(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 103840935 B
(45)授权公告日 2018.01.30

(21)申请号 201310751499.5

(22)申请日 2013.12.31

(65)同一申请的已公布的文献号
申请公布号 CN 103840935 A

(43)申请公布日 2014.06.04

(73)专利权人 技嘉科技股份有限公司
地址 中国台湾新北市新店区宝强路6号

(72)发明人 林纬政

(74)专利代理机构 隆天知识产权代理有限公司 72003
代理人 章侃铱 郑特强

(51)Int.Cl.
*H04L 9/06*(2006.01)

(56)对比文件
US 8312431 B1,2012.11.13,
US 8312431 B1,2012.11.13,
CN 102812473 A,2012.12.05,
US 2003203755 A1,2003.10.30,
CN 102799815 ,2012.11.28,

审查员 马旗超

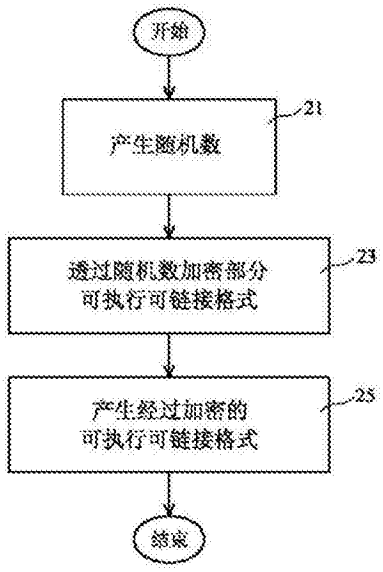
权利要求书1页 说明书6页 附图5页

(54)发明名称

开放系统之函式库的加密及解密方法

(57)摘要

本发明有关于一种开放系统之函式库的加密及解密方法，其中可执行可链接格式相依于函式库，并透过对部分可执行可链接格式的内容进行加密，使得没有密钥的操作系统无法使用经过加密的可执行可链接格式，亦无法将函式库加载内存内并执行该函式库，藉此以达到保护函式库的目的。




CN 103840935 B

1.一种开放系统之函式库的加密方法，其特征在于，其中该开放系统之函式库与一可执行可链接格式相依，且该可执行可链接格式包括一可执行可链接格式文件头、至少一程序文件头表及至少一区块，包括以下步骤：

产生一随机数；

透过该随机数加密部分该可执行可链接格式文件头及该程序文件头表；

定义一字符串密码；

对该字符串密码进行杂凑演算，并产生一杂凑数；及

透过该杂凑数对该随机数进行加密演算，并产生一经过加密的随机数。

2.如权利要求1所述的加密方法，其特征在于，其中该可执行可链接格式文件头包括一识别区块。

3.如权利要求2所述的加密方法，其特征在于，包括以下步骤：加密该识别区块以外的该可执行可链接格式文件头。

4.如权利要求3所述的加密方法，其特征在于，包括以下步骤：更改该识别区块，以产生一经过更改的识别区块。

5.如权利要求1所述的加密方法，其特征在于，其中该区块包括一动态区块。

6.如权利要求5所述的加密方法，其特征在于，包括以下步骤：加密该动态区块。

7.如权利要求1所述的加密方法，其特征在于，包括以下步骤：将该杂凑数储存于一硬件安全引擎。

8.一种开放系统之函式库的解密方法，其特征在于，包括以下步骤：

判断欲执行的程序为一经过加密的可执行可链接格式；

透过一杂凑数解密一经过加密的随机数，并产生该随机数；

透过该随机数解密该经过加密的可执行可链接格式；及

还原产生一可执行可链接格式。

9.如权利要求8所述的解密方法，其特征在于，该透过一杂凑数解密一经过加密的随机数，并产生该随机数包括以下步骤：

将该经过加密的随机数及该经过加密的可执行可链接格式传送至一硬件安全引擎；以及

透过该硬件安全引擎储存之该杂凑数解密该经过加密的随机数，并产生该随机数。

10.如权利要求8所述的解密方法，其特征在于，还包括以下步骤：

回传解密后的该可执行可链接格式。

# 开放系统之函式库的加密及解密方法

## 【技术领域】

[0001] 本发明有关于一种开放系统之函式库的加密方法，主要透过加密部分可执行可链接格式，以达到保护函式库的目的。

## 【背景技术】

[0002] 请参阅图1，为习用开放系统加载及执行函式库的步骤流程图。如图所示，在计算机科学的领域中，透过共享函式库(shared library)的使用，可有效节省应用程序所需的硬盘及内存空间。函式库是一个档案，包括许多可被外部程序使用的应用程序接口(Application Programming Interface，API)，在可在执行程序时连结使用。

[0003] 在开始执行程序前，操作系统会判断程序是否相依于其它的函式库，如步骤11所示。若操作系统判断被执行的程序不相依于其它的函式库，便直接执行该程序，而不会连结或加载函式库的数据。

[0004] 反之，若操作系统判断被执行的程序相依于其它的函式库，便会经由被执行的程序连结至其它的函式库。此时操作系统会将控制权交给程序加载器(loader)，并透过程序加载器将相依的函式库加载内存，如步骤13所示。在完成加载函式库的步骤之后，操作系统便可开始执行加载的函式库，并完成加载函式库的步骤，如步骤15所示。

[0005] 在实际应用时函式库可能是由第三方所编写，使用者可透过付费的方式取得函式库的数据。然而在实际应用时，函式库则可能被他人所窃取，且窃取者可无偿的使用函式库，如此一来将会对付费的使用者或收费端造成损失。

## 【发明内容】

[0006] 本发明之一目的，在于提供一种开放系统之函式库的加密方法，其中可执行可链接格式相依于函式库，并透过对加密部分可执行可链接格式进行加密，使得未取得密钥的操作系统或程序加载器无法经由加密的可执行可链接格式取得函式库的数据，藉此以达到保护函式库的目的。

[0007] 本发明之一目的，在于提供一种开放系统之函式库的加密方法，仅需要加密部分可执行可链接格式，便可达到保护函式库的目的。由于在加密的过程中不需要加密欲保护之函式库的所有数据，可省去对大量数据进行加密所耗费的系统资源及时间。

[0008] 本发明之一目的，在于提供一种开放系统之函式库的加密方法，主要透过随机数加密部分可执行可链接格式，并透过杂凑数加密随机数。此外可进一步将杂凑数储存在硬件安全引擎内部，藉此将可有效提高可执行可链接格式及函式库的安全性，并可降低随机数被窃取的风险。

[0009] 本发明之一目的，在于提供一种开放系统之函式库的解密方法，当操作系统或程序加载器判断欲执行的程序为经过加密的可执行可链接格式时，可进一步透过密钥(随机数)解密经过加密的可执行可链接格式，并透过可执行可链接格式连结至相依的函式库，藉此以加载及执行函式库的数据。

[0010] 为达到上述目的，本发明提供一种开放系统之函式库的加密方法，其中开放系统之函式库与一可执行可链接格式(ELF)相依，且可执行可链接格式包括一可执行可链接格式档头(ELF header)、至少一程序文件头表(Program header table)及至少一区块(Segment)，包括以下步骤：加密部分可执行可链接格式文件头及程序文件头表。

[0011] 此外，本发明还提供一种开放系统之函式库的解密方法，包括以下步骤：判断欲执行的程序为一经过加密的可执行可链接格式；透过一随机数解密经过加密的可执行可链接格式；及还原产生一可执行可链接格式。

[0012] 在本发明加密方法一实施例中，其中可执行可链接格式档头包括一识别区块。

[0013] 在本发明加密方法一实施例中，包括以下步骤：加密识别区块以外的可执行可链接格式档头。

[0014] 在本发明加密方法一实施例中，包括以下步骤：更改识别区块，以产生一经过更改的识别区块。

[0015] 在本发明加密方法一实施例中，其中区块包括一动态区块(PT_DYNAMIC)。

[0016] 在本发明加密方法一实施例中，包括以下步骤：加密动态区块(PT_DYNAMIC)。

[0017] 在本发明加密方法一实施例中，包括以下步骤：产生一随机数；及透过随机数加密部分可执行可链接格式文件头及程序文件头表。

[0018] 在本发明加密方法一实施例中，包括以下步骤：定义一字符串密码；对字符串密码进行杂凑演算，并产生一杂凑数；及透过杂凑数对随机数进行加密演算，并产生一经过加密的随机数。

[0019] 在本发明加密方法一实施例中，包括以下步骤：将杂凑数储存于一硬件安全引擎。

[0020] 在本发明解密方法一实施例中，包括以下步骤：透过一杂凑数解密一经过加密的随机数，并产生随机数。

[0021] 在本发明解密方法一实施例中，包括以下步骤：将一经过加密的随机数及经过加密的可执行可链接格式传送至一硬件安全引擎；透过硬件安全引擎储存之一杂凑数解密经过加密的随机数，并产生随机数；及回传解密后的可执行可链接格式。

## 【附图说明】

[0022] 图1为习用开放系统加载及执行函式库的步骤流程图；

[0023] 图2为本发明开放系统之函式库的加密方法一实施例的步骤流程图；

[0024] 图3为本发明可执行可链接格式一实施例的方块示意图；

[0025] 图4为本发明密钥(随机数)之加密方法一实施例的步骤流程图；

[0026] 图5为本发明系统芯片连接硬件安全引擎一实施例的方块示意图；

[0027] 图6为本发明具有硬件安全引擎之系统芯片一实施例的方块示意图；

[0028] 图7为本发明开放系统之函式库的解密方法一实施例的步骤流程图；及

[0029] 图8为本发明开放系统之函式库的解密方法又一实施例的步骤流程图。

[0030] 虽然已透过举例方式在图式中描述了本创作的具体实施方式，并在本文中对其作了详细的说明，但是本创作还允许有各种修改和替换形式。本创作之图式内容可为不等比例，图式及其详细的描述仅为特定型式的揭露，并不为本创作的限制，相反的，依据专利范围之精神和范围内进行修改、均等构件及其置换皆为本创作所涵盖的范围。

[0031] 【主要组件符号说明】：

[0032] 30 可执行可链接格式

[0033] 31 可执行可链接格式文件头

[0034] 311 识别区块

[0035] 33 程序文件头表

[0036] 35 区块

[0037] 37 分段档头表

[0038] 51 芯片系统

[0039] 53 硬件安全引擎

[0040] 531 内存

[0041] 533 运算单元

[0042] 55 硬件输入输出接口

[0043] 60 芯片系统

[0044] 61 中央处理器

[0045] 63 硬件安全引擎

[0046] 631 内存

[0047] 633 运算单元

[0048] 65 硬件输入输出接口

**【具体实施方式】**

[0049] 下面结合附图对本发明的结构原理和工作原理作具体的描述：

[0050] 请参阅图2，为本发明开放系统之函式库的加密方法一实施例的步骤流程图。开放系统之函式库(shared library)与一可执行可链接格式相依，操作系统及/或程序加载器可经由可执行可链接格式连结相依的函式库。可执行可链接格式(Extensible Linking Format，ELF)30是计算机科学中用于执行档、目的檔、共享库和核心转储的标准文件格式。此外可执行可链接格式30具有可扩展性及灵活性，并可在各种不同的操作系统或平台上使用，且不局限在特定的处理器或计算机结构。

[0051] 可执行可链接格式30包括一可执行可链接格式文件头(ELF header)31及后续的档案数据。在本发明一实施例中，请配合参阅图3所示，后续的档案数据可包括至少一程序文件头表(Program header table)33、至少一区块(Segment)35及/或至少一分段档头表(section header table)37。

[0052] 可执行可链接格式文件头31用以描述可执行可链接格式30的相关数据，例如辨别执行的程序是否为ELF档案格式、ELF的类型、ELF的平台、ELF的版本、程序的起始地址、程序文件头表33的偏移值、分段档头表37的偏移值、可执行可链接格式文件头31的长度、程序文件头表33之条目(entry)的长度及个数、分段档头表37之条目的长度及个数…等。

[0053] 程序文件头表33用以描述各个区块(segments)35，包括区块的位移位置(segments offset)、虚拟位置(virtual address)、实体位置(physical address)、区段档案大小(size in file)、区段内存大小(size in memory)，而分段档头表37则用以描述零或多个分段(section)。

[0054] 操作系统在开始执行程序前，会判断程序是否相依于其它的函式库，并依据判断的结果决定是否加载函式库的资料。例如Linux系统在执行程序前，会依据可执行可链接格式文件头31判断被执行的程序是否为可执行可链接格式30，当Linux系统判断被执行的程序为可执行可链接格式30，便会将控制权交给程序加载器(loader)，程序加载器则会依据可执行可链接格式30内容将相依的函式库加载内存。

[0055] 换言之，操作系统必须透过可执行可链接格式30的数据，才能将相依的函式库加载内存，并执行内存内加载的函式库。在本发明当中主要是对部分可执行可链接格式30的内容进行加密，使得没有密钥的操作系统或程序加载器无法使用经过加密的可执行可链接格式30将函式库加载内存内，亦无法执行该函式库，藉此以达到保护函式库的目的。

[0056] 在本发明一实施例中，可产生一密钥，例如随机数E，如步骤21所示，并透过密钥(随机数E)对部分的可执行可链接格式30进行加密，如步骤23所示。藉此以得到经过加密的可执行可链接格式30，如此一来没有密钥的操作系统便无法解开经过加密的可执行可链接格式30，亦无法加载函式库的数据，如步骤25所示。

[0057] 在本发明一实施例中，如图3所示，可执行可链接格式30包括一可执行可链接格式文件头31(ELF header)、至少一程序文件头表(Program header table)33、至少一区块(Segment)35及/或至少一分段档头表(section header table)37。

[0058] 在实际应用时可透过密钥(随机数E)加密部分可执行可链接格式文件头31及程序文件头表33，使得没有密钥的操作系统无法解开或使用经过加密的可执行可链接格式30，亦无法连结及加载相对应之函式库的数据，藉此以达到保护函式库的目的。

[0059] 在本发明一实施例中，可执行可链接格式文件头31包括一识别区块311，操作系统可透过识别区块311得知执行的档案是否为可执行可链接格式30，例如识别区块311可为魔数(magic number)。在加密可执行可链接格式文件头31时，可透过密钥(随机数E)加密识别区块311以外的可执行可链接格式文件头31，使得操作系统仍旧可透过识别区块311辨识可执行可链接格式30。

[0060] 在本发明一实施例中，亦可更改识别区块311，并产生一经过更改的识别区块，例如将ELF更改为SLF。操作系统及/或程序加载器在读取经过更改的识别区块311时，将可得知被读取的档案为经过加密的可执行可链接格式30，并可使用密钥(随机数E)解密经过加密的可执行可链接格式30，例如透过密钥(随机数E)解密经过加密的可执行可链接格式文件头31及/或程序文件头表33。

[0061] 在本发明另一实施例中，区块(Segment)35包括一动态区块(PT_DYNAMIC)，其中动态区块包括一些动态连结所需的信息。在实际应用时可同时加密可执行可链接格式30中部分的可执行可链接格式文件头31、程序文件头表33及动态区块，以进一步提高函式库的安全层级。

[0062] 请参阅图4，为本发明密钥(随机数)之加密方法一实施例的步骤流程图。在本发明图2所示之实施例中，主要透过密钥(随机数E)对部分的可执行可链接格式30进行加密，藉此以保护函式库之目的，并可防止没有密钥(随机数E)的使用者或操作系统透过可执行可链接格式30加载及使用函式库的数据。在本发明实施例中进一步对密钥(随机数E)进行加密，以提高对函式库的保密层级。

[0063] 在对密钥(随机数E)进行加密的过程中，首先会定义一字符串密码S，在本发明一

实施例中，定义字符串密码S的动作仅需要进行一次，如步骤41所示。而后对定义的字符串密码S进行杂凑演算(Secure Hash Algorithm，SHA)，以得到杂凑数M，并可进一步储存杂凑数M，例如可透过SHA-256对字符串密码S进行演算，以得到32位的杂凑数M，如步骤43所示。

[0064] 透过杂凑数M的取得，可进一步使用杂凑数M对密钥进行加密演算，例如可透过杂凑数M对随机数E进行加密演算，产生经过加密的随机数e，如步骤45所示。而后再储存经过加密的随机数e，例如可将经过加密的随机数e编译在动态加载器，如步骤47所示。

[0065] 在本发明实施例中，除了透过随机数E加密部分的可执行可链接格式30之外，还透过杂凑数M将随机数E加密成为经过加密的随机数e，因此即便第三者取得经过加密的随机数e，亦无法透过经过加密的随机数e解密经过加密的可执行可链接格式30，以进一步提高可执行可链接格式30及/或相依之函式库的安全性。

[0066] 在本发明一实施例中，如图5所示，芯片系统51可透过硬件输入输出接口(I/O接口)55连接硬件安全引擎53，并透过硬件安全引擎53管理及/或储存杂凑数M。在本发明另一实施例中，如图6所示，可将硬件安全引擎63整合在芯片系统60内部，其中硬件安全引擎63透过硬件输入输出接口(I/O接口)65连接芯片系统60内部的中央处理器(CPU)61，并可透过硬件安全引擎63管理及/或储存杂凑数M。

[0067] 透过硬件安全引擎53/63的使用，可再是提高密钥(随机数)、可执行可链接格式30及/或函式库的安全性，并可有效降低密钥(随机数)被窃取的风险。在本发明一实施例中，硬件安全引擎53/63包括一内存531/631及一运算单元533/633，在实际应用时可将杂凑数M储存在硬件安全引擎53/63内部，例如储存在内存531/631内，并将经过加密的随机数e编译(compile)在程序加载器。在解密的过程中，可将经过加密的随机数e及/或经过加密的可执行可链接格式30传送至硬件安全引擎53/63，并以硬件安全引擎53/63储存之杂凑数M解密经过加密的随机数e及/或经过加密的可执行可链接格式30，详细的解密方法将会在下面的实施例进行说明。

[0068] 请参阅图7，为本发明开放系统之函式库的解密方法一实施例的步骤流程图。如图所示，请配合参阅图3所示，在进行解密的过程中，操作系统及/或程序加载器可由可执行可链接格式30的密识别区块311，例如魔数(magic number)，得知即将执行的程序为经过加密的可执行可链接格式30，例如操作系统及/或程序加载器可由可执行可链接格式30的SLF，得知被执行的程序为经过加密的可执行可链接格式30，如步骤71所示。

[0069] 当操作系统及/或程序加载器确定可执行可链接格式30是经过加密的档案，便可进一步透过随机数E解密经过加密的可执行可链接格式30，如步骤73所示，经过解密之后将会还原产生可执行可链接格式30，如步骤75所示。操作系统及/或程序加载器将可透过解密的可执行可链接格式30连结至相依的函式库，并将相依于可执行可链接格式30的函式库加载内存。

[0070] 请参阅图8，为本发明开放系统之函式库的解密方法又一实施例的步骤流程图。如图所示，请配合参阅图3、图5及图6所示，其中可执行可链接格式30是透过随机数E进行加密，并进一步透过杂凑数M加密随机数E，以产生经过加密的随机数e。此外将加密随机数E的杂凑数M储存在硬件安全引擎53/63，藉此进一步提高随机数E、可执行可链接格式30及/或函式库的安全性，并可有效降低因随机数E被窃取，而导致函式库被盗用的机率。

[0071] 在进行解密的过程中，操作系统及/或程序加载器可由可执行可链接格式30的密

识别区块311，例如魔数(magic number)，得知即将执行的程序为经过加密的可执行可链接格式30，例如操作系统及/或程序加载器可由可执行可链接格式30的SLF，得知被执行的程序为经过加密的可执行可链接格式30，如步骤81所示。

[0072] 当操作系统及/或程序加载器确定可执行可链接格式30是经过加密的档案，可透过硬件输入输出接口55/65将经过加密的随机数e及经过加密的可执行可链接格式30传送至硬件安全引擎53/63，如步骤83所示。硬件安全引擎53/63会透过储存的杂凑数M解密经过加密的随机数e，并还原产生随机数E，如步骤85所示。

[0073] 在产生随机数E之后，可进一步透过随机数E解密经过加密的可执行可链接格式30，并还原产生可执行可链接格式30，如步骤87所示。而后再经由硬件输入输出接口55/65，将经过解密的可执行可链接格式30回传至程序加载器，如步骤89所示。藉此程序加载器便可透过可执行可链接格式30，将相依的函式库加载内存，并执行内存内加载的函式库。

[0074] 在本发明中所述之连接指的是一个或多个物体或构件之间的直接连接或者是间接连接，例如可在一个或多个物体或构件之间存在有一个或多个中间连接物。

[0075] 说明书之系统中所描述之也许、必须及变化等字眼并非本发明之限制。说明书所使用的专业术语主要用以进行特定实施例的描述，并不为本发明的限制。说明书所使用的单数量词(如一个及该个)亦可为复数个，除非在说明书的内容有明确的说明。例如说明书所提及之一个装置可包括有两个或两个以上之装置的结合，而说明书所提之一物质则可包括有多种物质的混合。

[0076] 以上所述者，仅为本发明之较佳实施例而已，并非用来限定本发明实施之范围，即凡依本发明申请专利范围所述之形状、构造、特征及精神所为之均等变化与修饰，均应包括于本发明之申请专利范围内。

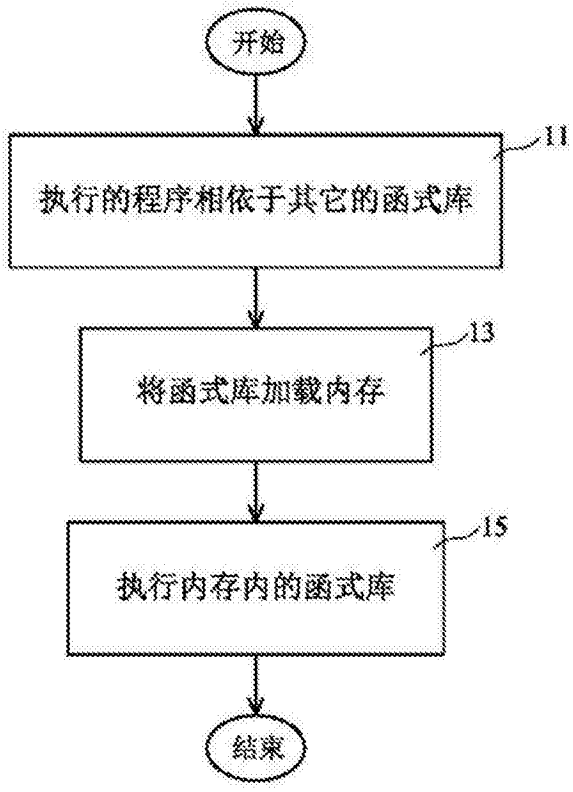

开始
11
执行的程序相依于其它的函式库
13
将函式库加载内存
15
执行内存内的函式库
结束

图1

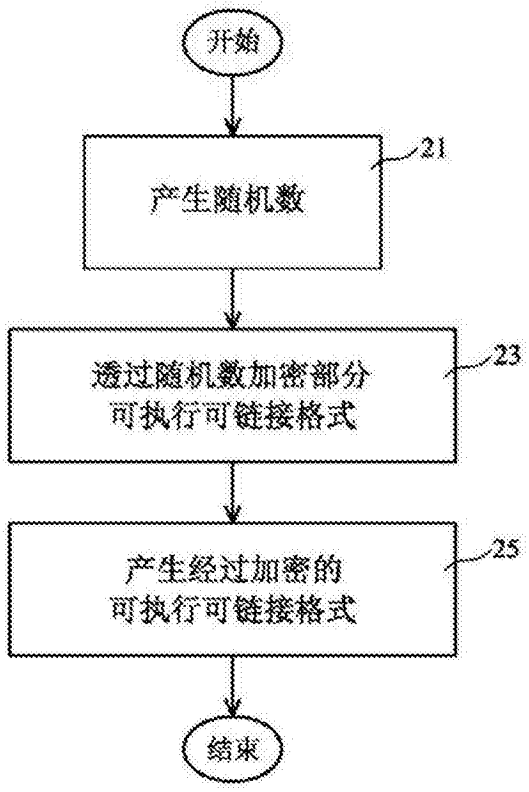

开始
21
产生随机数
23
透过随机数加密部分
可执行可链接格式
25
产生经过加密的
可执行可链接格式
结束

图2

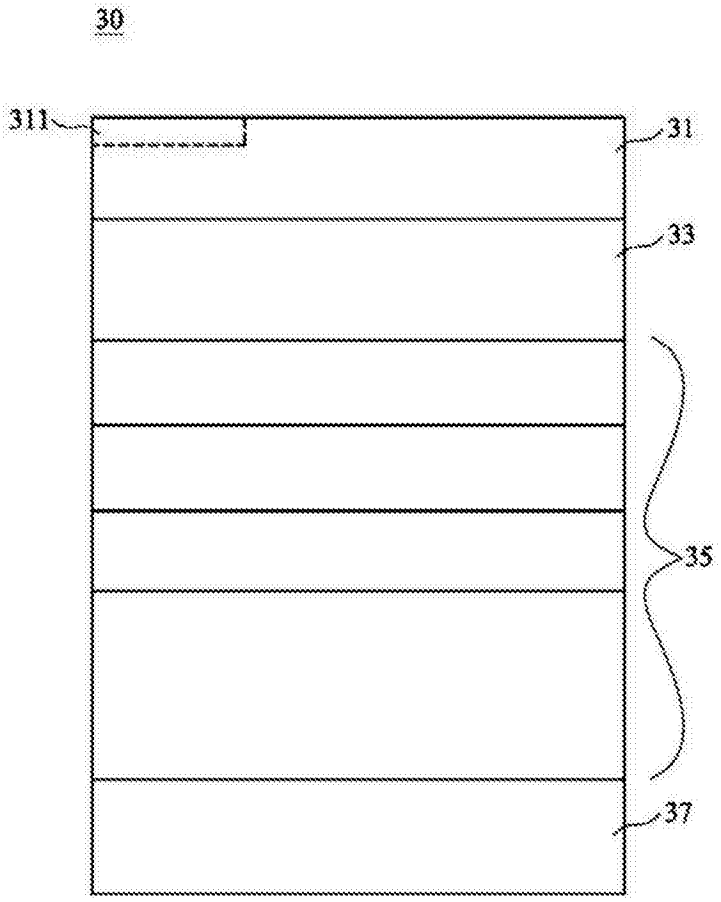



图3

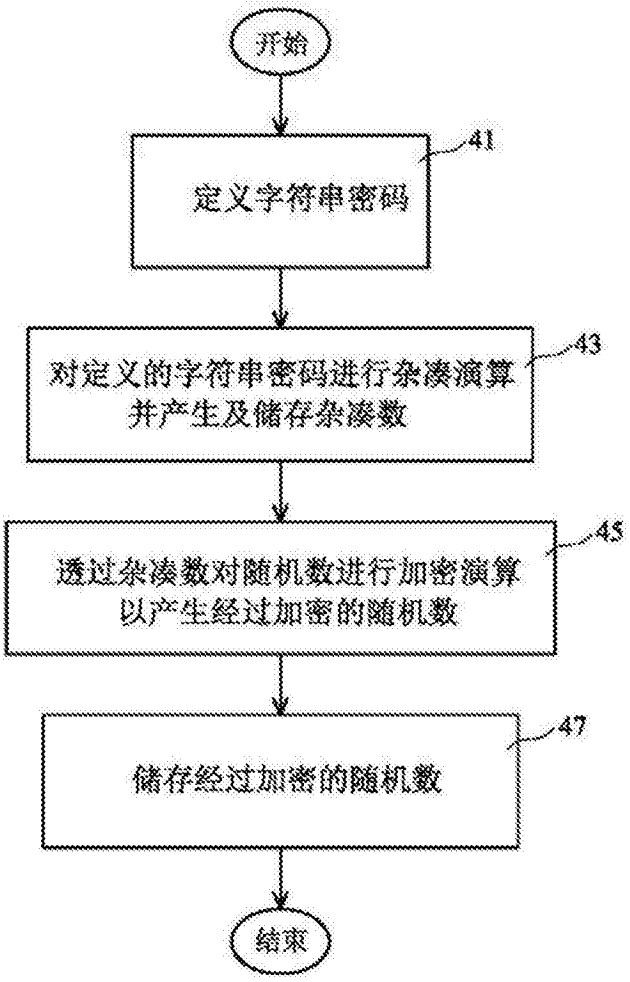

开始
41
定义字符串密码
43
对定义的字符串密码进行杂凑演算
并产生及储存杂凑数
45
透过杂凑数对随机数进行加密演算
以产生经过加密的随机数
47
储存经过加密的随机数
结束

图4

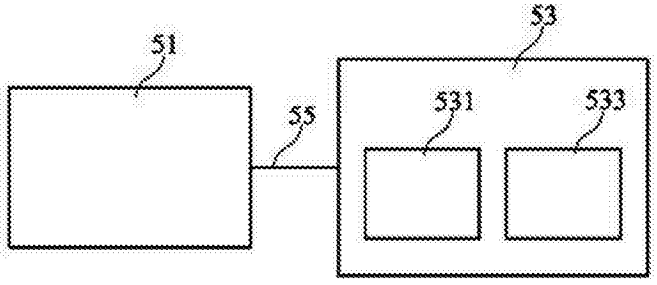

51
53
55
531
533

图5

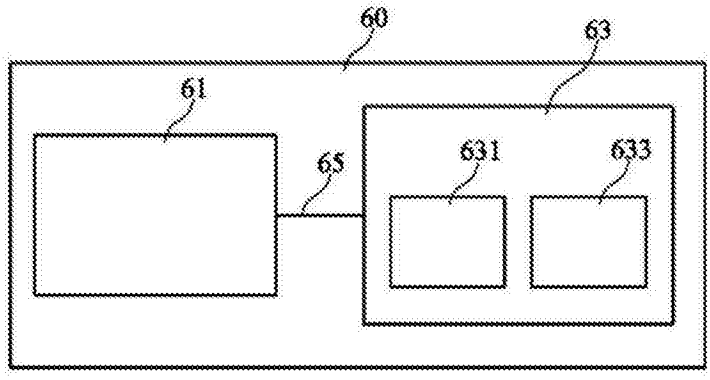

60
63
61
65
631
633

图6

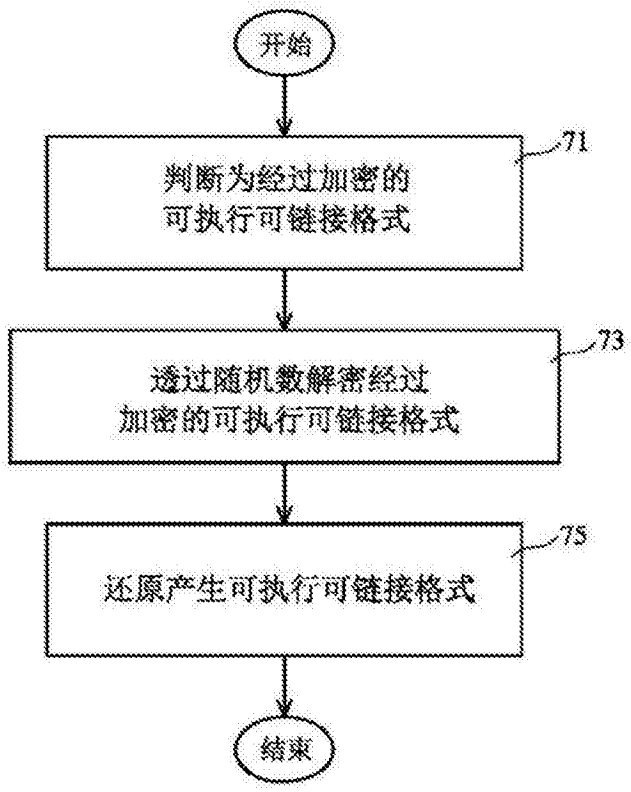

开始
判断为经过加密的
可执行可链接格式
71
通过随机数解密经过
加密的可执行可链接格式
73
还原产生可执行可链接格式
75
结束

图7

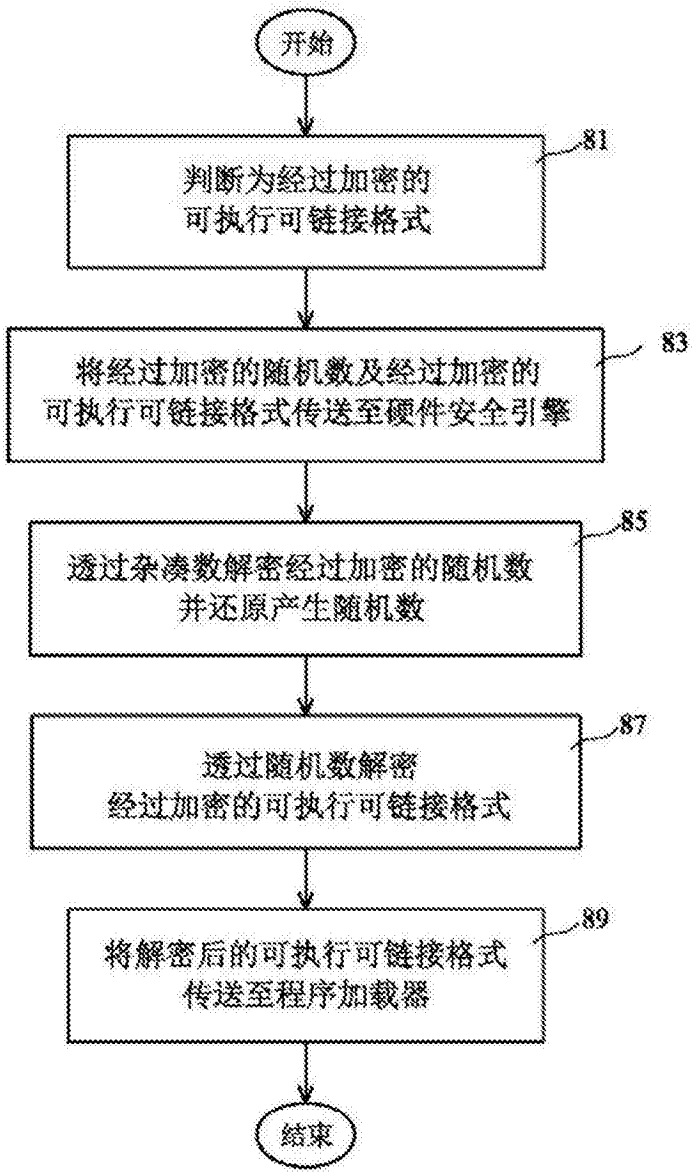

开始
判断为经过加密的
可执行可链接格式
81
将经过加密的随机数及经过加密的
可执行可链接格式传送至硬件安全引擎
83
透过杂凑数解密经过加密的随机数
并还原产生随机数
85
透过随机数解密
经过加密的可执行可链接格式
87
将解密后的可执行可链接格式
传送至程序加载器
89
结束

图8