Les mots de passe : failles et sécurisation par l'empreinte digitale.

16 juin 2022

Introduction :

La cybersécurité étant un domaine qui me passionne, dans lequel j'aimerais beaucoup travailler, j'ai décidé d'orienter mon TIPE sur les mots de passe. Je me suis fixé pour objectifs d'étudier le fonctionnement des mots de passe, les attaques qui les ciblent, afin de définir, à mon sens, la robustesse, et de créer un système d'évaluation de cette dernière. Par la suite, je veux utiliser l'empreinte digitale pour générer des mots de passe qui soient les plus robustes possibles. L'utilisation de l'empreinte permet de prévenir les vols de mots de passe liés à des failles humaines. En effet, un mot de passe robuste est souvent difficile à mémoriser. Il est souvent noté de manière manuscrite, ou stocké de manière non sécurisée sur ordinateur, ce qui donne lieu à des vols.

1 Etudes des attaques et définition de la robustesse :

Tout d'abord, j'ai étudié différentes attaques. En premier lieu, les attaques par force brute, où tous les mots de passe sont testés, et par dictionnaire, où les mots de passe d'un dictionnaire sont expérimentés [1]. Ces deux premières attaques m'ont permis de mettre en place un premier critère de robustesse, portant sur la longueur et la complexité, c'est-à-dire le nombre de caractères utilisables. Pour moi, la robustesse est la difficulté rencontrée par un pirate lorsqu'il essaye de dérober le mot de passe.

Pour éviter le vol des mots de passe, lorsqu'ils sont stockés ou lorsqu'ils sont transmis, ils sont chiffrés par une fonction de hachage [2]. C'est une fonction qui transforme un message de taille arbitraire en un message de taille fixe, souvent un nombre en hexadécimal, de taille fixe, appelé empreinte numérique. Les fonctions de hachage sont conçues pour être théoriquement à sens unique. C'est-à-dire qu'il est impossible de déterminer le mot de passe original en partant de son empreinte numérique. Ainsi, un moyen pour un pirate de retrouver le mot de passe original serait de stocker tous les mots de passe existants et leur empreinte numérique pour une fonction de hachage donnée. Réalisée de manière naïve, cette méthode n'est pas envisageable. Les tables arc-en-ciel ont ainsi vu le jour. Elles optimisent ce procédé de stockage, en utilisant des fonctions de réduction qui transforment une empreinte numérique en un nouveau mot de passe. En créant des chaînes de mots de passe et d'empreintes numériques, et en ne stockant que les extrémités, il est possible de retrouver toutes les valeurs de la chaîne. C'est pourquoi j'ai programmé une fonction de hachage, MD5 [3], afin de réaliser un prototype de tables arc-en-ciel et d'évaluer le temps nécessaire et la complexité de la création de telles tables.

Pour finir cette première phase, j'ai défini une échelle de notation de la robustesse, en parallèle de celles déjà existantes [1]. J'ai attribué une note sur cinq, représentant la difficulté rencontrée par le pirate pour voler le mot de passe, résumée ci-dessous.

Je considère quatre types de caractères : majuscules, minuscules, chiffres et caractères spéciaux. Si le mot de passe est présent dans un dictionnaire connu, si sa longueur est inférieure ou égale à 8, ou si l'un des quatre types de caractères est absent, la robustesse est nulle. Sinon, des points sont attribués comme suit :

Pour chaque type	Occurrences des caractères du type considéré	
	1	2 ou plus
Points	0.125	0.25

Longueur	9 - 10	11 - 12	13 - 14	15 et plus
Points	1	2	3	4

2 Squelettisation de l'empreinte :

Pour générer des mots de passe robustes et pallier le problème des vols de ces derniers dû à des failles humaines, comme le stockage de manière manuscrite ou informatique non sécurisée, j'ai décidé d'utiliser l'empreinte digitale pour générer des mots de passe robustes, sans les stocker. Pour cela, j'ai utilisé des photos d'empreintes, et j'ai procédé à une squelettisation [5], afin d'extraire le squelette et de faciliter l'étape de détection des points caractéristiques de l'empreinte, appelés minuties.

La squelettisation consiste à extraire le squelette des lignes de crêtes ou des vallées, qui constituent l'empreinte (figure 1). Autrement dit, je cherche à obtenir une image où les lignes significatives ont une épaisseur de un pixel. Pour cela, j'ai programmé plusieurs versions d'un algorithme de squelettisation. J'ai toujours travaillé sur des pixels en nuances de gris, représentés avec la convention RGB, où les trois composantes sont au même niveau. J'utilise uniquement le voisinage trois par trois de chaque pixel. L'objet est représenté par des pixels blancs, et le fond par des pixels noirs.

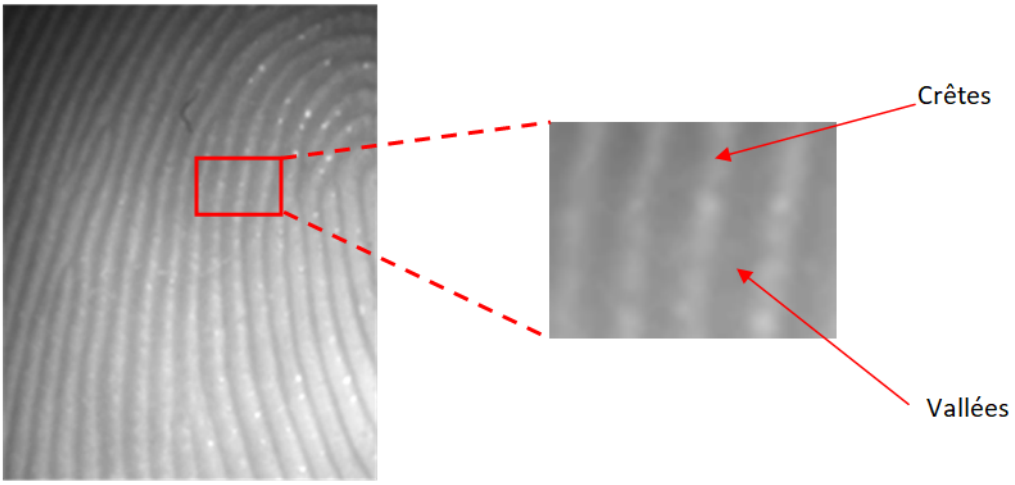


Figure 1

Pour commencer, il m'a fallu définir certaines notions, illustrées en figure 2 : les voisins 4 et 8-connexes, ainsi que les composantes connexes, qui permettent de regrouper les pixels voisins de même couleur en contact. Je distingue les composantes de l'objet et du fond. Si une composante contient un pixel du voisinage 4-connexe, la composante est dite 4-connexe, 8-connexe sinon.

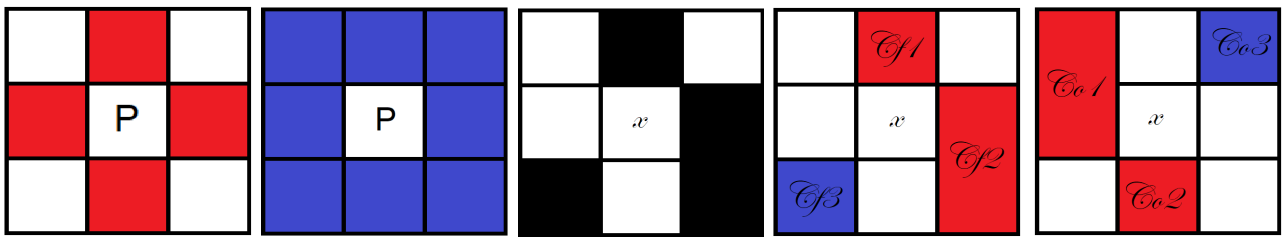
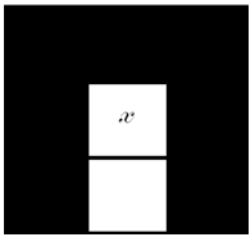
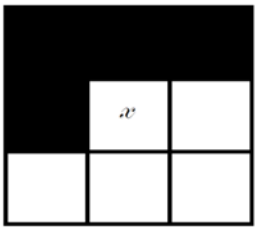


Figure 2 : Voisins 4-connexes , 8-connexes, pixel x, composantes 4-connexes et 8-connexes du fond, composantes 4-connexes et 8-connexes de l'objet.

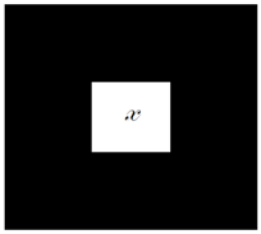
Cela me permet de définir des points particuliers en binaire, en figure 3. Seul le pixel central de l'objet (blanc) nous intéresse dans ces définitions.



Extrémité : Un seul voisin de l'objet dans son voisinage 8-connexe.



Point simple : au moins un voisins 4-connexe du fond, et une seule composante 4-connexe du fond.



Pic : Uniquement des voisins du fond dans le voisinage 8-connexe.

Figure 3

J'ai pu étendre ces définitions en nuances de gris, en définissant les coupes binaires. Pour un pixel en nuances de gris, tous ses voisins de niveau inférieur forme la coupe binaire inférieure, et les autres la coupe binaire supérieure(figure 4).

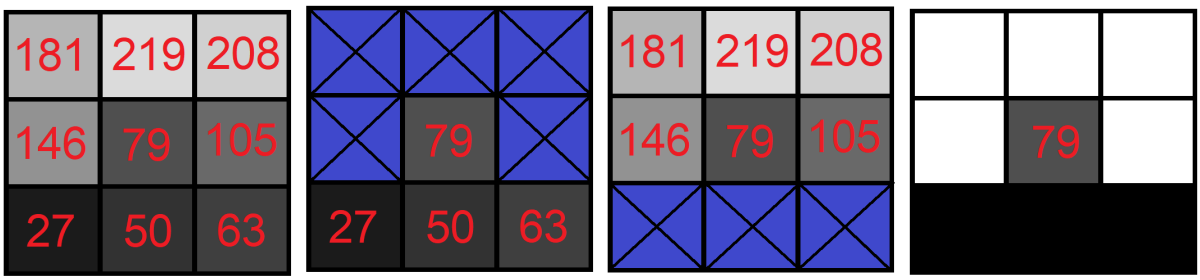


Figure 4 : Pixel, coupe inférieure, coupe supérieure, équivalent binaire.

On peut ainsi transformer le voisinage binaire, et donc étendre les définitions (figure 5).

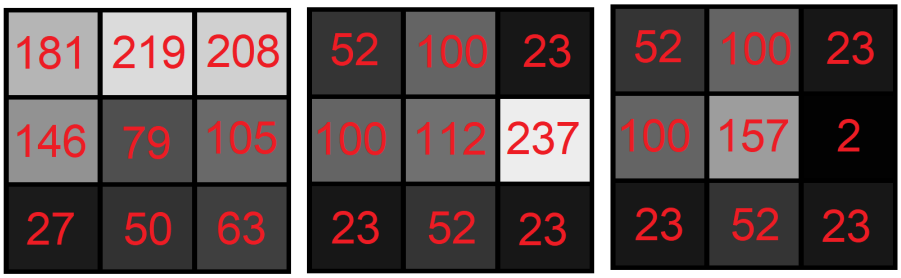


Figure 5 : Point simple, extrémité, pic.

Pour traiter le bruit, j'ai introduit un paramètre λ, qui sera calculé localement, et ajouté une condition aux définitions (formule donné p.50, éq (2.15) et p.54, éq (2.19) de [5]). Si C est une composante connexe et x un pixel, avec $I(x)$ son intensité, je définis la distance de x à C en utilisant les niveaux de gris initiaux comme : $d_0(x, C) = I(x) - min(C)$ (respectivement $d_t(x, C)$ si on utilise les niveaux de gris courants, c'est-à-dire au moment du traitement dans l'algorithme).

Un pixel x est λ-pic (resp. λ-extrémité) s'il est pic (resp. extrémité) et si, en notant C son unique composante 4-connexe du fond, $d_0(x, C) \leq \lambda$. Un pixel x est λ-crête (figure 6) s'il possède au moins deux composantes 4-connexes du fond, et si, en notant $C_1, ..., C_k$ ces k composantes connexes du fond, au moins k – 1 vérifient : $d_t(x, C_i) \leq \lambda, i \in [\![1; k]\!]$

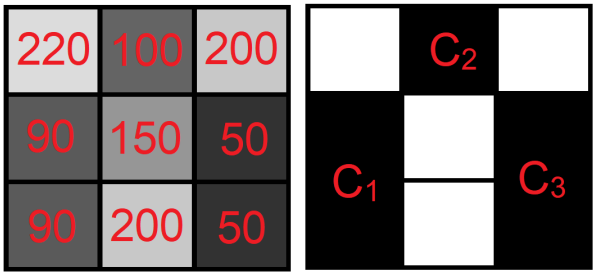


Figure 6 : Pixel 70-crête et ses composantes connexes.

Un pixel est λ-abaissable s'il est λ-crête, λ-pic ou λ-extrémité. Abaisser un pixel signifie mettre ce pixel au maximum des valeurs de ses voisins les plus sombres (de sa coupe binaire inférieure).

La première version de la squelettisation consistait en la binarisation de l'image, et en la mise à 0 des pixels simples et non extrémités [4]. Néanmoins, cette version avait plusieurs inconvénients : perte d'informations sur les nuances de gris, due à la binarisation, influence du parcours, en partie solutionnée par le parcours séquentiel des contours, et absence de traitement du bruit.

La deuxième version utilisait les nuances de gris, et consistait en l'abaissement des pixels simples et non extrémités, en parcourant les pixels du plus sombre au plus clair. Cette version avait l'avantage de diminuer l'influence du parcours, en ne traitant que les pixels par ordre croissant de niveau et uniquement ceux de plus bas niveau. Néanmoins, le bruit n'était pas traité, et l'étape de tri des pixels par niveau augmentait le temps d'exécution, ce qui rendait l'algorithme inutilisable en pratique.

La dernière version [5] est présentée dans la figure 7. Elle utilise une structure de priorité, que l'on appellera « piles de priorité ». Elle est formée de 256 piles, représentant les 256 valeurs possibles pour les pixels en nuances de gris. Chaque pixel, qui vérifie une certaine condition, est empilé dans la pile correspondant à sa valeur. On traite ensuite les pixels de plus bas niveau, en empilant leurs voisins qui vérifient la condition.

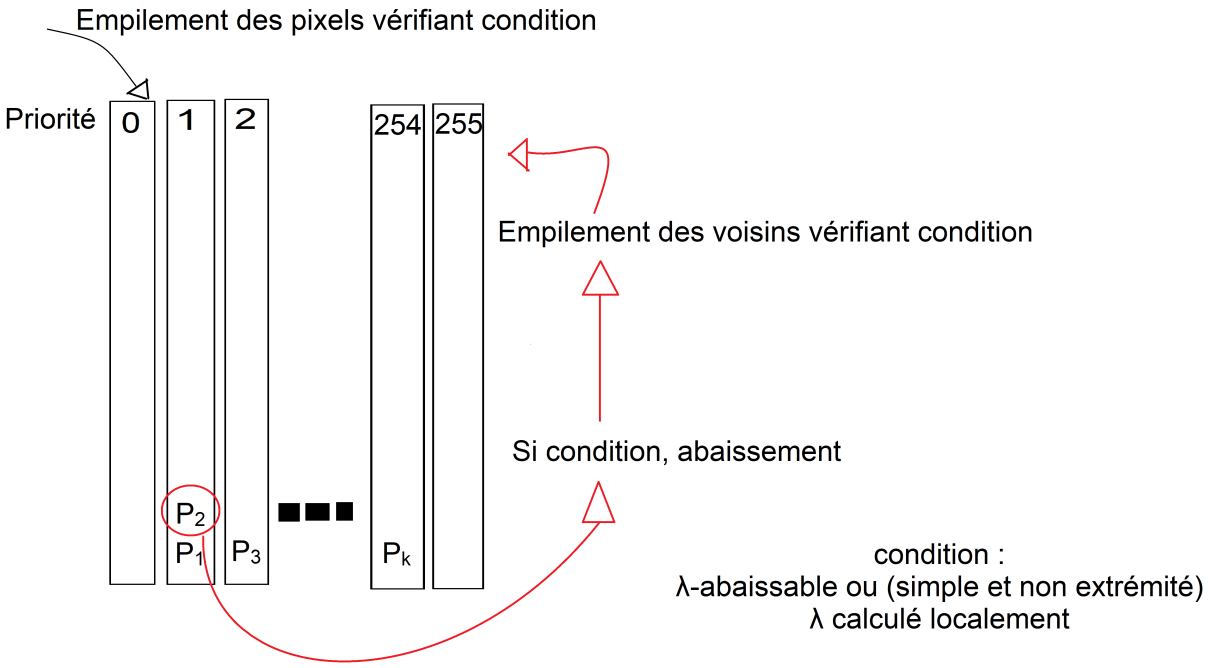


Figure 7 : Algorithme de squelettisation finale.

Un exemple de squelettisation est illustré en figure 8. Comme j'utilise les voisins, les bords de l'image ne sont pas traités.

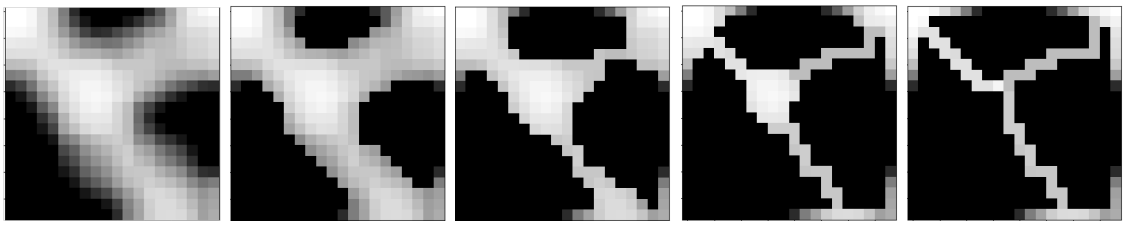


Figure 8 : Etape 0, 100, 200, 300, 332.

3 Extraction de mot de passe :

Une fois le squelette obtenu, la détection des points caractéristiques est plus facile. J'ai décidé de ne considérer que les deux types de minuties suivantes : terminaisons et bifurcations, qui sont prépondérantes (figure 9).

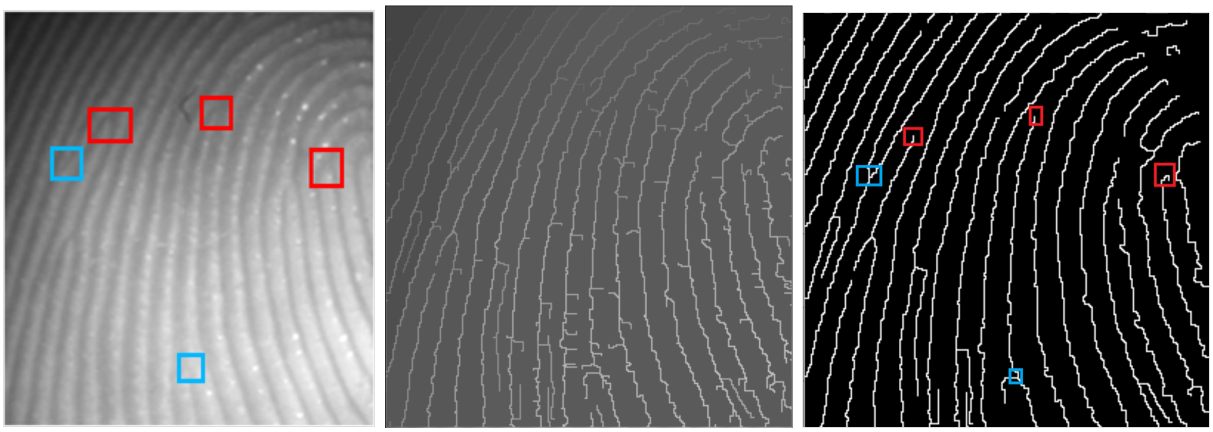


Figure 9 : Empreinte avec des terminaisons et des bifurcations, son squelette brut et nettoyé.

J'effectue un traitement des aberrations en considérant que, partant d'une extrémité, la distance parcourue pour arriver à une bifurcation est négligeable en-dessous de 20 pixels. Je supprime donc, dans ce cas, l'aberration(figure 9). Pour détecter les minuties, je binarise le squelette, en utilisant la méthode d'Otsu [4], qui consiste à détecter un premier plan et un deuxième plan sur une image, et donner un seuil pour la binarisation. Ce seuil est déterminé en calculant, pour chaque valeur des pixels, la variance inter-classe, en considérant la classe des pixels inférieurs et ceux supérieurs à la valeur en cours de traitement (figure 10). Afin que la détection du seuil soit optimale, je découpe l'image en sous blocs de 20 par 20 pixels, afin que les niveaux des deux plans soient plus uniformes, et j'applique cet algorithme sur chaque bloc.

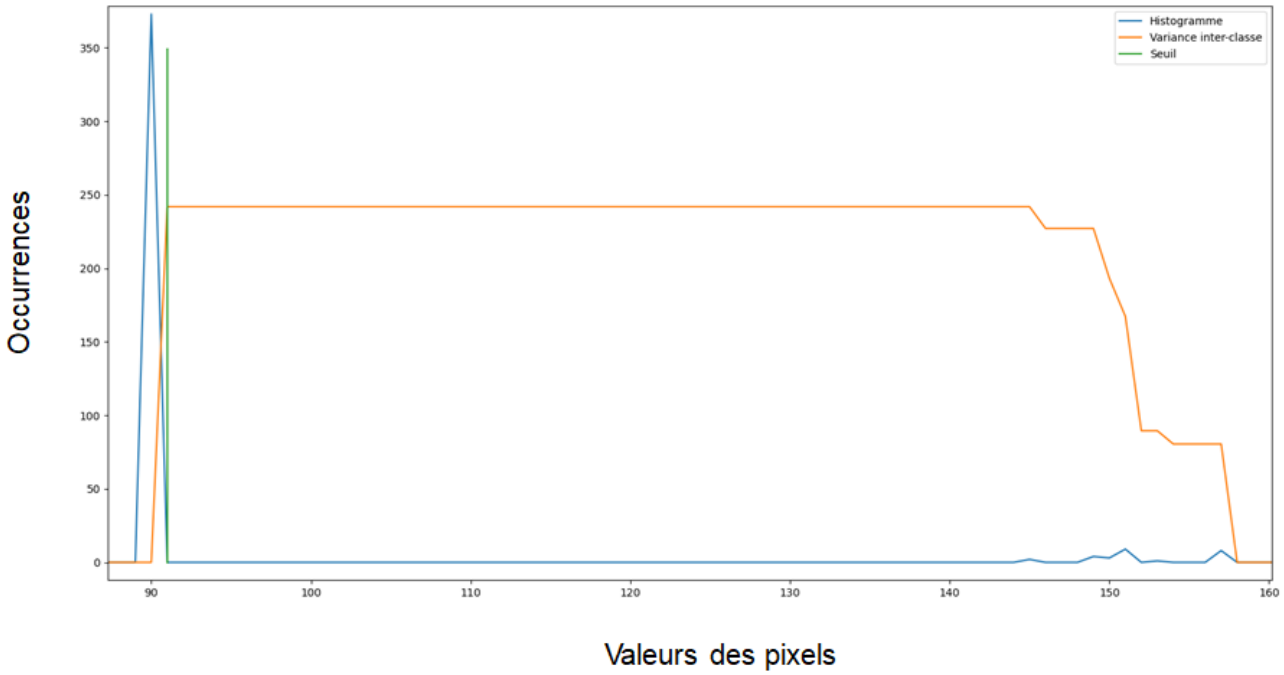


Figure 10 : Histogramme et résultat sur un bloc de 20 par 20 blocs avec un seuil de 91.

Pour chaque point de l'objet, je parcours le voisinage, en relevant le nombre de transitions noir/blanc et blanc/noir. La correspondance entre nombre de transitions et type de minuties est donnée en figure 11.

Types	Point isolé	Terminaison	Ligne	Bifurcation	Bifurcation
Transitions	0	2	4	6	8

Figure 11 : Correspondance entre nombre de transitions et type de minuties (les points isolés et les lignes ne sont pas des minuties).

Je calcule deux barycentres, un pour chaque type de minuties. Je ne garde ensuite que les six plus proches de chaque barycentre, en accord avec la législation française, qui admet que si deux empreintes correspondent en six points caractéristiques, elles proviennent d'un même individu. Je calcule également les angles entre chaque minutie, comme présentée en figure12.

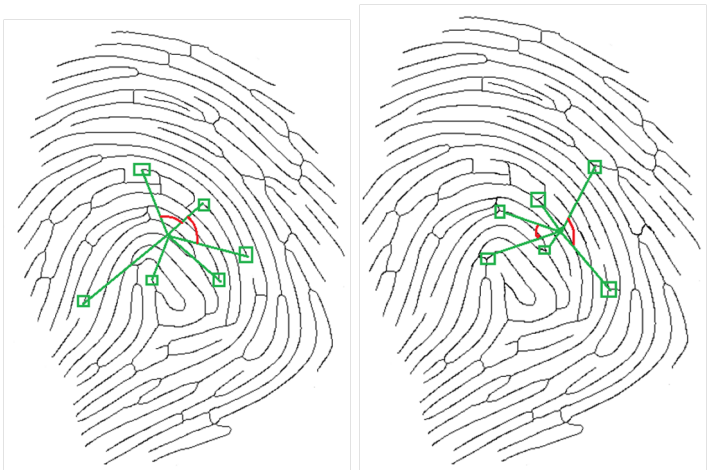


Figure 12 : Les 6 bifurcations et terminaisons que je retiens, avec les distances et les angles.

J'obtiens ainsi deux listes d'angles et deux listes de distances aux barycentres. Je les trie par ordre croissant. Je les concatène toujours de la même manière pour obtenir une chaîne de caractères, en utilisant trois caractères pour chaque élément, avec éventuellement un ou plusieurs zéros ajoutés au début. Je la passe dans la fonction MD5, afin d'obtenir un nombre en hexadécimal sur 32 bits, que je convertis en un mot de passe, de la base 16 vers la base 95, car j'utilise 95 caractères différents.

Tout ce processus a plusieurs buts : tout d'abord, il est invariant par rotation et translation. En effet, si l'utilisateur pose son doigt légèrement différemment, le mot de passe doit être identique. L'utilisation des barycentres et des quatre listes assure cette propriété. Par ailleurs, en considérant que MD5 est injective, le processus de concaténation l'étant par construction, deux mots de passe sont identiques, si et seulement si, ils proviennent d'une même empreinte. Par ailleurs, la propriété de répartition uniforme des fonctions de hachage assure que les mots de passe obtenus aient, avec une très forte probabilité, une longueur suffisante, au vu de la définition de robustesse donnée plus haut. J'évalue et je donne également la robustesse des mots de passe ainsi générés.

4 Application sur une base de données :

Je me suis procuré une base de données de l'institut Polytechnique de Hong Kong contenant des photos d'empreintes digitales d'environ deux cents personnes. J'ai pu tester mes programmes. Néanmoins, lorsque je teste deux photos du même doigt, les résultats ne coïncident pas, car une moindre différence entre les deux images provoque une modification non négligeable du squelette (figure 13).

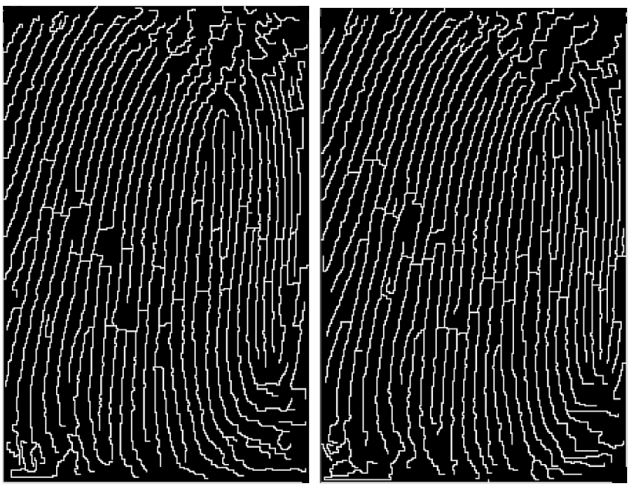


Figure 13 : Deux squelettes provenant de deux images du même doigt, identiques à l'œil nus. L'algorithme détecte 93 et 102 bifurcations et terminaisons pour le premier, et 89 98 pour le deuxième.

Quant à la phase d'extraction, un arrondi sur les angles et les distances la rend plutôt fiable. Un squelette, même tourné de quelques degrés par rapport à l'original, donne des résultats quasiment identiques (figure 14). Les résultats pourraient le devenir totalement sous réserve d'une modification des arrondis effectués.

Image	Droite	Rotation 2°
Algorithme d'extraction : Bifurcation	Longueurs (unité arbitraire) : [11, 46, 54, 57, 62, 65] Angles (en radians) : [0.0, 0.3, 0.3, 1.2, 2.0, 2.3]	Longueurs (unité arbitraire) : [10, 46, 54, 57, 62, 64] Angles (en radians) : [0.0, 0.3, 0.3, 1.2, 2.0, 2.2]

Figure 14 : Similitudes des résultats entre une image et sa version tournée de 2 degrés.

Conclusion :

Les objectifs que je m'étais fixés sont atteints. Néanmoins, la phase de squelettisation pourrait être perfectionnée. La qualité des images pourrait être améliorée, ce qui aurait un impact significatif sur le résultat. La phase d'extraction est, à l'inverse, plutôt fiable.

Bibliographie :

[1] Grégory Trolliet : Robustesse d'un mot de passe : article publié le 25 avril 2020, https://trolliet.info/blog/014-passwordrobustesse/
[2] Jean-Paul DELAHAYE : L'art et la science des mots de passe : Pour la science n°492, octobre 2018, p.80-85
[3] Franck Jeannot : MD5 message-digest algorithm : publié en septembre 2018, https://www.franckjeannot.com/wp-content/uploads/md5.pdf
[4] Hasnaoui Nassim Aboubakr : Mémoire de projet de fin d'études : La reconnaissance automatique des empreintes digitales : Université Abou Bakr Belkaïd de Tlemcen, https://docplayer.fr/88496529-La-reconnaissance-automatique-des-empreintes-digitales.html
[5] RabaaDouss : Thèse : Squelettisation d'images en niveaux de gris et applications : Université Paris DESCARTES, https://tel.archives-ouvertes.fr/tel-01578084