Perícia forense computacional em telefones celulares com sistema operacional Android

Sibelius Lellis Vieira[1]
Adriano Rodrigues da Cruz[2]

Resumo: Os *smartphones*, telefones celulares com maior poder de processamento e armazenamento, são cada vez mais populares em todo mundo. Além das tradicionais funções de originar e receber ligações, estes aparelhos permitem a troca de arquivos, a conexão a vários tipos de redes e o acesso à internet, tornando-os verdadeiros computadores. Da mesma forma que os computadores, os telefones celulares também armazenam dados dos usuários, tais como agendas, ligações originadas e recebidas, mensagens, *emails*, fotos e outros. Estes dados, denominados evidências digitais no âmbito da computação forense, podem ser de grande valia para investigação e solução de crimes. Contudo, o processo de aquisição de dados de telefones celulares é uma tarefa difícil, especialmente quando os aparelhos encontram-se bloqueados e com a função de depuração USB desabilitada. O objetivo deste trabalho é descrever técnicas, métodos e ferramentas que podem ser aplicadas para a aquisição de evidências digitais em *smartphones* que utilizam o sistema operacional Android, estão bloqueados e com a função de depuração USB desabilitada. Ao final, é proposto o método de extração de dados através da substituição da partição *recovery* e os resultados dos testes realizados com este método.

Palavras-chave: 1. Evidências digitais. 2. *Smartphones.* 3. Android. 4. Extração de dados.

Abstract: *Smartphones are mobile phones with greatest processing power, and storage and are becoming more and more popular around the world. These devices can originate and receive calls, exchange files, connect to networks and allow internet access, making them real computers. Just as a computer, a cell phone also store user's data, such as agendas, originated and received calls and messages, emails and photos, to name a few. These data, known as digital evidence in the computer forensic field, can be valuable asset for crime investigation and solving. However, the process of data acquisition of mobile phones is a difficult task, especially when the devices are locked and USB debugging function disabled. The objective of this paper is to describe techniques, methods and tools that can be applied to the digital data acquisition on smartphones with Android operating system, wich are locked and its USB debugging disabled. At the end, we present a method of data extraction by replacing the recovery partition, and its results.*

Keywords: *1. Digital evidence. 2. Smartphones. 3. Android. 4. Data extraction.*

1 Doutor em Engenharia Elétrica, com Pós-doutorado em Ciência da Computação, Perito Criminal da SPTC/GO e Professor da PUC Goiás junto ao Departamento de Computação.
2 Graduando em Ciência da Computação pela PUC GOIÁS, programador sênior do SENAC/GO.

1 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. Qualquer dispositivo computacional que possua memória de armazenamento permanente ou temporário de dados pode ser alvo da análise forense. Dentre estes dispositivos destacam-se os computadores, *notebooks*, *laptops*, *tablets*, telefones celulares, máquinas fotográficas entre outros.

De acordo com a previsão da International Communication Union (ITU), o número de telefones celulares, em 2014, ultrapassará o número de habitantes do planeta, chegando à marca de 7,3 bilhões de linhas ativas. (ITU, 2013) Isto torna o telefone celular um importante alvo da análise forense.

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. (MAHAPATRA, 2013)

Com o crescente número de telefones celulares e a grande diversidade de tecnologias envolvidas, faz-se necessário um estudo no campo da computação forense e da investigação digital sobre as ferramentas existentes para identificação, extração e preservação de possíveis evidências digitais que podem ser encontradas nestes aparelhos.

1.1 Definição do problema

O Android se tornou o sistema operacional móvel mais popular do mundo no começo de 2011. (HOOG, 2011) 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 desativação 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. (HOOG, 2011) 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. Alguns estudos afirmam que a extração de dados neste cenário é inviável. (SIMÃO, 2011)

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. Neste cenário, o perito fica impossibilitado de se conectar ao ADB e não é possível ter acesso ao menu do sistema para habilitar a depuração USB.

1.2 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.

2 REFERENCIAL TEÓRICO

2.1 Computação forense

A computação forense tem como objetivo desenvolver técnicas e ferramentas para a investigação e análise de potenciais evidências digitais. (SCHWEITZER, 2003) 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. (KLEIMAN, 2007)

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. (CASEY, 2011)

2.2 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. (HOOG, 2011)

2.2.1 Aplicativos

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. (HASEMAN, 2008)

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. (HASEMAN, 2008) Estes dados podem ser utilizados pelo perito forense na elaboração de um laudo pericial, por exemplo.

2.2.2 Armazenamento dos dados

Os aplicativos armazenam grande quantidade de dados do usuário, tais como mensagens de texto, contatos, agenda, fotos, vídeos, histórico de chamadas, músicas, coordenadas do GPS, compromissos agendados no calendário, dados de mídias sociais (Facebook, Twitter etc.), entre outros. Estes dados são armazenados basicamente em dois locais: interno e externo. (HOOG, 2011)

O armazenamento externo dos dados é feito por Secure Digital Card (cartão SD), geralmente formatado com o sistema de arquivos Microsoft FAT32 (HOOG, 2011), o que facilita o trabalho do perito, uma vez que o cartão SD pode ser removido e analisado em outra máquina. No armazenamento externo, os aplicativos podem gravar os dados em qualquer estrutura de pasta.

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. (HOOG, 2011) A Tabela 1 mostra a estrutura básica do diretório dos aplicativos.

Tabela 1
Estrutura básica dos diretórios dos aplicativos em /data/data/<app>

shared_prefs	Diretório para armazenar preferências compartilhadas no formato XML
Lib	Bibliotecas personalizadas
Files	Arquivos que o desenvolvedor salvou no armazenamento interno
Cache	Arquivos de cache
Databases	Banco de dados SQLite

2.2.3 O Android Software Development Kit (SDK) e o banco de dados SQLite

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. 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.

O Android 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. Estes bancos de dados são armazenados normalmente no subdiretório /data/data/<app>/databases. (HOOG, 2011) Aqui reside uma importante fonte de dados para análise pericial. 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.

2.2.4 Android Debug Bridge (ADB)

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 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 (ANDROID DEVELOPERS, 2014a):

- um utilitário de linha de comando, que é executado pelo usuário para emitir os comandos;
- 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;
- 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 2 lista alguns comandos do cliente ADB.

Tabela 2
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: Android Developers, 2014a.

Caso o aparelho apreendido esteja com a opção 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. Existem alguns aplicativos que podem ser instalados para tentar quebrar o bloqueio de tela e permitir o acesso às funções do aparelho.

Figura 1
Ativação da depuração USB

2.2.5 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).

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.

Figura 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.

2.2.6 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 3 estão presentes na maioria dos *smartphones* com Android. As partições de sistema, dados do usuário, *boot*, *cache* e *recovery* tipicamente estão presentes nos aparelhos. (VIDAS; ZHANG; CHRISTIN, 2011) Os aplicativos que armazenam dados do usuário na memória interna do telefone gravam estes dados na partição /data, no caminho /data/data/<app>. (HOOG, 2011)

Tabela 3
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/ mtd1	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: Vidas, Zhang e Christin, 2011.

2.2.7 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 4 exibe as combinações de teclas para alternar o modo de inicialização de alguns modelos. (VIDAS; ZHANG; CHRISTIN , 2011)

Tabela 4
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*	câmera + *power*	Modo que permite *flashing* via RSD Lite
Motorola Droid	*Recovery*	*power* + x	*Boot* na partição *recovery* (após, câmera + vol. *up* 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*	vol. *up* + vol. *down* (então insira USB)	*Boot* no modo Samsung *force download*
Samsung Captivate	*Recovery*	*power* + vol. *up* + vol. *down*	*Boot* na partição *recovery*
Samsung Galaxy Tab	*Flash*	*power* +vol. *down*	*Boot* no modo Samsung *force download*
Samsung Galaxy Tab	*Recovery*	*power* + vol. *up*	*Boot* na partição *recovery*

Fonte: Vidas, Zhang e Christin, 2011.

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. Esta partição contém seu próprio *kernel* do Linux, independente do *kernel* da instalação principal.

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. (HOOG, 2011)

2.2.8 A partição recovery

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 (XDA DEVELOPERS, 2014a) 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. (HOOG, 2011)

Esta partição geralmente possui um tamanho pequeno e seu dispositivo associado pode ser diferente, dependendo do modelo e do fabricante. Detalhes sobre esta partição podem ser vistos examinando o arquivo /proc/mtd (HOOG, 2011), conforme mostra a Figura 3.

Figura 3
Detalhes da partição *recovery*

```
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: Hoog, 2011.

2.3 A extração de dados

Existem vários métodos para extração de dados. Estes métodos são basicamente divididos entre extração física e extração lógica. (SON et al., 2013)

2.3.1 Extração física

De acordo com Hoog (2011), 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.

Porém, não podem ser adequadamente utilizados. Já a extração por *software* utiliza a técnica de executar *software* no aparelho, fornecendo uma imagem física completa das partições. (HOOG, 2011) É a técnica normalmente empregada em diversas ferramentas de extração de dados.

A técnica de *chip-off* efetivamente separa a memória *flash* do aparelho, para que esta possa ser lida por *hardware* especializado. Esta é uma técnica destrutiva, pois além de inutilizar o aparelho, também pode provocar estragos na memória durante o processo de separação, fazendo com que os dados se percam. (SON et al., 2013)

O JTAG é o nome popular do padrão IEEE 1449.1. Este padrão especifica regras para escrita e interconexão de placas de circuito impresso. A aquisição de dados por JTAG somente é possível quando se identificam as portas de *debug* (também conhecidas como portas de teste) no processador do aparelho. Então, cabos devem ser soldados nestas portas e ligados a um aparelho conhecido como *flasher box.* A partir daí, *softwares* são usados para realizar a extração dos dados. O problema é que, além de necessitar de engenheiros especializados para solda, nem sempre é possível soldar corretamente os cabos nas portas do processador, mesmo quando o fabricante divulga o esquema de portas. (HOOG, 2011)

2.3.2 Extração lógica

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 Hoog (2011), as técnicas lógicas de extração de dados apenas necessitam que a opção 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. 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. (VIAFORENSICS, 2014)

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 Depuração USB estiver desabilitada, também não é possível instalar aplicativos ou executar comandos. Neste cenário, ainda existe outra possibilidade: a substituição da partição *recovery*.

2.4 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 a técnica de *ADB pull* para extração dos dados.

2.4.1 O boot loader

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.* (XDA DEVELOPERS, 2014b) Alguns fabricantes desenvolvem seus próprios *boot loaders* e *softwares* para interagir com eles. Uma das funções destes *softwares* é permitir a substituições de partições do aparelho. (HOOG, 2011) 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 ser utilizada em modelos diferentes.

O *boot loader* pode estar bloqueado ou não. Um *boot loader* bloqueado somente carrega sistema operacional com assinatura válida. Isso significa que não é possível instalar versões modificadas do Android como a CyanogemMod, por exemplo. Da mesma forma, não é possível instalar uma imagem personalizada na partição *recovery*. Nestes casos, é necessário realizar a substituição do próprio *boot loader* primeiro. Esse procedimento varia de acordo com o aparelho e pode violar a garantia do fabricante. (XDA DEVELOPES, 2014b)

2.4.2 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 *flash* de imagens para partições do aparelho. Para utilizar o *fastboot*, é necessário que o *boot loader* do aparelho seja compatível com o modo *fastboot* e esteja desbloqueado. 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.* (HOOG, 211) 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 *flash* de novas partições.

Figura 4
Comando *fastboot devices*

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

C:\Users\Adriano>
```

2.4.3 Partições recovery personalizadas

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 (XDA DEVELOPERS, 2014a);

b. TWRP: Team Win Recovery Project possui, além das opções padrão, funções como *backup*, restauração e acesso via ADB habilitado. Sua interface é sensível ao toque e é personalizável. (TEAM WIN, 2014)

3 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. De maneira geral, os trabalhos pesquisados na fase inicial tratavam-se, em sua essência, de aspectos gerais da perícia forense em si, desconsiderando aspectos específicos de cada plataforma.

A partir da fase inicial de pesquisa, foi possível encontrar estudos que tratavam especificamente do tema proposto por este trabalho. Além disto, livros, artigos, tutoriais, e fóruns relacionados ao tema foram pesquisados. Desta forma, foi possível analisar diversas características do sistema operacional Android e elaborar um método. O método foi elaborado baseando-se nos princípios gerais da forense computacional. Todavia, durante a pesquisa bibliográfica foi verificado que o processo de análise pericial em *smartphones* é sempre mais invasivo do que análises feitas de computadores pessoais. Desta forma, o método proposto foi criado no intuído de preservar ao máximo a integridade dos dados, apesar de ser minimamente 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.

Este seção é inteiramente 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.

3.1 O método proposto

A substituição da partição *recovery* padrão como método para extração dos dados do usuário é utilizada neste trabalho, visto que este método pode ser utilizado quando não é possível habilitar a opção de depuração USB através da inicialização normal do telefone celular. 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 a imagem ClockworkMod. Ela foi escolhida porque existem diversas versões compatíveis com vários modelos de aparelhos, além de possuir uma vasta documentação em fóruns *online* de como instalá-la.

3.2 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 etc. 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. (ANDROID DEVELOPERS, 2014b) A versão 2.3 do Android, por exemplo, foi criada utilizando o nível de API número 9. Aplicativos criados utilizando o nível de API número 10 não podem ser instalados nesta versão, pois os desenvolvedores podem ter utilizado alguma função que não existia na versão anterior da API. A Tabela 5 relaciona a versão do Android ao nível de API suportado.

Tabela 5
Versões do Android e seus níveis de API

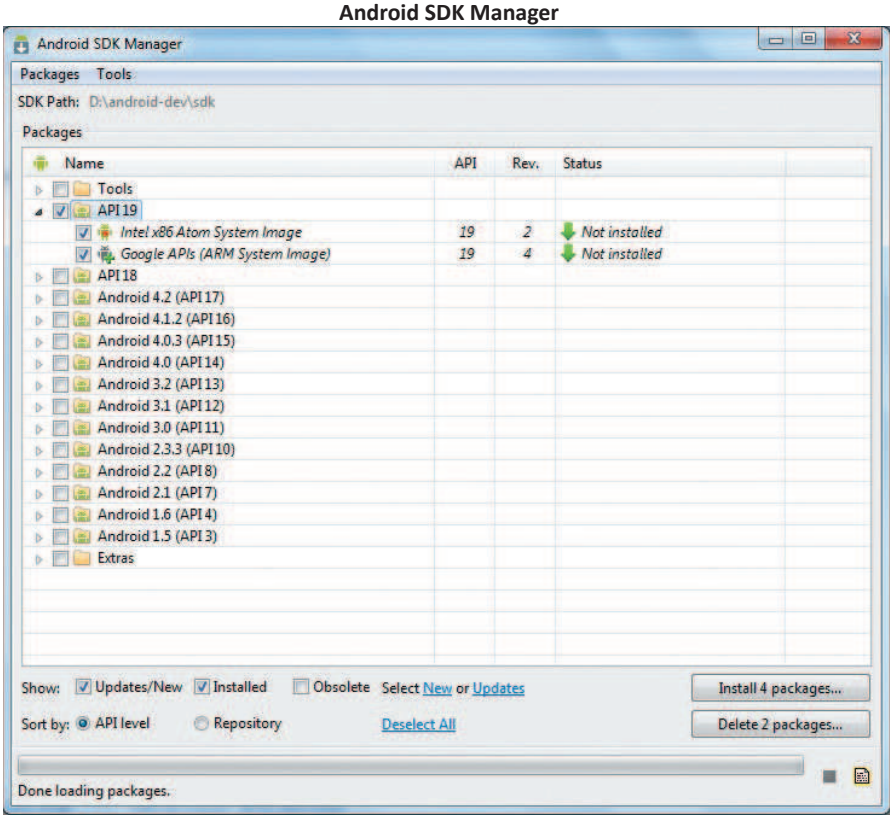
Versão	Nível da API	Codenome
Android 4.4	19	KITKAT
Android 4.3	18	JELLY_BEAN_MR2
Android 4.2.2 Android 4.2	17	JELLY_BEAN_MR1
Android 4.1.1 Android 4.1	16	JELLY_BEAN
Android 4.0.4 Android 4.0.3	15	ICE_CREAM_SANDWICH_ MR1
Android 4.0.2 Android 4.0.1 Android 4.0	14	ICE_CREAM_SANDWICH

Versão	Nível da API	Codenome
Android 3.2	13	HONEYCOMB_MR2
Android 3.1.x	12	HONEYCOMB_MR1
Android 3.0.x	11	HONEYCOMB
Android 2.3.4 Android 2.3.3 Android 2.3.2	10	GINGERBREAD_MR1
Android 2.3.1 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: Android Developers 2014b.

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.

Figura 5
Android SDK Manager



Embora não seja necessário, também é possível acrescentar o caminho de instalação do SDK na variável de ambiente PATH. Isso evita que o examinador, ao tentar executar o comando adb, por exemplo, tenha que digitar o caminho completo onde o utilitário está instalado, economizando tempo e evitando erros.

3.3 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. (LG ELETRONICS, 2011) A Figura 6 exibe a aparência do aparelho e mostra suas teclas principais.

Figura 6
LG Optimus 3D

O processo de extração de dados de um P920h começa por uma análise do estado inicial do aparelho. Se o aparelho estiver desligado, deve ser verificada a presença de cartões SIM e SD. O cartão SD, caso presente, pode ser removido para análise na máquina do examinador. O cartão SIM deve ser removido para evitar que o aparelho tente se conectar à Estação Rádio Base (ERB) quando ligado. Ambos os cartões ficam na parte traseira do aparelho, como mostrado na Figura 7.

Figura 7
P920h - SIM *Card* e SD *Card*

Após a constatação de que a opção 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*.

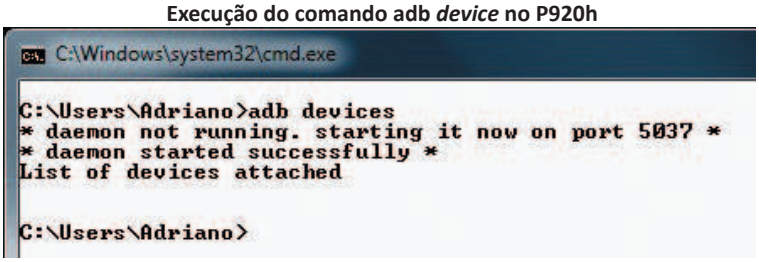
3.3.1 *Inicialização no modo recovery*

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 experimento realizado com o P920h, uma ilustração da tela é mostrada na Figura 8, indicando que o Android foi iniciado pela partição *recovery*.

Figura 8:
P920h - Tela do modo *recovery* original

No modo *recovery*, o perito deve ligar o cabo USB e tentar verificar se é possível conectar-se ao aparelho usando o ADB. Isso pode ser feito com o comando adb *devices*, que lista todos os aparelhos conectados à porta USB e que estão com o serviço do adb em execução. A execução deste comando em um P920h com a partição *recovery* padrão não listou dispositivos, como mostra a Figura 9.

Figura 9
Execução do comando adb *device* no P920h



3.3.2 *Substituição da partição recovery usando o fastboot*

A substituição da partição *recovery* original do P920h é 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. (GITHUB - OMAP4BOOT, 2014) 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*.

Esse processo de substituição pode ser resumido basicamente em quatro etapas:

a. obter uma imagem da partição *recovery* compatível;

b. ligar o celular no modo *fastboot*;

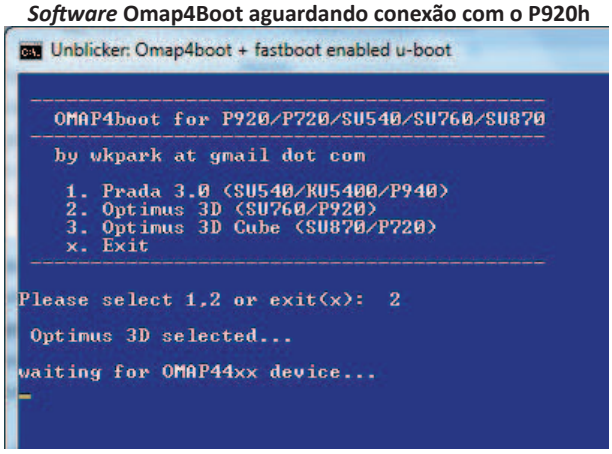
c. executar o comando para realizar a substituição da partição;

d. ligar o celular no modo *recovery*.

O primeiro passo para substituição da partição *recovery* é encontrar outra partição *recovery* compatível. Neste trabalho, a ClockworkMod versão 6.0.1.9 compatível com o P920h foi utilizada. Essa imagem pode ser encontrada no *site* https://www.clockworkmod.com/rommanager.

Em seguida, o celular deve ser ligado no modo *fastboot*. Conforme descrito anteriormente, o *software* Omap4Boot-for-optimus é utilizado nesse momento. Após a descompactação, o arquivo start_fastboot.bat deve ser executado. Então é apresentada uma tela solicitando 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 à USB, conforme ilustra a Figura 10.

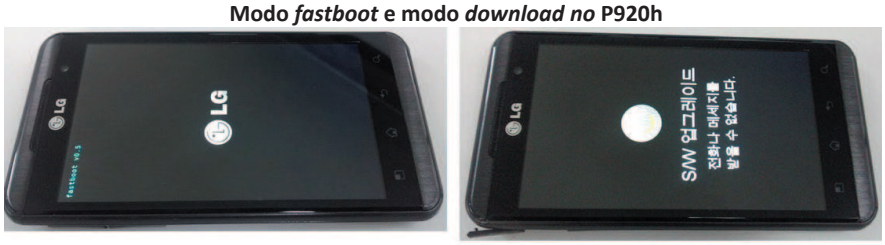
Figura 10
***Software* Omap4Boot aguardando conexão com o P920h**



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, como mostrado na Figura 11(a), indicando que o modo *fastboot* está ativo. Esse mesmo procedimento pode ser feito para ligar o telefone no modo *download*, bastando segurar a tecla Volume(+). Neste caso, a tela do celular fica como mostrado na Figura 11(b), indicando que o modo *download* está ativo.

Figura 11
Modo *fastboot* e modo *download no* P920h



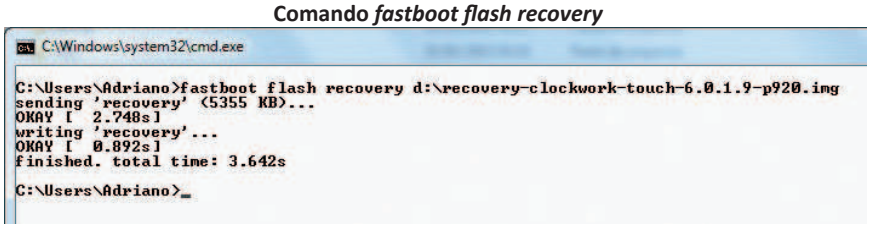
(a) (b)

Para testar se o modo *fastboot* está funcionando corretamente, o comando *fastboot devices* pode ser executado. De forma similar ao comando adb *devices*, este comando lista todos os dispositivos conectados ao computador que são compatíveis com o modo *fastboot*. O resultado da execução deste comando pode ser visualizado na Figura 12.

Figura 12
Comando *fastboot devices*

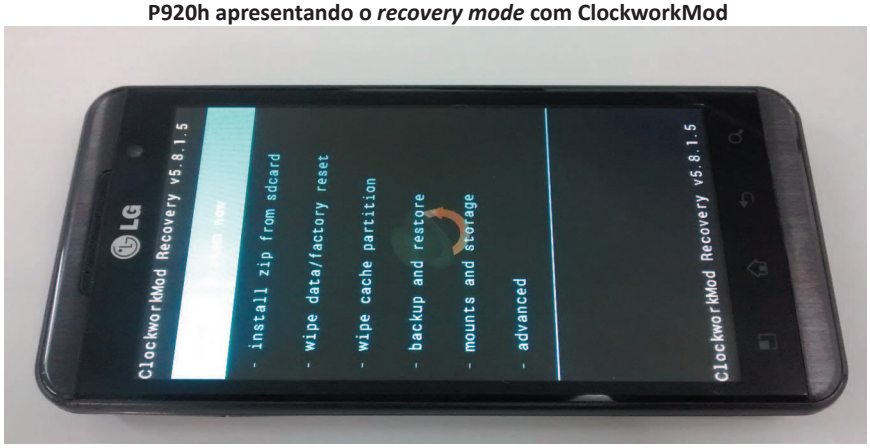
A próxima etapa é a substituição da partição em si. Isso pode ser feito com o comando *fastboot flash recovery* recovery.img. Este comando substitui a partição *recovery* pela imagem recovery.img fornecida. O resultado deste comando pode ser visualizado na Figura 13.

Figura 13
Comando *fastboot flash recovery*



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 14. Agora, é possível conectar o cabo USB e comunicar-se ao aparelho utilizando a partição *recovery* instalada.

Figura 14
P920h apresentando o *recovery mode* com ClockworkMod

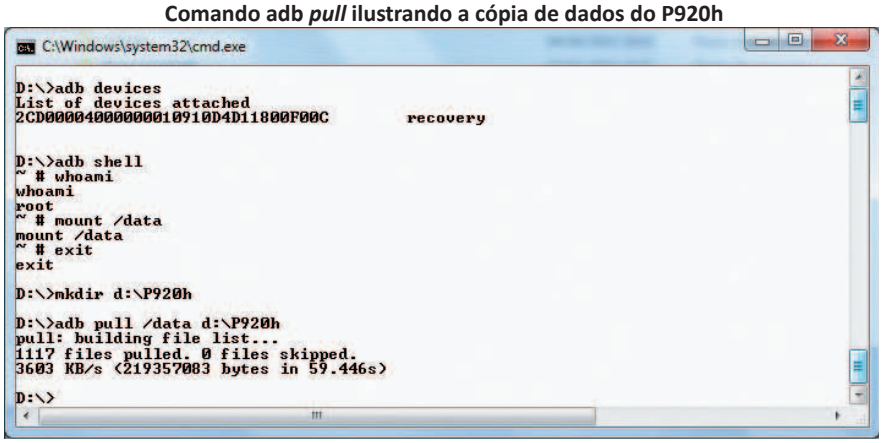


3.3.3 Cópia dos dados

O método de extração de dados utilizado neste experimento foi o adb *pull*. Este é um método rápido e simples de ser executado, pois consiste simplesmente na cópia da partição /data para a máquina do examinador utilizando o ADB.

Inicialmente, é necessário que o aparelho seja ligado no modo *recovery*. Em seguida, é necessário que a partição /data seja montada. Para fazer isso, é preciso conectar-se com o aparelho através do comando adb *shell* e executar o comando *mount* /data. Um diretório na máquina do examinador também deve ser criado para armazenar os arquivos copiados do telefone. Por fim, o comando adb *pull* deve ser emitido para realizar a cópia dos arquivos. A execução destes comandos no P920h é mostrada na Figura 15.

Figura 15
Comando adb *pull* ilustrando a cópia de dados do P920h



3.4 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*).

Da mesma forma que o P920h, 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.

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

Neste experimento, a substituição da partição *recovery* foi feita utilizando o *software* Odin3. 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. (SAMSUNG ELECTRONICS, 2011) 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* (Odin *mode*).

Neste momento, 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. Caso o aparelho não seja reconhecido os *drivers* devem 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*".

Após a conclusão deste procedimento, o aparelho deve ser imediatamente desconectado da USB e a bateria removida. Isto deve ser 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. Então, o cabo USB deve ser conectado, ligando o celular ao computador.

A primeira 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 os dados foram copiados corretamente.

O GT-I9100 possui memória interna de 16 GB ou 32 GB. Parte desta memória é formatada com o sistema de arquivos FAT e disponibilizada na forma de cartão SD interno. A cópia do cartão SD interno é necessária, pois nele também são armazenados dados do usuário, como fotos e dados de aplicativos.

Os passos para extração dos dados do cartão SD interno são os mesmos para extração da partição /data, trocando-se apenas o nome da partição (de /data para /sdcard). Contudo, no experimento realizado com o GT-I9100, o comando mount /sdcard não obteve êxito. Neste caso, é necessário descobrir qual dispositivo está associado à partição do cartão SD interno e executar o comando para montá-lo.

Uma listagem dos dispositivos relacionados por nome pode ser obtida no GT-I9100 pelo comando ls -l /dev/block/platform/dw_mmc/by-name. O dispositivo relacionado à abreviação UMS (USB Mass Storage) deve ser montado no diretório /sdcard. O comando digitado para montar o cartão SD interno foi mount /dev/block/mmcblk0p11 /sdcard. Após isto, o processo de cópia segue de forma adequada.

3.5 Procedimento de extração de dados do Samsung Galaxy 5 (GT-I5500)

O GT-I5500 também foi desenvolvido pela Samsung. Ele foi lançado no mercado em agosto de 2010 com sistema Android 2.1 (*Eclair)* atualizável até a versão Android 2.2 (Froyo). Possui processador de 600 MHz, tela de 2.8 polegadas e aceita cartão de memória externo de até 16 GB.

Da mesma forma que os outros modelos, é necessário fazer uma avaliação inicial, remover cartão SD e SIM se presentes, verificar bateria e carregador. Também é necessário fazer a instalação dos *drivers* compatíveis com este modelo para que seja possível acessá-lo via ADB. Diferentemente dos outros modelos, não existe uma combinação de teclas documentada para inicialização do GT-I5500 no modo *recovery*. A inicialização neste modo só é possível depois de fazer a substituição da partição *recovery*. Imediatamente após o término do procedimento de substituição, o próprio Odin3 reinicializa o telefone neste modo.

A partição *recovery* do GT-I5500 também pode ser substituída usando o Odin. Como este modelo é mais antigo, a versão 4.28 do Odin foi usada para substituir a partição *recovery* padrão pelo ClockworkMod 4.0.15. Para utilizar o Odin é necessário iniciar o telefone no modo *download*. Isso foi feito pressionando simultaneamente as teclas Volume(-), *Home* e *Power*, com o telefone desligado. Ao iniciar, a tela do telefone apresentou um ícone triangular amarelo e o texto "*DOWNLOADING*...", indicando que o modo *download* está ativo.

Em seguida, o celular foi conectado à USB e o Odin aberto. O texto "*ADDED*" apareceu no campo *Message* do Odin, indicando que o aparelho foi reconhecido. Então, o arquivo com a imagem do ClockworkMod e o arquivo com o mapa de partições foram selecionados e o botão *Start* clicado. A opção Auto *Reboot* foi deixada marcada. Assim que o Odin finalizou a substituição, a mensagem "*PASS*" foi apresentada e o aparelho foi reiniciado no modo *recovery*.

3.6 Procedimento de extração de dados do 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 1 GB e armazenamento interno de 8 GB ou 16 GB. Sua versão original vem com Android 4.3 (*Jelly Bean*) atualizável para Android 4.4.2 (*KitKat*).

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 cartões SIM 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 desbloqueado. Caso o *boot loader* esteja bloqueado, é necessário desbloqueá-lo primeiro. Esse desbloqueio pode ser feito através de um código obtido no *site* da própria Motorola. (MOTOROLA BOOTLOADER UNLOCK, 2014) Para obter este código, é necessário digitar os dados de desbloqueio do telefone no *site* da Motorola. Estes dados de desbloqueio são conseguidos através do próprio *fastboot*.

4 RESULTADOS E DISCUSSÃO

4.1 Resultados para o LG Optimus 3D (P920h)

Conforme apresentado na seção 3.3, 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. Uma vez extraídos os dados, o examinador deve analisar fotos, contatos, imagens, bancos de dados etc. em busca de possíveis evidências digitais.

4.2 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 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 inoperável se 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.

4.3 Resultados para o Samsung Galaxy 5 (GT-I5500)

Assim como nos casos anteriores, a extração de dados das partições /data e /sdcard foram extraídos do GT-I5500 com o comando adb *pull*. Primeiramente, estas duas partições foram montadas através do próprio ClockworkMod, na opção *Mounts and storage*. Em seguida, duas pastas foram criadas na máquina e o comando para copiar os dados foi executado. Após a conclusão da cópia, os dados da partição /data e da partição /sdcard foram armazenados em diretórios da máquina do examinador, podendo ser objeto de análise a partir de então.

4.4 Resultados para o Motorola Moto G Dual SIM (XT1033)

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

Neste momento, o telefone estava no modo *fastboot* mostrando a informação de que o *boot loader* estava bloqueado. Então, o cabo USB foi conectado ao telefone e ligado ao computador e o comando *fastboot* em get_unlock_data executado. Os dados que foram mostrados na tela foram os dados de desbloqueio do telefone.

Os dados de desbloqueio foram tratados de forma a não conter espaços ou informações adicionais. O *site* de desbloqueio da Motorola exige que apenas a sequência de caracteres seja informada. Ao entrar com os dados de desbloqueio no *site* da Motorola, a mensagem "*Your device does not qualify for bootloader unlocking.*" foi mostrada. (MOTOROLA BOOTLOADER UNLOCK, 2014) Desta forma, conclui-se que este aparelho não pode ter seu *boot loader* desbloqueado.

Sem o código de desbloqueio, não foi possível desbloquear 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 desbloqueado.

5 CONCLUSÕES

Este trabalho teve como principal objetivo analisar, propor e demostrar 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.

Após a revisão bibliográfica, uma hipótese para solução do problema foi elaborada. Tal hipótese propõe a substituição da partição *recovery* padrão por outra que possibilite o acesso como usuário *root* via ADB, possibilitando, desta forma, a extração dos dados do usuário, sem comprometer a integridade das informações extraídas. Essa foi a hipótese principal desenvolvida neste trabalho.

Para verificação da hipótese proposta, quatro experimentos foram realizados. Estes experimentos foram feitos com diferentes modelos de aparelhos, todos no mesmo cenário do problema, conforme descrito anteriormente. Todos os passos para realização dos experimentos foram descritos neste trabalho em um item específico para isto.

O primeiro experimento foi feito com um aparelho LG Optimus 3D. A partição *recovery* padrão foi substituída e os dados extraídos com sucesso. O segundo experimento foi feito com um Samsung Galaxy S2. Neste modelo, também foi possível realizar a extração dos dados. Porém, a substituição da partição *recovery* afetou a instalação do sistema principal, levando a aparelho a não inicializar novamente. O terceiro experimento também foi feito com um aparelho da marca Samsung modelo Galaxy 5. Neste modelo, a extração dos dados foi feita com sucesso e a instalação principal do sistema não foi afetada. O quarto experimento 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 bloqueado e não foi possível desbloqueá-lo.

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* desbloqueado. 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 continuar realizando os experimentos em aparelhos que possuam o *boot loader* bloqueado, através da substituição do próprio *boot loader*. Este procedimento é específico para cada aparelho e não foi abordado neste trabalho. Outro ponto é a criação

de um aplicativo que seja instalado do sistema, colete os dados de todos os outros aplicativos instalados e envie através da internet para o perito forense. Isso evitaria o uso de *exploits* na intenção de obter acesso como *root* para utilizar o ADB, bastando apenas instalar um aplicativo direto do Google Play e executá-lo. Porém, a criação deste aplicativo esbarra nos princípios do modelo de segurança dos aplicativos, necessitando assim de um estudo mais aprofundado sobre o assunto.

REFERÊNCIAS BIBLIOGRÁFICAS

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

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.

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

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

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

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

ITU. News Technology. BBC, 2013. Disponível em: <http://www.bbc.co.uk/news/technology-22464368>. Acesso em: 23 nov. 2013.

KLEIMAN, Dave. *The Official CHFI Study Guide (Exam 312-49) for Computer Hacking Forensic Investigators*. Burlington: Syngress, 2007.

LG ELETRONICS. LG-P920h. Manual do usuário, 2011.

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.

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.

SAMSUNG ELECTRONICS. *Service manual for GSM telephone GT-I9100*, 2011. 103 p.

SCHWEITZER, Douglas. *Incident Response: Computer Forensics Toolkit*. Indianapolis: Wiley: 2003.

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

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.

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

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

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.

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

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