(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 104111832 A
(43)申请公布日 2014.10.22

(21)申请号 201410314808.7

(22)申请日 2014.07.03

(71)申请人 北京思特奇信息技术股份有限公司
地址 100086 北京市海淀区中关村南大街6号中电信息大厦16层

(72)发明人 吕麟

(74)专利代理机构 北京轻创知识产权代理有限公司 11212
代理人 杨立

(51)Int.Cl.
G06F 9/44(2006.01)

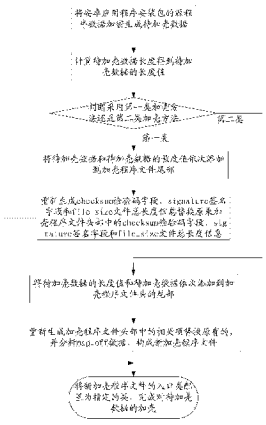
权利要求书2页 说明书6页 附图3页

(54)发明名称

一种安卓应用程序安装包加壳方法及系统及解壳方法

(57)摘要

本发明涉及一种安卓应用程序安装包加壳方法，包括步骤：步骤1：将源程序数据加密生成待加壳数据；步骤2：计算待加壳数据长度得到待加壳数据的长度值；步骤3：判断采用第一类加壳方法还是第二类加壳方法，如果是第一类，执行步骤4；否则，执行步骤6；步骤4：将待加壳数据和待加壳数据的长度值依次添加到加壳程序文件尾部；步骤5：重置相关项，执行步骤8；步骤6：将待加壳数据的长度值和待加壳数据依次添加到加壳程序文件头的尾部；步骤7：重新生成加壳程序文件头部中的相关项替换原有的，修改相关的数据偏移量；步骤8：将加壳程序文件的入口类配置为指定的类，完成对待加壳数据的加壳。本发明达到保护源文件的目的。



CN 104111832 A

1. 一种安卓应用程序安装包加壳方法，其特征在于，具体包括以下步骤：

步骤 1：将安卓应用程序安装包的源程序数据加密生成待加壳数据；

步骤 2：计算待加壳数据长度得到待加壳数据的长度值；

步骤 3：判断采用第一类加壳方法还是第二类加壳方法，如果是第一类，执行步骤 4；如果是第二类，执行步骤 6；

步骤 4：将待加壳数据和待加壳数据的长度值依次添加到加壳程序文件尾部；

步骤 5：重新生成 checksum 检验码字段、signature 签名字段和 file_size 文件总长度信息替换原来加壳程序文件头部中的 checksum 检验码字段、signature 签名字段和 file_size 文件总长度信息，构成新加壳程序文件，执行步骤 8；

步骤 6：将待加壳数据的长度值和待加壳数据依次添加到加壳程序文件头的尾部；

步骤 7：重新生成加壳程序文件头部中的相关项替换原有的，并分析 map_off 数据，修改相关的数据偏移量，构成新加壳程序文件；

步骤 8：将新加壳程序文件的入口类配置为指定的类，完成对待加壳数据的加壳。

2. 根据权利要求 1 所述的一种安卓应用程序安装包加壳方法，其特征在于，步骤 7 中的所述相关项包括 checksum 检验码字段、signature 签名字段、file_size 文件总长度信息、header_size、string_ids_off、type_ids_off、proto_ids_off、field_ids_off、method_ids_off、class_defs_off 和 data_off 相关项。

3. 根据权利要求 1 或 2 所述的一种安卓应用程序安装包加壳方法，其特征在于，所述步骤 8 中，如果源程序数据已经配置了入口类需要把这个入口类用 <meta-data> 保存在加壳程序文件中，并将加壳程序文件的入口类配置为源文件配置的入口类。

4. 根据权利要求 3 所述的一种安卓应用程序安装包加壳方法，其特征在于，所述加壳程序文件采用 dex 文件。

5. 一种安卓应用程序安装包加壳系统，其特征在于，包括加密模块、长度计算模块、第一类加壳模块、第一类加壳模块和配置类模块；

所述加密模块将源程序数据加密生成待加壳数据；

所述长度计算模块用于计算待加壳数据长度得到待加壳数据的长度值，并将长度值和待加壳数据发送到第一类加壳模块或第一类加壳模块；

所述第一类加壳模块将待加壳数据写入加壳程序文件的尾部，并在加壳程序文件尾部添加待加壳数据的长度值；重新生成加壳程序文件头部中的相关项替换原有的，并将加壳程序文件发送到配置类模块；

所述第二类加壳模块将待加壳数据的长度值和待加壳数据依次添加到加壳程序文件头的尾部；重新生成加壳程序文件头部中的相关项替换原有的，并分析 map_off 数据，修改相关的数据偏移量，并将加壳程序文件发送到配置类模块；

所述配置类模块将加壳程序文件的入口类配置为指定的类，使解壳程序在运行的时候还原此类。

6. 根据权利要求 5 所述的一种安卓应用程序安装包加壳系统，其特征在于，所述第一类加壳模块中的所述相关项包括 checksum 检验码字段、signature 签名字段和 file_size 文件总长度信息。

7. 根据权利要求 6 所述的一种安卓应用程序安装包加壳系统，其特征在于，所述第二

类加壳模块中的所述相关项包括 checksum 检验码字段、signature 签名字段、file_size 文件总长度信息、header_size、string_ids_off、type_ids_off、proto_ids_off、field_ids_off、method_ids_off、class_defs_off 和 data_off 相关项。

8. 根据权利要求 5-7 任一项所述的一种安卓应用程序安装包加壳系统，其特征在于，所述加壳程序文件采用 dex 文件。

9. 一种安卓应用程序安装包解壳方法，其特征在于，具体包括以下步骤：

步骤 1：判断是第一类加壳方法还是第二类加壳方法处理的加壳程序文件，如果是第一类，执行步骤 2；如果是第二类，执行步骤 3；

步骤 2：读取加壳程序文件尾部数据，获得待加壳数据的长度值，执行步骤 4；

步骤 3：读取加壳程序文件头的末尾处数据，获得待加壳数据的长度值；

步骤 4：从加壳程序文件中读取待加壳数据，对待加壳数据进行解密得到解密数据；

步骤 5：将解密数据以文件形式保存到 a.apk 文件中；

步骤 6：通过类装载器动态加载 a.apk 文件，读取源文件。

10. 根据权利要求 9 所述的一种安卓应用程序安装包解壳方法，其特征在于，所述步骤 6 中的类装载器采用 DexClassLoader。

一种安卓应用程序安装包加壳方法及系统及解壳方法

技术领域

[0001] 本发明涉及一种安卓应用程序安装包加壳方法及系统及解壳方法，涉及到手机App安全领域。

背景技术

[0002] 目前，市场上智能手机的品牌种类繁多，比如说：苹果，三星，HTC，诺基亚，小米，LG等等。但是，从智能手机操作系统层面上进行划分的话，大体上只有三种：Android，iOS，Windows Phone（这三种的市场占有率最高，其它的由于市场占有率过低，暂时忽略）。

[0003] 在上面提到的三种操作系统中，由于Android是唯一的开源平台，所以，目前采用Android平台的智能手机最多，市场占有率也是最大的。

[0004] 但是，“开放性”在给Android带来巨大的竞争力的同时，也带来了很多安全方面的问题，其中，一个比较明显的是，App很容易被破解，然后被恶意加入一些代码，执行与当初设计不相符的逻辑。比如说：App被反编译，被破解后，界面被加入一些垃圾广告；用户私秘信息被获取后用于他用等等。

发明内容

[0005] 本发明所要解决的技术问题是提供一种主要针对Android平台App，通过对APK安卓应用程序安装包加壳来预防App被破解的安卓应用程序安装包加壳方法。

[0006] 本发明解决上述技术问题的技术方案如下：一种安卓应用程序安装包加壳方法，具体包括以下步骤：

[0007] 步骤1：将安卓应用程序安装包的源程序数据加密生成待加壳数据；

[0008] 步骤2：计算待加壳数据长度得到待加壳数据的长度值；

[0009] 步骤3：判断采用第一类加壳方法还是第二类加壳方法，如果是第一类，执行步骤4；如果是第二类，执行步骤6；

[0010] 步骤4：将待加壳数据和待加壳数据的长度值依次添加到加壳程序文件尾部；

[0011] 步骤5：重新生成checksum检验码字段、signature签名字段和file_size文件总长度信息替换原来加壳程序文件头部中的checksum检验码字段、signature签名字段和file_size文件总长度信息，构成新加壳程序文件，执行步骤8；

[0012] 步骤6：将待加壳数据的长度值和待加壳数据依次添加到加壳程序文件头的尾部；

[0013] 步骤7：重新生成加壳程序文件头部中的相关项替换原有的，并分析map_off数据，修改相关的数据偏移量，构成新加壳程序文件；

[0014] 步骤8：将新加壳程序文件的入口类配置为指定的类，使解壳程序在运行的时候还原此类，完成对待加壳数据的加壳。

[0015] 本发明的有益效果是：本发明对Android平台的APK进行加壳，反编译工具还原Android源代码的原理是根据虚拟机的可执行文件.DEX格式进行代码还原，反编译工具只

能窥测外层 APK 程序文件而看不到源文件里的东西，从而达到保护源文件的目的。

[0016] 在上述技术方案的基础上，本发明还可以做如下改进。

[0017] 进一步，步骤 7 中的所述相关项包括 checksum 检验码字段、signature 签名字段、file_size 文件总长度信息、header_size、string_ids_off、type_ids_off、proto_ids_off、field_ids_off、method_ids_off、class_defs_off 和 data_off 相关项。

[0018] 进一步，所述步骤 8 中，如果源文件已经配置了入口类需要把这个入口类用 <meta-data> 保存在加壳程序文件中，并将加壳程序文件的入口类配置为源文件配置的入口类。

[0019] 进一步，所述加壳程序文件采用 dex 文件。

[0020] 本发明所要解决的技术问题是提供一种主要针对 Android 平台 App，通过对 APK 安卓应用程序安装包加壳来预防 App 被破解的安卓应用程序安装包加壳系统。

[0021] 本发明解决上述技术问题的技术方案如下：一种安卓应用程序安装包加壳系统，包括加密模块、长度计算模块、第一类加壳模块、第二类加壳模块和配置类模块；

[0022] 所述加密模块将源程序数据加密生成待加壳数据；

[0023] 所述长度计算模块用于计算待加壳数据长度得到待加壳数据的长度值，并将长度值和待加壳数据发送到第一类加壳模块或第二类加壳模块；

[0024] 所述第一类加壳模块将待加壳数据写入加壳程序文件的尾部，并在加壳程序文件尾部添加待加壳数据的长度值；重新生成加壳程序文件头部中的相关项替换原有的，并将加壳程序文件发送到配置类模块；

[0025] 所述第二类加壳模块将待加壳数据的长度值和待加壳数据依次添加到加壳程序文件头的尾部；重新生成加壳程序文件头部中的相关项替换原有的，并分析 map_off 数据，修改相关的数据偏移量，并将加壳程序文件发送到配置类模块；

[0026] 所述配置类模块将加壳程序文件的入口类配置为指定的类，使解壳程序在运行的时候还原此类。

[0027] 本发明的有益效果是：本发明对 Android 平台的 APK 进行加壳，反编译工具还原 Android 源代码的原理是根据虚拟机的可执行文件 .DEX 格式进行代码还原，反编译工具只能窥测外层 APK 程序文件而看不到源文件里的东西，从而达到保护源文件的目的。

[0028] 在上述技术方案的基础上，本发明还可以做如下改进。

[0029] 进一步，所述第一类加壳模块中的所述相关项包括 checksum 检验码字段、signature 签名字段和 file_size 文件总长度信息。

[0030] 进一步，所述第二类加壳模块中的所述相关项包括 checksum 检验码字段、signature 签名字段、file_size 文件总长度信息、header_size、string_ids_off、type_ids_off、proto_ids_off、field_ids_off、method_ids_off、class_defs_off 和 data_off 相关项。

[0031] 进一步，所述加壳程序文件采用 dex 文件。

[0032] 本发明所要解决的技术问题是提供一种主要针对 Android 平台 App，通过对 APK 安卓应用程序安装包加壳来预防 App 被破解的安卓应用程序安装包解壳方法。

[0033] 本发明解决上述技术问题的技术方案如下：一种安卓应用程序安装包解壳方法，具体包括以下步骤：

[0034] 步骤 1 :判断是第一类加壳方法还是第二类加壳方法处理的加壳程序文件,如果是第一类,执行步骤 2 ;如果是第二类,执行步骤 3 ;

[0035] 步骤 2 :读取加壳程序文件尾部数据,获得待加壳数据的长度值,执行步骤 4 ;

[0036] 步骤 3 :读取加壳程序文件头的末尾处数据,获得待加壳数据的长度值 ;

[0037] 步骤 4 :从加壳程序文件中读取待加壳数据,对待加壳数据进行解密得到解密数据 ;

[0038] 步骤 5 :将解密数据以文件形式保存到 a.apk 文件中 ;

[0039] 步骤 6 :通过类装载器动态加载 a.apk 文件,读取源文件。

[0040] 在上述技术方案的基础上,本发明还可以做如下改进。

[0041] 进一步,所述步骤 6 中的类装载器采用 DexClassLoader。

附图说明

[0042] 图 1 为本发明所述的一种安卓应用程序安装包加壳方法流程图 ;

[0043] 图 2 为本发明所述的一种安卓应用程序安装包加壳系统结构框图 ;

[0044] 图 3 为本发明所述的一种安卓应用程序安装包解壳方法流程图。

[0045] 附图中,各标号所代表的部件列表如下 :

[0046] 1、加密模块,2、长度计算模块,3、第一类加壳模块,4、第二类加壳模块,5、配置类模块。

具体实施方式

[0047] 以下结合附图对本发明的原理和特征进行描述,所举实例只用于解释本发明,并非用于限定本发明的范围。

[0048] 如图 1 所示,为本发明所述的一种安卓应用程序安装包加壳方法,具体包括以下步骤 :

[0049] 步骤 1 :将安卓应用程序安装包的源程序数据加密生成待加壳数据 ;

[0050] 步骤 2 :计算待加壳数据长度得到待加壳数据的长度值 ;

[0051] 步骤 3 :判断采用第一类加壳方法还是第二类加壳方法,如果是第一类,执行步骤 4 ;如果是第二类,执行步骤 6 ;

[0052] 步骤 4 :将待加壳数据和待加壳数据的长度值依次添加到加壳程序文件尾部 ;

[0053] 步骤 5 :重新生成 checksum 检验码字段、signature 签名字段和 file_size 文件总长度信息替换原来加壳程序文件头部中的 checksum 检验码字段、signature 签名字段和 file_size 文件总长度信息,构成新加壳程序文件,执行步骤 8 ;

[0054] 步骤 6 :将待加壳数据的长度值和待加壳数据依次添加到加壳程序文件头的尾部 ;

[0055] 步骤 7 :重新生成加壳程序文件头部中的相关项替换原有的,并分析 map_off 数据,修改相关的数据偏移量,构成新加壳程序文件 ;

[0056] 步骤 8 :将新加壳程序文件的入口类配置为指定的类,使解壳程序在运行的时候还原此类,完成对待加壳数据的加壳。

[0057] 步骤 7 中的所述相关项包括 checksum 检验码字段、signature 签名字段、file_size

文件总长度信息、header_size、string_ids_off、type_ids_off、proto_ids_off、field_ids_off、method_ids_off、class_defs_off 和 data_off 相关项。

[0058] 所述步骤 8 中，如果源程序数据已经配置了入口类需要把这个入口类用 <meta-data> 保存在加壳程序文件中，并将加壳程序文件的入口类配置为源文件配置的入口类。

[0059] 所述加壳程序文件采用 dex 文件。

[0060] 如图 2 所示，为本发明所述的一种安卓应用程序安装包加壳系统，包括加密模块 1、长度计算模块 2、第一类加壳模块 3、第二类加壳模块 4 和配置类模块 5；

[0061] 所述加密模块 1 将源程序数据加密生成待加壳数据；

[0062] 所述长度计算模块 2 用于计算待加壳数据长度得到待加壳数据的长度值，并将长度值和待加壳数据发送到第一类加壳模块 3 或第二类加壳模块 4；

[0063] 所述第一类加壳模块 3 将待加壳数据写入加壳程序文件的尾部，并在加壳程序文件尾部添加待加壳数据的长度值；重新生成加壳程序文件头部中的相关项替换原有的，并将加壳程序文件发送到配置类模块 5；

[0064] 所述第二类加壳模块 4 将待加壳数据的长度值和待加壳数据依次添加到加壳程序文件头的尾部；重新生成加壳程序文件头部中的相关项替换原有的，并分析 map_off 数据，修改相关的数据偏移量，并将加壳程序文件发送到配置类模块 5；

[0065] 所述配置类模块 5 将加壳程序文件的入口类配置为指定的类，使解壳程序在运行的时候还原此类。

[0066] 所述第一类加壳模块 1 中的所述相关项包括 checksum 检验码字段、signature 签名字段和 file_size 文件总长度信息。

[0067] 所述第二类加壳模块 2 中的所述相关项包括 checksum 检验码字段、signature 签名字段、file_size 文件总长度信息、header_size、string_ids_off、type_ids_off、proto_ids_off、field_ids_off、method_ids_off、class_defs_off 和 data_off 相关项。

[0068] 所述加壳程序文件采用 dex 文件。

[0069] 如图 3 所示，本发明所述的一种安卓应用程序安装包解壳方法，具体包括以下步骤：

[0070] 步骤 1：判断是第一类加壳方法还是第二类加壳方法处理的加壳程序文件，如果是第一类，执行步骤 2；如果是第二类，执行步骤 3；

[0071] 步骤 2：读取加壳程序文件尾部数据，获得待加壳数据的长度值，执行步骤 4；

[0072] 步骤 3：读取加壳程序文件头的末尾处数据，获得待加壳数据的长度值；

[0073] 步骤 4：从加壳程序文件中读取待加壳数据，对待加壳数据进行解密得到解密数据；

[0074] 步骤 5：将解密数据以文件形式保存到 a.apk 文件中；

[0075] 步骤 6：通过类装载器动态加载 a.apk 文件，读取源文件。

[0076] 所述步骤 6 中的类装载器采用 DexClassLoader。

[0077] 第一类加壳程序工作流程：

[0078] 1. 加密源程序 APK 文件为待加壳数据；

[0079] 2. 把待加壳数据写入解壳程序 Dex 文件尾部，并在文件尾部添加待加壳数据的大

小；

[0080] 3. 修改解壳程序 Dex 头中 checksum、signature 和 file_size 头信息；

[0081] Checksum：检验码字段是一个整数值，对 dex 数据采用 Adler32 算法生成，用来检查这个字段开始到文件结尾数据的完整性；对解壳程序 dex 数据重新用 Adler32 算法生成一个整数并替换原来的校验码字段即可；

[0082] Signature：签名字段，是一个 20 字节的数据，对 dex 数据采用 SHA 安全散列算法生成，用来检查 dex 数据的完整性；对解壳程序 dex 数据重新采用 SHA 安全散列算法生成签名数据并替换原来的签名字段即可；

[0083] file_size：dex 文件的总长度，对解壳程序 dex 文件重新计算长度并替换原来的长度字段；

[0084] 4. 修改源程序 AndroidMainfest.xml 文件并覆盖解壳程序 AndroidMainfest.xml 文件；

[0085] AndroidMainfest 文件可以通过在 application 节点 name 属性上配置程序的入口类，所以需要吧入口类配置为解壳程序指定的类，如果源文件已经配置了入口类需要把这个入口类用 <meta-data> 保存在此文件中，以便解壳程序在运行的时候还原此类。

[0086] 对应的解壳程序工作流程：

[0087] 1. 读取 Dex 文件尾部数据获取待加壳数据长度。

[0088] 2. 从 Dex 文件读取待加壳数据，解密待加壳数据。以文件形式保存解密数据到 a.apk 文件。

[0089] 3. 通过 DexClassLoader 动态加载 a.apk。

[0090] 第二类加壳方法待加壳数据位于加壳程序文件头

[0091] 该种方式相对比较复杂，合并后 Dex 文件结构如下：

[0092] 第二类加壳程序工作流程：

[0093] 1. 加密源程序 APK 文件为待加壳数据；

[0094] 2. 计算待加壳数据长度，并添加该长度到解壳程序 Dex 文件头末尾，并继续添加待加壳数据到文件头末尾（插入数据的位置为 0x70 处：这个偏移位置指向的是 dex 头文件的结尾，因为 dex 头文件的大小是固定的 70 字节）；

[0095] 3. 修改解壳程序 Dex 头中 checksum、signature、file_size、header_size、string_ids_off、type_ids_off、proto_ids_off、field_ids_off、method_ids_off、class_defs_off 和 data_off 相关项。分析 map_off 数据，修改相关的数据偏移量；

[0096] 4. 修改源程序 AndroidMainfest.xml 文件并覆盖解壳程序 AndroidMainfest.xml 文件。

[0097] 解壳程序工作流程：

[0098] 1. 从 0x70 处读取待加壳数据长度。

[0099] 2. 从 Dex 文件读取待加壳数据，解密待加壳数据。以文件形式保存解密数据到 a.apk。

[0100] 3. 通过 DexClassLoader 动态加载 a.apk。

[0101] 特殊说明：

[0102] 为了防止解壳程序被反编译，被破解，解壳程序可以考虑采用更加安全的

[0103] C 代码来实现，而不采用 Java，但是，两者的实现思路是相似的。

[0104] 以上所述仅为本发明的较佳实施例，并不用以限制本发明，凡在本发明的精神和原则之内，所作的任何修改、等同替换、改进等，均应包含在本发明的保护范围之内。

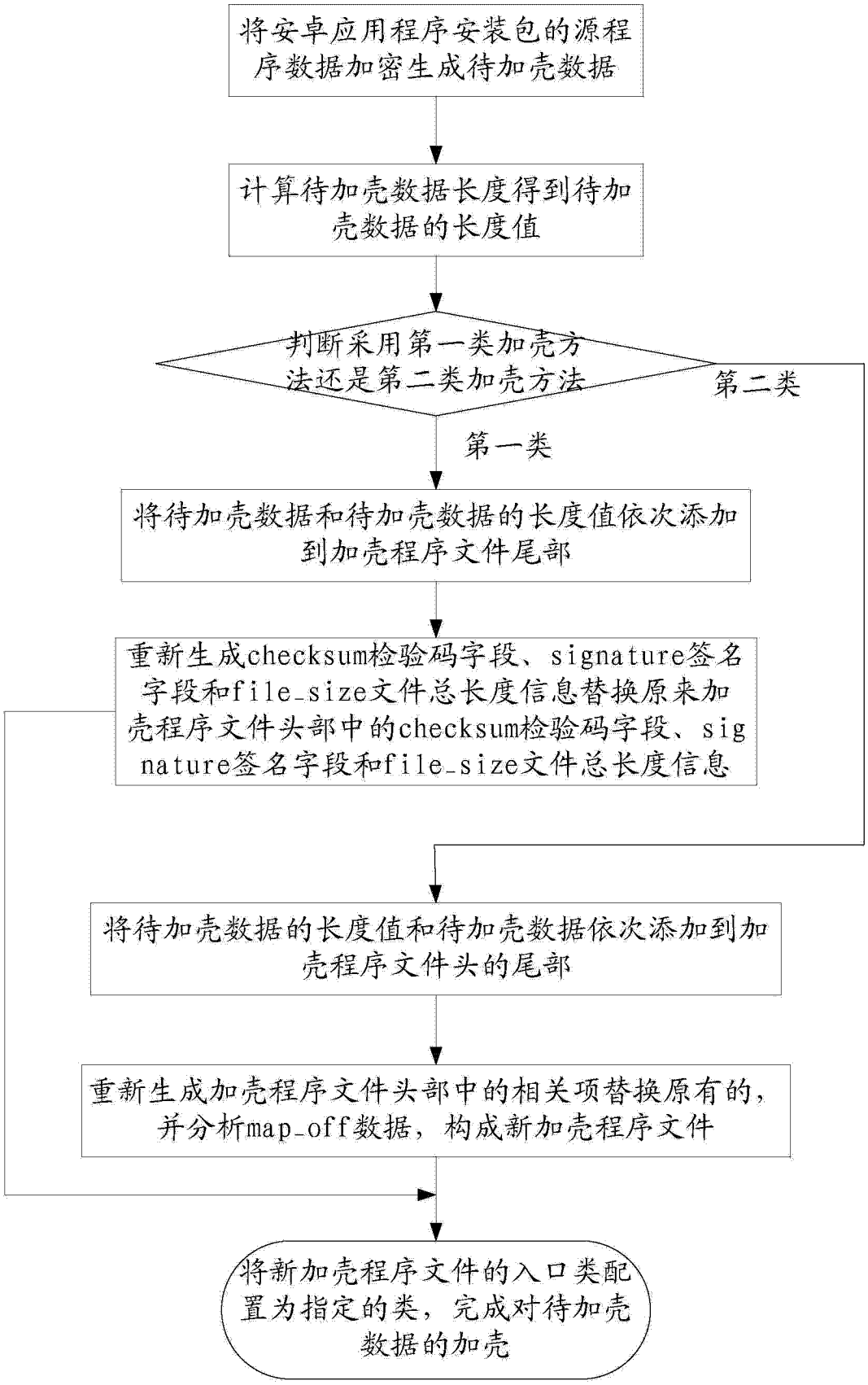

将安卓应用程序安装包的源程序数据加密生成待加壳数据
计算待加壳数据长度得到待加壳数据的长度值
判断采用第一类加壳方法还是第二类加壳方法
第二类
第一类
将待加壳数据和待加壳数据的长度值依次添加到加壳程序文件尾部
重新生成checksum检验码字段、signature签名字段和file_size文件总长度信息替换原来加壳程序文件头部中的checksum检验码字段、signature签名字段和file_size文件总长度信息
将待加壳数据的长度值和待加壳数据依次添加到加壳程序文件头的尾部
重新生成加壳程序文件头部中的相关项替换原有的，并分析map_off数据，构成新加壳程序文件
将新加壳程序文件的入口类配置为指定的类，完成对待加壳数据的加壳

图 1

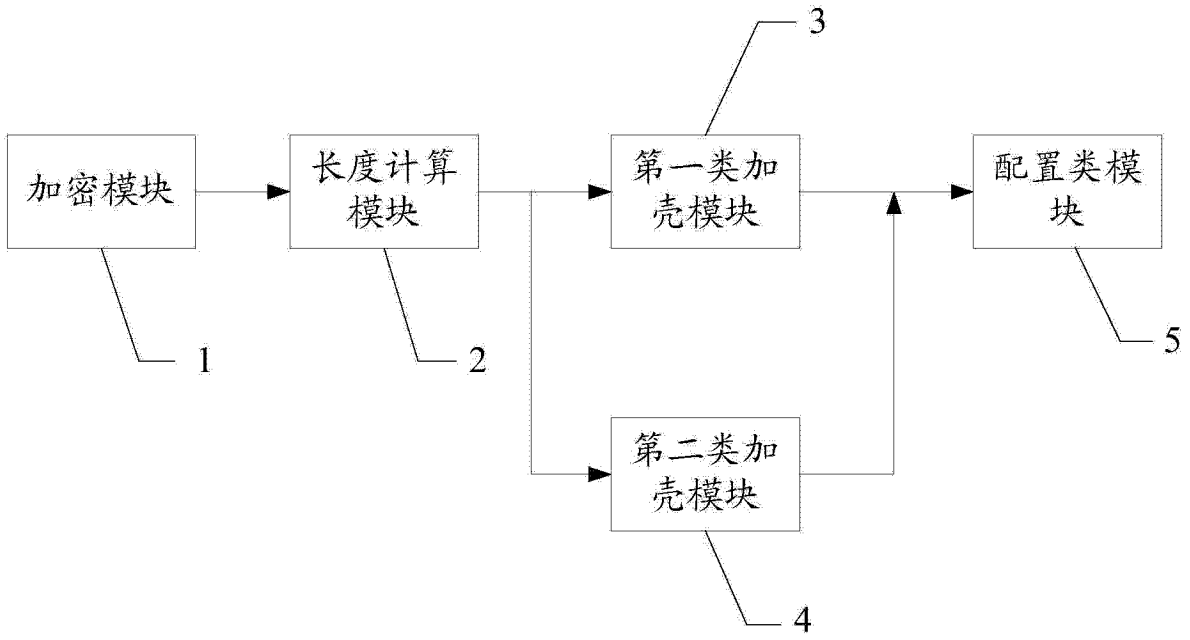


图 2

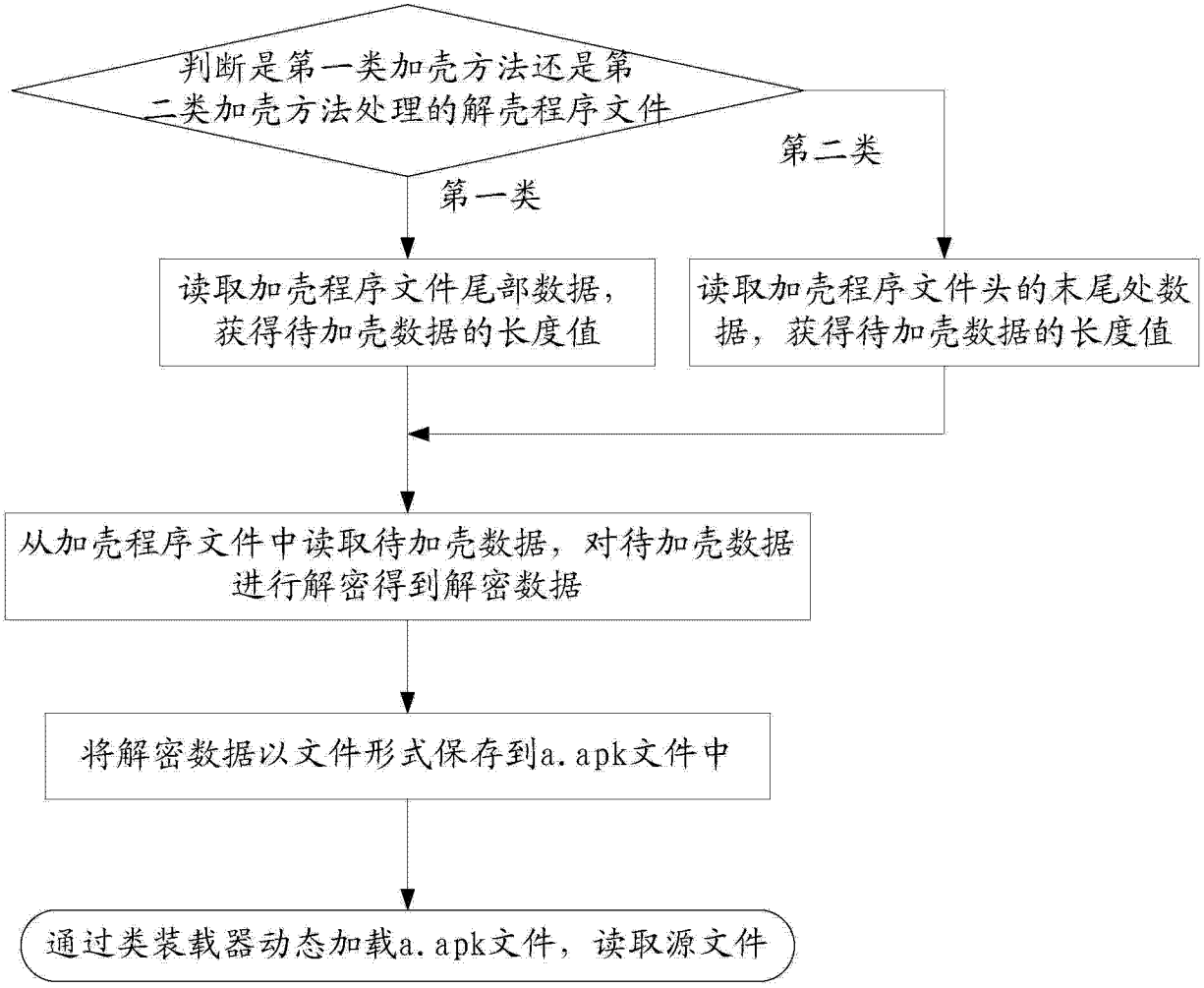

判断是第一类加壳方法还是第二类加壳方法处理的解壳程序文件
第一类
第二类
读取加壳程序文件尾部数据，获得待加壳数据的长度值
读取加壳程序文件头的末尾处数据，获得待加壳数据的长度值
从加壳程序文件中读取待加壳数据，对待加壳数据进行解密得到解密数据
将解密数据以文件形式保存到a.apk文件中
通过类装载器动态加载a.apk文件，读取源文件

图 3