[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 11/00
G06F 11/28

[12] 发明专利说明书

[21] ZL 专利号 00813113.9

[45] 授权公告日 2004 年 3 月 31 日

[11] 授权公告号 CN 1144126C

[22] 申请日 2000.9.18 [21] 申请号 00813113.9
[30] 优先权
[32] 1999. 9.20 [33] DE [31] 19944991.0
[86] 国际申请 PCT/EP00/09131 2000.9.18
[87] 国际公布 WO01/22223 德 2001.3.29
[85] 进入国家阶段日期 2002.3.20
[71] 专利权人 德国捷德有限公司
地址 德国慕尼黑
[72] 发明人 迈克尔·巴尔迪希韦勒
审查员 李 菲

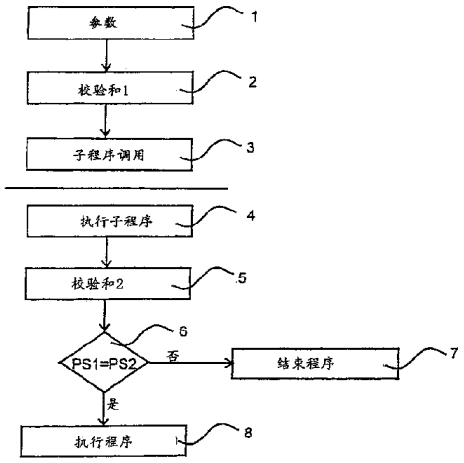
[74] 专利代理机构 北京市柳沈律师事务所
代理人 黄小临 王志森

权利要求书 1 页 说明书 4 页 附图 2 页

[54] 发明名称 保护程序流的方法

[57] 摘要

本发明涉及一种在子程序调用时保护程序流的方法。 已知的数据保护方法能够抵御通过特殊中断程序来获得数据的方法，却不能有效地保护模块化程序，特别是对子程序调用。 根据本发明，在程序执行前或执行过程中，被调用程序实施对从调用程序直接或间接传递的数据进行检查。



ISSN1008-4274

1. 一种在子程序调用时保护程序运行的方法，在程序执行前或执行过程中，被调用程序实施对从调用程序直接或间接传递的数据进行检查，其特征在于，

- 所述调用程序生成要传递的参数的第一校验和（步骤 2），

- 所述第一校验和存储在特殊提供的存储区域内，

- 所述被调用程序在其执行前，对接收到的参数生成第二校验和（步骤 5），并检查其是否与第一校验和相等（步骤 6），以及

- 在第一和第二校验和不相等的情况下，该程序被终止（步骤 7），或输出一个错误消息。

2. 如权利要求 1 所述的方法，其中在程序执行前或执行过程中，被调用程序实施对从调用程序直接或间接传递的数据进行检查，其特征在于，在调用子程序时，启动一个定时器（步骤 22），该定时器对执行程序所需的时钟周期数计数，并且如果在子程序终止之前超过预设的时钟周期数，则终止该程序（步骤 26）。

3. 如权利要求 1 所述的方法，其特征在于，用于存储校验和的存储区域是 RAM 或寄存器区域。

4. 如权利要求 2 至 3 中任一项所述的方法，其特征在于，在一些预定的点读该定时器值（步骤 24），并将其与同样预设的中间值进行比较（步骤 25），如果超过所述预设的中间值，则程序终止（步骤 26）。

5. 如权利要求 1 至 3 中任一项所述的方法，其特征在于，调用函数的返回地址被输入到一个表中，并且所述被调用程序通过基于该表检查所述返回地址的存在，检查由该调用程序报告的该返回地址（步骤 13）。

6. 如权利要求 4 所述的方法，其特征在于，在一些预定的点读该定时器值（步骤 24），并将其与同样预设的中间值进行比较（步骤 25），如果超过所述预设的中间值，则程序终止（步骤 26）。

保护程序流的方法

本发明涉及一种保护程序运行的方法。

特别是在与安全相关的应用中，例如在 IC 卡应用领域，有必要防止对程序运行的未授权的操纵。为了保护保密数据，如保密关键字数据，已知将要保护的数据以加密的方式存储，以防被未经授权的人读出。

但对保密数据的访问也可以这样实现：有选择地中断程序的运行导致加密程序出错，由此，在重复选择中断后，可以推断出保密数据。

为了避免这样的入侵，须要可靠地识别是程序运行的错误还是对其的骚扰。在德国专利 DE 37 09 524 C2 中公开了一种检查计算机中程序存储器的存储单元内容的方法。其中，存储若干校验和，这些校验和是由不同地址和不同数据存储区域的存储单元的内容形成的。所述校验和在计算机操作开始时和/或在操作中被确定，并与已存储的校验和相比较。当确定有差异时，输出一个错误信号。

由 DE 37 09 524 C2 所知的方法主要适用于检查程序中使用的数据的正确性。没有考虑程序运行的操作也会发生在特别是程序调用时，即执行子程序或函数程序时的事实。

在美国专利 US-PS5，715，389 中，公开了一种计算机系统的在线监控系统，其中，程序运行所需的时间被确定并与预设的时间进行比较。如果被监控的程序的执行时间长于预期的时间，则认为是发生了功能错误或缺乏处理效率。

因此本发明要解决的问题是，提出一种方法，能够对模块结构的程序进行可靠的检查，特别是在子程序调用时。

根据本发明，通过被调用的程序实施数据检查解决了这个问题，该数据检查确认从调用程序传输过来的数据的可靠传送。

本发明取得了附加的安全性，不仅能保证每个程序部分可靠并完整地执行，而且还能保证整个程序的运行不被干扰并免于被操纵。

本发明的一种优选实施方式提供了，调用程序首先生成要从调用程序传递到被调用程序的参数的校验和，所述校验和被存储在为其特别提供的存储

区域内。在参数被传递之后，被调用程序也对接收到的参数生成一个校验和。如果调用程序和被调用程序所生成的校验和不一致，则程序被终止。

以这种方式可以保证，函数程序，特别是执行与安全相关的数据的函数程序，在开始时就已经对操作进行了检查，因此可以从开始就避免用错误参

数启动被调用程序，并且不允许对错误数据的分析。

用于存储校验和的存储区域最好建立在 RAM 中或寄存器区域内。

由对返回地址的检查产生了形成要传递的参数的校验和的另一个实施方式。调用函数的返回地址被输入一个表，被调用程序可以借助于该表检查调用程序传输的返回地址是否在表内。在错误报告返回地址的情况下，可以中断该程序。

在调用子程序或函数程序时，通过启动一个定时器，可以实现其他的或附加的安全检查。所述定时器对执行该程序所需的时钟周期计数。常规子程序运行所需的时钟周期数首先被预设为该定时器的极限值。如果在子程序结束之前超过预设的时钟周期数，则该程序终止。

优点还在于，在子程序的预定点，读所述定时器值，并将其与同样也是预设的中间值进行比较。在这种情况下同样，如果超过预设的中间值，该程序被终止。

下面将参照图 1 至图 3 对本发明作更详细的描述，其中：

图 1 为利用校验和进行检查的流程图，

图 2 为利用返回地址表进行检查的流程图，

图 3 为利用定时器进行检查的流程图。

图 1 描述子程序调用，特别是函数调用的运行，功能步骤 1 至 3 涉及被调用程序，而功能步骤 4 至 8 涉及子程序的评价。

在被调用程序中，在步骤 1 首先提供执行子程序所需的参数，在步骤 2 形成对所述参数的校验和，在最简单的情况下是奇偶校验。当然也可以使用形成校验和的常规方法，例如，CRC（循环冗余校验），或 EDC（错误检测码）。由此确定的校验和被写入特殊的存储区域。所述存储区域可为易失性存储器（RAM）或非易失性、可重写存储器（如 EEPROM）。

在生成并存储校验和 1 之后，在步骤 3 调用子程序。在步骤 4 开始执行该子程序。在所述子程序中，首先对传递的参数生成校验和 2。所述校验和是以与在调用程序中确定校验和 1 相同的方法生成的。

下一步，在步骤 6 检查校验和 PS1 和 PS2 是否相等。如果在步骤 6 确认该两个校验和不等，则可假设在传递程序参数的过程中发生了错误，这可能表示意在确定保密数据的有意干扰。作为一种措施，可以在步骤 7 结束该程序，或采取相应的其他措施，例如向主程序发出错误消息。

如果在步骤 6 确认校验和 PS1 和 PS2 相等，则开始函数的实际执行。

图 2 所示为通过检查返回地址保护程序的一种可能性。在函数调用时，返回地址被存入硬件堆栈。在本例中，在步骤 11 子程序调用时，该信息同样地从调用程序（如返回地址）传给子程序。根据本发明，在表 17 中管理该返回地址，并且在执行该子程序时，只要该返回地址存储在 RAM 中，在步骤 12 将首先检查其一致性，并在步骤 13 基于表 17 对其进行检查。如果在步骤 14 确认出所传递的返回地址不在该表中，则程序在步骤 15 结束。否则，在步骤 16 开始执行该函数程序。

图 3 示出一种实施方式，其中，借助于定时器检查正确的或未被干扰的程序运行。直接在步骤 21 启动子程序之后，在步骤 22 启动定时器。所述定时器被设计用来对执行该子程序所需的时间进行测量，或对时钟周期进行计数。在步骤 22 启动定时器之后，在步骤 23 执行该子程序函数，在该函数结束后，在步骤 24 停止该定时器。在步骤 25 检查执行该函数程序所需的时钟周期数是否与预设时钟周期数相匹配。如果不匹配，则该程序在步骤 26 结束。否则该程序例如通过跳回到主程序在步骤 27 继续执行。

图 3 示出在函数或函数程序执行之后停止并检查定时器。在实践中，通过在函数程序中提供附加地检查定时器的某些点，可以提高安全性。这或许能够防止函数程序在有错误或入侵的情况下仍被大规模执行。

作为另一种选择，还可以在定时器启动后，连续地将定时器值与极限值相比较，如果达到或超过该极限值，则程序终止。

图 1 至图 3 所示的每个例子是作为独立的、不同的方法示出的。将这些例子加以组合，可以提高安全性。同时使用校验和检查，返回地址检查和定时器检查可以获得最大的安全性。

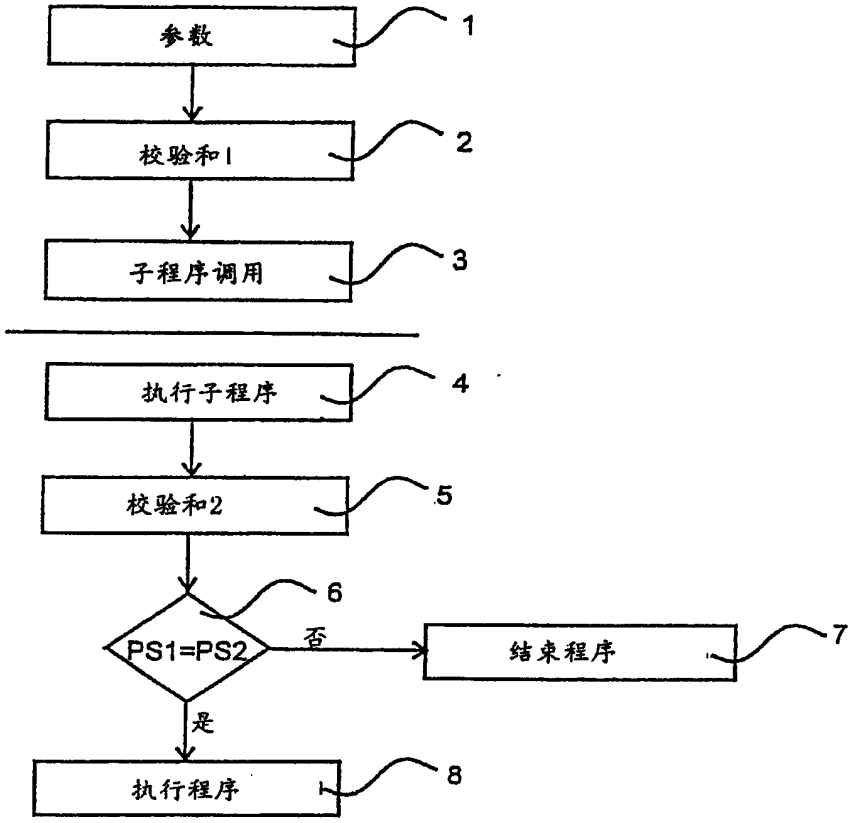

参数
1
校验和1
2
子程序调用
3
执行子程序
4
校验和2
5
6
PS1=PS2
否
结束程序
7
是
执行程序
8

图 1

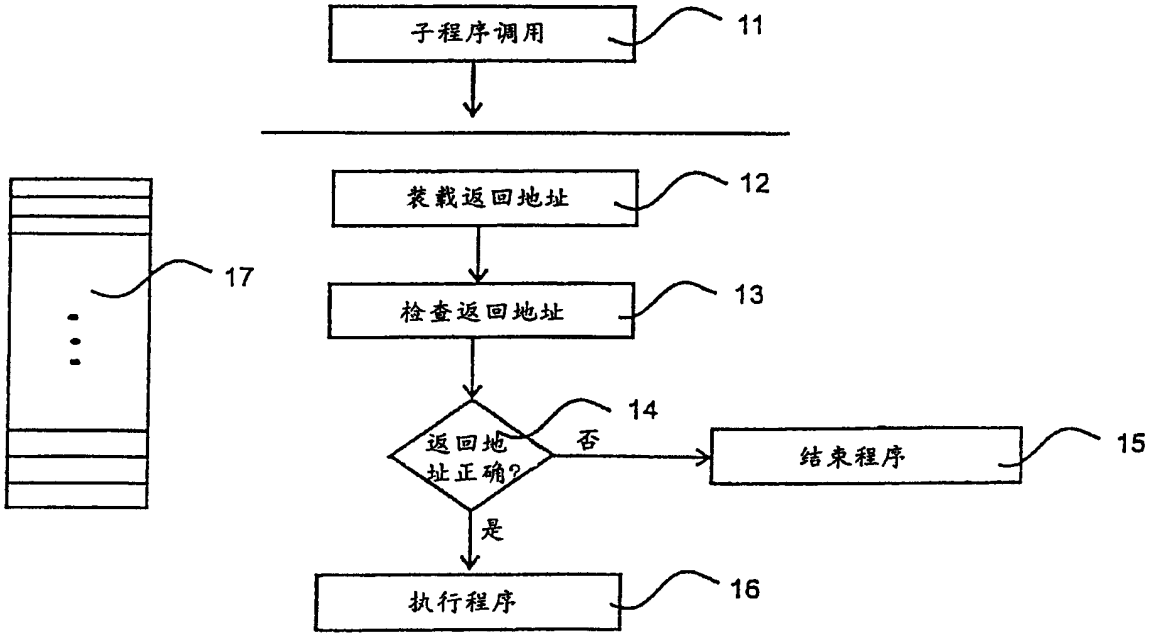

子程序调用
11
装载返回地址
12
检查返回地址
13
14
返回地址正确?
否
结束程序
15
是
执行程序
16
17

图 2

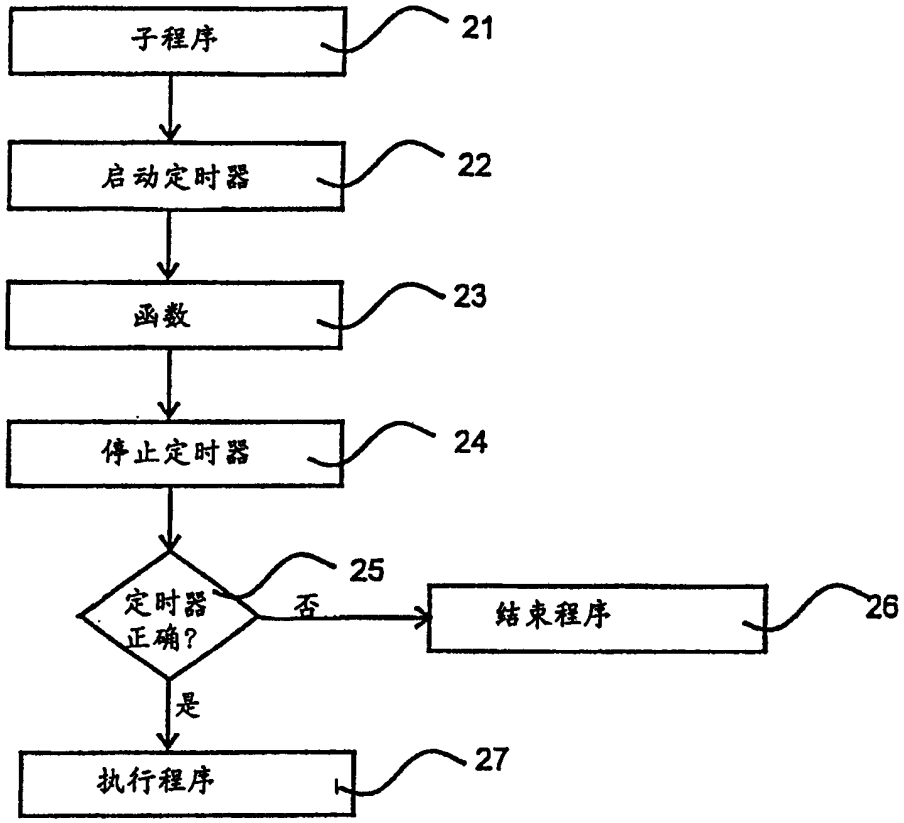


图 3