(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 107391973 A
(43)申请公布日 2017.11.24

(21)申请号 201710581056.4

(22)申请日 2017.07.17

(71)申请人 北京深思数盾科技股份有限公司
地址 100193 北京市海淀区西北旺东路10号院东区5号楼5层510

(72)发明人 孙吉平 荣国枫

(74)专利代理机构 北京品源专利代理有限公司 11332
代理人 孟金喆

(51)Int.Cl.
G06F 21/12(2013.01)

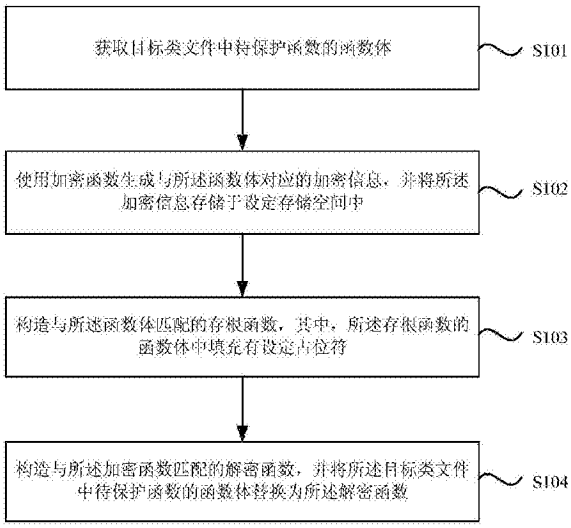
权利要求书2页 说明书10页 附图4页

(54)发明名称

一种函数保护方法及装置

(57)摘要

本发明实施例公开了一种函数保护方法及装置，该方法包括：获取目标类文件中待保护函数的函数体；使用加密函数生成与所述函数体对应的加密信息，并将所述加密信息存储于设定存储空间中；构造与所述函数体匹配的存根函数，其中，所述存根函数的函数体中填充有设定占位符；构造与所述加密函数匹配的解密函数，并将所述目标类文件中待保护函数的函数体替换为所述解密函数。本发明实施例能够极大地降低代码被破解的可能性，达到了隐藏代码的目的，能够避免函数代码被反编译的情况发生，有效地提升了函数代码的安全性。



CN 107391973 A

1.一种函数加密方法,其特征在于,包括:

获取目标类文件中待保护函数的函数体;

使用加密函数生成与所述函数体对应的加密信息,并将所述加密信息存储于设定存储空间中;

构造与所述函数体匹配的存根函数,其中,所述存根函数的函数体中填充有设定占位符;

构造与所述加密函数匹配的解密函数,并将所述目标类文件中待保护函数的函数体替换为所述解密函数。

2.根据权利要求1所述的方法,其特征在于,所述存根函数包括下述至少一项特征:

所述存根函数的函数体与所述待保护函数的函数体长度相等;

所述存根函数的参数类型与所述待保护函数的参数类型相同;

所述存根函数的返回值类型与所述待保护函数的返回值类型相同。

3.根据权利要求1所述的方法,其特征在于,使用加密函数生成与所述函数体对应的加密信息,并将所述加密信息存储于设定存储空间中,包括:

使用中间组件从加密锁中调用加密函数;

使用所述加密函数对待保护函数的函数体进行加密,生成与所述函数体对应的加密信息;

将所述加密信息存储于加密锁中;

其中,所述加密锁包括与所述目标类文件进行通信的接口。

4.根据权利要求1-3任一项所述的方法,其特征在于,获取目标类文件中待保护函数的函数体,包括:

获取目标类文件中的待保护函数;

使用解析工具解析所述目标类文件的文件格式,获取所述待保护函数的起始地址和终止地址;

根据所述起始地址和所述终止地址,获取所述待保护函数的函数体。

5.一种函数解密方法,其特征在于,包括:

在检测到目标函数被运行时,在目标类文件中获取所述目标函数的函数体;

如果识别出所述函数体中包括解密函数,则在设定存储空间中,获取与所述函数体对应的加密信息,并使用所述解密函数对所述加密信息进行解密,获取解密数据;

获取与所述函数体匹配的存根函数,其中,所述存根函数的函数体中填充有设定占位符;

将所述存根函数的函数体替换为所述解密数据后,运行所述存根函数;

在保存所述存根函数的运行结果后,重新将所述存根函数的函数体替换为所述占位符。

6.一种函数加密装置,其特征在于,包括:

函数体获取模块,用于获取目标类文件中待保护函数的函数体;

加密模块,用于使用加密函数生成与所述函数体对应的加密信息,并将所述加密信息存储于设定存储空间中;

存根函数构造模块,用于构造与所述函数体匹配的存根函数,其中,所述存根函数的函

数体中填充有设定占位符；

解密函数构造模块，用于构造与所述加密函数匹配的解密函数，并将所述目标类文件中待保护函数的函数体替换为所述解密函数。

7.根据权利要求6所述的装置，其特征在于，所述存根函数构造模块包括：

第一构造单元，用于所述存根函数的函数体与所述待保护函数的函数体长度相等；

第二构造单元，用于所述存根函数的参数类型与所述待保护函数的参数类型相同；

第三构造单元，用于所述存根函数的返回值类型与所述待保护函数的返回值类型相同。

8.根据权利要求6所述的装置，其特征在于，所述加密模块包括：

调用单元，用于使用中间组件从加密锁中调用加密函数；

加密单元，用于使用所述加密函数对待保护函数的函数体进行加密，生成与所述函数体对应的加密信息；

存储单元，将所述加密信息存储于加密锁中；

其中，所述加密锁包括与所述目标类文件进行通信的接口。

9.根据权利要求6-8任一项所述的装置，其特征在于，所述函数体获取模块包括：

函数获取单元，用于获取目标类文件中的待保护函数；

地址获取单元，用于使用解析工具解析所述目标类文件的文件格式，获取所述待保护函数的起始地址和终止地址；

函数体获取单元，用于根据所述起始地址和所述终止地址，获取所述待保护函数的函数体。

10.一种函数解密装置，其特征在于，包括：

目标函数获取模块，用于在检测到目标函数被运行时，在目标类文件中获取所述目标函数的函数体；

解密模块，用于如果识别出所述函数体中包括解密函数，则在设定存储空间中，获取与所述函数体对应的加密信息，并使用所述解密函数对所述加密信息进行解密，获取解密数据；

存根函数获取模块，用于获取与所述函数体匹配的存根函数，其中，所述存根函数的函数体中填充有设定占位符；

运行模块，用于将所述存根函数的函数体替换为所述解密数据后，运行所述存根函数；

替换模块，用于在保存所述存根函数的运行结果后，重新将所述存根函数的函数体替换为所述占位符。

一种函数保护方法及装置

技术领域

[0001] 本发明实施例涉及信息安全技术领域，尤其涉及一种函数保护方法及装置。

背景技术

[0002] 随着互联网的蓬勃发展，计算机编程语言(JAVA)以其可移植性、多线程性等特点提升了电子产品的智能化程度。但是由于JAVA字节码的抽象级别较高，因此容易被反编译为源代码，造成程序代码被破解的安全性问题。

[0003] 现有技术中，为了保护JAVA字节码的安全性，常使用三种解决方案：

[0004] 1、加密类文件，为了防止类(.class)文件被直接反编译，开发人员将关键的类文件进行加密，并且通过自定义的类加载机制(ClassLoader)进行解密之后装载到虚拟机中执行。但是，由于自定义的类加载机制本身没有被加密，破解者容易获取类文件。

[0005] 2、将关键代码转换为本地代码，将程序关键代码转换为本地代码是一种防止反编译的有效方法，但是是以牺牲JAVA的跨平台移植性为代价的，还会加重软件支持和维护的工作，增加了系统的复杂性。

[0006] 3、使用代码混淆技术，通过对JAVA字节码使用混淆技术，使代码逻辑产生改变，不易被读取。但是，对混淆技术依赖性较大，而且可以被反编译。

发明内容

[0007] 本发明实施例提供一种函数保护方法及装置，以解决现有技术中函数代码容易被反编译的问题。

[0008] 第一方面，本发明实施例提供了一种函数加密方法，包括：

[0009] 获取目标类文件中待保护函数的函数体；

[0010] 使用加密函数生成与所述函数体对应的加密信息，并将所述加密信息存储于设定存储空间中；

[0011] 构造与所述函数体匹配的存根函数，其中，所述存根函数的函数体中填充有设定占位符；

[0012] 构造与所述加密函数匹配的解密函数，并将所述目标类文件中待保护函数的函数体替换为所述解密函数。

[0013] 进一步地，所述存根函数包括下述至少一项特征：

[0014] 所述存根函数的函数体与所述待保护函数的函数体长度相等；

[0015] 所述存根函数的参数类型与所述待保护函数的参数类型相同；

[0016] 所述存根函数的返回值类型与所述待保护函数的返回值类型相同。

[0017] 进一步地，使用加密函数生成与所述函数体对应的加密信息，并将所述加密信息存储于设定存储空间中，包括：

[0018] 使用中间组件从加密锁中调用加密函数；

[0019] 使用所述加密函数对待保护函数的函数体进行加密，生成与所述函数体对应的加

密信息；

[0020] 将所述加密信息存储于加密锁中；

[0021] 其中，所述加密锁包括与所述目标类文件进行通信的接口。

[0022] 进一步地，获取目标类文件中待保护函数的函数体，包括：

[0023] 获取目标类文件中的待保护函数；

[0024] 使用解析工具解析所述目标类文件的文件格式，获取所述待保护函数的起始地址和终止地址；

[0025] 根据所述起始地址和所述终止地址，获取所述待保护函数的函数体。

[0026] 第二方面，本发明实施例提供了一种函数解密方法，包括：

[0027] 在检测到目标函数被运行时，在目标类文件中获取所述目标函数的函数体；

[0028] 如果识别出所述函数体中包括解密函数，则在设定存储空间中，获取与所述函数体对应的加密信息，并使用所述解密函数对所述加密信息进行解密，获取解密数据；

[0029] 获取与所述函数体匹配的存根函数，其中，所述存根函数的函数体中填充有设定占位符；

[0030] 将所述存根函数的函数体替换为所述解密数据后，运行所述存根函数；

[0031] 在保存所述存根函数的运行结果后，重新将所述存根函数的函数体替换为所述占位符。

[0032] 第三方面，本发明实施例还提供了一种函数加密装置，包括：

[0033] 函数体获取模块，用于获取目标类文件中待保护函数的函数体；

[0034] 加密模块，用于使用加密函数生成与所述函数体对应的加密信息，并将所述加密信息存储于设定存储空间中；

[0035] 存根函数构造模块，用于构造与所述函数体匹配的存根函数，其中，所述存根函数的函数体中填充有设定占位符；

[0036] 解密函数构造模块，用于构造与所述加密函数匹配的解密函数，并将所述目标类文件中待保护函数的函数体替换为所述解密函数。

[0037] 进一步地，所述存根函数构造模块包括：

[0038] 第一构造单元，用于所述存根函数的函数体与所述待保护函数的函数体长度相等；

[0039] 第二构造单元，用于所述存根函数的参数类型与所述待保护函数的参数类型相同；

[0040] 第三构造单元，用于所述存根函数的返回值类型与所述待保护函数的返回值类型相同。

[0041] 进一步地，所述加密模块包括：

[0042] 调用单元，用于使用中间组件从加密锁中调用加密函数；

[0043] 加密单元，用于使用所述加密函数对待保护函数的函数体进行加密，生成与所述函数体对应的加密信息；

[0044] 存储单元，将所述加密信息存储于加密锁中；

[0045] 其中，所述加密锁包括与所述目标类文件进行通信的接口。

[0046] 进一步地，所述函数体获取模块包括：

[0047] 函数获取单元，用于获取目标类文件中的待保护函数；

[0048] 地址获取单元，用于使用解析工具解析所述目标类文件的文件格式，获取所述待保护函数的起始地址和终止地址；

[0049] 函数体获取单元，用于根据所述起始地址和所述终止地址，获取所述待保护函数的函数体。

[0050] 第四方面，本发明实施例还提供了一种函数解密装置，包括：

[0051] 目标函数获取模块，用于在检测到目标函数被运行时，在目标类文件中获取所述目标函数的函数体；

[0052] 解密模块，用于如果识别出所述函数体中包括解密函数，则在设定存储空间中，获取与所述函数体对应的加密信息，并使用所述解密函数对所述加密信息进行解密，获取解密数据；

[0053] 存根函数获取模块，用于获取与所述函数体匹配的存根函数，其中，所述存根函数的函数体中填充有设定占位符；

[0054] 运行模块，用于将所述存根函数的函数体替换为所述解密数据后，运行所述存根函数；

[0055] 替换模块，用于在保存所述存根函数的运行结果后，重新将所述存根函数的函数体替换为所述占位符。

[0056] 本发明实施例通过对待保护函数的函数体进行加密，并构造解密函数以及与函数体匹配的存根函数，解密时运行函数体为解密数据的存根函数，保存运行结果之后，恢复存根函数的占位符。本发明实施例能够使待保护的函数只有在运行的过程中被加载至内存，在运行结束后从内存中删除，且函数运行时待保护函数不存在于类文件中，极大地降低了代码被破解的可能性，达到了隐藏代码的目的，能够避免函数代码被反编译的情况发生，有效地提升了函数代码的安全性。

附图说明

[0057] 图1是本发明实施例一中的一种函数加密方法的流程图；

[0058] 图2是本发明实施例二中的一种函数加密方法的流程图；

[0059] 图3是本发明实施例三中的一种函数解密方法的流程图；

[0060] 图4是本发明实施例四中的一种函数加密装置的结构示意图；

[0061] 图5是本发明实施例五中的一种函数解密装置的结构示意图。

具体实施方式

[0062] 下面结合附图和实施例对本发明作进一步的详细说明。可以理解的是，此处所描述的具体实施例仅仅用于解释本发明，而非对本发明的限定。另外还需要说明的是，为了便于描述，附图中仅示出了与本发明相关的部分而非全部结构。

[0063] 实施例一

[0064] 图1为本发明实施例一提供的一种函数加密方法的流程图，本实施例可适用于通过加密来进行函数保护的情况，该方法可以由一种函数加密装置来执行，该装置可以采用软件和/或硬件的方式实现，一般集成于应用系统中。

[0065] 本发明实施例一的方法具体包括：

[0066] S101、获取目标类文件中待保护函数的函数体。

[0067] 在一种可能的设计中，获取目标类文件中待保护函数的函数体可以包括：

[0068] 获取目标类文件中的待保护函数；

[0069] 使用解析工具解析所述目标类文件的文件格式，获取所述待保护函数的起始地址和终止地址；

[0070] 根据所述起始地址和所述终止地址，确定所述待保护函数的函数体。

[0071] 在一种可能的示例中，可以使用编译器将Java源程序代码编译为至少一个类文件，虚拟机根据需要加载类文件，用户选择目标类文件之后，根据目标类文件，在目标类文件的输入框中输入待保护函数的名称来获取待保护函数，和/或读取目标类文件的函数集，在函数集中选择待保护函数。解析工具用于解析目标类文件的文件格式，类文件是一种8位字节的二进制流文件，类文件包括魔数、版本号以及常量池，获取待保护函数在类文件中的起始地址和终止地址，根据起始地址和所述终止地址，获取待保护函数的函数体。

[0072] S102、使用加密函数生成与所述函数体对应的加密信息，并将所述加密信息存储于设定存储空间中。

[0073] 具体的，可以通过中间组件从设定存储空间中调用加密函数，利用加密函数对待保护函数的函数体加密，生成与函数体对应的加密信息，并将加密信息存储于设定存储空间中。其中，中间组件调用加密函数及利用加密函数对函数体加密的过程均在内存中执行，上述的中间组件可以为软件开发工具包，设定存储空间可以为可拆卸的硬件，如可以是加密锁，加密卡等信息安全设备。

[0074] S103、构造与所述函数体匹配的存根函数，其中，所述存根函数的函数体中填充有设定占位符。

[0075] 所述存根函数包括下述至少一项特征：

[0076] 所述存根函数的函数体与所述待保护函数的函数体长度相等；

[0077] 所述存根函数的参数类型与所述待保护函数的参数类型相同；

[0078] 所述存根函数的返回值类型与所述待保护函数的返回值类型相同。

[0079] 具体的，在类文件中构造存根函数，该存根函数与待保护函数的函数体相匹配，即存根函数与待保护函数的函数体所占字节数相等，存根函数与待保护函数参数类型相同，存根函数与待保护函数返回值类型相同。类文件中方法的描述符包括所有参数类型和返回值类型，使用对应的字符和字符串表示。参数类型可以为基本数据类型，返回值类型可以为对象或基本数据类型，存根函数的函数体中设置占位符或者填充空效果的代码。

[0080] S104、构造与所述加密函数匹配的解密函数，并将所述目标类文件中待保护函数的函数体替换为所述解密函数。

[0081] 具体的，构造与加密函数匹配的解密函数，可以通过解密函数解密加密信息以获得解密数据，还可以通过解密函数查找存根函数，使解密后获得的解密数据存储至存根函数。将解密函数存储至目标类文件中待保护函数的函数体所占的存储空间中。

[0082] 本发明实施例一提供的函数加密方法，能够使待保护的函数只有在运行的过程中被加载至内存，在运行结束后从内存中删除，且函数运行时待保护函数不存在于类文件中，极大地降低了代码被破解的可能性，达到了隐藏代码的目的，能够避免函数代码被反编译

的情况发生，有效地提升了函数代码的安全性。

[0083] 实施例二

[0084] 图2为本发明实施例二提供的一种函数加密方法的流程图，本发明实施例二以实施例一为基础进行了优化，具体是对使用加密函数生成与所述函数体对应的加密信息，并将所述加密信息存储于设定存储空间中的操作进一步优化，如图2所示，本发明实施例二的具体包括：

[0085] S201、获取目标类文件中待保护函数的函数体。

[0086] S202、使用中间组件从加密锁中调用加密函数。

[0087] S203、使用所述加密函数对待保护函数的函数体进行加密，生成与所述函数体对应的加密信息。

[0088] S204、将所述加密信息存储于加密锁中；其中，所述加密锁包括与所述目标类文件进行通信的接口。

[0089] 具体的，设定存储空间可以为加密锁，使用软件开发工具包从加密锁中调用加密函数，使用加密函数对待保护函数的函数体进行加密，加密的过程在内存中进行，生成与待保护函数的函数体相匹配的加密信息，并将加密信息存储于加密锁中。加密锁可以是硬件，可拆卸地安装于终端中。加密锁可以通过接口与目标类文件以及内存进行通信。

[0090] S205、构造与所述函数体匹配的存根函数，其中，所述存根函数的函数体中填充有设定占位符。

[0091] S206、构造与所述加密函数匹配的解密函数，并将所述目标类文件中待保护函数的函数体替换为所述解密函数。

[0092] 本发明实施例二提供的一种函数加密方法，通过重新构建函数内部逻辑使待保护函数存储于加密锁中，不存在于类文件中，也仅在运行该函数时短时间存在于内存中，提升了函数代码的安全性。

[0093] 实施例三

[0094] 图3为本发明实施例三提供的一种函数解密方法的流程图，本发明实施例三以上述各实施例为基础进行了优化改进，对函数解密过程进行了进一步说明，本实施例可适用于函数保护的情况，该方法可以由一种函数加密装置来执行，该装置可以采用软件和/或硬件的方式实现，一般集成于应用系统中。如图3所示，本发明实施例三的方法具体包括：

[0095] S301、在检测到目标函数被运行时，在目标类文件中获取所述目标函数的函数体。

[0096] 具体的，目标类文件中待保护的函数作为目标函数被运行时，获取目标函数的函数体。

[0097] S302、如果识别出所述函数体中包括解密函数，则在设定存储空间中，获取与所述函数体对应的加密信息，并使用所述解密函数对所述加密信息进行解密，获取解密数据。

[0098] 具体的，识别所述函数体中的解密函数，如果识别成功，则在加密锁中获取与待保护函数的函数体相匹配的加密信息，使用解密函数解密加密信息，得到解密后的解密数据。

[0099] S303、获取与所述函数体匹配的存根函数，其中，所述存根函数的函数体中填充有设定占位符。

[0100] S304、将所述存根函数的函数体替换为所述解密数据后，运行所述存根函数。

[0101] 具体的，通过解密函数根据存根函数的地址/名称等标识信息获取存根函数，并且

将解密数据复制到存根函数的函数体中，以替换函数体中填充的设定占位符之后，执行存根函数。

[0102] S305、在保存所述存根函数的运行结果后，重新将所述存根函数的函数体替换为所述占位符。

[0103] 具体的，在内存中保存存根函数的运行结果之后，清理存根函数的函数体，即使用占位符或者填充空效果的代码替换函数体中存储的解密数据，以使待保护函数的程序代码仅在运行时被加载至内存，运行完毕后删除。

[0104] 本实施例中，例如，函数保护方法的代码如下：

（1）

MyClass.cls

```
class MyClass
{
    public void run()
    {
        System.out.println("hello")
    }
    /*
    public void run1()
    {

    }
```

[0105]

```
        */
    }
（2）
    class DecryptCls
    {
        public static int decrypt()
        {

        }
    }
（3）
    class MyClass
    {
        public void run()
        {
            byte [] decode_data = decrypt(g_encode_data);
            run1();
        }
        public void run1()
        {
         //
         // 空数据
         //
        }
    }
（4）
class MyClass
{
    public void run()
    {
        byte [] decode_data = decrypt(g_encode_data);
        run1();
    }
    public void run1()
    {
        System.out.println("hello")
        //清理 run1 的内存空间
    }
```

[0106]

[0107] 输入MyClass.cls在类文件中获取待保护函数，从MyClass.cls中抽取出run函数的字节码，获取待保护函数的函数体，对run函数的函数体进行加密处理，并在类文件中生成存根函数run1，存根函数与待保护函数所占字节相同，或者存根函数所占字节数大于待保护函数所占字节数，存根函数中填充的代码仅用来占位。构造与加密函数匹配的解密函数，将解密函数复制到目标类文件中待保护函数的函数体所占的存储空间中，即：将DecryptCls这个解码类的decrypt成员函数复制到run函数中。执行待保护的函数时，在待

保护函数的函数体中获取解密函数，可以使用解密函数对加密锁中的加密信息进行解密，生成解密数据，将解密数据写入至存根函数中，即：将代码(3)中的获得的decode_data写入到run1函数，在run1函数中写入清理run1函数的代码之后，在run函数中调用run1函数，运行存根函数，运行完毕后，对存根函数进行还原处理。

[0108] 本发明实施例三提供的一种函数解密方法，在函数加密方法的基础上，从加密锁中获取加密信息并使用解密函数进行解密，将解密数据复制到存根函数中执行，执行完毕后释放存根函数的空间，能够隐藏程序代码，有效防止程序代码被反编译。

[0109] 实施例四

[0110] 图4是本发明实施例四中的一种函数加密装置的结构示意图，该装置应用于函数保护的情况，该装置可以采用软件和/或硬件的方式实现，一般集成于应用系统中。如图4所示，装置包括：函数体获取模块401、加密模块402、存根函数构造模块403以及解密函数构造模块404。

[0111] 函数体获取模块401，用于获取目标类文件中待保护函数的函数体；

[0112] 加密模块402，用于使用加密函数生成与所述函数体对应的加密信息，并将所述加密信息存储于设定存储空间中；

[0113] 存根函数构造模块403，用于构造与所述函数体匹配的存根函数，其中，所述存根函数的函数体中填充有设定占位符；

[0114] 解密函数构造模块404，用于构造与所述加密函数匹配的解密函数，并将所述目标类文件中待保护函数的函数体替换为所述解密函数。

[0115] 本发明实施例能够使待保护的函数不存在于类文件中，仅在运行过程中存在于内存中，极大地降低了代码被破解的可能性，达到了隐藏代码的目的，能够避免函数代码被反编译的情况发生，有效地提升了函数代码的安全性。

[0116] 在上述实施例的基础上，所述存根函数构造模块403包括：

[0117] 第一构造单元，用于所述存根函数的函数体与所述待保护函数的函数体长度相等；

[0118] 第二构造单元，用于所述存根函数的参数类型与所述待保护函数的参数类型相同；

[0119] 第三构造单元，用于所述存根函数的返回值类型与所述待保护函数的返回值类型相同。

[0120] 在上述实施例的基础上，所述加密模块402包括：

[0121] 调用单元，用于使用中间组件从加密锁中调用加密函数；

[0122] 加密单元，用于使用所述加密函数对待保护函数的函数体进行加密，生成与所述函数体对应的加密信息；

[0123] 存储单元，将所述加密信息存储于加密锁中；

[0124] 其中，所述加密锁包括与所述目标类文件进行通信的接口。

[0125] 在上述实施例的基础上，所述函数体获取模块401包括：

[0126] 函数获取单元，用于获取目标类文件中的待保护函数；

[0127] 地址获取单元，用于使用解析工具解析所述目标类文件的文件格式，获取所述待保护函数的起始地址和终止地址；

[0128] 函数体获取单元，用于根据所述起始地址和所述终止地址，获取所述待保护函数的函数体。

[0129] 本实施例中，通过函数体获取模块的函数体获取单元，获取目标类文件中待保护的函数，解析目标类文件的格式之后，在地址获取单元获取待保护函数在类文件中的起始地址和终止地址，在函数体获取单元按照起始地址和终止地址获取待保护函数的函数体。利用加密模块的调用单元，从加密锁中调用加密函数，在加密单元中使用加密函数对待保护函数的函数体进行加密处理，生成与函数体对应的加密信息，通过存储单元将加密信息存储于加密锁中。利用存根函数构造模块构造与函数体匹配的存根函数，通过第一构造单元构造存根函数，使存根函数的字节长度与待保护函数的函数体字节长度相等，通过第二构造单元构造存根函数，使存根函数的参数类型与待保护函数的参数类型相同，通过第三构造单元构造存根函数，使存根函数的返回值类型与待保护函数的返回值类型相同。在解密函数构造模块中构造与加密函数匹配的解密函数，并将目标类文件中待保护函数的函数体替换为解密函数，即将解密函数存储于待保护函数的函数体的原始存储空间中。

[0130] 本发明实施例使内存中不存在未加密的待保护函数的函数体，通过重新构建函数内部逻辑使待保护函数存储于加密锁中，不存在于类文件中，也仅在运行该函数时短时间存在于内存中，降低函数代码被反编译的可能性。

[0131] 实施例五

[0132] 图5是本发明实施例五中的一种函数解密装置的结构示意图，该装置应用于函数保护的情况，该装置可以采用软件和/或硬件的方式实现，一般集成于应用系统中。如图5所示，装置包括：目标函数获取模块501、解密模块502、存根函数获取模块503、运行模块504以及替换模块505。

[0133] 目标函数获取模块501，用于在检测到目标函数被运行时，在目标类文件中获取所述目标函数的函数体；

[0134] 解密模块502，用于如果识别出所述函数体中包括解密函数，则在设定存储空间中，获取与所述函数体对应的加密信息，并使用所述解密函数对所述加密信息进行解密，获取解密数据；

[0135] 存根函数获取模块503，用于获取与所述函数体匹配的存根函数，其中，所述存根函数的函数体中填充有设定占位符；

[0136] 运行模块504，用于将所述存根函数的函数体替换为所述解密数据后，运行所述存根函数；

[0137] 替换模块505，用于在保存所述存根函数的运行结果后，重新将所述存根函数的函数体替换为所述占位符。

[0138] 本发明实施例能够使待保护的函数只有在运行的过程中被加载至内存，在运行结束后从内存中删除，且函数运行时待保护函数不存在于类文件中，极大地降低了代码被破解的可能性，达到了隐藏代码的目的，能够避免函数代码被反编译的情况发生，有效地提升了函数代码的安全性。

[0139] 本实施例中，当待保护函数作为目标函数被运行时，目标函数获取模块获取目标函数的函数体，即待保护函数的函数体，识别函数体中的解密函数，通过解密模块，使用解密函数对加密锁中存储的加密信息进行解密，获取解密数据。通过解密函数在存根函数获

取模块中根据存根函数的地址获取存根函数，存根函数中填充占位符或者填充空效果的代码。将解密数据复制到存根函数之后，在运行模块中运行存根函数，在内存中保存存根函数的运行结果，通过替换模块将存根函数中存储的解密数据替换为占位符或者空效果的代码，还原存根函数的内容。

[0140] 本发明实施例五提供了一种函数解密装置，能够保护函数代码，通过构造存根函数使待保护函数仅在运行时存在于内存中，运行完毕后被释放，减少了函数代码被破解的可能性。

[0141] 本发明实施例提供的函数加密装置可执行本发明任意实施例提供的函数加密方法，具备执行方法相应的功能模块和有益效果。

[0142] 本发明实施例提供的函数解密装置可执行本发明任意实施例提供的函数解密方法，具备执行方法相应的功能模块和有益效果。

[0143] 注意，上述仅为本发明的较佳实施例及所运用技术原理。本领域技术人员会理解，本发明不限于这里所述的特定实施例，对本领域技术人员来说能够进行各种明显的变化、重新调整和替代而不会脱离本发明的保护范围。因此，虽然通过以上实施例对本发明进行了较为详细的说明，但是本发明不仅仅限于以上实施例，在不脱离本发明构思的情况下，还可以包括更多其他等效实施例，而本发明的范围由所附的权利要求范围决定。

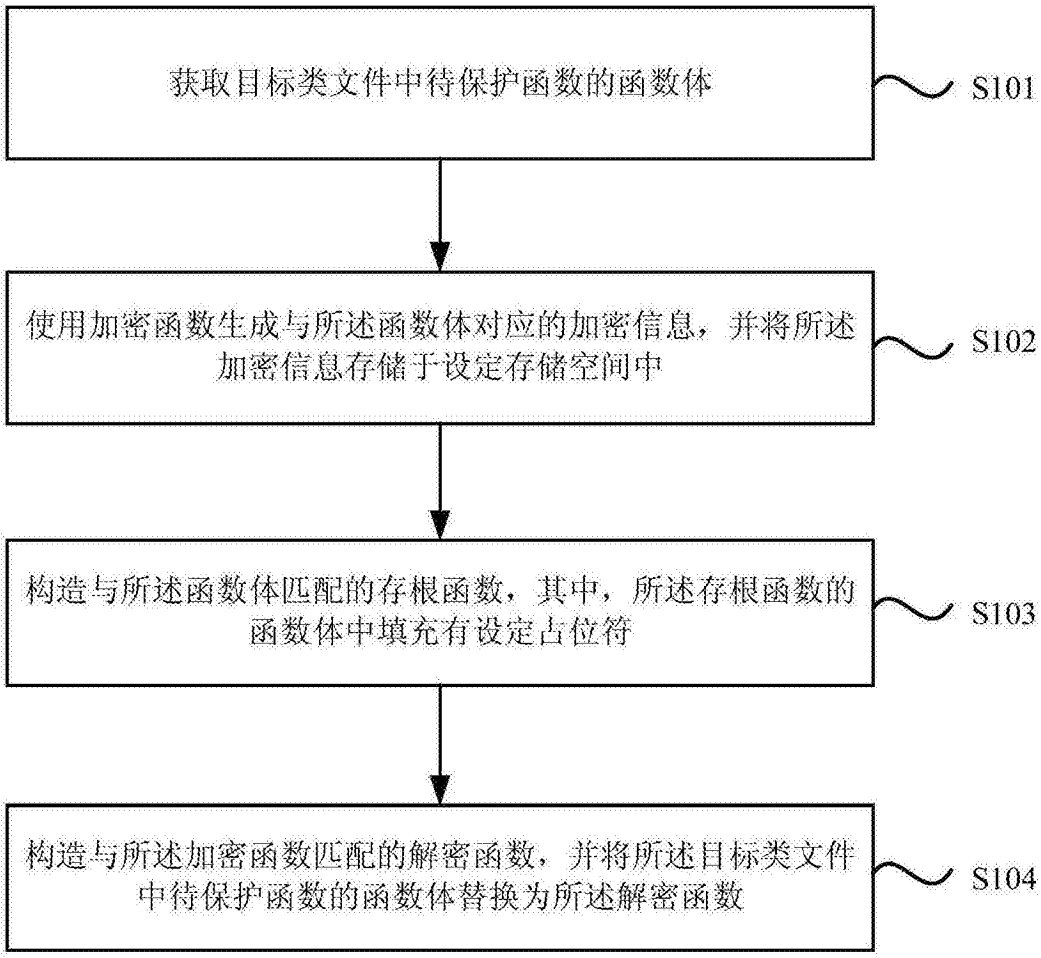

获取目标类文件中待保护函数的函数体
S101
使用加密函数生成与所述函数体对应的加密信息，并将所述加密信息存储于设定存储空间中
S102
构造与所述函数体匹配的存根函数，其中，所述存根函数的函数体中填充有设定占位符
S103
构造与所述加密函数匹配的解密函数，并将所述目标类文件中待保护函数的函数体替换为所述解密函数
S104

图1

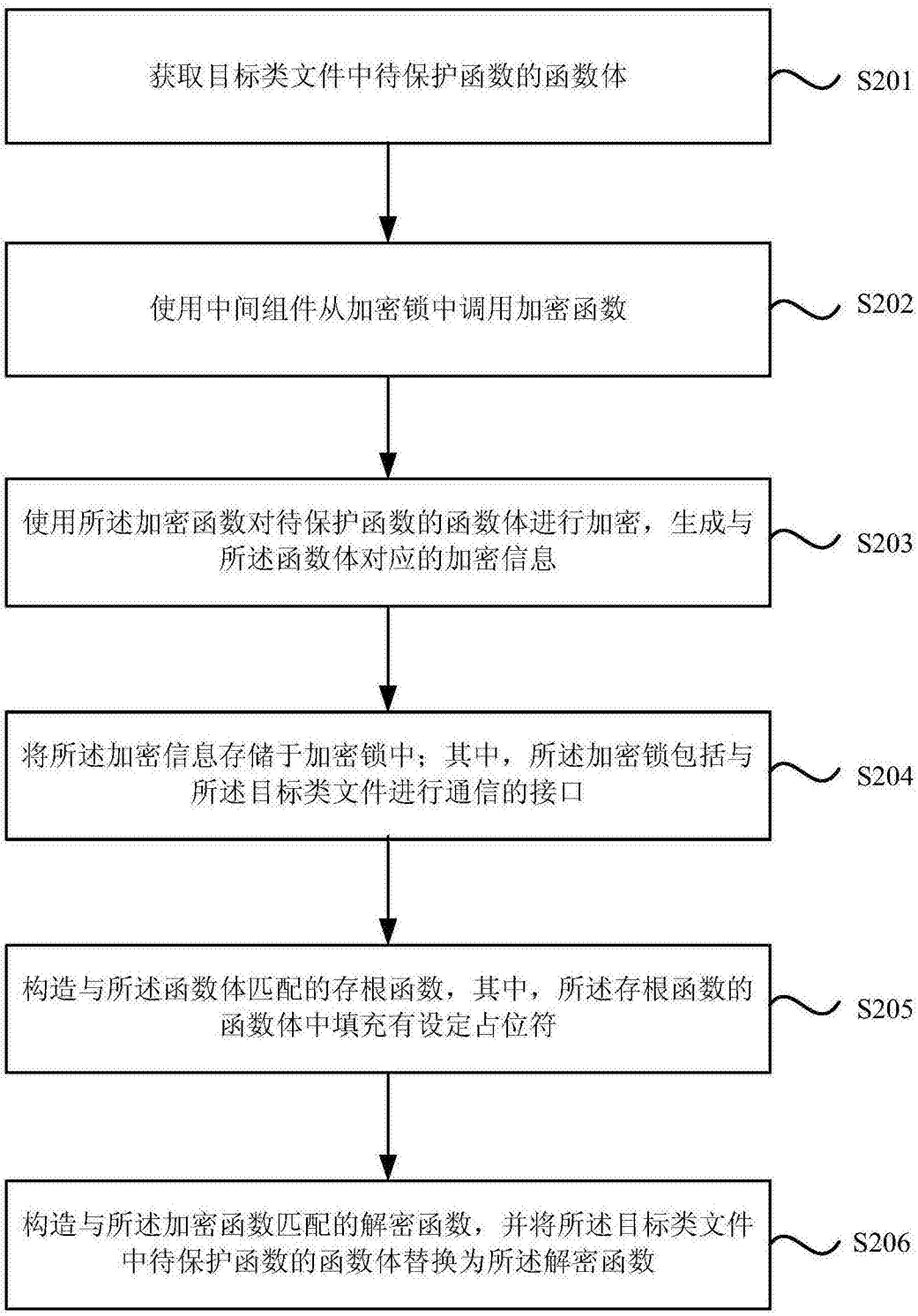

获取目标类文件中待保护函数的函数体
S201
使用中间组件从加密锁中调用加密函数
S202
使用所述加密函数对待保护函数的函数体进行加密，生成与所述函数体对应的加密信息
S203
将所述加密信息存储于加密锁中；其中，所述加密锁包括与所述目标类文件进行通信的接口
S204
构造与所述函数体匹配的存根函数，其中，所述存根函数的函数体中填充有设定占位符
S205
构造与所述加密函数匹配的解密函数，并将所述目标类文件中待保护函数的函数体替换为所述解密函数
S206

图2

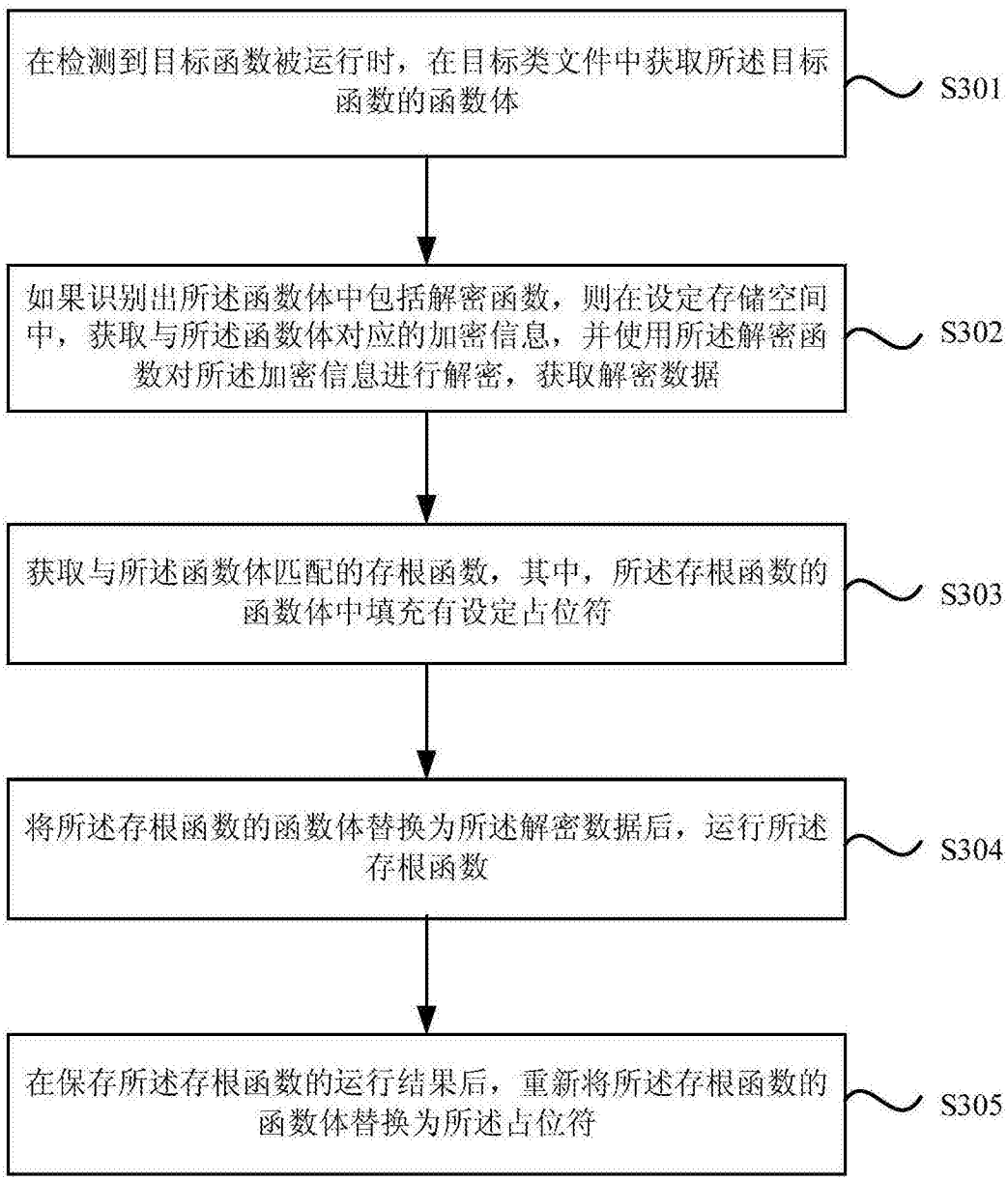

在检测到目标函数被运行时，在目标类文件中获取所述目标函数的函数体
S301
如果识别出所述函数体中包括解密函数，则在设定存储空间中，获取与所述函数体对应的加密信息，并使用所述解密函数对所述加密信息进行解密，获取解密数据
S302
获取与所述函数体匹配的存根函数，其中，所述存根函数的函数体中填充有设定占位符
S303
将所述存根函数的函数体替换为所述解密数据后，运行所述存根函数
S304
在保存所述存根函数的运行结果后，重新将所述存根函数的函数体替换为所述占位符
S305

图3

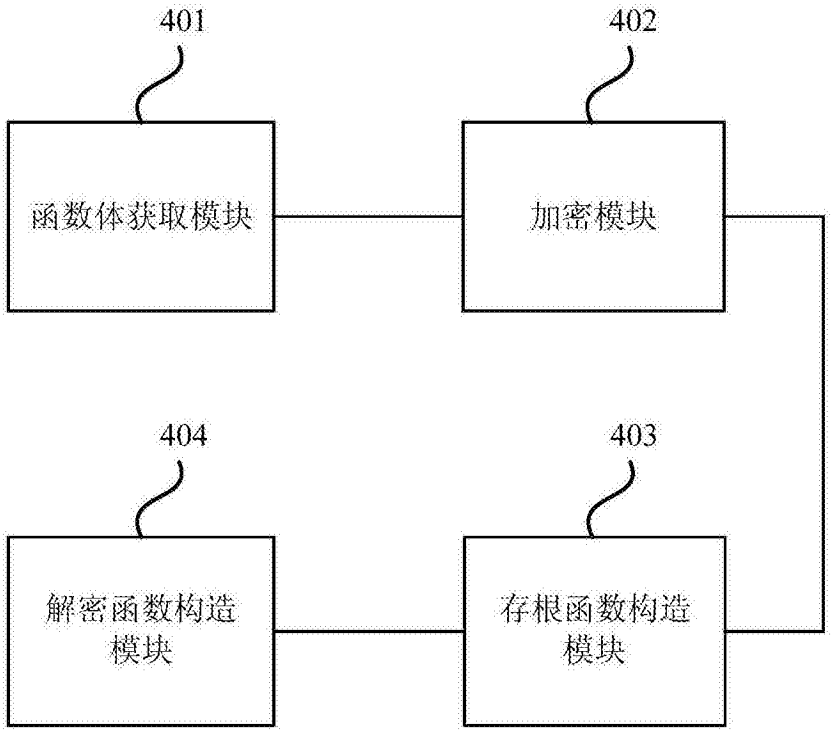

401
402
函数体获取模块
加密模块
404
403
解密函数构造模块
存根函数构造模块

图4

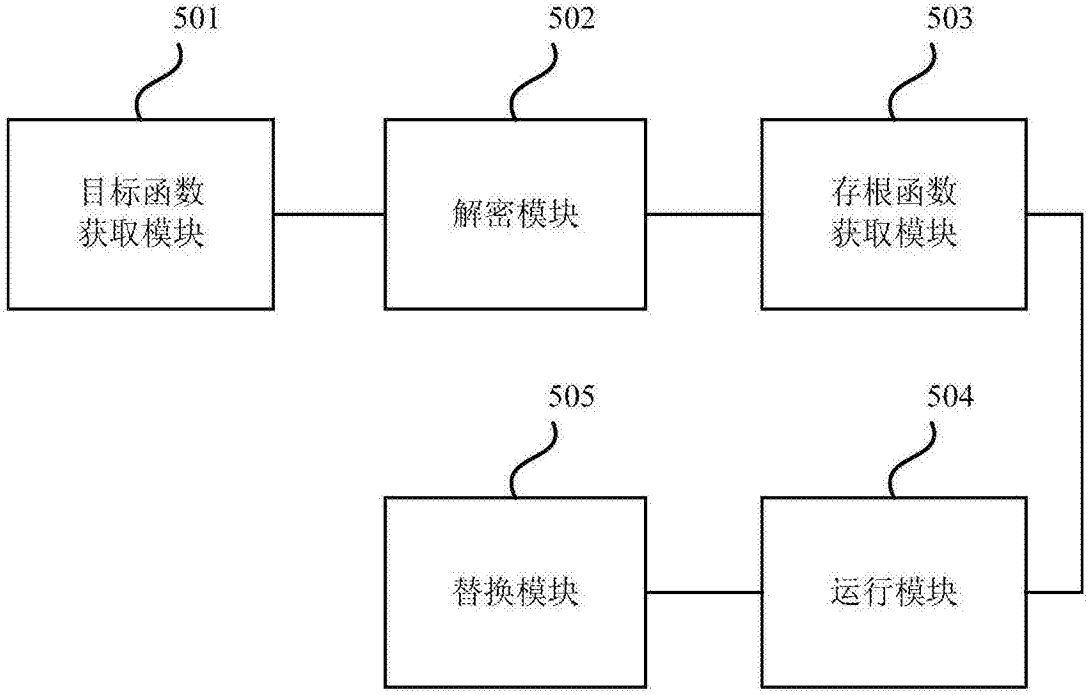

501
502
503
目标函数获取模块
解密模块
存根函数获取模块
505
504
替换模块
运行模块

图5