Samuel Rodriguez-Lozano
Hugo Bollon

Info 002 - Cryptologie

L'algorithme NeuralHash d’Apple et son utilisation pour la lutte contre la pédo-pornographie

Introduction

De nos jours, avec l'avènement d'internet, la possession et le partage de données numériques est devenu extrêmement simple et fait partie de notre quotidien.
Ces données peuvent prendre la forme de messages, de photos, de documents ou de n'importe quoi d'autre. La majeure partie du temps ces données sont anodines et parfaitement légales, malheureusement elles ne le sont pas toujours.

En effet, certains types de contenu sont interdits à la possession, ce qui est le cas par exemple pour tout contenu à caractère pédo-pornographique. Néanmoins, ces données sont difficiles à détecter de manière automatique.
Cette problématique est grave et doit être une préoccupation majeure. Apple a pris les devants en créant et en utilisant un nouvel algorithme de hash nommé *NeuralHash* afin de détecter ces contenus tout en préservant la vie privée des utilisateurs !

Dans ce document nous allons étudier en détails cet algorithme qu'est *NeuralHash*, voir comment il fonctionne, une implémentation de ce dernier et enfin les limites et controverses de *NeuralHash*.

I. L’algorithme *NeuralHash*

A. Les principes

NeuralHash a été créé par Apple afin de sécuriser leur système de cloud-storage *iCloud*, il a été conçu en ayant comme priorité la vie privée des utilisateurs.
En effet, contrairement à la plupart des solutions existantes de détection comme une analyse dans le cloud ou manuelle, l’algorithme d’Apple effectue une analyse directement sur l’appareil en comparant le hash généré par *NeuralHash* avec une base de donnée regroupant l’ensemble des hash signalé pour contenu illicite.

En effet, il est encore possible d’avoir des images d’enfants nus dans son téléphone. Cela est surement le cas, notamment chez les parents, qu’ils aient des images de leurs enfants nus. Pour limiter le nombre de faux-positifs, la détection d’une photo se fait sur une liste pré-établie. Cette base de données est fournie à Apple par des organismes de protection de l’enfance dont notamment la NCMEC (National Center for Missing & Exploited Children).

Le principe est simple, *NeuralHash* va analyser l’image en la convertissant vers un nombre unique (hash) spécifique à cette dernière. Seulement une image ressemblant fortement pourra générer ce même hash.

Les hash ne sont pas directement transmis à Apple, puisque ce serait un manquement au chiffrement end-to-end d'iCloud, qui se veut être justement bon pour la protection de la vie privée.

NeuralHash est un algorithme de hachage d'images basé sur un réseau de neurones. Le réseau de neurones est entraîné pour comprendre la structure d’informations sur l’image. L’algorithme peut donc passer outre les transformations tel que le recadrage, l’ajout de flou, l’ajout de bruit…
Cela signifie qu’une image redimensionnée, rognée ou encore d’une résolution inférieure conservera le même hash que l’originale généré par *NeuralHash*.

Il ne serait en effet pas possible d’avoir le même résultat avec un algorithme plus conventionnel et plus populaire comme SHA ou encore MD5, car il serait possible de contourner la détection en modifiant un seul bit de l'image.

Ici un exemple avec une image libre de droit téléchargée sur Unsplash. Nous allons simplement modifier la composante rouge du premier pixel avec le code python suivant:

```python
from PIL import Image

img = Image.open('input.jpeg')
pixels = img.load()
r, g, b = pixels[0, 0]
r = 255
g = 0
b = 0
pixels[0, 0] = r, g, b
img.save('output.jpeg');
```

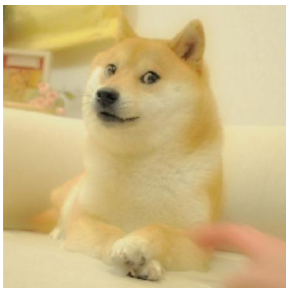
	Image original	Image modifié
		
SHA-1	bd18b9c423b1f56e82c51112e48f4363fc50336d	abd430b763210509471055f06aea239dfe074deb
SHA-256	276b3c6074089b54721b459280a2ab7ef73482de4a01a204f7d86f6d25a4b194	049bfd27ea73e5ede9bec541a0c12b510c722fe53ac5127eecdf2f93a1b7540f
SHA-384	7e20e1b7f84c71c0489d0c0db2b5d6ab35606ad8c53c4942b8393e79d7750161a4faf600ff6b80cc7a1e6776cdebbb69	5fa32a69fc1677b7abff770dc84c3bcb5af8a72a69bcb0bcb327dde7b05998572acc7fab0385d67879a158c832d9a903

SHA-512	5775074b1cdcc6d751be67f37fd23c41f9e5703ab211a0913efde6e96de83724c0eace98f63146f33d944bad6e87d3166ceed819cfa7ebdb4f5d6acc95538e99	996d5c02d91f7457b57486a68328c1c8b1838e1f50a11c7c0a6bfff1a06832b8ba1ce8ead6578ca859858092f09a2e2674096fee482da23513e2476999f3f685

On peut voir qu'avec un seul pixel modifié, le SHA de l'image change intégralement.

Dans l'idée, voici ce que devrait nous rendre NeuralHash sur une même image avec des modifications :

	Image original	With blur	With rotate
			
NeuralHash	11d9b097ac960bd2c6c131fa	11d9b097ac960bd2c6c131fa	11d9b097ac960bd2c6c131fa

	Image brightness	With smooth	With more red
			
NeuralHash	11d9b097ac960bd2c6c131fa	11d9b097ac960bd2c6c131fa	11d9b097ac960bd2c6c131fa

Tous les hash de l'image et des images avec des modifications doivent être les mêmes. L'algorithme reconnais le contexte de l'image et comprend qu'il s'agit de la même image.

Malheureusement, Apple n'a pas rendu public son algorithme qui est à ce jour propriétaire. Nous ne pourrons donc ni étudier son code ni essayer de créer notre propre implémentation.

B. Fonctionnement

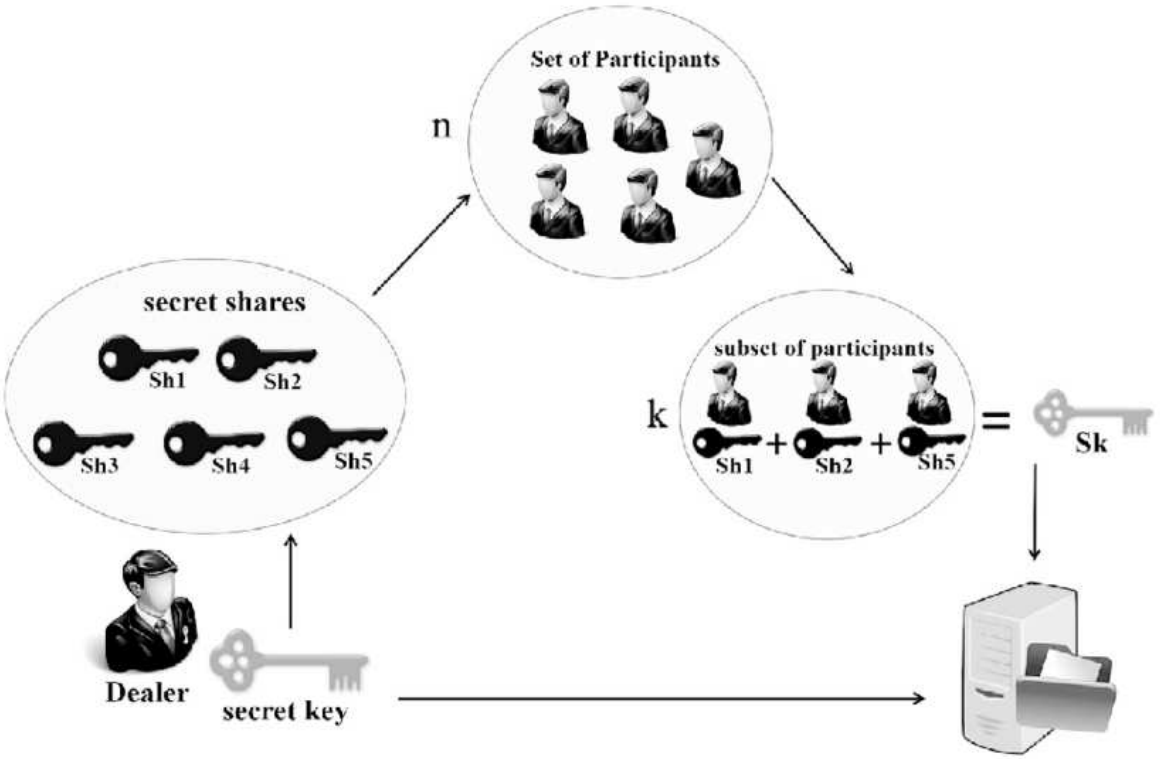
Apple a mis en ligne un document scientifique, accessible à tous, expliquant le protocole en détail. La publication scientifique est coécrite par Dan Boneh, une pointure en cryptographie déjà reconnue dans son milieu (né en 1969, il est chercheur en cryptographie. Il est professeur à Stanford et a reçu le prix Gödel en 2013).

Le réseau va comprendre le contexte de l'image en transformant l'image en vecteur de nombres.
On aura des vecteurs proches si les images se ressemblent. Dans le cas inverse, les vecteurs seront éloignés les uns par rapport aux autres.
Les hashs sont donc calculés mais ne sont pas transmis à Apple car ce serait contraire au chiffrement end-to-end d'iCloud.

Apple utilise donc le secret réparti. D'après Wikipédia:
"*Le secret réparti ou le partage de secret consiste à distribuer un secret, par exemple une clé ou un mot de passe, entre plusieurs dépositaires. Le secret ne peut être découvert que si un nombre suffisant de dépositaires mettent en commun les informations qu'ils ont reçues. En revanche, un nombre inférieur de dépositaires n'apporte aucune information sur le secret.*"

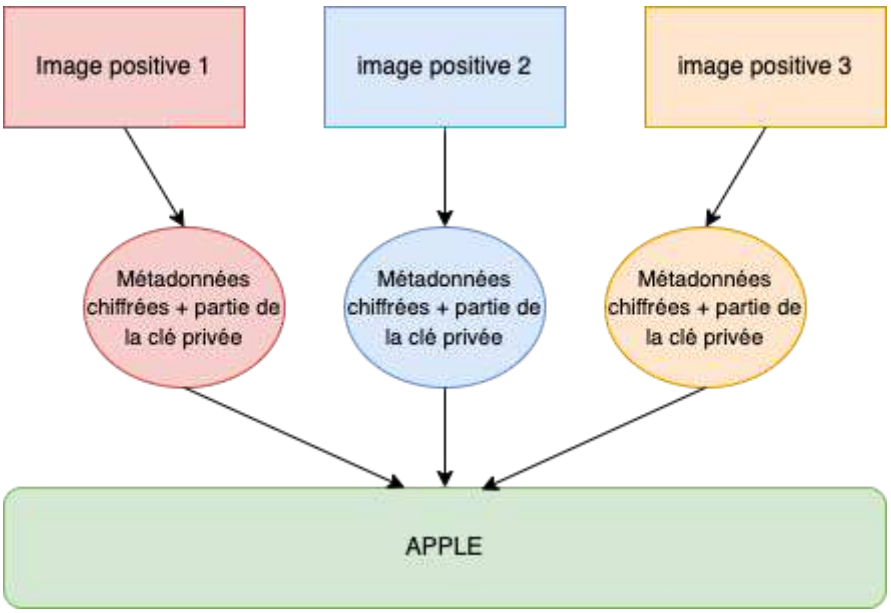
Ce système à été créé par Adi Shamir (aussi connu pour avoir inventé l'algorithme RSA, avec Ron Rivest et Len Adleman) et George Blakley en 1979.
Le secret répartie permet de révéler un secret uniquement si l'on possède ***k*** parties du secret.

Par exemple, pour un secret divisé en ***n*** parts avec un seuil ***k***, il faut que ***k*** parts soient réunis pour révéler le secret. Il n'y a pas besoin des ***n*** parts.



En quoi ce système est-il vraiment très intéressant dans notre cas? Car on est sûr qu'Apple ne peut avoir aucune donnée tant que plusieurs images ne sont pas considérées comme positives.

A chaque fois que le hash d'une image correspond à un mauvais hash, ses métadonnées cryptées (à l'aide d'une clé) sont remontées à Apple avec une petite partie de la clé.



Lorsque Apple aura ***k*** partie de la clé, ils pourront déchiffrer les métadonnées.

Ce nombre ***k*** n'est pas public, seuls les ingénieurs travaillant sur ce sujet semblent le connaître.

C. L'algorithme

Comme évoqué précédemment, l'algorithme en lui-même n'est pas connu du grand public car il n'est malheureusement pas open-source.
Nous allons donc étudier une autre facette de cette technologie que l'on appelle les ***hash collisions***. Une collision est le fait d'obtenir le même hash à partir de deux objets différents, en l'occurrence des images dans le cas de *NeuralHash*.

Une attaque très connue nommée "**adversarial attack**" dans le domaine de l'intelligence artificielle consiste à ajouter du "bruit" à une image afin de tromper le réseau neuronal utilisé par l'algorithme de machine learning afin d'obtenir un hash différent tout en gardant une image similaire.
Ce type d'attaque peut typiquement être appliqué à *NeuralHash*. En Août dernier, un utilisateur Reddit du nom de *AsuharietYgvar* à expliquer comment exporter et utiliser le modèle de données d'Apple pour *NeuralHash* (disponible ici: https://github.com/AsuharietYgvar/AppleNeuralHash2ONNX).

Dans le cas de *NeuralHash* donc, les attaques de type “**adversarial attack**” peuvent prendre deux formes:

- Le même hash mais deux images différentes
- Un hash différents mais deux mêmes images

Dans le premier cas, un “bruit” (en anglais noise) va être ajouté à une image afin de transformer son hash en celui d’une autre. Pour que cette méthode fonctionne il faut donc mettre un noise exact et donc le déterminer, nous verrons un peu plus tard comment le faire.

Dans l’exemple ci-dessus, un bruit à été ajouté à l’image de chat afin d’obtenir le hash de celle du chien avec *NeuralHash*. Le hash des ces image est 11d9b097ac960bd2c6c131fa. Ceci est donc ce qu’on appelle une ***hash collision*** avec *NeuralHash.*

Cette faiblesse est donc très problématique car certaines personnes mal intentionnés pourraient déguiser des images bénignes en images signalées comme contenu illégal ce qui signalait des utilisateurs n’ayant rien fait de mal. À l’inverse, des images de contenu illégal pourraient être camouflées en images neutres via ce procédé.

Dans le deuxième cas, un bruit aléatoire et léger peut être ajouté à une image au contenu néfaste. Ceci changerait son hash et camouflerait donc l’image aux yeux d’Apple.
Cela signifie donc que l’ensemble du système de détection d’Apple peut-être facilement berné tout simplement en ajoutant un filtre sur une image d’abus infantile afin de passer entre les mailles du filet sans pour autant changer drastiquement l’image…

Une solution afin d'essayer de contrer ce genre d'attaque serait de ne pas stocker les **CNN** (Convolutional neural network ou réseaux neuronal convolutif en français) sur les appareils destinés aux utilisateurs et donc des les décentralisés.
En effet, sans avoir accès à ces derniers, ces attaques seraient bien plus compliquées à mettre en œuvre.
Malheureusement, ce plan d'action signifierait également de devoir effectuer l'analyse des données directement sur les serveurs d'Apple signifiant que toutes les photos seraient partagées brisant ainsi la confidentialité de l'utilisateur.

Une solution alternative serait de faire usage de plusieurs **CNN** au lieu d'un unique, obligeant les attaquants à prendre en compte plusieurs modèles en même temps et rendant donc ces attaques plus difficiles.
Cette solution n'est malheureusement pas parfaite non plus, en effet, la charge de travail de l'opération de hash augmente sur l'appareil avec plusieurs modèles.

Enfin, une dernière solution envisageable serait de préparer les images avant de les passer dans l'algorithme *NeuralHash*. Par exemple en appliquant un filtre, en rognant l'image, etc.
Cette action pourrait contrer certaines attaques basiques. Néanmoins, cette solution ne pourrait pas être fiable dans tous les cas de figure.

Afin de calculer une collision, nous pouvons utiliser ce repositiory Github: https://github.com/anishathalye/neural-hash-collider .
Cependant, il faut au préalable réussir à se procurer un modèle NeuralHash au format ONNX. Pour ce faire, l'internaute connu sous le nom de *AsuharietYgvar* et dont nous avons parlé précédemment, a rendu disponible un tutoriel: https://github.com/AsuharietYgvar/AppleNeuralHash2ONNX .

Il faut tout d'abord avoir accès à un appareil MacOS Ou IOS (Jailbreak) et récupérer les fichiers:

- neuralhash_128x96_seed1.dat
- NeuralHashv3b-current.espresso.net
- NeuralHashv3b-current.espresso.shape
- NeuralHashv3b-current.espresso.weights

Ensuite, il nous faudra décoder les fichiers .net et .shapes du modèle et enfin les convertir au format ONNX grâce à l'outil TNN créé par *Tencent Youtu Lab* and *Guangying Lab*: https://github.com/AsuharietYgvar/TNN (forké sur son compte).

À partir de là nous avons tous les outils nécessaires pour faire fonctionner le générateur de collision, il suffit de le lancer en lui donnant une image et un hash que doit match l'image de sortie.
Exemple:

```
python collide.py --image picard.png --target
59a34eabe31910abfb06f308
```

Image d'origine:

Une des images de sortie:

II. Limites

A. Private Set Intersection

Nous avons donc vu dans les sections précédentes, que les hash d'images sont donc comparées à d'autres hash dans une base de données, et que les matchs dans la base d'images étaient cachés par un seuil pour éviter les faux positifs.

On sait aussi qu'Apple ne doit avoir accès, ni au photos, ni aux méta-données, ni aux NeuralHash des photos d'un utilisateur, à cause du chiffrement end-to-end d'iCloud (ou grâce, pour les utilisateurs) et qu'Apple ne peut pas diffuser la liste de photos interdites :

- Pour ne pas diffuser eux-même de contenu pornographique.
- Pour limiter les abus comme les "adversarial attack".

Il semble donc impossible, en sachant que les usagers ne peuvent pas avoir accès aux hashs d'Apple, et qu'Apple ne peut pas avoir accès aux informations de ses clients, de savoir si une des images est dans la base de données de contenu illégale.
Apple à réussi à résoudre le problème grâce à Private set intersection

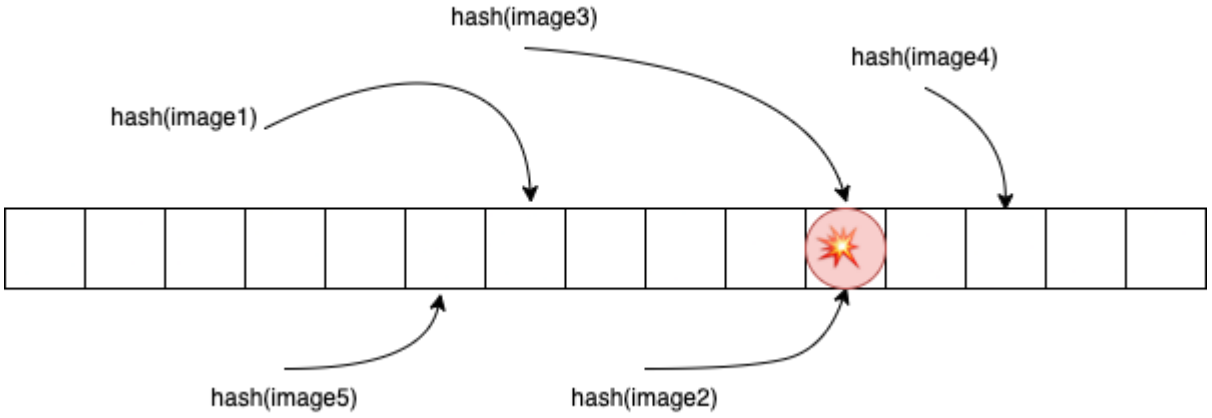
D'après Wikipédia:
"*Private set intersection is a secure multiparty computation cryptographic technique that allows two parties holding sets to compare encrypted versions of these sets in order to compute the intersection. In this scenario, neither party reveals anything to the counterparty except for the elements in the intersection*."

Dans un premier temps il faut expliquer ce qu'est une "cuckoo table".

D'après Wikipédia:
"Cuckoo hashing is a scheme in computer programming for resolving hash collisions of values of hash functions in a table, with worst-case constant lookup time. The name derives from the behavior of some species of cuckoo, where the cuckoo chick pushes the other eggs or young out of the nest when it hatches; analogously, inserting a new key into a cuckoo hashing table may push an older key to a different location in the table."

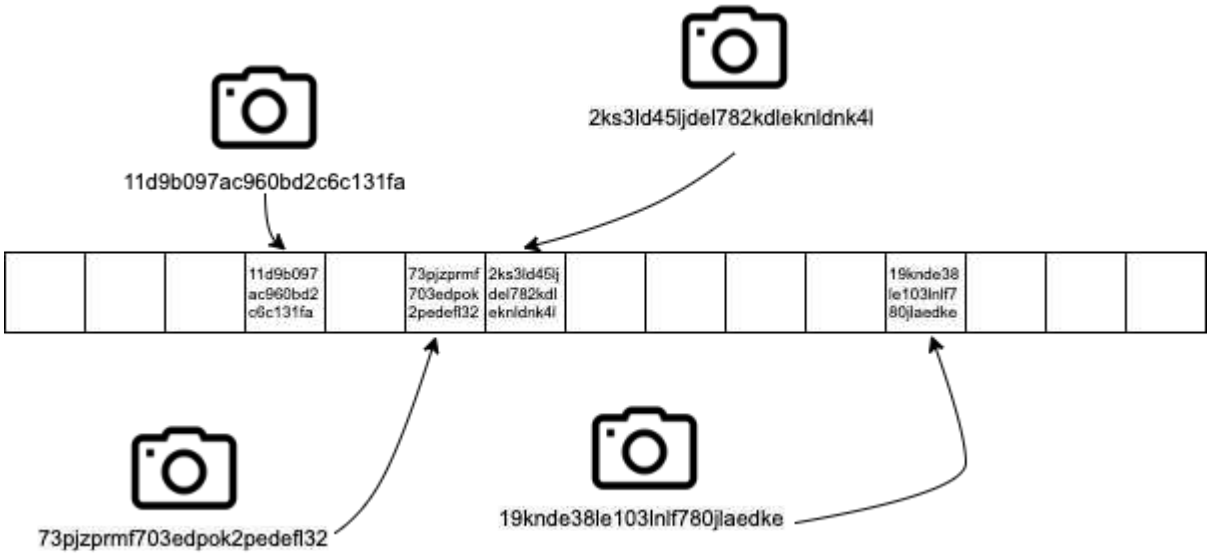
Pour ranger des hash, on pourrait créer une table de hachage classique, qui est une structure de données qui permet une association clé–valeur. Les tables de hachage ont quand même une faiblesse, elles ne gèrent pas les collisions. En effet, différentes clés, deux voire davantage, peuvent avoir le même hash en sortie.

Lors du rangement, plusieurs clés se retrouveraient donc dans la même case du tableau, il y a donc une collision.

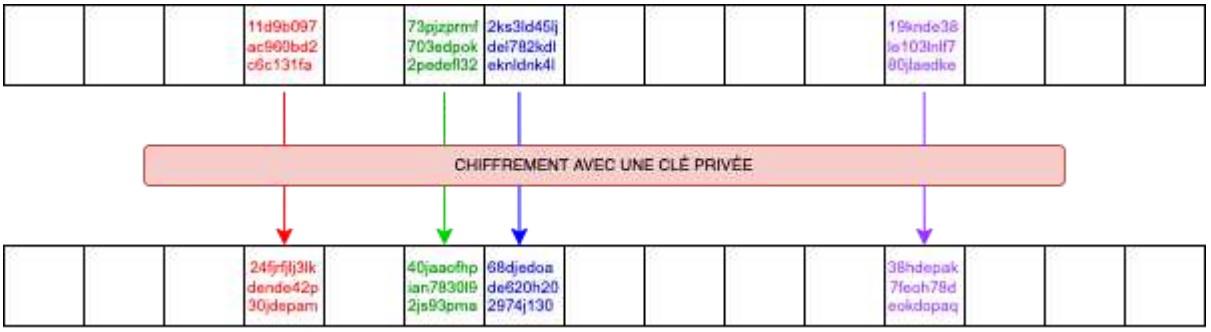


Une Cuckoo table règle ce problème. A la fin, tous les hashs seront correctement rangés sans qu'il y ait de collisions.

Apple calculent tous les hash de ses images, puis les ranges dans une cuckoo table en trouvant une position unique pour chaque hash :

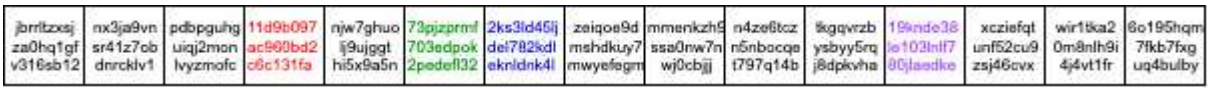


Puis, les hash sont cachés avec une clé privée. Il est important qu'ils gardent la même position dans la table après le hachage avec la clé privée.



Le reste de la table, c'est-à-dire toutes les cases vides, qui n'ont pas étés remplis par un hash à la base, sont remplis par un hash au hasard.

Sur notre exemple, il s'agit des cases en noires:



C'est cette table qui va être envoyée à tous ses utilisateurs. Apple réussit donc à envoyer sur les différents appareils la liste des images interdites, et les utilisateurs n'ont aucun moyen de voir à quoi elles ressemblent, ni de savoir quel est leur hash car ils ne possèdent pas la clé secrète d'Apple.

Les utilisateurs vont envoyer les métadonnées de toutes leurs images. Pour chaque image, on va calculer sa position dans la cuckoo table, et si son index correspond à l'index d'une image interdite par Apple, alors ce sera la même image donc Apple va pouvoir considérer l'utilisateur comme ayant une photo interdite.

B. Apple seul décisionnaire

Comme nous l'avons vu précédemment, la base de données d'images à été fournie à Apple par des organismes de protection de l'enfance. Il est naturel de se demander si Apple à fait ou non des modifications sur cette base de données.
De plus, c'est donc les organismes qui ont choisi ce qu'était du contenu pédo-pornographique ou ce qui ne l'était pas. Nous savons pertinemment que la data d'entraînement est primordial et peut clairement orienter le résultat de classification final. C'est bien la data qui sera donnée en entrée qui créera toute "l'intelligence" du réseau de neurones.
De plus, on peut se demander si la liste est la même pour tout le monde, il pourrait être facile pour un gouvernement de faire pression sur Apple pour appliquer une certaine censure. En effet, dans certains pays, des contenus qui pourraient paraître anodins sont complètement interdits et peuvent même envoyer des personnes en prison. On pense par exemple à la loi adoptée en juin 2021 en Hongrie qui prévoit "que la pornographie et les contenus qui représentent la sexualité ou promeuvent la déviation de l'identité de genre, le changement de sexe et l'homosexualité ne doivent pas être accessibles aux moins de 18 ans."

Il n'y a pour l'instant aucune autorité ou observateur externe qui pourrait vérifier qu'Apple joue le jeu.

C. Contourner cette interdiction

Malheureusement, il faut aussi prendre en compte le fait qu'il est très simple de contourner le système mis en place par Apple, malgré tous les efforts qu'ils ont mis en place.

En effet, la protection d'Apple est effective sur les médias stockés sur leur propre service de cloud storage, *ICloud*. Cependant, si un utilisateur d'Iphone ou de tout autre produit Apple décidait de se servir d'un autre service pour la sauvegarde de ses données, comme *Google Image* ou *Dropbox* par exemple, il passerait alors outre la surveillance d'Apple.

Ensuite, comme nous l'avons vu en détails précédemment, certaines personnes mal intentionnés pourraient masquer des médias interdits en y appliquant du noise pour le "camoufler" en contenu lambda. Ce qu'on appelle les "hash collisions"

Enfin, cette technologie est efficace pour les contenus statiques, autrement dit les images et photos. Néanmoins, en ce qui concerne les contenus vidéo c'est inefficace et le contenu illicite est donc indétectable.

Conclusion

Le système mis en place par Apple pour lutter contre la pédo-pornographie est une technologie intéressante et démontre une volonté de la part d'Apple de lutter contre ces crimes tout en protégeant la vie privé de ses uutilisateurs.

Malgré tout, lorsque l'on creuse un petit peu, on se rend rapidement compte des limitations de ce système et du nombre de possibilités de contournement de ce dernier.
La confidentialité des utilisateurs est aussi une zone un peu floue, en effet, sachant qu'il est possible de créer des faux positifs et qu'après un certain nombre de signalements une vérification manuelle entre en œuvre, des attaques pourraient être envisagées voir même des "ransomware" visant à compromettre la confidentialité des utilisateurs.

Néanmoins, *NeuralHash* reste très intéressant et pourrait être utilisé dans d'autres domaines d'applications.

III. Bibliographie

Documents Apple :

- https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf
- https://www.apple.com/child-safety/pdf/Apple_PSI_System_Security_Protocol_and_Analysis.pdf

Fonctionnement :

- https://datascientest.com/nlp-word-embedding-word2vec

Cuckoo table :

- https://www.youtube.com/watch?v=HRzg0SzFLQQ
- https://www.youtube.com/watch?v=OBuGqu2d4v4

Controverses et limites:

- https://www.20minutes.fr/monde/2958019-20210121-etats-unis-accusee-censure-application-wechat-poursuivie-californie
- https://towardsdatascience.com/apples-neuralhash-how-it-works-and-ways-to-break-it-577d1edc9838
- https://theconversation.com/apple-peut-scanner-vos-photos-pour-lutter-contre-la-pedocriminalite-tout-en-protegeant-votre-vie-privee-si-la-societe-tient-ses-promesses-166074