DOI: 10.5769/C2015005 or http://dx.doi.org/10.5769/C2015005

# Extração de dados em *smartphones* com sistema Android usando substituição da partição *recovery*

Sibelius Lellis Vieira e Adriano Rodrigues da Cruz

*Resumo*—**Os *smartphones* são telefones celulares que podem armazenar dados de grande valia para investigação de crimes. Contudo, o processo de aquisição de dados destes aparelhos pode ser inviável quando estes estiverem bloqueados e com a função de depuração USB desabilitada. O objetivo deste trabalho é descrever uma técnica e os procedimentos associados que podem ser aplicados na aquisição de dados em *smartphones* que utilizam o sistema operacional Android, estão bloqueados e com a função de depuração USB desabilitada. Ao final, ilustra-se o método de extração através da substituição da partição *recovery* e os resultados dos testes realizados com este método**.

***Palavras-Chave—Evidências Digitais, Smartphone, Android, Extração de dados.***

***Abstract*—Smartphones are mobile phones that store valuable in a crime investigation. However, the process of data acquisition can be unfeasible when the devices are locked and the USB debugging is disabled. This work describes a technique and its procedures that can be applied to mobile data acquisition in which the Android operating system is used, the mobile is locked and its USB debugging is disabled. At the end, we present a method of data extraction by replacing the recovery partition and the results of tests that employed this method.**

***Keywords—Digital evidence, Smartphone, Android, Data extraction.***

## I. INTRODUÇÃO

A computação forense é uma área da computação científica cujo objetivo é examinar dispositivos computacionais com a intenção de identificar, preservar, recuperar e apresentar evidências digitais que possam ser úteis para tipificação de crimes. Dentre estes dispositivos, destacam-se os computadores, *notebooks*, *laptops*, *tablets*, telefones celulares, máquinas fotográficas entre outros. De acordo com a estimativa da International Communication Union (ITU), o número de linhas ativas de telefones celulares, em 2014, é de quase 7 bilhões [1]. Isto torna o telefone celular um importante alvo da análise forense, uma vez que pode ser utilizado como meio para o ilícito penal (envio de mensagens de ameaça, calúnia, armazenamento de imagens relacionadas à pedofilia, etc), ou mesmo em crimes informáticos próprios [2].

Existem telefones celulares de diversas marcas e modelos. Os aparelhos construídos com maior poder de processamento e conectividade são denominados *smartphones*. Vários sistemas operacionais foram desenvolvidos para serem utilizados pelos *smartphones*. Os mais populares são o Android, iOS, Symbian, Series 40, BlackBerry, Samsung e Windows [3].

### A. *Definição do problema*

O Android se tornou o sistema operacional móvel mais popular do mundo no começo de 2011 [4]. Desta forma, é natural que a quantidade de aparelhos apreendidos para perícia com este sistema também seja proporcionalmente grande. Um dos recursos de segurança que o Android possui é permitir o bloqueio da tela do aparelho. Esse bloqueio pode ser feito de diferentes maneiras, tais como: senha numérica, senha alfanumérica, padrões etc. Outro recurso é a ativação da depuração USB, utilizada pelos desenvolvedores para acesso ao telefone através do PC durante a depuração de aplicativos e utilizada também pelos peritos criminais, para extração dos dados do usuário.

Após a apreensão de um aparelho celular, o primeiro passo para uma análise pericial é realizar a extração dos dados do aparelho para um computador, de forma a preservar o artefato original e não comprometer a integridade dos dados extraídos. A extração de dados de *smartphones* pode ser feita de forma física ou de forma lógica. A extração física é mais complexa, pois envolve *hardwares* especiais e conhecimento em eletrônica [4]. A extração lógica é feita com o uso de *softwares* que se conectam ao aparelho através do Android Debug Bridge (ADB), serviço que é executado no Android quando a função de depuração USB está ativada. Existem diversas técnicas e ferramentas para extração dos dados através da depuração USB. No entanto, quando esta opção está desabilitada e não é possível habilitá-la, o trabalho do perito é dificultado, podendo tornar a extração inviável [5].

Visto que não é possível acessar o ADB se a função de depuração USB estiver desativada, o primeiro procedimento a ser adotado pelo perito é a ativação desta opção. Contudo, em alguns casos isto não é possível. Por exemplo, se o aparelho apreendido possui bloqueio de tela ativo e a senha padrão ou PIN de desbloqueio não for conhecido, o perito fica impossibilitado de se conectar ao ADB e não é possível ter acesso ao menu do sistema para habilitar a depuração USB.

### B. *Objetivo geral*

O objetivo geral deste trabalho é analisar, propor e testar um método para extração de dados de *Smartphones* de diversas marcas, que se enquadram no cenário mencionado anteriormente, a saber, Sistema Operacional Android, com bloqueio de tela ativado e a opção de depuração USB desabilitada.

Sibelius Lellis Vieira, Professor do Departamento de Computação, PUC GOIAS e Perito Criminal do Estado de Goiás, Goiânia-GO, Brasil, sibelius@pucgoias.edu.br, e Adriano Rodrigues da Cruz, Bacharel em Ciência da Computação pela PUC GOIAS, Programador Sênior CTI/SENAC-GO, Goiânia, GO, Brasil, adrianohck@gmail.com.

## II. REFERENCIAL TEÓRICO

### A. *Computação forense*

A computação forense tem como objetivo desenvolver técnicas e ferramentas para a investigação e análise de potenciais evidências digitais [6]. Tais técnicas e ferramentas podem ser empregadas na investigação e tipificação de crimes que envolvam dispositivos computacionais. Entende-se por dispositivos computacionais qualquer aparelho capaz de processar informação, ou seja, a computação forense pode ser utilizada na investigação de computadores, *notebooks*, *laptops*, celulares, *tablets*, máquinas fotográficas digitais, televisores digitais etc.

Uma evidência digital é qualquer informação, armazenada ou transmitida por um dispositivo computacional, que pode ser utilizada como prova em um processo judicial para tipificar um crime ou estabelecer uma ligação entre um crime e sua vítima ou um crime e seu autor [7].

### B. *Android OS*

O Android é um sistema operacional móvel de código aberto, baseado no *kernel* 2.6 do Linux e gerenciado pela Open Handset Alliance, um grupo de empresas de tecnologia lideradas pelo Google. Este sistema está presente principalmente em telefones celulares. Porém, também é possível encontrá-lo em *tablets*, mini-PCs, televisores e GPS. No começo de 2011, se tornou o sistema operacional mais popular do mundo para celulares [4].

É por meio dos aplicativos que o Android oferece funcionalidades para o usuário do celular. Existem vários tipos de aplicativos, tais como jogos, redes sociais, organizadores pessoais, calendários etc. De fato, até as funcionalidades básicas do celular, tais como enviar e receber mensagens e originar e receber ligações, são aplicativos [8].

Alguns aplicativos armazenam dados do usuário. O aplicativo de telefone, por exemplo, armazena as chamadas originadas e recebidas e duração das mesmas. O aplicativo de mensagens SMS armazena as mensagens enviadas e recebidas pelo usuário [8]. Estes dados podem ser utilizados pelo perito forense na elaboração de um laudo pericial, por exemplo, e são armazenados basicamente em dois locais: interno e externo [4].

O armazenamento externo dos dados é feito por Secure Digital Card (cartão SD), geralmente formatado com o sistema de arquivos Microsoft FAT32 [4], o que facilita o trabalho do perito, uma vez que o cartão SD pode ser removido e analisado em outra máquina.

Os dados são armazenados internamente em um *chip* de memória *flash*. Além de dados de usuário, a memória também armazena arquivos de sistema. O armazenamento interno é gerenciado pela Application Program Interface (API) do Android e segue uma estrutura pré-determinada. Assim que os aplicativos são instalados, uma pasta para o aplicativo é criada no subdiretório /data/data. O navegador padrão do Android, por exemplo, armazena os dados no subdiretório /data/data/com.android.browser [4].

O Android Software Development Kit (SDK) é um conjunto de bibliotecas, documentos, utilitários e compiladores necessários para a codificação, compilação, teste e distribuição de aplicativos. O SDK contém, por exemplo, o utilitário adb usado para depuração e o utilitário fastboot, utilizado para *flash* de partições.

O SDK do Android permite que os desenvolvedores criem banco de dados SQLite para os aplicativos. O SQLite é um banco de dados leve, pequeno, de código fonte aberto e que possui as características básicas, tais como tabelas, gatilhos e visões, necessárias para a estruturação de dados [9]. Outra característica é que todos os dados são armazenados em um único arquivo *cross-platform*, ou seja, o arquivo de dados pode ser lido tanto na implementação do SQLite para Android quanto na implementação para Windows.

Estes bancos de dados são armazenados normalmente no subdiretório /data/data/<app>/databases [4]. A análise destes bancos de dados de aplicativos como telefone e mensagens permite que o perito identifique, por exemplo, chamadas originadas para determinado número ou troca de mensagens suspeitas.

### C. *Android Debug Bridge*

O Android Debug Bridge (ADB*)* é uma ferramenta do próprio SDK que permite a comunicação entre um computador e um dispositivo com Android. Ela assemelha-se ao (Secure Shell) SSH do Linux. Entre as várias utilidades desta ferramenta, destacam-se: instalação de aplicativos, execução de comandos diretamente no *shell* do dispositivo e cópia de arquivos entre o computador e o dispositivo e vice-versa. O ADB possui três componentes [10]:

a) um utilitário de linha de comando, que é executado pelo usuário para emitir os comandos;
b) um processo servidor, que executa no mesmo computador do usuário e é responsável por receber os comandos do utilitário e transmiti-los para o dispositivo;
c) um serviço, que executa como processo de segundo plano no dispositivo e é responsável por receber os comandos transmitidos pelo processo servidor.

O serviço do ADB no dispositivo fica ativo somente se a opção *Depuração USB* estiver habilitada. Portanto, se esta opção estiver desabilitada, não é possível utilizar o ADB para comunicar-se com o aparelho. A Tabela I lista alguns comandos do cliente ADB.

TABELA I. COMANDOS DO CLIENTE ADB

| **Comando** | **Descrição** |
|---|---|
| adb shell | Inicia um *prompt* de comando no dispositivo. |
| adb push <local> <remoto> | Copia o arquivo especificado do computador local para o dispositivo. |
| adb pull <remoto> <local> | Copia o arquivo remoto especificado do dispositivo para o computador local. |
| adb install app.apk | Instala um aplicativo no aparelho. |

Fonte: [10].

Caso o aparelho apreendido esteja com a opção de depuração USB desabilitada, o perito pode habilitá-la no menu de configurações de opções do desenvolvedor, conforme mostrado na Figura 1. Quando o aparelho possui a tela

bloqueada, primeiro é necessário desbloqueá-la para acessar o menu de configurações. Se o desbloqueio não for possível, não há como acessar tal tela.

Após habilitar a depuração USB, o perito deve conectar o cabo USB ao telefone e tentar se conectar ao aparelho usando o ADB. Caso obtenha sucesso, alguns aplicativos podem ser instalados ou até mesmo o comando adb *pull* pode ser usado para extrair os dados do aparelho para a máquina do examinador.

Em uma situação em que o telefone possua bloqueio de tela ativado, o perito ainda pode tentar conectar o cabo USB no aparelho para verificar se o usuário deixou a opção de depuração USB ativada.


Fig. 1. Ativação da depuração USB.

## D. *Senhas, padrões e PIN Lock*

O Android fornece para o usuário diversas maneiras de configurar o bloqueio de tela. Algumas são básicas e úteis apenas para o travamento do teclado. Já outras são mais sofisticadas e permitem o desbloqueio da tela apenas para quem conhece a senha, padrão ou Personal Identification Number (PIN) [11].

O PIN é essencialmente numérico e não pode ser combinado com letras e outros caracteres. Também é possível bloquear a tela por um padrão, desenhando ligações entre pontos em uma matriz. Outra forma é usar uma senha tradicional, combinando letras e números. A Figura 2(a) exibe a tela de desbloqueio por PIN. A Figura 2(b) exibe a tela de desbloqueio por padrão. A Figura 2(c) exibe a tela de desbloqueio por senha.


(a) (b) (c)

Fig. 2. Bloqueios de tela.

Caso não seja possível realizar o desbloqueio da tela, somente chamadas de emergência podem ser feitas. Também não é possível acessar a tela de configuração do sistema. Existem alguns aplicativos que tentam realizar a quebra do bloqueio de tela. Porém, eles só podem ser instalados se a opção de depuração USB estiver habilitada.

## E. *Partições típicas*

Uma partição é uma divisão lógica de um dispositivo de armazenamento de dados. Embora o fabricante do aparelho possa modificar o esquema de partições padrão, as partições mostradas na Tabela II estão presentes na maioria dos *smartphones* com Android. As partições de sistema, dados do usuário, *boot*, *cache* e *recovery* tipicamente estão presentes nos aparelhos [12]. Os aplicativos que armazenam dados do usuário na memória interna do telefone gravam estes dados na partição /data, no caminho /data/data/<app> [4].

TABELA II. TABELA DE PARTIÇÕES TÍPICAS DO ANDROID

| Caminho | Nome | Sistema de Arquivos | Ponto de Montagem | Descrição |
|---|---|---|---|---|
| /dev/mtd/mtd0 | pds | yaffs2 | /config | Dados de configurações |
| /dev/mtd/mtdl | misc | - | N/A | Memória |
| /dev/mtd/mtd2 | boot | Bootimg | N/A | Inicializável (partição padrão de *boot*) |
| /dev/mtd/mtd3 | recovery | Bootimg | N/A | Inicializável (partição *recovery*) |
| /dev/mtd/mtd4 | system | yaffs2 | /system | Arquivos de sistemas e aplicativos |
| /dev/mtd/mtd5 | cache | yaffs2 | /cache | Arquivos de *cache* |
| /dev/mtd/mtd6 | userdata | yaffs2 | /data | Dados do usuário |
| /dev/mtd/mtd7 | kpanic | - | N/A | Logs de falhas |

Fonte: [12].

## F. *Modos de inicialização*

Os aparelhos podem ser inicializados de diferentes modos. Alguns fabricantes disponibilizam *softwares* específicos para isto. Porém, na maioria dos dispositivos o modo de inicialização pode ser alterado através de uma combinação de teclas enquanto o dispositivo está sendo ligado.

Embora o Android forneça para os desenvolvedores de aplicativos certo nível de abstração de *hardware*, existe uma grande diversidade de fabricantes e modelos. Alguns aparelhos possuem grande quantidade de teclas. Já outros possuem apenas uma ou duas teclas. Justamente por causa desta diversidade, nem sempre a mesma combinação de teclas para alternar o modo de inicialização funciona em modelos diferentes. A Tabela III exibe as combinações de teclas para alternar o modo de inicialização de alguns modelos [12].

TABELA III. MODOS DE INICIALIZAÇÃO DE ALGUNS APARELHOS COM ANDROID

| Modelo | Modo | Combinação | Descrição |
|---|---|---|---|
| Motorola Droid | *Flash* | D-Pad UP + power | Modo que permite *flashing* via *RSD Lite* |
| Motorola Droid | *Flash* | camera +power | Modo que permite *flashing* via *RSD Lite* |
| Motorola Droid | *Recovery* | power + x | *Boot* na partição *recovery* (após, camera + volup |

| | | | |
|---|---|---|---|
| | | | para mostrar menu) |
| HTC G1 | *Flash* | power + back | Modo *Fastboot* |
| HTC G1 | *Flash* | power + câmera | Modo de *boot* (‘*back*’ para trocar para *Fastboot*) |
| HTC G1 | *Recovery* | power + home | *Boot* na partição *recovery* |
| Samsung Captivate | *Flash* | volup + voldn (então insira USB) | *Boot* no modo “Samsung *force download*” |
| Samsung Captivate | *Recovery* | power + volup + voldn | *Boot* na partição *recovery* |
| Samsung Galaxy Tab | *Flash* | power +voldn | *Boot* no modo “Samsung *force download*” |
| Samsung Galaxy Tab | *Recovery* | power + volup | *Boot* na partição *recovery* |

Fonte: [12].

Quando o aparelho é ligado normalmente sem nenhuma combinação de teclas, ele está no modo normal. Neste modo, o sistema principal, comumente instalado na partição *system*, é iniciado. Para que o aparelho seja inicializado em um modo diferente, é necessário ligá-lo, pressionando a combinação de teclas corresponde ao modo desejado. Um modo especial de inicialização é chamado de modo de recuperação ou modo *recovery*. Ao ligar o aparelho, pressionando a combinação de teclas para inicialização no modo *recovery*, os arquivos da partição de mesmo nome, partição *recovery*, são carregados [13].

A inicialização no modo *recovery* permite ao usuário formatar o dispositivo, restaurar as configurações de fábrica, limpar dados de *cache* e realizar tarefas de manutenção. Esse modo de inicialização também é utilizado pelo próprio Android para aplicação de pacotes de atualização [4].

A partição *recovery* contém os arquivos de inicialização para o modo *recovery*. Ela possui seu próprio *kernel* Linux, separado do *kernel* do sistema principal do Android [14] e pode ser iniciada mesmo que a instalação principal do sistema esteja com problemas. O modo *recovery* padrão de fábrica normalmente oferece apenas funcionalidades básicas e sem acesso ao ADB [4].

Esta partição geralmente possui um tamanho pequeno e seu dispositivo associado pode ser diferente, dependendo do modelo e fabricante. Detalhes sobre esta partição podem ser vistos examinando o arquivo /proc/mtd [4], conforme mostra a Figura 3.

```
ahoog@ubuntu:~$ adb shell cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00040000 00020000 "misc"
mtd1: 00500000 00020000 "recovery"
mtd2: 00280000 00020000 "boot"
mtd3: 04380000 00020000 "system"
mtd4: 04380000 00020000 "cache"
mtd5: 04ac0000 00020000 "userdata"
```

Fonte: [4].

Fig. 3. Detalhes da partição *recovery*.

### G. *A extração de dados*

Existem vários métodos para extração de dados. Estes métodos são basicamente divididos entre extração física e extração lógica [15].

De acordo com [4], a extração física pode ser classificada em extração por *hardware* e por *software*. A extração por *hardware* é realizada utilizando duas técnicas conhecidas como *chip-off* e Joint Test Action Group (JTAG). Este tipo de extração só é útil quando os dados armazenados na memória *flash* não estão criptografados. Do contrário, os dados podem até ser extraídos, mas não serão adequadamente utilizados. Já a extração por *software* utiliza a técnica de executar *software* no aparelho, fornecendo uma imagem física completa das partições [4]. É a técnica normalmente empregada em diversas ferramentas de extração de dados.

As técnicas de extração lógica dos dados são menos destrutivas ao aparelho, pois não necessitam de alterações de *hardware* no dispositivo a ser analisado. Segundo [4], as técnicas lógicas de extração de dados apenas necessitam que a opção de depuração USB esteja habilitada.

A técnica conhecida como *ADB pull* utiliza o comando *pull* do ADB para realizar uma cópia recursiva dos diretórios e arquivos a serem analisados do aparelho para a máquina do perito. Apesar de simples, essa técnica nem sempre é viável, pois na maioria dos casos, o usuário sob o qual o *ADB* é executado não possui permissão de leitura nos diretórios dos aplicativos. A partição de maior interesse é a /data, onde residem todos os arquivos do usuário. Se o *ADB* possui acesso de *root*, esta partição pode ser inteiramente copiada.

O aplicativo AFLogical também pode ser utilizado para extração dos dados [9]. Este aplicativo foi desenvolvido pela empresa viaForensics e pode ser instalado no aparelho através do ADB. Uma vez instalado, ele extrai os dados de diversos aplicativos como SMS, contatos, registros de chamada, Facebook, *browser*, entre outros. Os dados extraídos são armazenados no cartão SD, em formato csv [16]

Outra maneira de analisar os dados é examinando o cartão SD externo. Diversos aplicativos permitem que o usuário realize um *backup* dos dados para a memória externa. O cartão SD pode ser removido do aparelho e analisado na máquina do perito. O problema é que nem sempre os dados do *backup* estarão atualizados.

Caso o bloqueio de tela esteja ativo, não é possível realizar *backup* dos aplicativos. Se a opção de depuração USB estiver desabilitada, também não é possível instalar aplicativos ou executar comandos. Neste cenário, ainda existe outra possibilidade: a substituição da partição *recovery*.

### H. *A substituição da partição recovery*

A troca da partição *recovery* padrão pode ser realizada em aparelhos cujo *boot loader* seja compatível com o modo *fastboot* ou ofereça a opção de substituição de partições (também conhecida como *flash* de partições). Existem diversas partições *recovery* modificadas que podem ser utilizadas para substituição da partição padrão. A maioria destas partições permite acesso via ADB como *root*. Desta forma, o aparelho pode ser inicializado no modo *recovery* e o perito pode utilizar, por exemplo, a técnica de *ADB pull* para extração dos dados.

O *boot loader* é um pequeno programa responsável por carregar o sistema operacional. No ambiente Linux, os principais *boot loaders*, GRUB e LILO, estão presentes na maioria das distribuições. Em dispositivos com Android, o *boot loader* é responsável por carregar o sistema operacional Android ou a partição *recovery* [14]. Alguns fabricantes desenvolvem seus próprios b*oot loaders* e *softwares* para

interagir com eles. Uma das funções destes *softwares* é permitir a substituição de partições do aparelho [4]. Alguns exemplos destes *softwares* são o Motorola RSD Lite, o Samsung Odin Multiloader e o LG Flashtool.

O processo de substituição de partições é específico de cada aparelho. A imagem da partição a ser substituída deve ser compatível com o aparelho em questão e nem sempre a mesma imagem pode utilizada em modelos diferentes.

O *boot loader* pode estar travado ou não. Um *boot loader* travado somente carrega sistema operacional com assinatura válida. Da mesma forma, não é possível instalar uma imagem personalizada na partição *recovery*. Nestes casos, é necessário destravar o próprio *boot loader* primeiro. Esse procedimento varia de acordo com o aparelho e pode violar a garantia do fabricante [17].

### I. *O modo fastboot*

O modo *fastboot* foi inicialmente implementado em um Android Developer Phone (ADP), fabricado pela HTC. Neste modo, é possível usar o utilitário de linha de comando *fastboot*, que já vem compilado com a SDK do Android, para gravação de imagens em partições do aparelho. Para utilizar o *fastboot*, é necessário que o *boot loader* do aparelho seja compatível com o modo *fastboot* e esteja destravado. Então o aparelho deve ser conectado na porta USB e ligado (ou reiniciado) segurando-se as teclas VOLDN e BACK. Essa combinação pode ser diferente dependendo do aparelho. Ao entrar neste modo, é mostrada na tela do aparelho a palavra *FASTBOOT* [4]. Neste momento, é possível emitir comandos para listar aparelhos conectados, como mostrado na Figura 4. Depois da confirmação de que o aparelho realmente está conectado, é possível fazer a gravação de novas partições.

```
C:\Users\Adriano>fastboot devices
0910D4D11800F00C        fastboot

C:\Users\Adriano>
```

Fig. 4. Comando *fastboot devices*.

Existem várias imagens da partição *recovery* modificadas que podem ser utilizadas em substituição à partição de fábrica. Na escolha de uma imagem apropriada, deve ser levado em consideração se a nova imagem permite ou não acesso como *root* via ADB. As mais populares são:

a) ClockworkMod: escrita por Koush Dutta, é baseada na imagem da partição *recovery* do Android 2.1. Possui diversas opções como *backup*, restauração, atualização do aparelho através de arquivos .zip e acesso via ADB habilitado [13];
b) TWRP: Team Win Recovery Project possui, além das opções padrão, funções como *backup* e restauração e acesso via ADB habilitado. Sua interface é sensível ao toque e é personalizável [18].

## III. MATERIAIS E MÉTODOS

Para a realização dos experimentos deste trabalho, quatro modelos diferentes de *smartphones* foram utilizados. As especificações de cada aparelho foram descritas em cada experimento. Além disso, acessórios como cabos USB e carregadores compatíveis com cada modelo de aparelho foram necessários. Para a elaboração do método de extração de dados proposto, optou-se incialmente pela realização de uma ampla pesquisa bibliográfica sobre perícia forense em dispositivos móveis. A partir da fase inicial de pesquisa, foi possível encontrar estudos que tratavam especificamente do tema proposto por este trabalho e analisar diversas características do sistema operacional Android e elaborar um método. O método foi elaborado baseando-se nos princípios gerais da computação forense. Todavia, durante a pesquisa bibliográfica foi verificado que o processo de análise pericial em *smartphones* é sempre mais invasivo do que análises de computadores pessoais. Desta forma, o método proposto foi criado no intuído de preservar ao máximo a integridade dos dados, apesar de ser intrusivo.

Após a proposição do método, diversos experimentos foram realizados. Com estes experimentos foi possível constatar a aplicabilidade e viabilidade do método proposto. Também foi possível identificar cenários nos quais a aplicação do método é inviável, ora por questões de incompatibilidade, ora por questões de restrições do aparelho.

Esta seção é dedicada a descrever os experimentos realizados com o método proposto. Para tal foram utilizados quatro aparelhos de telefonia celular no seguinte cenário: sistema operacional Android, bloqueio de tela ativo com padrão, senha ou PIN de desbloqueio desconhecido e opção de depuração USB desabilitada.

### A. *O método proposto*

A substituição da partição *recovery* padrão como método para extração dos dados do usuário é utilizada neste trabalho. A partição *recovery* original é substituída por outra que possibilite o acesso através do ADB, permitindo, desta forma, que o perito examinador seja capaz de realizar a extração dos dados do aparelho.

Contudo, a decisão de aplicação ou não deste método deve ser tomada pelo perito levando em consideração diversos aspectos, entre eles, a possibilidade da restauração das configurações de fábrica. A substituição da partição *recovery* pode causar incompatibilidade com o sistema Android instalado, levando-o a não inicializar novamente. Embora esta situação não prejudique a perícia em si e nem comprometa a integridade dos dados do usuário, esta possibilidade deve ser analisada pelo perito, pois, neste caso, a única maneira de deixar o celular utilizável novamente é restaurando a imagem original do Android para o aparelho, o que leva à perda de todos os aplicativos, históricos e configurações do usuário. Neste trabalho os experimentos foram realizados utilizando tanto a imagem ClockworkMod quanto a TWRP.

### B. *Instalação do Software Development Kit (SDK)*

A instalação do SDK no sistema operacional Windows 7 pode ser feita baixando-se o arquivo de instalação direto do portal do desenvolvedor para Android (https://developer.android.com/sdk/). Após obter o arquivo de instalação é necessário executá-lo, aceitar termos de uso e confirmar os locais de instalação.

Um ponto chave da instalação do Android SDK é a escolha correta do nível da API (API *Level*). A cada alteração no *framework* de desenvolvimento são acrescentadas e removidas funções, suporte a novas plataformas, entre outras. Para solucionar problemas de compatibilidade de aplicativos, foi criado o conceito de nível de API. Uma determinada versão do Android suporta instalação de aplicativos criados

até certo nível de API. Aplicativos criados com níveis de API mais recentes não podem ser instalados em versões antigas do Android [19]. A Tabela IV relaciona a versão do Android ao nível de API suportado.

TABELA IV. VERSÕES DO ANDROID E SEUS NÍVEIS DE API

| Versão | Nível da API | Codinome |
|---|---|---|
| Android 4.4 | 19 | KITKAT |
| Android 4.3 | 18 | JELLY_BEAN_MR2 |
| Android 4.2.2<br>Android 4.2 | 17 | JELLY_BEAN_MR1 |
| Android 4.1.1<br>Android 4.1 | 16 | JELLY_BEAN |
| Android 4.0.4<br>Android 4.0.3 | 15 | ICE_CREAM_SANDWICH_MR1 |
| Android 4.0.2<br>Android 4.0.1<br>Android 4.0 | 14 | ICE_CREAM_SANDWICH |
| Android 3.2 | 13 | HONEYCOMB_MR2 |
| Android 3.1.x | 12 | HONEYCOMB_MR1 |
| Android 3.0.x | 11 | HONEYCOMB |
| Android 2.3.4<br>Android 2.3.3 | 10 | GINGERBREAD_MR1 |
| Android 2.3.2<br>Android 2.3.1<br>Android 2.3 | 9 | GINGERBREAD |
| Android 2.2.x | 8 | FROYO |
| Android 2.1.x | 7 | ECLAIR_MR1 |
| Android 2.0.1 | 6 | ECLAIR_0_1 |
| Android 2.0 | 5 | ECLAIR |
| Android 1.6 | 4 | DONUT |
| Android 1.5 | 3 | CUPCAKE |
| Android 1.1 | 2 | BASE_1_1 |
| Android 1.0 | 1 | BASE |

Fonte: [19].

Após a instalação, é possível verificar qual o nível da API está instalado usando o programa SDK Manager, instalado junto com o SDK. Nele também é possível acrescentar ou remover outros níveis de API, como ilustrado na Figura 5.

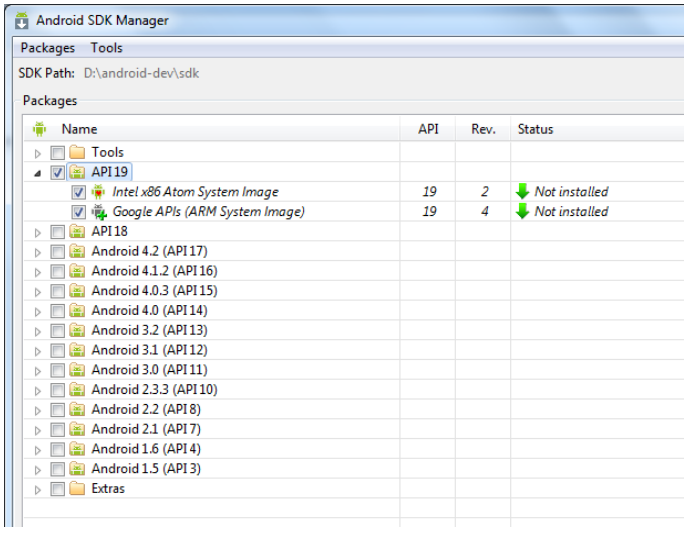


Fig. 5. Android SDK Manager.

### C. *Procedimento de extração de dados do Samsung Galaxy S2 (GT-I9100)*

O GT-I9100, desenvolvido pela companhia sul coreana Samsung e lançado em abril de 2011 com o nome comercial Samsung Galaxy S II, possui processador Dual-Core Cortex A9 de 1.2 Ghz, *chipset* Exynos 4210, memória RAM de 1 GB e armazenamento interno de 16 GB ou 32 GB. Sua versão original vem com Android 2.3.4 (Gingerbread) que pode ser atualizada para a versão 4.1 (Jelly Bean).

O perito deve fazer uma avaliação inicial do aparelho para decidir quais procedimentos devem ser adotados, verificando a presença de cartão SD, cartão SIM e bateria. Para que seja possível realizar o acesso via ADB ao telefone, os *drivers* compatíveis com o modelo GT-I9100 devem ser instalados. No experimento realizado, os *drivers* foram obtidos do *site* da própria Samsung e instalados em um computador com Windows 7. Após a constatação de que a opção de depuração USB está desabilitada, o bloqueio de tela está ativo e o padrão, PIN ou senha de desbloqueio não é conhecido, o próximo passo é inicializar o celular no modo *recovery*.

A inicialização do GT-I9100 no modo *recovery* é feita pressionando simultaneamente as teclas de Volume (+), *Home* e *Power*, com o aparelho desligado. No experimento realizado, a tela do aparelho apresentou o modo *recovery* padrão, como mostrado na Figura 6.


Fig. 6. GT-I9100: tela do modo *recovery* padrão.

Em seguida, o cabo USB deve ser conectado ao aparelho e o comando adb *devices* executado, como ilustrado na Figura 7. Caso nenhum aparelho seja listado, a partição *recovery* presente não permite acesso ADB.

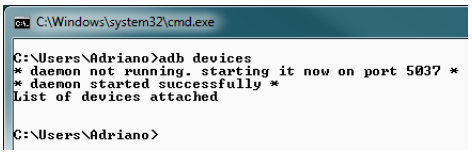


Fig. 7. GT:I9100: Comando `adb devices`.

Neste experimento, a substituição da partição *recovery* foi feita utilizando o *software* Odin3 [13]. Para usar este *software*, o celular deve estar no modo *download* (também conhecido como Odin *mode*). A inicialização no modo *download* é feita pressionando as teclas Volume (-), *Home* e *Power* com o aparelho desligado [20]. Ao ligar o aparelho segurando estas teclas, é mostrada uma tela de confirmação. Em seguida, a tecla de Volume (+) deve ser pressionada e o telefone entrará no modo *download*.


(a) (b)

Fig. 8. GT-I9100: Modo *download*.

A seguir, o celular deve ser conectado à USB e o *software* Odin3 v.1.85 inicializado. Ao iniciar, o Odin3 exibe a mensagem "Added", indicando que o aparelho foi reconhecido [13]. Caso a aparelho não tivesse sido reconhecido, os *drivers* deveriam ser reinstalados.

A versão compatível com o GT-I9100 do ClockworkMod utilizada neste experimento foi a 6.0.2.9. Para fazer a substituição, o arquivo do ClockworkMod deve ser selecionado na opção PDA do Odin3 e depois, o botão *Start* deve ser clicado. Quando o procedimento finaliza, o Odin3 exibe a mensagem "PASS", como pode ser observado na Figura 9.

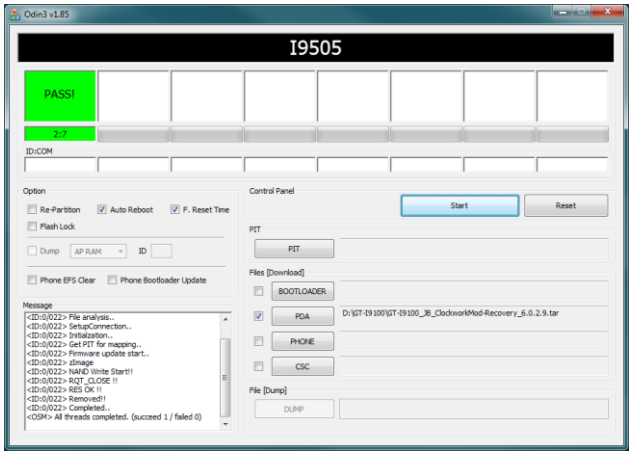


Fig. 9. GT-I9100: conclusão da substituição da partição *recovery*.

Após a conclusão deste procedimento, o aparelho deve ser imediatamente desconectado da USB e a bateria removida. Isto foi feito porque algumas versões do Android para aparelhos da marca Samsung, no momento da inicialização, restauram a partição *recovery* para a versão original caso esta partição tenha sido modificada. Em seguida, ele foi ligado novamente no modo *recovery*. Neste momento, a tela do ClockworkMod é apresentada indicando que o procedimento ocorreu com sucesso.

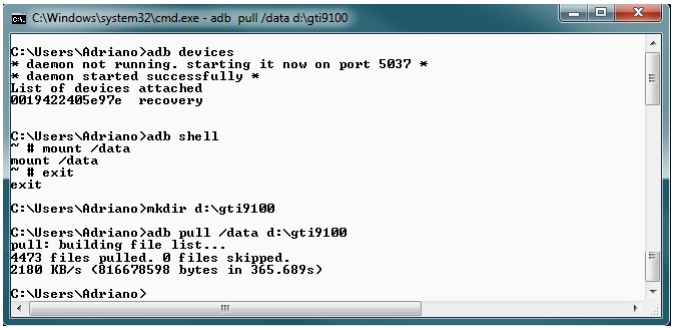


Fig. 10. GT-I9100: comando `adb pull /data`.

A próxima etapa é fazer a extração dos dados da partição /data. Para isto, é necessário acessar o aparelho via ADB e montar a partição com o comando *mount* /data. Em seguida, um diretório deve ser criado na máquina do examinador e os dados copiados através do comando adb *pull*. Esses comandos foram executados no GT-I9100 do experimento e o resultado mostrado na Figura 10.

### *D. Procedimento de extração de dados do LG Optimus 3D (P920h)*

O P920h, denominado comercialmente de LG Optimus 3D, foi desenvolvido pela companhia sul coreana LG Eletronics em 2011. Sua principal característica é a tela que pode exibir imagens em terceira dimensão sem o uso de óculos especiais. Esse modelo vem com memória Random Access Memory (RAM) de 512 MB, processador ARM Cortex A9 de 1 GHz Dual Core, *chipset* OMAP4430 desenvolvido pela Texas Instruments, tela de 4.3 polegadas e Android 2.2 [21].

A inicialização do P920h no modo *recovery* é feita pressionando simultaneamente as teclas *Power*, Volume (-) e 3D, com o aparelho desligado. Assim que o aparelho for iniciado, a imagem da partição *recovery* é carregada.

No caso em tela, a substituição da partição *recovery* original do P920h foi necessária, pois esta partição não permite conexão via ADB. Essa substituição pode ser feita utilizando a ferramenta LGTool (http://www.lgtool.net/) ou utilizando o *fastboot*. Neste experimento, a substituição foi feita com o utilitário *fastboot*, pois a ferramenta LGTool é proprietária e necessita de ativação.

Embora o *boot loader* padrão do P920h forneça suporte, não existe uma combinação de teclas documentada para iniciar o telefone no modo *fastboot*. Para isto, foi utilizado o *software* Omap4Boot-for-optimus, desenvolvido pela comunidade XDA Developers e de código fonte aberto [22]. Este *software* foi criado para ser utilizado em procedimentos de recuperação de celulares com *chipset* OMAP44XX que não estejam inicializando, permitindo que o usuário consiga fazer o *boot* tanto no modo *download* como no modo *fasboot*. Neste trabalho, a ClockworkMod versão 6.0.1.9 compatível com o P920h foi utilizada.

Em seguida, o celular deve ser ligado no modo *fastboot,* sendo o *software* Omap4Boot-for-optimus utilizado nesse momento. Após a descompactação, o arquivo start_fastboot.bat deve ser executado. Então é apresentada uma tela solicitando qual o modelo do telefone. A opção 2 (Optimus 3D P920) deve ser selecionada. Agora, a mesma tela apresenta uma mensagem informando que o telefone deve ser conectado a USB, conforme ilustra a Figura 11.

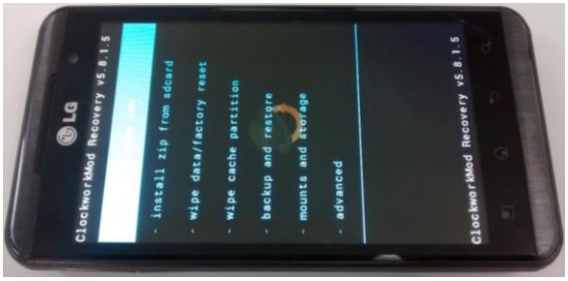


Fig. 11. P920h apresentando o *recovery mode* com ClockworkMod.

Nesse momento, o telefone desligado deve ser conectado à USB sem a bateria. Então, o *software* identifica o dispositivo OMAP4430, instala os *drivers* necessários e para em um segundo estágio. Só então a bateria deve ser acoplada novamente. Se o Windows não conseguir encontrar os *drivers* na pasta do Omap4Boot e instalar automaticamente, a instalação deve ser feita de forma manual e o procedimento repetido.

Feito isso, a tela do celular deve apresentar o logotipo da LG em tom de cinza e o texto "fastboot v0.5" no canto superior esquerdo. Esse mesmo procedimento pode ser feito para ligar o telefone no modo *download*, bastando segurar a tecla Volume(+).

A próxima etapa é a substituição da partição em si. Isso pode ser feito com o comando *fastboot flash recovery* recovery.img. Este comando substitui a partição *recovery* pela imagem recovery.img fornecida. Por fim, o telefone deve ser ligado novamente no modo *recovery*. Se todas as etapas foram concluídas com sucesso, a tela do aparelho deve apresentar a interface da ClockworkMod, como ilustra a Figura 12.

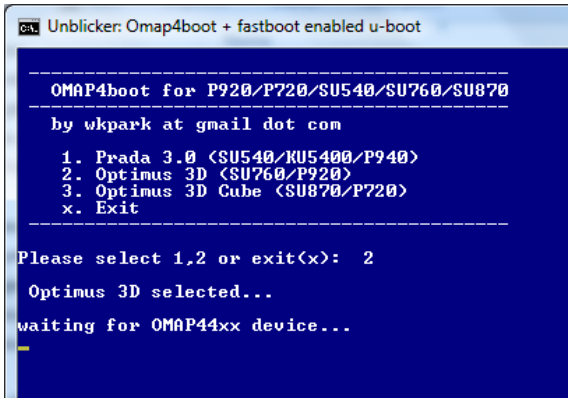


Fig. 12. Software OMAP4Boot aguardando conexão com o P920h

O método de extração de dados utilizado neste experimento também foi o adb *pull*, de forma similar ao aparelho GT-I9100.

### *E. Procedimento de extração de dados Motorola Moto G Dual SIM (XT1033)*

O XT1033 foi desenvolvido pela Motorola e lançado no mercado em janeiro de 2014. Possui processador Quad-core Cortex-A7 de 1.2 Ghz, *chipset* Qualcomm MSM8226 Snapdragon 400, memória RAM de 1GB e armazenamento interno de 8 GB ou 16 GB. Sua versão original vem com Android 4.3 (Jelly Bean).

O XT1033 não aceita cartões de memória externos. Logo, todos os dados do aparelho são armazenados na memória interna. Então, na análise inicial do aparelho o perito deve observar a presenta de cartões SIM. No aparelho deste experimento, dois SIMs foram encontrados, pois este aparelho possui a tecnologia Dual SIM.

A partição *recovery* do XT1033 pode ser substituída através do *fastboot*. Porém, esta substituição só pode ser feita em aparelhos cujo *boot loader* esteja destravado. Caso o *boot loader* esteja travado, é necessário destravá-lo primeiro. Esse procedimento pode ser feito através de um código obtido no *site* da própria Motorola [23]. Para obter este código, é necessário digitar os dados de destravamento do telefone no *site* da Motorola. Estes dados são conseguidos através do próprio *fastboot*.

### *F. Procedimento de extração de dados Motorola Moto G 2ª Geração (XT1069)*

O XT1069, também conhecido como Motorola Moto G2, foi desenvolvido pela Motorola e lançado no mercado em setembro de 2014. Possui processador e *chipset* Qualcomm Snapdragon 400 MSM8226 / ARM Cortex-A7 de 1.2 Ghz, memória RAM de 1GB e armazenamento interno de 16 GB. Sua versão original vem com Android 4.4.4 (KitKat), atualizável para Android 5.0 (Lollipop). Todas as considerações anteriores feitas ao XT1033 em relação à substituição da partição recovery também se aplicam ao XT1069.

## IV. RESULTADOS E DISCUSSÃO

### *A. Resultados para o Samsung Galaxy S2 (GT-I9100)*

Ao final deste experimento, os dados da partição /data e os dados do cartão SD interno (partição /sdcard) foram armazenados em diretórios da máquina do examinador, sendo a extração efetivada utilizando o adb *pull*.

No experimento realizado, após a extração dos dados, o aparelho foi ligado segurando-se somente a tecla *Power*. Durante a inicialização, o logotipo da Samsung foi mostrado na tela, juntamente com um ícone de advertência. O aparelho ficou travado nesta tela por cerca de 10 segundos. Após esse período, ele desligou. A tentativa de inicialização normal do aparelho foi repetida por mais três vezes, sem sucesso.

Desta forma, é possível concluir que em aparelhos do modelo GT-I9100, a substituição da partição *recovery* pode afetar também o sistema operacional Android, levando o aparelho a não inicializar normalmente. No caso do celular deste experimento, a imagem original do sistema Android foi restaurada usando o próprio Odin3. Após a restauração, o celular iniciou normalmente, porém com as configurações originais de fábrica. Todos os aplicativos instalados, contatos, históricos e mensagens se perderam.

Esta é uma hipótese que deve ser levada em consideração na decisão do perito em fazer ou não a substituição da partição *recovery*. Embora isto não prejudique a análise pericial em si, uma vez que os dados já tinham sido copiados para a máquina, após o procedimento de substituição, o celular pode se tornar inoperantese o sistema Android instalado for incompatível com a nova partição *recovery*. E a maneira encontrada neste trabalho para deixar o celular utilizável novamente envolve a restauração da imagem original do Android, o que leva à perda dos dados do aparelho.

### *B. Resultados para o LG Optimus 3D (P920h)*

Conforme apresentado na seção III.D, a substituição da partição *recovery* foi realizada, o que permitiu a inicialização do aparelho no modo *recovery* com o ClockworkMod, e a subsequente extração de dados via adb *pull*, conectando-se ao aparelho através do comando adb *shell*. A partir daí, todos os arquivos da partição de dados puderam ser copiados para a máquina do examinador.

### C. *Resultados para o Motorola Moto G Dual SIM (XT1033)*

No experimento realizado com o XT1033, o *boot loader* estava travado. Para fazer esta verificação, o aparelho foi ligado no modo *fastboot*, pressionando simultaneamente as teclas Volume (-) e *Power* com o aparelho desligado, por 3 segundos. Assim que estas teclas foram liberadas, a tela do modo *fastboot* foi mostrada. Nela foi possível verificar o texto "*Device is LOCKED*", conforme mostrado na Figura 13.


Fig. 13. XT1033: tela do modo fastboot.

Neste momento o telefone estava no modo *fastboot* mostrando a informação de que o *boot loader* estava travado. Então, o cabo USB foi conectado ao telefone e ligado ao computador e o comando *fastboot oem get_unlock_data* executado (Figura 14). O retorno da execução deste comando é o código de verificação a ser passado para o site da Motorola.

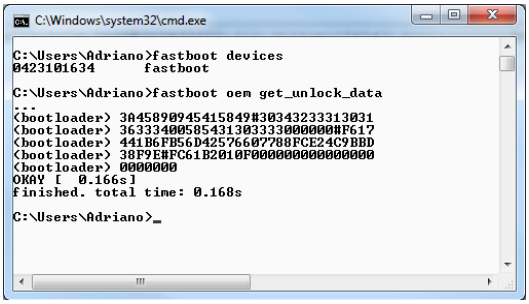


Fig. 14. XT1033: execução do comando para obter unlock_data.

O código de verificação é tratado de forma a não conter espaços ou informações adicionais. Ao entrar com o código no *site* da Motorola, a mensagem "*Your device does not qualify for bootloader unlocking.*" foi mostrada, conforme ilustra a Figura 15. Desta forma, conclui-se que este aparelho não pode ter seu *boot loader* destravado.

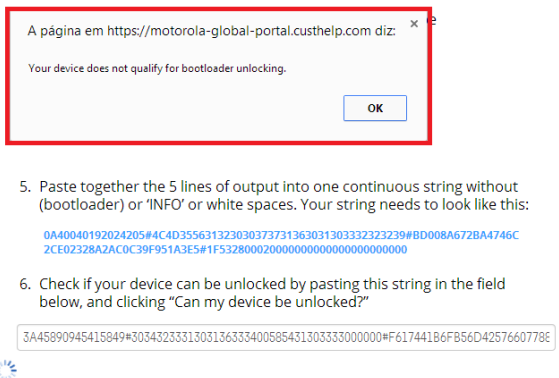


Fig. 15. XT1033: mensagem indicando dispositivo *unlockable*.

Sem o código de destravamento, não foi possível destravar o *boot loader*. Logo, também não foi possível fazer a substituição da partição *recovery* do XT1033 neste experimento. Isto, porém, não invalida a aplicação deste método em modelos cujo *boot loader* esteja destravado.

### D. *Resultados para o Motorola Moto G 2ª Geração (XT1069)*

Assim como ocorreu com o XT1033, o *boot loader* estava travado. Procedeu-se, então, à tentativa de destravamento, através da execução do comando *fastboot oem get_unlock_data* e a obtenção do código de verificação. Desta vez, a entrada do código de verificação no *site* da Motorola gerou uma mensagem eletrônica enviada ao usuário do aparelho, com o código de destravamento do *bootloader*, conforme pode ser observado parcialmente na Figura 16.


Fig. 16. Mensagem eletrônica retornando o código de destravamento.

O destravamento é efetivado executando o comando *fastboot oem unlock* UNIQUE_KEY, sendo UNIQUE_KEY o código retornado na mensagem de correio eletrônico. Uma vez destravado, a substituição da partição em si pôde ser realizada também utilizando o comando *fastboot flash recovery* recovery.img. O arquivo imagem compatível com o XT1069 do TWRP utilizado neste experimento foi a twrp-2.8.6.0-titan.img. Por fim, o telefone deve ser ligado novamente no modo *recovery*. Após a conclusão das etapas com sucesso, a tela do aparelho apresentou a interface da TWRP, como ilustra a Figura 17. A opção de backup permite a realização do backup do aparelho, que pode ser copiado para o computador através do comando adb *pull*. Entretanto, o destravamento do bootloader acarretou o retorno do aparelho às configurações originais de fábrica, implicando na perda das informações anteriormente presentes.

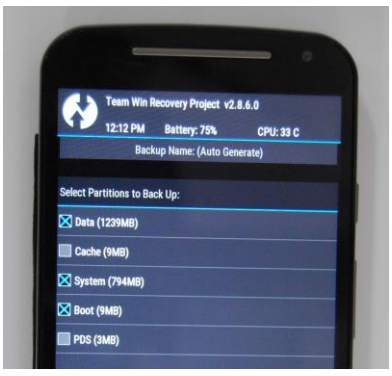


Fig. 17. XT1069: tela de *backup* do modo *recovery* TWRP.

## V. CONCLUSÕES

Este trabalho teve como principal objetivo propor, analisar, e demonstrar a viabilidade de se utilizar um método para o problema de extração de dados de telefones *smartphones* com sistema operacional Android, bloqueio de tela ativo e senha, PIN ou padrão de desbloqueio desconhecido e função de depuração USB desabilitada.

A hipótese de trabalho pressupõe que a substituição da partição *recovery* padrão por outra que possibilite o acesso via ADB possibilita a extração dos dados do usuário, sem comprometer a integridade das informações extraídas.

Para verificação da hipótese proposta, quatro experimentos foram realizados. O primeiro experimento foi feito com um Samsung Galaxy S2. Neste modelo, foi possível realizar a extração dos dados. Entretanto, a substituição da partição *recovery* afetou a instalação do sistema principal, levando a aparelho a não inicializar novamente. O segundo experimento apresentado foi relativo a um aparelho LG Optimus 3D. A partição *recovery* padrão foi substituída e os dados extraídos com sucesso. O terceiro experimento apresentado foi feito com um aparelho Morotola Moto G Dual SIM. Neste aparelho, não foi possível realizar a substituição da partição *recovery*, uma vez que o *boot loader* estava travado e não foi possível destravá-lo. Por fim, o quarto experimento, também com um aparelho Motorola com o *boot loader* travado, embora tenha permitido a substituição da partição *recovery*, não garantiu a extração com sucesso dos dados, uma vez que foram apagados no processo de destravamento do *boot loader*.

Com base nos experimentos realizados neste trabalho, é possível concluir que a substituição da partição *recovery*, como método para extração de dados de *smartphones*, pode ser realizada em aparelhos que possuem o *boot loader* destravado. Nos aparelhos em que o método pode ser aplicado, os dados foram extraídos com sucesso. Contudo, em um deles a instalação principal do Android foi prejudicada e o aparelho não iniciou novamente. Embora isto não tenha comprometido a extração e a integridade dos dados em si, esta é uma possibilidade que deve ser considerada antes de aplicar este método.

Como proposta de trabalho futuro, pretende-se analisar as situações em que a manipulação do aparelho acaba por submetê-lo a um *reset* de fábrica, o que não implica, necessariamente, na remoção de dados presentes [24]. A análise do aparelho com Android nestas condições poderia ainda permitir a recuperação de dados pessoais importantes para a perícia.

## REFERÊNCIAS


[1] INTERNATIONAL TELECOMMUNICATION UNION. Disponível em: <http://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx>. Acesso em: 06 mai. 2015.

[2] VECCHIA, E. D. Perícia Digital – Da Investiação à Análise Forense. Millenium Editora, 2014.

[3] MAHAPATRA, L. *Tech / Sci. International Business Times*. 2013. Disponível em: <http://www.ibtimes.com/android-vs-ios-whats-most-popular-mobile-operating-system-your-country-1464892>. Acesso em: 26 nov. 2013.

[4] HOOG, Andrew. *Android Forensics - Investigation, Analysis and Mobile Security for Google Android*. Waltham: Elsevier, 2011.

[5] SIMÃO, André Morum de L. *Proposta de método para Análise Pericial em Smartphone com Sistema Operacional Android*. Dissertação. (Mestrado)- Departamento de Engenharia Elétrica, Universidade de Brasília. Brasília, DF, 2011.

[6] ELEUTÉRIO, P. M. S. e MACHADO, M. P. Desvendando a Computação Forense. Novatec, 2011.

[7] CASEY, Eoghan. *Digital Evidence and Computer Crime*. Waltham: Elsevier, 2011.

[8] HASEMAN, C. *Android Essentials*. New York: Apress, 2008.

[9] GUPTA, Aditya. *Learning Pentesting for Android Devices*. Packt Publishing, 2014.

[10] ANDROID DEVELOPERS. *Android Debug Bridge*. Disponível em: <http://developer.android.com/tools/help/adb.html>. Acesso em: 13 jan. 2014a.

[11] SUN, Chen, WANG, Yang and ZHENG, Jun. Dissecting Pattern Unlock: The effect of pattern strength meter on patterin selection. *Journal of Information Security and Applications*, 19, 308-320, 2014.

[12] VIDAS, Timothy; ZHANG, Chengye; CHRISTIN, Nicolas. Toward a general collection methodology for Android devices. *Digital Investigation: The International Journal of Digital Forensics & Incident Response*, 8, p. S14-S24, ago. 2011.

[13] TYLER, J. and VERDUZCO, W. *XDA Developers' AndroidTM Hacker's Toolkit*. John Wiley and Sons, 2012.

[14] XDA DEVELOPERS. *Recovery*. Disponível em: <http://forum.xda-developers.com/wiki/Recovery>. Acesso em: 01 mar. 2014a.

[15] SON, Namheun; LEE, Yunho; KIM, Dohyun; JAMES, Joshua; LEE, Sangjin; LEE, Kyungho. A study of user data integrity during acquisition of Android devices. *Digital Investigation: The International Journal of Digital Forensics & Incident Response*, 10, p. S3-S11, ago., 2013.

[16] VIAFORENSICS. *AFLogical tool*. Disponível em: <https://viaforensics.com/resources/tools/android-forensics-tool/>. Acesso em: 03 mar. 2014.

[17] XDA DEVELOPERS. *Boot Loader*. Disponível em: < http://forum.xda-developers.com/wiki/Bootloader>. Acesso em: 03 mar. 2014b

[18] TEAM WIN. *Team Win Recovery Project*. Disponível em: <http://teamw.in/project/twrp2>. Acesso em: 03 mar. 2014.

[19] ANDROID DEVELOPERS. *What is API Level?* Disponível em: <http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels>. Acesso em: 01 abr. 2014b.

[20] SAMSUNG ELECTRONICS. *Service manual for GSM telephone GT-I9100*. 2011. 103 p.

[21] LG ELETRONICS. LG-P920h. Manual do usuário, 2011.

[22] GITHUB – OMAP4BOOT. *Tools to boot omap4xx over USB*. Disponível em: <https://github.com/swetland/omap4boot>. Acesso em: 18 abr. 2014.

[23] MOTOROLA BOOTLOADER UNLOCK. *Unlock your Bootloader*. Disponível em: <https://motorola-global-portal.custhelp.com/app/standalone/bootloader/unlock-your-device-a/action/auth >. Acesso em: 21 de abr. 2014.

[24] SCHWAMM, Riqui and ROWE, Neil. Effects of the Factory Reset on Mobile Devices. *The Journal of Digital Forensics, Security and Law*, 9(2), 205-220, 2014.