(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 109543433 B
(45)授权公告日 2022.06.24

(21)申请号 201811424766.7

(22)申请日 2018.11.27

(65)同一申请的已公布的文献号
申请公布号 CN 109543433 A

(43)申请公布日 2019.03.29

(73)专利权人 杭州网易智企科技有限公司
地址 310000 浙江省杭州市滨江区长河街道网商路399号3幢408室

(72)发明人 朱星星

(74)专利代理机构 北京汉昊知识产权代理事务所(普通合伙) 11370
专利代理师 朱海波

(51)Int.Cl.
G06F 21/60(2013.01)
G06F 8/41(2018.01)

(56)对比文件
CN 105930695 A,2016.09.07
CN 107103211 A,2017.08.29
CN 103713896 A,2014.04.09
CN 108768649 A,2018.11.06

审查员 龚洁

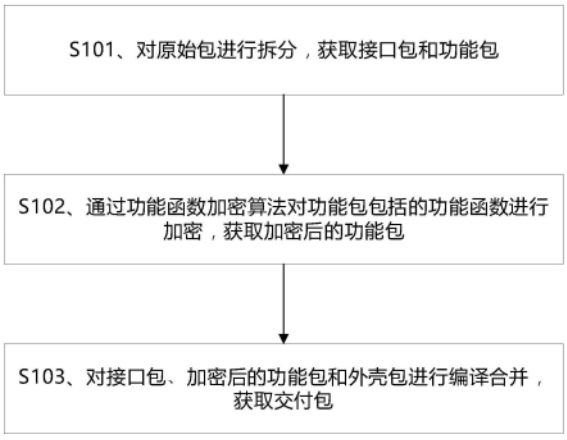
权利要求书3页 说明书12页 附图5页

(54)发明名称

软件开发工具包加密方法、装置、计算机和存储介质

(57)摘要

本发明的实施方式提供了一种软件开发工具包加密方法、装置、计算机和存储介质，涉及信息安全领域，软件开发工具包包括原始包和外壳包，原始包包括接口和功能函数，外壳包包括加密解密策略，该方法包括：对原始包进行拆分，获取接口包和功能包，接口包包括接口，功能包包括功能函数；通过功能函数加密算法对功能包包括的功能函数进行加密，获取加密后的功能包；对接口包、加密后的功能包和外壳包进行编译合并，获取交付包；其中，加密解密策略与功能函数加密算法相关。本发明能够在保证外部接口一致，不改变用户体验的基础上，极大提高软件开发工具包的保护强度。



CN 109543433 B

1.一种软件开发工具包加密方法,所述软件开发工具包包括混淆后的原始包和外壳包,所述混淆后的原始包包括接口包和功能包,所述外壳包包括加密解密策略、保护策略和安全策略,所述方法包括:

获取映射文件;

根据所述映射文件和所述混淆后的原始包,获取待保护的原始包,即通过所述映射文件和指定的原始类名映射出混淆后的原始包的类名,以确定待保护的原始包的类名;

对所述待保护的原始包进行拆分,获取所述接口包和所述功能包,所述接口包包括接口,所述功能包包括功能函数,其中所述功能函数包括第一组功能函数和第二组功能函数;

将所述第一组功能函数保留在所述功能包中,将所述第二组功能函数置于所述接口包中,其中,第一组功能函数,包括需要保密的核心功能函数,第二组功能函数,包括不需要进行特别保密的非核心功能函数;

通过所述外壳包中包括的加密解密策略相关的功能函数加密算法对所述功能包包括的第一组功能函数进行加密,获取加密后的功能包;

对所述加密后的功能包进行隐藏操作;

对所述外壳包进行动态编译以及动态加密,获取编译和加密后的外壳包;

对所述接口包、所述加密后的功能包以及所述编译和加密后的外壳包进行编译合并,获取交付包。

2.如权利要求1所述的加密方法,其特征在于,所述混淆后的原始包包括AAR包和JAR包中的至少一种。

3.一种软件开发工具包加密装置,所述软件开发工具包包括混淆后的原始包和外壳包,所述混淆后的原始包包括接口包和功能包,所述外壳包包括加密解密策略、保护策略和安全策略,所述装置包括:

处理单元,用于

获取映射文件;

根据所述映射文件和所述混淆后的原始包,获取待保护的原始包,即通过所述映射文件和指定的原始类名映射出混淆后的原始包的类名,以确定待保护的原始包的类名;

拆分单元,用于

对所述待保护的原始包进行拆分,获取接口包和功能包,所述接口包包括接口,所述功能包包括功能函数,其中所述功能函数包括第一组功能函数和第二组功能函数;

将所述第一组功能函数保留在所述功能包中,将所述第二组功能函数置于所述接口包中,其中,第一组功能函数,包括需要保密的核心功能函数,第二组功能函数,包括不需要进行特别保密的非核心功能函数;

加密单元,用于

通过所述外壳包中包括的加密解密策略相关的功能函数加密算法对所述功能包包括的第一组功能函数进行加密,获取加密后的功能包;

对所述加密后的功能包进行隐藏操作;

编译单元,用于

对所述外壳包进行动态编译以及动态加密,获取编译和加密后的外壳包;

对所述接口包、所述加密后的功能包以及所述编译和加密后的外壳包进行编译合并,

获取交付包。

4.如权利要求3所述的加密装置,其特征在于,所述混淆后的原始包包括AAR包和JAR包中的至少一种。

5.一种计算机,包括处理器和存储器,

所述处理器用于执行所述存储器中保存的软件开发工具包加密程序,其中所述软件开发工具包包括混淆后的原始包和外壳包,所述混淆后的原始包包括接口包和功能包,所述外壳包包括加密解密策略、保护策略和安全策略;

所述存储器中保存的所述程序用于执行:

获取映射文件;

根据所述映射文件和所述混淆后的原始包,获取待保护的原始包,即通过所述映射文件和指定的原始类名映射出混淆后的原始包的类名,以确定待保护的原始包的类名;

对所述待保护的原始包进行拆分,获取接口包和功能包,所述接口包包括接口,所述功能包包括功能函数,其中所述功能函数包括第一组功能函数和第二组功能函数;

将所述第一组功能函数保留在所述功能包中,将所述第二组功能函数置于所述接口包中,其中,第一组功能函数,包括需要保密的核心功能函数,第二组功能函数,包括不需要进行特别保密的非核心功能函数;

通过所述外壳包中包括的加密解密策略相关的功能函数加密算法对所述功能包包括的第一组功能函数进行加密,获取加密后的功能包,对所述加密后的功能包进行隐藏操作;

对所述外壳包进行动态编译以及动态加密,获取编译和加密后的外壳包;

对所述接口包、所述加密后的功能包以及所述编译和加密后的外壳包进行编译合并,获取交付包。

6.如权利要求5所述的计算机,其特征在于,所述混淆后的原始包包括AAR包和JAR包中的至少一种。

7.一种存储介质,用于存储软件开发工具包加密程序代码,其中所述软件开发工具包包括混淆后的原始包和外壳包,所述混淆后的原始包包括接口包和功能包,所述外壳包包括加密解密策略、保护策略和安全策略;

所述程序代码用于执行:

获取映射文件;

根据所述映射文件和所述混淆后的原始包,获取待保护的原始包,即通过所述映射文件和指定的原始类名映射出混淆后的原始包的类名,以确定待保护的原始包的类名;

对所述待保护的原始包进行拆分,获取接口包和功能包,所述接口包包括接口,所述功能包包括功能函数,其中所述功能函数包括第一组功能函数和第二组功能函数;

将所述第一组功能函数保留在所述功能包中,将所述第二组功能函数置于所述接口包中,其中,第一组功能函数,包括需要保密的核心功能函数,第二组功能函数,包括不需要进行特别保密的非核心功能函数;

通过所述外壳包中包括的加密解密策略相关的功能函数加密算法对所述功能包包括的第一组功能函数进行加密,获取加密后的功能包,对所述加密后的功能包进行隐藏操作;

对所述外壳包进行动态编译以及动态加密,获取编译和加密后的外壳包;

对所述接口包、所述加密后的功能包以及所述编译和加密后的外壳包进行编译合并,

获取交付包。

8.如权利要求7所述的存储介质，其特征在于，所述混淆后的原始包包括AAR包和JAR包中的至少一种。

软件开发工具包加密方法、装置、计算机和存储介质

技术领域

[0001] 本发明的实施方式涉及信息安全领域，更具体地，本发明的实施方式涉及软件开发工具包加密方法、装置、计算机和存储介质。

背景技术

[0002] 由于软件开发工具包(Software Development Kit,SDK)基于Java语言开发，很容易被反编译后得到源码，为了防止攻击者对SDK包进行源码分析，需要对SDK包进行保护。

[0003] 现有技术中，技术开发者一般会在打包的时候通过ProGuard工具来对程序代码进行混淆，再结合防篡改、防调试等手段得到被保护的SDK。在被保护的SDK被使用时，也就是动态运行时，再解密被保护的SDK。

[0004] 但是混淆后的SDK依然具有一定的可读性，一旦通过逆向分析等手段绕过防篡改、防调试等手段的保护，在加载SDK包的关键函数处仍然可以释放出混淆后的SDK包，然后通过阅读混淆后的SDK包，可以破解和逆向分析获得程序代码的主要逻辑，因此现有技术中对SDK的保护效果并不理想。

发明内容

[0005] 本发明实施例提供了一种软件开发工具包加密方法和装置。旨在解决现有技术中对软件开发工具包保护效果差的问题。为了对披露的实施例的一些方面有一个基本的理解，下面给出了简单的概括。该概括部分不是泛泛评述，也不是要确定关键/重要组成元素或描绘这些实施例的保护范围。

[0006] 根据本发明实施例的第一方面，提供了一种软件开发工具包加密方法，软件开发工具包包括原始包和外壳包，原始包包括接口和功能函数，外壳包包括加密解密策略，方法包括：

[0007] 对原始包进行拆分，获取接口包和功能包，接口包包括接口，功能包包括功能函数；

[0008] 通过功能函数加密算法对功能包包括的功能函数进行加密，获取加密后的功能包；

[0009] 对接口包、加密后的功能包和外壳包进行编译合并，获取交付包；

[0010] 其中，加密解密策略与功能函数加密算法相关。

[0011] 可选的，对接口包、加密后的功能包和外壳包进行编译合并，获取交付包，包括：

[0012] 对外壳包进行动态编译，获取编译后的外壳包；

[0013] 对接口包、加密后的功能包和编译后的外壳包进行编译合并，获取交付包。

[0014] 可选的，对外壳包进行动态编译，获取编译后的外壳包，还包括：

[0015] 对外壳包进行动态加密。

[0016] 可选的，外壳包还包括保护策略和安全策略。

[0017] 可选的，原始包包括的功能函数，包括第一组功能函数和第二组功能函数，功能包

包括第一组功能函数，接口包还包括第二组功能函数。

[0018] 可选的，原始包包括混淆后的原始包，方法还包括：

[0019] 获取映射文件；

[0020] 根据映射文件和混淆后的原始包，获取待保护的原始包；

[0021] 对原始包进行拆分，获取接口包和功能包，包括：

[0022] 对待保护的原始包进行拆分，获取接口包和功能包。

[0023] 可选的，原始包包括AAR包和JAR包中的至少一种。

[0024] 根据本发明实施例的第二方面，提供了一种软件开发工具包加密装置，软件开发工具包包括原始包和外壳包，原始包包括接口和功能函数，外壳包包括加密解密策略，装置包括：

[0025] 拆分单元，用于对原始包进行拆分，获取接口包和功能包，接口包包括接口，功能包包括功能函数；

[0026] 加密单元，用于通过功能函数加密算法对功能包包括的功能函数进行加密，获取加密后的功能包；

[0027] 编译单元，用于对接口包、加密后的功能包和外壳包进行编译合并，获取交付包；

[0028] 其中，加密解密策略与功能函数加密算法相关。

[0029] 可选的，编译单元还用于：

[0030] 对外壳包进行动态编译，获取编译后的外壳包；

[0031] 对接口包、加密后的功能包和编译后的外壳包进行编译合并，获取交付包。

[0032] 可选的，编译单元还用于：

[0033] 对外壳包进行动态加密。

[0034] 可选的，外壳包还包括保护策略和安全策略。

[0035] 可选的，原始包包括的功能函数，包括第一组功能函数和第二组功能函数，拆分单元还用于：

[0036] 对原始包进行拆分，获取接口包和功能包，功能包包括第一组功能函数，接口包还包括第二组功能函数。

[0037] 可选的，原始包包括混淆后的原始包，加密装置还包括：

[0038] 处理单元，用于获取映射文件；

[0039] 根据映射文件和混淆后的原始包，获取待保护的原始包；

[0040] 拆分单元，还用于对待保护的原始包进行拆分，获取接口包和功能包。

[0041] 可选的，原始包包括AAR包和JAR包中的至少一种。

[0042] 根据本发明实施例的第三方面，提供了一种计算机，包括处理器和存储器，处理器用于执行存储器中保存的程序，存储器中保存的程序用于执行：

[0043] 对原始包进行拆分，获取接口包和功能包，接口包包括接口，功能包包括功能函数；

[0044] 通过功能函数加密算法对功能包包括的功能函数进行加密，获取加密后的功能包；

[0045] 对接口包、加密后的功能包和外壳包进行编译合并，获取交付包；

[0046] 其中，软件开发工具包包括原始包和外壳包，原始包包括接口和功能函数，外壳包

包括加密解密策略，加密解密策略与功能函数加密算法相关。

[0047] 可选的，程序还用于执行：

[0048] 对外壳包进行动态编译，获取编译后的外壳包；

[0049] 对接口包、加密后的功能包和编译后的外壳包进行编译合并，获取交付包。

[0050] 可选的，程序还用于执行：

[0051] 对外壳包进行动态编译的过程中，对外壳包进行动态加密。

[0052] 可选的，外壳包还包括保护策略和安全策略。

[0053] 可选的，原始包包括的功能函数，包括第一组功能函数和第二组功能函数，功能包包括第一组功能函数，接口包还包括第二组功能函数。

[0054] 可选的，原始包包括混淆后的原始包，程序还用于执行：

[0055] 获取映射文件；

[0056] 根据映射文件和混淆后的原始包，获取待保护的原始包；

[0057] 对原始包进行拆分，获取接口包和功能包，包括：

[0058] 对待保护的原始包进行拆分，获取接口包和功能包。

[0059] 可选的，原始包包括AAR包和JAR包中的至少一种。

[0060] 根据本发明实施例的第四方面，提供了一种存储介质，用于存储程序代码，程序代码用于执行：

[0061] 对原始包进行拆分，获取接口包和功能包，接口包包括接口，功能包包括功能函数；

[0062] 通过功能函数加密算法对功能包包括的功能函数进行加密，获取加密后的功能包；

[0063] 对接口包、加密后的功能包和外壳包进行编译合并，获取交付包；

[0064] 其中，软件开发工具包包括原始包和外壳包，原始包包括接口和功能函数，外壳包包括加密解密策略，加密解密策略与功能函数加密算法相关。

[0065] 可选的，程序代码还用于执行：

[0066] 对外壳包进行动态编译，获取编译后的外壳包；

[0067] 对接口包、加密后的功能包和编译后的外壳包进行编译合并，获取交付包。

[0068] 可选的，程序代码还用于执行：

[0069] 对外壳包进行动态编译的过程中，对外壳包进行动态加密。

[0070] 可选的，外壳包还包括保护策略和安全策略。

[0071] 可选的，原始包包括的功能函数，包括第一组功能函数和第二组功能函数，功能包包括第一组功能函数，接口包还包括第二组功能函数。

[0072] 可选的，原始包包括混淆后的原始包，程序还用于执行：

[0073] 获取映射文件；

[0074] 根据映射文件和混淆后的原始包，获取待保护的原始包；

[0075] 对原始包进行拆分，获取接口包和功能包，包括：

[0076] 对待保护的原始包进行拆分，获取接口包和功能包。

[0077] 可选的，原始包包括AAR包和JAR包中的至少一种。

[0078] 本发明实施例公开的技术方案，可以对SDK中包括的原始包进行拆分，拆分后的接

口包能够保证接口被正确引用，对功能函数进行加密，能够有效降低功能函数被破解并获取的可能性，极大增强了对SDK的保护效果，且适用场景广泛。

附图说明

[0079] 通过参考附图阅读下文的详细描述，本发明示例性实施方式的上述以及其他目的、特征和优点将变得易于理解。在附图中，以示例性而非限制性的方式示出了本发明的若干实施方式，其中：

[0080] 图1示意性地示出了一种软件开发工具包加密方法的流程图；

[0081] 图2示意性地示出了另一种软件开发工具包加密方法的流程图；

[0082] 图3示意性地示出了另一种软件开发工具包加密方法的流程图；

[0083] 图4示意性地示出了一种软件开发工具包加密装置的示意图；

[0084] 图5示意性地示出了另一种软件开发工具包加密装置的示意图；

[0085] 图6示意性地示出了一种计算机的示意图；

[0086] 图7示意性地示出了一种存储介质的示意图。

[0087] 在附图中，相同或对应的标号表示相同或对应的部分。

具体实施方式

[0088] 以下描述和附图充分地示出本发明的具体实施方案，以使本领域的技术人员能够实践它们。实施例仅代表本发明公开的技术方案的可能的变化形式，本发明请求保护的内容并不仅限于此。除非明确要求，否则单独的部件和功能是可选的，并且操作的顺序可以变化。一些实施方案的部分和特征可以被包括在或替换其他实施方案的部分和特征。本发明的实施方案的范围包括权利要求书的整个范围，以及权利要求书的所有可获得的等同物。在本文中，各实施方案可以被单独地或总地用术语"发明"来表示，这仅仅是为了方便，并且如果事实上公开了超过一个的发明，不是要自动地限制该应用的范围为任何单个发明或发明构思。本文中，诸如第一和第二等之类的关系术语仅仅用于将一个实体或者操作与另一个实体或操作区分开来，而不要求或者暗示这些实体或操作之间存在任何实际的关系或者顺序。而且，术语"包括"、"包含"或者其任何其他变体意在涵盖非排他性的包含，从而使得包括一系列要素的过程、方法或者设备不仅包括那些要素，而且还包括没有明确列出的其他要素。本文中各个实施例采用递进的方式描述，每个实施例重点说明的都是与其他实施例的不同之处，各个实施例之间相同相似部分互相参见即可。对于实施例公开的结构、产品等而言，由于其与实施例公开的部分相对应，所以描述的比较简单，相关之处参见方法部分说明即可。

[0089] 本发明实施例公开了一种软件开发工具包加密方法，软件开发工具包包括原始包和外壳包，原始包包括接口和功能函数，外壳包包括加密解密策略，如图1所示，加密方法包括：

[0090] S101、对原始包进行拆分，获取接口包和功能包，接口包包括接口，功能包包括功能函数；

[0091] S102、通过功能函数加密算法对功能包包括的功能函数进行加密，获取加密后的功能包；

[0092] S103、对接口包、加密后的功能包和外壳包进行编译合并，获取交付包；

[0093] 其中，加密解密策略与功能函数加密算法相关。

[0094] 本发明实施例公开的技术方案，可以对SDK中包括的原始包进行拆分，拆分后的接口包能够保证接口被正确引用，对功能函数进行加密，能够有效降低功能函数被破解并获取的可能性，极大增强了对SDK的保护效果。

[0095] SDK可以包括AAR(Android Archive)包和JAR(Java Archive)包中的至少一种，即SDK可以包括至少一个AAR包，或至少一个JAR包，或至少一个AAR包和至少一个JAR包的组合。进一步的，AAR包可以包括JAR包。

[0096] 在S101中，原始包可以包括AAR包和JAR包的至少一种，即，原始包可以为SDK包括的AAR包和JAR包。

[0097] 一般的，SDK一般是一些被软件工程师用于为特定的软件包、软件框架、硬件平台或操作系统等创建应用软件的开发工具的集合。示例性的，SDK可以包括AAR包或JAR包。

[0098] AAR包是一个安卓库项目的二进制归档文件，AAR包可以包括JAR包。

[0099] JAR包是与平台无关的文件格式，允许将多个文件组合成为一个压缩文件。

[0100] 示例性的，当AAR包包括JAR包时，可选的，还可以将JAR包作为原始包，即在S101之前，还可以包括：根据AAR包，获取JAR包。之后即可对JAR包进行上述提到的拆分等操作。

[0101] 示例性的，对原始包进行拆分，可以通过AsmTools工具进行操作，拆分出接口包和功能包，接口包包括接口，功能包包括功能函数。本领域技术人员应知，对原始包进行拆分，是对接口和功能函数的分离，并非对文件进行分段或分片处理。

[0102] 接口包包括的接口只用于声明接口形式，保证接入者在开发阶段对接口的正确引用，并不具有实际的逻辑运行功能，不能用于分析和破解。经过拆分后获取的接口包，不包含全部功能函数的代码，即使通过反编译工具、逆向工具等，也无法获取完整的功能函数的代码，进一步的，经过拆分后获取的功能包，是残缺和离散包，无法直接使用。因此，采用对原始包进行拆分，获取残缺和离散的功能包的方式，能够为功能函数提供足够的保护强度，进而达到防止SDK被破解或被分析的目的。

[0103] 在S102中对功能函数进行加密，在S101中将功能函数拆分至功能包后，加密操作可以进一步提升保护强度。特别的，对加密后的功能包还可以进行隐藏操作，示例性的，未加密的功能包可以为JAR文件或DEX文件，加密后的功能包可以为PNG文件或其他格式和名称的文件，即能够起到隐藏功能包的目的，本发明实施例对加密后的功能包的格式和名称不做限定。

[0104] 在S103中，对接口包、加密后的功能包和外壳包进行编译合并，获取的交付包可以用于交付用户使用。特别的，接口包虽然不包括完整的功能函数，但接口包包括的对外声明的接口与未进行拆分时的原始包包括的接口一致，能够保证用户使用体验的一致性。

[0105] 功能函数加密算法用于对功能函数进行加密，外壳包包括的加密解密策略与功能函数加密算法相关，方便用户在具体使用过程中，通过加密解密策略完成对功能函数的解密，保证功能函数的准确运行。

[0106] 示例性的，加密解密策略可以包括加密因子，加密因子与功能函数加密算法相关。

[0107] 可选的，当外壳包使用防篡改保护、防调试和防DUMP保护或混淆技术等进一步增强安全性的技术方案时，外壳包还可以包括保护策略和安全策略，具体的，保护策略和安全

策略能够保证用户正常使用经过拆分处理后的SDK。进一步的，外壳包还可以包括接口与功能函数的映射说明等，用于保证功能函数的准确运行。

[0108] 可选的，如图2所示，S103还可以包括：

[0109] S1031、对外壳包进行动态编译，获取编译后的外壳包；

[0110] S1032、对接口包、加密后的功能包和编译后的外壳包进行编译合并，获取交付包。

[0111] 在S1031中，对外壳包进行动态编译的过程中，还可以对外壳包进行动态加密，即，每一次对外壳包进行编译时可以使用不同的加密算法，亦即，对不同外壳包进行编译时，以及对同一外壳包进行多次编译时使用的加密算法均不相同，可以进一步增强保护强度。

[0112] 应知，当对外壳包进行编译过程中同时使用加密算法对外壳包进行加密，则可以获取经过动态加密的编译后的外壳包。

[0113] 在S1032中，将编译后的外壳包、接口包和加密后的功能包进行合并编译，最终生成一个提供给用户使用的交付包。

[0114] 由于动态编译后的外壳包，每次编译生成的结果并不相同，因此每次最终生成的SDK也不相同，增加了破解SDK的难度，保证了SDK的安全性。

[0115] 在一些特别的情况下，如原始包中包括了较多的功能函数，但并不需要对全部功能函数进行加密时，可以仅对部分功能函数进行加密。

[0116] 示例性的，功能函数可以包括第一组功能函数和第二组功能函数，可选的，S101可以包括：

[0117] S1011、对原始包进行拆分，获取接口包和功能包，接口包包括接口和第二组功能函数，功能包包括第一组功能函数。

[0118] 其中，第一组功能函数，可以包括核心功能函数等需要保密的功能函数，第二组功能函数，可以包括非核心功能函数等不需要进行特别保密的功能函数。此种情况下，在S102中仅对功能包中包括的第一组功能函数进行加密，而对接口包包括的接口和第二组功能函数不进行加密。

[0119] 虽然第二组功能函数未进行加密，但由于其并不涉及核心功能函数，因此也不会存在安全性问题。仅对核心功能函数进行加密，可以在保证安全性的同时，节约系统的计算资源，减少加密和解密操作所需的时间，提升用户体验。

[0120] 本领域技术人员在具体实施过程中，可以结合S1011，根据具体实施条件对功能函数进行分组，并仅对分组后的部分功能函数进行加密，分组的标准可以包括是否为核心功能函数，也可以包括其他标准，示例性的，本着解决计算量的原则，可以只将原创的功能函数拆分至功能包，并进行加密等，本发明对需要加密的功能函数的数量和选择标准并不限定。

[0121] 此外，在具体实施过程中，原始包可能包括混淆后的原始包，示例性的，技术人员在技术开发过程中，对原始包进行过代码混淆，如ProGuard混淆等，可选的，如图3所示，在S101之前，还可以包括：

[0122] S104、获取映射文件；

[0123] S105、根据映射文件和混淆后的原始包，获取待保护的原始包。

[0124] S101可以包括：

[0125] S1012、对待保护的原始包进行拆分，获取接口包和功能包。

[0126] 经过混淆的原始包，程序代码中包括的类名等可能发生改变，可以通过映射文件和指定的原始类名映射出混淆后的类名，以确定待保护的原始包的类名。基于此，在本发明实施例中，当原始包为经过混淆的原始包时，可以先根据映射文件获取待保护的原始包，并对获取到的待保护的原始包进行拆分，并进一步进行安全加密等处理。

[0127] 本发明实施例公开的技术方案，能够在保证外部接口一致，不影响用户体验的基础上，极大提高SDK的保护强度，且能够同时适用于JAR包和AAR包，适用场景广泛。

[0128] 更进一步的，本领域技术人员在本发明实施例公开的技术方案的基础上，还可以结合ProGuard混淆等操作，更加提升SDK的安全性，本发明实施例对此并不限定。

[0129] 本发明实施例还公开了一种软件开发工具包加密装置40，如图4所示，软件开发工具包包括原始包和外壳包，原始包包括接口和功能函数，外壳包包括加密解密策略，装置40包括：

[0130] 拆分单元401，用于对原始包进行拆分，获取接口包和功能包，接口包包括接口，功能包包括功能函数；

[0131] 加密单元402，用于通过功能函数加密算法对功能包包括的功能函数进行加密，获取加密后的功能包；

[0132] 编译单元403，用于对接口包、加密后的功能包和外壳包进行编译合并，获取交付包；

[0133] 其中，加密解密策略与功能函数加密算法相关。

[0134] 特别的，拆分单元401对原始包进行拆分，是对接口和功能函数的分离，并非对文件进行分段或分片处理。

[0135] 功能函数加密算法用于对功能函数进行加密，外壳包包括的加密解密策略与功能函数加密算法相关，方便用户在具体使用过程中，通过加密解密策略完成对功能函数的解密，保证功能函数的准确运行。

[0136] 本发明实施例公开的技术方案，可以通过拆分单元对SDK中包括的原始包进行拆分，拆分后的接口包能够保证接口被正确引用，加密单元对功能函数进行加密，能够有效降低功能函数被破解并获取的可能性，因此，能够在对接口的使用者透明的情况下，增加功能函数的保密强度，极大增强了对SDK的保护效果。

[0137] 可选的，编译单元403还可以用于：

[0138] 对外壳包进行动态编译，获取编译后的外壳包；

[0139] 对接口包、加密后的功能包和编译后的外壳包进行编译合并，获取交付包。

[0140] 可选的，编译单元403还可以用于：

[0141] 对外壳包进行动态加密。

[0142] 编译单元403在对外壳包进行动态编译的过程中，还可以对外壳包进行动态加密，即，每一次对外壳包进行编译时可以使用不同的加密算法，亦即，对不同外壳包进行编译时，以及对同一外壳包进行多次编译时使用的加密算法均不相同，可以进一步增强保护强度。

[0143] 当对外壳包进行编译过程中同时使用加密算法对外壳包进行加密，则可以获取经过动态加密的编译后的外壳包。

[0144] 接口包包括的接口只用于声明接口形式，保证接入者在开发阶段对接口的正确引

用，并不具有实际的逻辑运行功能，不能用于分析和破解。经过拆分后获取的接口包，不包含全部功能函数的代码，即使通过反编译工具、逆向工具等，也无法获取完整的功能函数的代码，进一步的，经过拆分后获取的功能包，是残缺和离散包，无法直接使用。因此，采用对原始包进行拆分，获取残缺和离散的功能包的方式，能够为功能函数提供足够的保护强度，进而达到防止SDK被破解或被分析的目的。

[0145] 可选的，外壳包还包括保护策略和安全策略。

[0146] 保护策略和安全策略能够保证用户正常使用经过拆分处理后的SDK。进一步的，外壳包还可以包括接口与功能函数的映射说明等，用于保证功能函数的准确运行。

[0147] 可选的，原始包包括的功能函数，包括第一组功能函数和第二组功能函数，拆分单元401还可以用于：

[0148] 对原始包进行拆分，获取接口包和功能包，功能包包括第一组功能函数，接口包还包括第二组功能函数。

[0149] 其中，第一组功能函数，可以包括核心功能函数等需要保密的功能函数，第二组功能函数，可以包括非核心功能函数等不需要进行特别保密的功能函数。此种情况下，拆分单元401可以将第一组功能函数拆分至功能包，将第二组功能函数拆分至接口包，加密单元402可以仅对功能包中包括的第一组功能函数进行加密，而对接口包包括的接口和第二组功能函数不进行加密。

[0150] 虽然第二组功能函数未进行加密，但由于其并不涉及核心功能函数，因此也不会存在安全性问题。仅对核心功能函数进行加密，可以在保证安全性的同时，节约系统的计算资源，减少加密和解密操作所需的时间，提升用户体验。

[0151] 可选的，加密装置40还可以包括处理单元404，如图5所示，其中：

[0152] 处理单元404，用于获取映射文件；根据映射文件和混淆后的原始包，获取待保护的原始包。

[0153] 拆分单元401，还用于对待保护的原始包进行拆分，获取接口包和功能包。

[0154] 经过混淆的原始包，程序代码中包括的类名等可能发生改变，可以通过映射文件和指定的原始类名映射出混淆后的类名，以确定待保护的原始包的类名。基于此，在本发明实施例中，当需要对经过混淆的原始包进行加密保护时，可以先根据映射文件获取待保护的原始包，并对获取到的待保护的原始包进行拆分，并进一步进行安全加密等处理。

[0155] 可选的，原始包包括AAR包和JAR包中的至少一种。

[0156] 原始包可以包括AAR包和JAR包中的至少一种，即原始包可以包括至少一个AAR包，或至少一个JAR包，或至少一个AAR包和至少一个JAR包的组合，进一步的，AAR包可以包括JAR包。

[0157] 本发明实施例公开的技术方案，能够在保证外部接口一致，不影响用户体验的基础上，极大提高SDK的保护强度，且能够同时适用于JAR包和AAR包，适用场景广泛。

[0158] 本发明实施例还公开了一种计算机60，如图6所示，包括处理器601和存储器602，处理器601用于执行存储器602中保存的程序，存储器602中保存的程序用于执行：

[0159] 对原始包进行拆分，获取接口包和功能包，接口包包括接口，功能包包括功能函数；

[0160] 通过功能函数加密算法对功能包包括的功能函数进行加密，获取加密后的功能

包；

[0161] 对接口包、加密后的功能包和外壳包进行编译合并，获取交付包；

[0162] 其中，软件开发工具包包括原始包和外壳包，原始包包括接口和功能函数，外壳包包括加密解密策略，加密解密策略与功能函数加密算法相关。

[0163] 特别的，对原始包进行拆分，是对接口和功能函数的分离，并非对文件进行分段或分片处理。

[0164] 功能函数加密算法用于对功能函数进行加密，外壳包包括的加密解密策略与功能函数加密算法相关，方便用户在具体使用过程中，通过加密解密策略完成对功能函数的解密，保证功能函数的准确运行。

[0165] 本发明实施例公开的技术方案，可以通过对SDK中包括的原始包进行拆分，拆分后的接口包能够保证接口被正确引用，对功能函数进行加密，能够有效降低功能函数被破解并获取的可能性，因此，能够在对接口的使用者透明的情况下，增加功能函数的保密强度，极大增强了对SDK的保护效果。

[0166] 可选的，程序还可以用于执行：

[0167] 对外壳包进行动态编译，获取编译后的外壳包；

[0168] 对接口包、加密后的功能包和编译后的外壳包进行编译合并，获取交付包。

[0169] 可选的，程序还可以用于执行：

[0170] 对外壳包进行动态编译的过程中，对外壳包进行动态加密。

[0171] 在对外壳包进行动态编译的过程中，还可以对外壳包进行动态加密，即，每一次对外壳包进行编译时可以使用不同的加密算法，亦即，对不同外壳包进行编译时，以及对同一外壳包进行多次编译时使用的加密算法均不相同，可以进一步增强保护强度。

[0172] 当对外壳包进行编译过程中同时使用加密算法对外壳包进行加密，则可以获取经过动态加密的编译后的外壳包。

[0173] 接口包包括的接口只用于声明接口形式，保证接入者在开发阶段对接口的正确引用，并不具有实际的逻辑运行功能，不能用于分析和破解。经过拆分后获取的接口包，不包含全部功能函数的代码，即使通过反编译工具、逆向工具等，也无法获取完整的功能函数的代码，进一步的，经过拆分后获取的功能包，是残缺和离散包，无法直接使用。因此，采用对原始包进行拆分，获取残缺和离散的功能包的方式，能够为功能函数提供足够的保护强度，进而达到防止SDK被破解或被分析的目的。

[0174] 可选的，外壳包还可以包括保护策略和安全策略。

[0175] 保护策略和安全策略能够保证用户正常使用经过拆分处理后的SDK。进一步的，外壳包还可以包括接口与功能函数的映射说明等，用于保证功能函数的准确运行。

[0176] 可选的，原始包包括的功能函数，可以包括第一组功能函数和第二组功能函数，功能包包括第一组功能函数，接口包还包括第二组功能函数。

[0177] 其中，第一组功能函数，可以包括核心功能函数等需要保密的功能函数，第二组功能函数，可以包括非核心功能函数等不需要进行特别保密的功能函数。此种情况下，可以将第一组功能函数拆分至功能包，将第二组功能函数拆分至接口包，仅对功能包中包括的第一组功能函数进行加密，而对接口包包括的接口和第二组功能函数不进行加密。

[0178] 虽然第二组功能函数未进行加密，但由于其并不涉及核心功能函数，因此也不会

存在安全性问题。仅对核心功能函数进行加密，可以在保证安全性的同时，节约系统的计算资源，减少加密和解密操作所需的时间，提升用户体验。

[0179] 可选的，原始包可以包括混淆后的原始包，程序还可以用于执行：

[0180] 获取映射文件；

[0181] 根据映射文件和混淆后的原始包，获取待保护的原始包；

[0182] 对原始包进行拆分，获取接口包和功能包，包括：

[0183] 对待保护的原始包进行拆分，获取接口包和功能包。

[0184] 经过混淆的原始包，程序代码中包括的类名等可能发生改变，可以通过映射文件和指定的原始类名映射出混淆后的类名，以确定待保护的原始包的类名。基于此，在本发明实施例中，当需要对经过混淆的原始包进行加密保护时，可以先根据映射文件获取待保护的原始包，并对获取到的待保护的原始包进行拆分，并进一步进行安全加密等处理。

[0185] 可选的，原始包可以包括AAR包和JAR包中的至少一种。

[0186] 原始包可以包括AAR包和JAR包中的至少一种，即原始包可以包括至少一个AAR包，或至少一个JAR包，或至少一个AAR包和至少一个JAR包的组合，进一步的，AAR包可以包括JAR包。

[0187] 本发明实施例公开的技术方案，能够在保证外部接口一致，不影响用户体验的基础上，极大提高SDK的保护强度，且能够同时适用于JAR包和AAR包，适用场景广泛。

[0188] 本领域技术人员应知，本发明实施例公开的计算机60，还可以包括内存储器、总线、输入输出装置和显示装置等其他相关组件，计算机60可以用于执行如图1至3所示的任一种加密方法，此处不再赘述。

[0189] 本发明实施例还公开来了一种存储介质70，如图7所示，用于存储程序代码，程序代码用于执行：

[0190] 对原始包进行拆分，获取接口包和功能包，接口包包括接口，功能包包括功能函数；

[0191] 通过功能函数加密算法对功能包包括的功能函数进行加密，获取加密后的功能包；

[0192] 对接口包、加密后的功能包和外壳包进行编译合并，获取交付包；

[0193] 其中，软件开发工具包包括原始包和外壳包，原始包包括接口和功能函数，外壳包包括加密解密策略，加密解密策略与功能函数加密算法相关。

[0194] 特别的，对原始包进行拆分，是对接口和功能函数的分离，并非对文件进行分段或分片处理。

[0195] 功能函数加密算法用于对功能函数进行加密，外壳包包括的加密解密策略与功能函数加密算法相关，方便用户在具体使用过程中，通过加密解密策略完成对功能函数的解密，保证功能函数的准确运行。

[0196] 本发明实施例公开的技术方案，可以通过对SDK中包括的原始包进行拆分，拆分后的接口包能够保证接口被正确引用，对功能函数进行加密，能够有效降低功能函数被破解并获取的可能性，因此，能够在对接口的使用者透明的情况下，增加功能函数的保密强度，极大增强了对SDK的保护效果。

[0197] 可选的，程序代码还可以用于执行：

[0198] 对外壳包进行动态编译，获取编译后的外壳包；

[0199] 对接口包、加密后的功能包和编译后的外壳包进行编译合并，获取交付包。

[0200] 可选的，程序代码还可以用于执行：

[0201] 对外壳包进行动态编译的过程中，对外壳包进行动态加密。

[0202] 在对外壳包进行动态编译的过程中，还可以对外壳包进行动态加密，即，每一次对外壳包进行编译时可以使用不同的加密算法，亦即，对不同外壳包进行编译时，以及对同一外壳包进行多次编译时使用的加密算法均不相同，可以进一步增强保护强度。

[0203] 当对外壳包进行编译过程中同时使用加密算法对外壳包进行加密，则可以获取经过动态加密的编译后的外壳包。

[0204] 接口包包括的接口只用于声明接口形式，保证接入者在开发阶段对接口的正确引用，并不具有实际的逻辑运行功能，不能用于分析和破解。经过拆分后获取的接口包，不包含全部功能函数的代码，即使通过反编译工具、逆向工具等，也无法获取完整的功能函数的代码，进一步的，经过拆分后获取的功能包，是残缺和离散包，无法直接使用。因此，采用对原始包进行拆分，获取残缺和离散的功能包的方式，能够为功能函数提供足够的保护强度，进而达到防止SDK被破解或被分析的目的。

[0205] 可选的，外壳包还可以包括保护策略和安全策略。

[0206] 保护策略和安全策略能够保证用户正常使用经过拆分处理后的SDK。进一步的，外壳包还可以包括接口与功能函数的映射说明等，用于保证功能函数的准确运行。

[0207] 可选的，原始包包括的功能函数，可以包括第一组功能函数和第二组功能函数，功能包包括第一组功能函数，接口包还包括第二组功能函数。

[0208] 其中，第一组功能函数，可以包括核心功能函数等需要保密的功能函数，第二组功能函数，可以包括非核心功能函数等不需要进行特别保密的功能函数。此种情况下，可以将第一组功能函数拆分至功能包，将第二组功能函数拆分至接口包，仅对功能包中包括的第一组功能函数进行加密，而对接口包包括的接口和第二组功能函数不进行加密。

[0209] 虽然第二组功能函数未进行加密，但由于其并不涉及核心功能函数，因此也不会存在安全性问题。仅对核心功能函数进行加密，可以在保证安全性的同时，节约系统的计算资源，减少加密和解密操作所需的时间，提升用户体验。

[0210] 可选的，原始包可以包括混淆后的原始包，程序代码还可以用于执行：

[0211] 获取映射文件；

[0212] 根据映射文件和混淆后的原始包，获取待保护的原始包；

[0213] 对原始包进行拆分，获取接口包和功能包，包括：

[0214] 对待保护的原始包进行拆分，获取接口包和功能包。

[0215] 经过混淆的原始包，程序代码代码中包括的类名等可能发生改变，可以通过映射文件和指定的原始类名映射出混淆后的类名，以确定待保护的原始包的类名。基于此，在本发明实施例中，当需要对经过混淆的原始包进行加密保护时，可以先根据映射文件获取待保护的原始包，并对获取到的待保护的原始包进行拆分，并进一步进行安全加密等处理。

[0216] 可选的，原始包可以包括AAR包和JAR包中的至少一种。

[0217] 原始包可以包括AAR包和JAR包中的至少一种，即原始包可以包括至少一个AAR包，或至少一个JAR包，或至少一个AAR包和至少一个JAR包的组合，进一步的，AAR包可以包括

JAR包。

[0218] 本发明实施例公开的技术方案，能够在保证外部接口一致，不影响用户体验的基础上，极大提高SDK的保护强度，且能够同时适用于JAR包和AAR包，适用场景广泛。

[0219] 存储介质70可以包括磁带、软盘、光盘、硬盘和闪存盘等各类能够用于数据存储的介质，存储介质70所存储的程序可以用于执行如图1至3所示的任一种加密方法，本发明实施例对存储介质70的具体形态并不限定。

[0220] 虽然已经参考若干具体实施方式描述了本发明的精神和原理，但是应该理解，本发明并不限于所公开的具体实施方式，对各方面的划分也不意味着这些方面中的特征不能组合以进行受益，这种划分仅是为了表述的方便。本发明旨在涵盖所附权利要求的精神和范围内所包括的各种修改和等同布置。

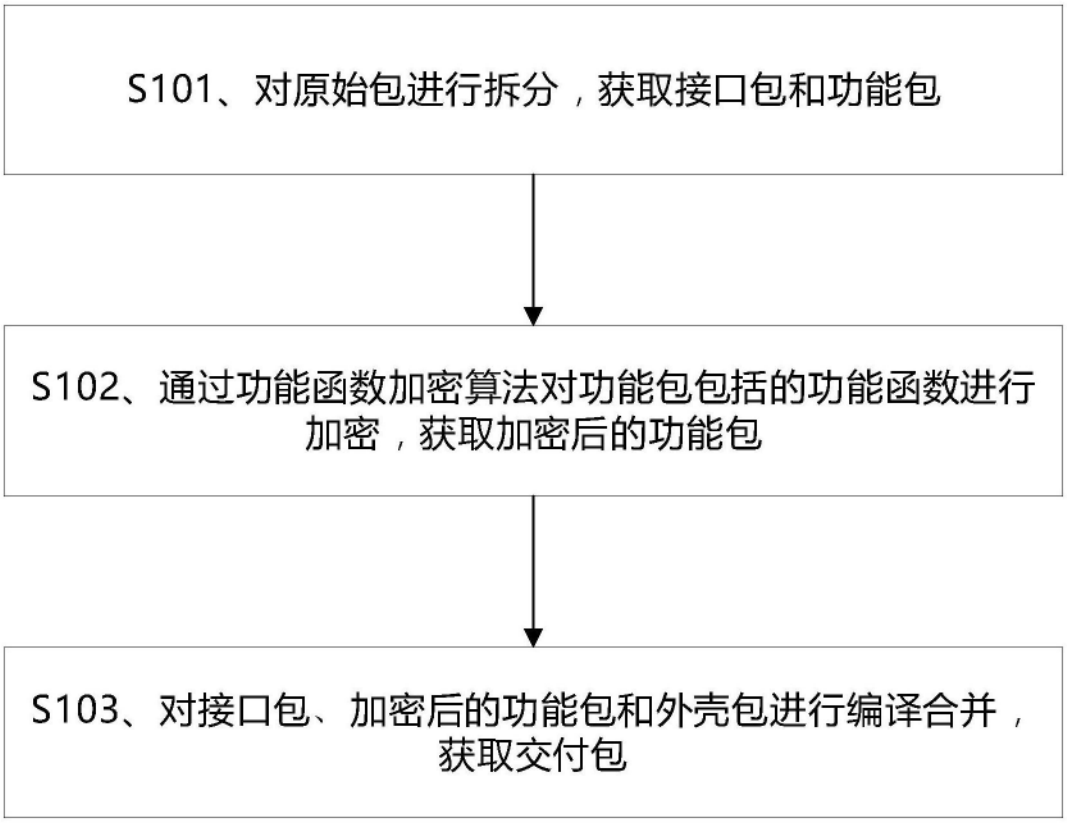

S101、对原始包进行拆分，获取接口包和功能包
S102、通过功能函数加密算法对功能包包括的功能函数进行加密，获取加密后的功能包
S103、对接口包、加密后的功能包和外壳包进行编译合并，获取交付包

图1

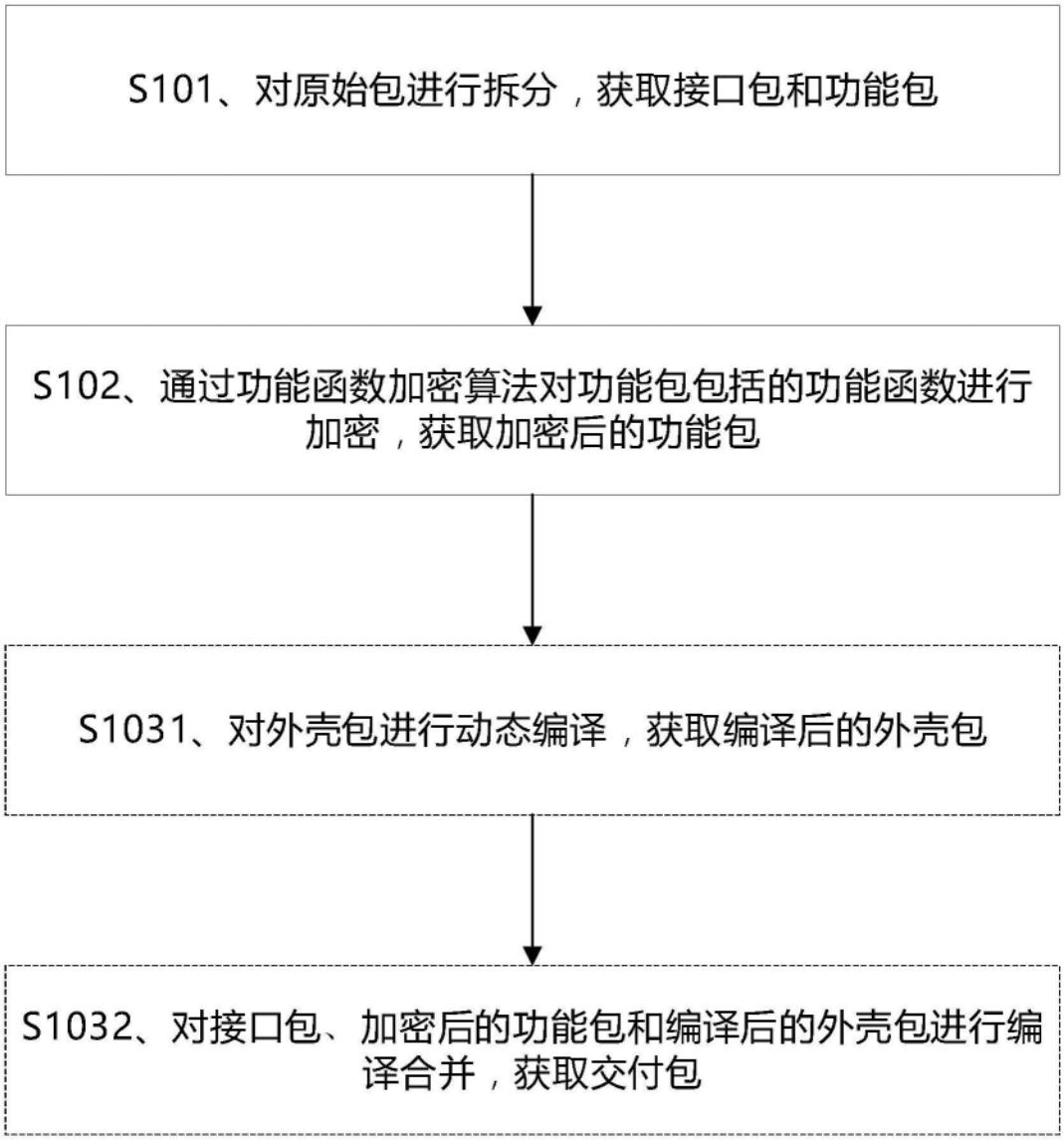

S101、对原始包进行拆分，获取接口包和功能包
S102、通过功能函数加密算法对功能包包括的功能函数进行加密，获取加密后的功能包
S1031、对外壳包进行动态编译，获取编译后的外壳包
S1032、对接口包、加密后的功能包和编译后的外壳包进行编译合并，获取交付包

图2

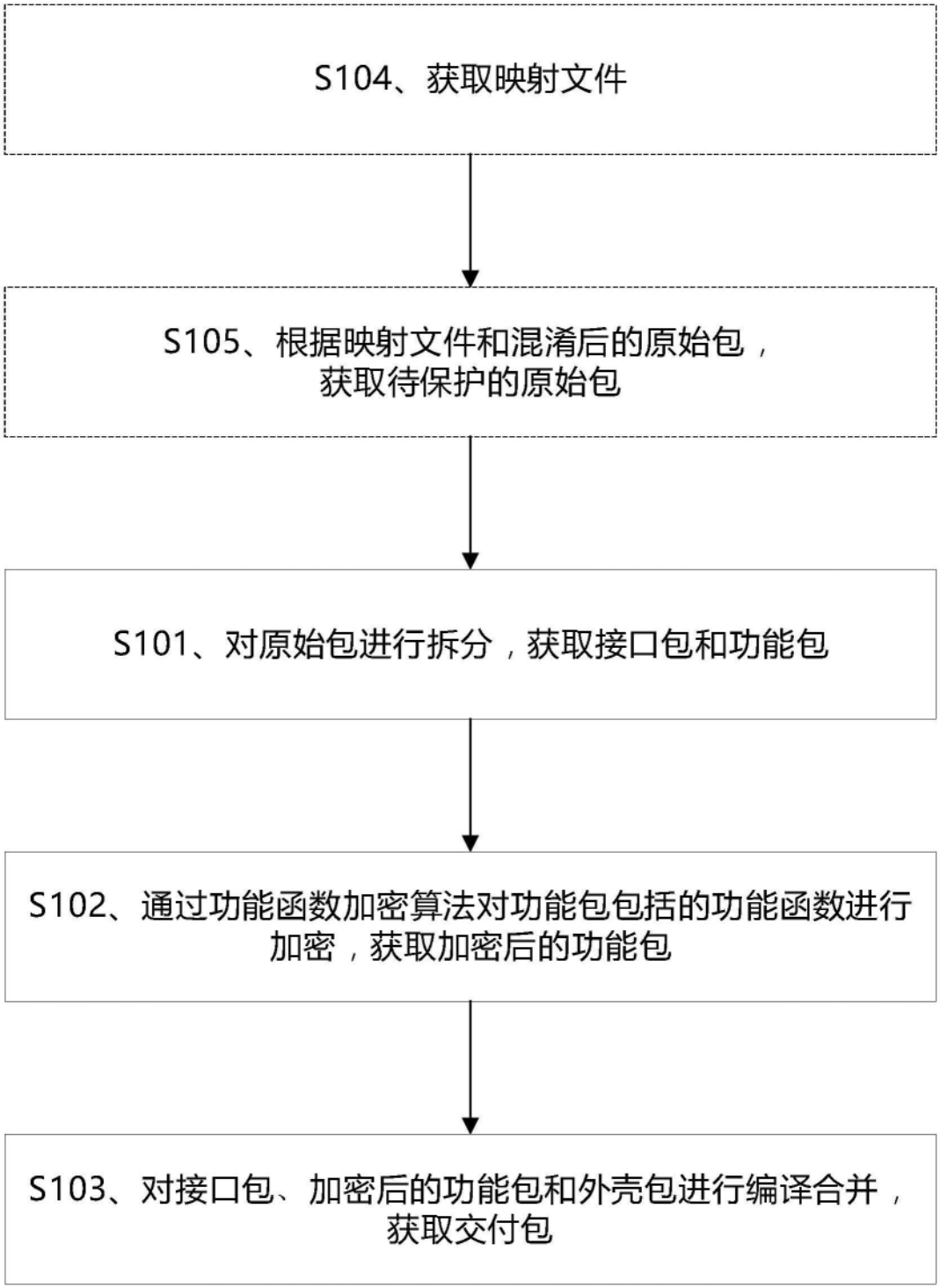

S104、获取映射文件
S105、根据映射文件和混淆后的原始包，获取待保护的原始包
S101、对原始包进行拆分，获取接口包和功能包
S102、通过功能函数加密算法对功能包包括的功能函数进行加密，获取加密后的功能包
S103、对接口包、加密后的功能包和外壳包进行编译合并，获取交付包

图3

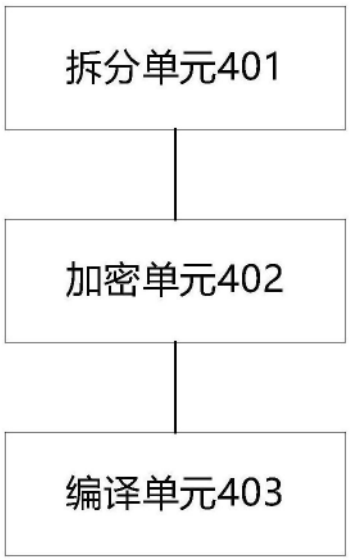


图4

图5

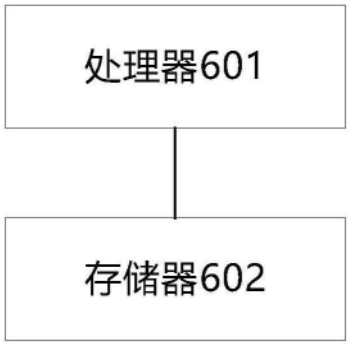


图6

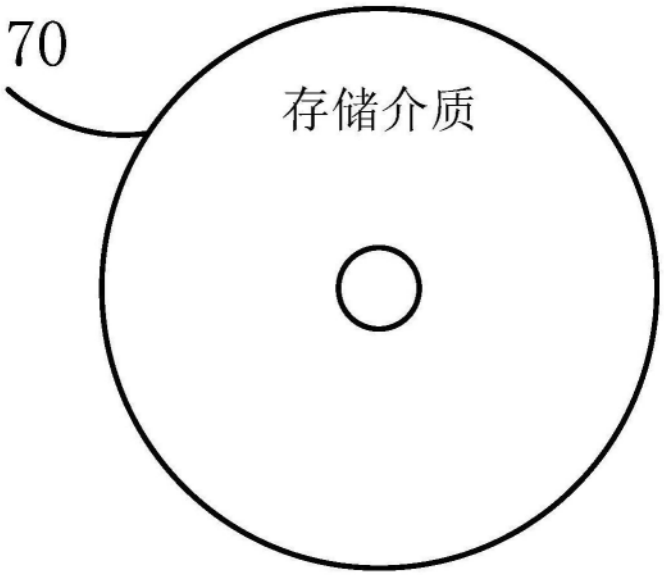


图7