



第十四届CCFC计算机取证技术峰会

The 14th China Computer Forensic Conference



报告主题： 安卓6.0以上提取技术和微信恢复研究

演讲： 董志国 指掌易大连研发中心

ABOUT ME

- 董志国

毕业于大连理工大学 电子工程硕士

2002年开始钻研手机开发

2008年获周鸿祎先生天使投资

2009年~2013年担任360手机安全技术经理

2016年加入指掌易科技，任大连研发中心总经理

- 擅长领域

手机应用安全、数据安全、系统安全、软件架构设计、嵌入式开发、硬件电路设计

前言



随着安卓系统加密级别的提高，提取涉案手机里的数据变得越来越难，经过大量技术研究工作，我们整理出一些方案，能有效解决安卓6以上加密机型取证难的问题。同时，我们也分享一些微信数据恢复的经验。

主要内容



- 1 主流手机数据提取技术介绍
- 2 安卓全盘加密机型提取技术的研究
- 3 安卓文件级加密机型提取技术的研究
- 4 微信恢复的研究

数据取证

主流手机数据提取技术介绍

1

1.1 Android提取技术金字塔

提取技术	开机	解锁	root
芯片提取	不需要	不需要	不需要
物理镜像	需要	不需要	不需要
定向提取	需要	不需要	不需要
超级提取	需要	需要	不需要
高级ADB	需要	需要	需要
互传备份	需要	需要	不需要
降级备份	需要	需要	不需要
自备份	需要	需要	不需要
逻辑提取	需要	需要	不需要



- **芯片提取:**

手机无法开机的请款下，拆芯片，使用专用设备提取全镜像。不适合6.0以后加密机型

- **物理镜像:**

绕过锁屏密码，最有效，但适配机型有限，技术难度较高

- **定向提取:**

在物理提取的基础上，指定提取内容，缩短提取时间，加快提取效率

- **超级提取:**

利用系统漏洞，取出微信QQ等APP数据，支持分身，是否支持要视系统版本而定

- **高级ADB:**

利用系统漏洞root手机，适配性较差

- **互传备份:**

利用手机厂商自带备份APP传输数据，只有少数手机厂商有此功能

- **降级备份:**

APP分身数据会丢失，有机型适配问题

- **自备份:**

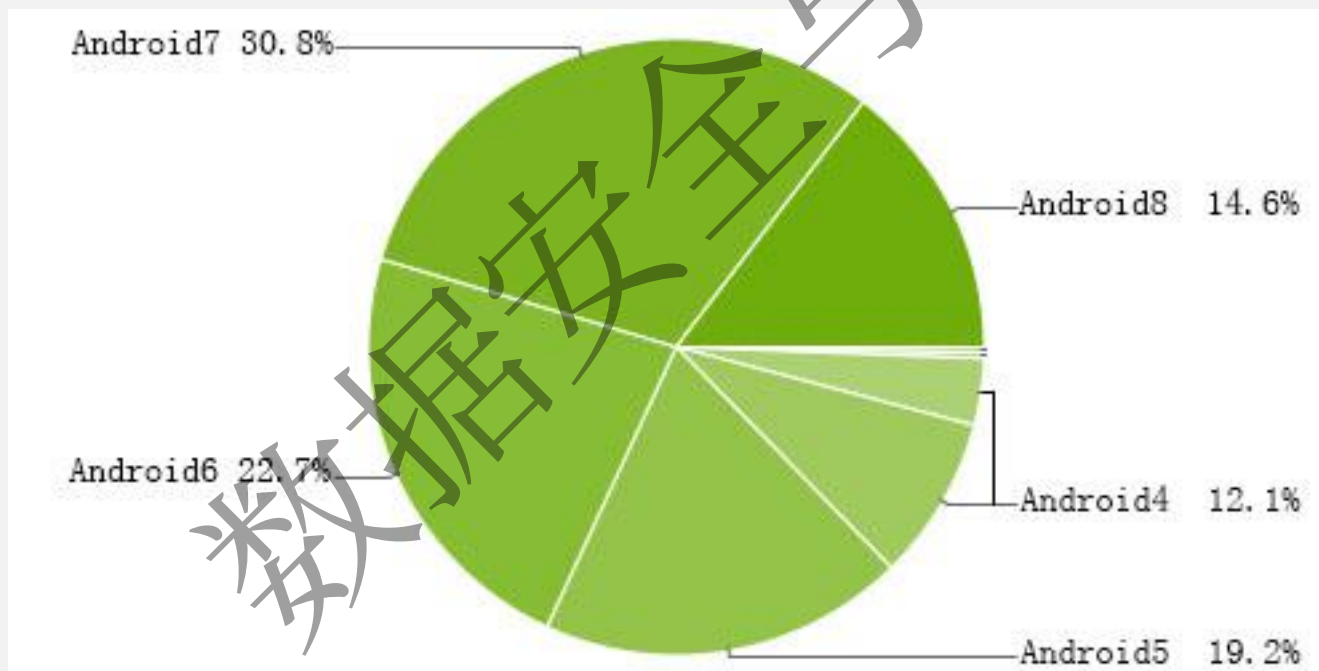
操作较繁琐，有的厂商不提供此功能

- **逻辑提取:**

类似手机助手的做法，取基本数据，无APP数据，机型通杀100%

1.2 Android提取的难点

- 系统加密。从Android 6.0之后系统全盘加密，7.0之后文件系统加密
- 碎片化严重。相同机型但系统小版本不同，方案可能不同



截至2018年9月12日

1.3 IOS提取技术

- 解锁密码提取

2016年以色列Cellebrite帮助FBI

2018年美国GrayKey



- 越狱提取

需要解锁密码，利用系统漏洞，越狱对手机有改变

- 逻辑提取

需要解锁密码，无需越狱

安卓全盘加密FDE机型 提取技术的研究

2

2.1 没加密的安卓4.0 5.0

1) 高通 9008 9006模式

- 三星、小米、VIVO、OPPO、一加、360、少量华为等等
- 骁龙系列提取镜像主要是通过9008模式提取镜像（9006模式一般是在手机异常状态下才会使用，极少使用）



2) MTK串口模式

- oppo、vivo、小米、魅族、金立、中兴、美图、360
- 主流MTK芯片Helio系列 MT6752，MT6753，MT6750，MT6739，MT6735，MT6595，MT6592，MT6582等。
- MTK通用的提取镜像方法，需要各厂商提供的MTK_AllInOne_DA、Preloader文件、Auth文件，有的厂商还需要联网



2.1.1 高通9008模式

- 全称Qualcomm HS-USB QDloader 9008（也称 EDL 模式），主要用于刷机救砖和镜像提取，相对于recovery、fastboot和Android系统是独立的。

- 操作步骤：

1. 三种进入9008模式

- 1) 手机关机后按住特定组合键，然后连接数据线至PC端
- 2) 手机关机状态下通过深刷线 / 深刷板，连接数据线至PC端
- 3) 拆机后短接特定的触点也可进入

2. 联机后可以在设备管理器下的“端口（COM和LPT）”中找到如图所示。

3. 通过通讯类Firehose刷入开机引导文件

4. 通过高通协议，发送指令集，提取镜像



Firehose类用于串口通信

```
typedef struct {
    BYTE Version;
    char MemoryName[8];
    bool SkipWrite;
    bool SkipStorageInit;
    bool ZLPAwareHost;
    int ActivePartition;
    int MaxPayloadSizeToTargetInBytes;
} fh_configure_t;

class Firehose : public Protocol {
public:
    Firehose(SerialPort *port, HANDLE hLogFile = NULL);
    ~Firehose();
    int WriteData(BYTE *writeBuffer, __int64 writeOffset, DWORD writeBytes, DWORD *bytesWritten, UINT8 partNum);
    int ReadData(BYTE *readBuffer, __int64 readOffset, DWORD readBytes, DWORD *bytesRead, UINT8 partNum);
    int DeviceReset(void);
    int FastCopy(HANDLE hRead, __int64 sectorRead, HANDLE hWrite, __int64 sectorWrite, UINT64 sectors, UINT8 partNum);
    int ProgramPatchEntry(PartitionEntry pe, TCHAR *key);
    int ProgramRawCommand(TCHAR *key);
    // Firehose specific operations
    int CreateGPP(DWORD dwGPP1, DWORD dwGPP2, DWORD dwGPP3, DWORD dwGPP4);
    int SetActivePartition(int prtn_num);
    int ConnectToFlashProg(fh_configure_t *cfg);
protected:
private:
    int ReadData(BYTE *pOutBuf, DWORD uiBufSize, bool bXML);
    int ReadStatus(void);
    SerialPort *sport;
    UINT64 diskSectors;
    bool bSectorAddress;
    BYTE *m_payload;
    BYTE *m_buffer;
    BYTE *m_buffer_ptr;
    DWORD m_buffer_len;
    UINT32 dwMaxPacketSize;
    HANDLE hLog;
    char *program_pkt;
};
```

2.1.2 MTK串口模式

原理：通过USB串口通信，向手机发送MTK协议指令集

MTK三个端口：USB\VID_0E8D&PID_0003 普通端口

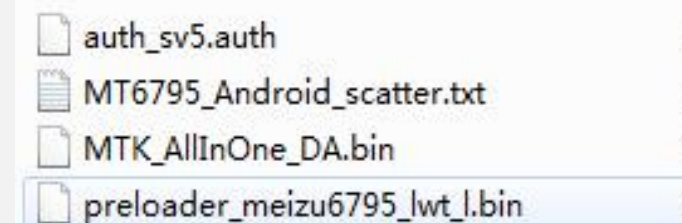
USB\VID_0E8D&PID_2000 全速端口

USB\VID_0E8D&PID_2001 高速端口

- 1.根据手机型号，确定需要准备刷入的资源文件
- 2.按住音量上键（有的手机按住音量下键）将手机连接至PC端
- 3.刷入auth文件（乐视、魅族及小众YunOS厂商）
- 4.向官方发送验证码，进行验证（仅魅族手机）
- 5.刷入MTK_AllInOne_DA文件
- 6.刷入Preloader文件
- 7.自动判断是否支持高速端口，如果支持高速端口，则切换为高速端口
- 8.获取分区信息列表，查找userdata分区的地址及大小
- 9.提取userdata分区镜像文件



```
typedef struct FilePath{  
    char *AuthPath;  
    char *DAPath;  
    char *Scatter;  
    char *PreloaderPath;  
}filePath;
```



获取端口

```
typedef struct {  
    const char *usb_pid;  
    const char *usb_vid;  
    const char *guid_type;  
} USBID;  
  
typedef struct {  
    const USBID *p_usb_id_set;  
    size_t ports;  
    int time_out;  
} USBScanArg;  
  
typedef struct {  
    char p_friendly_name[256];  
    char p_symbolic_name[256];  
    int *p_stop_flag;  
} USBScanResult;
```

```
typedef enum  
{  
    NORMAL_SPEED,  
    DA_HIGH_SPEED  
}usb_speed;  
  
typedef struct  
{  
    std::string device_pid;  
    std::string device_vid;  
    std::string guid_type;  
}USBInfoPair;
```

```
USBScanResult UsbSearch(int *stop_flag, int timeout)  
{  
  
    USBInfoPair usb_port_info[3] = { { "2001", "0E8D", "port-guid-interface" }, { "2000", "0E8D", "port-guid-interface" }, { "0003", "0E8D", "port-guid-interface" } };  
    std::list<USBInfoPair> usb_port_pool_;  
    bool is_found = false;  
  
    for (int i = 0; i < 3; i++){  
        usb_port_pool_.push_back(usb_port_info[i]);  
    }  
  
    std::vector<USBID> id_array(usb_port_pool_.size());  
    USBID *p_usb_id_array = static_cast<USBID*>(&id_array[0]);  
  
    std::list<USBInfoPair>::iterator it = usb_port_pool_.begin();  
    for (size_t i = 0; it != usb_port_pool_.end(); it++, i++){  
        {  
            p_usb_id_array[i].usb_pid = it->device_pid.c_str();  
            p_usb_id_array[i].usb_vid = it->device_vid.c_str();  
            p_usb_id_array[i].guid_type = it->guid_type.c_str();  
        }  
    }  
  
    USBScanArg usb_scan_arg = { const_cast<const USBID*>(p_usb_id_array),  
                                id_array.size(), timeout };  
    USBScanResult usb_scan_result = { "", "", stop_flag };  
  
    if (SearchUSBPortPool(&usb_scan_arg, &usb_scan_result))  
    {  
        is_found = true;  
    }  
  
    LOGD("Scanning USB port %s\n", is_found ? "succeeded!" : "failed!");  
    return usb_scan_result;  
}
```


指令集

```

/* RETURN VALUE */
#define SOC_OK 0xC1
#define SOC_FAIL 0xCF
#define SYNC_CHAR 0xC0
#define CONT_CHAR 0x69
#define STOP_CHAR 0x96
#define ACK 0x5A
#define NACK 0xA5
#define UNKNOWN_CMD 0xBB

/* COMMANDS */
#define DA_DOWNLOAD_BLOADER_CMD 0x51
#define DA_NAND_BMT_REMARK_CMD 0x52

#define DA_SDMMC_SWITCH_PART_CMD 0x60
#define DA_SDMMC_WRITE_IMAGE_CMD 0x61
#define DA_SDMMC_WRITE_DATA_CMD 0x62
#define DA_SDMMC_GET_CARD_TYPE 0x63
#define DA_SDMMC_RESET_DIS_CMD 0x64

#define DA_UFS_SWITCH_PART_CMD 0x80
#define DA_UFS_WRITE_IMAGE_CMD 0x81
#define DA_UFS_WRITE_DATA_CMD 0x82
#define DA_UFS_READ_GPT_CMD 0x85
#define DA_UFS_WRITE_GPT_CMD 0x89

#define DA_UFS_OTP_CHECKDEVICE_CMD 0x8a
#define DA_UFS_OTP_GETSIZE_CMD 0x8b
#define DA_UFS_OTP_READ_CMD 0x8c
#define DA_UFS_OTP_PROGRAM_CMD 0x8d
#define DA_UFS_OTP_LOCK_CMD 0x8e
#define DA_UFS_OTP_LOCK_CHECKSTATUS_CMD 0x8f

#define DA_USB_SETUP_PORT 0x70
#define DA_USB_LOOPBACK 0x71
#define DA_USB_CHECK_STATUS 0x72
#define DA_USB_SETUP_PORT_EX 0x73

/*For EFFUSE CMD*/
#define DA_READ_REG32_CMD 0x7A
#define DA_WRITE_REG32_CMD 0x7B
#define DA_PWR_READ16_CMD 0x7C
#define DA_PWR_WRITE16_CMD 0x7D
#define DA_PWR_READ8_CMD 0x7E
#define DA_PWR_WRITE8_CMD 0x7F

```

```

#define DA_EMMC_OTP_GETSIZE_CMD 0x9A
#define DA_EMMC_OTP_READ_CMD 0x9B
#define DA_EMMC_OTP_PROGRAM_CMD 0x9C
#define DA_EMMC_OTP_LOCK_CMD 0x9D
#define DA_EMMC_OTP_LOCK_CHECKSTATUS_CMD 0x9E

#define DA_WRITE_USB_DOWNLOAD_CONTROL_BIT_CMD 0xA0
#define DA_WRITE_PARTITION_TBL_CMD 0xA1
#define DA_READ_PARTITION_TBL_CMD 0xA2
#define DA_READ_BMT 0xA3
#define DA_SDMMC_WRITE_PMT_CMD 0xA4
#define DA_SDMMC_READ_PMT_CMD 0xA5
#define DA_READ_IMEI_PID_SWV_CMD 0xA6
#define DA_READ_DOWNLOAD_INFO 0xA7
#define DA_WRITE_DOWNLOAD_INFO 0xA8
#define DA_SDMMC_WRITE_GPT_CMD 0xA9
#define DA_NOR_READ_PTB_CMD 0xAA
#define DA_NOR_WRITE_PTB_CMD 0xAB

#define DA_NOR_BLOCK_INDEX_TO_ADDRESS 0xB0 // deprecated
#define DA_NOR_ADDRESS_TO_BLOCK_INDEX 0xB1 // deprecated
#define DA_NOR_WRITE_DATA 0xB2 // deprecated
#define DA_NAND_WRITE_DATA 0xB3
#define DA_SECURE_USB_RECHECK_CMD 0xB4
#define DA_SECURE_USB_DECRYPT_CMD 0xB5
#define DA_NFB_BL_FEATURE_CHECK_CMD 0xB6 // deprecated
#define DA_NOR_BL_FEATURE_CHECK_CMD 0xB7 // deprecated
#define DA_SF_WRITE_IMAGE_CMD 0xB8 // deprecated

/* Android S-USEDL { */
#define DA_SECURE_USB_IMG_INFO_CHECK_CMD 0xB9
#define DA_SECURE_USB_WRITE 0xBA
#define DA_SECURE_USB_ROM_INFO_UPDATE_CMD 0xBB
#define DA_SECURE_USB_GET_CUST_NAME_CMD 0xBC
#define DA_SECURE_USB_CHECK_BYPASS_CMD 0xBE
#define DA_SECURE_USB_GET_BL_SEC_VER_CMD 0xBF
/* Android S-USEDL } */

```

```

#define DA_VERIFY_IMG_CHKSUM_CMD 0xBD

#define DA_GET_BATTERY_VOLTAGE_CMD 0xD0
#define DA_POST_PROCESS 0xD1
#define DA_SPEED_CMD 0xD2
#define DA_MEM_CMD 0xD3
#define DA_FORMAT_CMD 0xD4
#define DA_WRITE_CMD 0xD5
#define DA_READ_CMD 0xD6
#define DA_WRITE_REG16_CMD 0xD7
#define DA_READ_REG16_CMD 0xD8
#define DA_FINISH_CMD 0xD9
#define DA_GET_DSP_VER_CMD 0xDA
#define DA_ENABLE_WATCHDOG_CMD 0xDB
#define DA_NFB_WRITE_BLOADER_CMD 0xDC // deprecated
#define DA_NAND_IMAGE_LIST_CMD 0xDD
#define DA_NFB_WRITE_IMAGE_CMD 0xDE
#define DA_NAND_READPAGE_CMD 0xDF
#define DA_CHK_PC_SEC_INFO_CMD 0xE0
#define DA_UPDATE_FLASHTOOL_CFG_CMD 0xE1
#define DA_CUST_PARA_GET_INFO_CMD 0xE2 // deprecated
#define DA_CUST_PARA_READ_CMD 0xE3 // deprecated
#define DA_CUST_PARA_WRITE_CMD 0xE4 // deprecated
#define DA_SEC_RO_GET_INFO_CMD 0xE5 // deprecated
#define DA_SEC_RO_READ_CMD 0xE6 // deprecated
#define DA_SEC_RO_WRITE_CMD 0xE7 // deprecated
#define DA_ENABLE_DRAM 0xE8
#define DA_OTP_CHECKDEVICE_CMD 0xE9
#define DA_OTP_GETSIZE_CMD 0xEA
#define DA_OTP_READ_CMD 0xEB
#define DA_OTP_PROGRAM_CMD 0xEC
#define DA_OTP_LOCK_CMD 0xED
#define DA_OTP_LOCK_CHECKSTATUS_CMD 0xEE
#define DA_GET_PROJECT_ID_CMD 0xEF
#define DA_GET_FAT_INFO_CMD 0xF0 // deprecated
#define DA_FDM_MOUNTDEVICE_CMD 0xF1
#define DA_FDM_SHUTDOWN_CMD 0xF2
#define DA_FDM_READSECTORS_CMD 0xF3
#define DA_FDM_WRITESECTORS_CMD 0xF4
#define DA_FDM_MEDIACHANGED_CMD 0xF5
#define DA_FDM_DISCARDSECTORS_CMD 0xF6
#define DA_FDM_GETDISKGEOMETRY_CMD 0xF7
#define DA_FDM_LOWLEVELFORMAT_CMD 0xF8
#define DA_FDM_NONBLOCKWRITESECTORS_CMD 0xF9
#define DA_FDM_RECOVERABLEWRITESECTORS_CMD 0xFA
#define DA_FDM_RESUMSECTORSTATES 0xFB
#define DA_NAND_EXTRACT_NFB_CMD 0xFC // dep
#define DA_NAND_INJECT_NFB_CMD 0xFD // dep

```

2.2 安卓系统加密



2.2.1 安卓全盘加密

- 全盘加密概念：

Full Disk Encryption(FDE)

使用一个密钥来为android设备上所有的用户数据加密的过程。

一旦设备被加密，所有的用户创建的数据都将会在提交的磁盘之前自动加密，在读取之前都会自动解密。

Google从Android 6.0开始全面开启全盘加密。

- 全盘加密带来的困难：

无法使用第一代技术对于直接提取出来的镜像进行解析。

各手机厂商的加密安全机制不尽相同。

2.2.2 解决方案-安卓手机工作模式

- **正常模式**：手机正常工作模式
- **fastboot模式**：是一种线刷模式，就是使用USB数据线连接手机的一种刷机模式。
- **recovery模式**：是一种卡刷，就是将刷机包放在sd卡上，然后在recovery中刷机的模式。

Fastboot模式

Vivo手机

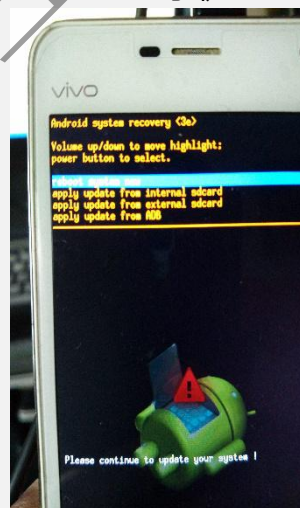


小米手机



Recovery模式

Vivo手机



小米手机



2.2.3 解决方案-手机分区

Android手机常用分区及用处

- **boot**

这个分区上有Android的引导程序，用于引导手机正常启动。

- **system**

包含了手机厂商预先安装的系统应用。

- **recovery**

recovery分区被认为是另一个启动分区，你可以启动设备进入recovery控制台去执行高级的系统恢复和管理操作。

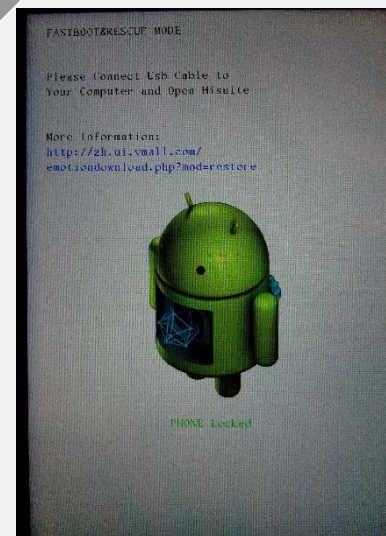
- **userdata**

这个分区保存着用户数据。通讯录、短信、设置和你安装的apps都在这个分区上。

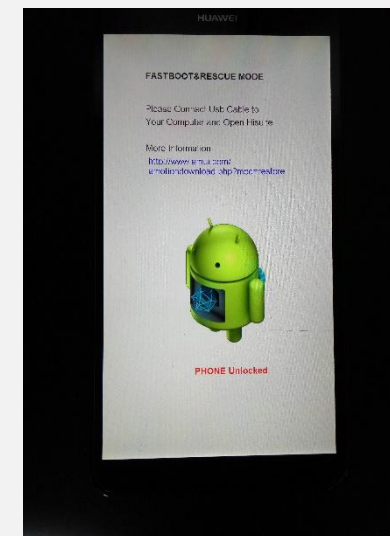
2.2.4 解决方案-BootLoader

Bootloader :

- BootLoader是在操作系统内核运行之前运行的一段小程序。
通过这段程序，可以将手机的软硬件环境带到一个合适状态，以便为启动操作系统做好准备。
- Bootloader被锁的手机必须要解锁才能刷第三方rom。
如果不解锁**bootloader**，就无法初始化手机硬件，手机也就无法使用。
- 国内大部分手机都是锁定bootloader
- 目前有些手机厂商支持在线申请解Bootloader锁(如小米)
有些厂商不支持在线申请解bootloader锁(如vivo)



华为fastboot未解锁



华为fastboot已解锁

2.2.5 解决方案-整体思路

1、解BL锁

2、刷入必要的资源文件

刷入方式有fastboot线刷，9008线刷，MTK线刷，odin线刷等

刷入资源有recovery镜像，boot镜像等

3、引导手机启动

根据刷入的镜像不同，将手机启动到的模式也不同

如果刷入的是recovery镜像，则把手机启动到recovery模式

如果刷入的是boot镜像，则把手机启动到正常模式

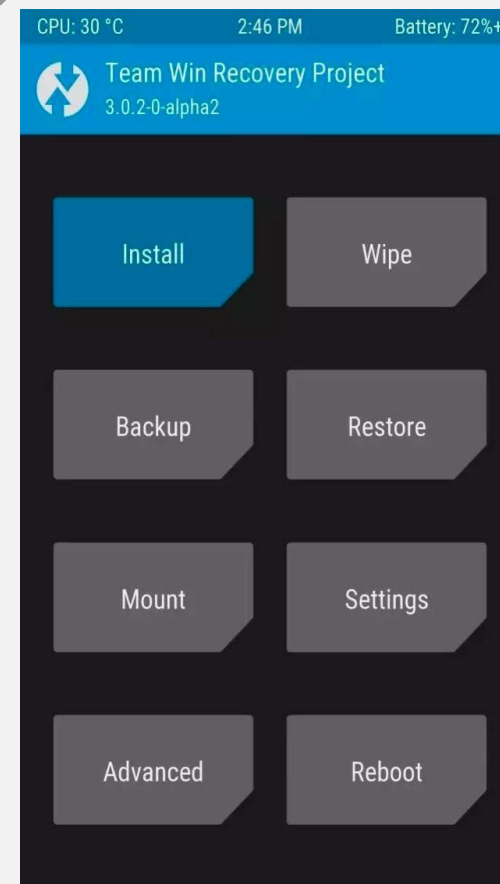
4、提取解密后的userdata分区

cat /proc/partitions 找到含有dm字样最大数的一行

adb pull /dev/block/dm-0 d:\userdata.img

或者 dd if=/dev/block/dm-0 of=/sdcard/userdata.img

TWRP



安卓文件级加密FBE机型 提取技术的研究

3

3.1 安卓文件级加密

- Android 7.0 推出文件级加密 FBE (File Based encryption)。

- 1、采用文件级加密时，可以使用不同的密钥对不同的文件进行加密，并且可以对这些文件进行单独解密。
- 2、不仅文件内容加密，连文件名也加密了。
- 3、只有当手机正常解开屏幕锁后，/data/data/下的文件才能被解密。否则看到的都是加密的文件名和文件内容。
- 4、与手机硬件信息绑定

手机未解屏幕锁

```
HWMHA:/data/data # ls
ls
+1nALtTqxDL4UT3WLig8gYsdUoN
+9q39GwtosNU247Yr+r,BD
+TWPIXBsfxsBPnUck3S14B19YygWSU9xoZrfDbXb+fB
,Mcc7XhEKNa9YIUqecjpfpn,40iAu7D3
,Qf++cb7e0MlokW+q0CvkiEzFZH
,Xn2fSsrc+0ogK10CE5We6kobitk07g2yyNppB
,boly4sA74sfxvq6rPhGAqn,40iAu7D30kPxeC
12fc1Ay71XY3pck,SAMLL+pH2XnBwgJKQa+oHA
17G,m,pEX+P0379w8ACz7rn,40iAu7D30kPxeeyyigI
2Y7fKwB3j9bvWP9AAYS5Hon,40iAu7D30kPxeeyyigI
3a90F+cm+XGar++GmiA1zM5JD7yOLzbN
4,+nH,Ob6czDP+iRAH7+q8pH2XnBwgJKQa+oHsn6g,A
```

手机未解屏幕锁

```
HWMHA:/data/data # ls
ls
android
androidhwext
cn.wps.moffice_eng
com.amap.android.ams
com.android.apps.tag
com.android.backupconfirm
com.android.bluetooth
com.android.bluetoothmidiservice
com.android.browser
com.android.calculator2
com.android.calendar
com.android.captiveportallogin
com.android.carrierconfig
```

3.2 全盘加密与文件级加密区别

1、对于全盘加密(FDE)：

- (1) 用户数据(/data/data文件夹下)会在手机启动过程中完成解密
- (2) 当手机出现锁屏画面时，用户数据已经解密完成，如图：

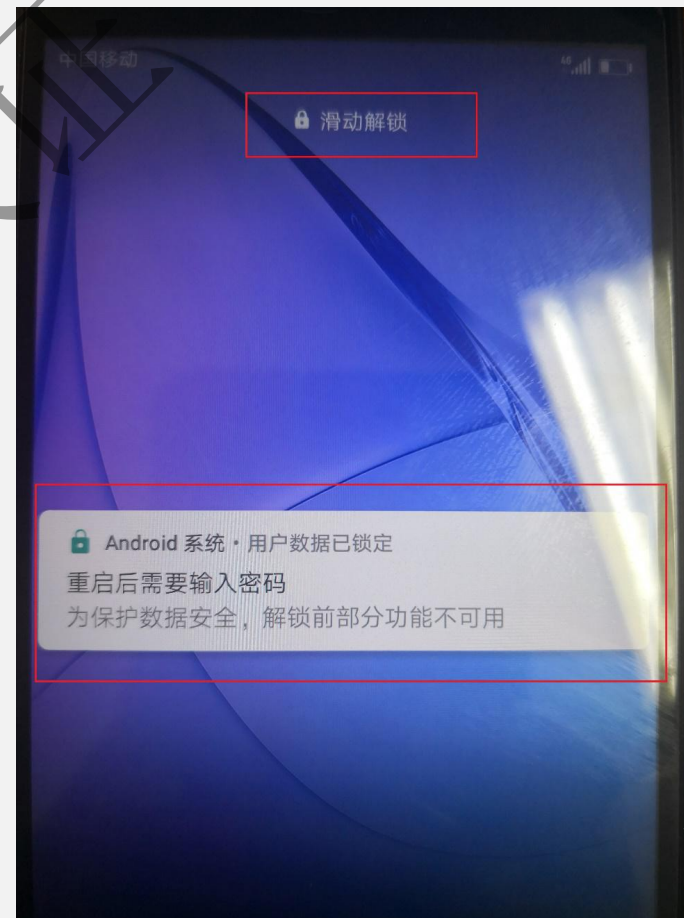
2、对于文件级加密(FBE)，有两个加密存储区域

(1) DE(Device Encrypted)区：

该区域存储的数据会在手机启动后直接解密
存储的数据比如闹钟，来电号码显示等

(2) CE(Credential Encrypted)区：

该区域存储的数据需要在手机验证锁屏密码后才会解密
比如微信，QQ等用户APP



3.3 解密码的尝试

1、全盘加密：

通过删除下列文件达到清除锁屏密码功能：

/data/system/gatekeeper.pattern.key

/data/system/gatekeeper.password.key

/data/system/locksettings.db

/data/system/locksettings.db-shm

/data/system/locksettings.db-wal

2、文件级加密

若删除上述文件，手机能进入系统，但是手机app无法正常运行

3、尝试输入密码，次数受系统限制，如何解决？

华为Mate10安卓8解密码视频



华为畅享7安卓7解密码视频



数据安全取证

微信恢复的研究

4

4.1 微信数据库文件

```
root@hnCHE-H:/data/data/com.tencent.mm # cd MicroMsg
root@hnCHE-H:/data/data/com.tencent.mm/MicroMsg # ls
ClickFlow
CompatibleInfo.cfg
NowRev.ini
SdcardInfo.cfg
bddbe0e4...545ded972d
channel_history.cfg
configlist
deviceconfig.cfg
eelda3ae...52382cfe79f
ipcallCountryCodeConfig.cfg
last_avatar
mobileinfo.ini
permissioncfg.cfg
regioncode
snsreport.cfg
staytime.cfg
stepcounter.cfg
systemInfo.cfg
version_history.cfg
root@hnCHE-H:/data/data/com.tencent.mm/MicroMsg #
```

存放用户数据

```
root@hnCHE-H:/data/data/com.tencent.mm/MicroMsg/bddbe...545ded972d # ls
AppBrandComm.db
AppBrandComm.db.ini
CommonOneMicroMsg.db
CommonOneMicroMsg.db.ini
EnMicroMsg.db
EnMicroMsg.db.bak
EnMicroMsg.db.bcur
EnMicroMsg.db.ini
EnMicroMsg.db.sm
IndexMicroMsg.db
SnsMicroMsg.db
SnsMicroMsg.db-shm
SnsMicroMsg.db-wal
SnsMicroMsg.db.ini
appbrand
bssidcache.bin
cdn
cdndnsinfo
checkmsgid.ini
eggingfo.ini
enFavorite.db
enFavorite.db.ini
logcat
pushSyncResp
sfs
talkroom
trackroom
version_history.cfg
```

数据库文件

目前微信存储聊天记录主要由4部分组成：

- 主库

- 1. EnMicroMsg.db(加密)

- 备份库

- 1. EnMicroMsg.db.bak (加密)

- 2. IndexMicroMsg.db (加密) 或 FTS5IndexMicroMsg.db (不加密)

4.2 加密算法

微信的数据库是通过手机的IMEI + 微信UIN 进行MD5加密，取32位小写的前7位就是破解数据库的密码。

```
root@hnCHE-H:/data/data/com.tencent.mm # cd MicroMsg
root@hnCHE-H:/data/data/com.tencent.mm/MicroMsg # ls
ClickFlow
CompatibleInfo.cfg
NowRev.ini
SdcardInfo.cfg
bddbe0e4...e545ded972d
channel_history.cfg
configlist
deviceconfig.cfg
eelda3ae...52382cfe79f
ipcallCountryCodeConfig.cfg
last_avatar
mobileinfo.ini
permissioncfg.cfg
regioncode
snsreport.cfg
staytime.cfg
stepcounter.cfg
systemInfo.cfg
version_history.cfg
root@hnCHE-H:/data/data/com.tencent.mm/MicroMsg #
```

存放IMEI

存放用户数据

存放UIN

4.3 解密数据库

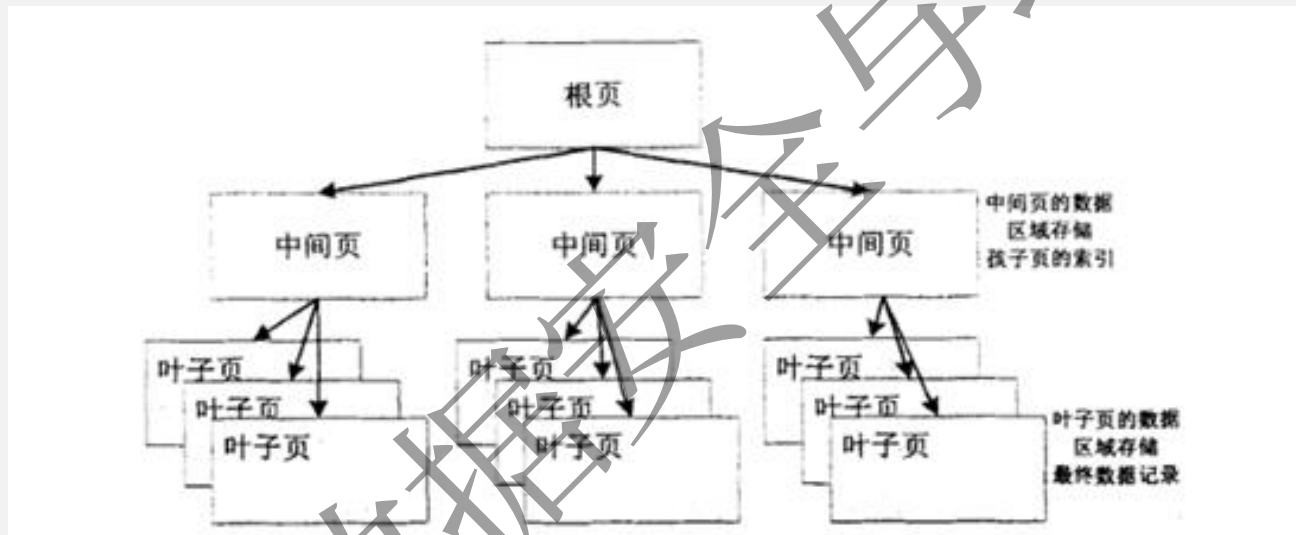
EnMicroMsg.db采用AES-256-CBC方式加密，解密代码较长，第三方工具sqlcipher

Table: chatroom

	chatroomname	addtime	memberlist	displayname	chatroomnick	roomflag	roomowner	roomdata
1	7957408596@chatr		0 zangzang-123-com	A 奔跑吧、			0 wxid_244d0f9ynvw ...	
2	8119020205@chatr		0 wxid_4tdwyjmb14t	吾王、AA强子恶霸			0 wxid_3psybnybziv ...	
3	7286976975@chatr		0 mrshi_1988;wxid_	【天安】石磊Big			0 wxid_95107651075 ...	
4	6954082615@chatr		0 hanjin77;wxid_0p	H、A 旭			0 wxid_tzuedpjgz79 ...	
5	7207782686@chatr		0 wxid_vdc79xo0ymf	亮少、不忘初心、			0 wxid_vdc79xo0ymf ...	
6	2677318379@chatr		0 shanshan14832003	蛋蛋头犬舍花美男			0 shiliu0809 ...	
7	7422815083@chatr		0 wxid_13zofdahiry	北京泰达陵园、王			0 wxid_lipcyf9pbrp ...	
8	7935084663@chatr		0 wxid_08644986447	神话(世界名犬)经			0 wxid_g6dmy5hk9v4 ...	
9	7508185759@chatr		0 wxid_0h0q0touulx	A 恶冠满福犬舍			0 wxid_jdrb0nplwi8 ...	
10	766329313@chatro		0 wxid_rzfvjb2uiyg	Qinqin、B.M.K.			0 wxid_w3xxbtq23ae ...	
11	1776782364@chatr		0 wxid_yugpitukn23	『领航』新疆孤洲			0 wxid_yugpitukn23 ...	
12	6836691439@chatr		0 wxid_1rid0b3jxgh	子豪大业15998846			0 wxid_1rid0b3jxgh ...	
13	6911026790@chatr		0 wxid_8xmr8330qtk	老男孩-小春恶霸			0 qzyangyuchen ...	
14	7676206597@chatr		0 wxid_xv43yf617yf	小九、Bingo高档			0 abc_67650411 ...	
15	1955811871@chatr		0 mahongfan123;wx	A 永牛团队 鞍山			0 wxid_zwfv14sd5sc ...	
16	6339416739@chatr		0 wxid_ggc3uakq2p7	文锦州★小旭☆、			0 wxid_12462624629 ...	
17	7911566875@chatr		0 wxid_jnh8n1v0j26	【荣鑫】美国恶霸			0 wxid_jnh8n1v0j26 ...	

4.4 数据恢复-SQLite数据库结构

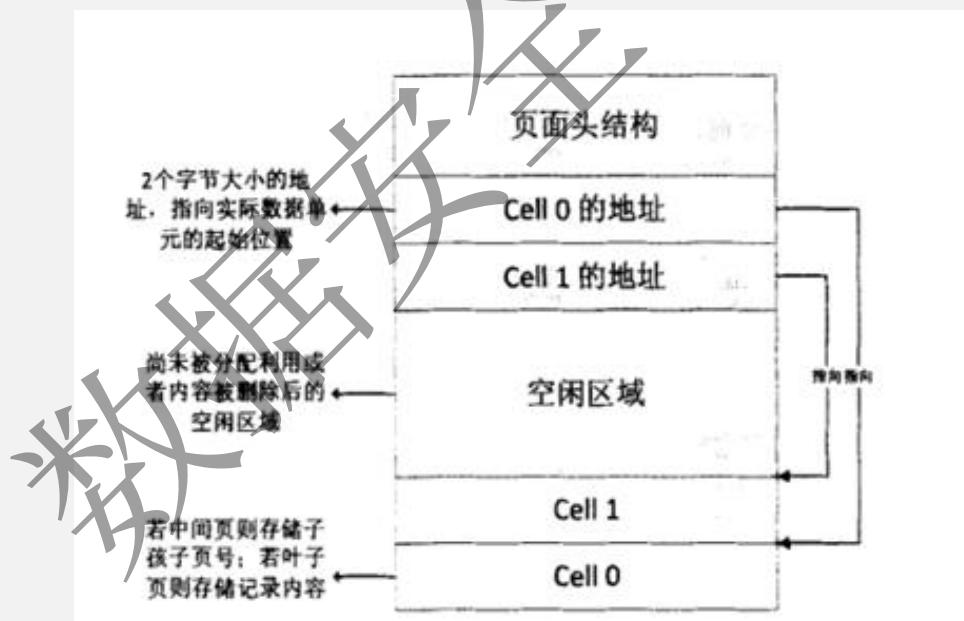
微信采用SQLite数据库存储，SQLite采用B-Tree为其存储结构，这是为了磁盘或其它存储设备而设计的一种多叉平衡查找树。Btree存储最小的结构为页，Btree页为固定大小，为4096字节。



4.4.1 页存储结构

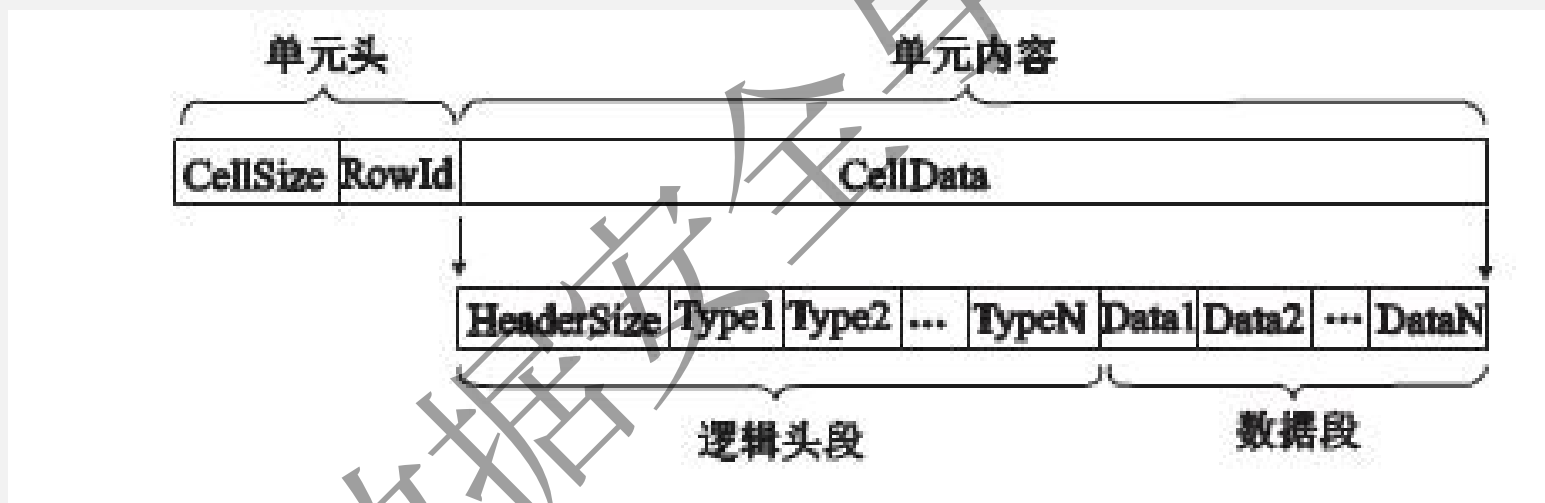
所有表和索引的根页面都存储在 sqlite_master 系统表中，根据此表可以遍历到所有的自由页。

一般情况下，当一个事务从数据库中删除了数据并提交后，数据库文件的大小保持不变。即使整页的数据都被删除，该页也会变成“自由页”等待再次被使用，而不会实际地被从数据库文件中删除。如下图，假如Cell 1被删除后，则Cell的位置转换为空闲区域。如果没有新数据写入的情况，多数情况下内容将会保存。



4.4.2 单元存储结构

数据库存储的最小单元为单元(Cell)，单元是变长的字节串，结构如下。其中单元头与逻辑头段的存储类型为可变长整形 (VARINT)。VARINT是8个bit为一组 (一个字节)，最高位是判断位，当最高位为0时表示当前字节为该整数的最后一个字节，当最高位为1时表示后面的一个字节也表示这个整数。



4.5 SQLite恢复原理

- 在逻辑层面，数据都存储在数据库的表结构中，而在物理层面，表是由B-Tree的树形结构组成，B-Tree结构又是由页组成，因此最终数据时存放在各页中，进行数据恢复，要解决两大核心问题。
 - 找到删除数据的页。
 - 恢复底层二进制文件中删除的数据。
- 恢复的过程如下：
 - 查找预恢复数据对应的表名
 - 查找表对应的树根页
 - 查找所有中间页
 - 查找所有叶子页
 - 检查删除区域，并按单元对Type字段进行估算，提取数据

4.6 微信恢复举例-叶子页

这是FTS5IndexMicroMsg.db数据库FTS5IndexMessage_content表删除前的一条数据

id	c0
87	今天晚上几点回家？

表结构

名	类型	长度	小数点	不是 null	
id	INTEGER	0	0	☐	1
c0		0	0	☐	

按照前面的思路通过SQL语句查找其根页.

type	name	tbl_name	rootpage	sql
table	FTS5Index	FTS5IndexMe: 21		CREATE TABLE 'FTS5IndexMessage'

此表在页号为0x15的页上，一页是0x1000（4096），所以该表根页是从0x14000开始。

其中0x0D表示该页是叶子页

```
00014000h: 0D 0C 50 00 44 02 A0 03 0F 9B 0F 20 0E F1 0E 01 ; ..P.D.?.? .?.
00014010h: 0B 31 0B 17 0A E5 0A CB 0A C0 0A B8 0A 94 0A 80 ; .1...?????€
00014020h: 0A 5D 09 BA 09 83 09 69 09 4F 09 3E 09 33 09 19 ; .].??i.O.>.3..
00014030h: 08 FC 08 DF 08 B2 08 9E 08 7D 08 60 08 40 07 9D ; .????}.~.@.?
00014040h: 07 5B 07 44 0D CC 0D 29 0D 1E 0D 07 0C FC 0C DC ; .[.D.?).....?.?
00014050h: 0C 88 0C 3C 0C 34 0C 12 0B E0 0B D5 0B C7 0B 54 ; .?<.4...???T
00014060h: 0C 71 0C CB 0C B7 0B 46 0C A6 0E E3 0C 55 0E 40 ; .q.??F.??U.@
00014070h: 06 50 0E 2C 06 2E 06 0E 05 EE 05 C8 05 98 05 60 ; .P.,.....???`
00014080h: 0F E9 05 43 04 1A 03 77 02 D4 0E 1E 02 AB 02 A0 ; .?C...w.????
00014090h: 02 A0 07 44 07 44 07 44 07 44 07 44 07 44 07 44 ; .?D.D.D.D.D.D
000140a0h: 07 44 00 00 00 00 00 00 00 00 00 00 00 00 00 ; .D.....
000140b0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; .....
```


- 今 E4 BB 8A
天 E5 A4 A9
- 使用16进制查看器打开数据库，可以发现原来存储此单元的结构前两字节已经被抹掉。
SQLite采用UTF-8编码存储，每个字符占3字节，“今天晚上几点回家？”正好为27字节。
0x03为固定大小，0x00为主键，0x43为TEXT类型换算后为27个长度

删除前

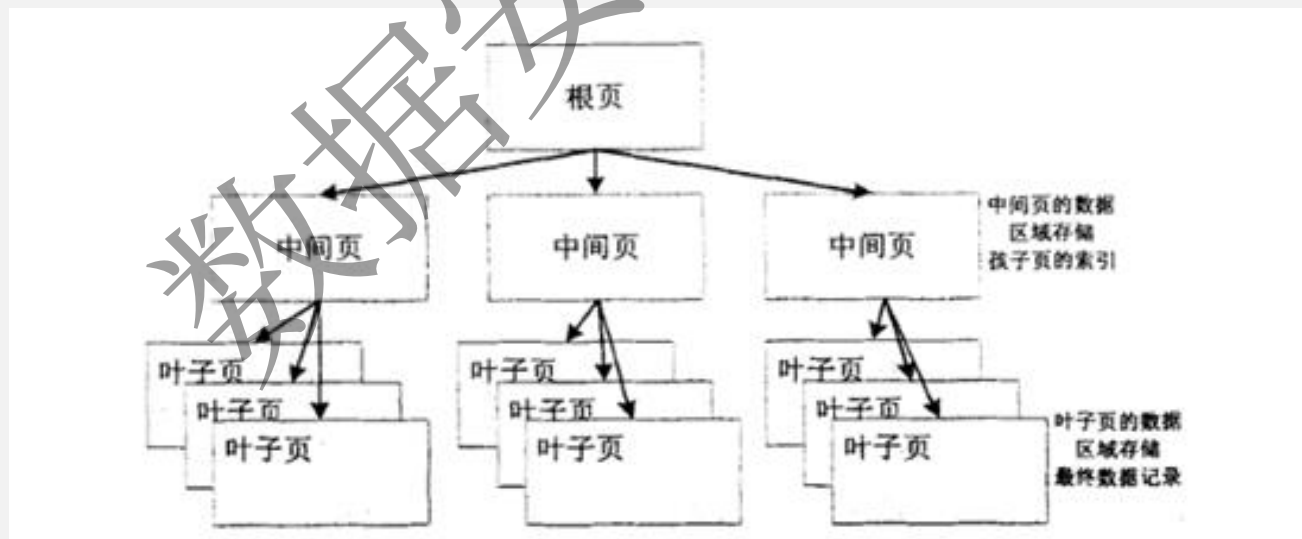
```
00014cb0h: B9 E8 B5 B0 E5 90 A7 12 67 03 00 2B E5 A6 B3 E7 ; 硅蛋錫?g..+濤崇
00014cc0h: 9A 84 E9 80 9A E5 AE B5 E5 95 8A 0F 66 03 00 25 ; 菟閭氫鎢鋰?f..%
00014cd0h: E4 B8 8B E4 B8 8D E4 BA 86 E4 BA 86 1E 57 03 00 ; 涓嬩笭浜嚟箇.W..
00014ce0h: 43 E4 BB 8A E5 A4 A9 E6 99 9A E4 B8 8A E5 87 A0 ; C浔娣#鎢氬筑鍅?
00014cf0h: E7 82 B9 E5 9B 9E E5 AE B6 EF BC 9F 09 56 03 00 ; 鏃瑰瓧濠瀹訛紵.V..
00014d00h: 19 E5 97 B7 E5 97 B7 15 55 03 00 31 E5 BF 98 E5 ; .鍶峰梳.U..1蹇樹
00014d10h: B8 A6 E4 BA 86 E5 95 8A E4 B8 AA E5 B1 81 09 54 ; 甫浜嚟晩涓 眳.T
00014d20h: 03 00 19 E5 95 8A EF BC 9F 81 20 53 04 00 82 45 ; ...鍶婉紵?S..偃
```

删除后

```
00014cb0h: B9 E8 B5 B0 E5 90 A7 12 67 03 00 2B E5 A6 B3 E7 ; 硅蛋錫?g..+濤崇
00014cc0h: 9A 84 E9 80 9A E5 AE B5 E5 95 8A 0F 66 03 00 25 ; 菟閭氫鎢鋰?f..%
00014cd0h: E4 B8 8B E4 B8 8D E4 BA 86 E4 BA 86 00 0B 03 00 ; 涓嬩笭浜嚟箇....
00014ce0h: 43 E4 BB 8A E5 A4 A9 E6 99 9A E4 B8 8A E5 87 A0 ; C浔娣#鎢氬筑鍅?
00014cf0h: E7 82 B9 E5 9B 9E E5 AE B6 EF BC 9F 09 56 03 00 ; 鏃瑰瓧濠瀹訛紵.V..
00014d00h: 19 E5 97 B7 E5 97 B7 15 55 03 00 31 E5 BF 98 E5 ; .鍶峰梳.U..1蹇樹
00014d10h: B8 A6 E4 BA 86 E5 95 8A E4 B8 AA E5 B1 81 09 54 ; 甫浜嚟晩涓 眳.T
00014d20h: 03 00 19 E5 95 8A EF BC 9F 81 20 53 04 00 82 45 ; ...鍶婉紵?S..偃
```

4.7 微信恢复举例-中间页

- 中间页是用来导航的，内部页的指针域都是指向下级页的指针，数据域仅仅包含关键字。
- 所有的数据库记录都存储在叶子页内。在叶节点一级，页和页内的单元都是按照关键字的顺序排列的，所以B+tree可以沿水平方向遍历，时间复杂度为 $O(1)$ 。
- 根页和内部页统称为内部页，它们的结构是相同的，其逻辑结构如下：
| Ptr(0) | Key(0) | Ptr(1) | Key(1) | ... | Key(N-1) | Ptr(N) |
Ptr为固定4字节长度，Key为varint型。
- 中间页包含N个关键值(Key(0)~ Key(N-1))和N+1个子页指针(Ptr(0)~ Ptr(N))，其值为子页的页号。其中，Ptr(N)存储在页头中偏移为8的地方



- 依然以FTS5IndexMessage_content表为例，通过查询可知跟页为0x15，则定位到0x14000的偏移上。
其中0x05表示该页是中间页

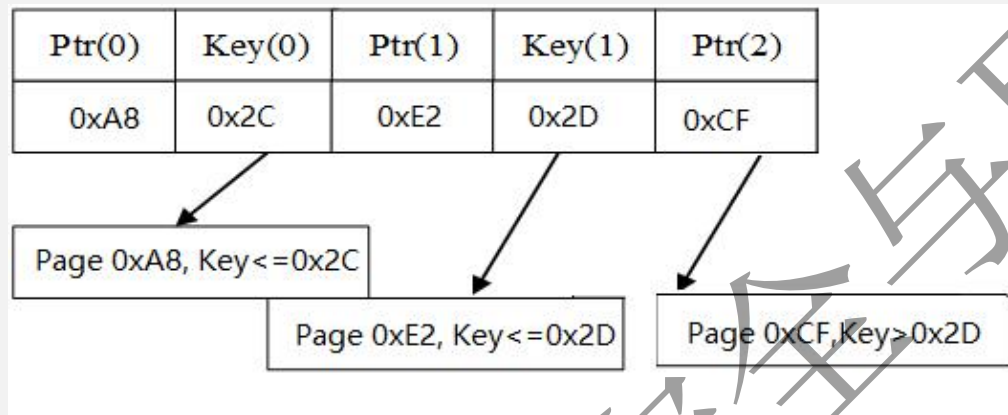
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	
00014000h:	05	00	00	00	02	0F	F6	00	00	00	00	CF	0F	FB	0F	F6	;?..?..??
00014010h:	0E	C1	0E	B9	0E	9F	0E	7C	0E	74	0E	54	0E	46	0E	26	; .??? .t.T.F.&
00014020h:	0E	1B	0D	6A	0D	5C	0D	51	0D	1D	0C	E9	0C	D8	0C	BB	; ...j.\.Q...???
00014030h:	0C	9B	0C	81	0C	76	0C	5C	0C	29	0B	F0	0B	D3	0B	A4	; .??v.\.)...???
00014040h:	0B	87	0B	7C	0B	5F	0B	33	0B	25	0B	02	0A	E5	0A	CE	; .? ._.3.%...??
00014050h:	0A	9D	0A	92	0A	87	0A	73	0A	5F	0A	3F	0A	2B	05	E9	; .???s._.?+.?
00014060h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	;

- 此页为中间页，有两个单元分别是0x0FFB，0x0FF6，最右叶子页号为0xCF。

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	
00014ff0h:	8C	E7	9A	84	E9	83	00	00	00	E2	2D	00	00	00	A8	2C	; 出脱閨...?..??
00015000h:	0D	00	00	00	32	0E	CF	00	0F	FA	0F	F4	0F	EE	0F	E8	;2.?..????
00015010h:	0F	E2	0F	DC	0F	D6	0F	D0	0F	CA	0F	C4	0F	BE	0F	B8	; .?????????
00015020h:	0F	B2	0F	AC	0F	A6	0F	A0	0F	9A	0F	94	0F	8E	0F	88	; .?????????
00015030h:	0F	82	0F	7C	0F	76	0F	70	0F	6A	0F	64	0F	5E	0F	58	; .? .v.p.j.d.^.X
00015040h:	0F	52	0F	4C	0F	46	0F	40	0F	3A	0F	34	0F	2E	0F	28	; .R.L.F.@.:.4...(
00015050h:	0F	22	0F	1C	0F	16	0F	10	0F	0A	0F	04	0E	FE	0E	F7	; .".....??

- 由图可知，两个单元对应的页号Ptr为0xE2，0xA8

把上述中间页的逻辑结构图如下所示：



0xA8, 0xE2, 0xCF 则为 3 个二级页。

二级页面可能还是中间页，按照这方法递归解析即可找到叶子页。

如果是叶子页，按照前面介绍的方法解析即可恢复删除数据。

附录：字段的数据类型和宽度说明

用可变长整数(varint)表示各字段类型和宽度的规定如下表所示:

类型值	含义	数据宽度(字节数)
0	NULL	0
N in 1..4	有符号整数	N
5	有符号整数	6
6	有符号整数	8
7	IEEE符点数	8
8	有符号整数	0
9	有符号整数	1
10, 11	未使用	N/A
N>12的偶数	BLOB	$(N-12)/2$
N>13的奇数	TEXT	$(N-13)/2$

CONTANT ME

董志国

电话：13511083747 (同微信)

邮件：dongzhiguo@zhizhangyi.com

@版权归作者所有，仅供学习，不得转载和复制

谢谢！



CHINA-FORENSIC.COM

数据安全取证