La récupération de données supprimées dans une base SQLite

Alexandre CILIA[1], Pascal PERENON[2]

[1 2]Sous Direction de la Police Technique et Scientifique,
Service Central de l'Informatique et des Traces Technologiques

31 avenue Franklin Roosevelt, 69134 ECULLY cedex

alexandre.cilia@interieur.gouv.fr pascal.perenon@interieur.gouv.fr

Résumé – SQLite est une base de données compatible avec le langage SQL couramment utilisée dans les téléphones mobiles, ordinateurs et tablettes. Cet article décrit une méthode pour récupérer des données supprimées dans un fichier au format SQLite. Cette méthode est d'abord basée sur la localisation des espaces non alloués dans la structure d'une base de données. L'étape suivante consiste en l'analyse de chaque espace afin d'identifier et d'afficher les enregistrements effacés. La caractéristique de cette méthode est d'être efficace sur tout fichier SQLite, indépendamment de la connaissance de la structure des données. Cette méthode a été implémentée dans un logiciel, nommé « SqliteFreespaceCarver »

Abstract – SQLite is an embedded SQL language compliant database used in cellphones, computers, tablets, etc. This paper presents a new method to recover deleted data from a SQLite file. This one is first based on the localisation of the free spaces in SQLite format structure. The next step is to analyze each free space in order to uncover and display deleted records. The main property of this method is to be efficient on any SQLite file independently of the knowledge of the database. We implemented it in a software named "SqliteFreespaceCarver".

1. Introduction

SQLite (Standard Query Language Lite) est un système de gestion de bases de données relationnelles compatible avec le langage SQL. Cette base de données a été conçue pour être légère et simple d'intégration. Sa principale caractéristique est de proposer une bibliothèque C dont les sources sont distribuées dans le domaine public [1], permettant aux programmes l'utilisant d'accéder à une base de données SQLite de manière directe, sans nécessiter l'emploi d'une architecture client-serveur lourde. Les bases de données ainsi créées sont réduites à un unique fichier dont la structure est documentée et publique [2].

Ce format est aujourd'hui utilisé comme moyen de stockage de données par des logiciels libres ou commerciaux, tant pour micro ordinateur que dans des systèmes embarqués. On le retrouve aujourd'hui, en particulier, employé dans les smartphones de la marque Apple et par le système d'exploitation Android, aussi bien pour enregistrer des informations techniques, telles que journaux et fichiers de paramètres, que des données personnelles. Il est dès lors apparu comme intéressant d'étudier la possibilité de récupérer les données supprimées présentes dans ces fichiers.

Bien que cet article ne cherche pas à décrire la conception d'un logiciel permettant de consulter le contenu apparent dans une base de données SQLite, la première partie de ce document est consacrée à l'étude du format de fichier. En effet, une étude approfondie de celui-ci a été nécessaire pour identifier les zones dans lesquelles des données supprimées peuvent être retrouvées afin de parvenir à leur reconstruction. Lorsque nous employons des termes techniques issus de la documentation officielle qui ne peuvent être traduits, nous écrivons ceux-ci entre guillemets.

La seconde partie de ce document décrit la méthode choisie pour permettre la collecte et la reconstruction des données supprimées ; que nous avons intégrée dans un outil développé pour notre usage.

2. Le format de fichier SQLite

Les fichiers créés par une base de données SQLite contiennent à la fois la définition de la structure des tables et l'intégralité des données que celles-ci contiennent. Ce fichier est désigné comme le « main database file » par la norme SQLite. Ces fichiers sont couramment désignés par le terme « fichier SQLite » que nous employons dans le présent document.

La structure interne des fichiers SQL est organisée en pages dont la taille est définie dans l'en-tête du fichier, qui est contenu sur les 100 premiers octets de la première page.

Cet en-tête contient des informations essentielles pour la reconstruction des données.

TAB. 1 : Extrait de l'en-tête de fichier SQLite

Description	Offset	Taille en octets
La chaîne de caractères: « SQLite format 3 »	0	16
La taille des pages en octets	16	2
...		
Nombre de pages de la base de données	28	4
Numéro de la première « freepage »	32	4
Nombre total de « freepages »	36	4
...		
Mode d'encodage du texte	56	4

Les autres pages de la base de données sont définies en fonction de leur rôle.

Les « b-tree pages » sont les pages affectées à la structure et au contenu de la base de données,

Les « overflow pages » contiennent les données des enregistrements ne tenant pas sur une seule page,

Les « pointer-map pages» sont les pages n'existant que dans les bases de données auto-nettoyées,

Les « locking pages » n'existent que dans les bases de données de plus de 230 octets,

Les « freepages » sont d'anciennes pages des catégories précédentes qui ont été par la suite libérées.

2.1 Les b-tree pages

Les pages de type «b-tree page », sont organisées, comme leur nom l'indique, dans une structure en arbre. Il existe quatre catégories de « b-tree page » :

interior index b-tree page

interior table b-tree page

leaf index b-tree page

leaf table b-tree page

Les « leaf table b-tree page » sont celles qui nous intéressent le plus dans le cadre d'une récupération de données. L'en-tête des « b-tree page » nous donne les informations permettant d'en identifier le type.

TAB. 2 : En-tête de « b-tree page »

Description	Offset	Taille en octet
Type de b-tree page	0	1
Offset du premier bloc libre	1	2
Nombre de cellules de la page	3	2
Offset de la première cellule de contenu	5	2
Nombre d'octets libres fragmentés	7	1
Pointeur right-most	8	4
Tableau de pointeur de cellules	8 ou 12	2 par cellule

2.1.1 Le stockage des données dans les pages

Chaque « leaf table b-tree page » de la base contient des enregistrements appartenant à une seule table. Au sein d'une page de données, les données de la table sont enregistrées au sein de blocs appelés cellules. Ces blocs sont enregistrés à partir de la fin de la page. Cela conduit à l'existence d'un espace non alloué entre l'en-tête de la page et la première cellule de contenu.

Chaque cellule de la page contient les données d'un enregistrement de la table. Lorsqu'un enregistrement est supprimé dans la table, la cellule qu'il occupait dans la page devient un bloc libre pour une nouvelle utilisation.

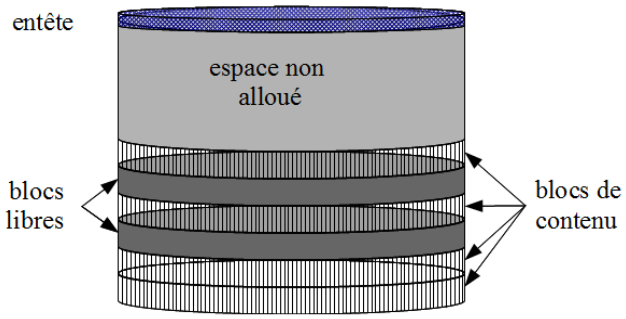

FIG. 1 : Structure d'une page

2.1.2 Structure des cellules et des données dans une page

Chaque cellule est composée d'un en-tête et d'une portion de données.

TAB. 3 : Structure de cellule

En-tête de cellule		Portion de données	Numéro de page de débordement
Taille de l'enregistrement	Clé		

La portion de données est organisée selon sa propre structure, qui se compose d'un en-tête décrivant le type des données enregistrées et d'un tableau comprenant les données elles-mêmes.

TAB. 4 : Structure de la portion de données d'une cellule

En-tête					Tableau de valeurs des données			
longueur en-tête	Tableau de définition des données							
	définit° champ 1	définit° champ 2	. . .	définit° champ N	val. champ 1	val. champ 2	. . .	val champ N

Les longueurs, les clés et les définitions de données ainsi décrites sont enregistrées dans un format de données à longueur variable appelée « varint » (variable length integer). Ce format permet de stocker des valeurs numériques sur un espace compris entre 1 et 9 octets.

Pour chaque octet, le premier bit permet de définir si la valeur s'étend sur l'octet suivant (bit à 1) ou si l'octet en cours de lecture est le dernier (bit à 0). La valeur numérique est enregistrée dans les 7 bits suivants.

TAB. 5 : Exemple de décodage d'une donnée « varint »

Varint (hexadécimal)	8CA06F
Varint (binaire)	**1**0001100 **1**0100000 **0**1101111
Entier (binaire)	00000011 00010000 01101111
Entier (hexadécimal)	03106F
Entier (décimal)	200815

Les en-têtes de cellules utilisent ce format de données pour encoder les définitions de champs.

TAB. 6 : Définitions de champs SQLite

Valeur décimale de la définition de champ	**Longueur du champ en octets**	**Définition**
0	0	NULL
1	1	Entier sur 8 bits
2	2	Entier sur 16 bits
3	3	Entier sur 24 bits
4	4	Entier sur 32 bits
5	6	Entier sur 48 bits
6	8	Entier sur 64 bits
7	8	Big-endian IEEE 754-2008 64-bit floating point number
8	0	Valeur 0
9	0	Valeur 1
10,11		réservé
≥12 et pair	(N-12)/2	blob de (N-12)/2 octets
≥13 et impair	(N-13)/2	chaîne de (N-13)/2 octets

On peut remarquer que certaines optimisations d'occupation d'espace ont été prévues. Ainsi les constantes « 0 » et « 1 » définies respectivement par les valeurs « 8 » et « 9 » n'auront pas de champ correspondant dans le tableau de valeur des données. On constate aussi que la définition des champs « blob » et chaîne de caractère est à la fois porteuse du type et de la taille.

2.2 Les espaces libres

L'étude de la structure par les informations contenues dans les en-têtes nous permet d'identifier des espaces contenant des zones libres décrites explicitement.

- Les « freepages » qui sont des pages libérées au cours de la vie de la base. En fonction de leur ancien type, elles peuvent contenir des enregistrements supprimés. Les « freepages » sont référencées par la base de données dans une structure appelée « freelist ».
- Les « freeblocks » sont des enregistrements supprimés dont l'espace est devenu disponible. Ces zones sont intercalées entre les enregistrements apparents d'une page.

Deux autres types d'espaces libres existent mais ne sont pas référencés explicitement par la base :

- La « unallocated region » peut être présente dans chaque page de données (« b-tree page » et « overflow page »). Il s'agit de l'espace disponible présent entre l'en-tête et la première cellule de données de chaque page.
- Les « fragmented bytes », sont les espaces de moins de 4 octets ne contenant pas de données dans une page.

Ces espaces libres contiennent les données suivantes :

- De l'espace réellement vide dans la « unallocated region » d'une nouvelle page qui n'aurait pas été complètement remplie.
- Des copies de données, produites en raison de la réorganisation des données effectuée automatiquement par le moteur de base de données.
- Les données issues d'enregistrements de la base de données qui ont été supprimés et qui sont ceux qui nous intéressent.

2.3 Les freepages (pages libres)

Les pages libres sont référencées dans une liste, appelée « freelist ». La « freelist » est une liste de numéros de pages libres. Les pages qui composent la « freelist » sont désignées comme « freelist trunck pages » (tronc). Les pages libres contenant des données sont quant à elles les « freelist leaf pages » (feuilles). Le numéro de la première « freelist trunck pages » est indiqué dans l'en-tête du fichier, à l'offset 32. Le contenu des pages n'est pas modifié lorsqu'elles sont libérées : elles conservent donc leur en-tête qui permet de déterminer le type auquel elles

appartenaient et donc d'identifier les anciennes « leaf table b-tree page » susceptibles de contenir des données.

TAB. 7 : Structure des freelist trunck pages

Taille en octets	Donnée
4	Nombre d'éléments du tableau
4	Numéro de la prochaine freelist trunck pages
4	Numéro de freelist leaf pages
4	Numéro de freelist leaf pages
...	
4	Numéro de freelist leaf pages
4	Numéro de freelist leaf pages

2.3.1 Les freeblocks

Les « freeblocks » sont les espaces libres dans une page de la base de données. Chaque page gère sa propre liste. L'adresse du premier bloc est indiquée dans l'en-tête de la page, à l'offset 1. Les « freeblocks » sont ensuite organisés en chaine, comme décrit ci-après.

Les 2 premiers octets de chaque « freeblock » indiquent l'adresse du prochain. Les 2 octets suivants indiquent la taille du bloc (les 4 premiers octets compris). Cet en-tête de « freeblock » est écrit à la place des anciennes données. Cela peut conduire à la perte des premières données de l'ancienne cellule. (Taille de la cellule, clé, taille de l'en-tête de définition des données et définition du premier champ)

TAB. 8 : Structure des freeblocks

En-tête de freeblock		contenu
Adresse du prochain freeblock	Taille du freeblock	

2.4 Etat de l'art

2.4.1 Exploitation manuelle à l'aide d'une visionneuse de fichiers binaires

L'exploitation des données supprimées présentes dans un fichier SQLite est réalisable à l'aide d'une visionneuse de fichiers binaires telle que WinHex [3]. Cette approche est pertinente lorsqu'il s'agit de rechercher des éléments de texte dans un volume restreint de données. En effet, ces outils ne permettant pas de différencier les données apparentes des autres données, l'exploitation manuelle doit être faite sur l'ensemble du fichier, ce qui rend inefficace une telle exploitation sur un grand volume de données. De plus, si le décodage des données qui ne sont pas en texte clair est généralement proposé pour les formats numériques et de date les plus courants, le format « varint » n'est lui pas pris en charge par les outils d'interprétation de données des visionneuses de fichiers binaires. Or, ce format de données est nécessaire à l'identification des structures utilisées par la base de données.

2.4.2 Recherche spécifique aux applications

Cette méthode consiste à exploiter la connaissance de la structure des bases SQLite dans leurs implémentations spécifiques. En effet, les données sont toujours enregistrées dans une base par une application en respectant la même séquence d'enregistrement des champs. Cette approche a été étudiée [6] et est probablement employée dans certains outils d'analyse forensique (Microsystemation Xry [4]. Cellebrite UFED [5]) qui permettent la récupération d'éléments supprimés dans certains appareils. Pour les appareils utilisant SQLite, en particulier l'iPhone, ces outils retrouvent des données dans les espaces libres de ces fichiers. Ces outils sont néanmoins limités car ils ne traitent que les fichiers des applications dont le décodage a été implémenté. Les fichiers ainsi exploités correspondent à des applications considérées comme prioritaires par ces éditeurs, mais il n'est pas possible d'utiliser ces outils pour analyser un fichier dont le traitement n'a pas été prévu. De plus nous avons pu constater que la récupération des éléments supprimés n'est pas toujours complète.

2.4.3 CLL Forensics - Epilog

La société CLL Forensics propose un logiciel « Epilog » permettant l'exploitation de données supprimées dans les fichiers SQLite [4]. L'approche choisie par cette société est d'employer selon les cas des signatures spécifiques à certaines applications ou une reconstruction des enregistrements par un algorithme que nous ne connaissons pas.

3. Méthode de récupération mise en œuvre

Les outils existants nous fournissant des résultats incomplets, nous avons décidé d'implémenter notre propre méthode de localisation des zones libres de la base et une méthode de traitement de ces zones libres, afin de récupérer l'ensemble des données supprimées d'une base SQLite.

Parmi les quatre types de zones libres, nous ne traitons pas les zones de type « fragmented bytes » que nous considérons comme inexploitables en raison de leur taille trop réduite (4 octets). La localisation des « freepages » est effectuée par la lecture de leur numéro dans la « freelist trunck page », page dédiée au référencement de ce type de page. Nous gardons en mémoire tous les numéros des « freelist leaf pages ». Par la suite, à chaque traitement de page, nous pourrons vérifier si le type de la page appartient à cette liste et la traiter en tant que telle. La localisation des zones « unallocated region » est effectuée en interprétant l'en-tête d'une page de données active qui indique l'adresse de la fin de l'en-tête et le début de la première cellule (selon l'adresse de la page). Le troisième type,

certainement le plus intéressant, est le « freeblock ». C'est une zone libre qui contient potentiellement une ou plusieurs cellules supprimées. L'adresse du premier « freeblock » est indiquée dans l'en-tête de la page de données active. Les quatre premiers octets de chaque « freeblock » indiquent la taille du « freeblock » ainsi que l'adresse du prochain « freeblock » dans la page. Ces informations sont lues séquentiellement jusqu'à la localisation de l'ensemble des « freeblocks » de la page.

Lorsqu'un type de zone libre est localisé, il est immédiatement traité. Nous avons défini actuellement deux traitements des zones libres : L'affichage en mode brut des données et l'affichage formaté des données.

Les données supprimées étant issues d'une base de données, l'idéal est de les représenter formatées suivant leur format d'origine (texte, nombre, date, ...) en indiquant si possible le nom de leur champ (nom, âge, numéro de téléphone, adresse), informations que nous pouvons trouver dans la description de la table à laquelle ont appartenu les données supprimées.

Pour notre première implémentation, nous avons affiché en mode brut les zones de type « freepage » et « unallocated region ». Nous les traiterons en mode formaté dans une version future. Concernant les zones de type « freeblock », le début de chaque cellule contient dans son en-tête la définition du format de ses données : nous pouvons donc théoriquement récupérer celle-ci pour formater ses données. Cette méthode serait complète et suffisante si l'ensemble des données d'une cellule supprimée était intègre. Or les 4 premiers octets de chaque cellule supprimée ont été effacés et remplacés par 4 octets de l'en-tête d'une zone « freeblock ».

Bien que la définition du format de données soit précédée par quelques octets de l'en-tête de la cellule active, il existe une possibilité que cet en-tête soit inférieur à 4 octets et que le début de la définition soit effacé. Pour vérifier cette éventualité, nous récupérons la définition complète d'une cellule active de la page en cours. Les types de champs et leur ordonnancement doivent correspondre à ceux de la cellule supprimée. La comparaison des types de champs nous permet de valider si la définition de la cellule supprimée est intègre ou non.

Si l'en-tête (altéré) de la cellule supprimée est supérieur ou égal à 4 octets, les données sont formatées et affichées suivant la définition (cas 1 de la figure ci-dessous). Sinon, si l'en-tête est inférieur à 4 octets, il existe deux cas de figure possibles : si le ou les premier(s) type(s) de champ de définition de la cellule active est (sont) un type de longueur fixe alors nous reconstruisons le début de la définition effacée avec celle de la cellule active (cas 2). Si le premier type de champ est de longueur variable (texte, blob) alors nous affichons les données en mode brut car nous ne pouvons pas retrouver la longueur de ce champ de manière automatique (cas 3).

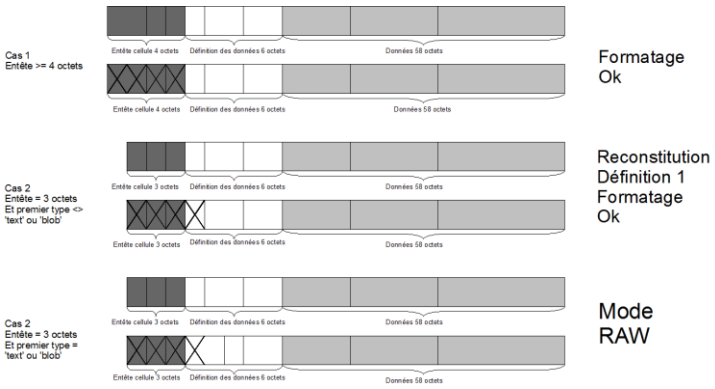

FIG. 2 : Méthodes de reconstruction des enregistrements

Nous avons implémenté ces méthodes de localisation et de traitement dans un programme nommé « SQLiteFreeSpaceCarver » (SFC).

3.1 Application

Le programme SFC est né dans le cadre d'une affaire judiciaire réelle. L'objectif de la mission était de récupérer des messages textuels qui auraient été envoyés d'un téléphone mobile vers un numéro de téléphone utilisé par la victime. Le téléphone mobile de marque Apple modèle iPhone 3Gs stocke l'ensemble de ces SMS dans un fichier au format SQLite. Dans notre cas, la taille de ce fichier est de 184 kilooctets et le nom de ces tables est le suivant :

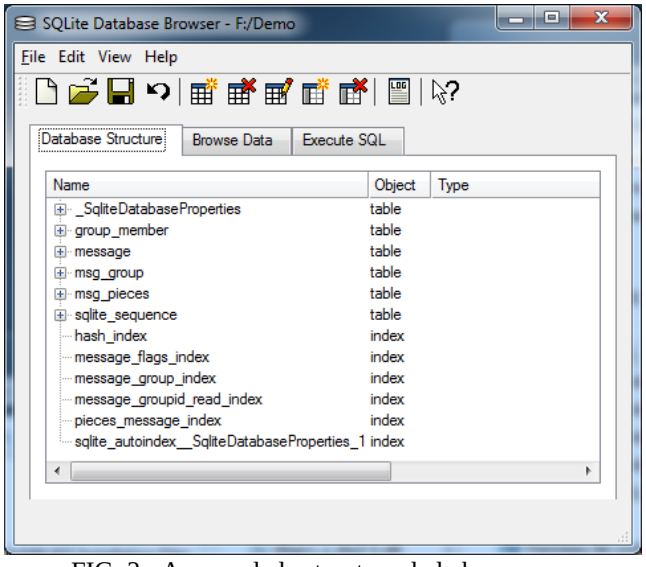

FIG. 3 : Aperçu de la structure de la base

Nous nous intéressons à la table « message » qui contient l'intégralité des SMS reçus ou envoyés par ce téléphone. La figure ci-dessus indique la structure interne de cette table notamment les noms et le type des champs. L'exploitation de cette table par une visionneuse de

fichiers SQLite (SQLite Database Browser par exemple [8]) nous a permis de visualiser les 904 SMS reçus ou envoyés. Aucun ne correspondait au message recherché.

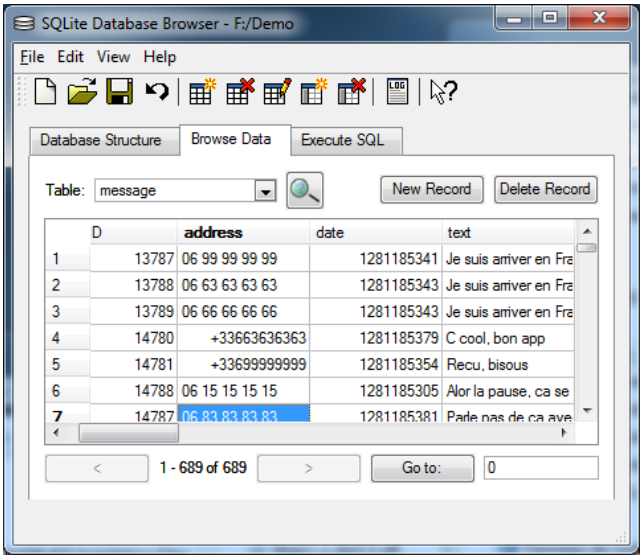

FIG. 4 : Aperçu du contenu de la table

Les résultats en récupération de données supprimées obtenus par les logiciels commerciaux que nous utilisons ne permettaient pas de mettre en évidence le message recherché. Une recherche de texte brut nous avait permis de mettre en évidence l'existence d'éléments de texte supplémentaires, non traités.

Le programme a donc été élaboré et utilisé pour extraire et exploiter l'ensemble des zones libres du fichier en question en espérant récupérer des données supprimées intéressantes. Le programme SFC demande en paramètre le nom du fichier SQLite à exploiter. Il suffit ensuite de lancer le traitement en appuyant sur le bouton « lancer l'extraction » :

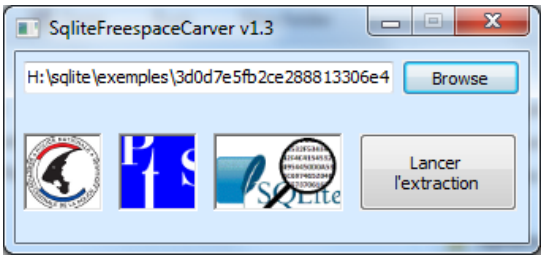

FIG. 5 : Interface de SQLiteFreeSpaceCarver

Après exécution, les zones libres sont concaténées dans un fichier de même nom que le fichier traité mais avec l'extension « .freespace ». Ce fichier est au format XML et chaque zone libre récupérée est encapsulée par des métadonnées indiquant : l'adresse de la zone libre, la taille de la zone (en octets), le type de la zone libre, le numéro de la page et son type d'affichage (brut ou formaté).

```
<contenuEfface offset=65116 longueur=420 typeFreeSpace=FreeBlock
page=31 Affichage=RAW>
Ò  % □K            +33666666666L^Ò±J'ai fais un hors forfait de
15,50â,¬. Je crois c'est quand j'Ã©tais en Espagne.    |    fr
A % /            +33777777777LÀÓ/C'est pas grave  biz   |    fr
</contenuEfface>
```
FIG. 6 : Aperçu d'un résultat brut

Lorsque la zone est de type « freeblock », un formatage des données est possible. Dans ce cas, chaque donnée récupérée est précédée du type de champ et de sa longueur en octets.

```
<contenuEfface offset=124679 longueur=123 typeFreeSpace=FreeBlock
page=31 Affichage=formate>
<numerique 16 bits - longueur 2 octets>##
<text - longueur 13 octets>+33666666666
<numerique - longueur 4 octets>L♦Z♦
<text - longueur 43 octets>Wesh poto bien? Samedi je sai pa si
jpourais
<...>
</contenuEfface>
```
FIG. 7 : Aperçu d'un résultat formaté

A l'analyse des zones libres, nous avons visualisé une zone « freeblock » contenant du texte correspondant au message recherché et envoyé à un numéro connu de la victime.

```
<contenuEfface offset=81993 longueur=581
typeFreeSpace=Unallocated page=21 Affichage=formate>
<numerique 16 bits - longueur 2 octets>##
<text - longueur 13 octets>+33699999999
<numerique - longueur 4 octets>####
<text - longueur 135 octets>fai attention tt se paye ds la vi
mec. je sais que tu dis chair des saloperies sur moi a t potes et
aussi ce que tu a eecri sur Facebook
...
</contenuEfface>
```
FIG. 8 : Aperçu d'un résultat formaté

En faisant manuellement la correspondance entre la définition de la table « message », à laquelle appartient le message supprimé, et celle du « freeblock », nous avons pu reconstruire la structure du SMS recherché et l'ensemble de ses paramètres : adresse du destinataire, date, texte, sens, ...

4. Conclusion

SFC nous a permis de répondre à la mission qui nous avait été demandée en indiquant des informations importantes en relation avec le SMS envoyé. Nous utilisons SFC dorénavant pour chaque mission analogue indifféremment sur tous types de fichier SQLite provenant de tous types d'applications. Notons toutefois que certaines applications (Firefox, Chrome) en fonction de leurs versions, utilisent l'effacement automatique des données supprimées ce qui rend l'utilisation de SFC caduque. Cependant, le nombre important et croissant d'applications utilisant SQlite, nous encourage à travailler sur une prochaine version qui intégrera les tentatives de reformatage de l'ensemble des zones libres ainsi que l'affichage des noms de champs directement dans le fichier résultat.

Références

[1] http://www.sqlite.org/fileformat2.html
[2] SQLite Home Page; http://www.sqlite.org/

[3] http://www.x-ways.net/winhex/index-f.html
[4] http://www.msab.com
[5] http://www.cellebrite.com
[6] Forensic analysis of the Firefox 3 Internet history and recovery of deleted SQLite records, Murilo Tito Pereira, Digital Investigation, Volume 5, Issues 3-4, March 2009, Pages 93-103
[7] http://www.ccl-forensics.com/Software/epilog-from-ccl-forensics.html
[8] http://sqlitebrowser.sourceforge.net