[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 15/00

[12] 发明专利说明书

[21] ZL 专利号 96121380.9

[43] 授权公告日 2003 年 3 月 26 日

[11] 授权公告号 CN 1103968C

[22] 申请日 1996.12.8 [21] 申请号 96121380.9

[30] 优先权

[32] 1995.12.8 [33] US [31] 569397

[71] 专利权人 太阳微系统有限公司

地址 美国加利福尼亚州

[72] 发明人 查尔斯·E·麦克梅尼斯

弗兰克·耶林

[56] 参考文献

EP0328232 1989.08.16 H04L9/00

EP0464526 1992.01.08 G06F9/45

US5359659 1994.10.25 H04L9/00

US5432937 1995.07.11 G06F9/44

审查员 刘春霞

[74] 专利代理机构 北京市柳沈律师事务所

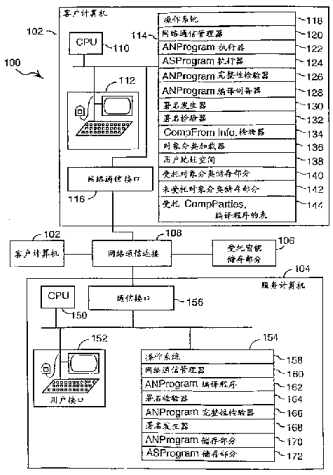
代理人 杨 梧 张玉红

权利要求书 5 页 说明书 24 页 附图 3 页

[54] 发明名称 受托编译中性结构程序版本产生特定结构的系统和方法

[57] 摘要

一个分布计算机系统，具有一程序编译计算机，它由一编译方操作，并包括一个编译程序，当检验了一中性结构程序的始发方数字署名时，编译程序通过(A)将中性结构程序码编译成特定结构程序码，和(B)将编译方数字署名加到特定结构程序码以产生一特定结构程序，该系统还具有一程序执行计算机，它由一执行方运行，并包括一特定结构程序执行器，当验证 3 中性结构程序的始发方数字署名、特定结构程序编译方数字署名和编译方是规定的受托编译方时，该特定结构程序执行器执行特定结构程序码。



ISSN1008-4274

知识产权出版社出版

1、一种计算机网络，包括

由一个编译方操作的程序编译计算机，该程序编译计算机接收一个由始发方产生的中性结构程序，该中性结构程序包含有中性结构程序码和所述始发方的数字署名，当其被检验时检验由所述始发方所标记的中性结构程序，所述程序编译计算机包括:

一个署名检验器，用于检验所述始发方的数字署名;

一个编译单元，用于当所述始发方的数字署名被检验完毕时，产生特定结构程序，所述编译单元，通过下述方式产生所述特定结构程序，所述方式是: (A)按照特定结构语言将所述中性结构程序码编译成特定结构程序码，和(B)附加一个编译方的数字署名，当其被检验时，检验由所述编译方产生的特定结构程序;

署名发生器，用于产生编译方的数字署名; 和

由一个执行方操作的程序执行计算机，该程序执行计算机接收特定结构程序，和接收或始发该中性结构程序，所述程序执行计算机包括:

一个署名检验器，用于检验所述编译方的数字署名;

一个执行器，用于执行按所述特定结构语言写成的程序码，当所述编译方的署名已被检验，并且所述的编译方是指定设定的受托编译方时，所述执行器执行特定结构程序码。

2、如权利要求1所述的计算机网络，其中:

所述署名发生器产生编译单元的数字署名，当其被检验时，验证该特定结构程序是用所述编译单元产生的;

所述编译单元还通过将所述编译单元的数字署名附加到所述特定结构程序码上产生所述特定结构程序;

该执行计算机的署名检验器检验所述计算机的数字署名;

所述执行器仅在所述编译单元的数字署名已经被验证和该编译单元被确定是一指定设定的受托编译单元之后才执行所述特定结构程序码。

3、如权利要求1所述的计算机网络，其中:

对于始发方和编译方来讲，所述的网络包括相应的专用和公用加密密

钥和相应的散列函数;

所述始发方的数字署名包括通过在中性结构程序基础上执行对应于始发方的散列函数产生的中性结构程序的消息摘要，所述中性结构程序的消息摘要利用对应于该始发方的专用密钥加密;

所述程序编译计算机的署名检验器包括通过下述方式用于检验所述始发方数字署名的多个指令，所述方式是: (A)利用所述始发方的公用加密密钥去解密所述中性结构程序的消息摘要，(B)通过在所述中性结构程序码的基础上执行所述始发方的散列函数来产生所述中性结构程序对应的测试消息摘要，和(C)比较所述中性结构程序的解密消息摘要和测试消息摘要;

所述署名发生器包括用于以如下方式产生编译方数字署名的多个指令，所述方式是: (A)通过在所述特定结构程序码的基础上执行所述编译方的相应散列函数产生中性结构程序的消息摘要，和(B)利用所述编译方的相应专用密钥加密所述特定结构程序的消息摘要; 和

所述程序执行计算机的署名检验器包括以如下方式检验所述编译方数字署名的多个指令，所述方式是: (A)利用所述编译方的公用加密密钥去解密所述特定结构程序的消息摘要，(B)通过在所述特定结构程序码的基础上执行所述编译方的散列函数产生所述特定结构程序的相应测试消息摘要，和(C)比较所述特定结构程序的解密消息摘要和测试消息摘要。

4、如权利要求1所述的计算机网络，其中:

所述程序执行计算机还包括一个中性结构程序完整性检验器，通过检验所述中性结构程序码满足预定程序完整性标准来检验所述中性结构程序码的完整性;

只有在所述中性结构程序码的完整性被验证以后，所述执行器才执行所述特定结构程序码。

5、如权利要求1所述的计算机网络，其中还包括:

一个初始程序计算机，该计算机提供中性结构程序，所述初始程序计算机包括一个署名发生器，用于产生被附加到所述中性结构程序码上的始发方的数字署名;

所述程序编译计算机与所述初始程序计算机进行通信，以从所述初始程序计算机接收中性结构程序并向所述初始程序计算机提供所述特定结构程序;

所述程序执行计算机与所述初始程序计算机进行通信，以从所述初始程序计算机接收所述中性结构和特定结构程序；

所述程序执行计算机的署名检验器还检验所述始发方的数字署名；

所述执行器只有在所述始发方的数字署名被验证之后才执行所述特定结构程序。

6、一种操作计算机网络的方法，包括如下步骤：

在由一个编译方操作的程序编译计算机处：

接收由一个始发方产生的中性结构程序，该中性结构程序含有中性结构程序码和所述始发方的数字署名，当其被检验时，检验由所述始发方所标记的所述中性结构程序；

检验所述始发方的数字署名；和

当所述始发方的数字署名被检验完毕时，利用一个编译程序对所述中性结构程序进行编译，从而产生一个特定结构程序，并附加一个编译方的数字署名，当其被检验时，检验由所述编译方产生的特定结构程序；和

在由一个执行方操作的程序执行计算机处：

接收所述特定结构程序和接收或始发所述中性结构程序；

检验所述编译方的数字署名；和

当所述编译方的数字署名被验证完毕且所述编译方被确定是一指定设定的受托编译方时，执行所述特定结构程序。

7、如权利要求 6 的方法，其中，包括：

在程序编译计算机处：

产生一个编译程序的数字署名，当其被检验时，检验用所述编译程序产生的所述特定结构程序；和

将所述编译程序的数字署名附加到所述特定结构程序码中；和

在程序执行计算机处：

检验所述编译程序的数字署名；和

仅在所述编译程序的数字署名被检验完毕且所述编译程序被确定是一指定设定的受托编译程序之后才执行所述特定结构程序。

8、如权利要求 6 的方法，其中，

对于所述始发方和编译方来讲，所述网络包括相应的专用和公用加密密钥和对应的散列函数；

所述始发方的数字署名包括通过在所述中性结构程序的基础上执行所述始发方的相应散列函数产生的所述中性结构程序的消息摘要，利用所述始发方的相应专用密钥加密所述中性结构程序的消息摘要;

所述方法包括如下步骤:

在程序编译计算机处:

通过如下方式检验所述始发方的数字署名，所述方式是: (A)利用所述始发方的公用加密密钥对所述中性结构程序的消息摘要解密，(B)通过在所述中性结构程序码的基础上执行所述始发方的散列函数来产生所述中性结构程序对应的测试消息摘要，和(C)比较所述中性结构程序的解密消息摘要和测试消息摘要; 和

通过如下方式产生编译方的数字署名，所述方式是: (A)通过在所述特定结构程序码的基础上执行所述编译方的相应散列函数，产生一个所述特定结构程序的消息摘要，和(B)利用所述编译方的相应专用密钥加密所述特定结构程序的消息摘要; 和

在程序执行计算机处:

通过如下方式检验所述编译方的数字署名，所述方式是: (A)利用所述编译方的公用加密密钥解密所述特定结构程序的消息摘要，(B)通过在所述特定结构程序码的基础上执行所述编译方的散列函数，产生所述特定结构程序的相应测试消息摘要，和(C)比较所述特定结构程序的解密消息摘要和测试消息摘要。

9、如权利要求 6 的方法，其中，包括:

在程序执行计算机处:

由检验所述中性结构程序码满足预定程序完整性标准来检验所述中性结构程序的完整性; 和

仅在所述中性结构程序码的完整性被检验完毕之后才执行所述特定结构程序码。

10、如权利要求 6 的方法，其中，还包括:

在提供所述中性结构程序的初始程序计算机中:

产生所述始发方的数字署名并将它附加到所述中性结构程序码中;

在所述程序编译计算机处:

与所述初始程序计算机进行通信，以从所述初始程序计算机中接收所

述中性结构程序并将所述特定结构程序提供给所述初始程序计算机;

在所述程序执行计算机处:

与所述初始程序计算机进行通信，以从所述初始程序计算机中接收，所述中性结构和特定结构程序;

检验所述始发方的数字署名；和

仅在所述始发方的数字署名被检验完毕之后才执行所述特定结构程序。

11、如权利要求 6 的方法，其中，包括:

在所述程序执行计算机处:

提供所述中性结构程序;

产生一个始发方的数字署名，该署名标记所述中性结构程序并将它附加到所述中性结构程序码中；和

在所述程序编译计算机处:

与所述程序执行计算机进行通信，以从所述程序执行计算机中接收所述中性结构程序并将所述特定结构程序提供给所述程序执行计算机。

受托编译中性结构程序版本产生特定结构的系统和方法

技术领域

本发明一般涉及一种分布式计算机系统，特别涉及一种程序编译系统(Program Compilation System)和方法，其中，由受托的第三方以如下方式对中性结构可执行程序进行编译，所述的方式是指该编译程序的接受者能够检验所述相应中性结构程序的同一性(identity)并能够检验它由所述受托第三方(trusted third party)所编译。

背景技术

术语“结构”被规定用于这个文件的目的意指一系列型号计算机(family of computer models)的运行特性。不同的结构例子是：Macintosh计算机、使用DOS或Windows操作系统的IBM PC兼容计算机、运行Solaris操作系统的SUN微系统计算机和使用Unix操作系统的计算机系统。

术语“中性结构”(architecture neutral)被规定用于这个文件的目的涉及到使用一些不同的计算机结构在多种计算机平台上执行诸如用Java(Sun Microsystem公司的商标)语言编写的程序等某些程序的能力。

术语“特定结构”被规定用于这个文件的目的涉及到仅使用一个单一的计算机结构在计算机平台上执行某些程序的请求。例如，用80486汇编程序语言编写的目标代码程序只能在使用IBM PC兼容计算机结构的计算机(以及含有IBM PC兼容计算机仿真器的其它计算机)上执行。

中性结构程序(ANPrograms)的重要特性包括用所述中性结构语言(ANLanguage)编写的程序的结构独立性。例如，Java字节码程序可以在具有Java字节码解译程序的任意一个计算机平台上执行。Java字节码程序的另一个重要特性是在执行以前利用Java字节码检验器(verifier)可以直接检验它们的完整性。Java字节码检验器确定所述的程序是否符合预定的完整性标准。这些标准包括操作数栈(operand stack)和数据类型使用限制，该使用限制保证Java字节码程序不会在执行计算机的操作数栈中上溢或下溢，并保证

所有的程序指令仅利用已知数据类型的数据。其结果是 Java 字节码程序不能够建立目标指针，并且，通常不能访问除用户显然已经被授权允许使用的系统资源以外的系统资源。

遗憾的是，分布在 ANLanguage 中的可执行程序使得所述 ANProgram 的运行效率要低于如果利用特定结构性能时所能达到的效率。例如，由 Java 字节码解译程序执行的 Java 字节码程序的运行速度通常要比在相应的特定结构语言(ASLanguage)中编译的等效特定结构程序(ASProgram)慢 2.5 到 5 倍。在速度减少 5 倍被认为对于一个 ANProgram 执行器(即，解译程序)来讲是不好的同时，它会在效率方面带来足够大的损失，所以某些用户将需要或坚持要求按 ASLanguage 编译的等效程序的能力使用。

编写的编译程序可将 ANProgram 编译成等效 ASProgram，但是，对于终端用户来讲，它们特别昂贵。另外，不能利用一个 ANProgram 完整性检验器根据编译的ASProgram代码来直接检验所述等效编译的ASProgram的完整性。因此，在 Java 字节码程序的情况下，将 ANProgram 编译成等效 ASProgram 的这种使用可能导致 ANLanguage 的一种最重要特性的丧失。

但是，存在有某些合法的(或正当的)任务，这些任务可以利用完整性不可检验的 ASProgram 加以执行，但不能利用完整性可检验的 ANProgram 加以执行。这些任务包括其它违背设置在完整性可检验的 ANProgram 上的操作数栈和数据类型使用限制的任务，另外，这种 ASProgram 能够比 ANProgram 执行得更快，其结果是就存在有为什么希望具有如下计算机系统的理由，这种计算机系统被设计成除了具有执行完整性不可检验的 ASProgram 的能力以外，它主要用于执行完整性可检验的 ANProgram。

虽然通过第三者执行 ANProgram 的编译是可能的，但是，这种编译需要将所述的第三者认证，就是说，必须可以对由特定受托的第三者编译的 ASProgram 中的信息进行检验。最好，它还能够确认编译后的 ASProgram 是由一个特定受托编译程序产生的，并且由于相对于预定完整性标准编译后的 ASProgram 的完整性不能够被直接检验，所以该编译后的 ASProgram 应当包括某种信息，这种信息可以用能检验的方式识别依据其进行编译的相应的 ANProgram 和对其进行编译的 ASLanguage。

发明内容

因此，本发明的一个目的是提供一种 ANProgram 编译程序和编译方法，这种编译方法使根据相应的 ANProgram 所编译的 ASProgram 的用户能够确认编译该 ANProgram 的编译者的同一性以及所述相应的 ANProgram 的同一性和其中所编译的 ASProgram 的 ASLanguage 语言。

本发明的另一个目的是提供一种 ANProgram 执行器(executer)和一种执行方法，这种方法是在防止调用其源、编译信息和完整性不能检验的 ASProgram 的同时，允许正在执行的、完整性可检验的 ANProgram 去调用受托的或具有可检验源和编译信息的完整性不可检验的 ASProgram，从而使得几乎所有的合法任务都可以被执行。

本发明提供一种计算机网络，包括：由一个编译方操作的程序编译计算机，该程序编译计算机接收一个由始发方产生的中性结构程序，该中性结构程序包含有中性结构程序码和所述始发方的数字署名，当其被检验时检验由所述始发方所标记的中性结构程序，所述程序编译计算机包括：一个署名检验器，用于检验所述始发方的数字署名；一个编译单元，用于当所述始发方的数字署名被检验完毕时，产生特定结构程序，所述编译单元，通过下述方式产生所述特定结构程序，所述方式是：(A)按照特定结构语言将所述中性结构程序码编译成特定结构程序码，和(B)附加一个编译方的数字署名，当其被检验时，检验由所述编译方产生的特定结构程序；署名发生器，用于产生编译方的数字署名；和由一个执行方操作的程序执行计算机，该程序执行计算机接收特定结构程序，和接收或始发该中性结构程序，所述程序执行计算机包括：一个署名检验器，用于检验所述编译方的数字署名；一个执行器，用于执行按所述特定结构语言写成的程序码，当所述编译方的署名已被检验，并且所述的编译方是指定设定的受托编译方时，所述执行器执行特定结构程序码。

本发明还提供一种操作计算机网络的方法，包括如下步骤：在由一个编译方操作的程序编译计算机处：接收由一个始发方产生的中性结构程序，该中性结构程序含有中性结构程序码和所述始发方的数字署名，当其被检验时，检验由所述始发方所标记的所述中性结构程序；检验所述始发方的数字署名；和当所述始发方的数字署名被检验完毕时，利用一个编译程序对所述中性结构程序进行编译，从而产生一个特定结构程序，并附加一个编译方的数字署

名，当其被检验时，检验由所述编译方产生的特定结构程序；和在由一个执行方操作的程序执行计算机处：接收所述特定结构程序和接收或始发所述中性结构程序；检验所述编译方的数字署名；和当所述编译方的数字署名被验证完毕且所述编译方被确定是一指定设定的受托编译方时，执行所述特定结构程序。

总之，本发明是一种计算机网络，该网络包括一个程序编译计算机和一个程序执行计算机。

所述程序编译计算机是由编译方(compiling party)运行的，它包括一个存储器，用于存储由一个始发方(originating party)产生的中性结构程序。所述中性结构程序包含有中性结构程序码和所述始发方的一个数字署名(signature)。所述程序编译计算机还包括一个署名检验器，该检验器检验所述始发方的数字署名，以验证所述始发方的数字署名匹配与附加到中性结构程序的署名(即：是由其产生)。

所述程序编译计算机还包括一个编译程序，当验证完所述始发方的数字署名时，该编译程序按照由对信息编译所识别的特定结构语言，将该中性结构程序码编译成特定结构程序码。该编译程序使用一个署名发生器将一个所述编译程序的数字署名附加到所述特定结构程序码上，其中，该编译程序的署名标记一组信息，该信息包括所编译的特定结构程序码加上位于所述中性结构程序上的署名。在最佳实施例中，所述编译程序使用上述署名发生器将所述编译方的数字署名也附加到特定结构程序码上，其中，编译方署名标记一组信息，该信息包括所编译的特定结构程序码、位于中性结构程序上的署名和该编译程序署名。

所述程序执行计算机由执行方操作，它包括一个存储器，用于存储中性结构和特定结构程序。它还包括一个署名检验器，用于(A)检验处于中性结构程序形式下的所述始发方的数字署名，和(B)检验处于特定结构程序形式下的编译程序的数字署名和/或检验处于特定结构程序形式下的编译方的数字署名。术语“检验一个署名”的意思是执行一个过程以确定所述的署名与被所述署名标记的该组信息相匹配(即：实际上是由其产生)。

所述程序执行计算机还包括一个特定结构程序执行器，当处于所述特定结构程序形式下的数字署名被验证完毕时，该特定程序执行器执行所述特定结构程序的特定结构程序码。

在最佳实施例中，在一个对象中表示了所述中性结构程序，所述对象含有一个数字署名，该数字署名包括唯一与所述中性结构程序相关的一个消息摘要。由所述编译程序产生的该特定结构程序包括:

·该编译的、特定结构码;

·由所述中性结构程序的提供方标记的该相应的中性结构程序的数字署名;

·编译程序本身的数字署名，包括所编译的程序的消息摘要和用于识别编译该程序所使用的和利用编译程序的专用加密密钥标记的编译程序的信息; 和

·执行该编译的受托方的数字署名，包括所编译的程序的消息摘要和用于识别该受托方的信息和使用编译方的专用加密密钥标记的信息。

通常可以获得的公用加密密钥的受托储存部分(repository)，有时调用一个命名服务，并保持该公用密钥以用于该编译程序和受托的编译方。使用这些公用加密密钥，该编译程序的所有接受者都能够对编译后的程序中的数字署名进行解密，以验证所述的编译程序是由指定的受托方编译以及是由指定的编译程序编译，并且，检验相应的中性结构程序的同一性。作为选择，编译程序的接受者能够使用一个程序检验器在执行所编译的特定结构程序之前去检验相应的中性结构程序的正确运行。

附图简述

本发明的其它目的和特征通过下述结合附图的详细描述及所附权利要求将变得更清楚。

图 1 是本发明最佳实施例中的一个分布计算机系统方框图;

图 2 是根据本发明最佳实施例的一个中性结构程序的结构;

图 3 是根据本发明最佳实施例产生的一个编译的、特定结构程序结构;

图 4 是根据本发明一个最佳实施例的一个对象和相关的对象分类。

具体实施方式

参看图 1，这里示出了一计算机网络 100，它包括很多客户计算机 102、服务计算机 104 和受托密钥储存部分 106。经过网络通信连接器 108，客户计算机 102 彼此相连并与服务计算机 104 和受托密钥储存部分 106 相互连接。

所述网络通信连接器可以是一个局部或广域网络、互联网络、这些网络的结合或某些其它类型的网络通信连接器。

虽然大多数客户计算机 102 是诸如 Sun 工作站、IBm 兼容计算机、Macintosh 计算机等桌面计算机，但是实际上任何类型的计算机都可作一客户计算机。这些客户计算机中的每一个都包括一个 CPU 110，一个用户接口 112，一个存储器 114 和一个网络通信接口 116。所述网络通信接口允许多个客户计算机彼此间以及与服务计算机 104 和与受托密钥储存部分 108 之间经过网络通信连接器 106 相互通信。

每个客户计算机 102 的存储器 114 存储有一个操作系统 118，一个网络通信管理器 120，一个 ANProgram(中性结构程序)执行器 122，一个 ASProgram(特定结构程序)执行器 124,和一个 ANProgram 完整性检验器 126，一个 ANProgram 编译制备器 128，一个署名发生器 130，一个署名检验器 132，一个编译信息(CompInfo)检验器 134，一个对象分类加载器 136，一个用户地址空间 138，一个受托对象分类储存部分 140，一个未受托对象分类储存部分 142 和一个已知的受托编译方和受托编译程序的表 144。所述操作系统运行在 CPU 110 上，并响应由一用户通过所述用户接口 112 发出的命令而控制和协调在所述 CPU 上运行的程序 120－136。

每个客户计算机 102 的 ANProgram 执行器 122 执行存储在所述受托和未受托对象分类储存部分 140 和 142 内所述对象分类中的 ANProgram。并且所述 ANProgram 按照一个 ANLanguage 编写，对于这个 ANLanguage，用户可建立诸如堆栈和数据使用限制的预定完整性标准，从而使得该 ANProgram 不执行非法的任务。因此，在执行之前，通过确定该程序是否满足所述预定完整性标准，ANProgram 完整性检验器 126 可以直接验证所述 ANProgram 的完整性。因此，这个 ANProgram 被认为是完整性可检验的 ANProgram。

在最佳实施例中，所述完整性可检验的 ANProgram 由所述 Java 字节码语言编写。并且,ANProgram 执行器 122 和 ANProgram 检验器 124 分别是 Java 字节码程序解译器和 Java 字节码程序检验器,它们分别被用于执行和检验所述的 Java 字节码程序。所述 Java 字节码检验器和解译器是由 Sun Microsystem 公司生产的。

但是，每个客户计算机 102 都具有一个相关的特定结构，对于这个结构，许多程序可以按相应的 ASLanguage 编写并由所述 ASProgram 执行器 122 执

行。所述ASLanguage不需要按ASLanguage编写的ASProgram满足ANLanguage的预定完整性标准。其结果是所述ASProgram能够执行某些任务，而这些任务不能够由所述 ANProgram 执行由于其没有被装载而受到所述 ANLanguage的预定完整性标准所设置的限制。但是，遗憾的是这也意味着它们的完整性不能够利用所述ANProgram完整性检验器126直接进行检验，并因此被认为完整性不可检验。

尽管如此，如前面所指出的，ANProgram运行的效率要低于按ASLanguage编译的同一程序。因此考虑到与用户的客户计算机相关的ASLanguage，客户计算机102的用户可以希望具有一个由服务计算机(server computer)104编译的ANProgram，以使所编译的ASProgram能够在那里被ASProgram执行器124执行，或者，如果利用其它客户计算机的ASProgram执行器124分布和执行此编译的 ASProgram，那么用户可以希望具有考虑到与其它客户计算机相关的ASLanguage编译的ANProgram。

制备一个用于编译的中性结构程序(ANProgram)

参看图1和2，当一个始发方(OrigParty)希望具有一个由所述服务计算机104编译的ANProgram 200时，OrigParty利用用户接口112发出一个命令以对所述ANProgram编译制备器128提出一个请求并指令它制备用于编译的ANProgram。该ANProgram可以位于包含在受托或未受托对象分类储存部分140或142的其中之一内的一个对象分类中。表1包含有由ANProgram编译制备器128使用的该过程的伪代码表示，用于制备由服务计算机104编译的ANProgram。表1－3中使用的伪代码采用通用计算机语言规范。这里使用伪代码的主要目的是用于叙述，同时，它可以很容易被本专业技术领域内的计算机程序编制人员所理解。

参看图1和图2以及表1，所述ANProgram编译制备器128首先调用所述ANProgram完整性检验器126并指令它检验ANProgram 200的ANProgram代码202的完整性。做这项工作的目的在于确认在将其传送给服务计算机104进行编译之前，ANProgram码满足所述ANLanguage的预定完整性标准。如果所述ANProgram码不满足预定的完整性标准，那么，所述ANProgram完整性检验器将一失效结果送回ANProgram编译制备器。作为响应，所述ANProgram编译制备器中止编译制备过程并产生一个适当的消息以指示该状态。

但是，如果该 ANProgram 码 202 满足所述预定的完整性标准，那么，

ANProgram 完整性检验器 126 将一个通过的结果回送给 ANProgram 编译制备器 128。该 ANProgram 编译制备器然后调用署名发生器 130 并指令它产生 origParty 的数字署名(DigitalSignature$_{OP}$)210，该署名 210 可以被检验以保证所述 ANProgram 200 由受托的 OrigParty 产生。署名发生器通过首先产生所述 ANProgram 码 202 的消息摘要(MD_{OP})212 产生所述的 DigitalSignature$_{OP}$。这是通过在 ANProgram 码的数据比特上计算一个散列函数、即 HashFunction$_{OP}$ 而进行的。所使用的该散列函数可以是一个预定的散列函数，也可以是由所述 OrigParty 选择的一个散列函数。本文为描述的目的，该 HashFunction$_{OP}$ 对应于所述 OrigParty，由于它用于所述 OrigParty 的 DigitalSignature$_{OP}$。

然后，署名发生器 130 利用 OrigParty 的专用加密密钥(OrigParty 与 Privatekey) 加密所产生的消息摘要(MD_{OP})212 和 HashFunction$_{OP}$ (HashFunction$_{OP}$ ID)214 的该 ID。然后，该署名发生器将所述 OrigParty 的 ID 216 加到位于加密项 212 和 214 末端处明文(clear text)中，以形成所述 DigitalSignature$_{OP}$。OrigParty 的专用密钥和 ID 由具有用户接口 112 的 OrigParty 提供。

在产生 DigitalSignature$_{OP}$ 210 以后，所述 ANProgram 编译制备器 128 将它附加到 ANProgram 码 202 上。然后，ANProgram 编译制备器产生一个已经制备的所述服务计算机 104 用于编译的 ANProgram 200 的消息。

然后，OrigParty 利用用户接口 112 向网络通信管理器 120 发出一个命令，以便将所述的 ANProgram 200 和一些变元(argument)一起传送给服务计算机 104，这些变元说明将被编译成该程序的特定结构语言(ASLanguage ID)和所使用的编译程序(Compiler ID)。该网络通信管理器从配置有受托或未受托的对象分类储存部分 140 或 142 中提取所述 ANProgram，并将它提供给网络通信接口 116。然后，该网络通信管得器指令网络通信接口将 ANProgram 和所述特定变元一起传送给服务计算机。

编译一个中性结构程序

然后，服务计算机 104 接收被传送的 ANProgram 200。该服务计算机包括 CPU 150，用户接口 152，存储器 154 和网络通信接口 156。该网络通信接口允许该服务计算机经过网络通信连接器 108 与客户计算机 102 和受托密钥储存部分 106 进行通信。

服务计算机 104 的存储器 154 存储一个操作系统 158，一个网络通信管理器 160，一个 ANProgram 编译程序 162，一个署名检验器 164，一个 ANProgram 完整性检验器 166，一个署名发生器 168，一个 ANProgram 储存部分 repository 170 和一个 ASProgram 储存部分 repository 172。操作系统在 CPU 150 上运行，并响应编译方(CompParty)利用用户接口 152 发出的命令控制和协调在 CPU 上运行的程序 160－168。

该网络通信接口 156 接收该 ANProgram 200 并将正发生的该情况指示给网络通信管理器 160。作为响应，该网络通信管理器将所接收的 ANProgram 放置入 ANProgram 储存部分 170 中。如果所述服务器 104 被设置成自动编译程序服务器，那么，通过网络通信管理器 160 自动执行编译。反之，当该 CompParty 利用所述用户接口发出一个命令时，所述网络通信管理器将所述 ANProgram 移入储存部分 170。

然后，或是自动地、或是根据该 CompParty 利用所述用户接口 252 发出的命令，该 ANProgram 编译程序 162 被请求对 ANProgram 200 进行编译。表 2 包含有该 ANProgram 编译程序对所述 ANProgram 进行编译所使用的编译过程伪码表示。

参看图 1－2 和表 2，该 ANProgram 编译程序 162 首先调用该署名检验器 164 去检验在所接收的 ANProgram 200 中的 $DigitalSignatuye_{OP}$ 210，从而建立如下状态，即：所述 $DigitalSignatuye_{OP}$ 210 实际上是始发方用于该 ANProgram 的署名(例如，与在所述 ANProgram 某些其它版本上的 OrigParty 署名或伪造的署名相对照)。特别是，该署名检验器使用在所接收 ANProgram 中的明文 Orig Party 的 ID 216 以便从受托密钥储存部分 106 中获得所述 OrigParty 的公用密钥。然后，该署名检验器使用所述 OrigParty 的公用加密密钥(OrigParty's publickey)对在 $DigitalSignatuye_{OP}$ 中的被加密的 MD_{OP} 212 和 $HashFuction_{OP}$ ID 214 进行解密。

接着，该署名检验器 164 根据所接收的 ANProgram 200 的 ANProgram 码 202 计算相应的 $Hahs\ Fuction_{OP}$ 而产生一个测试消息摘要($TestMD_{OP}$)，该测试消息摘要应当与被解密的 MD_{OP} 相匹配。解密的 $DigitalSignatuye_{OP}$ 中的 $HashFuction_{OP}$ ID 214 用于识别所使用的适当的 $HashFuction_{OP}$。然后，将解密的 MD_{OP} 和所产生的 $TestMD_{OP}$ 进行比较以检验所述的 $DigitalSignatuye_{OP}$ 210。

如果 MD_{OP} 212 和 $TestMD_{OP}$ 不相匹配，那么，该署名检验器 162 回送一个失效结果给该 ANProgram 编译程序 162。作为响应，该 ANProgram 编译程序中止编译过程并产生一个适当的消息。

另一方面，如果 MD_{OP} 212 和 $TestMD_{OP}$ 相匹配，那么，署名检验器回送一个通过结果给 ANProgram 编译程序 162，并且所述 ANProgram 编译程序 162 调用 ANProgram 完整性检验器 166。它指令所述 ANProgram 完整性检验器去检验所接收 ANProgram 200 的 ANProgram 码 202 的完整性。与在前面讨论的制备用于编译的 ANProgram 部分相比，其执行的方式相同并具有相同的目的。因此，如果 ANProgram 码不满足所述预定完整性标准，那么，ANProgram 完整性检验器就要回送一个失效结果给 ANProgram 编译程序。作为响应，所述 ANProgram 编译程序中止编译过程并产生一个指示该状态的适当消息。

但是，如果所接收 ANProgram 200 的 ANProgram 码 202 满足了所述的预定完整性标准，那么，ANProgram 完整性检验器 166 回送一个通过结果给 ANProgram 编译程序 162。编译程序 162 然后将 ANProgram 码编译成利用由所述 OrigParty 指定的 ASL anguage ID 识别的 ASLanguage。参看图 1－3 和表 2，所述编译程序将 ANProgram 码 202，$DigitalSignature_{OP}$ 210 和编译后的 ASProgram 码 302 放置入 ASProgram 300，该 ASProgram 300 被存储在 ASProgram 储存部分 172 中。

ANProgram 编译程序 162 然后调用署名发生器 168 并指令它产生 ANProgram 编译程序的数字署名 ($DigitalSignature_C$) 320，该数字署名 320 可以被检验以保证所述 ASProgram 300 由受托的 ANProgram 编译程序编译。它的执行方式类似于前面描述的有关产生 $DigitalSignature_{OP}$ 的执行方式。但是，在这种情况下，所标记的一组信息是 ASProgram 码和 $DigitalSignature_{OP}$。具有相应 $HashFunction_C$ ID 234 的其它预定散列函数可以被用于产生将由 $DigitalSidnature_C$ 标记的该组信息的消息摘要 MD_C 322，ANProgram 编译程序的专用加密密钥(编译程序的专用密钥)被用于加密所述 MD_C 和 $HashFunction_C$ ID，ANProgram 编译程序的识别符(编译程序的 ID)被加到位于被加密的 MD_C 和 $HashFunction_C$ 末端处的明文中。所述编译程序的专用密钥和 ID 由 ANProgram 编译程序提供。

ANProgram 编译程序 162 第二次调用署名发生器 168 以产生 CompParty 的数字署名 ($DigitalSignature_{CP}$) 312，该数字署名 312 可以被终端用户 (end

user)检验以保证所述 ASProgram 300 由受托的 CompParty 产生。关于这一点的执行方式类似于前面所述的关于(在制备用于编译的 ANProgram 的讨论中)产生 $DigitalSignature_{OP}$ 的方式。但是，这里为 $DigitalSignature_{CP}$ 产生的消息摘要 (MD_{CP}) 314 是通过在 ASProgram 码、$DigitalSignature_{OP}$ 和 $DigitalSignature_{C}$ 的数据比特的基础上计算一个预定的或所选择的散列函数 ($HashFunction_{CP}$) 产生的。与所述 $HashFunction_{OP}$ 类似，为该文件叙述的目的，$HashFunction_{CP}$ 对应于所述 CompParty，由于它被用于 CompoParty 的 $DigitalSignature_{CP}$。

然后，署名发生器 168 利用 CompParty 的专用加密密钥 (CompParty 的专用密钥) 加密所述 MD_{CP} 314 和 $HashFanction_{CP}$ 的 ID ($HashFunction_{CP}$ ID) 316。然后，署名发生器将 CompParty 的识别符 (CompParty 的 ID) 318 加到位于被加密项 314 和 316 末端处明文中以形成 $DigitalSignature_{CP}$312。CompParty 的专用密钥和 ID 是由 CompParty 利用所述用户接口 152 提供的。

在产生 $DigitalSignature_{C}$320 和 $DigitalSignature_{CP}$312 以后，ANProgram 编译程序 162 将它们附加到 ASProgram 码 302 上，从而使在所产生的编译的 ASProgram 文件或对象中具有下述成分：

ANProgram 码，
$DigitalSignature_{OP}$，
ASProgram 码，
$DigitalSignature_{C}$，和
$DigitalSignature_{CP}$。

然后，该 ANProgram 编译程序产生一个消息，该消息表示 ANProgram 200 已经被编译成 ASProgram 300，并已经准备传送给所述 OrigParty。

CompParty 然后使用网络通信管理器 160 将 ASProgram 300 传送给 OrigParty 的客户计算机 102。网络通信管理器是通过从其中设置有 ASProgram 的 ASProgram 储存部分 172 中提取所述 ASProgram 的并将它提供给所述网络通信接口 156 而作到这一点的。然后，网络通信管理器指令所述网络通信接口将所述 ASProgram 传送给 OrigParty 的客户计算机。

对象和对象分类的建立和分布

然后，所传送的 ASProgram 300 由 OrigParty 客户计算机的通信接口 116 接收并将该发生的情况指示给网络通信管理器 120。作为响应，OrigParty

利用用户接口 252 发出一个命令，以命令从网络通信接口中提取接收的 ASProgram，从而使得网络通信管理器将所接收的 ASProgram 放置入所述 OrigParty 客户计算机的未受托对象分类储存部分 142 中。一旦做了这件事情，OrigParty 可以将所接收的 ASProgram 作为仅具有一种方法的新对象分类(例如编译程序的代码)进行处理，或者它可以建立一个对象分类，该对象分类包括 ASProgram 300 以及其它的 ANProgram 和 ASProgram。

图 4 示出了根据本发明的一个典型的对象分类 400。该对象分类可以包括一个或多个 ASProgram 402 和/或一个或多个 ANProgram 404，以及一个虚拟函数表 410。对于每个 ASProgram 来讲，所述虚拟函数表包含着指示一个不是用 ANLanguage 编写的 ASProgram(即本机程序)的相应的识别符(identifiet (native_ASProgram ID)) 412 并包含一个到本机程序的相应指针(Ptr) 414。类似的，对于每个 ANProgram 来讲，虚拟函数表包含一个相应的识别符(AN-Program ID) 416 和一个到所述 ANProgram 的相应指针 418。这个对象分类的每个对象 420 包括一个指向对象分类 400 的对象标题 422。

因此，OrigParty 可以利用 ASProgram 300 建立对象 420 和对象分类 400，所述 ASProgram 300 从服务计算机 104 中被接收而作为所述对象分类内多个 ASProgram 402 中的一个。

当所述 OrigParty 希望将一个对象和包括 ASProgram 300 和 ANProgram 在内的对象分类分布给各个 ExecuteParty(执行方)时，OrigParty 利用用户接口 112 发出一个命令，以指令所述网络通信管理器将这些项(items)传送给 ExecutePartys 的客户计算机 102。网络通信管理器通过从其中设置有这些项的未受托对象分类储存部分 142 中提取它们、并利用适当的传送指令将它们提供给网络通信接口 116 而做这项工作。另外，OrigParty 的网络通信管理器可以响应由一个 ExecuteParty 启动的请求，以对指定的对象分类 400 进行复制。

在对象分类中的中性结构程序和特定结构程序的执行

客户计算机 102 的网络通信接口 156 接收所传送的对象和对象分类并将已发生的状态指示给网络通信接口 160。作为响应，ExecuteParty 利用用户接口 112 发出一个命令，该命令指令所述网络通信管理器提取从网络通信接口所接收的对象和对象分类。然后，网络通信管理器将所接收的对象和对象分类存储到未受托对象分类储存部分 142 中。

每个客户计算机102的未受托对象分类储存部分142包含有多个未受托的对象和与其有关的多个对象分类。当根据适当的ANProgram进行编译时，由于它们所包括的任意的ANProgram都还不具有已经过检验的其完整性和它们所包括的任意的ASProgram还都不具有已经过检验或还没有经过检验的其源，所以，这些对象分类不是受托的。

每个客户计算机的受托对象分类储存部分140包含有多个受托对象和其多个对象分类。由于它们所包括的任意ANProgram已经可具有经过ANProgram完整性检验器166检验过的其完整性和它们所包括的任意ASProgram已经被确定为是值得信赖的，所以，这些对象分类是受托的。实际上，由于这些对象分类是受托的，因此就没有理由在这些对象分类的所述方法上执行完整性检查，所以，在受托对象分类储存部分140中的某些或全部对象分类并不需要具有数字署名。

下面这一点是希望的，即如前面所建议的，一个对象分类主要包括ANProgram，但还可以包括ASProgram，从而使得利用所述对象分类基本上可以执行所有的合法任务如上述所建议的。因此，ANProgram执行器122能够执行完整性可检验的ANProgram并能调用ASProgram执行器去执行完整性不可检验的ASProgram，所述完整性不可检验的ASProgram位于(1)在受托对象分类储存部分140内的受托对象分类中，或(2)在未受托对象分类储存部分142内的未受托对象分类中并具有可检验的$DigitalSignature_{OP}$、$DigitalSignature_{CP}$和$DigitalSignature_{C}$信息，从而，基本上可以执行所有的合法任务。在这种方式下，不具有$DigitalSignature_{OP}$、$DigitalSignature_{CP}$和$DigitalSignature_{C}$信息或它们的数字署名不能够被检验的未受托对象分类的ASProgram将被避免执行。表3包含有由所述ANProgram执行器使用的执行过程的伪码表示。

参看图1－4和表3，在一个ExecuteParty(例如，OrigParty或另外的一方)的客户计算机102上，ANProgram执行器124可以正在执行ANProgram其试图调用一个特定对象分类的方法。所述方法调用最初是由对象分类加载器136处理的，它确定所述对象分类是否已经被装载。如果所述对象分类已经被装载到ExecuteParty的用户地址空间138中，那么，假如所调用的方法是一个ANProgram，则ANProgram执行器122执行所调用的方法，假如所调用的方法是一个ASProgram，则ASProgram执行器124执行所调用的方法。

但是，如果所述对象分类还没有被装入到 ExecuteParty 的地址空间 138 中，那么，对象分类加载器 136 就要将所述对象分类装入到 ExecuterParty 的地址空间中并确定所调用的方法是否被允许执行。例如，如果所述对象分类是从受托对象分类储存部分 140 装入的，那么，所调用的方法被允许执行，并调用所述 Execute 过程。如果所调用的方法是一个 ANProgram，Execute 过程(见表 3)调用所述 ANProgram 执行器去执行所调用的方法。反之，则调用 ASProgram 执行器 124 去执行所调用的方法。

但是，如果所述对象分类是从未受托的对象分类储存部分 142 装入的，那么，分类加载器 136 检查所述对象的对象标题以确定它的对象分类是否还包括任意的 ASProgram。这是通过确定在该对象的虚拟函数表中是否存在任意 native_ASProgram ID 进行的。

如果在所述对象分类中不存在 ASProgram，那么，分类加载器 136 调用 ANProgram 完整性检验器 166 去检验在该对象分类中 ANProgram 的完整性。其执行方式和目的与前面所述检验 ANProgram 200(在讨论对 ANProgram 进行编译的部分中)所使用的方式和目的相类似。因此，如果多个 ANProgram 中任意一个的完整性没有被检验，那么，ANProgram 完整性检验器将回送一个失效结果给所述分类加载器，并且，所述分类加载器中止分类装载过程并产生一个适当的消息以指示该状态。但是，如果 ANProgram 完整性检验器回送一个表示该对象分类的所有 ANProgram 都已经被检验通过的结果(passed result)时，那么，分类加载器能执行所调用的方法。

如果在所述对象分类中存在有任意一个 ASProgram，那么，分类加载器 136 调用署名检验器 132 去检验编译程序 $DigitalSignature_C$ 和 CompParty 的署名 $DigitalSignature_{CP}$。如果 ASPrograms 中的任意一个不包括 $DigitalSignature_{CP}$ 和 $DigitalSignature_C$，那么，ASProgram 源的完整性不能被检验，因此，署名验证器回送一个失效结果给 ANProgram 执行器。作为响应，分类加载器中止对象分类装载过程并产生一个适当的消息指明发生了该状态。

另外，如果在所述对象分类中的所有 ASPrograms 都包括 $DigitalSignature_{CP}$ $DigitalSignature_C$，那么，将在这两个数字署名中所指出的 CompParty 和 Ciompiler(编译程序)的同一性与已知的表 144(见图 1)。受托的 Compiler Parties 与受托的 Compilers(编译程序)进行比较。如果在

由 CompParty 或 Compiler 编译的对象分类中的 ASProgram 内的任意一个 ASProgram 都不包括在已知的、受托的 Compiler Parties 和受托的 Comiler 设定中，那么，分类装载过程被中止，借此以阻止所调用方法的执行。类似的，如果在 ASProgram 中的任意一个 ASProgram 内识别的 ASLanguage 与由 ASProgram 执行器 124 使用的 ASLanguage 不匹配，那么，分类装载过程被中止。

但是，如果在该对象分类中的所有 ASPrograms 都包括 $DigitalSignature_{CP}$ 和 $DigitalSignature_{C}$，和用于所有 ASPrograms 的、被识别的 CompParty 和 Compiler 是受托的 Compiler Parties 和 Compilers，并且，由所有 ASProgram 使用的 ASLanguage 是由 ASProgram 执行器使用的语言，那么，署名检验器以与前面所述(在有关对 ANProgram 200 进行编译的讨论部分中)用于检验 $DigitalSignature_{OP}$ 的类似方式检验这些署名。但是，在这种情况下，所述 Compiler 和 CompParty 的公用密钥是从受托密钥储存部分 106 中提取的，并被分别用于对在 $DigitalSignature_{C}$ 中的 MD_{C} 和 $HashFunction_{C}$ ID 和在 $DigitalSignature_{CP}$ 中的 MD_{CP} 和 $HashFunction_{CP}$ ID 进行解密。另外，与解密的 MD_{CP} 和 MD_{C} 相应的测试消息摘要($TestMD_{C}$ 和 $TestMD_{CP}$)是通过计算散 列码产生，其是在 ASProgram 码数据比特上加 $TestMD_{C}$ 的 $DigitalSignature_{OP}$ 和在相同数据比特上加 $TestMD_{CP}$ 的 $DigitalSignature_{C}$，分别根据由解密的 $HashFunction_{C}$ ID 和 $HashFunction_{CP}$ ID 所识别的 $HashFunction_{C}$ 和 $HashFunction_{CP}$ 而计算的。

如果用于每个 ASProgram 的 $DigitalSignature_{C}$ 和/或 $DigitalSignature_{CP}$ 没有被验证(即：$MD_{C} \neq TestMD_{C}$ 和/或 $MD_{CP} \neq TestMD_{CP}$)，那么，署名检验器 132 回送一个失效结果给分类加载器 136，作为响应，分类加载器中止分类装载过程并产生一个适当的消息以指示该状态已经发生。

但是，如果用于每个 ASProgram 的 $DigitalSignature_{C}$ 和 $DigitalSignature_{CP}$ 都已经被验证(即：$MD_{C} = TestMD_{C}$ 和 $MD_{CP} = TestMD_{CP}$)，那么，ASProgram 执行器 124 再一次调用署名检验器 132 去检验用于 ANProgram 的所述 OrigParty(Digital $Signatore_{CP}$)的署名，所述 ASProgram 是根据所述 ANProgram 进行编译的。为了检验 OrigParty 的数字署名，每个 $DigitalSignature_{OP}$ 都是以与前面有关对 ANProgram 200 进行编译的讨论中所涉及的相同方式检验的。

如果，作为 ASProgram 编译基础的 ANProgram 中每一个 $DigitalSignature_{OP}$ 被验证，那么，分类加载器调用 ANProgram 完整性检验器去检验在所述对象分类中每个 ANProgram 的完整性和作为 ASProgram 编译基础的 ANProgram 的完整性。其执行方式与前述相同。如果这些 ANPrograms 中任意一个 ANProgram 的完整性没有被验证，那么，ANProgram 完整性检验器将回送一个失效结果给分类加载器，该分类装载器将中止分类装载过程并产生一个适当的消息。

但是，如果这些 ANrograms 中的每一个 ANProgram 的完整性都已经被验证，那么，ANProgram 完整性检验器 126 将回送一个通过的结果给分类加载器 136。作为响应，该分类加载器将请求 ANProgram 执行器或 ASProgram 执行器在适当的时候去执行所调用的方法。

从前述可看出，ExecuterParty 被保证只有在具有其数字署名可以被检验的完整性可验证的 ANProgram 和 ASProgram 的未受托储存部分 142 中的那些未受托对象分类将被装入并具有它们执行的程序。

另外的实施例

上述讨论的本发明的某些特征是可以选择的。因此，本专业技术领域内的技术人员可以认识到存在有某些未包括这些特性的另外一些实施例。

例如，已经描述了作为分别对 CompParty 和 ANProgram 编译程序产生的 $DigitalSignature_{CP}$ 和 $DigitalSignature_{C}$ 的 ANProgram 编译程序。但是，所述 ANProgram 编译程序能够被简单的构成以产生这些数字署名中的一个署名，该署名为了能对 ANProgram 进行编译的编译程序进行验证或对编译方(compiling party)进行验证。

类似的，已经描述了作为用于请求对 $DigitalSignature_{CP}$ 和 $DigitalSignature_{C}$ 进行检验的程序执行器。但是，所述的程序执行器能够被构成仅请求对这些数字署名中的一个进行检验，并且，如果正在被检验的 ASProgram 包括它的话，应当对另一个的数字署名进行可选择的检验。再有，在假设的基础上即在假设所述编译方是受托的并且该编译方的一项工作就是在执行所述编译之前去检验编译成 ASProgram 的每一个 ANProgram 的完整性的基础上，所述程序执行器能够被构成跳过检验与每一个 ASProgram 相对应的 ANProgram 的完整性的步骤。

当 Executer Party 是所述 Orig Party 时，Executer Party 知道，它实

际上是将 ANProgram 200 传送给 Comp Party 的服务计算机 104 以便将其编译成 ASProgram 300。在这种情况下，分类加载器 136 能够被结构成不调用署名检验器去检验在 ANProgram 中的 $DigitalSignature_{OP}$。而是使得所述 Executer Party 能够简单地将在 ANProgram 的局部复制中的 $DigitalSignature_{OP}$和在编译后 ASProgram 中的 $DigitalSignature_{OP}$加以比较。另外，由于在被传送给所述编译服务计算机之前的编译过程准备期间已经检查了 ANProgram 的完整性，所以，所述分类加载器能够被结构成不调用 ANProgram 完整性检验器去检验对应于所调用 ASProgram 的 ANProgram 的完整性。另外，由于所述 ANProgram 的完整性已经利用所述编译程序在执行相应 ASProgram 之前由所述分类加载器调用所述 ANProgram 完整性检验器进行了检查，所以，ANProgram 编译准备器 128 能够被构成在制备编译过程期间不调用所述 ANProgram 完整性检验器。

在结合本发明几个特殊实施例对本发明进行描述的同时，这指出这些描述仅仅是为了对本发明进行的说明，并不是对本发明的限制。本专业技术领域内的普通技术人员可以在不脱离本发明权利要求限定的本发明的构思和范围的前提下作出许多修改。

表 1

制备用于编译中性结构程序方法的伪代码表示

```
Procedure: Prepare for Compiling (ANProgram code, OrigParty's PrivateKey, and
        OrigParty's ID)
        {
        Verify integrity of ANProgram with ANProgram integrity verifier
        If failed result
                { abort and generate failed result message }
        Generate MD_OP = HashFunction_OP (ANProgram code)
        Generate DigitalSignature_OP = Encrypt (MD_OP + HashFunction_OP ID, OrigParty's
                PrivateKey) + ClearText (OrigParty's ID)
        Append DigitalSignature_OP to ANProgram code
        Generate message that ANProgram is prepared for compiling
        Return
        }
```

表 2

编译 ANProgram 和产生 ASProgram 方法的伪码表示

```
Procedure: Compile (ANProgram, CompParty's ID, ASLanguageID, CompParty's
    PrivateKey, Compiler's ID, and Compiler's PrivateKey)
    {
    Retrieve OrigParty's PublicKey from trusted key repository using ClearText
        OrigParty's ID in DigitalSignature_OP
    Decrypt (MD_OP + HashFunction_OP ID in DigitalSignature_OP, OrigParty's
        PublicKey)
    Generate TestMD_OP = HashFunction_OP (ANProgram code) using
        HashFunction_OP identified by decrypted HashFunction_OP ID
    Compare decrypted MD_OP and TestMD_OP
    If decrypted MD_OP ≠ TestMD_OP
        {
        /* DigitalSignature_OP of OrigParty not verified */
        Generate failed result message
        }
    Else
        {
        /* DigitalSignature_OP of OrigParty has been verified */
        Verify integrity of ANProgram with ANProgram integrity verifier
        If failed result
            {
            abort and generate failed result message
            }
        Else
            {
            /* ANProgram has been verified */
            Compile ANProgram code into ASLanguage identified by
                ASLanguage ID to generate ASProgram code
            Generate MD_C = HashFunction_CS (ASProgram code +
                DigitalSignature_OP)
            Generate DigitalSignature_C = Encrypt (MD_C + HashFunction_C ID,
                ANProgrom Compiler's PrivateKey) + ClearText
                ANProgram Compiler's ID
            Generate MD_CP = HashFunction_CP (ASProgram code +
                DigitalSignature_OP + DigitalSignature_C)
            Generate DigitalSignature_CP = Encrypt (MD_CP + HashFunction_CP
                ID, CompParty's PrivateKey) + ClearText CompParty's ID
```

```
                    Generate and Return File or Object containing:
                        ANProgram Code,
                        DigitalSignature_OP,
                        ASProgram Code,
                        DigitalSignature_C, and
                        DigitalSignature_CP
                    /* ASProgram has been compiled and generated */
                    }
                }
            }
```

表 3

执行 ASProgram 方法的伪码表示

```
Procedure: Execute (ObjectClass, Program)
    {
    If the Program is a verifiable program
        { Execute Program using the Bytecode Interpreter }
    Else
        { Execute Program using the compiled program executer }
    }

Procedure: ClassLoad (ObjectClass, Program)
    {
    If Object Class has already been loaded into ExecuterParty's address space
        {
        Call Execute (ObjectClass, Program)
        Return
        }

    /* The Object Class has not been loaded */
    Load Object Class into ExecuterParty's address space
    If Object Class was loaded from Trusted Object Class Repository
        {
        Call Execute (ObjectClass, Program)
        Return
        }

    /* Object Class was loaded from Untrusted Object Class Repository */
    If Object Class does not contain any ASPrograms designated as
        native_ASPrograms in Object Header of Object
        {
        Verify integrity of all ANPrograms of Object Class with ANProgram integrity
            verifier
        If failed result
            {
            Abort with appropriate failed result message
            }
        Else
            /* Integrity of all ANPrograms of Object Class have been verified */
            {   Call Execute (ObjectClass, Program)   }
```

```
        Return
        }

    /* Object Class does contain ASPrograms designated as native_ASPrograms in
        Object Header of Object */
    If any ASProgram does not contain a DigitalSignature_CP and a DigitalSignature_C
        {
        /* Compiling Party and Compiler of every ASProgram cannot be verified */
        Generate appropriate message
        Return
        }

    For each ASProgram in Object Class:
        {   Determine identity of CompParty and Compiler and determine
            ASLanguage used by ASProgram   }
    If identity of CompParty for any ASProgram is not a known, trusted, Compiling
        Party, or the identity of Compiler is not a known, trusted Compiler, or the
        identified ASLanguage is not one used by the ASProgram Executer
        {
        Generate appropriate message
        Return
        }

    For each ASProgram in Object Class:
        {
        Retrieve CompParty's PublicKey from trusted key repository using ClearText
            CompParty's ID in DigitalSignature_CP
        Decrypt (MD_CP + HashFunction_CP ID in DigitalSignature_CP, CompParty's
            PublicKey)
        Generate TestMD_CP = HashFunction_CP (ASProgram code + DigitalSignature_OP
            + DigitalSignature_C in ASProgram) using HashFunction_CP identified by
            decrypted HashFunction_CP ID
        Compare decrypted MD_CP and TestMD_CP
        }
    If decrypted MD_CP ≠ TestMD_CP for any ASProgram
        {
        /* DigitalSignature_CP for every ASProgram has not been verified */
        Generate appropriate failed result message
        Return
        }

    /* DigitalSignature_CP for every ASProgram has been verified*/
```

```
For each ASProgram in Object Class:
    {
    Retrieve ANProgram Compiler's PublicKey from trusted key repository using
        ClearText ANProgram Compiler's ID in DigitalSignature_C
    Decrypt (MD_C + HashFunction_C ID in DigitalSignature_C, ANProgram
        Compiler's PublicKey)
    Generate TestMD_C = HashFunction_C (ASProgram code + DigitalSignature_OP)
        using HashFunction_C identified by decrypted HashFunction_C ID
    Compare decrypted MD_C and TestMD_C
    }
If decrypted MD_C ≠ TestMD_C for any ASProgram
    {
    /* DigitalSignature_C for every ASProgram in Object Class has not been
        verified */
    Generate appropriate failed result message
    Return
    }

/* DigitalSignature_C for every ASProgram in Object Class has been verified */
For each ANProgram from which an ASProgram in Object Class was compiled:
    {
    Retrieve OrigParty's PublicKey from trusted key repository using ClearText
        OrigParty's ID in DigitalSignature_OP
    Decrypt (MD_OP + HashFunction_OP ID in DigitalSignature_OP, OrigParty's
        PublicKey)
    Generate TestMD_OP = HashFunction_OP (ANProgram code) using
        HashFunction_OP identified by decrypted HashFunction_OP ID
    Compare decrypted MD_OP and TestMD_OP
    }
If decrypted MD_OP ≠ TestMD_OP for any ANProgram
    {
    /* DigitalSignature_OP for every ANProgram from which an ASProgram in
        Object Class was compiled not verified */
    Generate failed result message
    Return
    }

/* The DigitalSignature_OP in every ASProgram in Object Class is verified */
Verify integrity of ANPrograms in Object class and ANPrograms from which
    ASPrograms in Object Class were compiled with ANProgram integrity verifier
If failed result
    {
```

```
        Generate failed result message
        Return
        }

/* Integrity of all ANPrograms in Object class and all ANPrograms from which
    ASPrograms in Object Class were compiled have been verified */
Call Execute (ObjectClass, Program)
}
```

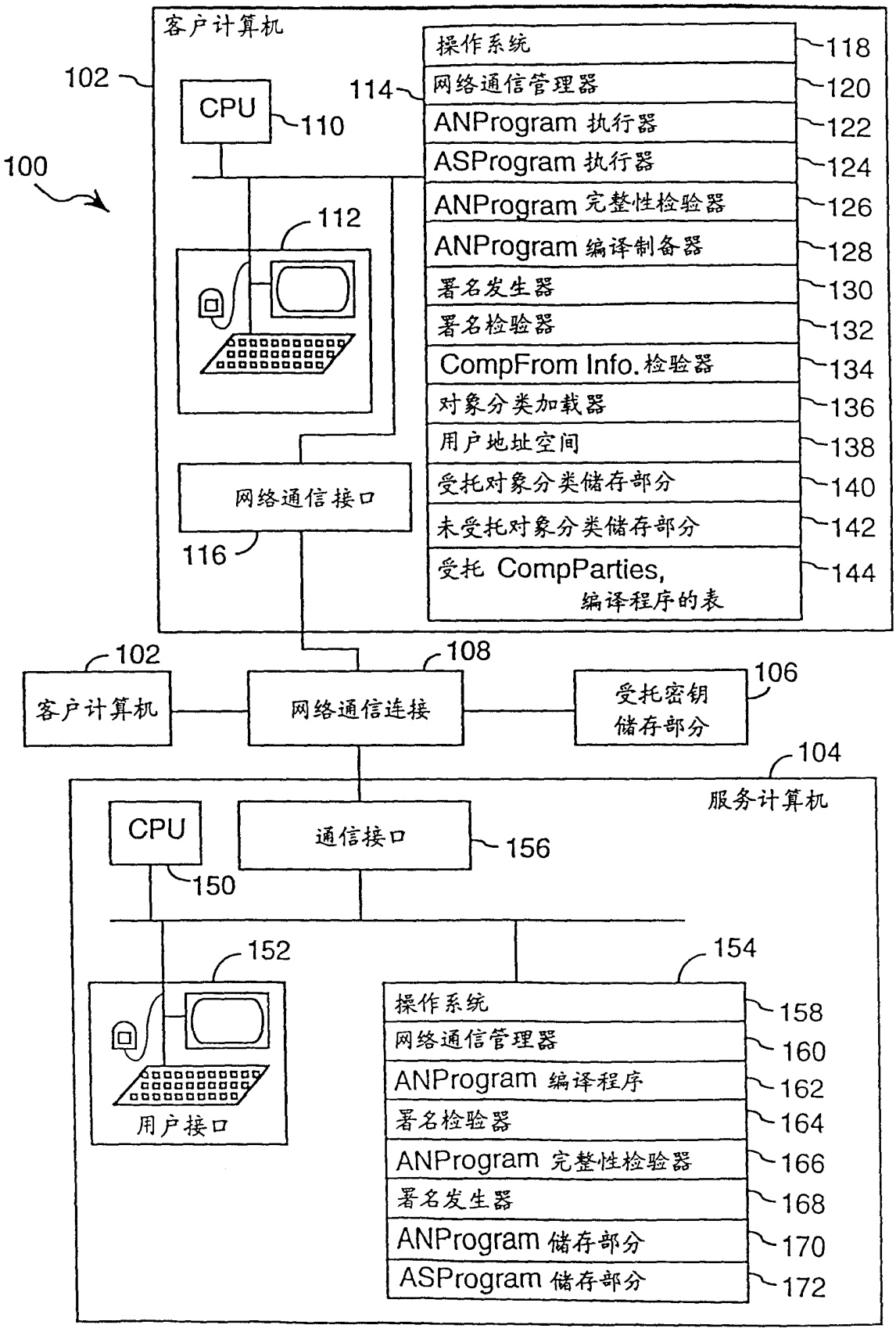

客户计算机
102
100
CPU
110
114
112
网络通信接口
116
操作系统
网络通信管理器
ANProgram 执行器
ASProgram 执行器
ANProgram 完整性检验器
ANProgram 编译制备器
署名发生器
署名检验器
CompFrom Info.检验器
对象分类加载器
用户地址空间
受托对象分类储存部分
未受托对象分类储存部分
受托 CompParties, 编译程序的表
118
120
122
124
126
128
130
132
134
136
138
140
142
144
102
客户计算机
网络通信连接
108
受托密钥储存部分
106
104
服务计算机
CPU
150
通信接口
156
152
用户接口
154
操作系统
网络通信管理器
ANProgram 编译程序
署名检验器
ANProgram 完整性检验器
署名发生器
ANProgram 储存部分
ASProgram 储存部分
158
160
162
164
166
168
170
172

图 1

图 2

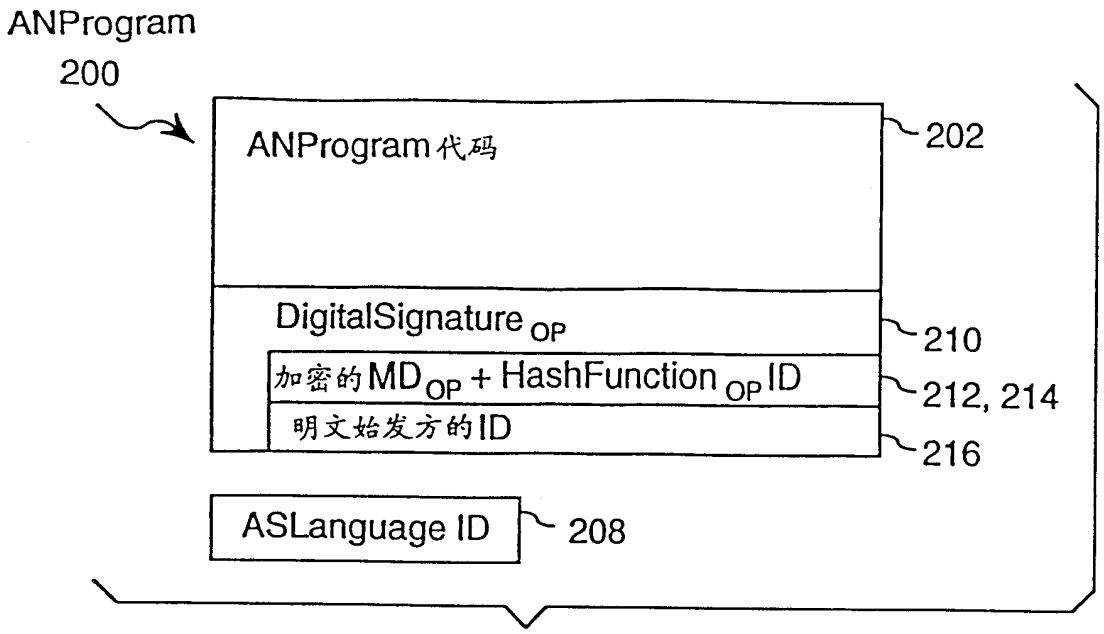
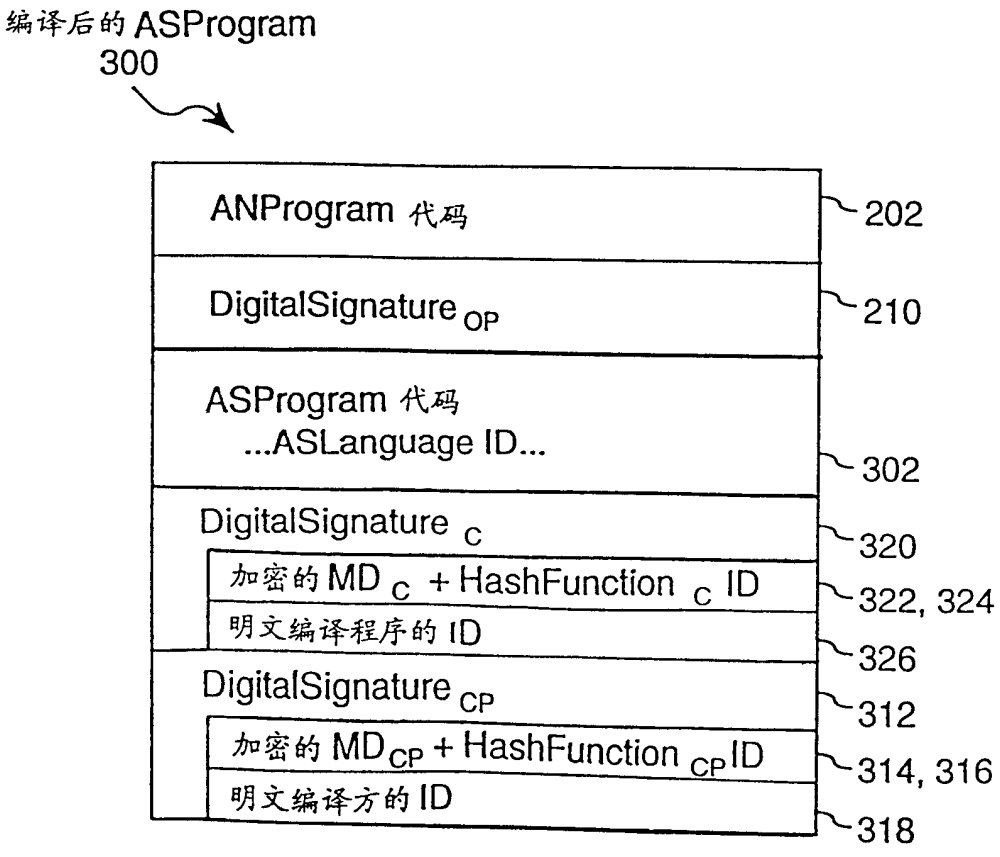

ANProgram
200
ANProgram 代码
202
DigitalSignature OP
210
加密的 MD OP + HashFunction OP ID
212, 214
明文始发方的 ID
216
ASLanguage ID
208

图 3


编译后的 ASProgram
300
ANProgram 代码
202
DigitalSignature OP
210
ASProgram 代码
...ASLanguage ID...
302
DigitalSignature C
320
加密的 MD C + HashFunction C ID
322, 324
明文编译程序的 ID
326
DigitalSignature CP
312
加密的 MD CP + HashFunction CP ID
314, 316
明文编译方的 ID
318

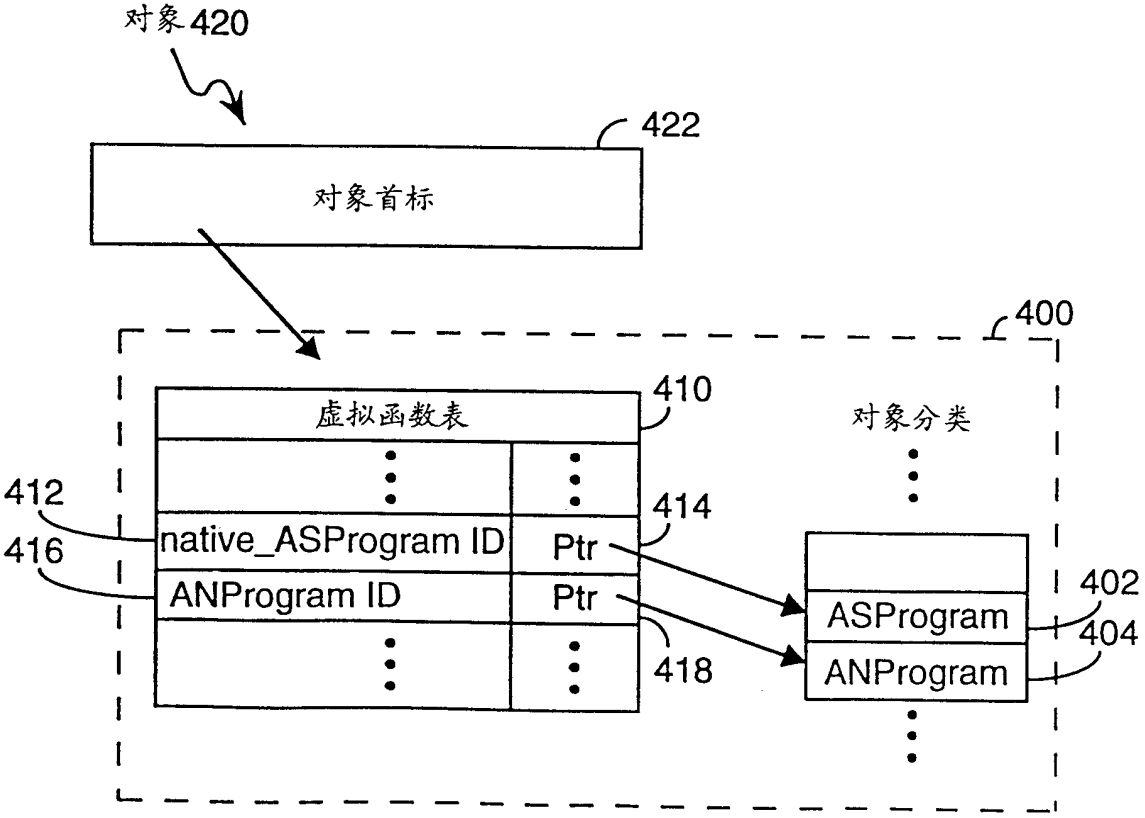

对象420
422
对象首标
400
虚拟函数表
410
对象分类
412
416
native_ASProgram ID
414
Ptr
ANProgram ID
Ptr
418
402
ASProgram
404
ANProgram

图 4