(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 113282294 B
(45)授权公告日 2024.07.05

(21)申请号 202110547088.9

(22)申请日 2021.05.19

(65)同一申请的已公布的文献号
申请公布号 CN 113282294 A

(43)申请公布日 2021.08.20

(73)专利权人 武汉极意网络科技有限公司
地址 430223 湖北省武汉市东湖开发区大学园路武汉大学科技园内兴业楼2单元2楼204室—020号

(72)发明人 陈颂颂 谢强 许伟

(74)专利代理机构 武汉卓越志诚知识产权代理事务所(特殊普通合伙) 42266
专利代理师 胡婷婷

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

(56)对比文件
CN 107908933 A,2018.04.13
CN 111159662 A,2020.05.15

审查员 齐凌云

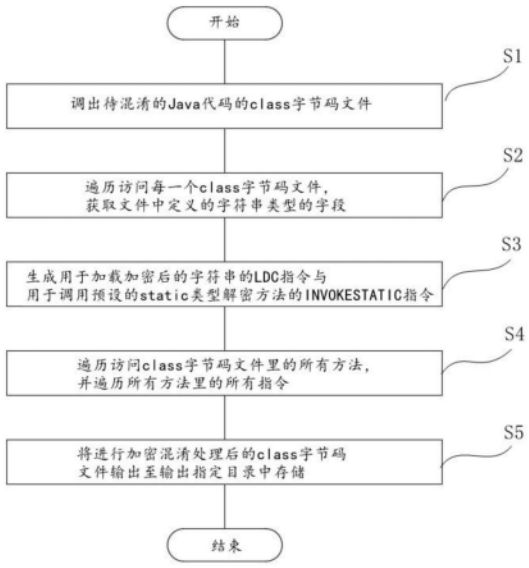
权利要求书2页 说明书6页 附图2页

(54)发明名称

基于安卓平台Java字符串混淆的方法和装置

(57)摘要

本发明公开了一种基于安卓平台Java字符串混淆的方法和装置,该装置包括信息获取与过滤模块、class字节码解析处理模块、加密混淆模块、输出模块以及解密模块,在编译时调用class字节码文件中的字符串类型的字段,用预设的加密算法对其加密,并生成相应的解密指令对加密后的字符串在运行时进行解密。通过上述方式,本发明借助于Gradle插件,融入正常的开发编译流程,通过对Java代码中的字符串进行加密混淆,并在运行期间解密使用,大大提高了Java源代码的安全性与逆向分析的难度。



CN 113282294 B

1.一种基于安卓平台Java字符串混淆的方法,其特征在于,包括以下步骤:

S1、在进行代码编译时,调出待混淆的Java代码的class字节码文件;

S2、遍历访问每一个所述class字节码文件,获取所述class字节码文件中定义的字符串类型的字段,所述字符串类型的字段包括static修饰的字段、以及无static修饰的字段,对字段值字符串调用预设的加密算法(131)进行加密操作,生成加密后的字符串;

S3、生成一条用于加载加密后的字符串的LDC指令、以及一条用于调用预设的static类型解密方法的INVOKESTATIC指令;在步骤S3中,所述LDC指令与INVOKESTATIC指令在运行时能够调用预设的解密方法解密所述加密后的字符串,并还原出原始字符串;

S4、遍历访问所述class字节码文件里的所有方法,并遍历所述所有方法里的所有指令;在步骤S4中,在编译过程中出现操作数为字符串类型的LDC指令时,对操作数字符串调用预设的加密算法(131)进行加密操作,生成加密后的字符串,并将所述加密后的字符串作为所述LDC指令新的操作数,并生成一条新的用于调用预设的static类型解密方法的INVOKESTATIC指令,从而在运行所述LDC指令与INVOKESTATIC指令时调用解密方法解密加密后的字符串,还原出原始字符串;

S5、将进行加密混淆处理后的所述class字节码文件输出至输出指定目录中存储。

2.根据权利要求1所述的基于安卓平台Java字符串混淆的方法,其特征在于,在步骤S1中,所述class字节码文件为待混淆的Java代码经javac编译器编译后产生的文件。

3.根据权利要求1所述的基于安卓平台Java字符串混淆的方法,其特征在于,在步骤S2中,所述预设的加密算法(131)包括AES加密算法以及RSA加密算法。

4.根据权利要求1所述的基于安卓平台Java字符串混淆的方法,其特征在于,在对所述static修饰的字段进行解密操作时,将所述LDC指令与INVOKESTATIC指令插入“<clinit>”方法中;在对所述无static修饰的字段进行解密操作时,将所述LDC指令与INVOKESTATIC指令插入“<init>”方法中。

5.一种基于安卓平台Java字符串混淆的装置,采用权利要求1至4任意一项所述的基于安卓平台Java字符串混淆的方法;其特征在于,包括信息获取与过滤模块(110)、class字节码解析处理模块(120)、加密混淆模块(130)、输出模块(140)以及解密模块(150);

所述信息获取与过滤模块(110)用于调出待混淆的代码信息并过滤掉不需要处理的代码信息,或者筛选出需要处理的代码信息;所述加密混淆模块(130)包括加密接口(132)以及多种加密算法(131);

所述输出模块(140)用于将加密处理后的所述class字节码内容输出至新的class字节码文件,并将加密处理后的所述class字节码内容存储在指定位置,以便于下一级编译工具对加密处理后的所述class字节码内容进行编译处理;在编译过程中出现操作数为字符串类型的LDC指令时,对操作数字符串调用预设的加密算法(131)进行加密操作,生成加密后的字符串,并将所述加密后的字符串作为所述LDC指令新的操作数,并生成一条新的用于调用预设的static类型解密方法的INVOKESTATIC指令,从而在运行所述LDC指令与INVOKESTATIC指令时调用解密方法解密加密后的字符串,还原出原始字符串。

6.根据权利要求5所述的基于安卓平台Java字符串混淆的装置,其特征在于,所述代码信息包括class字节码文件;所述class字节码解析处理模块(120)用于读取与解析所述class字节码文件,并遍历访问所述class字节码文件中的所有字符串常量,根据所述字符

串常量生成新的字节码指令,并将所述字节码指令插入至指定位置后生成混淆后的class字节码内容。

7.根据权利要求5所述的基于安卓平台Java字符串混淆的装置,其特征在于,所述多种加密算法(131)为系统默认的不同的加密算法(131),以实现快速集成使用;所述解密模块(150)包括解密接口(152)以及多种解密算法(151),所述多种解密算法(151)分别与所述多种加密算法(131)对应。

基于安卓平台Java字符串混淆的方法和装置

技术领域

[0001] 本发明涉及安卓平台开发领域,特别是涉及一种基于安卓平台Java字符串混淆的方法和装置。

背景技术

[0002] 目前,对于AndroidAPP/SDK的开发,主流开发工具为Android Studio,编译工具链为Gradle。在进行项目开发时,程序员可通过minify Enabled与proguard Files配置启用ProGuard混淆插件并指定混淆配置,在编译期间对java代码进行压缩、优化、混淆。在编译过程中,java代码里定义的常量字符串变量会被直接插入到引用变量的地方。由于现在市面上java反编译工具比较成熟,编译产物经第三方工具反编译后,可以直接查看到这些常量字符串,这会导致应用里面使用到的URL地址、接入第三方SDK需要的APPID与APPKEY、部分加密算法的key、公钥、密钥、日志TAG、以及前后缀等字符串很容易被逆向分析得到,从而造成信息泄露,或者降低逆向分析源代码的成本,存在很大的安全隐患问题。

[0003] 现有技术提出了一种源代码中字符串的混淆方法和装置,该方法和装置应用于软件开发工具中,具体为利用Clang工具对待混淆源代码进行编译,得到语法树;对语法树进行遍历,得到源代码中字符串的字符串常量列表和字符串引用列表;根据字符串常量列表中指示的位置读出字符串常量并加密处理,并以加密后的加密字符串替换原有的字符串常量;根据字符串引用列表查找加密字符串,并在加密字符串所在源代码中的位置插入解密代码。通过替换处理,使源代码中所有的字符串常量转换为加密后的加密字符串,而不再是可能会造成泄密的明文信息,从而避免了相应的安全风险。这种技术虽然将源代码中字符串常量列表和字符串引用列表替换成加密字符串,并插入了解密代码,但是这种直接替换的方式,很容易被不法分子利用反编译软件寻回到原址,并加以破解,加密的效果不好。

[0004] 现有技术提供了一种基于Java代码的混淆方法和装置,该基于Java代码的混淆方法包括:获取待混淆的代码信息,该代码信息包括源代码、源代码中需要编译的类、函数和变量;对代码信息进行预编译,以确定具有预设格式的归档文件;根据利用预设的反编译工具对归档文件进行的反编译,结合混淆需求,确定是否对归档文件进行混淆;当确定对归档文件进行混淆时,根据混淆需求,对归档文件执行混淆操作,以确定混淆后的目标文件;其中,混淆操作包括对字符串常量进行加密和对类、函数以及XML文件中的类进行混淆改名。这种技术采用预编译以及反编译的方式,并对需要混淆的字符串常量进行混淆改名,很容易被反追踪,造成信息泄漏。

[0005] 因此,设计一种集成方式简单、不需要手动修改源代码、能配置多种加解密算法、加密效果好的基于安卓平台Java字符串混淆的方法和装置就很有必要。

发明内容

[0006] 为了克服上述问题,本发明提供了一种基于安卓平台Java字符串混淆的方法和装置,本发明借助于Gradle插件,融入正常的开发编译流程,在编译过程中通过对Java代码中

的字符串进行加密混淆,并在运行期间解密使用,大大提高了Java源代码的安全性与逆向分析的难度。

[0007] 为实现上述的目的,本发明采用的技术方案是:

[0008] 一种基于安卓平台Java字符串混淆的方法,包括以下步骤:

[0009] S1、在进行代码编译时,调出待混淆的Java代码的class字节码文件;

[0010] S2、遍历访问每一个所述class字节码文件,获取所述class字节码文件中定义的字符串类型的字段,所述字符串类型的字段包括static修饰的字段、以及无static修饰的字段,对字段值字符串调用预设的加密算法进行加密操作,生成加密后的字符串;

[0011] S3、生成一条用于加载加密后的字符串的LDC指令、以及一条用于调用预设的static类型解密方法的INVOKESTATIC指令;

[0012] S4、遍历访问所述class字节码文件里的所有方法,并遍历所述所有方法里的所有指令;

[0013] S5、将进行加密混淆处理后的所述class字节码文件输出至输出指定目录中存储。

[0014] 进一步的,在步骤S1中,所述class字节码文件为待混淆的Java代码经javac编译器编译后产生的文件。

[0015] 进一步的,在步骤S2中,所述预设的加密算法包括AES加密算法以及RSA加密算法。

[0016] 进一步的,在步骤S3中,所述LDC指令与INVOKESTATIC指令在运行时能够调用预设的解密方法解密所述加密后的字符串,并还原出原始字符串。

[0017] 进一步的,在对所述static修饰的字段进行解密操作时,将所述LDC指令与INVOKESTATIC指令插入“<clinit>”方法中;在对所述无static修饰的字段进行解密操作时,将所述LDC指令与INVOKESTATIC指令插入“<init>”方法中。

[0018] 进一步的,在步骤S4中,在编译过程中出现操作数为字符串类型的LDC指令时,对操作数字符串调用预设的加密算法进行加密操作,生成加密后的字符串,并将所述加密后的字符串作为所述LDC指令新的操作数,并生成一条新的用于调用预设的static类型解密方法的INVOKESTATIC指令,从而在运行所述LDC指令与INVOKESTATIC指令时调用解密方法解密加密后的字符串,还原出原始字符串。

[0019] 一种基于安卓平台Java字符串混淆的装置,采用所述基于安卓平台Java字符串混淆的方法;包括信息获取与过滤模块、class字节码解析处理模块、加密混淆模块、输出模块以及解密模块;

[0020] 所述信息获取与过滤模块用于调出待混淆的代码信息并过滤掉不需要处理的代码信息,或者筛选出需要处理的代码信息;

[0021] 所述加密混淆模块包括加密接口以及多种加密算法;

[0022] 进一步的,所述代码信息包括class字节码文件;所述class字节码解析处理模块用于读取与解析所述class字节码文件,并遍历访问所述class字节码文件中的所有字符串常量,根据所述字符串常量生成新的字节码指令,并将所述字节码指令插入至指定位置后生成混淆后的class字节码内容。

[0023] 进一步的,所述输出模块用于将加密处理后的所述class字节码内容输出至新的class字节码文件,并将加密处理后的所述class字节码内容存储在指定位置,以便于下一级编译工具对加密处理后的所述class字节码内容进行编译处理。

[0024] 进一步的,所述多种加密算法为系统默认的不同的加密算法,以实现快速集成使用;所述解密模块包括解密接口以及多种解密算法,所述多种解密算法分别与所述多种加密算法对应。

[0025] 与现有技术相比,本发明的有益效果是:

[0026] 1.本发明的基于安卓平台Java字符串混淆的方法,通过Gradle插件融入正常的开发编译流程,在编译过程中通过对Java代码中的字符串进行加密混淆,并在运行期间解密使用,大大提高了Java源代码的安全性与逆向分析的难度。Gradle插件不与现有的ProGuard混淆装置冲突,二者可以配合使用,并且,采用Gradle插件的形式,使得整体集成方式简单,不需要手动修改源代码,不影响开发流程的正常进行。

[0027] 2.本发明的基于安卓平台Java字符串混淆的装置,通过设置信息获取与过滤模块,该模块内置灵活的文件过滤筛选功能,能够快速对项目中的Java字节码文件进行筛选过滤,以筛选出需要处理的文件,大大提高了本装置的工作效率。

[0028] 3.本发明的基于安卓平台Java字符串混淆的装置,通过设置加密混淆模块,该模块内置多种系统默认的加密算法,并在解密模块中设置对应的解密算法,采用系统默认的不同的加解密算法供使用者选择,能够快速集成使用,大大提高了将本发明应用在开发过程中的速度。并且,本装置还提供加解密接口,方便使用者按照接口预设的规范格式,自行实现相应的加解密算法,能够实现装置的个性化定制,适用于不同的场合,更具有推广价值。

附图说明

[0029] 图1是本发明的基于安卓平台Java字符串混淆的方法的步骤流程图;

[0030] 图2是本发明的基于安卓平台Java字符串混淆的方法的运行示意图;

[0031] 图3是本发明的基于安卓平台Java字符串混淆的装置的结构示意图;

[0032] 图4是本发明的基于安卓平台Java字符串混淆的装置的加解密模块结构对应示意图;

[0033] 附图中各部件的标记如下:110、信息获取与过滤模块;120、class字节码解析处理模块;130、加密混淆模块;131、加密算法;132、加密接口;140、输出模块;150、解密模块;151、解密算法;152、解密接口。

具体实施方式

[0034] 下面结合附图对本发明的较佳实施例进行详细阐述,以使本发明的优点和特征能更易于被本领域技术人员理解,从而对本发明的保护范围做出更为清楚明确的界定。显然,所描述的实施例仅仅是本发明的一部分实施例,而不是全部的实施例。基于本发明的实施例,本领域普通技术人员在没有做出创造性劳动的前提下所得到的所有其它实施例,都属于本发明所保护的范围。

[0035] 实施例1

[0036] 如图1至3所示,一种基于安卓平台Java字符串混淆的方法,通过Gradle插件融入正常的开发编译流程,在编译过程中通过对Java代码中的字符串进行加密混淆,并在运行期间解密使用,大大提高了Java源代码的安全性与逆向分析的难度。Gradle插件不与现有

的ProGuard混淆装置冲突,二者可以配合使用,并且,采用Gradle插件的形式,使得整体集成方式简单,不需要手动修改源代码,不影响开发流程的正常进行。

[0037] 值得注意的是,Gradle插件具有非常好的拓展性,将其应用在开发项目中,能够在项目中添加可依赖的第三方库,或者在项目中添加有用的默认设置和约定。并且Gradle插件提供了可配置的依赖管理方案,从而实现了本发明在不同的平台和机器上产生相同的构建结果。

[0038] 该方法包括以下步骤:

[0039] S1、在进行代码编译时,调出待混淆的Java代码的class字节码文件。

[0040] 本步骤中,代码为开发人员提前开发好的源代码,这些源代码包括许多的字符串,例如URL地址、接入第三方SDK需要的APPID与APPKEY、部分加密算法131的key、公钥、密钥、日志TAG、以及前后缀等字符串。

[0041] class字节码文件为待混淆的Java代码经javac编译器编译后产生的文件。

[0042] S2、遍历访问每一个class字节码文件,获取class字节码文件中定义的字符串类型的字段,字符串类型的字段包括static修饰的字段、以及无static修饰的字段,对字段值字符串调用预设的加密算法131进行加密操作,生成加密后的字符串。

[0043] 本步骤中,预设的加密算法131包括AES加密算法以及RSA加密算法,预设的加密算法131也可以为其他等常见的对称或非对称加密算法,具体并不以此为限。便于加密算法131的快速集成使用,提高加密的效率。

[0044] static修饰的字段属于类级别,而不是对象级别,这些static修饰的字段会随着所在类的加载而加载,即static修饰的字段能够被该类型的所有对象共享,便于在加载时更好的调用。特别的,在static方法中,加载时只能调用静态成员(static成员),而对于非static方法,可以调用静态成员,也可以调用实例成员。

[0045] S3、生成一条用于加载加密后的字符串的LDC指令、以及一条用于调用预设的static类型解密方法的INVOKESTATIC指令。

[0046] 在本步骤中,LDC指令与INVOKESTATIC指令在运行时能够调用预设的解密方法解密加密后的字符串,并还原出原始字符串。在采用具体的操作方式时,在对static修饰的字段进行解密操作时,将LDC指令与INVOKESTATIC指令插入“<clinit>”方法(类构造器方法)中。在对无static修饰的字段进行解密操作时,将LDC指令与INVOKESTATIC指令插入“<init>”方法(类实例构造器方法)中。

[0047] LDC指令用于把常量池中对应的内容压入操作数栈中,LDC指令之后紧跟的操作数是一个指向常量池偏移量的数字。

[0048] INVOKESTATIC指令为调用一个类方法(static修饰的方法),用于调用预设的解密方法,将加密后的字符串解密还原成原始字符串。

[0049] S4、遍历访问class字节码文件里的所有方法,并遍历所有方法里的所有指令。

[0050] 在本步骤中,在编译过程中出现操作数为字符串类型的LDC指令时,对操作数字符串调用预设的加密算法131进行加密操作,生成加密后的字符串,并将加密后的字符串作为LDC指令新的操作数,并生成一条新的用于调用预设的static类型解密方法的INVOKESTATIC指令,从而在运行LDC指令与INVOKESTATIC指令时调用解密方法解密加密后的字符串,还原出原始字符串。

[0051] 特别的,预设的static类型解密方法的INVOKESTATIC指令能够更快速的找到解密方法的地址,并获得相应的数据。

[0052] S5、将进行加密混淆处理后的class字节码文件输出至输出指定目录中存储。

[0053] 采用本发明的方法,在编译打包期间,对编译java代码后的中间文件class字节码进行处理,将class字节码里面出现的各种字符串常量进行加密混淆,并植入解密代码逻辑,保证编译后的编译产物既不容易被他人逆向分析直接拿到字符串原文,又能够在运行时解密字符串正常使用,从而保护代码信息安全,提升应用被逆向分析的难度。另外,解密代码逻辑可以通过c/c++语言实现,借助反调试、反hook等手段可防止解密逻辑被动态调试分析,从而更好的确保加密/混淆逻辑的可靠性。

[0054] 一种基于安卓平台Java字符串混淆的装置,采用基于安卓平台Java字符串混淆的方法。该装置以Gradle插件的形式应用于软件开发工具中,用于对应用程序项目的源代码中的字符串进行加密混淆处理,从而保护代码信息安全,并提升应用被逆向分析的难度。

[0055] 基于安卓平台Java字符串混淆的装置包括信息获取与过滤模块110、class字节码解析处理模块120、加密混淆模块130、输出模块140以及解密模块150。

[0056] 信息获取与过滤模块110用于调出待混淆的代码信息并过滤掉不需要处理的代码信息,或者筛选出需要处理的代码信息。信息获取与过滤模块110的过滤功能,能够在调用代码信息时,将不需要的代码信息直接排除,以集中对需要处理的代码信息进行加密混淆,大大提高了开发过程中的加密效率。特别的,代码信息包括class字节码文件。

[0057] class字节码解析处理模块120用于读取与解析class字节码文件,并遍历访问class字节码文件中的所有字符串常量,根据字符串常量生成新的字节码指令,并将字节码指令插入至指定位置后生成混淆后的class字节码内容。从而将源代码中的class字节码文件快速调用后进行处理。

[0058] 加密混淆模块130包括加密接口132以及多种加密算法131。特别的,多种加密算法131为系统默认的不同的加密算法131,以实现快速集成使用。加密接口132能够根据加密接口132预设的格式规范,自行实现相应的加密算法131,方便使用者进行个性化设计。

[0059] 输出模块140用于将加密处理后的class字节码内容输出至新的class字节码文件,并将加密处理后的class字节码内容存储在指定位置,以便于下一级编译工具对加密处理后的class字节码内容进行编译处理。

[0060] 解密模块150包括解密接口152以及多种解密算法151,多种解密算法151分别与多种加密算法131对应。解密接口152与加密接口132对应,由使用者根据自设的加密接口132设计相应的解密接口152,从而实现相应的加解密算法,增强了本装置的多样性以及实用性。

[0061] 在采用具体的实施方式时,程序员对源代码进行编译,信息获取与过滤模块110获取待混淆的class字节码文件,并过滤掉不需要处理的代码。随后,class字节码解析处理模块120读取class字节码文件。加密混淆模块130对读取的class字节码文件进行加密。输出模块140将加密后的class字节码文件输出到新的class字节码文件中,并存放到指定位置。同时,在对源代码进行编译时,解密模块150将加密后的字符串还原成原始字符串,保证程序的正常运行。

[0062] 以上所述仅用以说明本发明的技术方案,而非对其进行限制;尽管参照前述实施

例对本发明进行了详细的说明,本领域的普通技术人员应当理解:其依然可以对前述实施例所记载的技术方案进行修改,或者对其中部分或者全部技术特征进行等同替换;凡是利用本发明说明书及附图内容所作的等效结构或等效流程变换,或直接或间接运用在其他相关的技术领域,均同理包括在本发明的专利保护范围内。

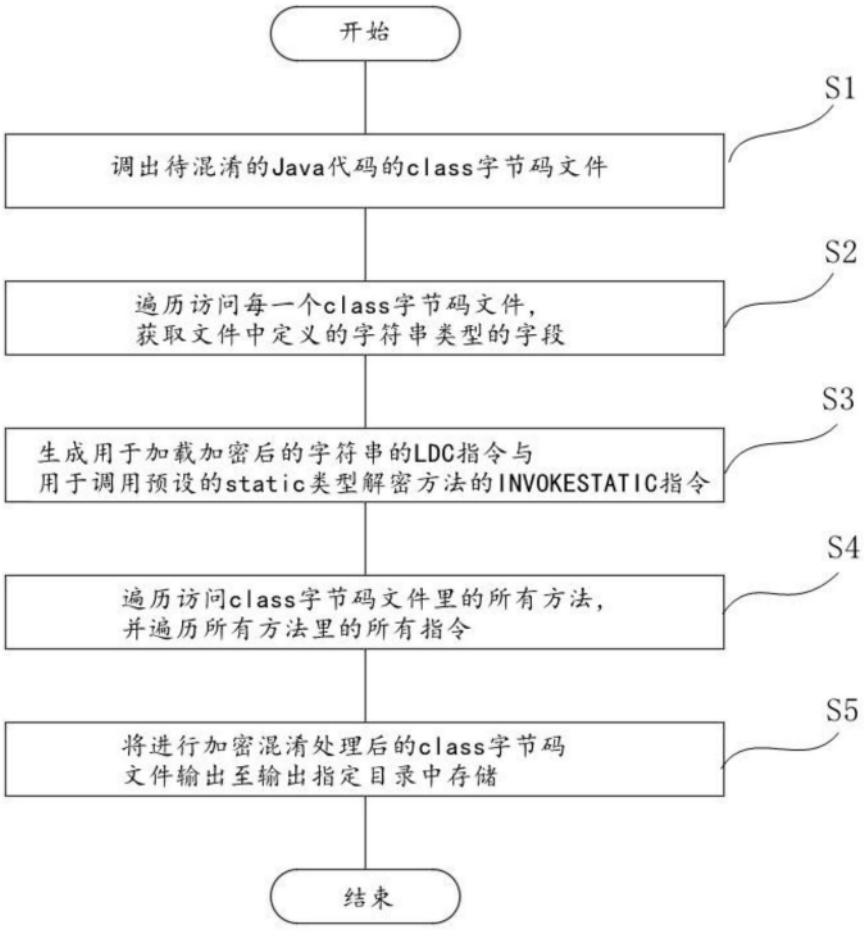

开始
S1
调出待混淆的Java代码的class字节码文件
S2
遍历访问每一个class字节码文件，
获取文件中定义的字符串类型的字段
S3
生成用于加载加密后的字符串的LDC指令与
用于调用预设的static类型解密方法的INVOKESTATIC指令
S4
遍历访问class字节码文件里的所有方法，
并遍历所有方法里的所有指令
S5
将进行加密混淆处理后的class字节码
文件输出至输出指定目录中存储
结束

图1

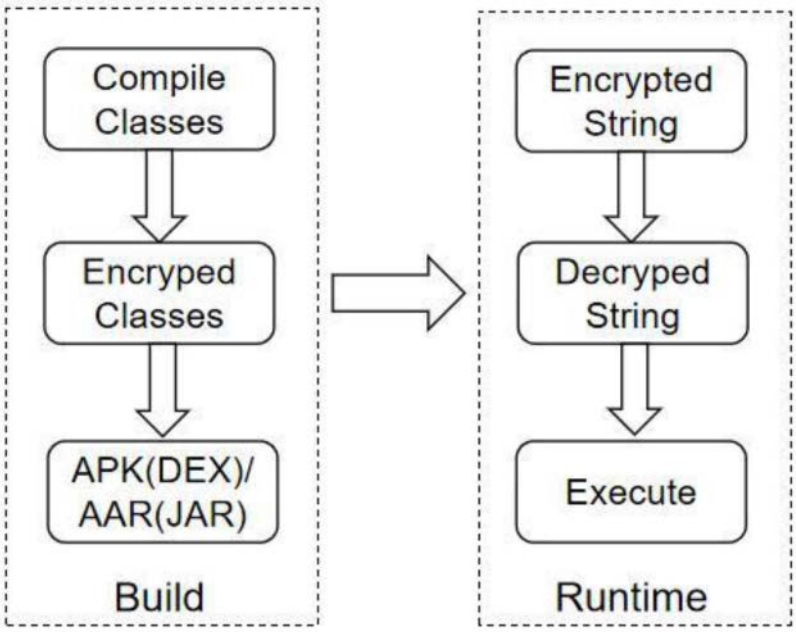

Compile Classes
Encryped Classes
APK(DEX)/ AAR(JAR)
Build
Encrypted String
Decryped String
Execute
Runtime

图2

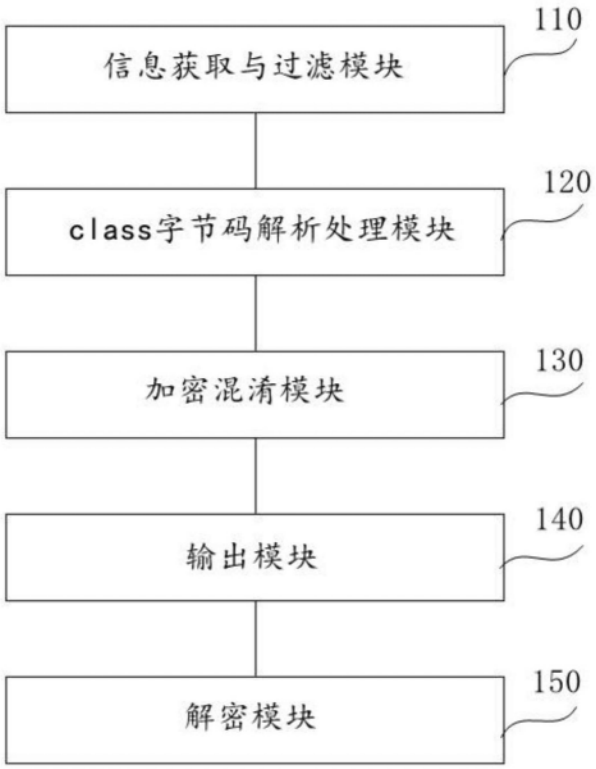

110
信息获取与过滤模块
120
class字节码解析处理模块
130
加密混淆模块
140
输出模块
150
解密模块

图3

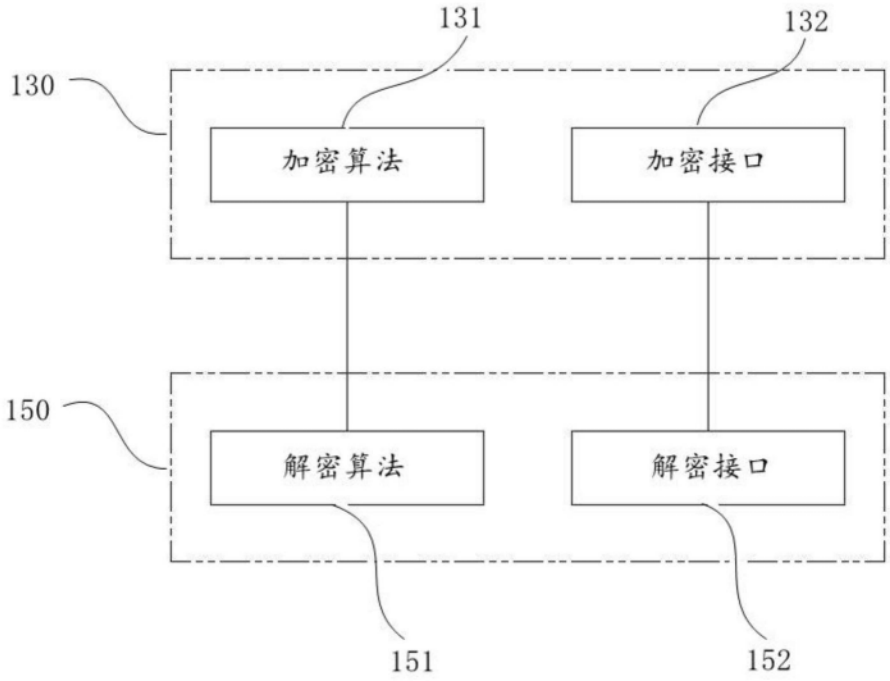

131
132
130
加密算法
加密接口
150
解密算法
解密接口
151
152

图4