DOI: 10.5769/C2015006 or http://dx.doi.org/10.5769/C2015006

Método de recuperação de mensagens apagadas do *SQLite* no contexto do aplicativo *WhatsApp* para plataforma *Android*

Alberto Magno M. Soares, João Paulo C. de Sousa, Juliano K. M. Oya

Resumo. **Discutimos um método de recuperação de mensagens apagadas do aplicativo *WhatsApp*, cujo armazenamento se dá em base de dados utilizando sistema gerenciador de banco de dados *SQLite*, disponível na plataforma *Android*. No desenvolvimento da técnica, foram analisadas as estruturas internas do arquivo da base de dados *SQLite* com potencial interesse para uma investigação forense dessa natureza. Especificamente, foram exploradas as regiões não alocadas (*freespace*) e desalocadas (*freeblocks*) das bases de dados *SQLite*, com foco na recuperação estruturada de mensagens do aplicativo *WhatsApp*.**

Palavras-chave: *Análise Forense, Android, Recuperação de dados, WhatsApp, SQLite, Freeblock, Freespace*

Abstract. **We discuss a recovery method for WhatsApp application's deleted messages, which occurs in stored database using database management system SQLite, available on the Android platform. In the technique's development, was examined the internal structures of the SQLite database file with potential interest for a forensic investigation of this nature. Specifically, the unallocated (freespace) and deallocated (freeblocks) regions of SQLite databases were explored, focusing on structured recovery of WhatsApp application messages.**

Keywords: *Forensics Analysis, Android, Data Recovery, WhatsApp, SQLite, Freeblock, Freespace*

I. Introdução

WhatsApp é uma aplicação de mensagens instantâneas com versões disponíveis para *smartphones* com os sistemas *Android*, *BlackBerry*, *iPhone*, *Windows Phone* e *Symbian*, e até janeiro de 2015 atingiu a marca de 700 milhões de usuários [1].

A plataforma *Android* já contabilizou, segundo Gartner [2], mais de um bilhão de usuários em 2014, liderando o mercado de sistemas operacionais [2].

Em 2014, os exames de informática em aparelhos de telefonia celular que utilizam *WhatsApp* sobre plataforma *Android* representaram grande parte dos exames periciais em dispositivos móveis que foram requisitados ao Instituto de Criminalística da Polícia Civil do Distrito Federal e, em muitos casos, o examinador necessitou pesquisar a existência de vestígios de mensagens apagadas ou fragmentos destas no dispositivo.

O método que será apresentado permite recuperação de mensagens apagadas do aplicativo *WhatsApp*, cujo armazenamento se dá em base de dados utilizando sistema gerenciador de banco de dados *SQLite*, disponível na plataforma *Android* [3].

O presente trabalho inicia com a análise de características do aplicativo *WhatsApp* e do sistema gerenciador de banco de dados *SQLite*. Em seguida, são apresentados detalhes de interesse forense, incluindo análise das estruturas internas da base de dados envolvidas no processo de apagamento e recuperação de mensagens. Ao final, é detalhado o algoritmo de recuperação de mensagens, construído com base no método apresentado.

II. Materiais e Métodos

Para o estudo, foi utilizado ambiente *Android* emulado (*Android SDK for x86, Android 5.1 (Lollipop), Kernel 3.4.67+digit@tyrion.par.corp.google.com#3, Build sdk_phone_x86-eng 5.1 LKY45 1737576*), possibilitando acesso irrestrito ao sistema de arquivos da memória interna.

Para construção da base *SQLite* de exame, foi ativada conta no aplicativo WhatsApp de testes no dispositivo emulado, e foram enviadas mensagens no formato 'Mensagem #', com cenários de deleção individual ou de várias mensagens simultâneas.

A análise da estrutura interna dos arquivos SQLite foi realizada mediante uso do software *FTK Imager*, versão 3.2.0.0 [4].

Detalhamento do método de recuperação desenvolvido está descrito nos itens IV e V.

III. Referencial Teórico

Nesta seção são apresentados os conceitos relacionados ao aplicativo *WhatsApp* e ao banco de dados *SQLite*.

A. Banco de dados SQLite

O *SQLite* é uma biblioteca de *software* de código aberto que implementa um sistema de banco de dados SQL transacional, sem a necessidade de um servidor dedicado e com pouca ou nenhuma configuração para seu funcionamento. Além disso, é um sistema autossuficiente e extremamente compacto – *SQLite* lê e escreve diretamente para arquivos de disco comuns e não possui um processo tipo servidor separado – e, dessa forma, é uma escolha bastante popular para o uso em dispositivos móveis como celulares. Um banco de dados *SQLite* completo contendo tabelas, índices, gatilhos e visões, subsiste em um único arquivo em disco [5] [6].

Sobre sua arquitetura, os componentes do *SQLite* são agrupados em categorias denominadas *Core*, *SQL Compiler*, *Backend* e *Acessories*. Especificamente na categoria *Backend*, há os componentes diretamente relacionados com a estrutura de armazenamento de dados, que são as árvores-b (*b-tree*) e as

páginas (*Page*) [5]. A Figura 1 apresenta uma visão geral desses componentes e seus principais relacionamentos.

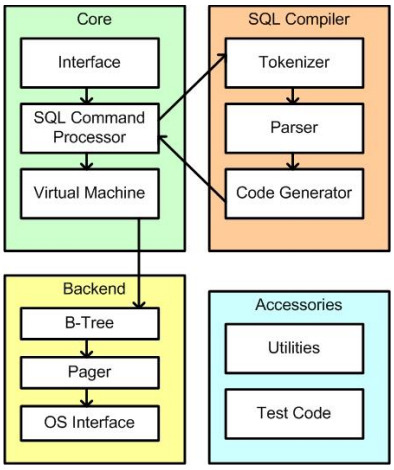

Fig. 1. Visão geral da arquitetura do sistema gerenciador de banco de dados *SQLite*.
Fonte: adaptado de [5]

Uma visão geral da estrutura de um arquivo de banco de dados do *SQLite* pode ser vista na Figura 2. Assim, um arquivo de banco de dados é dividido em várias unidades de armazenamento, chamadas páginas. As páginas são numeradas sequencialmente – iniciando em 1 – e possuem o mesmo tamanho, que pode ser entre 512 (2^9) e 65.536 (2^{16}) *bytes*.

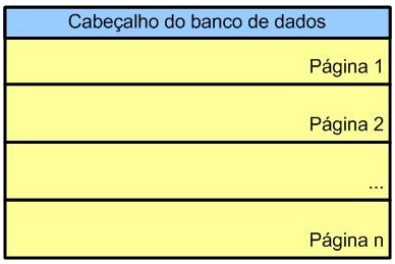

Fig. 2. Visão geral da estrutura de um arquivo de banco de dados.

No início da primeira página é armazenado o cabeçalho do banco de dados (uma sequência de 100 *bytes*), cujos primeiros 16 *bytes* são a assinatura `SQLite format 3`, indicando que se trata de um banco de dados *SQLite*. Ainda no cabeçalho pode ser extraída a informação do tamanho das páginas (*offset* 16, tamanho de 2 *bytes* os quais devem ser interpretados como um inteiro de 16 *bits* no formato *big endian*). Essa mesma página é raiz de uma *b-tree* que contém uma tabela especial denominada `sqlite_master` que armazena o esquema completo do banco de dados.

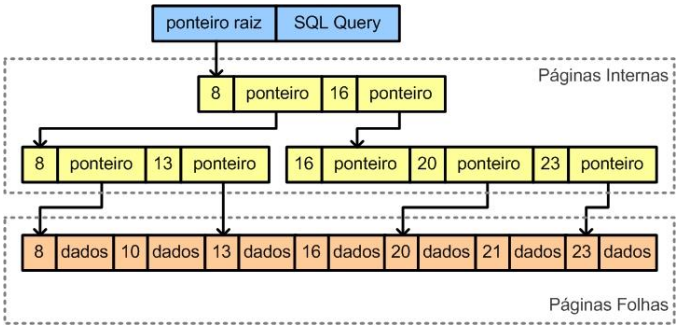

Fig. 3. Exemplo de uma estrutura de árvore balanceada.
Fonte: adaptado de [6]

Uma grande parte do arquivo do banco de dados *SQLite* é organizada em uma ou mais estruturas *b-tree*. Uma única estrutura *b-tree* é armazenada usando uma ou mais páginas, e cada página contém um único nó da *b-tree*. As páginas usadas para armazenar uma única estrutura de *b-tree* não precisam formar um bloco contíguo [5]. Essa estrutura é exemplificada na Figura 3.

Ainda na Figura 3, podemos observar que as páginas podem ser de tipos diferentes, de acordo com o tipo de informação que ela armazena. Dentre os tipos documentados em [5], as páginas folhas da *b-tree* são as de maior interesse pericial, já que nelas são armazenados os dados dos registros.

Uma página folha pode ser identificada pelo primeiro campo do seu cabeçalho, que são 2 *bytes* com o valor de *flag* `0x0D`. Logo após o cabeçalho da página, fica armazenada uma lista de ponteiros (inteiros de 2 *bytes* no formato *big endian*) cujos valores são os *offsets* para cada célula de dados da página. Na Figura 4 é exemplificada a estrutura de uma página.

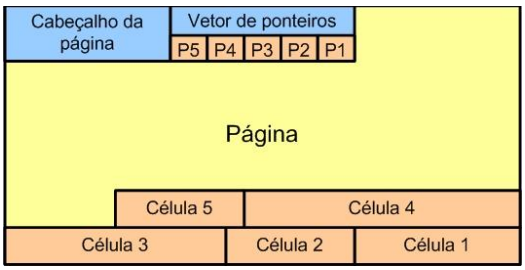

Fig. 4. Visão geral da estrutura de uma página do *SQLite*.

Assim, dentro de uma página, a célula é a unidade elementar de armazenamento de dados, sendo, portanto, a unidade de armazenamento dos registros do banco de dados. Cada célula possui uma estrutura exemplificada nas Figuras 5 e 6. No cabeçalho da célula estão os dados do tamanho da célula, excluindo o seu cabeçalho, e o campo *row id*, cujo formato é no padrão *varint* (*variable length integers*) [5].

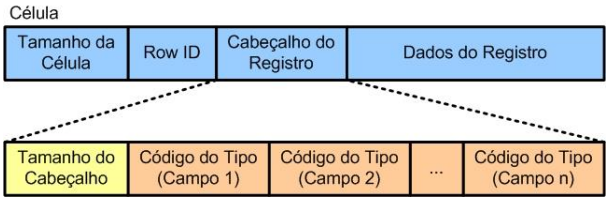

Fig. 5. Estrutura de uma célula e do cabeçalho do *payload*.

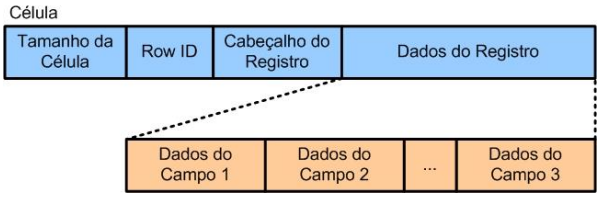

Fig. 6. Estrutura de uma célula e do *payload*.

Ainda na Figura 5, no cabeçalho do registro é armazenado o tamanho do cabeçalho (no formato *varint*) e, em seguida, o tipo de cada campo, onde cada *byte* representa o tipo de dado armazenado na área de dados, como mostrado na Figura 6. O valor do *byte* e o tipo de dado que ele representa é mostrado na Tabela I.

TABELA I. DESCRIÇÃO DOS CÓDIGOS DO TIPO DOS DADOS DO REGISTRO.

Valor do Tipo	Descri□□o	Tamanho
0	0	NULL
1 a 4	Inteiro com sinal	1 a 4
5	Inteiro com sinal	6
6	Inteiro com sinal	8
7	Ponto flutuante	8
8	Constante de valor 0	0
9	Constante de valor 1	0
N ≥ 12 e par	BLOB	(N-12)/2
N ≥ 13 e par	*String*	(N-13)/2

B. Aplicativo WhatApp

O *WhatsApp* é um aplicativo de mensagens multiplataforma que permite trocar mensagens através do celular e da *Internet*. Tal aplicativo está disponível em diversas plataformas de comunicação móvel e seu uso é bastante popular em telefones celulares do tipo *smartphone*.

Em investigações criminais, tal aplicativo é bastante utilizado como meio de comunicação pelos criminosos e o conteúdo das mensagens pode ser indício da materialidade do crime, da possível autoria e do modo de operação de uma associação criminosa. Versões atuais do aplicativo, na plataforma *Android*, usam o *SQLite*, embarcado na plataforma *Android*, como sistema gerenciador de banco de dados.

Dentre várias tabelas de armazenamento do aplicativo, há a tabela `messages`, onde são armazenadas as mensagens enviadas e recebidas pelo usuário do aplicativo [7]. Para ilustrar, alguns dos campos dessa tabela foram extraídos de uma mensagem real e interpretados, conforme a descrição dos tipos da Tabela I, e o resultado é apresentado na Tabela II.

TABELA II. DESCRIÇÃO DOS CAMPOS DA TABELA MESSAGES.

Nome	C□digo	Tipo	Tamanho
key_remote_jid	0x43	*String*	27
key_id	0x25	*String*	12
status	0x01	Inteiro	1
data	0x21	*String*	10
timestamp	0x05	Inteiro	6
send_timestamp	0x01	Inteiro	1
received_timestamp	0x05	Inteiro	6

Na Figura 7 é apresentado o registro com dados dessa mensagem. Considerando os valores de tamanhos obtidos na Tabela II, é possível extrair o conteúdo de cada campo da `messages`.

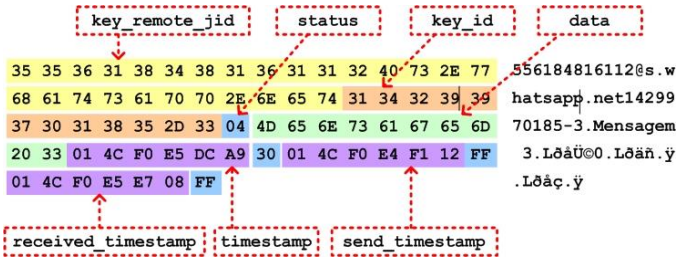

Fig. 7. Exemplo de interpretação da estrutura de dados de um registro de mensagem do aplicativo *WhatsApp*.

IV. RECUPERAÇÃO DE MENSAGENS

A. Recuperação de dados na região desalocada (freeblock) de páginas folha.

No cenário de recuperação de dados, deve-se primeiramente compreender o funcionamento das páginas folha, visto que estas páginas são as responsáveis por armazenar o conteúdo dos registros em um banco de dados *SQLite*. Uma página folha, assim como qualquer outra página, é estruturada em células. Cada página folha possui um cabeçalho composto de 8 *bytes*, que contém as informações descritas na Tabela III.

O cabeçalho é seguido por uma lista de *offsets* de 2 *bytes* que apontam para a área na página onde cada célula está posicionada. Caso haja "n" células na página, existirão "n" ponteiros de 2 *bytes* para os conteúdos das células, que estarão ordenados pelo valor das chaves dos registros. As células são alocadas do endereço final para o endereço inicial da página, conforme ilustrado na Figura 8.

TABELA III. CAMPOS DO CABEÇALHO DE UMA PÁGINA FOLHA

Offset/Tamanho	Descri□□o
0 / 1	Um *byte* indicando o tipo de p□gina. O valor 13 (0x0D) □ usado para indicar que a p□gina □do tipo folha.
1 / 2	Quando um registro □ exclu□do, a c□lula correspondente ao registro torna-se desalocada (*freeblock*), liberando mais espa□o na p□gina para futuro uso. Os dois *bytes* deste campo indicam o *offset* para o in□cio do primeiro *freeblock*. O valor 0 indica que n□o h□ regiı es desalocadas. Cabe ressaltar que o valor □0□ n□o indica que a p□gina est□ cheia, mas sim que n□o h□ c□lulas removidas.
3 / 2	Dois *bytes* indicando o n□mero de c□lulas da p□gina.
5 / 2	Dois *bytes* indicando o *offset* do in□cio da □rea com conte□do de c□lulas.
7 / 1	Um *byte* indicando o n□mero de fragmentos na p□gina. O *freeblock* requer pelo menos 4 *bytes* de espa□o. Grupos de 1,2 ou 3 *bytes* isolados compı em os fragmentos.

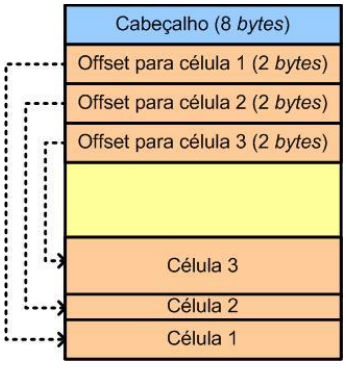

Fig. 8. Organização de uma página folha.

Quando um registro do banco de dados é excluído, algumas alterações são realizadas na estrutura da página folha. A célula com o conteúdo é desalocada, liberando espaço na página para inserção de novos registros. Entretanto, para que o espaço da célula seja liberado, tanto o cabeçalho da página, quanto os

offsets para os conteúdos das células devem ser reajustados. Quando a célula é excluída, o campo do cabeçalho da página que registra o número de células deve ser decrementado. Com a liberação de espaço, o *offset* para o início do primeiro *freeblock* deve ser ajustado. Assim, quando uma nova célula precisar ser alocada, a região referente ao *freeblock* poderá ser utilizada. Além disso, os *offsets* para as células são organizados sequencialmente, devendo, também, ser reajustados. A Figura 9 ilustra a remoção de uma célula:

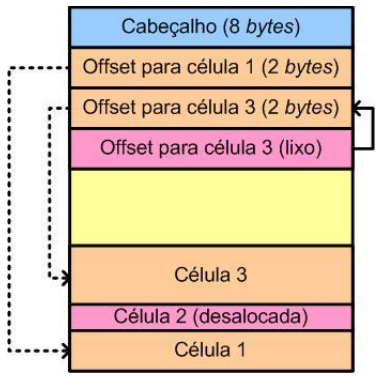

Fig. 9. Organização de uma página folha, após a remoção de uma célula.

Para a recuperação de conversas do aplicativo *WhatsApp*, as regiões desalocadas das páginas possuem grande utilidade forense. Será mostrado, adiante, como é possível recuperar mensagens dessas regiões, realizando-se uma análise do banco de dados em nível de *bytes*.

Em sistemas *Android*, o banco de dados *SQLite* que armazena as conversas é denominado `msgstore.db`. O referido banco tem em sua estrutura interna a tabela `messages` que é responsável por armazenar, além de outras informações, o conteúdo das conversas. No exemplo mostrado na Figura 10, foram enviadas três mensagens entre dois usuários de teste do aplicativo *WhatsApp*, sendo uma delas removida logo após o envio das três mensagens.

Fig. 10. Troca de mensagens através do aplicativo *WhatsApp*.

Realizando uma consulta (SQL *query*) na tabela `messages`, é possível observar que, antes da exclusão da "mensagem 2", havia, na tabela, quatro registros. O primeiro registro é criado quando o *WhatsApp* inicia uma conversa e não possui dados sobre as mensagens. Os demais registros estão associados às mensagens e possuem informações relevantes. Após a remoção da "mensagem 2", um dos registros é apagado, reduzindo as entradas na tabela `messages`. Tal processo pode ser visualizado na Figura 11.

Fig. 11. Consulta no *SQLite* mostrando diferenças no banco antes e após a exclusão de uma mensagem.

Analisando a estrutura interna do banco, é possível observar que os dados são armazenados em células no interior de uma página folha. Antes da "mensagem 2" ser excluída, a página que armazena os dados possui quatro células e, como o banco nunca tinha sido utilizado anteriormente, toda a região vazia está preenchida com zeros. Não existe também nenhuma informação relativa a *freeblocks*, ou seja, o campo no cabeçalho da página associado a este tipo de informação está zerado. A Figura 12 ilustra a estrutura da página antes da exclusão da "mensagem 2". Ressalte-se que os *offsets* mostrados são relativos ao início da página.

Após a exclusão da mensagem, é possível observar uma série de mudanças na estrutura da página. Primeiramente, a célula que armazena o registro apagado é liberada para uso futuro, tornando-se parte dos *freeblocks*. A região que armazenava a célula excluída passa a ter grande relevância forense, uma vez que os dados são desalocados, porém a informação continua integra até que uma nova mensagem seja incluída e faça uso do *freeblock*. Desta forma, é possível criar mecanismos de buscas nestas regiões visando recuperar conversas do aplicativo *WhatsApp*.

Para tornar a página livre, algumas alterações devem ser feitas no cabeçalho da página. Primeiramente, o número de células é reduzido, implicando na alteração do campo do cabeçalho que guarda no número de células. Como a célula é liberada, o *offset* para o início dos *freeblocks* é alterado. Neste caso específico, o *offset* para o *freeblock* passa a ter o antigo valor do *offset* para a célula excluída. Os primeiros *bytes* da célula excluída também são alterados para manutenção da estrutura de ponteiros dos *freeblocks*. Outros campos também podem ser alterados à medida que são realizadas operações na página, podendo, inclusive, gerar fragmentação interna. Os *offsets* para as páginas também precisam ser rearranjados. A Figura 13 mostra o conteúdo da página após a remoção da "mensagem 2".

Como é possível observar, com exceção dos primeiros *bytes* que são alterados para manter a estrutura de ponteiros do *freeblock*, a região desalocada continua mantendo os dados da conversa, que pode ser recuperada por uma análise forense. Estas informações podem ser de grande relevância em uma investigação e não são visualizadas por meio das ferramentas de navegação SQLite convencionais.

B. Recuperação de dados na região não alocada (freespace) de páginas folha.

Na arquitetura do banco *SQLite*, as páginas folha estão constantemente sofrendo alterações. Em algumas situações, todas as células da página podem ser liberadas, tornando a página livre para novas inserções de registros. Como visto anteriormente, o conteúdo das células, mesmo após a exclusão de uma mensagem, permanece acessível e recuperável. Contudo, à medida que novos registros vão sendo inseridos, a região desalocada vai sendo ocupada, destruindo a informação das conversas mais antigas.

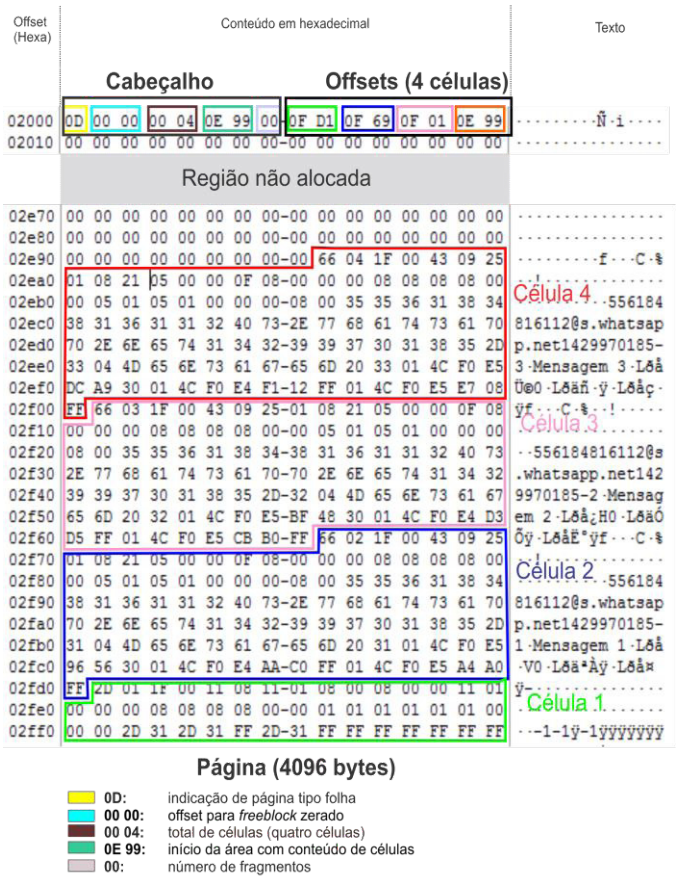

Fig. 12. Estrutura interna da página folha antes da exclusão da "mensagem 2". Conteúdo gerado pelo software *FTK Imager* [4] e posteriormente adaptado.

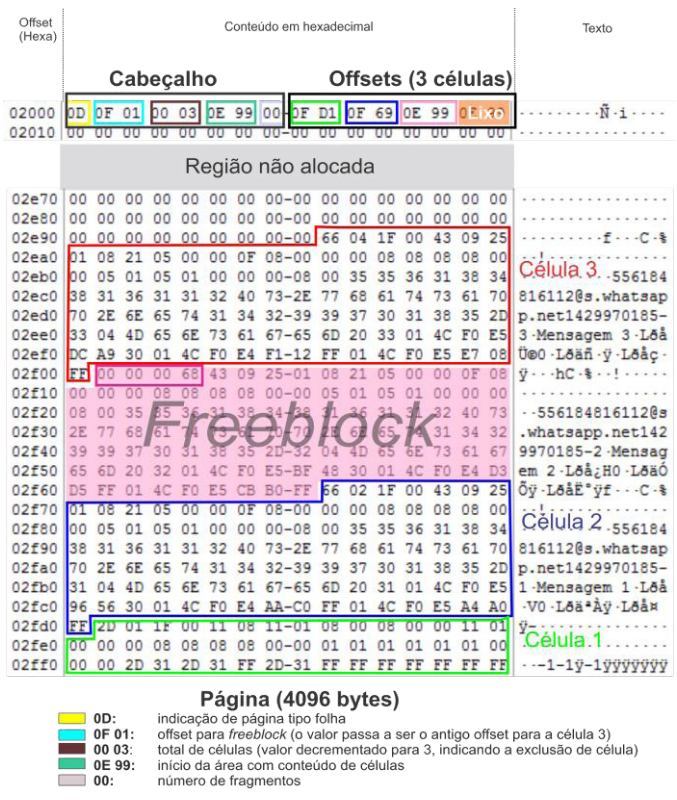

Fig. 13. Estrutura interna da página folha após exclusão da "mensagem 2". Conteúdo gerado pelo software *FTK Imager* [4] e posteriormente adaptado.

No cenário ilustrado na Figura 14, foram enviadas diversas mensagens ao usuário "Teste WhatsApp 2". Após o envio, todas as conversas foram excluídas e uma nova mensagem foi enviada ao usuário "Teste WhatsApp 3".

Fig. 14. Chats com os usuários "Teste WhatsApp 2" e "Teste WhatsApp 3".

Neste cenário, mesmo com a exclusão do chat com o usuário "Teste WhatsApp 2" e liberação das células, o banco irá manter as conversas recuperáveis. Todavia, quando a mensagem do usuário "Teste WhatsApp3" é incluída na página folha, parte dos dados são perdidos, tornando a recuperação das mensagens e fragmentos de mensagens mais complexa. A Figura 15 ilustra, em nível de *bytes*, como a inclusão da nova mensagem altera o conteúdo da região não alocada.

C. Outras possibilidades de recuperação de mensagens.

De acordo com o observado nos exames periciais cotidianos, um usuário do aplicativo *WhatsApp*, geralmente, realiza poucas ações de exclusão de mensagens, e quando o faz, é comum a exclusão de poucas linhas de conversa, com isso, grande parte dessas ações atuam em regiões analisadas de páginas de dados desalocadas e livres.

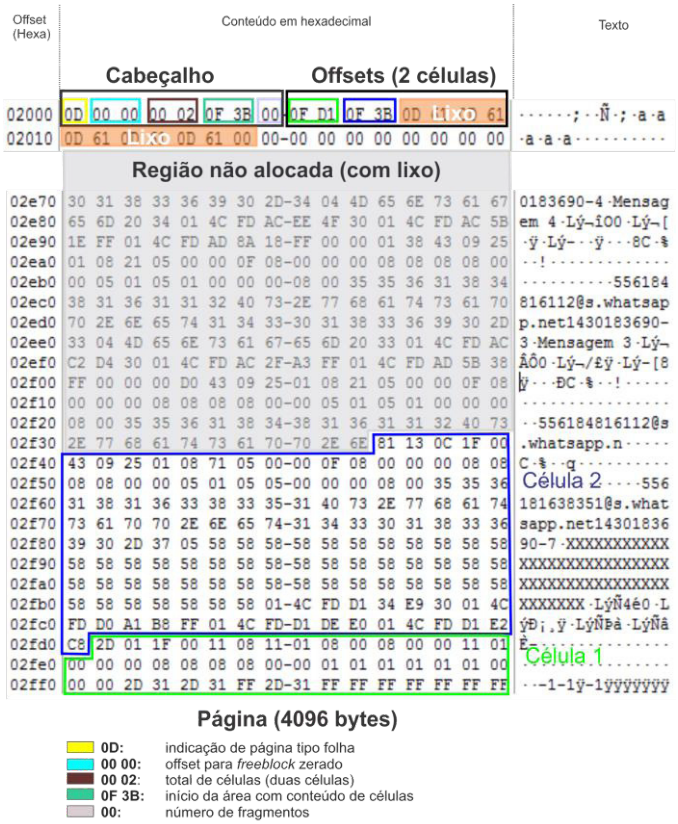

Fig. 15. Estrutura interna da página folha após exclusão do chat com o usuário "Teste WhatsApp 2" e do envio de mensagem para o usuário "Teste WhatsApp 3". Conteúdo gerado pelo software *FTK Imager* [4] e posteriormente adaptado.

Contudo, a especificação do formato *SQLite* define como deve ser feito o armazenamento de registros extensos que extrapolem o espaço disponível da página de dados. Para isso, é indicado ao final da célula, um ponteiro de 4 *bytes*, para página de *overflow*, onde é armazenado o conteúdo excedente.

Outra possibilidade de recuperação é quando há exclusão de páginas inteiras com mensagens, que resultariam no descarte de toda página, e no registro dessa como página desalocada. Tais páginas são armazenadas em uma lista de páginas livres (*freelist page*) e podem conter informação de relevância pericial.

O *SQLite* conta ainda com sistema de *rollback journal*, mantido em arquivo de log auxiliar, que pode conter mensagens recuperáveis não gravadas na base dados devido a interrupções inesperadas do funcionamento da *engine*. Existe mecanismo semelhante de operação de log, que pode ser ativado opcionalmente a partir da versão 3.6.0 do *SQLite*, denominado *Write-Ahead-Log*, conforme [5], onde há a possibilidade de recuperação de mensagens não gravadas em definitivo na base de dados, mas na versão do aplicativo *WhatsApp* analisado e em versões anteriores, não foi detectado ativação desse recurso como parte do mecanismo de manutenção de consistência.

Por abranger grande parte do foco usual dos exames executados, as regiões tratadas no método abordado restringiram-se inicialmente às regiões no interior das páginas de dados, mas, para um exame mais aprofundado, pretende-se estender o método para todas as estruturas do arquivo da base de dados e do sistema jornalístico cujas estruturas possam conter mensagens excluídas recuperáveis.

V. Algoritmo de Recuperação

Nesta seção é apresentado algoritmo de recuperação de mensagens apagadas do aplicativo *WhatsApp*, utilizando os conceitos apresentados.

A. Acesso a base de dados SQLite

O aplicativo *WhatsApp* para plataforma *Android*, na sua versão 2.12.58, armazena os seus arquivos de banco de dados de mensagens no diretório, de acesso restrito, `/data/data/com.whatsapp/databases`, da memória interna, onde se encontra o arquivo da base de dados *SQLite* denominado `msgstore.db`, objeto de análise. Existe mecanismo de *backup* do próprio aplicativo que realiza cópia criptografada do arquivo de mensagens para mídia de armazenamento removível do dispositivo, sendo que a chave criptográfica simétrica está armazenada no arquivo denominado `key`, localizado no diretório, de acesso restrito, `/data/data/com.whatsapp/files`, dificultando a extração.

Existem técnicas de extração do arquivo da base de dados através da exploração de vulnerabilidades da aplicação ou do próprio sistema operacional. A partir das últimas versões do aplicativo, há um aumento dos mecanismos de segurança, dentre eles a restrição com relação à extração de dados do aplicativo através do mecanismo de *backup* do sistema *Android*, muitas vezes utilizado em análise forense para ganho de acesso a diretórios restritos, o que exige do examinador o desenvolvimento contínuo de novas técnicas de extração.

B. Processo de recuperação

O método de recuperação de mensagens *WhatsApp* que é apresentado adota estratégia baseada na recuperação de páginas de dados, tipo folha, da estrutura b-*tree*, e inspeção das áreas de *freespace* (área não alocada) e *freeblock* (região desalocada) dessas páginas, em busca de células marcadas como apagadas que possibilitem recuperação das mensagens.

Para o processo de recuperação de mensagens, foi desenvolvido programa, na linguagem JAVA, para percorrer os dados do arquivo da base de dados, conforme algoritmo descrito na Figura 16.

Essencialmente, o algoritmo desenvolvido realiza leitura do tamanho das páginas do arquivo, inspeciona a área desalocada e levanta informações do esquema de dados da tabela `messages`. Iterativamente percorre todas as páginas de dados examinando o espaço não-alocado (*freespace*) e os blocos desalocados (*freeblocks*) e, em cada uma dessas áreas, faz análise léxica dos *bytes* no formato definido no esquema, localizando a posição da sequência de caracteres *@s.whatsapp*, característica do campo denominado `key_remote_jid` que deve conter identificação do interlocutor nos registros pertencentes à tabela de mensagens. Esta técnica é semelhante à utilizada em [8], para recuperação de registros da tabela `moz_places` da base de dados *SQLite* do *browser Firefox*.

Como exemplo de recuperação na área não-alocada (*freespace*) execução do algoritmo, inicialmente, foi analisado cenário da base de dados com exclusão simultânea de várias mensagens de uma conversa. O programa identificou mensagens em área não alocada, conforme descrito na Figura 17, indicando que aquelas mensagens foram apagadas e que a página foi restruturada pelo *engine SQLite*. Essa restruturação demonstra o aumento do tamanho da área não alocada, englobando a região das mensagens apagadas. Na Figura 18 pode-se observar registros da mensagem recuperada.

No cenário para recuperação em área desalocada (*freeblock*), foi recuperada mensagem fruto de uma deleção individual, conforme saída do processamento descrita na Figura 19.

Fig. 16. Processo de recuperação de mensagens.

```
Tamanho da Página do banco SQLite:4096
->Varrendo Free SPACE de 3887 byte(s)
  localizado no offset:8204
Buscando célula de mensagem WhatsApp ...
Offset: 8204
Tamanho:3887
<início>
?a?a?a?a?a? .... <bytes omitidos>
... ??????????????????????????????pC?%??!??????????????
????????556184816112@s.whatsapp.net1430183690-6?Mensagem
6?L??1?0?L??????L???8????C?%??!??????????????????????5
56184816112@s.whatsapp.net1430183690-5?Mensagem
5?L????0?L??????L?????????C?%??!??????????????????????5
56184816112@s.whatsapp.net1430183690-4?Mensagem
4?L???O0?L??[???L????????8C?%??!??????????????????????5
56184816112@s.whatsapp.net1430183690-3?Mensagem
3?L????0?L??/???L??[8?????C?%??!??????????????????????
556184816112@s.whatsapp.n
<fim>
```

Fig. 17. Trecho de saída do programa de recuperação que contem trecho da área não alocada (*freespace*).

```
1 Campo:id-Tipo:0-Tamanho:0-Conteúdo:0
2 Campo:key_remote_jid-Tipo:67-Tamanho:27-
  Conte□do:556184816112@s.whatsapp.net
3 Campo:key_from_me-Tipo:9-Tamanho:0-Conteúdo:1
4 Campo:key_id-Tipo:37-Tamanho:12-Conteúdo:1430183690-6
5 Campo:status-Tipo:1-Tamanho:1-Conteúdo:4
6 Campo:needs_push-Tipo:8-Tamanho:0-Conteúdo:0
7 Campo:data-Tipo:33-Tamanho:10-Conte□do:Mensagem 6
8 Campo:timestamp-Tipo:5-Tamanho:6-
  Conte□do:1430185259494(27/04/2015 22:40:59)
9 Campo:media_url-Tipo:0-Tamanho:0-Conteúdo:0
  ...
```

Fig. 18. Exemplo de trecho de mensagem recuperada da área não-alocada (*freespace*).

```
Varrendo Free BLOCK de 3723 byte(s)
  localizado no offset:8206
Buscando celula de mensagem WhatsApp ...
Offset: 12033
Tamanho:104
<início>
C?%??!??????????????????????556184816112@s.whatsapp.net
1429970185-2?Mensagem 2?L??H0?L??????L????f???
<fim>
<início - em HEX>
430925010821050000F0800000008080808000005010501000000080
0035353631383438313631313240732E776861747361706702E6E6574
3134323939373031383352D32044D656E736167656D2032014CF0E5BF
4830014CF0E4D3D5FF014CF0E5CBB0FF66021F00
<fim>
 1 Campo:id-Tipo:0-Tamanho:0-Conteúdo:0
 2 Campo:key_remote_jid-Tipo:67-Tamanho:27-
   Conte□do:556184816112@s.whatsapp.net
 3 Campo:key_from_me-Tipo:9-Tamanho:0-Conteúdo:1
 4 Campo:key_id-Tipo:37-Tamanho:12-Conteúdo:1429970185-2
 5 Campo:status-Tipo:1-Tamanho:1-Conteúdo:4
 6 Campo:needs_push-Tipo:8-Tamanho:0-Conteúdo:0
 7 Campo:data-Tipo:33-Tamanho:10-Conte□do:Mensagem 2
 8 Campo:timestamp-Tipo:5-Tamanho:6-
   Conte□do:1429970730824(25/04/2015 11:05:30)
 9 Campo:media_url-Tipo:0-Tamanho:0-Conteúdo:0
10 Campo:media_mime_type-Tipo:0-Tamanho:0-Conteúdo:0
  ...
```

Fig. 19. Exemplo de trecho de mensagem recuperada da área desalocada (*freeblock*).

Para mensagens muito longas, cujo conteúdo extrapola o espaço especificado na célula, são definidos 4 *bytes* no final da área destinada à célula que contém o endereço da página de *overflow* com o restante da mensagem que pode ser recuperada.

Observou-se também a possibilidade de recuperação de fragmentos de mensagens em áreas residuais, tipo *slack*, das áreas desalocadas (*freeblock*); porém, essas áreas são apresentadas pelo algoritmo para inspeção direta, pois devido a inconsistências detectadas nos dados, não há como ser efetuada leitura de acordo com a definição dos tipos definidos.

VI. Conclusão e Trabalhos Futuros

A recuperação de mensagens ou de fragmentos de mensagens excluídas do aplicativo *WhatsApp* pode ser de grande importância forense.

Neste artigo, foi apresentado um método desenvolvido para recuperação de mensagens apagadas do aplicativo *WhatsApp* , versão 2.12.58, sobre a plataforma *Android*, baseado na inspeção das estruturas internas do arquivo de base de dados *SQLite*, na localização de regiões livres ou desalocadas e na análise dos dados dessas áreas, permitindo a extração de dados e a potencial recomposição de mensagens excluídas.

Observou-se que é possível recuperar as mensagens apagadas das regiões referidas, inclusive com a possibilidade de os dados estarem íntegros.

Como trabalhos futuros, existem as possibilidades de recuperação de mensagens com grande extensão (que extrapolam o conteúdo de uma página de dados), de recuperação de páginas livres (*freepages*) e de recuperação de mensagens do arquivo do sistema de *journaling*.

Além disso, como trabalhos futuros, também podem ser produzidas abordagens para recuperação de mensagens excluídas em diferentes versões do aplicativo *WhatsApp*, bem como diferentes sistemas operacionais que façam uso de bases de dados *SQLite*.

Referências

[1] PORTAL G1, "WhatsApp atinge os 700 milhões de usuários por mês em todo o mundo", Disponível em <http://g1.globo.com/tecnologia/noticia/2015/01/whatsapp-atinge-os-700-milhoes-de-usuarios-por-mes-em-todo-o-mundo.html>. Acesso em 09 de maio de 2015.

[2] STAMFORD, Conn, "Android to Surpass One Billion Users Across all Devices in 2014", 2014, Disponível em <http://www.gartner.com/newsroom/id/2645115>. Acesso em 30 de maio de 2015.

[3] KOMATINENI,Satya; MACLEAN,Dave, "Pro Android 4", 2012, APRESS, ISBN:978-1-4302-3931-4, http://www.it-ebooks.info/go.php?id=657-1393515887-0abbc5d732ddac7ff07e9f6a175c8667.

[4] ACCESS DATA CORPORATION. FTK imager. Disponível em, <http://www.accessdata.com/support/product-downloads>; Acesso em 30 de maio de 2015.

[5] SQLite. Disponível em <http://www.sqlite.org>. Acesso em 27 de abril de 2015.

[6] JEON, Sangjun, et al. "A recovery method of deleted record for SQLite database. Personal and Ubiquitous Computing 16.6",2012,p 707-715.

[7] ANGLANO, Cosimo, "Forensic analysis of WhatsApp Messenger on Android smartphones", 2014,Elsevier, Digital Investigation nº11,p.201-213 http://www.journals.elsevier.com/digital-investigation.

[8] PEREIRA, Murilo Tito, "Forensic analysis of the Firefox 3 Internet history and recovery of deleted SQLite records", 2009,Elsevier, Digital Investigation nº5,p.93-103 http://www.journals.elsevier.com/digital-investigation.

Peritos Criminais Alberto Magno M. Soares, João Paulo C. de Sousa, Juliano K. M. Oya
Seção de Perícias em Informática, Instituto de Criminalística – Polícia Civil do Distrito Federal, Brasília-DF, Brasil
E-mails: alberto.magno@gmail.com, jpclaudino@gmail.com, juliano.oya@gmail.com