultats 19

6 Conclusion **21**
6.1 Annexe 22

Remerciements

Dans le cadre de mon M2 informatique spécialisé "Sécurité des Systèmes Informatiques" à l'Université de Caen, j'ai réalisé un stage au sein du laboratoire de recherche GREYC et ce pour une durée de 6 mois. Je tiens donc à remercier le laboratoire GREYC de m'avoir accueilli en son sein et plus particulièrement j'aimerais remercier Mr. Christophe Rosenberger et Mr. Emmanuel Giguet de m'avoir épaulé tout au long de mon stage.

Résumé

Dans le cadre de mon Master 2 en informatique spécialité SSI, j'ai réalisé un stage au laboratoire de recherche GREYC, au sein de l'équipe SAFE. J'ai choisi de faire mon stage là bas car j'avais envie de découvrir en quoi consistait le travail dans le domaine de la recherche qui m'était jusqu'alors inconnu.

Le stage que j'ai réalisé était composé de deux axes principaux. Le premier axe du stage consistait à pouvoir identifier un individu uniquement grâce au son que celui-ci peut émettre quand il tape sur les touches de son clavier. Le deuxième axe consistait quant à lui à essayer de deviner le texte tapé par une personne sur un clavier et ce encore une fois uniquement grâce au bruit que peut produire un individu en tapant sur les touches de son clavier.

Les résultats obtenus grâce aux expérimentations et à la base de données d'enregistrement de personnes tapant au clavier (que nous avons créé pour l'occasion), nous montre que le son que l'on peut émettre quand nous tapons au clavier pourrait s'avérer dangereux si des personnes mal intentionnées le récupérait.

Abstract

As part of my Master's degree in computer science, I did an internship at the GREYC research laboratory, in the SAFE team. I choose to do my internship there because I wanted to discover what the work in the research field consisted of, which was unknown to me until then.

The internship I did was composed of two main axes. The first axis of the internship consisted in being able to identify a person only thanks to the sound that this one can do when he is typing on the keys of his keyboard. The second axis consisted in trying to guess the text tapped by a person on a keyboard and this only thanks to the noise that an individual can produce while tapping on the keys of his keyboard.

The results obtained from the experiments and due to the database of people typing on keyboard (which we created for the occasion), show us that the sound we make when we type could be dangerous if it is collected by people with bad intentions.

1 Le laboratoire de recherche GREYC

1.1 Présentation du GREYC

Le GREYC est un laboratoire de recherche créé en 1995 à Caen (voir le logo ci-dessous fig. 1) qui se situe sur le campus 2, bâtiment F, 6 Boulevard Maréchal Juin. Le GREYC est depuis 2000 devenu une unité de recherche mixte associé au CNRS, à l'Université de Caen ainsi qu'à l'École nationale supérieure d'ingénieur. Le GREYC est séparé en 6 équipes distinctes chacune travaillant dans un domaine de l'informatique.
Il y a l'équipe AMAAC qui est spécialisée dans la recherche en informatique mathématique ainsi que dans les modèles de calculs, et ce que ce soit de façon aléatoire ou bien en prenant en compte la notion de complexité.
Il y a l'équipe CODAG qui est spécialisée dans la fouille de données, l'équipe se concentre plus particulièrement sur l'ensemble de la chaîne de traitement de données tout en y ajoutant des contraintes qui peuvent être de diverses natures (déontologique, légale, physique, ...).
Il y a l'équipe ÉLECTRONIQUE qui est spécialisée dans les composants électroniques avancés ainsi que dans les capteurs à haute sensibilité, cette équipe est organisée en 3 axes thématiques :

- Axe 1 : Physique des composants à semi-conducteurs
- Axe 2 : Capteurs en couches minces d'oxyde fonctionnels.
- Axe 3 : Systèmes complexes de mesure et de détection.

Il y a l'équipe IMAGE qui est spécialisée dans le développement de nouvelles méthodes de traitement et d'analyse de signaux, vidéos, images ou de données discrètes. Ces méthodes sont ensuite employées dans des cas concrets et ce en collaboration avec des centres de recherche en imagerie médicale.
Il y a l'équipe MAD qui est spécialisée dans l'intelligence artificielle et plus particulièrement dans le raisonnement, la représentation des connaissances, la planification sous incertitude et les systèmes multi-agents.
Il y a l'équipe SAFE qui est spécialisée dans la sécurité informatique, cette équipe est organisée en 3 axes différents :

- Biométrie.
- Architecture et modèles de sécurité.
- Science de l'investigation (Forensique).

Mon stage a donc été réalisé dans l'équipe SAFE du laboratoire de recherche GREYC, dont je vais parler plus en détails dans la sous-section suivante.

Figure 1 – Logo du GREYC

1.2 Présentation de l'équipe SAFE

C'est donc au sein de cette équipe SAFE que j'ai réalisé mon stage. L'équipe est constituée d'une dizaine de membres permanents et est sous la responsabilité de Mr. Christophe Charrier. Comme expliqué précédemment l'équipe SAFE divise son travail en trois axes distincts et je vais ici en parler plus en profondeur.

Biométrie : La biométrie correspond à l'analyse de données physiques d'un individu (voix, empreinte, visage, ...). Dans cette thématique, la team SAFE travaille sur trois axes différents qui sont la définition, l'évaluation et la protection des données biométriques. Le laboratoire a donc pour but de concevoir de nouveaux systèmes biométriques plus performants (plus résistants aux attaques (ex : attaque par présentation)) et qui peuvent être facilement mis à jour. En plus de cela la thématique biométrique est étroitement liée avec la thématique Forensique (que je présenterai par la suite) notamment pour l'identification d'individus ou encore la protection des données biométriques de ces derniers afin de rester dans les clous vis-à-vis du RGPD. La qualité des données biométriques est aussi au coeur de leurs travaux afin d'avoir des données optimales et le plus utiles possible. ;

Architecture et modèles de sécurité : Dans la thématique «Architecture et modèles de sécurité», la team SAFE travaille sur trois axes, la sécurisation des futures technologies des réseaux SDN/5G/6G, la détection des attaques et les contre-mesures associées et enfin l'étude et la modélisation de fonctions booléennes pour des fins de sécurité. La création de nouvelles architectures et de nouveaux modèles pour les réseaux SDN/5G/6G est donc un point important, pour cela l'équipe modélise ces nouvelles architectures à l'aide de logique du premier ordre ainsi que de théorie des graphes. La détection d'attaques fait aussi partie intégrante du domaine de recherche de l'équipe et ce, que ce soit sur des architectures réseaux ou bien même sur les systèmes biométriques (via des attaques par présentation). L'équipe s'intéresse aussi à la sécurité des systèmes basée sur les codes correcteurs, ainsi que les propriétés que ces derniers doivent vérifier afin de garantir la sécurité des utilisateurs. ;

Forensique : Le terme Forensique correspond à l'ensemble des méthodes d'analyse scientifique employées lors de travaux d'investigation (ex : Affaire criminelle). Sur ce thème la team SAFE travaille sur deux axes différents, le traitement automatique des langues et le développement d'une plateforme forensique pour la mise en oeuvre ou l'évaluation de méthodes d'analyse de traces numériques. Le traitement automatique des langues consiste en la détection dans différents types de documents, d'informations textuelles pouvant être intéressantes, telles que des noms, des adresses, des mots de passe, etc... L'équipe développe aussi des outils d'extraction afin de pouvoir récupérer des informations pouvant être de nature variée (analyse de paquets IP, analyse de log, ...) ;

1.3 Espace de travail et matériel

Le matériel mis à ma disposition est un ordinateur fixe avec Windows 10. J'ai ensuite créé un environnement virtuel afin de pouvoir exécuter mon code sans avoir à demander l'autorisation au responsable informatique pour installer toutes les bibliothèques dont j'avais besoin. J'ai utilisé l'IDE Pycharm tout au long de mon projet car celui-ci est très pratique lorsque l'on développe sous python. Par la suite on m'a aussi fournit un ordinateur portable DELL afin de pouvoir plus facilement réaliser mes enregistrements audio grâce aux deux microphones intégrés à cet ordinateur. Afin de communiquer de façon plus facile avec mon tuteur de stage nous avons utilisé l'application Discord qui permet une communication plus simple, surtout les jours de télétravail. Au sein du laboratoire GREYC mon espace de travail est situé au 3^{eme} étage dans une salle spécialement dédiée pour les stagiaires. Mes horaires de travail sont quant à eux de 35h par semaine ce qui fait du 7h par jour de 9h à 17h.

2 Introduction

2.1 Explication du sujet / Contexte

Dans le monde d'aujourd'hui l'informatique et les nouvelles technologies prennent de plus en plus de place dans notre société, un exemple plutôt évident est la démocratisation du télétravail qui est depuis le début de l'épidémie de Covid-19 devenu une norme. Le nombre de réunions réalisées sur Zoom ou encore Microsoft Teams a donc explosé. Cependant cela soulève de nouveaux problèmes. Imaginez qu'en pleine réunion vous répondiez à un mail personnel et que votre interlocuteur par le biais de votre microphone puisse récupérer le son que fait le clavier de votre ordinateur, et bien il pourrait essayer de récupérer des informations sur ce que vous tapez juste avec le son. Voilà le sujet de mon stage, qui a donc pour but de découvrir ce que quelqu'un tape sur son clavier juste avec le son qu'émettent les touches du clavier lorsqu'on les presse, ainsi que d'essayer d'identifier une personne uniquement avec ce son. J'ai présenté ci-dessus un cas concret pour un utilisateur lambda mais il y a bien d'autres domaines dans lesquels ce sujet de recherche pourrait être utile. Par exemple dans le cadre d'une enquête de police on pourrait récupérer le son qu'émet un suspect avec des micros pour savoir ce qu'il est en train d'écrire sur son ordinateur. On peut voir sur la figure(fig. 2), un schéma qui explique simplement en quoi consiste le procédé qui permet d'identifier une personne.

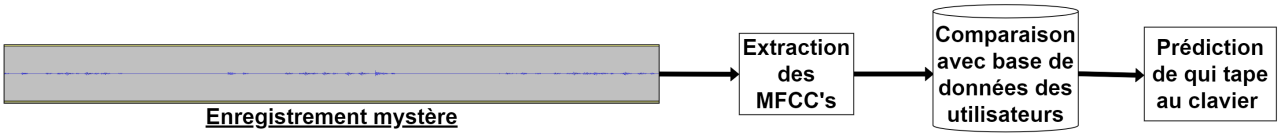

FIGURE 2 – Explication simple du procédé d'identification d'individu

On peut voir sur la figure (fig. 3), un schéma qui explique simplement en quoi consiste le procédé qui permet d'analyser ce qui a été tapé au clavier.

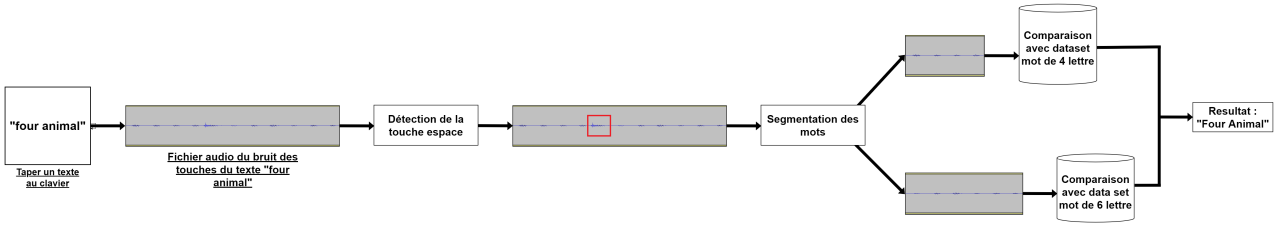

FIGURE 3 – Explication simple du procédé d'analyse du texte tapé au clavier

2.2 État de l'art

De nombreux travaux ont déjà été réalisés sur ce sujet, cependant la grande majorité du temps, les chercheurs ne s'intéressent au problème que de façon théorique sans avoir aucun moyen de l'appliquer dans un cas concret. On va parler de 3 articles mais plus d'articles sont répertoriés dans le tableau dans l'annexe 1

Dans l'article "Keyboard Acoustic Emanations Revisited" [5], les deux chercheurs ont choisi de segmenter les fichiers audio caractère par caractère. Pour cela ils utilisent la transformation de fourrier rapide (FFT) afin de trouver à quelle position se situent les lettres dans le fichier audio. Ils vont donc devoir parcourir le fichier audio par l'intermédiaire d'une fenêtre glissante et définir un seuil avec borne inférieure et supérieure qui, si le son est compris entre ces deux valeurs, dira qu'une touche de clavier a été pressée à cet endroit du fichier audio. Cependant si l'amplitude du son dépasse la borne supérieure, alors le son ne sera pas gardé car il ne provient probablement pas du clavier. Une fois qu'ils ont réussis à localiser où se situe le son d'une touche de clavier sur le fichier audio, les chercheurs vont calculer les MFCC's de cette partie du fichier audio, dans le but d'en récupérer un vecteur, vecteur qui sera ensuite utilisé pour identifier la touche pressée. Afin de faire cette classification, les auteurs ont utilisés un algorithme des "K plus proches voisins". Pour cela, ils ont au préalable enregistré le bruit émis par un clavier, et ce sur 4 datasets différents. Les deux premiers datasets ont été réalisés dans un environnement calme avec aucun bruit parasite. Le premier dataset est composé de 2514 touches de clavier pour un total de 12m17s d'enregistrement, le deuxième dataset est quant à lui composé de 5476 touches de clavier pour un total de 26m56s d'enregistrement. Les datasets 3 et 4 ont quant à eux été réalisés dans un environnement bruyant. Le dataset 3 est composé de 4188 caractères pour une durée d'enregistrement de 21m49s. Pour finir, le dataset 4 est composé de 4300 touches de clavier pour un temps total de 23m54s. Chacun de ces datasets a ensuite été segmentés en 50 classes différentes correspondant chacune à une touche du clavier. Les vecteurs correspondants aux touches que les chercheurs essayent d'identifier sont donc comparés aux différentes classes grâce à l'algorithme KNN et ce dans les 4 datasets afin de savoir quel dataset donnera le meilleur résultat. Une fois toutes les lettres d'un texte identifiées de cette manière, les chercheurs ont choisi d'implémenter du post-traitement TAL à leur algorithme, afin de corriger certaines des erreurs de classification survenues à l'étape précédente. Pour cela, ils ont choisi dans un premier temps d'y implémenter des chaînes de Markov. Ces chaînes de Markov permettent de calculer la probabilité qu'une lettre se situe après une autre (par exemple : Si l'algorithme a trouvé qu'après un "q" il y avait un "x", les chaînes de Markov pourront nous dire que la probabilité que cela arrive est très faible). Les chercheurs vont aussi utiliser un dictionnaire de mots du nom de "ASpell" qui regroupe une grande partie des mots anglais. Ils pourront donc calculer une distance entre le mot obtenu par l'algorithme et tous les mots de même taille présents dans "Aspell" (en se basant sur les espaces pour séparer deux mots). On retournera le mot avec la plus petite distance de hamming avec le mot à identifier. Les résultats obtenus par les chercheurs avec cette méthode sont plutôt encourageants. Dans les datasets 1 et 2, on obtient un total d'environ 35% des mots qui sont bien identifiés et d'environ 75% de caractères bien identifiés dans le cas où l'on ne recourt pas au post-traitement. Si le post-traitement est utilisé, on monte à un total d'environ 70% de mots bien identifiés et à un total d'environ 85% de caractères bien reconnus. Dans les datasets 3 et 4 qui, pour rappel, ont été enregistrés dans un environnement bruyant, on obtient environ 25% de mots bien identifiés et environ 70% de caractères bien reconnus sans post-traitement. Puis si on utilise le post-traitement, on monte à environ 55% de mots bien identifiés et environs 75% de caractères bien reconnus. Tous ces résultats sont plutôt encourageants, mais il y a cependant certaines petites choses qui pourraient être améliorées. Dans un premier temps, lorsque l'on compare le mot à identifier avec des mots du dictionnaire "Aspell", on regarde uniquement les mots de même taille, mais si jamais certaines lettres n'ont pas été captées par l'intermédiaire de la fenêtre glissante, alors le mot ne comportera pas le même nombre de lettres que le mot initial. Dans un deuxième temps, le programme n'arrive pas à savoir quand un utilisateur commence à taper au clavier, il ne fait que regarder quand la FFT d'un signal audio se situe entre deux bornes. Donc si un autre son de même amplitude qu'une touche de clavier se glisse dans ce signal audio, alors le programme essayera de le comparer à une touche parmi les 50 classes différentes, alors que cela n'en est pas une. Cela n'est donc

pas gênant dans l'environnement silencieux, mais dans l'environnement bruyant, si.

Dans l'article intitulé "Don't Skype & Type! Acoustic Eavesdropping in Voice-Over-IP"[3], les auteurs essayent d'appliquer leurs idées au monde réel en imaginant 3 scénarios différents d'attaques, prenant tous comme moyen d'attaque l'application Skype Application permettant de passer des appels téléphoniques ou vidéos via internet. Dans le premier scénario, l'attaquant connaît le bruit de chaque touche du clavier de sa cible. Dans le deuxième scénario, l'attaquant connaît le modèle du clavier de sa cible mais pas le son précis qu'il émet. Dans le dernier scénario, l'attaquant ne possède aucune information sur sa cible. Afin de tester l'algorithme qui sera présenté par la suite, les chercheurs ont dû réaliser une base de données. Cette base de données est constituée de 5 utilisateurs différents qui appuieront chacun leur tour 10 fois sur chacune des touches du clavier et ce sur 6 modèles de claviers différents. Pendant ces enregistrements, les utilisateurs doivent taper avec seulement un doigt pour la première fois où ils toucheront chaque lettre et après ils pourront taper avec tous les doigts. On obtiendra alors un grand nombre de dataset, car chaque dataset sera uniquement constituée d'un utilisateur et d'un modèle de clavier (User1Clavier1, User1Clavier2 , ... , User6Clavier6). Chaque dataset sera donc constitué de 260 touches de clavier au total. Pour le déroulement de l'algorithme en lui même, le procédé est similaire à celui décrit dans l'article précèdent. L'attaquant calcule la FFT d'un signal acoustique grâce à une fenêtre glissante de 10ms, et si la FFT dépasse un certain seuil alors on dira que c'est le son émis par une touche de clavier. On gardera donc en mémoire les données à partir du moment où la touche est repérée, ainsi que des 100ms qui la suivent Cette solution ne fonctionne que dans un environnement totalement silencieux. On utilise encore une fois les MFCC's, que l'on aura calculé grâce au bruit des touches pour faire de la classification, afin de savoir quel bruit correspond à quelle touche. Pour cela on utilisera un algorithme de régression logistique. Dans le cas où on ne connaît pas le modèle du clavier, on utilise un algorithme "KNN" sur plusieurs modèles de clavier, dans le but de trouver dans la base de données un clavier similaire avec lequel on pourra comparer les données du clavier cible. Au final dans le scénario 1, quand on connaît le modèle du clavier et que l'on sait qui tape dessus, on réussit à obtenir un résultat plutôt impressionnant de 97,1% de caractères bien identifiés. Dans le scénario 2, on connaît le modèle de clavier utilisé mais pas la personne qui tape sur celui-ci, les chercheurs comparent donc l'audio à identifier avec tous les datasets utilisant le même clavier. On obtient au final un résultat d'environ 65% de caractères bien identifiés. Pour le scénario 3 où l'on ne connaît pas le modèle du clavier utilisé lors de la saisie d'informations, on cherche à savoir quel modèle de clavier a été utilisé. Pour faire cela on utilise un algorithme de "KNN" puis on compare tous les claviers de nos datasets avec le clavier de la cible. On obtient alors un résultat d'environ 60% de caractères bien identifiés. Certains problèmes liés au support de cet article sont quand même à soulever. En effet on utilise skype pour essayer d'obtenir l'audio de notre cible, mais si jamais la bande passante est inférieur à 40 Kbits/s, la conversation skype va laguer ce qui rendra quasiment impossible la phase d'enregistrement de la personne que l'on essaye d'attaquer. On peut aussi voir que dans le cas ici présent, le même modèle de clavier se situe toujours dans les datasets, mais si on ne possède pas de données audio du bon modèle de clavier, on ne sait pas si cela marche toujours aussi bien. Comparé à l'article "Keyboard Acoustic Emanations Revisited" [5] il n'y a pas du tout de post-traitement TAL pour vérifier si les résultats obtenus sont cohérents.

Dans l'article intitulé "I Know Your Keyboard Input : A Robust Keystroke Eavesdropper Based-on Acoustics Signals" [2], les chercheurs ont essayé de mettre en place un système qui a pour but de reproduire l'environnement de la victime afin de pouvoir créer des données d'apprentissage sur un environnement quasi similaire à celui de la cible. Ils imaginent le scénario suivant : un attaquant place deux microphones à proximité de l'ordinateur de sa victime et va enregistrer le bruit qu'elle aura fait en appuyant sur les touches. Cependant il ne connaît ni le type de clavier ni la position exacte des micros par rapport à ce dernier. L'attaquant va donc devoir dans un premier temps trouver quel type/modèle de clavier a été utilisé, ensuite il devra identifier les "grosses touches" telle que la touche espace ou la touche entrée. Il devra ensuite faire une estimation de la position des micros en triangularisant le son de ces "grosses touches". Il pourra après cela recréer un environnement semblable à celui de sa cible. La première chose que les chercheurs vont faire est de normaliser les données qu'ils

ont obtenu afin qu'elles se situent entre -1 et 1. On choisit ensuite une valeur de seuil, ici cette valeur est définie à 0.2. Chaque fois que la valeur du signal normalisé dépassera cette valeur, on dira que c'est le bruit d'une touche pressée . Une fois que l'on a trouvé une touche , on prend les 180ms qui la suivent que l'on va segmenter en pleins de blocs de 10ms séparés les uns des autres par une durée de 2.5ms. On va ensuite appliquer la fonction de "Hann" sur chacun de ces blocs de 10 ms. Une fois cette étape terminée, on va pouvoir passer à l'étape de "l'estimation de l'environnement". On va donc devoir deviner quel type / modèle de clavier a été utilisé, pour cela on va transformer nos données pour en récupérer les MFCC's puis on va classifier le résultat obtenu avec un algorithme "SVM" pour savoir à quelle catégorie de clavier appartient le clavier qu'on cherche à identifier. Ensuite on va devoir récupérer les MFCC's correspondant au bruit des touches que nous avons identifié précédemment afin de les comparer avec les MFCC's des "grosses touches" que nous avons déjà enregistré au préalable sur différents types de claviers. Pour cela on devra encore une fois utiliser un algorithme "SVM". Une fois que l'on aura identifié les "grosses touches", on devra essayer de déterminer la position des deux micros par rapport au clavier. Pour cela on utilisera une méthode de "TdoA" et l'algorithme "CMA-ES", qui en regroupant les informations de chaque grosse touche me donnera la position des deux micros. Maintenant que l'on connaît le type de clavier et la position des micros par rapport à ce dernier, on va pouvoir reproduire un environnement presque identique à l'environnement de la victime. Dans cet environnement, on va pouvoir se créer un jeu de données que l'on pourra ensuite comparer aux données obtenues lors de la phase de normalisation. On aura donc pour chaque lettre détectée lors de la phase de normalisation, une lettre désignée lui correspondant et celle ci calculée grâce à un algorithme de "PSD". On va considérer que chaque fois que la touche espace ou entrée est détectée, on passera d'un mot à un autre. Ensuite grâce à l'aide d'un dictionnaire contenant 1500 mots (les plus utilisés) on va pouvoir faire un top 5 des mots les plus probables pour chaque mot, en fonction de la probabilité qu'une lettre soit la bonne. Dans le cadre de l'article la base de données utilisée est constituée de 9 claviers différents, sur lesquels auront tapés 9 volontaires (7 hommes et 2 femmes). Afin d'enregistrer le son, on aura utilisé un HUAWEI mate20 avec 2 micros intégrés. En ce qui concerne la reconnaissance des grosses touches du clavier, on obtient une précision d'environ 90%. En ce qui concerne la position des micros, on peut voir qu'en ayant obtenu la position de 6 grosses touches, on peut réussir à obtenir une position pour les microphones qui sera à 3cm près de la position originale. Les chercheurs ont aussi découvert qu'avec un décalage de 3cm sur la position du microphone, on ne perdait que 8% d'efficacité sur l'identification des lettres. On peut voir que pour le même clavier et des données d'apprentissages n'appartenant pas à la victime, on peut voir que la probabilité que le mot recherché fasse parti du top 5 des mots les plus probables est de 90%, si on ne garde que les 3 mots les plus probables on passe à 84% de chance et si on ne garde que le mot le plus probable on passe à 70%. Si par contre on ne possède pas le même clavier que la victime, on peut voir que dans le top 5 des mots les plus probables on est à 72% de chance que le mot mystère en fasse partie, si on passe au top 3 on diminue à 52% et si on ne garde qu'un seul mot on passe à 33%. Comme dans la plupart de ces articles, on se retrouve encore une fois très dépendant de l'environnement dans lequel on enregistre les données, car si un bruit autre qu'une touche se fait entendre durant l'enregistrement, l'algorithme l'interprétera comme le son d'une touche. En ce qui concerne les résultats, ils sont plutôt encourageants quand on utilise le même modèle de clavier, malheureusement si l'on ne possède pas le même modèle, les résultats sont moins spectaculaires.

2.3 Voie à suivre / Chemin choisi

On a pu voir que dans la grande majorité des articles, l'analyse de texte saisi au clavier se déroule toujours dans un lieu très silencieux pour être sûr de ne pas parasiter la phase de détection de touches via les FFT. On a pu voir aussi qu'afin de retrouver le texte saisi, on cherche à chaque fois lettre par lettre à savoir quelle lettre correspond à quel bruit. On a donc décidé d'essayer de prendre le problème sous un autre angle d'attaque, afin de voir si cela pouvait donner de meilleurs résultats. Dans un premier temps lors de la phase de détection de caractères nous avons décidé de partir, non pas comme tout le monde sur la FFT accompagnée d'un seuil, mais plutôt de classifier chaque son grâce à leur MFCC afin de déterminer à quel type de son cela correspond parmi une liste (bruit de touches, parole, souffle, silence, ...). Nous avons aussi choisi de ne pas classifier les touches une à une. Nous avons donc choisi de prendre une autre approche très peu exploitée dans les autres articles. Elle consiste dans un premier temps à essayer d'identifier les espaces qui séparent les mots, pour dans un deuxième temps faire de l'apprentissage mot par mot. Cette approche ne permet cependant pas de deviner des mots de passe aléatoires car elle va comparer les mots mystères à de vrais mots présents dans notre base d'apprentissage.

3 Création d'un corpus de mots

Afin de pouvoir tester nos algorithmes, nous avons dû dans un premier temps définir un corpus de mots. Nous avons donc choisi de partir sur un corpus entièrement en anglais afin de pouvoir parler au plus grand nombre. Mais comment choisir les mots qui composeront ce corpus me direz-vous, et bien c'est plutôt simple. Nous avons dans un premier temps choisi de donner une position "x" et "y" à chacune des touches du clavier, afin de les situer dans l'espace. On peut voir la répartition ci-dessous :

FIGURE 4 – Explication simple du procédé

Nous sommes ensuite allés chercher sur internet quels étaient les mots anglais les plus utilisés dans la vie de tous les jours et en avons récupéré une liste. Pour chacun de ces mots, nous avons changé leurs lettres par la position "x" et "y" qui leur était associée (exemple : cat –> [[2,2][0,0][0,4]]). On a ensuite décidé de calculer pour chaque mot une valeur d'espacement, pour cela nous avons utilisé la formule mathématique suivante :

$$d(A,B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

FIGURE 5 – Explication simple du procédé

On va donc calculer une distance d(A,B), avec "A" une lettre d'un mot et "B" la lettre qui la suit dans le mot, on obtiendra donc pour un mot de taille "n", "n-1" résultats de distance d(A,B) que l'on va ensuite additionner les uns avec les autres. (Dans l'exemple de tout à l'heure "cat", on aura d(c,a) = $\sqrt{8}$ et d(a,t) = 4. On obtiendra alors une distance totale d'environs 6,8 ($\sqrt{8}$+4≈6,8)). Après avoir calculé cette distance pour l'intégralité des mots que nous avions sélectionné, nous avons décidé de ne garder que les mots qui auront une distance totale d'au moins deux fois leur taille. Tout cela dans le but de ne pas avoir de regroupements de lettres, ce qui pourrait poser problème à notre algorithme (Dans le cadre de notre exemple "cat", on a comme valeur de distance 6,8. Le mot "cat" est un mot de 3 caractères, on obtient donc comme valeur à dépasser 6, comme 6,8 > 6 on peut dire que "cat" est un mot valide). On a aussi choisi de retirer tous les mots contenant uniquement une et deux lettres, en effet ces mots nous semblaient un peu trop courts. On a gardé 100 mots parmi la liste de tous les mots restants, ces 100 mots forment ainsi le corpus sur lequel nous avons travaillé. On peut voir ce corpus de mots ci-dessous (ainsi que sa répartition en nombre de lettres dans l'annexe 18) :

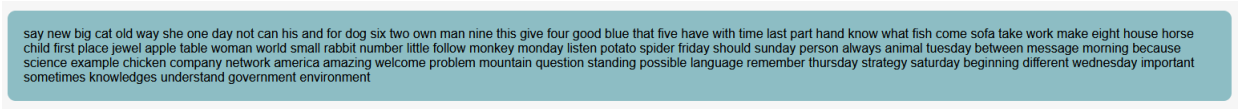

FIGURE 6 – Explication simple du procédé

Je soulève cependant un petit problème dans notre approche qui est que nous employons un clavier "azerty" pour la collecte des données. Cela signifie que si on utilise un clavier "qwerty" nous n'obtiendrons pas du tout les mêmes résultats et l'algorithme ne reconnaîtra aucun mot car les lettres seront à des positions différentes. Le fait d'utiliser un corpus anglais est donc à la fois une bonne chose pour l'accessibilité à ce projet, mais est aussi un peu contraignant car ça ne le rend pas réutilisable pour autant si notre clavier n'est pas un "azerty", ce qui est un comble pour un corpus entièrement en anglais.

La base de données a été réalisée grâce à un ordinateur portable de la marque "DELL" sous windows 10. Cet ordinateur est muni de deux micros intégrés sur le haut de l'écran, tous deux étant séparés par une webcam. Chaque individu de notre base de données s'est alors enregistré en train de taper la liste des mots à deux reprises. Au final nous avons eu 16 personnes différentes qui se sont enrôlés dans la base de données. On obtient donc 16 fichiers audio qui durent chacun entre 7 et 15 minutes en fonction de la vitesse de frappe au clavier de la personne enregistrée. Ce qui nous fait un total de 32 occurrences de chaque mot dans la base de données pour l'apprentissage. Soit un total de 3200 fichiers audio différents une fois ceux-ci segmentés. Tous ces fichiers audio ont été réalisés dans un endroit le plus silencieux possible.

4 Identification d'un individu

4.1 Procédé

Pour ce qui est de l'identification d'un individu, nous allons utiliser les grands fichiers audio dont j'ai parlé précédemment (ceux qui durent entre 7 et 15 minutes). Dans un premier temps, nous avons segmenté ces grands fichiers en pleins de petits fichiers de 10sec, puis dans un deuxième temps, nous avons segmenté ces grands fichiers en pleins de petits fichiers de 5 sec. Tout cela dans le but de savoir avec quelle durée de fichier audio il est le plus facile et le plus pratique d'identifier un individu. Une fois cette segmentation réalisée, on va calculer pour chaque individu un fichier "SVM" et ce autant de fois qu'il y a d'autres individus dans la base de données (exemple : Une base de données composée de 3 individus A, B et C. On va alors calculer 3 fichiers "SVM", respectivement SVM_AB, SVM_AC et SVM_BC). Pour cela on va tout simplement utiliser une librairie python du nom de pyAudioAnalysis qui nous permet de calculer ces fichiers automatiquement en lui donnant en entrée des données audio. On obtient alors dans notre cas 120 fichiers "SVM". On va ensuite prendre un fichier audio d'une personne pour essayer de l'identifier (On appellera par la suite ce fichier audio : "mystère"). Afin de savoir à quelle personne le fichier audio correspond, on va comparer notre fichier audio "mystère" et le comparer avec chacun des fichiers "SVM" que nous avons créé précédemment. On obtiendra alors pour chaque individu de la base de données, une probabilité d'être l'individu "mystère". Pour être plus clair dans mes explications, reprenons notre exemple de tout à l'heure. On avait 3 fichiers "SVM" (SVM_AB, SVM_AC et SVM_BC), prenons un fichier "mystère" que l'on nommera M. On va calculer avec SVM_AB la probabilité que M soit A et que M soit B, sur SVM_AC la probabilité que M soit A et que M soit C et sur SVM_BC la probabilité que M soit B et que M soit C. On va ensuite faire la moyenne de toutes les probabilités que M soit A et on obtiendra une valeur de probabilité de M = A, on fait ensuite la même chose pour tous les autres individus (ici B et C). Une fois cela fini, on regarde pour quel individu la valeur finale est la plus grande, on dira alors que M est cette personne.

4.2 Résultats

On va alors expérimenter le procédé expliqué ci-dessus sur deux ensembles de données. Dans un premier temps, on va essayer sur des échantillons de 10 sec de bruit de touches. On va donc avoir 40 fichiers audio de 10 sec par personne que l'on va utiliser pendant l'apprentissage et on va avoir deux fichiers audio "mystère" par personne pour voir si notre programme réussit à les identifier. On peut voir les résultats dans le tableau suivant :

Personne à identifier	Resultat le plus probable	Deuxieme resultat le plus probable
Cedric	Cedric	Lucinda
Cedric	Cedric	Laurine
Celine	Celine	Julie
Celine	Celine	Laurine
Emmanuel	Emmanuel	Lucille
Emmanuel	Emmanuel	Lucille
Florence	Vincent	Cedric
Florence	Florence	Laurine
Florent	Florent	Lucille
Florent	Justine	Emanuelle
Julie	Julie	Emanuelle
Julie	Julie	Emanuelle
Justine	Justine	Emanuelle
Justine	Justine	Noe
Laurine	Laurine	Celine
Laurine	Laurine	Lucinda
Lucille	Emanuelle	Lucille
Lucille	Lucille	Emanuelle
Lucinda	Lucinda	Laurine
Lucinda	Lucinda	Laurine
Max	Max	Laurine
Max	Max	Laurine
Noe	Noe	Laurine
Noe	Noe	Justine
Pierre-Louis	Pierre-Louis	Steve
Pierre-Louis	Pierre-Louis	Steve
Simon	Simon	Noe
Simon	Simon	Pierre-Louis
Steve	Steve	Pierre-Louis
Steve	Steve	Pierre-Louis
Vincent	Vincent	Laurine
Vincent	Vincent	Laurine
Bien reconnu :	90.63%	3.13%
	93.75%	

FIGURE 7 – Résultats avec des données audio de 10 sec

Dans la colonne de gauche, on peut voir qui a réalisé le fichier audio ("mystère") que l'on va essayer d'identifier. Dans la colonne centrale, on voit qui a été identifié par l'algorithme avec la plus grande probabilité. Dans la colonne de droite, on voit qui est la deuxième personne la plus probable à être identifiée. Dans le cas ou nous utilisons des données d'une durée de 10 sec, on peut voir que l'algorithme réussit à reconnaître environ 90% des personnes sans faire de faute. Si on regarde dans la colonne de droite le résultat augmente encore. Le résultat est déjà plutôt concluant, mais on va essayer de l'améliorer et pour cela on va maintenant essayer le même processus mais cette fois avec des fichiers audio de 5 sec. On aura cette fois pour l'apprentissage un total de 80 fichiers audio par personne et encore une fois 2 fichiers audio ("mystère") que l'on va essayer d'identifier par personne. On peut voir les résultats obtenus dans le tableau ci-dessous :

Personne à identifier	Resultat le plus probable	Deuxieme resultat le plus probable
Cedric	Cedric	Julie
Cedric	Cedric	Vincent
Celine	Celine	Julie
Celine	Celine	Julie
Emmanuel	Emmanuel	Lucille
Emmanuel	Emmanuel	Lucille
Florence	Florence	Lucinda
Florence	Florence	Cedric
Florent	Lucille	Florent
Florent	Florent	Lucille
Julie	Julie	Emmanuel
Julie	Julie	Emmanuel
Justine	Justine	Cedric
Justine	Justine	Lucinda
Laurine	Laurine	Pierre-Louis
Laurine	Laurine	Vincent
Lucille	Lucille	Emmanuel
Lucille	Lucille	Vincent
Lucinda	Lucinda	Laurine
Lucinda	Lucinda	Laurine
Max	Max	Laurine
Max	Max	Laurine
Noe	Noe	Pierre-Louis
Noe	Noe	Justine
Pierre-Louis	Pierre-Louis	Laurine
Pierre-Louis	Pierre-Louis	Noe
Simon	Simon	Steve
Simon	Simon	Pierre-Louis
Steve	Steve	Pierre-Louis
Steve	Steve	Pierre-Louis
Vincent	Vincent	Florence
Vincent	Vincent	Florence
Bien reconnu :	96.88%	3.13%
	100.00%	

FIGURE 8 – Résultats avec des données audio de 5 sec

Cette fois-ci, les résultats sont encore plus concluants. En effet on peut voir que le pourcentage de personnes bien identifiées est cette fois d'environ 96%. Si, pour chaque fichier audio, on regarde en plus de cela la deuxième personne la plus probable, on peut voir que l'on atteint les 100% d'identification. Les résultats obtenus pour des fichiers avec une durée de 5 sec sont donc plutôt encourageants. Cependant la base de données utilisée lors de cet exemple n'est composée que de 16 personnes. Il faudrait donc voir sur une base de données de taille supérieure si les résultats restent toujours aussi encourageants, et ce surtout avec des fichiers audio de 5sec.

5 Détection du texte saisi : "Détection mot par mot"

5.1 Données diverses

Une fois que l'on a récupéré toutes les données nécessaires de bruit de touches de clavier, on va devoir créer d'autres types de données. Ces données vont être des données audio de divers types comme des échantillons de voix humaine, des bruits de porte qui claque, du silence, le bruit du vent sur un micro, un bruit sourd d'un objet qui tombe par terre, des bruits de touches... Toutes ces données vont nous servir par la suite dans la phase de reconnaissance du texte saisi au clavier, afin de différencier le bruit d'une touche de divers autres bruits qui pourraient survenir dans la salle lors de la saisie de données. On va ensuite séparer tout ça en plusieurs dossiers, chacun contenant qu'un seul et unique type de données.

5.2 Explication du fonctionnement

Avant de commencer à expliquer comment fonctionne le programme, on va séparer les données ainsi récoltées pour former deux ensembles distincts, d'un coté avec une partie des données, on va créer un ensemble d'apprentissage qui nous servira durant la phase de préparation, de l'autre un ensemble de tests avec l'autre partie des données afin de tester le bon fonctionnement de notre programme.

5.2.1 Préparation

Une fois toutes les données récoltées durant la phase d'enrôlement, on se retrouve avec une multitude de fichiers audio qui contiennent chacun l'audio émis par le clavier d'ordinateur lorsque l'on y tape l'ensemble des mots du corpus. En plus de ces fichiers audio on possède aussi, pour chaque audio, un fichier texte correspondant à leur "keylogger". C'est à dire un fichier texte qui nous dit à quel moment précis la personne a tapé quelle lettre sur le clavier. On peut voir un exemple de ce type de fichier ci-dessous :

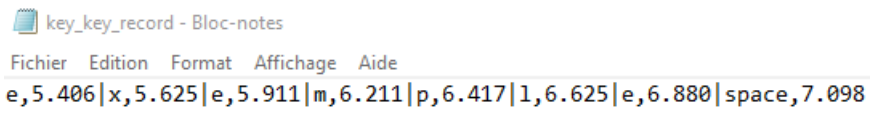

FIGURE 9 – Exemple de fichier de keylogger

Ces fichiers vont nous être très utiles pour pouvoir segmenter efficacement les fichiers audio. On va donc chercher à découper les fichiers audio de plusieurs façons différentes. Dans un premier temps, on va séparer les fichiers audio lettre par lettre, comme on sait à quel instant précis de l'audio se trouve chaque lettre grâce au «keylogger» cela va être facile. On va devoir définir une durée moyenne de son d'une touche sur le clavier, pour l'instant on va définir cette durée comme 100 ms. On va pour chaque lettre créer un nouveau fichier audio (au format «wav») qui commencera 10ms avant que la touche ne soit pressée et qui continuera 90 ms après. On va ensuite classer ces fichiers dans deux catégories, ceux qui correspondent à la touche espace et le reste. Cet ensemble de fichiers nous sera utile pendant la phase de reconnaissance du texte saisi au clavier, pour l'instant laissons la de coté. On va maintenant recommencer à découper les fichiers audio contenant l'intégralité des mots du corpus mais cette fois mot par mot. Pour faire cela, on va dans un premier temps découper le fichier "keylogger" en se basant sur la touche espace, si un espace est pressé alors on termine un mot et on en commence un nouveau. On a ensuite plus qu'à faire comme précédemment, on découpe lettre par lettre (avec une durée de 100 ms à chaque fois) mais cette fois on regroupe les lettres ainsi découpées pour former des mots. On peut voir sur l'exemple ci-dessous à quoi cela ressemble, avec en haut une lettre toute seul et en bas un mot :

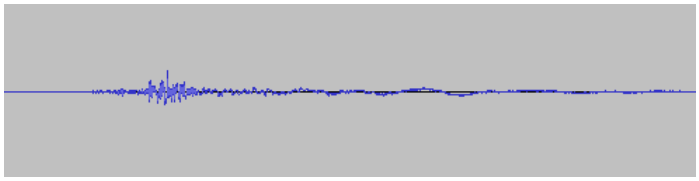

FIGURE 10 – Exemple de fichier audio d'une lettre

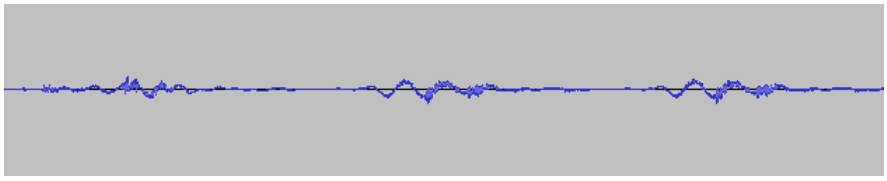

FIGURE 11 – Exemple de fichier audio d'un mot

Une fois que l'on a correctement segmenté l'intégralité de ces mots, on va devoir segmenter les fichiers audio contenant divers bruits (dont j'ai parlé précédemment 5.1) en multiples fichiers audio de 100 ms, afin qu'ils fassent la même taille que le bruit d'une seule et unique lettre. On va ensuite extraire des caractéristiques de ces données. Dans notre cas nous allons extraire leurs MFCC afin de créer un tableau dans lequel pour chacune de ces données on aura leur MFCC.

5.2.2 Reconnaissance sans connaître la position des touches sur le fichier audio

Maintenant que toutes nos données sont prêtes, on va pouvoir rentrer dans le vif du sujet. On va maintenant parler des données de test. Pour segmenter ces données cela ne va pas être aussi simple que tout à l'heure car ici on ne possède plus les fichiers "keylogger". Comment allons-nous savoir où se situent les bruits de touches sur le fichier audio me direz-vous ? Et bien c'est là qu'entre en jeu les diverses données de sons que nous avions extrait précédemment. Nous allons devoir parcourir notre fichier audio par l'intermédiaire d'une fenêtre glissante de taille 100ms, et avec un décalage de 10ms entre chaque fenêtre (tel un petit train de 100 mètres qui s'arrêterait tous les 10 mètres).

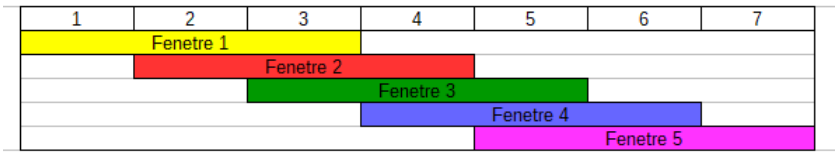

FIGURE 12 – Explication de la fenêtre glissante

On va ensuite classifier chacune de ces fenêtres de 100 ms grâce à un algorithme LogisticRegression, on aura donc pour chaque segment une prédiction de classes parmi les divers bruits possibles que nous avions listé plus tôt (voix humaine, bruits de porte qui claque, silence, bruits de touches, ... [5.1]). Cette classification ne se fait pas avec les données brutes, mais avec les MFCC extraites de ces données. Tant que l'algorithme ne classifie pas le contenu de la fenêtre glissante comme étant le bruit d'une touche de clavier, il continue de s'exécuter normalement. Cependant s'il classifie le contenu de la fenêtre glissante comme étant un bruit de touches, on va garder en mémoire la position (en terme de secondes / millisecondes) de cette touche puis on va faire un bond dans le fichier audio de 100 ms afin d'être sûr de ne pas redétecter la même touche une deuxième fois. Une fois que la fenêtre glissante a fini de parcourir l'intégralité du fichier, on va en extraire toutes les touches du clavier que notre algorithme de classification avait détecté, afin d'obtenir plusieurs fichiers contenant chacun le bruit d'une touche.

La deuxième étape va consister encore une fois à faire de la classification, cette fois nous n'aurons que deux classes, les touches espace d'un coté et le reste des touches de l'autre. Pour cela nous allons

comme précédemment extraire les MFCC des différents fichiers audio que nous avons obtenu à l'étape précédente et les MFCC des fichiers d'apprentissage des touches espace et des autres touches. Cette fois afin de classifier les données nous allons utiliser un algorithme de RidgeClassifier. Cette étape est l'une des plus importantes, en effet c'est elle qui déterminera la taille des mots. Si les touches espace sont mal détectées, cela peut devenir problématique (exemple : la phrase "hello everybody" est composée de 5 caractères suivit d'un espace puis de nouveau, 9 caractères. Si l'espace n'est pas détecté, l'algorithme pensera que c'est un mot de 15 caractères). On obtiendra donc comme résultat un tableau contenant la taille des mots.

Une fois les espaces correctement identifiés on va pouvoir reformer les mots en concaténant les différents fichiers de bruits de touches que nous avons créé précédemment [5.2.2]. On obtiendra alors plusieurs fichiers audio contenant chacun un mot. On peut voir un exemple de ce qu'il se passe ci-dessous :

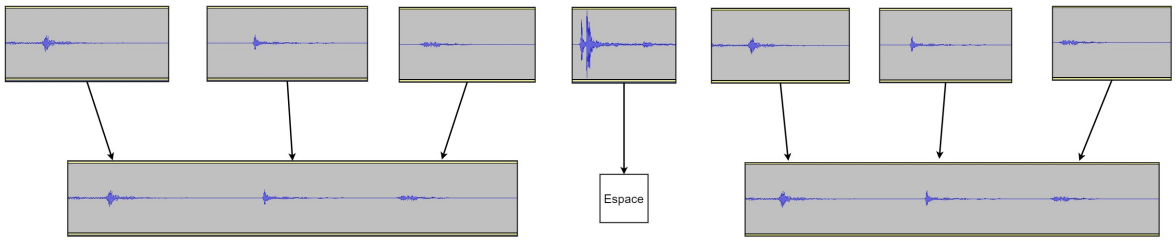

FIGURE 13 – Exemple de concaténation de fichier

On va maintenant récupérer les données de la phase de préparation que l'on n'avait jusqu'alors pas exploité. Dans un premier temps, nous allons extraire les MFCC de l'intégralité de ces fichiers que nous allons stoker sous forme de tableau (voir ci-dessous) :

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	numero	nom	mfcc0	mfcc1	mfcc2	mfcc3	mfcc4	mfcc5	mfcc6	mfcc7	mfcc8	mfcc9	mfcc10	mfcc11
2	0	always	-590,529	-563,92	-560,529	-527,472	-520,877	-541,809	-502,038	-410,588	-354,349	-372,263	-435,409	-381,199
3	1	always	-540,412	-498,308	-511,674	-501,708	-511,121	-523,833	-502,536	-476,717	-430,969	-442,014	-502,705	-391,38
4	2	always	-440,166	-381,669	-394,073	-367,455	-303,92	-326,81	-415,163	-395,499	-281,131	-279,299	-374,355	-424,226
5	3	always	-415,577	-375,311	-399,924	-438,907	-366,321	-323,754	-358,192	-425,839	-428,108	-389,443	-380,716	-421,834
6	4	always	-478,881	-393,619	-398,199	-450,118	-468,231	-436,647	-438,73	-477,699	-470,585	-417,96	-389,557	-412,026
7	5	always	-550,492	-421,503	-390,598	-419,367	-446,288	-458,096	-479,043	-510,873	-504,05	-459,065	-403,531	-410,377
8	6	always	-398,293	-327,312	-356,283	-376,34	-307,885	-269,045	-290,341	-376,385	-436,459	-402,532	-420,833	-465,568
9	7	always	-366,774	-324,754	-371,444	-389,696	-253,604	-206,498	-256,439	-360,225	-397,598	-377,846	-415,73	-487,595
10	8	always	-572,803	-477,483	-484,851	-497,307	-442,217	-446,955	-506,798	-589,345	-598,291	-507,036	-469,537	-498,397
11	9	always	-551,951	-491,352	-527,897	-605,423	-608,711	-512,685	-498,001	-557,122	-544,016	-479,658	-468,1	-467,163
12	10	always	-622,129	-547,17	-459,188	-449,395	-488,02	-447,67	-358,042	-359,858	-449,219	-515,452	-439,459	-371,351
13	11	always	-444,931	-340,379	-355,503	-396,166	-329,317	-346,718	-397,614	-421,965	-390,579	-255,539	-200,982	-248,43
14	12	always	-435,613	-426,872	-482,529	-517,858	-411,187	-382,462	-429,395	-481,54	-358,651	-315,596	-373,388	-475,891
15	13	always	-646,884	-592,74	-576,946	-572,234	-566,43	-564,093	-513,453	-509,916	-536,837	-549,892	-550,98	-536,364
16	14	always	-411,519	-368,822	-399,511	-404,745	-343,636	-367,727	-468,606	-514,291	-457,607	-409,613	-422,413	-487,736

FIGURE 14 – Exemple de tableau Mots/MFCC

Ensuite on va extraire les MFCC des mots que nous avons reformé juste avant. Puis nous utiliserons l'algorithme "ExtraTrees" qui est un algorithme de classification. Cet algorithme aura pour but de donner un score de similarité entre un mot mystère et chacun des mots présents dans le tableau (que l'on peut voir juste au dessus). On retournera ensuite les "n" mots qui ont la plus grande probabilité d'être le mot mystère et ce avec des mots de longueurs identiques au mot mystère ou avec une lettre de plus. On peut en voir un exemple ci-dessous :

FIGURE 15 – Exemple de résultat

Sur cet exemple, le mot "0" est probablement le mot "because", "network", "possible" ou "language" (rangé par ordre de probabilité).

5.3 Résultats

Pour évaluer la performance de notre outil, nous allons imaginer deux cas de figure distincts. Dans le premier cas de figure, on va avoir l'information d'à quel moment une touche a été pressée, sans savoir la touche en question. Dans le deuxième cas, on va laisser notre outil segmenter les mots, ce qui nous donnera probablement des mots dans lesquels il manquera certaines lettres ou alors il y en aura trop. Pour ce qui est de la méthode 1, nous avons testé de reconnaître du texte qui a été écrit par 3 personnes présentes dans notre base de données. Pour cela nous avons fait un top 3 des mots avec la plus grande probabilité d'être le mot mystère. On peut voir les résultats ci-dessous :

Individu n°1

Veritable Mot:	Prediction 1 :	Prediction 2 :	Prediction 3 :	Mots reconnus :
rabbit	animal	rabbit	always	70.00%
problem	company	problem	amazing	
friday	listen	potato	monday	
should	listen	should	number	
morning	morning	chicken	pamerica	
woman	world	woman	place	
good	take	good	five	
blue	know	good	last	
because	because	between	tuesday	
chicken	science	because	between	

Individu n°2

Veritable Mot:	Prediction 1 :	Prediction 2 :	Prediction 3 :	Mots reconnus :
say	six	dog	big	80.00%
eight	eight	jewel	woman	
between	message	between	because	
strategy	thursday	strategy	question	
cat	big	can	she	
make	take	make	hand	
tuesday	tuesday	bacause	message	
thursday	thursday	language	standing	
environment	environment	-	-	
sometimes	sometimes	important	different	

Individu n°3

Veritable Mot:	Prediction 1 :	Prediction 2 :	Prediction 3 :	Mots reconnus :
say	way	not	dog	83.33%
dog	big	dog	man	
take	take	make	good	
apple	apple	world	eight	
follow	follow	monkey	sunday	
should	potato	animal	always	
tuesday	network	problem	tuesday	
chicken	chicken	tuesday	morning	
amazing	amazing	science	morning	
language	language	remember	mountain	
beginning	beginning	important	wednesday	
sometimes	sometimes	beginnig	diferrent	

FIGURE 16 – Résultats méthode n°1

On peut déjà faire plusieurs remarques par rapport aux résultats que nous avons. Dans un premier temps, on peut voir que sur l'échantillon des personnes ici présentes, on obtient un ratio d'environ 75% de mots bien retrouvés, ce qui n'est pas mauvais. On peut aussi remarquer que si on ne garde que les deux premières prédictions, on a un résultat quasi identique donc il n'est pas forcément très pertinent de garder la prédiction n°3. On peut aussi remarquer que l'algorithme a plus de mal à retrouver les mots de petite taille (surtout ceux de 3 caractères) comparé aux plus grands mots. Cela peut être dû au fait que les mots de petite taille sont trop courts donc ils se ressemblent un peu tous.

Pour ce qui est de la méthode numéro 2, qui consistait, je le rappelle, à laisser notre outil segmenter les mots tout seul. On peut voir les résultats dans le tableau 17 ci-dessous :

Individu n°1

Veritable Mot:	Prediction 1 :	Prediction 2 :	Prediction 3 :	Prediction 4 :
say dog take apple follow should tuesday chicken amazing language beginning sometimes	day	say	come	have
	man	dog	hand	nine
	take	blue	place	horse
	-	-	-	-
	apple	small	person	rabbit
	-	-	-	-
	make	what	place	small
	-	-	-	-
	house	eight	monkey	rabbit
	thursday	saturday	beginning	different
	chicken	between	standing	remember
	morning	america	mountain	possible
	-	-	and	new
	always	person	problem	science
	beginning	important	understand	knowledges
	possible	question	sometimes	important
	-	-	-	-

Individu n°2

Veritable Mot:	Prediction 1 :	Prediction 2 :	Prediction 3 :	Prediction 4 :
say eight between strategy cat make tuesday thursday environment sometimes	eight	place	spider	friday
	number	listen	network	morning
	-	-	-	-
	-	-	-	-
	always	sunday	science	example
	different	sometimes	knowledges	understand
	what	work	woman	first
	say	man	last	five
	-	-	-	-
	tuesday	science	saturday	strategy
	dog	say	have	hand
	horse	house	follow	always
	-	-	-	-
	morning	america	language	mountain
	-	-	-	-
	-	-	-	-

Individu n°3

Veritable Mot:	Prediction 1 :	Prediction 2 :	Prediction 3 :	Prediction 4 :
rabbit problem friday should morno morning woman good blue because chicken	problem	between	thursday	language
	potato	rabbit	network	amazing
	blue	nine	apple	horse
	-	-	own	say
	place	child	always	monday
	give	make	house	place
	give	know	place	first
	animal	monkey	message	science
	good	four	world	apple
	can	she	have	sofa
	person	always	chicken	problem
	company	america	language	thursday

FIGURE 17 – Résultats méthode n°2

Dans la colonne de gauche de notre tableau, on peut voir ce qui a été tapé par l'utilisateur. Les mots coloriés en vert représentent les mots qui ont réussis à être bien identifiés. On voit que la méthode n°2 marche beaucoup moins bien que la méthode n°1. Cela est dû à l'étape de détection des caractères. En effet l'algorithme détecte beaucoup plus de caractères qu'il n'y en a réellement dans le fichier audio. Une des raisons de ce trop grand nombre de caractères détectés par rapport aux autres articles que nous avons vu dans l'état de l'art est probablement dû au fait que lors de notre phase d'enrôlement, nous avons laissé les gens taper comme bon leur semblaient. On obtient des fichiers audio où les personnes commencent à appuyer sur une autre touche avant même d'avoir relâché l'ancienne, ce qui n'était pas forcément le cas dans les autres articles que nous avons vu précédemment. Dans notre cas, l'individu n°1 est le seul cobaye qui appui sur les touches les unes après les autres, sans chevauchement de touches, ce qui explique pourquoi il a un meilleur ratio de mots identifiés.

6 Conclusion

Pour conclure, ce stage m'a permis d'enfin mettre en pratique un grand nombre de connaissances que j'avais acquis tout au long de mon cursus universitaire. Il m'a aussi permis d'apprendre à quoi ressemble le monde du travail dans le domaine de la recherche qui était jusqu'alors inconnu pour moi. Le sujet sur lequel j'ai travaillé tout au long de mon stage a aussi été une découverte inattendue. Avant de commencer ce stage, je n'aurais jamais soupçonné qu'il soit possible d'avoir un aussi bon taux de réussite pour identifier un individu ou encore deviner le contenu tapé au clavier, et ce juste avec le son qu'un individu peut faire en appuyant simplement sur les touches. Je pense que ce sujet a de belles perspectives d'avenir, et dans notre cas, la principale chose à améliorer serait de réussir à mettre en place un système de détection des lettres dans un fichier audio avec une plus grande précision que celui utilisé. En effet celui que j'ai présenté dans ce rapport ne fonctionne pas très bien, mais l'idée derrière reste cependant, à mon sens, une bonne piste à suivre pour pallier au problème du bruit ambiant lors de la prise d'informations.

6.1 Annexe

TABLE 1: Tableau récapitulatif des méthodes utilisées

Référence	Approche	Algorithme	Dataset	Résultat
Zhuang et al. (2009)[5]	Segmentation lettre par lettre / Post-traitement TAL	MFCC / Chaîne de Markov / FFT	4 datasets privés avec environ 20 minutes d'audio chacun (Avec et sans bruit alentours)	Sans bruit alentours environ 70% des mots identifiés / Avec bruit alentours environ 55% des mots identifiés
Slater et al. (2019) [4]	Segmentation lettre par lettre / Post-traitement TAL	FFT/ CNN	Dataset privé avec 8h d'audio (texte + code informatique)	CER = 36% / meilleur performance pour retrouver du code que un texte en anglais
Compagno et al. (2017) [3]	Segmentation lettre par lettre	FFT / MFCC / KNN / Logistic Regression	Dataset privé de 5 utilisateurs tapant chacun sur 6 claviers différents	Si même clavier et utilisateur 97% de bonnes lettres / Si même clavier mais utilisateur différent 65% de bonnes lettres / Si ni l'un ni l'autre 60% de bonnes lettres
Bai et al. (2021) [2]	Segmentation lettre par lettre / Post-traitement TAL	MFCC / SVM / TDoA / PSD / CMA-ES	Dataset de 9 claviers différents avec 9 volontaires enregistrés grâce à un HUAWEI MATE20	Avec clavier identique 91% de chance de reconnaître un mot dans le top 5 des mots les plus probables / Sans clavier identique 72% de chance de reconnaître un mot dans le top 5 des mots les plus probables
Anand et al. (2004) [1]	Défense contre les attaques	White Noise / Fake Keysroke / Combined Signal	-	SUS Score for white noise = 76 / fake keystroke = 69 / combined signal = 69

Nombre de lettre par mot :	Nombre de mot différent dans la base de données :
3	19
4	22
5	12
6	15
7	14
8	9
9	5
10	3
11	1

FIGURE 18 – Nombre de mots en fonction du nombre de lettres

Références

[1] S Abhishek Anand and Nitesh Saxena. A sound for a sound : Mitigating acoustic side channel attacks on password keystrokes with active sounds. In *International Conference on Financial Cryptography and Data Security*, pages 346–364. Springer, 2016.

[2] Jia-Xuan Bai, Bin Liu, and Luchuan Song. I know your keyboard input : A robust keystroke eavesdropper based-on acoustic signals. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 1239–1247, 2021.

[3] Alberto Compagno, Mauro Conti, Daniele Lain, and Gene Tsudik. Don't skype & type ! acoustic eavesdropping in voice-over-ip. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 703–715, 2017.

[4] David Slater, Scott Novotney, Jessica Moore, Sean Morgan, and Scott Tenaglia. Robust keystroke transcription from the acoustic side-channel. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 776–787, 2019.

[5] Li Zhuang, Feng Zhou, and J Doug Tygar. Keyboard acoustic emanations revisited. *ACM Transactions on Information and System Security (TISSEC)*, 13(1) :1–26, 2009.