View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by RERO DOC Digital Library

h e g

Haute école de gestion
Genève

Techniques de récupération de messages sur iOS dans le cadre d'une enquête de police

Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES

par :

Léa YOUYOU

Conseiller au travail de Bachelor :

David BILLARD

Genève, le 03 Juillet 2015

Haute École de Gestion de Genève (HEG-GE)

Filière Informatique de Gestion

Hes·so GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du diplôme d'informaticienne de gestion.
L'étudiante a envoyé ce document par email à l'adresse remise par son conseiller au travail de Bachelor pour analyse par le logiciel de détection de plagiat URKUND, selon la procédure détaillée à l'URL suivante : http://www.urkund.fr/student_gorsahar.asp.
L'étudiante accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seule le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Genève, le 03 Juillet 2015

Léa YOUYOU

Remerciements

Je souhaiterais avant tout remercier la Haute Ecole de Gestion ainsi que les professeurs m'ayant soutenue durant ces deux années de formations.

Je tiens, tout particulièrement, à remercier Monsieur David Billard qui a suivi l'avancé de mon travail de Bachelor, du début à la fin. Je le remercie pour sa disponibilité, sa patience ainsi que ses encouragements durant ce travail. Il a su me fournir de précieux conseils qui m'ont permis de progresser tout au long du travail de Bachelor.

Enfin, j'aimerais remercier mes proches, ma famille et mes amis, qui ont su me soutenir durant la réalisation de mon mémoire.

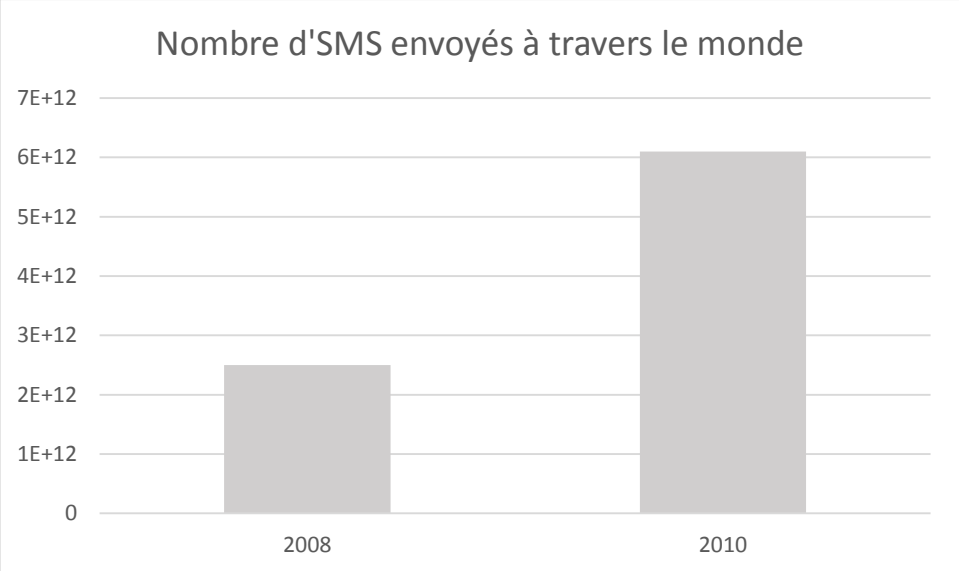
Résumé

La croissance du nombre d'appareils connectés est en net progression depuis quelques années. En effet, qu'ils s'agissent de smartphones ou de tablettes, l'appareil tactile fait partie intégrante de notre quotidien. C'est celui que l'on n'oublie jamais. Dès le réveil mais également pendant une pause, à midi ou le soir, nous sommes entrés dans une aire où la connexion est devenue indispensable. Rester connecté avec le monde est devenu une nécessité pour la grande majorité des personnes. Ce phénomène se réalise notamment par l'envoi de messages assez régulier.

Auparavant la vocation d'un message était assez minimale ; il s'agissait d'un mode de transmission permettant d'informer la personne qu'elle avait reçu un message vocal. L'orientation du message a pris un tournant radical, il est devenu un des moyens de communication le plus utilisé de par sa rapidité et son coût (seul une connexion wifi suffit, disponible gratuitement).

Et pour cause, le nombre de messages envoyés est devenu impressionnant. En effet, d'après les statistiques de « planetoscope », chaque seconde, 200'000 SMS sont envoyés à travers le monde.

Figure 1 : Statistique du nombre de SMS envoyés dans le monde en 2008 et 2010



(Planetoscope, 2015)

La réflexion n'est plus autant posée avant l'envoi d'un message.

Les conséquences de ce phénomène ? D'une part, les données stockées sur l'appareil sont de quantité plus importante mais également la volonté de supprimer certains messages, que nous ne souhaitons plus conserver pour une raison ou pour une autre, est présente.

Table des matières

Liste des tableaux

Liste des figures

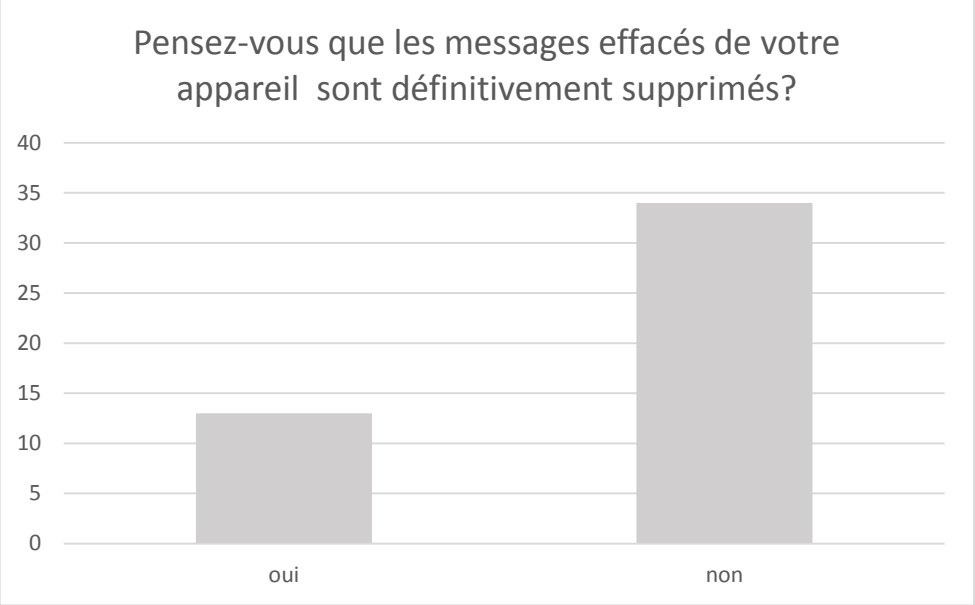
1. Introduction

Les appareils connectés tels que les smartphones ou tablettes font l'objet d'investigations relatives à la commission d'une infraction. Ces examens sont réalisés par la police judiciaire lors d'une enquête.

Le message est un moyen de communication devenu primordial et efficace entre individus. Par conséquent, le Short Message Service, abrévié SMS ou encore l'iMessage, service proposé par Apple, permettent à la police judiciaire la résolution d'enquêtes.

Après sondage effectué sur 47 participants, nous constatons aujourd'hui que la majorité des personnes (34) pensent qu'un message supprimé sur un appareil n'est pas définitivement effacée de ce dernier.

Figure 2 : Résultat du sondage concernant la suppression définitif des messages



(Sondage)

En effet, la suppression physique est une tâche trop compliquée à gérer, par conséquent les messages sont effacés de manière logique.

A Genève, la grande majorité de ces appareils sont des produits Apple (iPhone, iPad ou iTouch). Ainsi, lors de la réalisation de mon travail de Bachelor dans le cadre de ma formation d'informaticienne de gestion, le sujet concernera le système d'exploitation mobile iOS, donc les techniques de récupération de messages sur un iPhone dans le cadre d'une enquête de police.

2. Contexte

Les données sur iPhone sont stockées sur une mémoire de type flash (petite puce électronique). La marque à la pomme utilise la mémoire flash de type NAND puisque ce type de mémoire est rapide et coûte moins cher.

2.1 Base de données

2.1.1 SQLite, moteur de base de données embarqué

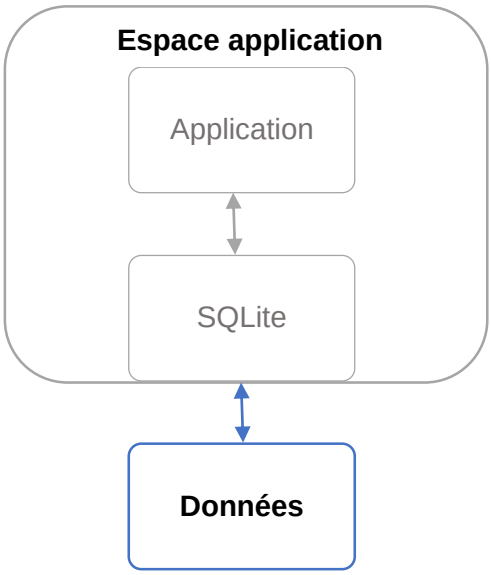
La base actuellement utilisée sur iOS est SQLite, il s'agit d'un moteur de base de données embarqué et extrêmement léger.

Le système de gestion de base de données est directement intégré à l'application, contrairement au schéma habituel client/serveur utilisé par la majorité des systèmes de gestion de base de données (MySQL par exemple).

La bibliothèque et son code source sont « open source », c'est-à-dire qu'ils peuvent être utilisés sans restriction.

Ainsi, les données telles que les messages ou encore les contacts sont stockées dans des bases de données de type SQLite.

Figure 3 : SQLite, Modèle embarqué



(Wikipédia, SQLite)

2.1.2 Fichier SQLite

Avant tout, il est important de décrire la structure de la base de données.

La base contient un fichier principal (.db) ainsi que deux fichiers (-wal et –shm) créés automatiquement dès la création de la base de données (.db).

Le format de fichier de données qui est utilisé est de type SQLite. C'est-à-dire que les données sont stockées dans des fichiers dont le langage est SQL.

Le fichier principal (sur la Figure 4, il s'agit du fichier « sms.db ») est divisé en plusieurs « pages » (terme donné par SQLite) de taille fixe. Ces pages sont numérotées à partir de 1 par SQLite. Chaque page a un rôle, par exemple : contenir des informations sur la structure de la base de données, contenir les données elles-mêmes etc.

Les 100 premiers octets du fichier principal (dans la première page) définissent et décrivent la base de données.

Figure 4 : Structure simplifiée du fichier principal « sms.db »

1	Page 1
2	Page 2
3	Page 3
…	…

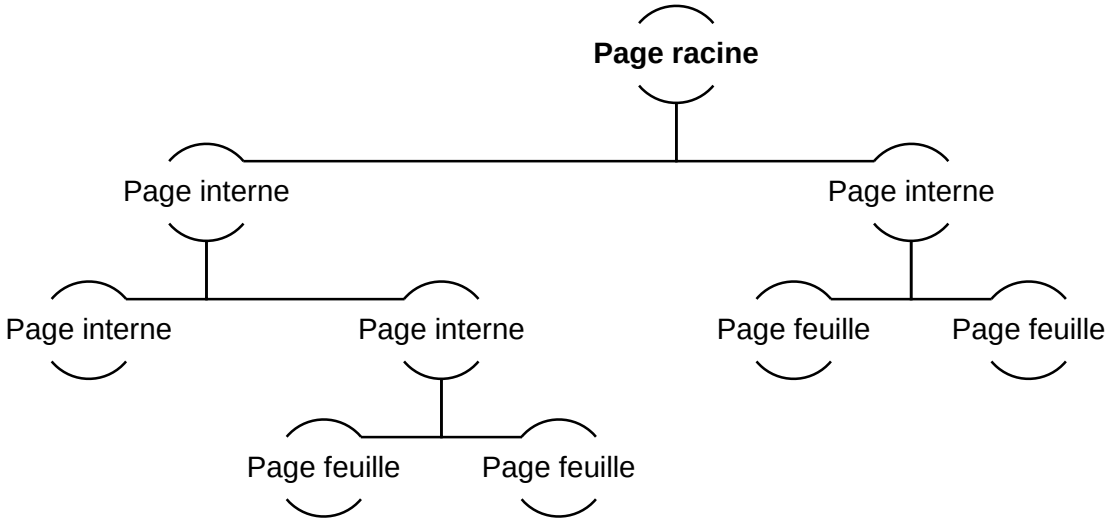
(D'après les explications de SQLite, The SQLite Database File Format)

Les pages que nous allons étudier dans le cadre de la récupération de messages sont les pages de type B-Tree, celles qui contiennent les données.
Les pages B-Tree peuvent être composées de page :

- Interne : contenant des pointeurs vers d'autres pages internes ou autre pages de type B-Tree
- Feuille : contenant des données
- Racine : page unique

Comme nous pouvons l'observer sur Figure 5 qui est un exemple de structure d'une page de type B-Tree.

Figure 5 : Structure d'une page de type B-Tree

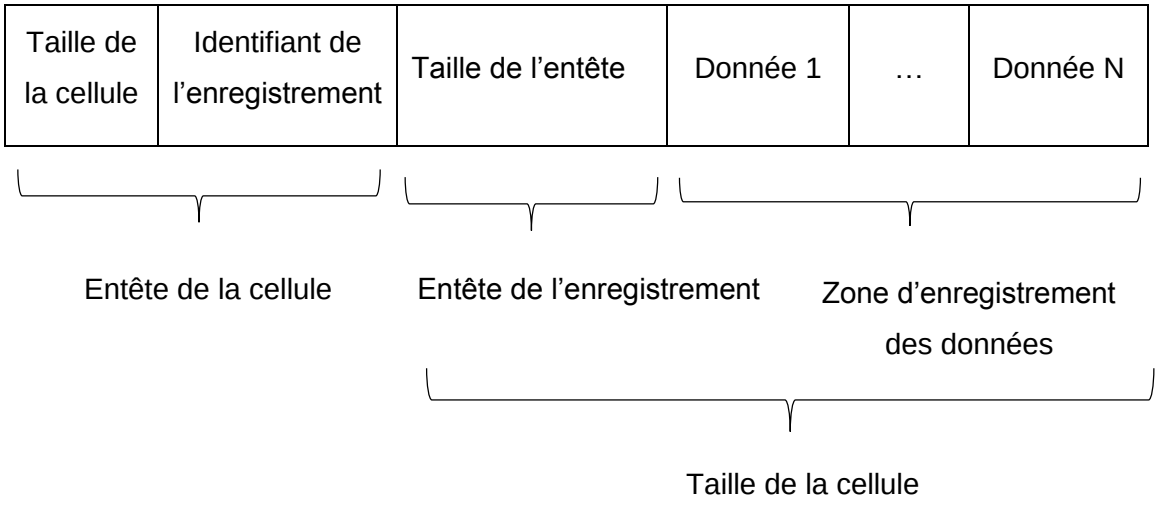


(D'après les explications de SQLite, The SQLite Database File Format)

Pour donner un exemple concret, la table Message (que nous analyserons plus tard) de la base de données « sms.db » est une page de type B-Tree composée d'une ou plusieurs pages.
La page racine de la table Message aura un pointeur vers un certain nombre de page(s) feuille(s) si la table est petite. En revanche, si la table contient un grand nombre de données, la page racine aura un pointeur sur une ou plusieurs pages internes qui pointera (ont) à leur tour soit sur une ou plusieurs page(s) interne(s) soit sur une page feuille.
Le nombre de pages utilisées par la table peut varier en fonction des modifications apportées à la base.

Voici la structure d'une cellule d'une page feuille, ces cellules sont importantes car il s'agit des cellules contenant les données enregistrées.

Figure 6 : Structure d'une cellule contenue dans une page feuille



(D'après les explications de SQLite, The SQLite Database File Format)

2.1.3 Gestion des erreurs

Dans la version précédant la 3.7.0 d'SQLite, le moteur SQLite utilisait un mécanisme dont le nom était « RollBack Journals » lui permettant de traiter les erreurs survenant lors de l'utilisation de la base de données.
Ainsi, lors d'une modification sur une page de la base de données, la page entière était sauvegardée avant modification dans un fichier de journal distinct.

Si la transaction se déroulait correctement (checkpoint), le fichier de journal distinct était supprimé. Dans le cas contraire, le fichier était conservé et lorsque SQLite voulait y accéder et trouvait qu'un fichier de journal était présent, cela signifiait qu'il y avait eu un problème. Par conséquent, le moteur se serait chargé de restaurer la base à son état précédent en utilisant le fichier de journal distinct.

Sur SQLite, depuis la version 3.7.0, un nouveau mécanisme appelé Whrite Ahead Log, a été introduit, dont l'abréviation est WAL.

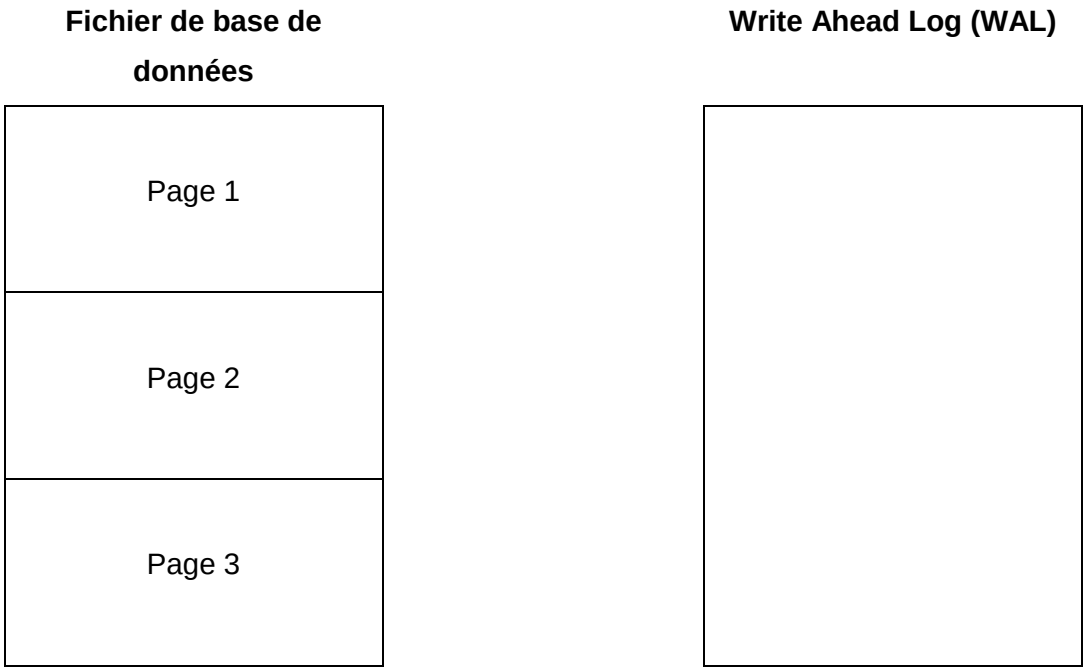
A la place de sauvegarder les pages originales avant la modification, les modifications (par exemple : suppression d'un message) sont directement effectuées.
Les modifications qui ont été effectuées sont écrites dans un fichier séparé (WAL) qui est automatiquement créé (en même temps que la base de données). Les pages modifiées restent dans le fichier WAL. Le moteur de la base de données lira les données depuis le WAL, jusqu'à ce qu'il y ait un checkpoint.
Lors d'un checkpoint, le moteur copiera les pages du fichier WAL vers le fichier de base de données principal.

Le checkpoint peut avoir lieu automatiquement lorsque la taille du fichier « sms.db-wal » atteint une certaine taille (par défaut 1000 pages) ou lors d'une commande SQL manuelle (wal_checkpoint PRAGMA;) ou encore par le biais d'un programme si une application accède à l'API interne du moteur SQLite.

Afin de mieux visualiser le mécanisme, voici quelques figures.

Sur la Figure 7, nous constatons qu'il n'y a aucune page dans le WAL. Il s'agit de l'état initial.

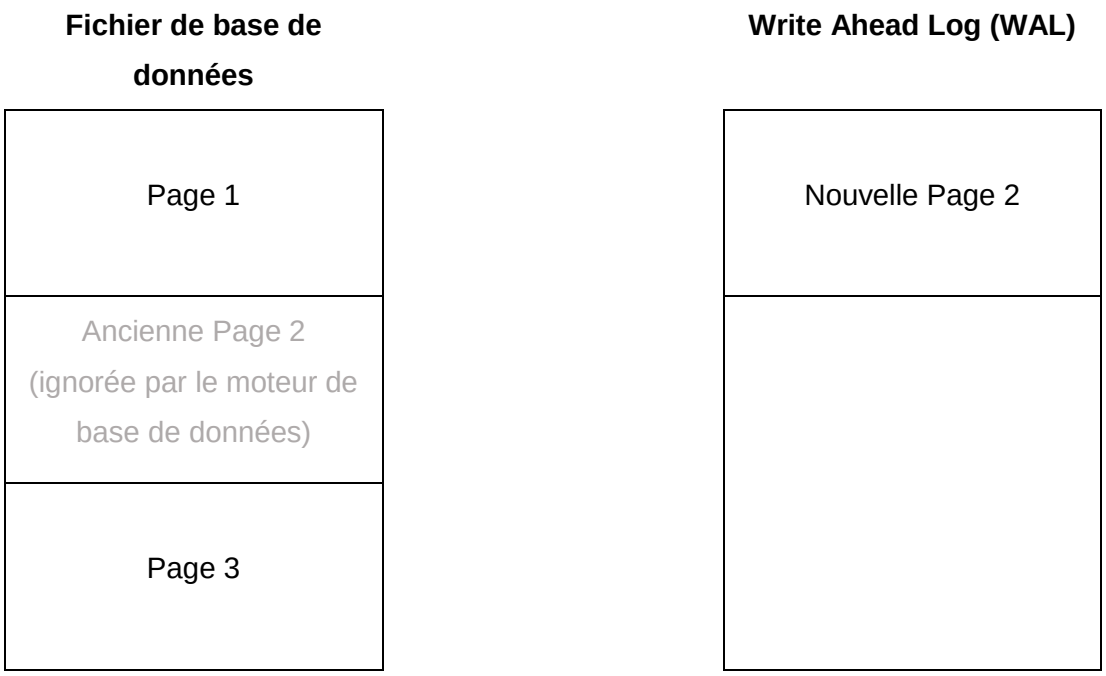
Figure 7 : Etat initial d'une base de données SQLite



(D'après les explications de SQLite, The SQLite Database File Format)

Sur la Figure 8, la page 2 est modifiée. La page 2 modifiée est écrite dans le WAL (Nouvelle Page 2). Le moteur de base de données utilise cette nouvelle version (modifiée) et non l'ancienne version (qui est ignorée) dans le fichier de base de données.

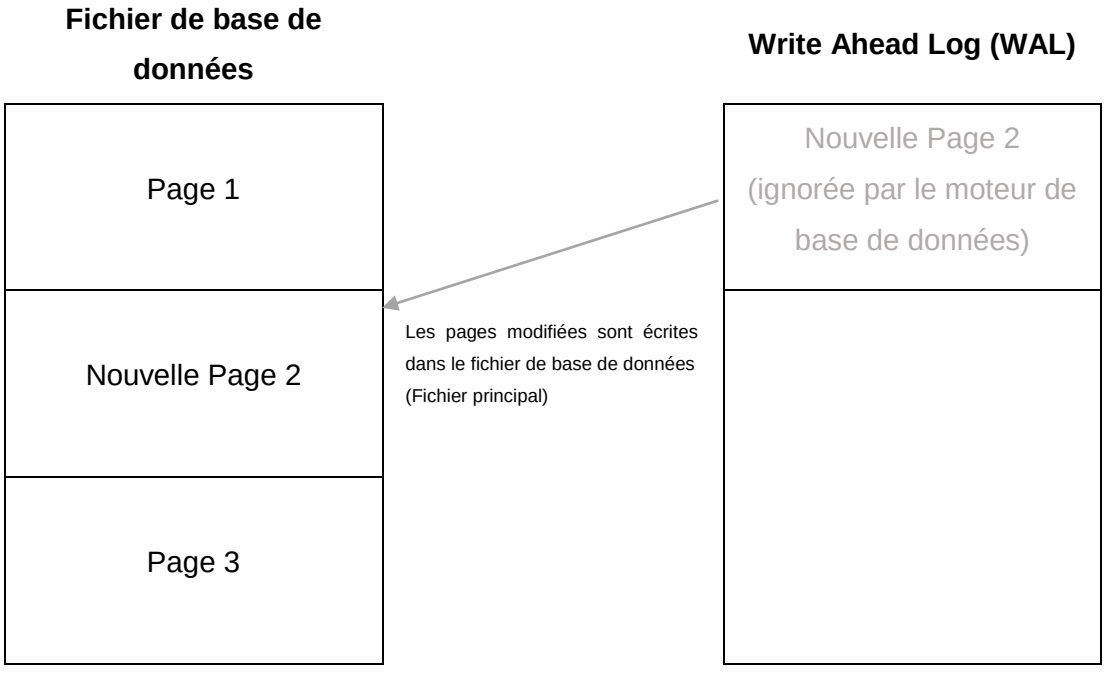
Figure 8 : Modification d'une page



(D'après les explications de SQLite, The SQLite Database File Format)

Lors d'une validation de transaction (opération checkpoint), la version modifiée de la page qui était dans le WAL est écrite dans le fichier de base de données.

Figure 9 : Validation d'une transaction



(D'après les explications de SQLite, The SQLite Database File Format)

3. Message sur iOS

Tout d'abord nous allons lister les différents types de messages que nous pouvons retrouver sur un appareil iOS. Nous avons la possibilité d'envoyer des messages via l'application Message inhérente à l'appareil iOS.

Voici la liste du type de messages que nous retrouvons :

- SMS (Couleur verte)
- iMessages (Couleur bleue) introduits à partir de la version 5 d'iOS
- Photo
- Vidéo
- Partitions audio

Mais nous pouvons également envoyer des messages via des applications telles que Whatsapp ou Viber.

Dans le cadre de ce travail, les SMS et les iMessages seront les types de messages que nous analyserons.

4. Sauvegarde

La sauvegarde est une fonctionnalité importante dans le cadre de la récupération de message. Il s'agit d'une mesure nécessaire pour éviter que les données ne soient corrompues, endommagées ou perdues. Cette dernière permet de prévoir l'imprévisible. En effet, tout support de stockage peut à tout moment tomber en panne et une erreur humaine peut également subvenir à tout moment (modification ou suppression par accident).

4.1 Techniques

Il existe différentes méthodes de sauvegarde et il est possible de combiner ces dernières.

4.1.1 ICloud

Sur les appareils, 5 gigas d'espace de stockage sont fournis gratuitement par l'entreprise lors de l'inscription à iCloud. Il s'agit d'un service de « cloud computing » proposé par Apple.

La sauvegarde iCloud se fait automatiquement lorsque l'appareil est connecté à un réseau Wifi. Le logiciel iTunes n'est donc pas nécessaire (bien qu'il soit également possible de l'effectuer à partir de ce dernier, le temps de sauvegarde sera moins long).

ICloud permet, non seulement, la sauvegarde automatique d'achats effectués dans les stores Apple mais également, celle des photos, vidéos, réglages, données des applications, l'organisation des applications, les sonneries, la messagerie vocale visuelle ainsi que les iMessages et SMS présents sur l'appareil.

Figure 10 : Sauvegarde iCloud sur iOS

4.1.2 Applications

Les sauvegardes peuvent également s'effectuer par le biais de logiciel tel que MobileTrans. Il s'agit d'applications de sauvegarde disponibles sur plusieurs systèmes d'exploitation permettant une sauvegarde manuelle des données d'un appareil iOS. Ainsi, l'utilisateur peut sauvegarder ses applications, contacts, messages, fichiers audio, vidéos, photos etc.

Nous avons analysé les méthodes de sauvegarde pour les appareils non jailbreakés (craqués). Mais lorsque l'appareil est jailbreaké, il existe des applications telles que PKGBackup ou AptBackup permettant d'effectuer une sauvegarde.

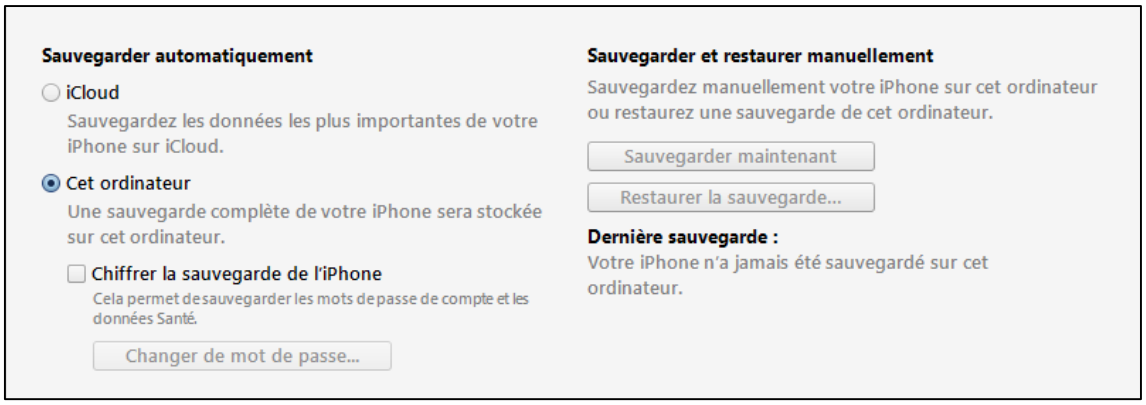
4.1.3 Ordinateur

4.1.3.1 Fonctionnement de la sauvegarde

Il s'agit ici de sauvegardes dites « manuelles » car l'utilisateur doit brancher l'appareil à l'ordinateur et effectuer la sauvegarde via le logiciel iTunes.

La sauvegarde iTunes comprend les contacts, les calendriers, les notes, l'historique des appels, les mémos vocaux, les achats sur les stores Apple, les photos et vidéos présentes sur l'appareil. Cette dernière comprend également les réglages, les données ainsi que l'organisation des d'applications, les sonneries ainsi que les iMessages et les SMS.

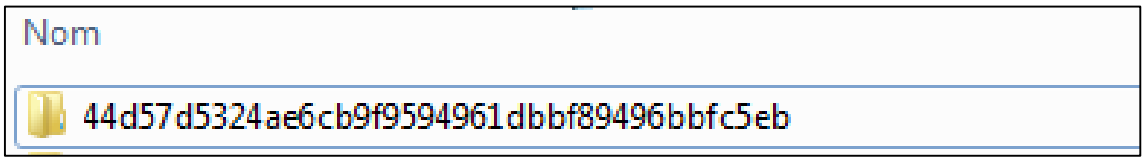
Figure 11 : Sauvegarde via le logiciel iTunes



4.1.3.2 Emplacement de stockage

L'emplacement de stockage des données de sauvegarde dépend du système d'exploitation. Les données sont stockées dans un dossier dont le nom est composé de chiffres et de lettres comme sur la Figure 12.

Figure 12 : Exemple d'un dossier de sauvegarde iTunes sur Windows 7



Voici les emplacements pour les systèmes d'exploitation les plus utilisés :

- **MAC OS :**
  « ~/Bibliothèque/Application Support/MobileSync/Backup/ »
  Le signe « ~ » représente votre dossier de départ.
- **Windows Vista, 7 et 8 :**
  « \Utilisateurs\(nom d'utilisateur)\AppData\Roaming\Apple Computer\MobileSync\Backup\ »
  AppData est un dossier caché, il faut donc faire afficher les fichiers, dossiers et lecteurs cachés pour y accéder.
  On peut également y accéder directement en cliquant sur Démarrer, en saisissant %appdata% dans la barre de recherche et entrer.
- **Windows XP:**
  « \Documents and Settings\(nom d'utilisateur)\Application Data\Apple Computer\MobileSync\Backup\ »
  Pareil que pour Windows vista, 7 et 8 pour les accès directs.

4.2 Suppression d'une sauvegarde

Il est simple d'effacer une sauvegarde effectuée avec iCloud ou iTunes (ordinateur) sans connecter l'appareil.

- **Mac OS :**

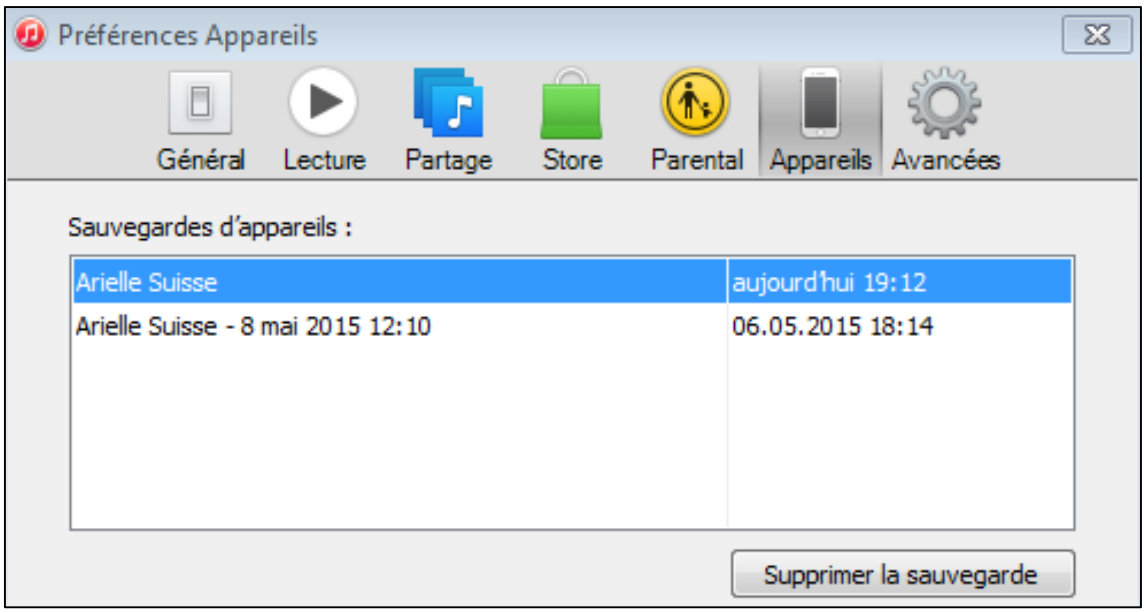
  Cliquer sur iTunes → Préférences →Appareils →Sélectionner la sauvegarde à supprimer →Cliquer sur Supprimer la sauvegarde

- **Windows :**

  Cliquer sur Édition → Préférences →Appareils →Sélectionner la sauvegarde à supprimer →Cliquer sur Supprimer la sauvegarde

Dans la fenêtre des préférences, le nom de l'appareil s'affiche, ainsi que la date et l'heure auxquelles la sauvegarde a été créée.

Figure 13 : Suppression d'une sauvegarde



4.3 Encryption

Le logiciel iTunes permet de chiffrer la sauvegarde que ce soit pour la méthode de sauvegarde iCloud ou ordinateur et sur n'importe quel système d'exploitation, ce qui rend son accès plus compliqué dans le cadre d'une enquête.

Dans le cas où le mot de passe est oublié, il sera difficilement possible de restaurer les données de la sauvegarde.

Figure 14 : Chiffrer une sauvegarde iTunes

5. Suppression de messages

Il y a deux manières de supprimer des messages sur un iPhone. En effet, l'utilisateur peut supprimer manuellement un ou plusieurs messages mais également activer l'option de suppression automatique.

5.1 Manuelle

5.1.1 Message particulier

Sur les iPhones, Apple nous offre la possibilité de supprimer un ou plusieurs messages (SMS ou iMessages) de notre conversation (voir Figure 15). Pour cela, il faut simplement sélectionner la conversation, puis sélectionner (clic long) le message à supprimer, des options apparaissent :

- **Copier :** Permettant de copier le message
- **Option :** Permettant de sélectionner le ou les messages à supprimer (comme sur l'image ci-dessous). En cliquant sur la petite corbeille, une confirmation de la suppression du ou des messages nous sera demandée.

Figure 15 : Suppression d'un ou plusieurs message(s)

5.1.2 Conversation

Dans cette méthode, il suffit de sélectionner la ou les conversations à supprimer, puis cliquer sur le bouton rouge « supprimer ».

Figure 16 : Suppression d'une ou plusieurs conversations

5.2 Automatique

Comme nous l'avons évoqué précédemment, il est possible de supprimer automatiquement les messages en activant l'option de suppression.

Pour cela, il faut se rendre dans les réglages, sélectionner messages puis se rendre dans la partie « Historique des messages », on se retrouve sur l'option « garder les messages ».

Il est possible de choisir entre 3 possibilités : supprimer les messages de plus de 30 jours ou de plus d'un an ou choisir de les garder indéfiniment.

Figure 17 : Suppression automatique

6. Récupération de messages

Avant de présenter les différentes techniques possibles de récupération de messages, il est important d'analyser la structure de la base de données où sont stockés les messages. C'est ce que nous ferons dans un premier temps.

6.1 Structure de stockage des messages

Les messages sont contenus dans différents fichiers de type SQLite sur l'appareil iOS. Il ne s'agit donc pas d'une seule base de données contenant les messages. Il s'agira, dans un premier temps, de déterminer où sont situés ces fichiers et à quoi servent-ils.

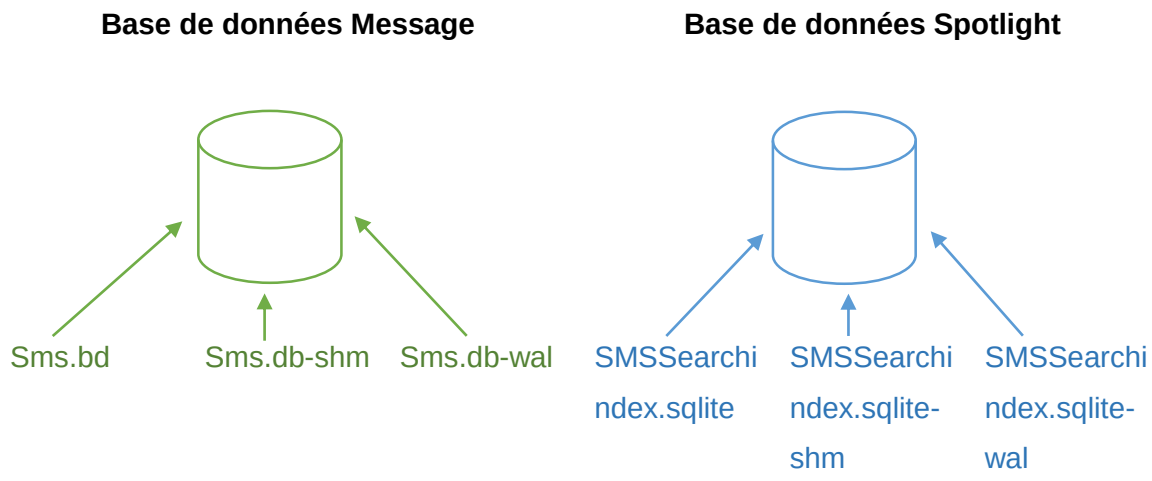
Puis, comment récupérer ces fichiers.

Les messages sont contenus dans les fichiers suivants :

- Sms.bd
- Sms.db-shm
- Sms.db-wal
- SMSSearchindex.sqlite
- SMSSearchindex.sqlite-shm
- SMSSearchindex.sqlite-wal

La Figure 18 permet de visualiser les bases de données ainsi que les fichiers qui les composent.

Figure 18 : Bases de données contenants des messages



(D'après les explications de SQLite, The SQLite Database File Format)

6.1.1 Fichier « sms.db »

Sur iOS, le fichier principal contenant les messages s'appelle « sms.db ».

La structure du fichier a évolué au cours des mises à jour qu'Apple a effectué sur iOS. En effet, auparavant il excitait dans la table Message un attribut nommé « flags ». Il s'agissait d'un entier permettant de connaitre l'état du message (lorsque l'enregistrement était supprimé la valeur de l'attribut flag était à 129).

Le fichier « sms.db » que nous allons étudier se base sur la version 7.1.2 d'iOS.

De plus, dans le cadre de ce travail, une comparaison avec la version 8.3 de la structure du fichier « sms.db » a été effectuée. Cette comparaison ne montre aucune différence.

Nous allons commencer par analyser la structure du fichier.

6.1.1.1 Structure du fichier « sms.db »

_**SqliteDatabaseProperties :** La table contient deux colonnes : key et value, qui contiennent des informations générales de configuration telles que :

counter_out_all	:	Le nombre de messages sortants depuis la dernière réinitialisation du compteur
counter_out_lifetime	:	Le nombre de messages sortants depuis toujours
counter_in_all	:	Le nombre de messages entrants depuis la dernière réinitialisation du compteur
counter_in_lifetime	:	Le nombre de messages entrants depuis toujours

Attachement : La table comporte des informations relatives aux pièces jointes envoyées ou reçues.

Tableau 1 : Dictionnaire de données de la table Attachement

Nom de la colonne	**Description**
ROWID	Clé primaire de la table Attachement
Guid	Correspond au nom du sous dossier dans le dossier Attachements
Create_date	La valeur est un entier correspondant à la date de création
Strart_date	Correspond à la date de début quand il y a des vidéos. La valeur est à 0 lorsqu'il n'y a rien
Filename	Correspond au nom du fichier complet (avec le chemin)
Uti	La valeur est soit « public.jpeg » soit « public.vcard » soit « public.3gpp » soit « com.apple.quicktime-movie »
Mime_type	La valeur est soit « image/jpeg » soit « image/png » soit « text/vcard » soit « text/x-vCard » soit « video/3gpp » soit video/quicktime »
Transfer_state	La valeur est toujours 5
Is_outgoing	S'il s'agit d'une pièce jointe reçue la valeur est à 0. Pour une pièce jointe envoyée, la valeur est à 1
Transfer_name	Il s'agit du nom de la pièce jointe
Total_bytes	Il s'agit de la taille de la pièce jointe

Chat : Cette table correspond aux conversations. Chaque conversation est unique et comprend un ou plusieurs messages (voir table Tableau 7).

Tableau 2 : Dictionnaire de données de la table Chat

Nom de la colonne	**Description**
ROWID	Clé primaire de la table Chat
Guid	Il s'agit de la même valeur que « chat_identifier » de cette même table, mais avec le type de service (SMS ou iMessage) suivi d'un '-' (par exemple : iMessage ;- ; exemple@outlook.com)
Style	Les valeurs connues sont 45 ou 43 lorsque le message est vide (valeur de la colonne « text » dans la table Message à NULL) ou qu'il s'agit de messages automatique (par exemple : répondeur vous informant qu'un correspondant à chercher à vous joindre)
State	Les valeurs connues sont 3 ou 2 lorsque la valeur de la colonne account_id dans la table chat est à NULL
Account_id	Correspond à la valeur de la colonne « account_id » de la table Message. Sa valeur est à NULL lorsque la valeur de la colonne « state » dans la table chat est à 2
Properties	La valeur est à NULL pour la deuxième rangée ou une bplist

Chat_identifier	Permet d'obtenir l'adresse mail, un identifiant ou le numéro duquel l'iMessage ou SMS est envoyé ou reçu
Service_name	Permet d'obtenir le type de Message : iMessages ou SMS. Cette colonne est importante car une ligne (dans la table chat) est créée en fonction du type de service même s'il s'agit du même numéro. C'est-à-dire que pour un numéro avec lequel une conversation est créée, si les messages sont envoyés par le service iMessage, un enregistrement sera créé dans la base de données dans la table chat et si un message de la même discussion est envoyé via le service SMS, une autre ligne sera ajoutée. On peut penser qu'il s'agit de deux discussions distinctes alors qu'il s'agit d'une même discussion utilisant deux modes de transmissions différents
Room_name	La valeur toujours à NULL
Account_login	La valeur est soit 'P:' suivie du numéro sur lequel le message est envoyé soit E : soit NULL
Is_archived	La valeur est toujours à 0
Last_addressed_handle	La valeur correspond au numéro sur lequel le message est envoyé ou NULL
Display_name	La valeur est toujours à NULL

Chat_handle_join : Cette table est une table d'association entre la table Chat et la table Handle.

Tableau 3 : Dictionnaire de données de la table Chat_handle_join

Nom de la colonne	**Description**
Chat_id	Clé étrangère de la table Chat.ROWID
Handle_id	Clé étrangère de la table Handle.ROWID

Chat_message_join : Cette table est une table d'association entre la table Chat et la table Message.

Tableau 4 : Dictionnaire de données de la table Chat_message_join

Nom de la colonne	**Description**
Chat_id	Clé étrangère de la table Chat.ROWID
Message_id	Clé étrangère de la table Message.ROWID

Handle : Cette table fournit un identifiant unique pour chaque numéro de téléphone avec lequel une conversation est créée. Les colonnes ROWID et id sont importantes.

Tableau 5 : Dictionnaire de données de la table Handle

Nom de la colonne	**Description**
ROWID	Correspond à l'identifiant unique pour un numéro avec lequel une conversation est créée
Id	Correspond au numéro de téléphone associé à une discussion (Chat)
Country	Code ISO du pays, par exemple: 'ch' pour la Suisse
Service	Permet d'avoir le type de messages : iMessages ou SMS
Uncanonicalized_id	Identifiant du numéro

Message_attachement_join : il s'agit d'une table d'association faisant la liaison entre la table Message et la table Attachement**.**

Tableau 6 : Dictionnaire de données de la table Message_attachement_join

Nom de la colonne	**Description**
Message_id	Clé étrangère de la table Message.ROWID
Attachement_id	Clé étrangère de la table Attachement.ROWID

Message : Il s'agit de la table où le contenu des messages est stocké. En effet, cette table contient l'ensemble des messages effacés ou encore présents sur le téléphone.

Tableau 7 : Dictionnaire de données de la table Message

Nom de la colonne	**Description**
ROWID	Clé primaire du message
Guid	Il s'agit de la même valeur que le Guid dans la table Attachement
Text	Contenu du message reçu ou envoyé
Replace	La valeur est toujours à 0
Service_center	La valeur est toujours à 0 à NULL
Handle_id	Correspond au ROWID de la table Handle, nous permettant ainsi de faire la correspondance avec le numéro. La valeur est à NULL si ce n'est ni un iMessage ni un numéro de téléphone d'une personne (expéditeur ou destinataire)
Subject	Il s'agit du sujet d'un message
Country	Code ISO du pays (par exemple: 'ch' pour la Suisse) valeur ou NULL
Version	Valeur inconnue
Type	La valeur est toujours à 0 ou à 1
Service	Type de message (SMS ou iMessage)
Account	Téléphone ou e-mail enregistré avec iMessage, vide s'il ne s'agit pas d'un iMessage
Account_guid	GUID du compte utilisé

Date	Date du message en format entier, pour convertir la date en format date il faut faire une requête SQL
Date_read	Valeur entière représentant la date de lecture. La valeur est à NULL si le message est envoyé depuis un appareil ayant une version iOS antérieure à la 5.0. La valeur est 0 s'il s'agit d'un SMS ou un iMessage reçu
Date_delivered	NULL si le message est pré-iOS 5.0, 0 si SMS ou iMessage, une valeur entière reçue représentant la date envoyée
Is_delivered	Valeur correspondante à l'envoi ou non d'un iMessage. La valeur est à 0 si l'iMessage n'est pas livré et 1 dans le cas contraire

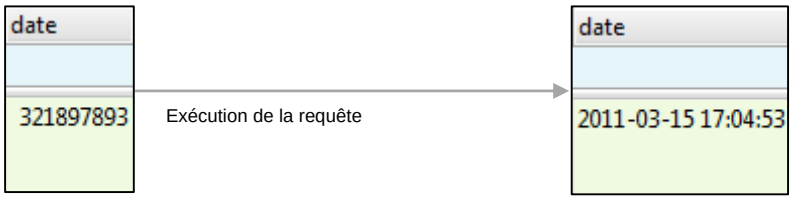
De nombreux autres attributs sont présents tels que : *is_read, is_prepared, is_sent* etc. en fonction de ce que l'on cherche à récupérer.

Requête SQL de conversion d'une date de format entier en format date :

Voici un script SQL qui permet de traduire les valeurs de la colonne Date (qui sont cryptées), voici le script SQL :

« SELECT datetime (date + strftime ('%s', '2001-01-01 00:00:00'), 'unixepoch', 'localtime') AS date, * FROM message »

Figure 19 : Résultat de la requête de conversion d'une date



6.1.1.2 Triggers affectés au fichier « sms.db »

Certains triggers ont été définis sur le fichier « sms.db ». Ces déclencheurs agissent sur différentes tables de la base de données. De plus, le fichier « sms.db » est mis à jour après un certain nombre de suppressions de messages ou conversations.

Table Attachement :

- *delete_attachment_files :* Après suppression.

  « CREATE TRIGGER delete_attachment_files AFTER DELETE ON attachment BEGIN SELECT delete_attachment_path (old.filename); END; »

Table Chat_hangle_join :

- clean_orphaned_handles : Après suppression.

  « CREATE TRIGGER clean_orphaned_handles AFTER DELETE ON chat_handle_join BEGIN DELETE FROM handle WHERE handle.ROWID = old.handle_id AND (SELECT 1 from chat_handle_join WHERE handle_id = old.handle_id LIMIT 1) IS NULL AND (SELECT 1 from message WHERE handle_id = old.handle_id LIMIT 1) IS NULL; END; »

Table Message :

- clean_orphaned_handles2 : Après suppression.

  « CREATE TRIGGER clean_orphaned_handles2 AFTER DELETE ON message BEGIN DELETE FROM handle WHERE handle.ROWID = old.handle_id AND (SELECT 1 from chat_handle_join WHERE handle_id = old.handle_id LIMIT 1) IS NULL AND (SELECT 1 from message WHERE handle_id = old.handle_id LIMIT 1) IS NULL; END; »

Table Chat_message_join :

- update_message_roomname_cache_insert : Après insertion.

  « CREATE TRIGGER update_message_roomname_cache_insert AFTER INSERT ON chat_message_join BEGIN UPDATE message SET cache_roomnames = (SELECT group_concat(c.room_name) FROM chat c INNER JOIN chat_message_join j ON c.ROWID = j.chat_id WHERE j.message_id = new.message_id) WHERE message.ROWID = new.message_id; END; »

- clean_orphaned_messages : Après suppression.

  « CREATE TRIGGER clean_orphaned_messages AFTER DELETE ON chat_message_join BEGIN DELETE FROM message WHERE (SELECT 1 FROM chat_message_join WHERE message_id = message.rowid LIMIT 1) IS NULL; END; »

- update_message_roomname_cache_delete : Après suppression.

  « CREATE TRIGGER update_message_roomname_cache_delete AFTER DELETE ON chat_message_join BEGIN UPDATE message SET cache_roomnames = (SELECT group_concat(c.room_name) FROM chat c INNER JOIN chat_message_join j ON c.ROWID = j.chat_id WHERE j.message_id = old.message_id) WHERE message.ROWID = old.message_id; END; »

Table Message_attachement_join :

- set_message_has_attachments : Après insertion.

  « CREATE TRIGGER set_message_has_attachments AFTER INSERT ON message_attachment_join BEGIN UPDATE message SET cache_has_attachments = 1 WHERE message.ROWID = new.message_id; END; »

- clear_message_has_attachments : Après suppression.

  « CREATE TRIGGER clear_message_has_attachments AFTER DELETE ON message_attachment_join BEGIN UPDATE message SET cache_has_attachments = 0 WHERE message.ROWID = old.message_id AND (SELECT 1 from message_attachment_join WHERE message_id = old.message_id LIMIT 1) IS NULL; END; »

- clean_orphaned_attachments : Après suppression.

  « CREATE TRIGGER clean_orphaned_attachments AFTER DELETE ON message_attachment_join BEGIN DELETE FROM attachment WHERE attachment.ROWID = old.attachment_id AND (SELECT 1 from message_attachment_join WHERE attachment_id = old.attachment_id LIMIT 1) IS NULL; END; »

6.1.1.3 Visualisation du fichier « sms.db »

L'outil « DBeaver Entreprise générale » permet de visualiser les tables contenues dans le fichier **3d0d7e5fb2ce288813306e4d4636395e047a3d28**, récupéré à partir de la sauvegarde iTunes.

Pour ce faire il faut :

- Télécharger l'outil[1]
- Lancer l'outil
- Créer une nouvelle connexion : Database → new connection
- Spécifier qu'il s'agit d'une base SQLite
- Tester la connexion
- Valider

Après ce processus effectué, il faut faire un clic droit sur « Tables », puis sélectionner « View Tables ».

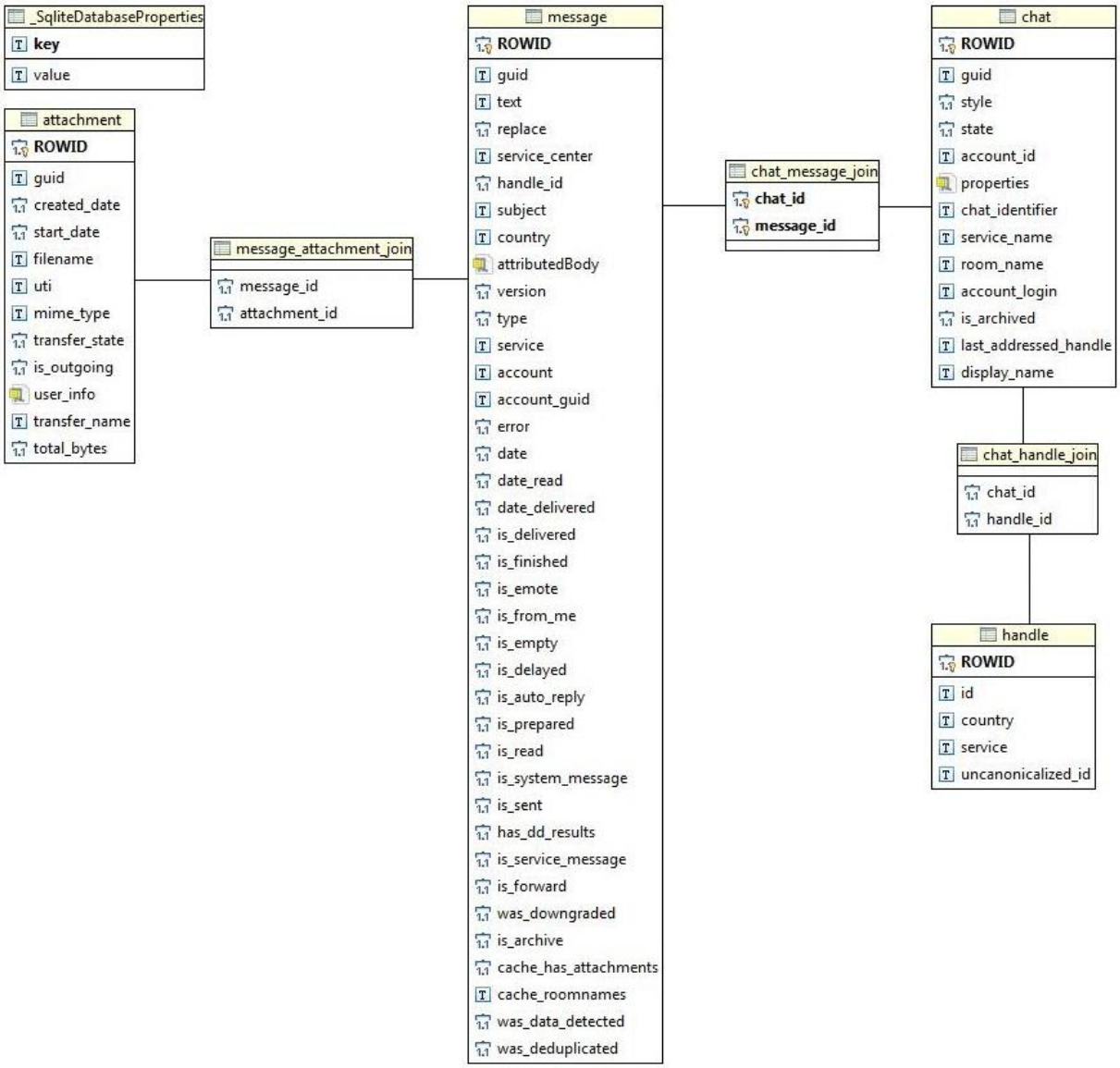
Un diagramme s'affichera avec les tables de la base de données et les associations correspondantes.

En revanche il n'est pas dit que les associations soient justes et complètes. C'est pour cela qu'il faut les ajuster.

[1] L'outil est disponible à l'adresse suivante : http://dbeaver.jkiss.org/download/#windows

Après analyses et corrections des associations, la base de données peut être visualiser comme sur la Figure 20.

Figure 20 : Diagramme de classe de la base de données « sms.db »



6.1.2 Autres fichiers

Sur l'appareil, dans le dossier « /var/mobile/Library/SMS/ », se trouve également d'autres fichiers et dossiers. Les fichiers « sms.db-shm » et « sms.db-wal » peuvent contenir des messages supprimés. Les fichiers « sms.db-shm » et « sms.db-wal » sont utilisés pour l'indexation ou la recherche et peuvent être recréés dans le cas où ils seraient supprimés.

Si la base de données est mise à jour manuellement, il est important de recréer ces fichiers.

6.1.2.1 Fichier « sms.db-wal »

Il s'agit d'un fichier temporaire créé par SQLite à partir de la base principale. Le fichier est situé dans le même répertoire que le fichier « sms.db ». Le –wal signifie « Write Ahead Log » (en référence à l'explication : Gestion des erreurs).

Ce fichier est responsable du comportement transactionnel. En effet, il contient les dernières modifications et sera mis à jour dès qu'il y a une modification de la base de données des messages (par exemple : la suppression d'un SMS) avant même que la base de données principale soit mise à jour.

Si ce fichier est supprimé, il sera automatiquement recréé lors de la création de la base « sms.db » (base principale).

6.1.2.2 Fichier « sms.db-shm »

Il s'agit d'un fichier temporaire créé par SQLite à partir de la base principale. Le fichier est situé dans le même répertoire que le fichier « sms.db ». Le –shm signifie « Shared Memory ».

Si ce fichier est supprimé, il sera automatiquement recréé lors de la création de la base « sms.db » (base principale).

6.1.3 Autres dossiers

6.1.3.1 Dossier Attachments et Parts

Les dossiers « Attachments » et « Parts » contiennent des sous-dossiers comportant les pièces jointes reçues et envoyées depuis l'appareil.

Le nom d'un sous-dossier est court : il est composé soit de deux chiffres, soit d'un chiffre et une lettre soit de deux lettres.

Le nom de la pièce jointe peut être retrouvé à partir de la base de données dans la table « Attachement ». En effet, l'attribut « Filename » définit où se trouve le chemin d'accès à la pièce jointe dans la base.

Comme pièce jointe nous pouvons retrouver des contacts (nomDuContact.vcf), des photos (.jpeg, .png), ou encore des vidéos (-movie).

6.1.3.2 Dossier Drafts

Le dossier « Draft » contient les messages qui ont été saisis puis abandonnés. C'est ce que l'on appelle un « message brouillon ». Le message n'a pas été envoyé mais il est stocké en tant que « message brouillon ». Ainsi, au moment où l'application Message est lancée à nouveau, on retrouve le message saisi, et ce, même si l'on ferme l'application.

Chaque « message brouillon » est contenu dans un dossier propre. Le contenu textuel du message ainsi que des informations supplémentaires concernant ce dernier, sont présents dans le fichier « message.plist » situé dans le dossier lié au brouillon.

6.1.3.3 Dossier EmergencyAlerts

Le dossier « EmergencyAlerts » contient un fichier « PriorAlerts.plist ». Le fichier .plist contient des données et des informations relatives à l'application Message.

6.1.4 Spotlight

6.1.4.1 Généralités

Spotlight est un moteur de recherche de fichiers intégré à iOS. Ce dernier permet à l'utilisateur de rechercher du contenu sur l'iPhone (messages, articles etc.) en saisissant ce qu'il souhaite rechercher dans le champ approprié.

Pour accéder à cette fonctionnalité il suffit de faire glisser le doigt de haut en bas sur l'écran principal pour afficher le champ de recherche (voir Figure 21).

Figure 21 : Recherche de messages via Spotlight

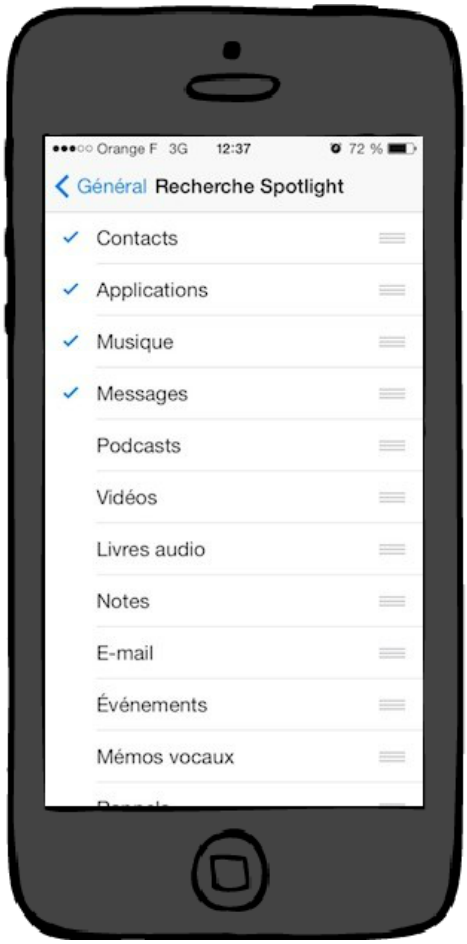
Spotlight possède un fonctionnement bien particulier, le moteur de recherche analyse le contenu présent sur l'appareil et y recherche du texte commençant par le terme saisi dans la barre de recherche. Spotlight ne respecte pas la casse, donc qu'il s'agisse de minuscule ou majuscule, les résultats de la recherche seront identiques.

Concernant la recherche de messages, il s'agit du point qui nous intéresse ici, Spotlight recherche du texte spécifique contenu dans les messages et dans le nom ou encore dans le numéro de l'expéditeur.

Attention, il est possible que les résultats n'affichent aucun message. Si tel est le cas, il faut aller dans la configuration de Spotlight sur l'appareil (voir Figure 22) et vérifier si les messages ont été sélectionnés (prise en compte des messages par le moteur de recherche).

Aller sur l'application Réglage→Général→Recherche Spotlight

Figure 22 : Configuration de Spotlight



6.1.4.2 Fonctionnement interne

A chaque fois que du contenu est ajouté sur l'appareil (contacts, messages, etc.), qu'une recherche sur Safari est effectuée ou encore lors de la réception d'un nouvel email, Spotlight indexe le contenu (par exemple : lors de la réception d'un message, voir Figure 23).

Figure 23 : Mécanisme d'indexation de Spotlight

(D'après les explications de Wikipédia, Spotlight)

Ainsi, dès qu'une recherche est effectuée via la barre de recherche Spotlight, le moteur retrouve le contenu désiré. Par défaut Spotlight indexe tout (par exemple : tous les mots des mails, dans les messages via les applications inhérentes à l'appareil : Mail et Message). Le moteur de recherche indexe aussi bien les iMessage que les SMS.

Alors malgré la suppression de certains messages et/ou conversations, Spotlight, étant une base de données indépendante, les conserves. Mais lorsque l'on essaie d'accéder au message complet, c'est impossible. Ce dernier étant bel et bien effacé de la liste des messages. En revanche l'index quant à lui est toujours présent.

Si on désactive l'indexation, Spotlight conserve en mémoire les anciens messages auxquels nous pourrons toujours accéder.

Spotlight utilise une base de données SQLite comme les autres fichiers que nous avons analysés.

6.1.4.3 Fichier « SMSSearchindex.sqlite »

La base de données Spotlight est composée du fichier principal « SMSSearchindex.sqlite » et de deux autres fichiers liés (-wal et –shm) comme pour la base de données « sms.db »

Le chemin permettant d'accéder au fichier directement sur l'appareil est le suivant :

« var/mobile/library/spotlight/com.apple.MobileSMS/SMSSearchindex.sqlite »

La base de données « SMSSearchindex.sqlite » est composée de quatre tables :

Z_METADATA : Cette table contient un seul enregistrement. Cette table est utilisée par un framework et possède la configuration des données de la base.

Z_PRIMARYKEX : Cette table est utilisée par un framework et possède un enregistrement par table. Pour cette base, il y a donc deux enregistrements pour les tables ZSPRECORD et ZSPTOPHIT.

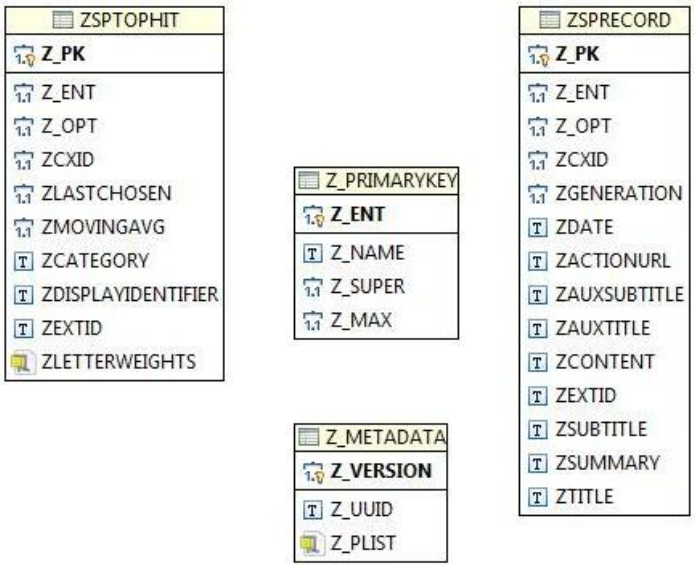
ZSPRECORD : Cette table contient les métadonnées indexées. Dans cette table nous pouvons retrouver des enregistrements supprimés.

ZSPTOPHIT : Cette table était vide au moment des tests.

6.1.4.4 Visualisation du fichier « SMSSearchindex.sqlite »

Voici le diagramme de classe nous permettant de visualiser la base de données du fichier « SMSSearchindex.sqlite ».

Figure 24 : Diagramme de classe de la base de données « SMSSearchindex.sqlite »



6.2 Techniques de récupération des fichiers

Avant toute chose, nous allons commencer par récupérer les messages contenus dans les différents fichiers. Bien que le format de base de données SQLite soit dans l'ensemble similaire, les techniques de récupération des fichiers sur un smartphone varient en fonction de différents paramètres (par exemple : sécurité placée sur le smartphone, appareil jailbreaké, version du système d'exploitation etc.). Il faut donc prendre en compte ces variations afin de pouvoir reconstruire les données effacées.

6.2.1 Récupération du fichier « sms.db »

A présent, analysons les différentes techniques de récupération du fichier « sms.db ». Ce dernier peut se récupérer par différentes techniques :

- Par le biais d'une sauvegarde iTunes
- Physiquement sur l'iPhone

6.2.1.1 Depuis l'ordinateur avec une sauvegarde iTunes

Cette technique s'applique à n'importe quel type d'iPhone. Ici, il n'est pas obligatoire d'avoir un téléphone. En effet, si l'on récupère un ordinateur et que la sauvegarde s'y trouve, il est possible de l'exploiter.

Dans le cas contraire, il suffit d'effectuer une sauvegarde de l'appareil. Il s'agit d'une des possibilités de récupération via le logiciel iTunes (voir Figure 11). En effet, la sauvegarde iTunes contient le fichier dont le nom est :

3d0d7e5fb2ce288813306e4d4636395e047a3d28

Ce dernier est contenu dans le dossier de sauvegarde iTunes (voir Figure 12).

Petit rappel, selon le système d'exploitation utilisé ce fichier est situé dans un répertoire différent.

- **Sur MAC OS :**

  « /Users/USERNAME/Library/Application Support/ MobileSync/ Backup/ NOMDOSSIER »

- **Sur Windows :**

  « /Users/USERNAME/AppData/Roaming/Apple Computer/MobileSync/Backup/ NOMDOSSIER »

Une fois le fichier récupéré, il faut le lire. Différents outils sont disponibles pour rendre la lecture de ce fichier facilement compréhensible. En effet, certains sites tels que « smsIphone », proposent de transformer le fichier de sauvegarde, non lisible, en format PDF, csv etc. Mais il existe également des logiciels tels que « SQLite Expert Personal », proposant d'importer le ficher de sauvegarde afin de le rendre lisible pour l'utilisateur. Nous utiliserons cet outil lors de la récupération de fichiers. Evidemment quel que soit l'outil de visualisation de requêtes, les mêmes tables seront créées, le fichier contenant un ensemble de commandes SQL.

Comme nous l'avons évoqué précédemment, « SQLite Expert Personal » est un outil permettant une meilleure visualisation des données contenues dans le fichier de sauvegarde. Nous allons à présent procéder à la visualisation des messages.

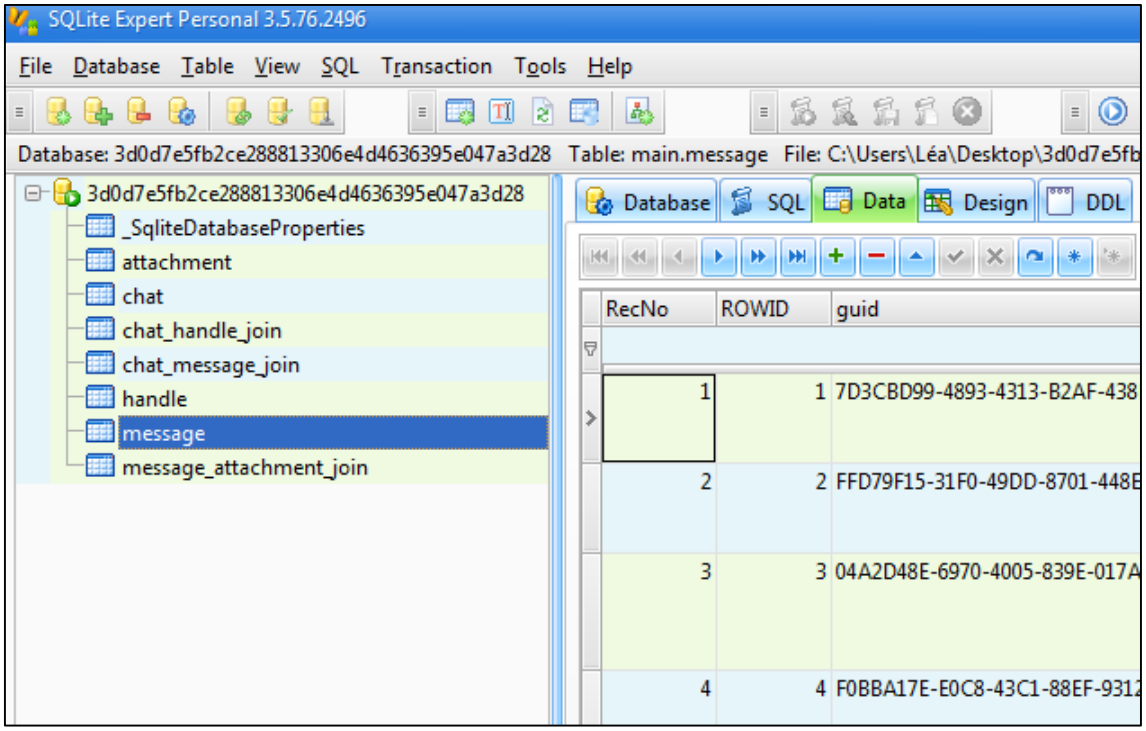
Démarche :

- Télécharger l'outil SQLite Expert Personal[2]
- Cliquer sur File → open database
- Rechercher le fichier **3d0d7e5fb2ce288813306e4d4636395e047a3d28**
- Cliquer sur ouvrir
- La base de données se crée automatiquement

[2] L'outil est disponible à l'adresse suivante : http://www.sqliteexpert.com/download.html

Nous obtenons une hiérarchie comme celle présente sur la Figure 25, une base de données avec des tables. Sur l'outil, les données sont situées dans la colonne « Data » de chaque table.

Figure 25 : Visualisation du fichier de sauvegarde avec SQLite Expert Personal



6.2.1.2 Depuis un iPhone jailbreaké

Afin de récupérer le fichier sur le téléphone directement, il faut qu'il soit jailbreaké.

Le jailbreak est un processus par lequel le système d'exploitation iOS est modifié pour exécuter du code non signé dans le but d'accéder à des fichiers auxquels Apple ne nous laisse pas accéder nativement.

A travers ces modifications, nous pouvons installer des applications non officielles sur l'iPhone par le biais de Cydia[3].

Pour accéder au fichier « sms.db », il faut que l'iPhone soit jailbreaké comme nous l'avons dit, deux manières d'y accéder sont possibles :

- **Par le biais d'un tweak**
- **En ligne de commande par le biais d'une connexion SSH**
- **Via un programme en utilisant une connexion SSH**

[3] Cydia est une application non officielle pour iOS conçue par Jay Ryan Freeman, en Open Source, et donnant accès à des applications non signées numériquement par Apple (Wikipédia)

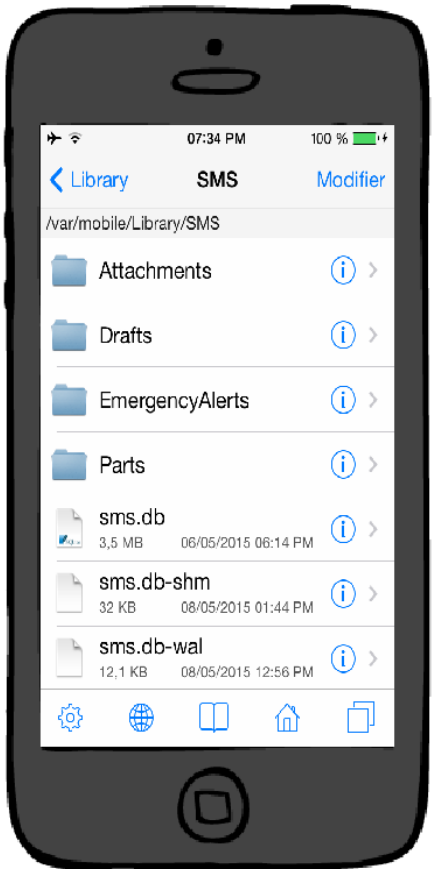
6.2.1.2.1 Par le biais d'un tweak[4]

Le paquet iFile disponible sur le store Cydia permet l'accès au fichier « sms.db ». Il suffit de se rendre sur Cydia, dans rechercher, saisir iFile, installer le tweak.

Après l'installation, lancer iFile puis aller dans « var/mobile/Library/SMS »

La Figure 26 présente le fichier « sms.db » comme nous pouvons l'observer.

Figure 26 : Base de données via iFile



[4] Tweak : paquet Cydia qui apporte une modification au système iOS (ex : thème ou modification du système)

6.2.1.2.2 En ligne de commande par le biais d'une connexion SSH

SSH est un protocole de communication permettant d'accéder physiquement à l'iPhone. Pour accéder au fichier, il faut établir une connexion SSH.

Se connecter en SSH (Secure Shell) sur l'iPhone :

Phase d'installation

Sur l'iPhone il faut :

- Se connecter au réseau sans fil (Wifi)
- Désactiver le verrouillage automatique : Il faut que l'iPhone soit constamment allumé, il faut donc désactiver le verrouillage automatique
- Installer le tweak OpenSSH. Par le biais de l'application Cydia, il faut aller dans Rechercher et saisir OpenSSH
- Activer SSH en sélectionnant l'icône SSH (l'icône devient vert une fois activé), voir Figure 27

Figure 27 : Activation de SSH sur iPhone jailbreaké

- Installer le tweak SBSettings : Par le biais de l'application Cydia, il faut aller dans Rechercher et saisir SBSettings. Après installation, relancer le SpringBoard.
- Lancer SBSettings et activer l'option SSH

Sur l'ordinateur il faut :

Téléchargez et installer un client SSH sur ordinateur : PuTTY sur PC ou Fugu sur Mac

Phase de connexion

Dans cette phase il s'agit d'établir la connexion entre l'iPhone et l'ordinateur.

Sur l'iPhone :

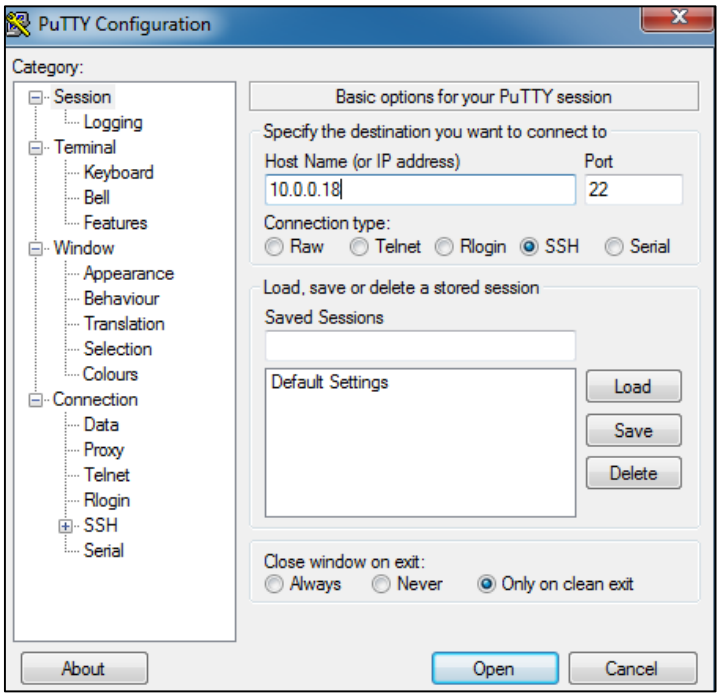
- Se rendre dans Réglages de l'iPhone → Wifi → Sélectionner le réseau sur lequel l'appareil est connecté
- Noter l'adresse IP (par exemple : 10.0.0.18)

L'adresse est également disponible directement depuis le gestionnaire présent dans la Figure 27 (il s'agit de la « Wi-Fi IP Address »).

Sur l'ordinateur :

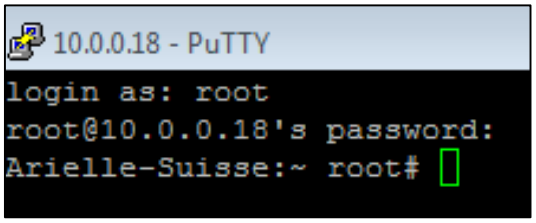
- Ouvrir le logiciel SSH (PuTTY)
- Saisir l'adresse IP puis cliquer sur « open » afin d'ouvrir la connexion comme sur la Figure 28.

Figure 28 : Fenêtre de configuration de PuTTY



Une fenêtre DOS s'ouvre comme sur la Figure 29, saisir « root » comme nom de login (« login as ») et « alpine » comme mot de passe (« password »).

Figure 29 : Fenêtre DOS de PuTTY



Par mesure de sécurité, il faut changer le mot de passe de connexion. Pour ce faire il faut avoir une connexion SSH établie comme expliqué au-dessus. Une fois la connexion établie, saisir dans la fenêtre DOS de PuTTY (celle de la Figure 29) la commande « passwd », un nouveau mot de passe est demandé (minimum 5 caractères). Le valider en cliquant sur la touche « entrée » du clavier puis confirmer le mot de passe et valider par la touche « entrée » du clavier.

Le fichier « sms.db » se trouve dans le dossier SMS (« /var/mobile/Library/SMS/ »).

On accède au dossier à travers les commandes linux standards. Pour se déplacer utiliser la commande « cd » suivie du chemin.

Copier les fichiers à travers une interface en ligne de commande n'était pas toujours évident, c'est pour cela que je me suis tournée vers un autre outil avec une interface utilisateur. Il s'agit de FileZilla, qui requiert également l'utilisation du protocole SSH.

6.2.1.2.3 Via un programme en utilisant une connexion SSH

Une manière plus facile de récupérer les fichiers de l'iPhone est FileZilla.

FileZilla est un client FTP possédant plusieurs fonctionnalités. Ce logiciel permet notamment de gérer des fichiers, dossiers via une interface utilisateur par le biais d'une connexion SSH. A travers ce logiciel, nous pouvons récupérer différents fichiers et éléments présents sur l'appareil tels que les fichiers de base de données des messages, les pièces jointes et de nombreux autres éléments.

Pour l'utilisation de ce programme il faut accomplir quelques manipulations :

Phase d'installation

Sur l'iPhone il faut :

Effectuer les mêmes manipulations que dans le point « En ligne de commande par le biais d'une connexion SSH »

Sur l'ordinateur il faut :

Télécharger et installer le logiciel disponible sur le site internet de l'éditeur

Phase de connexion

Dans cette phase il s'agit d'établir la connexion entre l'iPhone et l'ordinateur afin de pouvoir récupérer les éléments voulus.

Sur l'iPhone :

- Se rendre dans Réglages de l'iPhone → Wifi → Sélectionner le réseau sur lequel l'appareil est connecté
- Noter l'adresse IP (par exemple : 10.0.0.20)

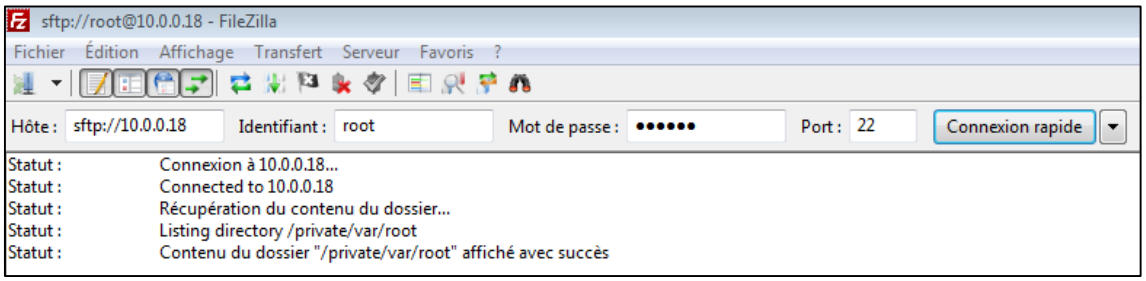
L'adresse est également disponible directement depuis le gestionnaire présent dans la Figure 27 (il s'agit de la « Wi-Fi IP Address »).

Sur l'ordinateur :

- Ouvrir le logiciel FileZilla
- Saisir les informations suivantes :
    - Hôte : *adresse IP*
    - Identifiant : *root*
    - Mot de passe : *alpine*
    - Port : *22*
- Cliquer sur « Connexion rapide » afin d'ouvrir la connexion.

L’interface du logiciel se présente comme sur la Figure 30 :

Figure 30 : Logiciel FileZilla



6.2.2 Récupération des autres fichiers et dossiers

Comme nous l’avons analysé précédemment les autres fichiers et dossier intéressants dans le cas d’une récupération de messages sont :

- Sms.db-shm
- Sms.db-wal
- SMSSearchindex.sqlite
- SMSSearchindex.sqlite-shm
- SMSSearchindex.sqlite-wal

Le logiciel FileZilla (voir Figure 30) permet de récupérer ces différents fichiers présents sur l’appareil.

7. Procédés disponibles sur iTunes

Le logiciel iTunes permet à l'utilisateur d'effectuer soit une sauvegarde de l'appareil (voir précédemment :Sauvegarde) soit une synchronisation de celui-ci. Il s'agit de deux processus bien distincts.

7.1 Sauvegarde iTunes

La sauvegarde est une copie des données du téléphone. Ainsi, les fichiers de données présents sur l'appareil sont copiés dans un fichier de sauvegarde.

Lors du processus l'appareil n'est, en aucun cas, impacté.

7.2 Synchronisation

La synchronisation iTunes est une opération permettant de rendre identiques les contenus des éléments synchronisés entre l'ordinateur et l'appareil.

Cette opération peut se réaliser en branchant l'appareil à l'ordinateur (par câble USB, de façon manuelle) ou par le biais d'une connexion Wifi (de façon automatique) via le logiciel iTunes.

Par le biais de cette fonctionnalité, l'utilisateur pourra transférer des musiques, films, photos etc. depuis l'ordinateur vers l'iPhone. En effet, le logiciel iTunes (disponible sur plusieurs systèmes d'exploitations) nous permet d'effectuer des synchronisations sélectives (nous choisissons ce que nous souhaitons synchroniser).

Il n'est pas requis de disposer d'un compte iCloud pour effectuer une synchronisation. Sur le logiciel iTunes, il suffit de sélectionner et synchroniser chaque média dont on souhaite une synchronisation (par exemple : synchroniser la musique).

La synchronisation ne réalise en aucun cas une sauvegarde de l'appareil. Par conséquent, si l'on souhaite conserver les données présentes sur l'iPhone, il faut sauvegarder régulièrement les données qui sont synchronisées. Concernant la première synchronisation, il faut donc effectuer une sauvegarde.

Lors du processus le contenu de l'appareil est modifié.

8. Restauration

La restauration sur iOS est un processus qui permet de remettre l'appareil à son état d'origine ou à l'état d'une sauvegarde existante. Dans le cadre de la récupération de messages supprimés il est important d'analyser les effets d'un tel processus sur l'appareil.

8.1 Sans reprise d'une sauvegarde existante

Il y a deux manières de remettre l'appareil à l'état d'origine : via le logiciel iTunes ou directement depuis l'appareil.

8.1.1 Via le logiciel iTunes

Il suffit de brancher l'appareil à l'ordinateur, de lancer le logiciel iTunes et de sélectionner « restaurer l'iPhone » en tant que nouvelle appareil.

8.1.2 Via l'appareil

Le terme n'est pas le même, ici il s'agit de réinitialisation des réglages et/ou effacement du contenu de l'appareil.

Pour cela il faut lancer l'application Réglages→ puis aller sur Général→ Réinitialiser (voir Figure 31).

Ainsi, depuis l'iPhone directement, du contenu peut être effacé.

Figure 31 : Réinitialiser le contenu de l'appareil

8.2 Avec reprise d'une sauvegarde existante

Une restauration de l'appareil à partir d'une sauvegarde existante est possible. Qu'il s'agisse d'une sauvegarde iCloud ou local, le procédé reste identique.

Il faut préalablement effectuer une sauvegarde (soit iCloud soit local, sur l'ordinateur), puis, brancher l'appareil à l'ordinateur, lancer le logiciel iTunes et sélectionner « restaurer la sauvegarde ».

9. Exploitation des fichiers récupérés

Précédemment, nous avons récupéré des fichiers tel que le fichier « sms.db », nous avons pu le lire à l'aide de l'outil « SQLite Expert Personal » (voir Figure 25).

En revanche, les données supprimées de l'appareil ne sont pas visibles.

Afin d'exploiter les données des différents fichiers que nous avons récupérés, nous allons utiliser l'outil Oxygen Forensic SQLiteViwer. Il s'agit de la version trial 3.0.0.3 (utilisable 30 jours).

Cet outil permet de récupérer des enregistrements supprimés et d'analyser les blocks contenant des données supprimées.

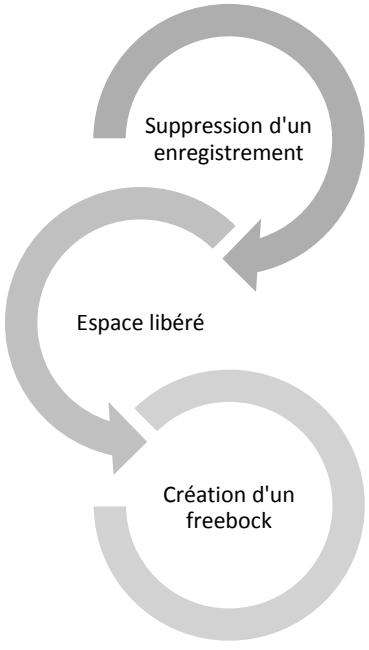
9.1 Mécanisme de suppression d'un enregistrement

Lors de la suppression d'un enregistrement dans la base de données, l'espace alloué qu'occupait le message est libéré. Cet espace est défini comme libre mais le message supprimé est toujours présent.

Ce n'est qu'au moment où il n'y aura plus d'espace disponible que les nouvelles données seront écrites à la place du message supprimé.

Afin de récupérer ce message, il faut donc lire l'espace défini comme libéré. Cet espace correspond aux freeblocks

Figure 32 : Processus de suppression d'un message



(D'après les explications de SQLite, The SQLite Database File Format)

9.2 Structure d’un freeblock

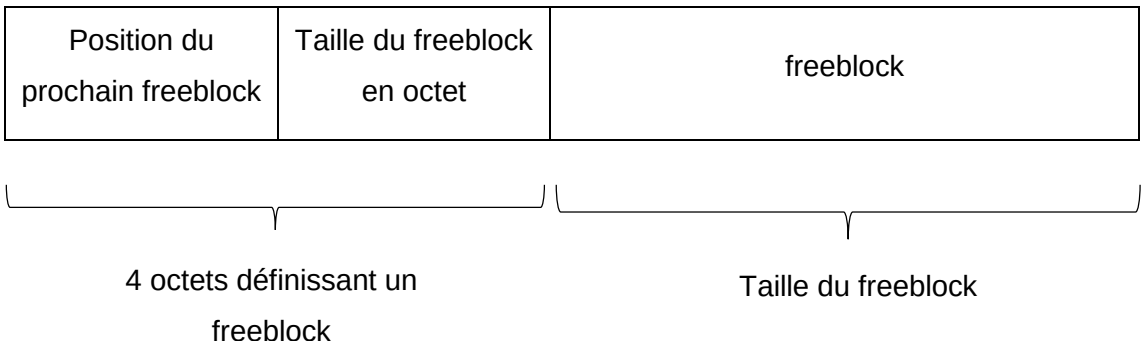
Un freeblock est une structure utilisée pour identifier l'espace non alloué dans une page de type B-Tree. Un freeblock est structuré comme une chaîne et a besoin d’au moins 4 octets d’espace.

Les deux premiers octets d'un freeblock sont des entiers définissant la position du prochain bloc disponible (freeblock). La valeur des deux premiers octets est de 00 00 s’il s’agit du dernier freeblock de la page.

Les freeblocks sont donc liés mais de taille différente.

Les troisième et quatrième octets sont également des nombres entiers définissant la taille en octet du freeblock.

Figure 33 : Structure d’un freeblock



(D’après les explications de SQLite, The SQLite Database File Format)

Il arrive parfois que les freeblocks soient fragmentés, SQLite réorganise donc de temps en temps la page B-Tree.

Dans les tests suivants nous donnerons un exemple de suppression d’enregistrement.

10. Tests effectués

Les tests sont effectués sur le même appareil mais avec des configurations différentes : selon si l'iPhone est jailbreaké ou non. Si l'appareil n'est pas jailbreaké, les données seront récupérées via une sauvegarde iTunes du téléphone comme vu précédemment. Dans le cas contraire, sur le téléphone directement grâce au protocole SSH.

Un test décrit une action que nous effectuons sur l'appareil (par exemple : effacer un SMS reçu, d'une conversation).

A présent, nous allons effectuer différents tests nous permettant d'analyser le fonctionnement lors de la suppression de message et lors de restaurations de l'appareil. Puis, nous noterons les constats que nous avons pu déduire de ces manipulations.

10.1 Analyse des tests après suppressions

Ici, il s'agit d'effectuer des tests de suppressions de messages et de conversations afin d'examiner les données que nous parvenons à récupérer.

Nous détaillerons un des tests effectué afin de comprendre le fonctionnement de la récupération de messages supprimés.

10.1.1 Effacer un SMS envoyé d'une conversation

Pour ce premier test, nous supprimons les SMS sélectionnés sur la Figure 34. Après avoir supprimé ces SMS nous allons récupérer le fichier « sms.db » puis le fichier de sauvegarde de l'appareil afin de tester les deux cas (appareil jailbreaké ou non).

Figure 34 : SMS supprimés pour le test de récupération via recovered records

Une fois les fichiers récupérés, nous utiliserons un outil nous permettant de récupérer ces messages. L'outil que nous utiliserons nous donne la possibilité de récupérer des enregistrements supprimés directement (dans la catégorie « recovered records ») et/ou de les rechercher via la barre de recherche.

En revanche, si on ne trouve pas les messages supprimés, il est possible qu'ils soient contenus dans les blocks disponibles (dans la catégorie « Blocks contening deleted data »)

En effet, lors des tests, j'ai supprimé une conversation. Ne l'ayant pas retrouvé dans les « recovered records », j'ai recherché dans les « Blocks contening deleted data » qui contenaient le dernier message de la conversation.

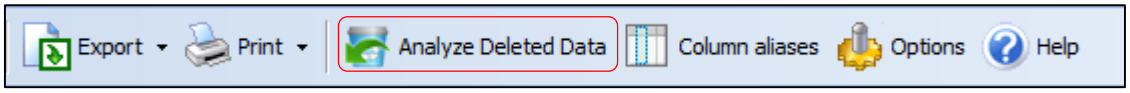
Il arrive parfois qu'un enregistrement supprimé se retrouve dans les deux catégories (« recovered records » et « Blocks contening deleted data »). C'est aussi pour ces raisons que nous allons analyser les deux cas.

10.1.1.1 Enregistrement récupéré via recovered records

Nous devons lancer l'outil « Oxygen Forensic SQLiteViwer » et importons, dans un premier temps le fichier « sms.db » (voir Figure 36).

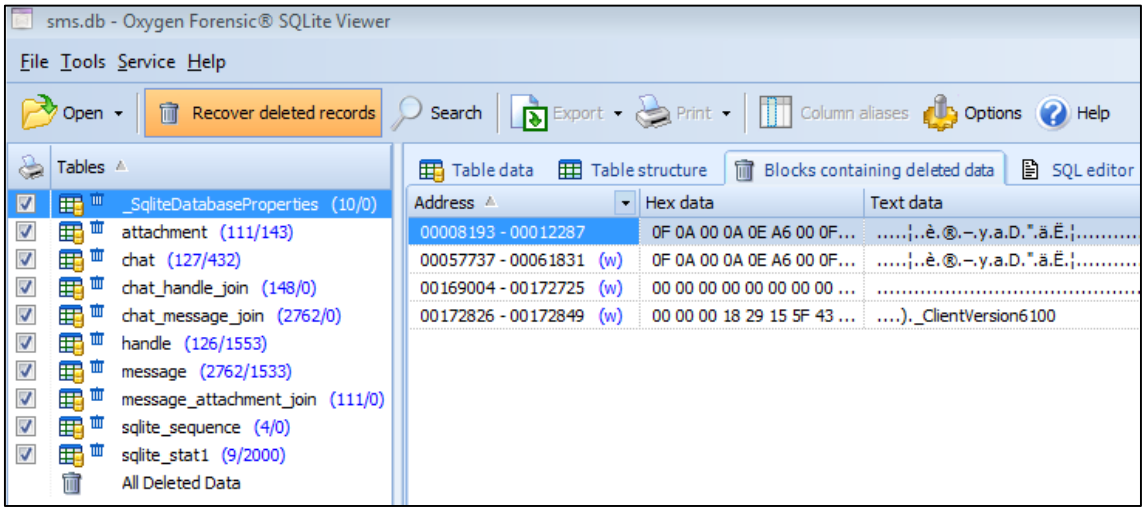
L'outil fait automatiquement l'analyse des blocks supprimés, sauf s'il s'agit d'une base de données importante. Dans ce cas, il faut cliquer sur le bouton *Analyse Deleted Data*.

Figure 35 : Analyse de données pour base de données importante



Après analyse des données supprimées, nous avons accès à de nouveaux éléments tels que la visualisation des blocs contenant les données supprimées, ou encore des enregistrements récupérés.

Figure 36 : Après import du fichier « sms.db » dans Oxygen Forensic SQLiteViwer



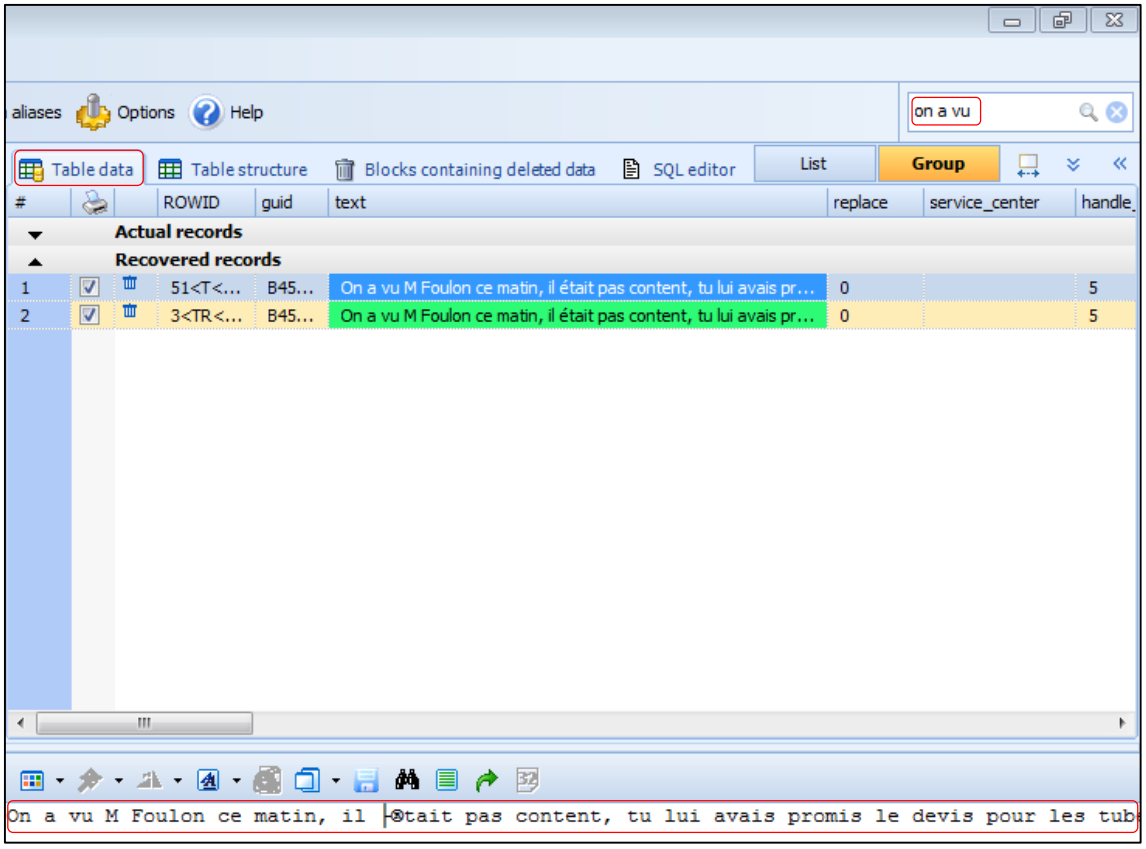
Pour accéder aux enregistrements récupérés il faut :

Nous allons effectuer notre analyse en recherchant un SMS envoyé parmi ceux que nous avons supprimés. En revanche le processus est le même pour les autres messages effacés.

Tout d'abord, il faut sélectionner la table Message (puisqu'il s'agit d'un SMS). Puis sélectionner la colonne « Table Data », puis lancer la recherche (située en haut à droite) du SMS ou parcourir l'ensemble des *Revovered records* sans recherche).

L'entièreté du SMS s'affiche en bas en sélectionnant ce dernier (voir Figure 37).

Figure 37 : Récupération du SMS supprimé



La catégorie *Actual records* contient les enregistrements de données actuelles sur l'appareil.

La catégorie *Revovered records* contient les enregistrements de données récupérées de l'appareil.

10.1.1.2 Enregistrement récupéré via blocks contening deleted data

Ici nous allons analyser en détail les freeblocks afin de récupérer l'enregistrement.

Le SMS sélectionné (en bleu) sur la Figure 38 représente le message que nous allons supprimer.

Figure 38 : SMS supprimé pour le test de récupération via « Blocks contening deleted data »

guid	text	r...	servi...	h...	subj...	country	a...	v...	t...	service	account	account_guid
Click here to define a filter												
9B686794-8E6D-FFFE-754D-16A4664BAB6A	Cher client, vous trouverez des informations et des conseils pour mettre en service votre téléphone portable à l'adresse www.swisscom.ch/configuration-portable	0	(null)	1	(null)	(null)		10	0	SMS	p:+41788833520	6D5ED1EC-EEE9-42A7-B6EC-9FF99EB613E8
EE53E325-C6A4-4738-9307-B8FF14F42977	Test 1(sur téléphone de jc)	0	(null)	4	(null)	(null)		10	0	iMessage	p:+41788833520	F25E8C44-77EE-4C2F-AB16-52FDC723D979
33C55699-BD32-486F-B0DA-D52D52EEDD49	Coucou amour ❤	0	(null)	4	(null)	(null)		10	0	iMessage	p:+41788833520	F25E8C44-77EE-4C2F-AB16-52FDC723D979

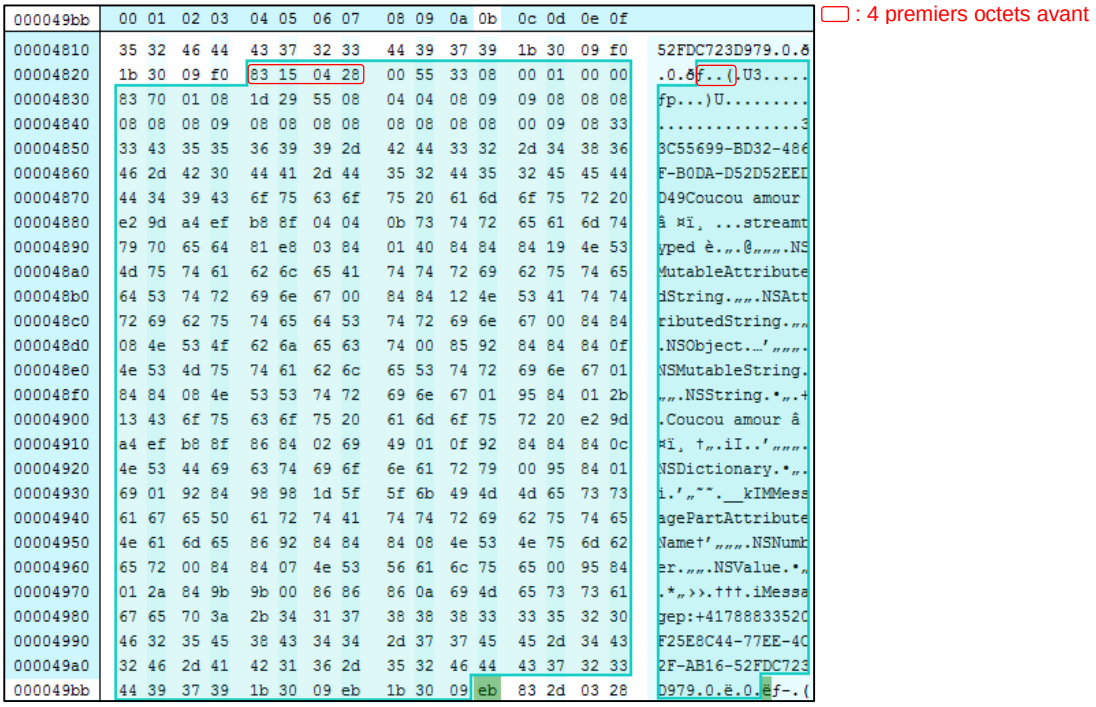
Afin de pouvoir comparer le contenu du SMS avant et après, nous avons utilisé un outil nous permettant de lire des données sous format hexadécimal, il s'agit du logiciel « Hex Editor Neo », ce logiciel est sensible à la casse lors de la recherche.

L'outil « Oxygen Forensic SQLiteViwer » nous permettant de récupérer le SMS supprimé dans un freeblocks, sous format hexadécimal, nous devons donc passer par ce format pour la comparaison.

Sur Figure 39, nous avons utilisé « Hex Editor Neo » pour lire le SMS avant suppression. Ce dernier est surligné en bleu. La partie centrale sélectionnée de la figure correspond au SMS sous format hexadécimal et la partie sélectionnée à droite de la figure correspond à l'interprétation textuelle du SMS.

Nous avons vu précédemment que lorsque le message est supprimé, l'espace qu'occupait ce message est rendu disponible et le freeblock est écrit. Il faut donc observer si les quatre premiers octets varient entre le message avant suppression et le message après suppression. Puisque, rappelons-le, le message reste présent jusqu'à ce qu'il n'y ait plus d'espace disponible.

Figure 39 : SMS sous format hexadécimal et textuel avant suppression

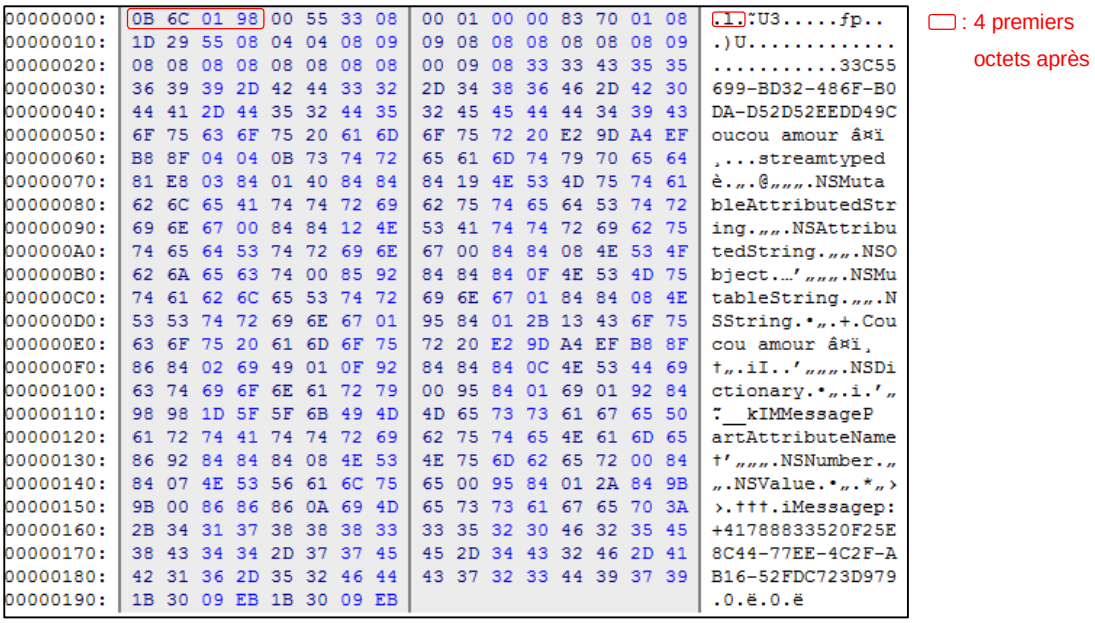


Après suppression du message et récupération du fichier, nous utilisons le logiciel « Oxygen Forensic SQLiteViwer » afin de récupérer les blocs contenant des données supprimées. A présent il s'agit de déterminer s'il s'agit bien de freeblocks. Donc que le message est toujours présent mais que le bloc est défini comme libre.

Sur la Figure 40, le message correspondant est surligné en bleu (il s'agit de l'ensemble de la figure). La partie centrale de la figure correspond au message sous format hexadécimal et la partie à droite de la figure correspond à l'interprétation textuelle du message.

L'analyse se porte sur les quatre premiers octets du message. Nous constatons que ces octets ont changé mais que la suite du message ne varie pas. Il s'agit donc bien d'un freeblock.

Figure 40 : SMS sous format hexadécimal et textuel après suppression



Voici, ci-dessous, un tableau récapitulatif de la comparaison des quatre premiers octets avant et après suppression du message.

Tableau 8 : Comparaison des quatre premiers octets (premier test)

Octets avant suppression	**Octets après suppression**
85 15 04 28	0B 6C 01 98

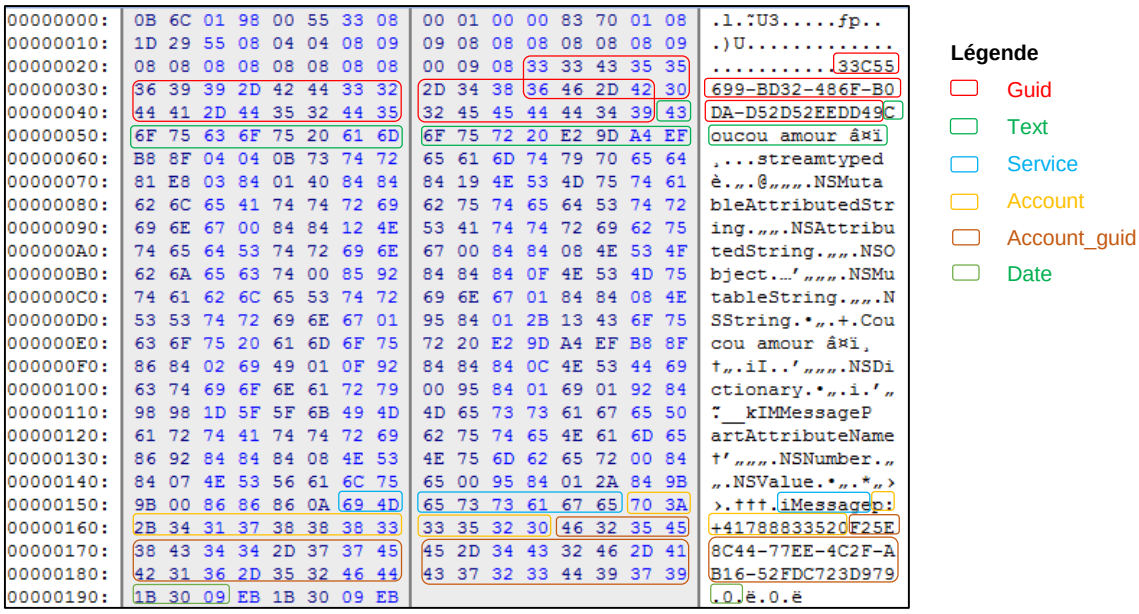
A présent analysons le message que nous avons récupéré. Voici le message que nous avions (Voir Figure 41) dans la table Message.

Figure 41 : Visualisation du message avant suppression

guid	text	r...	servi...	h...	subj...	country	a...	v...	t...	service	account	account_guid
33C55699-BD32-486F-B0DA-D52D52EEDD49	Coucou amour ❤□	0	(null)	4	(null)	(null)		10	0	iMessage	p:+41788833520	F25E8C44-77EE-4C2F-AB16-52FDC723D979

Voici les informations récupérées après suppression que nous pouvons identifier (voir Figure 42).

Figure 42 : Identification des informations récupérées



Pour résumer ce que nous venons d'analyser. Lorsqu'une suppression a lieu, SQLite réécrit les deux premiers octets en mettant comme information le prochain freeblock disponible. Pour cet exemple, 0B 6C correspond au prochain freeblock, cette information est en hexadécimal, il faut la traduire en décimal pour trouver le prochain freeblock (en décimal cette valeur équivaut à 45164).

Les deux octets suivants sont réécrits également avec comme valeur celle de la taille du freeblock courant. C'est pour cela que les valeurs des quatre premiers octets d'un enregistrement supprimé varient.

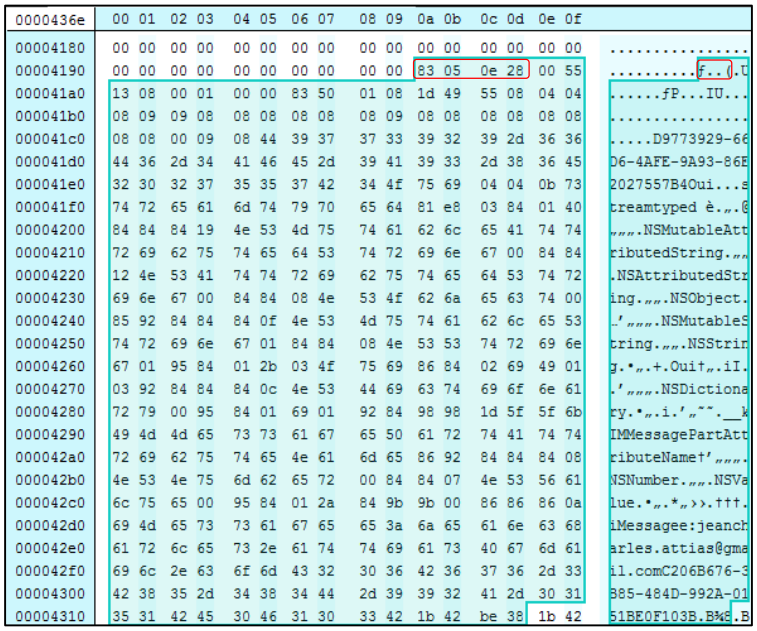
Les manipulations sont les mêmes pour les autres bases de données que nous avons analysées (par exemple : la base de données « SMSSearchindex.sqlite »).

10.1.2 Effacer un iMessage reçu d'une conversation

Afin de confirmer l'analyse, il est important d'effectuer un second test en supprimant un message plus court. Cette fois, il s'agit d'un iMessage.

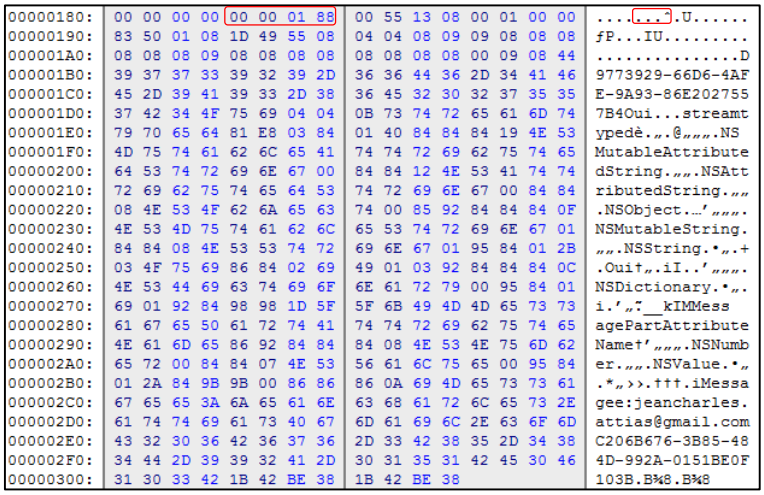
Après avoir effectué les manipulations des fichiers comme précédemment expliquées, voici, sur la Figure 43, le message correspondant avant suppression.

Figure 43 : iMessage sous format hexadécimal et textuel avant suppression



Après avoir supprimé l'iMessage dont le contenu était « Oui » et après avoir effectué les manipulations des fichiers comme précédemment expliquées, nous pouvons analyser sur la Figure 44 que le message est bien récupéré.

Figure 44 : iMessage sous format hexadécimal et textuel après suppression



L’analyse se porte sur les quatre premiers octets du message comme nous l’avons dit précédemment. Nous constatons que ces octets ont changé mais que la suite du message ne varie pas. Il s’agit donc bien d’un freeblock.

Voici, ci-dessous, un tableau récapitulatif de la comparaison des quatre premiers octets avant et après suppression du iMessage.

Tableau 9 : Comparaison des quatre premiers octets (second test)

Octets avant suppression	**Octets après suppression**
83 05 0E 28	00 00 01 88

10.1.3 Résultats des tests après suppressions

Après avoir effectué les différents tests, voici les résultats que nous obtenons.

Tableau 10 : Résultats des tests effectués sur l'appareil après suppressions

N°	Tests	iPhone non jailbreaké	iPhone jailbreaké
1	Effacer un SMS envoyé, d'une conversation	SMS envoyé, récupéré	
2	Effacer un SMS reçu, d'une conversation	SMS reçu, récupéré	
3	Effacer un iMessage envoyé, d'une conversation	iMessage envoyé, récupéré	
4	Effacer un iMessage reçu, d'une conversation	iMessage reçu, récupéré	
5	Effacer une conversation entière	Une partie de la conversation est récupérée avec la version d'essai du logiciel « Oxygen Forensic SQLiteViwer » (le programme limite le nombre d'éléments récupérés dans cette version)	

10.2 Analyse des tests après restaurations

Ici, il s'agit d'effectuer des tests de restaurations de l'appareil afin de vérifier, en fonction des tests que nous effectuerons, si nous pouvons récupérer les données ou si elles sont définitivement supprimées.

Au préalable, nous effectuons une sauvegarde dite « de base » afin de pouvoir comparer avec les différents tests réalisés.

Tableau 11 : Tests effectués sur l'appareil non jailbreaké après restaurations

N°	Tests	iPhone non jailbreaké
6	Après restauration comme nouvel iPhone depuis iTunes	Aucun message ni conversation récupéré
7	Après restauration comme nouvel iPhone depuis téléphone	Aucun message ni conversation récupéré
8	Après restauration de l'iPhone avec une sauvegarde iCloud	Récupération des mêmes données que pour la sauvegarde « de base »
9	Après restauration de l'iPhone avec sauvegarde iTunes (local)	Récupération des mêmes données que pour la sauvegarde « de base »

10.3 Analyse des tests avec encryption

Lorsque l'on chiffre une sauvegarde iTunes tous les fichiers sont encryptés, et non pas seulement le dossier de sauvegarde lui-même. De ce fait, on ne peut pas ouvrir ces fichiers pour les analyser. Il faudrait récupérer une clé de cryptage, mais il s'agit ici d'un autre sujet.

10.4 Constats

Premièrement, il faut qu'il n'y ait plus d'espace disponible dans la base de données SQLite avant la réécriture des nouvelles données sur les freeblocks, là où se trouvent les enregistrements supprimés).

De plus, et cela est constatable, la suppression ne change pas la taille du fichier puisque les données supprimées sont toujours présentes.

Par ailleurs, il est important de comprendre que les zones non allouées (disponibles) qui ont des valeurs autres que zéro peuvent être techniquement considérées comme une donnée supprimée.

11. Logiciels existants de récupération de messages

Certains logiciels proposent de récupérer les messages présents sur le téléphone ainsi que ceux qui ont été supprimés. Ces programmes demandent d'effectuer une sauvegarde de l'appareil.

Parmi eux le logiciel proposé par Istonsoft[5].

Par ailleurs, Le groupe CCL, important dans le domaine de la science légale, propose de nombreux logiciels[6] permettant de récupérer des informations à partir d'un appareil.

Malheureusement la plupart des logiciels sont payants et proposent une version d'essai qui ne suffit pas à évaluer la performance et la qualité du programme.

12. Utilisation d'applications tierces

Lorsque l'on utilise une application comme par exemple « ZMS », « FreeSMS global » ou encore « BiteSMS » pour envoyer ou recevoir des messages.

Ces messages seront logiquement stockés ailleurs, à savoir, dans la racine de l'application.

[5] L'outil est disponible à l'adresse suivante :

http://www.istonsoft.fr/ios-recovery/recover-deleted-text-messages-from-iphone.html

[6] Les logiciels sont disponibles à l'adresse suivante :

http://www.cclgroupltd.com/product-category/buy-softw/

13. Logiciels testés

Avant de parvenir à trouver des logiciels me permettant de récupérer, analyser ou créer des modèles, j'ai dû installer et tester plusieurs outils différents.

Le Tableau 12 présente ces différents outils, une description de ce qu'ils permettent d'effectuer ainsi que l'utilité dans le cadre de ce travail.

Tableau 12 : Analyse des outils testés

N°	Nom du logiciel	Description	Utilisé
1	DBeaver Entreprise par Free Universal Database Manager	Cet outil permet de réaliser des modèles de données à partir de base de données (SQLite faisait partie des bases prise en compte).	Oui
2	Sqlite expert personnal par SQLite Expert	Ce logiciel permet d'importer des bases de données SQLite et les interprète de manière à ce que la lecture des données soit plus facile (par le biais de couleurs).	Oui
3	Oxygen Forensic SQLiteViwer par Oxygen Sotware	Ce programme permet notamment d'analyser les blocks contenant des données supprimées dans un fichier SQLite. Il s'agit d'un bon outil. En revanche, la version d'essai ne dure que 30 jours et ne permet pas d'obtenir certaines données.	Oui
4	Hex Editor Neo	Cet outil permet d'analyser les fichiers sous format hexadécimal.	Oui
5	SystoolsSQLitedatabase recovery	Une des fonctionnalités de ce logiciel est de visualiser les triggers présents dans un fichier SQLite.	Oui
6	Filezilla	Ce logiciel permet de récupérer des éléments tels que des dossiers, fichiers et autres sur l'appareil en utilisant le protocole SSH.	Oui

N°	Nom du logiciel	Description	Utilisé
7	iBackup Extractor par Wideanglesoftware	Ce programme ne permet pas de récupérer les messages supprimés, seulement ceux présents sur l'appareil à partir d'une sauvegarde iTunes.	Non
8	Addon par Firefox	Cet outil permet de visualiser les données avec des couleurs prédéfinies que nous pouvons modifier. Cet outil concerne la lecture de base de données SQLite.	Non
9	Forensic Explorer par GetData Forensics	Je n'ai malheureusement pas pu tester ce logiciel qui permettait une analyse de fichier sous format hexadécimal car la fonctionnalité d'import n'était pas disponible sur la version d'essai.	Non
10	Driver ODBC SQLite	Le Driver permet notamment d'effectuer des requêtes en ligne de commande dans une fenêtre DOS. Les résultats des requêtes se présentent sur plusieurs lignes et sont difficiles à lire et à analyser.	Non
11	Sqlite Database Browser par SQLite Browser	Ce programme interprète les bases de données SQLite comme l'outil « Sqlite expert personnal », mais il est moins facile au niveau de la lecture	Non
12	Fileviewer Lite	Cet outil nous permet de lire les fichiers –wal, -shm et autres fichiers que nous ne pouvons pas lire par le biais d'un interpréteur de base de données SQLite (car ils sont dit « cryptés », en réalité ils n'ont pas le format de base de données SQLite). En revanche, la lecture du fichier n'est pas lisible en format texte comme proposé, nous pouvons seulement le lire en format hexadécimal. Il faut donc exporter ce fichier en format hexadécimal et utiliser un autre outil nous permettant d'interpréter l'hexadécimal.	Non

N°	Nom du logiciel	Description	Utilisé
13	Systools Sqlite Forensic Explorer	Ce logiciel permet d'analyser un fichier en hexadécimal par le biais de couleurs. L'outil a fonctionné une fois, mais certains boutons ne marchaient pas du tout. De plus, l'analyse du fichier prend beaucoup de temps.	Non
14	Elcomsoft Phone Viewver	Ce programme permet la récupération de messages présents sur l'appareil et certains messages supprimés. En revanche dans la version d'essai, seulement dix messages sont visibles. Le logiciel se base sur le fichier « Manifest.plist ». Ce fichier est présent dans le chemin suivant : « \Utilisateurs\(nom d'utilisateur)\AppData \Roaming\Apple Computer\MobileSync\Backup »	Non

Conclusion

Les techniques de récupération de messages sur iOS sont diverses et variées. Selon le modèle et la version du système d'exploitation de l'appareil, il faudra utiliser une, voire plusieurs méthodes afin de réussir à récupérer des données.

Dans ce travail de Bachelor, la recherche est axée sur la récupération logique d'informations. Dans le cas où ce type de récupération ne peut pas avoir lieu (par exemple : récupération d'un téléphone endommagé), il faudra utiliser du matériel (hardware) pour extraire des données. UFED Touch est notamment une des solutions de méthodes de récupération physique permettant d'extraire et de décoder des preuves numériques contenues dans des appareils mobiles. En revanche ce type d'appareil est conçu spécifiquement pour être utilisé par des organismes d'investigation autorisés tels que les services de police. L'analyste d'investigation étudie l'appareil pour en déterminer le fonctionnement interne afin de récupérer des informations : il s'agit bien de « reverse engineering » de données. Tout l'art de la reconstitution de l'historique des messages.

Grâce à ce travail de Bachelor, j'ai pu comprendre le fonctionnement de la suppression de messages sur iOS. De plus, j'ai découvert et testé des logiciels permettant de récupérer des messages supprimés. La partie de test était vraiment importante car elle m'a permis de vérifier et de valider la compréhension que j'ai acquise du fonctionnement. Ce travail m'a apporté des connaissances sur des éléments auxquels je n'avais pas songé. Je pense ici à la base de données Spotlight avec l'indexation de la totalité des métadonnées, réglages par défaut de l'appareil.

J'ajouterais à cela comme difficulté de ce travail, le manque de documentations lié à la notion de légalité du domaine et surtout le défaut de mises à jour de la documentation existante. Je fais allusion à mes recherches concernant Spotlight, et à la difficulté de trouver des explications sur les attributs de la base. Le peu de donnée disponible la plupart des informations étant en anglais.

Cette analyse a requis une importante rigueur afin de conserver le fil conducteur tout au long de ce développement. Beaucoup d'éléments étant liés, une simple modification aurait pu me conduire sur un autre sujet comme l'encryption par exemple.

Enfin, outre l'aspect technique, cette analyse nous ramène nécessairement à des questionnements en termes de confidentialité et de vie privée. .
J'en tire pour ma part une expérience extrêmement positive et un enrichissement personnel qui me motivent à découvrir d'autres outils et à poursuivre mes études dans le domaine.

Webographie

Apple - Sauvegarde iCloud [Consulté le 17.03.2015]. Disponible à l'adresse : https://support.apple.com/kb/PH12519?locale=en_US&viewlocale=fr_FR

Planetoscope – Statistiques en temps réel [Consulté le 17.03.2015]. Disponible à l'adresse : http://www.planetoscope.com/electronique/718-nombre-de-sms-envoyes-dans-le-monde.html

CCL Group – Fonctionnement du WAL [Consulté le 20.03.2015]. Disponible à l'adresse: http://www.cclgroupltd.com/the-forensic-implications-of-sqlites-write-ahead-log/

SQLite – Format de fichier SQLite [Consulté le 20.03.2015]. Disponible à l'adresse : http://sqlite.org/fileformat2.html

Apple – Sauvegardes sur iOS [Consulté le 25.03.2015]. Disponible à l'adresse : https://support.apple.com/fr-ch/HT204136

Apple – Restauration du contenu d'un appareil iOS [Consulté le 25.03.2015]. Disponible à l'adresse : https://support.apple.com/fr-ch/HT1766

The iPhone wiki – Détail de l'application message [Consulté le 30.03.2015]. Disponible à l'adresse : https://theiphonewiki.com/wiki/Messages

Mac Developer Library – Base de données Spotlight [Consulté le 30.03.2015]. Disponible à l'adresse : https://developer.apple.com/library/mac/documentation/Carbon/Conceptual/MetadataIntro/Concepts/HowDoesItWork.html

SMS iPhone – Transformer le fichier de sauvegarde en autre format [Consulté le 02.04.2015]. Disponible à l'adresse : http://www.smsiphone.org/

iPhoneTweak – Récupération de messages [Consulté le 02.04.2015]. Disponible à l'adresse : http://iphonetweak.fr/2012/11/05/tuto-comment-recuperer-des-sms-effaces-sur-son-iphone

Cellebrite - Type d'extraction [Consulté le 15.04.2015]. Disponible à l'adresse : http://fr.slideshare.net/cellebriteUFED/explaining-cellebrite-ufed-data-extraction-processes-final

Analyse forensic [Consulté le 20.04.2015]. Disponible à l'adresse : https://www.sstic.org/media/SSTIC2012/SSTIC-actes/forensicsios/SSTIC2012-Slides-forensicsios-sigwald_bedrune.pdf

John Lehr - B-Tree Leaf Pages [Consulté le 02.05.2015]. Disponible à l'adresse : http://linuxsleuthing.blogspot.ch/2013/09/recovering-data-from-deleted-sqlite.html

Apple - synchronisation iTunes [Consulté le 13.05.2015]. Disponible à l'adresse : https://support.apple.com/fr-ch/HT201253

Hacking and Securing iOS Application – Jonathan Zdziarski [Consulté le 14.05.2015]. Disponible à l'adresse : https://books.google.de/books/about/Hacking_and_Securing_iOS_Applications.html?hl=de&id=Youyu15xY9gC

Magnet forensics - Investigating iOS Phone Images, File Dumps & Backups [Consulté le 14.05.2015]. Disponible à l'adresse http://www.magnetforensics.com/mobile-forensics/investigating-ios-phone-images-file-dumps-backups

WikiHow - Comment récupérer des SMS supprimés de l'iPhone [Consulté le 23.05.2015]. Disponible à l'adresse : http://fr.wikihow.com/r%C3%A9cup%C3%A9rer-des-SMS-supprim%C3%A9s-de-l%27iPhone

Deadhardrive - Retrouver des sms effacés dans le Spotlight sur iPhone [Consulté le 26.05.2015]. Disponible à l'adresse : http://www.deadhardrive.com/retrouver-des-sms-effaces-dans-le-spotlight-sur-iphone/

Deadhardrive - Extraire la base de données sms.db d'un Iphone [Consulté le 28.05.2015]. Disponible à l'adresse : http://www.deadhardrive.com/extraire-la-base-de-donnees-sms-db-dun-iphone/

Wikipédia - Spotlight [Consulté le 29.05.2015]. Disponible à l'adresse : https://fr.wikipedia.org/wiki/Spotlight

Slideshare - iPhone forensics, without the iPhone [Consulté le 01.06.2015]. Disponible à l'adresse : http://fr.slideshare.net/hrgeeks/iphone-forensics-without-the-iphone

Open security research - Forwarding SMS to Email on [Jailbroken] iOS [Consulté le 01.06.2015]. Disponible à l'adresse : http://blog.opensecurityresearch.com/2013/02/forwarding-sms-to-email-on-jailbroken.html

Another Forensics Blog - Finding and Reverse Engineering Deleted SMS Messages [Consulté le 02.06.2015]. Disponible à l'adresse : http://az4n6.blogspot.fr/2013/02/finding-and-reverse-engineering-deleted_1865.html

Sean Morrissey - iOS Forensic Analysis [Consulté le 03.06.2015]. Disponible à l'adresse: https://sensperiodit.files.wordpress.com/2011/04/ios-forensic-analysis-for-iphone-ipad-and-ipod-touch.pdf

Safari – SQLite Databases [Consulté le 03.06.2015]. Disponible à l'adresse : https://www.safaribooksonline.com/library/view/hacking-and-securing/9781449325213/ch04s02.html

Researchgate – A recovery method of deleted record for SQLite database [Consulté le 05.06.2015]. Disponible à l'adresse : http://www.researchgate.net/publication/226423207_A_recovery_method_of_deleted_record_for_SQLite_database

Sandbox - Extracting SQLite records [Consulté le 05.06.2015]. Disponible à l'adresse : http://sandbox.dfrws.org/2011/fox-it/DFRWS2011_results/Report/Sqlite_carving_extractAndroidData.pdf

BinaryHexConverter – Hexadecimal to Decimal Converter [Consulté le 20.06.2015]. Disponible à l'adresse : http://www.binaryhexconverter.com/hex-to-decimal-converter

Sanderson Forensics - Recovering deleted records from an SQLite database (updated) [Consulté le 23.06.2015]. Disponible à l'adresse : http://sandersonforensics.com/forum/content.php?222-Recovering-deleted-records-from-an-SQLite-database