(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 101218588 B
(45)授权公告日 2010.05.19

(21)申请号 200680024593.9

(22)申请日 2006.05.05

(30)优先权数据
60/677,816 2005.05.05 US

(85)PCT申请进入国家阶段日
2008.01.04

(86)PCT申请的申请数据
PCT/CA2006/000711 2006.05.05

(87)PCT申请的公布数据
WO2006/116871 EN 2006.11.09

(73)专利权人 塞尔蒂卡姆公司
地址 加拿大安大略

(72)发明人 A·韦德卡 B·尼尔

(74)专利代理机构 中科专利商标代理有限责任公司 11021
代理人 王波波

(51)Int.Cl.
*G06F 21/00*(2006.01)
*G06F 12/00*(2006.01)

(56)对比文件
US 2004268339 A1,2004.12.30, 全文.
CN 1454338 A,2003.11.05, 全文.
EP 849657 A1,1998.06.24, 摘要、说明书第2页第41行-第3页第49行.
WO 0018162 A1,2000.03.30, 摘要、附图2.
US 4849927 A,1989.07.18, 全文.

审查员 崔哲

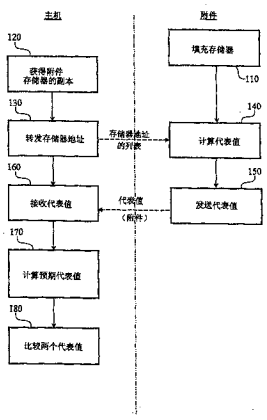
权利要求书 2 页 说明书 6 页 附图 3 页

(54)发明名称

具有可认证的固件的附件设备及布置和认证该固件的方法

(57)摘要

本发明提供了一种廉价的、基于软件的安全更新方案，以验证嵌入式系统中或附件中的程序代码的完整性，而不必诉诸昂贵的硬件改动。附件上所有可用来存储程序代码镜像的未使用的存储器都被填充随机数据。主机系统也在本地存储一个该附件的包括该随机数据的程序镜像的副本。该主机系统向该附件发送一个该附件上的存储器地址列表或存储器范围，其本质上总是随机的和不相同的。然后该附件将使用存储在该存储器地址中的值作为安全散列函数的输入产生一个摘要。主机系统通过验证由附件产生的并且从附件返回的结果摘要，来验证该嵌入式程序代码的完整性。




CN 101218588 B

1. 一种用于验证嵌入在附件设备中的程序代码的完整性的方法，所述附件设备具有一个用于存储该程序代码和与该程序代码相关的数据的存储器设备，该存储器设备中所有未使用的存储器空间被填充随机数据，主机可以访问该存储器设备的存储器镜像，所述方法包括如下步骤：

发送一个存储器地址列表到该附件设备；

从该附件设备接收一个值，所述值是根据所述存储器地址列表中的指定存储器地址处的存储器值生成的，并且代表了所述存储器地址列表中的指定存储器地址处的存储器值；

根据存储器镜像和所述存储器地址列表产生一个预期值；

将从该附件设备接收的所述值与所述预期值进行比较；

其中如果所述接收的值与所述预期值等价，则该程序代码的完整性得到验证。

2. 权利要求 1 的方法，其中所述值是一个在所述存储器地址列表中指定的序列中的所述指定存储器地址处的所述存储器值的连接。

3. 权利要求 1 的方法，其中所述值是一个用安全散列算法计算的摘要，该算法使用在所述指定存储器地址处的存储器的所述值作为其输入，且所述预期值是一个使用第二安全散列算法计算的预期摘要，该第二安全散列算法与所述安全散列算法等价。

4. 权利要求 3 的方法，还包括如下步骤：

向该附件设备发送一个随机数据字符串，

其中该摘要是通过把所接收的随机数据字符串作为附加输入计算的，且该预期摘要是把该随机数据字符串作为第二输入产生的。

5. 一种为认证附件设备的固件程序代码而在该附件设备的存储器空间中布置存储器存储的方法，所述固件程序代码和与所述固件程序代码相关的数据均被存储在所述附件设备的存储器设备中，所述方法包括下述步骤：

在所述存储器空间的第一存储器段中为所述固件程序代码提供一个用来从主机接收一个地址列表的第一 API，所述地址列表中的地址与所述固件程序代码可寻址的存储器地址相应，

在所述存储器空间的第二存储器段中为所述固件程序代码提供一个第二 API；

在所述存储器空间的第三存储器段中为所述固件程序代码提供一个用于根据所述地址列表和在所述地址列表中指定的所述地址处的存储器的值产生一个代表值的密码程序，

所述第二 API 将所述代表值返回所述主机，

给所述存储器设备中所有未被所述固件程序代码和所述与所述固件程序代码相关的数据占据的存储器空间填充随机数据，以及

在外部存储器设备中提供一个所述存储器设备的存储器镜像，用于参考，

其中如果在认证操作中所述代表值与预期代表值等价，则所述固件程序代码被认证，所述预期代表值是由所述主机使用所述地址列表根据所述存储器镜像的一个副本计算的。

6. 权利要求 5 的方法，其中所述代表值是一个使用安全散列程序计算的摘要。

7. 权利要求 6 的方法，其中所述预期代表值是一个使用与所述安全散列算法等价的第二安全散列算法计算的预期摘要。

8. 权利要求 7 的方法，还包括步骤：

向该附件设备发送一个随机数据字符串，

其中该摘要是通过把所接收的随机数据字符串作为附加输入计算的，且该预期摘要是把该随机数据字符串作为第二输入产生的。

9. 一种具有可认证的固件程序代码的附件设备，所述固件程序代码和与所述固件程序代码相关的数据被存储在所述附件设备的存储器设备中，所述存储器设备中未被所述固件程序代码和所述与所述固件程序代码相关的数据占据的所有存储器空间全部被随机数据所占据，所述附件设备具有所述存储器设备的外部存储的存储器镜像，所述存储器设备中被所述固件程序代码占据的存储器空间包括：

第一存储器段，所述第一存储器段被一个用于从主机接收一个地址列表的 API 占据，所述地址列表中的地址与所述固件程序代码可寻址的所述存储器设备上的存储器地址相应；

第二存储器段，所述第二存储器段被一个用于根据所述地址列表和在所述地址列表中指定的所述地址处的存储器的值产生摘要的安全散列程序占据；以及

第三存储器段，所述第三存储器段被一个用于将所述摘要返回所述主机的第二 API 占据，

其中所述主机使用所述地址列表根据所述外部存储的存储器镜像的一个可信任副本产生一个预期摘要，并将所述预期摘要与从所述固件程序代码返回的所述摘要进行比较，其中如果从所述固件程序代码返回的所述摘要与所述预期摘要等价，则所述固件程序代码被认证。

10. 一种具有可认证固件程序代码的附件设备，所述固件程序代码和与所述固件程序代码相关的数据被存储在所述附件设备的存储器设备中，所述附件设备具有所述存储器设备的外部存储的存储器镜像，所述存储器设备中被所述固件程序代码占据的存储器空间包括：

第一存储器段，所述第一存储器段被一个用于从主机接收输入和将一个值返回该主机的接口占据；

所述输入包括由该主机选择的存储器地址；

第二存储器段，所述第二存储器段被一个用于产生该值的程序占据，该值是所述输入和在所述存储器设备上的所述所选择的地址处的存储器的值的函数；

其中所述主机使用所述输入根据所述外部存储的存储器镜像的一个可信任副本产生一个预期值，并且将所述预期值与从所述固件程序代码返回的所述值进行比较，其中如果从所述固件程序代码返回的所述值与所述预期值等价，则所述固件程序代码被认证。

11. 权利要求 10 的附件设备，其中所述用于产生该值的程序是一个安全散列程序，所述值是一个由所述安全散列程序产生的摘要。

12. 权利要求 10 的附件设备，其中所述输入是一个由该主机生成的随机值。

# 具有可认证的固件的附件设备及布置和认证该固件的方法

## 发明领域

[0001] 本发明一般涉及密码学领域。具体而言，本发明涉及提供一种价格低廉、基于软件的安全更新解决方案，用于验证嵌入式系统中的程序代码的完整性。

## 背景技术

[0002] 计算机系统通常使用外围设备来补充其功能。在此处上下文中，计算机系统被称为主机系统，而其外围设备被称为附件。附件通常是能够进行计算的设备，因为它们通常是使用微处理器或微控制器构建的，这些微处理器或微控制器可以利用程序代码、微码或固件进行编程或重编程。这些附件的功能和正确运作均依赖于位于该附件中的程序代码的正确性。

[0003] 有些情况下，因为在程序代码中发现了缺陷，需要升级或更新已实施的附件。例如，有必要对在诸如医疗设备之类的高价值应用中的高风险仪器进行升级。这些类型的设备通常由政府来管制。当发现缺陷并进行重新设计时，可以使用快速的临时解决方案。因此，将这样的解决方案应用到已经配置在实地的设备，会是理想的。

[0004] 个体可能会恶意改动附件的程序代码，以使其执行从主机系统角度来说是未授权的操作，却通过报告行为正常来欺骗主机系统相信没有出差错。理想的是任何对附件程序代码的未授权修改可以在进行升级之前被检测到，且只有在未检测到此类未授权修改的时候才进行升级。

[0005] 对此问题的一个解决方案是，在实施附件之前，例如通过使用安全微控制器在附件中设计安全措施，该安全微控制器在接受微码升级之前对所有微码升级进行认证。然而有时情况是这样，当首次实施附件的时候，安全上的考虑和风险级别都较低，但随着时间的过去，情况变化了，以至于与不安全附件相关的风险不可预期地增加了。对附件进行替换或对附件作出硬件改变的成本可能被认为太过昂贵，在这种情况下，主机系统在高危环境中具有不安全的附件。

[0006] 对此问题的不成熟的解决方案是允许该主机系统从附件读取全部程序代码，使用安全散列函数计算摘要，并且将结果与某个本地维持的摘要进行比较。这种方法有两个问题。首先，该附件可能会被重新编程，以使之维持原程序代码的副本，篡改来自该主机系统的读取请求，并且向该主机系统返回所存储的原程序代码镜像。其次，存储在附件中的程序代码可能太大而无法有效地通过慢速串行连接来传输，或可能甚至不能被远程访问。在这些情况下，计算一个返回到主机系统用于验证的摘要更有效率。

[0007] 简单地计算一个摘要，即使使用挑战－响应机制，也许都不能有效地克服一个被改变的附件，该附件在其存储器中的某处维持了原程序代码的副本。

[0008] 本发明的一个目的是减轻或消除上述缺点中的至少一个。

## 发明内容

[0009] 一般地，本发明包括用高熵随机数据填充附件设备上的所有可以被用于存储程序

代码镜像的未使用的存储器。主机系统也必须能够访问该附件的包括该随机数据的存储器镜像的可信任副本。

[0010] 附件中现有的程序具有一个可被主机系统调用的应用程序接口 (API)。如果该附件尚未具有一个额外 API,可以在需要包括一个额外 API 时重写该附件。该主机系统使用该 API 向该附件发送一个该附件上的存储器地址列表或存储器范围。该列表总是不同的,并且确实是不可预测的,以防止该列表被该附件预先猜测到。然后,该附件将产生一个代表值,该代表值是根据该列表和该列表所涉及的附件设备存储器的值确定的。优选地,该代表值是一个摘要,该摘要使用提供的存储器地址处的存储器值作为安全散列函数的输入。由该附件产生的结果摘要被返回到主机系统用于有效性验证。

[0011] 通过将存储器填充随机数据,防止了攻击者在附件上拥有足够空间来替换程序镜像并维持旧的程序镜像。攻击者不能通过仅替换程序镜像而不先向该附件增加额外存储器来改变该附件,这使得攻击者需要对硬件作出改变。通过向存储器填充随机数据,从而在认证过程中隐含包括该随机数据,进一步防止了攻击者对程序镜像做出未授权的改动而不被发现。

[0012] 最终结果是,该主机系统能够一定程度地确信该附件的程序代码未被恶意改变。

## 附图说明

[0013] 为了说明,而不是限制,通过示例的方式参见附图更详细地解释实施方案,其中 :

[0014] 图 1 是示出了与附件进行通信的主机系统框图 ;

[0015] 图 2 是图解了用于验证图 1 中所示的附件中的程序代码的完整性的方法的步骤的流程图 ;

[0016] 图 3 示意性示出了在图 1 中所示的附件上的填充了随机数据的未使用的空间以及在该附件上由该主机系统随机选定的存储器的范围。

## 具体实施方式

[0017] 下面的描述以及在此描述的实施方案都是以对本发明原理的具体实施方案的一个实施例或多个实施例的说明的方式来提供。这些实施例是为了解释而不是为了限制那些原理和本发明而提供的。在下面的描述中,相同的部分在整个说明书中和附图中都各自用相同的参考数字标注。

[0018] 参见图 1,示出了与附件 22 进行通信的主机 20。主机 20 通常是具有 CPU 24、可由 CPU 24 访问的主机存储器设备 26、也可由 CPU 24 访问的主机存储媒介 28 以及一些输入和输出设备 ( 未示出 ) 的计算机系统。应用程序 30 在 CPU 24 上执行。应用程序 30 可被存储在主机存储媒介 28 上,其中主机存储媒介 28 可被永久性地安装在主机 20 之中、可从主机 20 中移除或被主机 20 远程访问。应用程序 30 与附件 22 通信,并指示它的操作。

[0019] 附件 22 一般具有某些计算能力。通常,它具有一个微处理器或一个微控制器 32。附件 22 也可以具有其他类型的具有足够计算能力的可编程处理器。比如,附件 22 可以是配备有 DSP( 数字信号处理器 ) 或 FPGA( 现场可编程门阵列 ) 的附件。该处理器或微控制器 32 通常可以访问存储器存储空间,该存储器存储空间可以分为附件存储器设备 34 以及易失性存储器设备 36。固件程序代码 38 或嵌入式程序代码或微码在微处理器 32 上执行,

且补充在主机 20 上执行的应用程序 30，以进一步控制附件 22 的操作。

[0020] 固件程序代码 38 和永久性（恒定）数据可以存储在附件存储器设备 34 中，或任何永久性媒介中。易失性数据，即涉及该程序操作状态的数据，可以存储在易失性存储器设备 36 上。附件存储器设备 34 或易失性存储器设备 36 也可以用来存储半永久性数据，即在供电周期 (power cycle) 之间恒定，但可在每个供电周期中被编程修改的数据。

[0021] 在需要时，数据链路 40 在应用程序 30 和固件程序代码 38 之间提供一个通信信道。数据链路 40 可以是有线的或无线的。比如，它可以是一个接线电缆或一个射频连接。它可以是一个在主机 20 与附件 22 之间的直接连接或是通过某些中间主机系统的中继连接。数据链路 40 可以是一个永久性的，或更优选地是一个应要求创建的连接。

[0022] 应用程序 30 和固件程序代码 38 每一个均具有应用程序接口 (API)，用于彼此进行通信。具体地说，如随后将描述的，固件程序代码 38 具有：一个 API，应用程序 30 调用该 API 来转发一个存储器地址列表；一个 API，通过该 API，固件程序代码 38 返回一个代表值，例如基于一个存储器地址列表的内容计算出的摘要。尽管从概念上讲，固件程序代码 38 被描述为具有两个独立的 API，这两个独立的 API 在实践中可以被实现为一个单 API。应用程序 30 也具有用于发送该存储器地址列表和用于接收返回的代表值的相应 API。

[0023] 在此处虽然作出了区分，一个主机存储器设备 26 倾向于用于存储更多易失性数据，而一个主机存储器媒介 28 倾向于用于存储更多永久性数据，但是主机 20 可以只具有一个用于既存储易失性数据又存储永久性数据的单一数据存储器设备。

[0024] 可以理解，主机 20 可以是一个通用计算机、一个自定义的专用计算机或者其它可编程计算设备。此外，尽管主机 20 被图示和描述为单个计算机系统，但这仅为方便描述。主机 20 应被整体理解为组合系统，它可以包括几个用于如所述执行任务的计算机系统。本领域技术人员将理解，主机 20 执行的计算和存储可以分布在多个联网计算机上，而不影响系统的功能和在此描述的方法的性能。

[0025] 参见图 2 和图 3，详细描述用于验证固件程序代码 38 的完整性的方法的实施例。在图 2 中，虚线的左边是通常由主机 20 或在主机 20 的组件上执行的操作。在图 2 中，虚线的右边是通常由附件 22 或在附件 22 的组件上执行的操作。

[0026] 通常，固件程序代码 38 的大小比附件存储器设备 34 中可用的存储器的大小要小一些。换句话说，附件存储器设备 34 通常具有未使用的存储器空间，即未被固件程序代码 38 或与固件程序代码 38 相关的数据占据的空间。为了限制对手使用未使用的存储器空间来存储任何未授权代码的能力，在验证之前，所有未使用的存储器空间均用随机数据填充。随机数据优选地具有高熵。高熵随机数据趋向于更难以压缩，而具有随机性却具有低熵的数据趋向于能够被良好压缩。优选地，附件存储器设备 34 上的所有存储器空间被不可压缩的数据所占据。用不可压缩的数据填充所有存储器空间，使得对手不能压缩数据而获得存储器空间。有许多随机数生成算法可用于生成高熵随机数据。例如，可以使用由美国国家标准与技术研究院 (NIST) 发布的“DigitalSignature Standard（数字签名标准）”，在联邦信息处理标准 186-2 版的“Change Notice 1（变化通知 1）”中详述的随机数字生成算法。此外，阻止对低熵数据进行加密，有利于产生高熵的输出。

[0027] 在步骤 110 中，附件设备 22 上的所有未使用的存储器均被填充随机数据，且检索被填充的存储器镜像的一个可信任副本以便随后引用或使用。尽管该步骤可以由附件设备

22 执行，但是存在允许利用随机生成的高熵数据填充所有未使用的存储器的现有工具。通常，数据的随机性是由用来填满未使用的存储器空间的工具来控制的。优选地，存储器填充是在该附件被投入市场之前，在开发场所或加工点执行的。

[0028] 在使用随机数据进行填充之后，附件 22 上所有的存储器空间均被占据。参见图 3，附件存储器空间 200 可以被分成相连的程序代码和永久性数据空间 202 以及未使用的存储器空间 204。未使用的存储器空间 204 被填充了随机数据 206。尽管图 3 示出固件程序代码 38 占据了一个相连的存储器段，即一个相连的程序代码部分和永久性数据空间 202，但是也可能固件程序代码 38 可以占据数个不相连的存储器段。相似地，永久性（恒定）数据可以占据一个相连的存储器段或者数个不相连的存储器段。无论程序代码和永久性数据空间 202 是否相连，所有未使用的存储器空间将都被随机数据填充或填满。

[0029] 将理解，附件存储器空间 200 可以包括在易失性存储器设备 36 上被易失性数据占据的存储器空间。在有效性验证过程中不改变（或以一种定义好了的形式改变）的存储器单元可以被有效性验证。附件存储器空间 200 也可以包括被诸如串形 EEPROM 之类的“外围”存储器设备上的数据占据的存储器空间。事实上，附件存储器空间 200 甚至还可以包括附件存储器设备 34 和易失性存储器设备 36 的物理空间之外的部分（但是这些地址都被作为物理地址的影子而返送）。当然，这些都要求附件 22（和主机 20）支持可以寻址任何和全部可用存储器空间的存储器寻址方案，并且支持特殊存储器配置，例如内存库切换 (bank-switching)、覆盖以及影子。事实上，不同的存储器寻址方案和特殊的存储器配置可以用来进一步增强安全性。例如，当已有效性验证的存储器地址相应于该物理存储器空间之外的存储器空间时，对于要求兼容的但较大的存储器占用量的被修改程序，当运行在必须较大的设备上时，对原程序和较小的设备的有效性验证过程进行模拟将会比较困难。

[0030] 检索附件 22 上的存储器的镜像，其中没有剩下未占据的存储器空间，以便随后引用或使用。以可信任的方式生成并且保存该镜像。例如，可以在一个可信任操作中从附件 22 获得该镜像，所述可信任操作例如在附件 22 的加工点的编程。也有可能，该工厂大批量地对附件镜像进行编程，而在别处生成镜像，该别处为在还生成主机代码和数据镜像的开发设施处。不管该镜像是如何生成的，都以可信任的方式产生并提供该镜像。当该镜像被保存时，该镜像也以可信任的方式被保存。最终结果是可以在需要的时候获得一个可信任的镜像的副本。

[0031] 在步骤 120，主机 20 获得在附件 22 上的存储器的镜像的可信任副本，没有剩余未占据的存储器空间。通常，为了最小化镜像被调节的风险，主机 20 不在本地存储可信任镜像。而是主机 20 被设置为访问该镜像的可信任副本。尽管此处仅有一个可信任的存储器镜像被引用，但是可能有多个不同的存储器镜像必须被设为对于主机 20 可用。这是因为随着时间的过去，附件 22 可能已被其生产者升级了多次。先前的每个升级都会导致一个不同的存储器镜像。当附件 22 在现场被升级时，附件 22 可以相应于这几个升级中的任何一个，或者甚至其最初版本。

[0032] 将理解，尽管本步骤被描述为紧随在保存该存储器镜像的可信任副本之后的步骤，但是这两个步骤可能相隔许多天、许多月或甚至许多年。有可能该存储器镜像的该可信任副本是在附件制造过程中保存的，而对该镜像的该可信任副本的检索在多年以后当对现场部署的附件进行修理的时候才发生。此外，尽管这一步骤被描述为在主机侧的第一步骤，

但这不是必须的。本步骤需要在计算预期代表值之前完成，且可以在计算之前的任何时间执行。

[0033] 参见图 2，主机 20 在步骤 130 向附件 22 发送一个存储器地址的列表，以初始化该验证过程。该列表可以是在附件存储器设备 34 上的一个存储器地址范围或者几个存储器地址范围。例如，图 3 示出了三个随机选择的存储器段，或存储器范围：一个第一存储器范围 208 或存储器段 A，在第一起始地址 210 和第一终止地址 212 之间；一个第二存储器范围 214 或存储器段 B，在第二起始地址 216 和第二终止地址 218 之间；以及一个第三存储器范围 220 或存储器段 C，在第三起始地址 222 和第三终止地址 224 之间。

[0034] 该列表或该存储器地址的范围本质上是随机的，或者至少是不可预测的，以致该范围不能被附件 22 预知，也就是说，该列表不能被试图对在附件 22 上存储的固件代码作恶意改变的对手预知。不仅该起始点和终止点的选择是不可预测的，而且所选存储器范围的顺序也是不可预测的。例如，该列表可以包括作为第一范围的段 C 的地址，作为第二范围的段 A 的地址，和作为第三范围的段 B 的地址。改变所选的存储器范围的顺序会产生一个不同的列表，也会在步骤 140 产生一个不同的代表值，如下文所述。该列表通常是由主机 20 生成的。然而，该列表可以以任意方式生成，只要该列表是真正不可预知的并且每次生成时是不同的。因为主机 20 通常具有更多计算能力，所以主机 20 通常生成该列表并发送到附件 22。

[0035] 在一个范例实现中，包括在列表中的存储器地址范围之一总是包括被认为对于附件 22 的运行起关键作用的程序代码。换句话说，主机 20 总是包括包含应用程序 30 的整个关键代码的存储器段。尽管包含关键代码的存储器总是包括在所选存储器范围内，但是如上所述，包括了该关键代码的存储器段可以被随机布置在列表内。

[0036] 一旦接收到该存储器地址的列表时，附件 22 在步骤 140 以存储器地址列表和在提供的地址处的存储器值作为输入产生一个值。所生成的该值是在提供的地址处的存储器值的代表值。倘若产生的值是该列表和所提供的存储器地址处的实际值的代表值，则许多算法可以用来产生该值。例如，可以首先读取在提供的存储器地址处的存储器的值，然后连接到一起成为一个字符串。那么该字符串将成为代表值。优选地，该代表值是一个使用安全散列算法计算的摘要。该安全散列算法使用提供的存储器地址处的存储器值作为输入，计算该摘要。可以使用任何具有 SHA（安全散列算法）的安全属性的计算摘要算法。比如，该安全散列函数可以是基于 SHA-1 或 MD5 的函数。优选地，可以使用具有已经存在于附件实现中的 SHA 的安全属性的计算摘要算法；另外，可以使用依赖于在附件 22 的特定处理器类型的效率的 SHA-1 或 SHA-256 之一。附件 22 在步骤 150 返回（即发送）结果代表值给主机 20，该结果代表值是由主机 20 在步骤 160 接收。

[0037] 图 3 中所示的实施例具有：第一存储器范围 208，相应于完全被永久性数据和固件程序代码 38 占据的存储器空间；第三存储器范围 220，相应于被随机数据填充的未使用的存储器空间 204；以及第二存储器范围 214，相应于部分包含固件程序代码 38 和部分包含随机数据的存储器部分。将理解其它选择也是可能的。重要的是，列表的可能集合（可以被主机 20 所请求的）要足够大，并且地址的选择是真正不可预测的。这往往阻碍预先计算出所有（或者甚至是一个有效子集的）相应代表值，因为从存储和计算角度，该计算可能会很昂贵或甚至不可行。

[0038] 固件程序代码 38 可以具有实现这些安全散列算法中的一个或几个的模块。在只实现一个安全散列算法的情况下，该附件使用已实现的安全散列函数，以所接收的存储器范围作为输入，计算摘要，并将结果摘要发送到该主机系统。在固件程序代码 38 实现多于一个安全散列算法的情况下，使用已实现的安全散列算法之一产生一个摘要。这个结果摘要与所使用的安全散列算法的指示一起被发送到主机 20。将理解，主机 20 或附件 22 可以选择一个特定的安全散列算法，并且通知用于产生摘要的其它安全散列算法。

[0039] 在更新固件时，该固件程序代码 38 可以不具有用于接收该随机生成的存储器地址的列表的 API。它还可以不具有使用安全散列算法来产生摘要的模块。为了准备对这样的附件进行安全升级，即升级只在该固件代码可被认证时才执行，有必要首先将认证功能更新到附件上。换句话说，有必要首先重写附件 22 上的现有程序，以包括一个可被主机 20 调用来接收该存储器地址列表的附加 API，以及一个可向该主机返回根据列表计算的代表值的附加API。重写的固件程序也将包括一个用于计算代表值的模块，或用于实现计算摘要的安全散列算法的一个或多个模块。

[0040] 在主机 20 在步骤 160 接收到来自附件 22 的结果摘要或该代表值之后，主机 20 在步骤 170 中使用该附件的存储器镜像的可信任副本来计算一个预期代表值。优选地，主机 20 使用与附件 22 使用的相同的算法来产生预期代表值。但是这并非必须的。主机 20 使用的算法只需要等价于附件 22 所使用的算法，以使得预期代表值与接收的代表值相同。如果附件 22 实现多于一个的安全散列算法，但是只使用一个来计算摘要，则主机 20 选择相同的或等价的算法来计算该预期代表值。

[0041] 在步骤 180，主机 20 将从附件 22 接收的代表值与本地计算的预期代表值进行比较，以验证该附件程序镜像。如果这两个值彼此不相同或不等价，则固件程序代码 38 将不被认证。

[0042] 由上所述，本方法的第一步骤是保证附件上的所有存储器都被占据。如果用于实现该附件的功能性的经过编译的程序比附件上的物理存储器小，则剩余部分就被真正随机且高熵的数据所填充。该填充数据变成在更新升级过程中可被该主机系统访问的存储器镜像的一部分。

[0043] 在另一个范例实现中，主机 20 在步骤 130 把一个数据的字符串与该存储器地址的列表一起发送到附件 22。该字符串可以是随机的，或者可以包括一个识别信息，例如附件（或甚至该主机系统）的唯一身份。当该字符串被用作对该安全散列函数的辅助输入时，该方法通常被称为挑战－响应方法。然而，它仍旧依赖于存储器镜像中的随机填充数据，以防止将非法程序代码注入附件中。

[0044] 现在已详细描述了各个实施例。本领域技术人员将理解，在不背离本发明范围的情况下，可以对这些实施例作出许多更正、修改和变化。由于在不背离本发明的本质、精神和范围的情况下可以对上述最佳实施方式做出改变和 / 或增加，所以本发明不限于这些细节，而只被所附的权利要求书所限定。

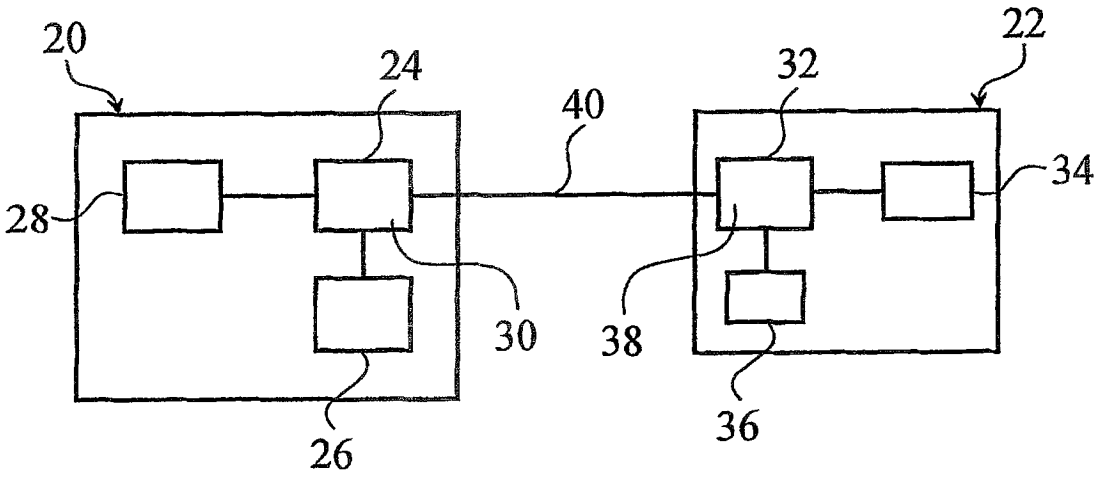



图 1

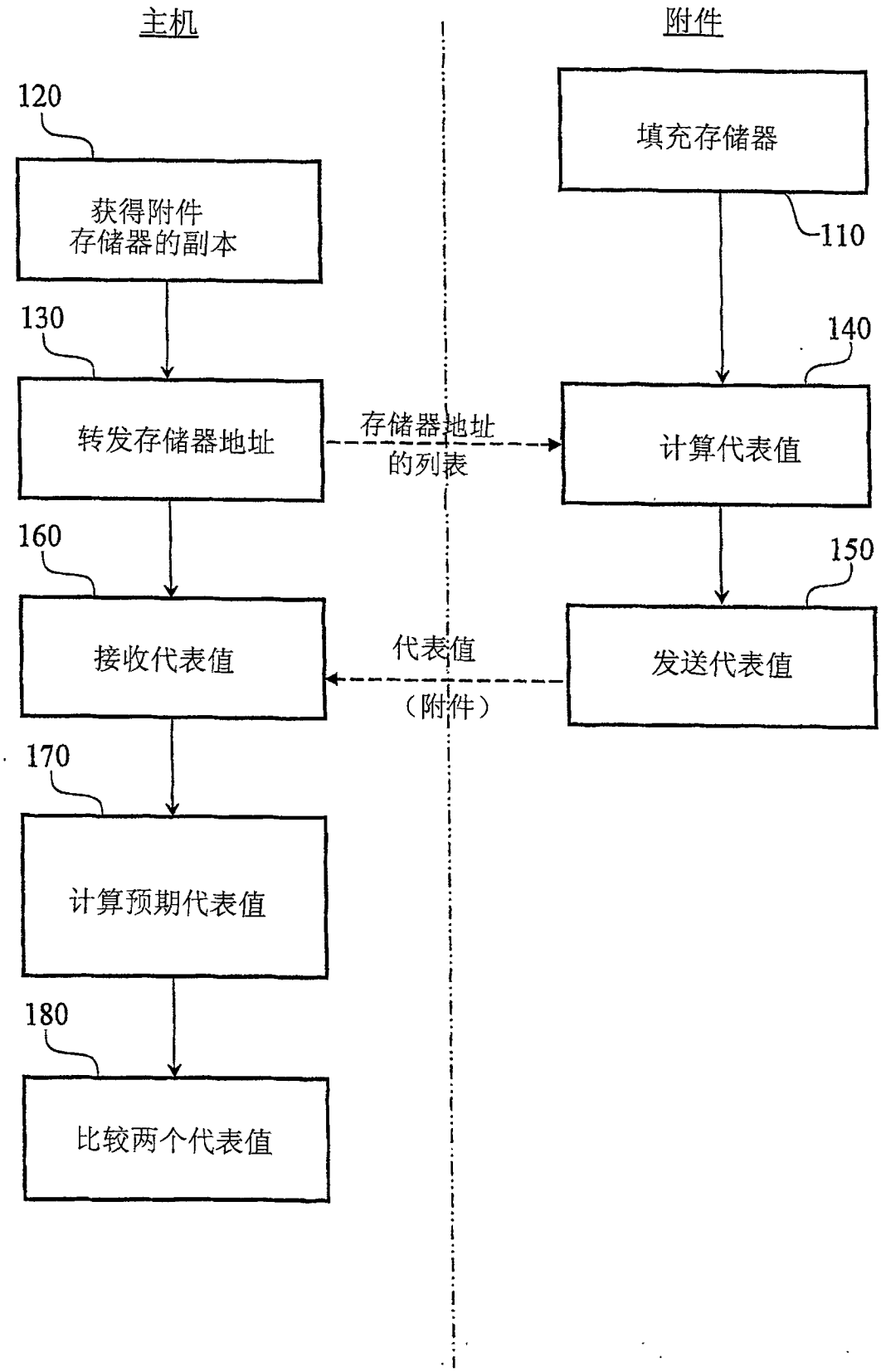

主机
附件
120
获得附件
存储器的副本
130
转发存储器地址
存储器地址
的列表
填充存储器
110
140
计算代表值
160
接收代表值
代表值
（附件）
150
发送代表值
170
计算预期代表值
180
比较两个代表值

图 2

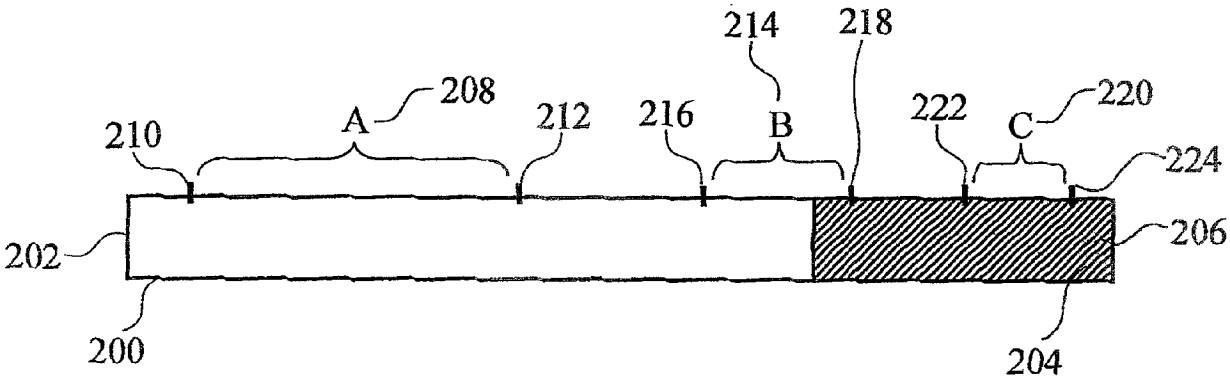



图 3