

[19]中华人民共和国专利局

[51]Int.Cl<sup>6</sup>



[12]发明专利申请公开说明书

G07F 7/10

G07F 19/00 G06F 17/60

[21]申请号 96197307.2

[43]公开日 1998年11月4日

[11]公开号 CN 1198233A

[22]申请日 96.9.26

[74]专利代理机构 上海专利商标事务所

[30]优先权

代理人 钱慰民

[32]95.9.29 [33]US[31]60 / 004,510

[32]96.1.31 [33]US[31]08 / 594,983

[86]国际申请 PCT / US96 / 15471 96.9.26

[87]国际公布 WO97 / 12344 英 97.4.3

[85]进入国家阶段日期 98.3.27

[71]申请人 达拉斯半导体有限公司

地址 美国得克萨斯州

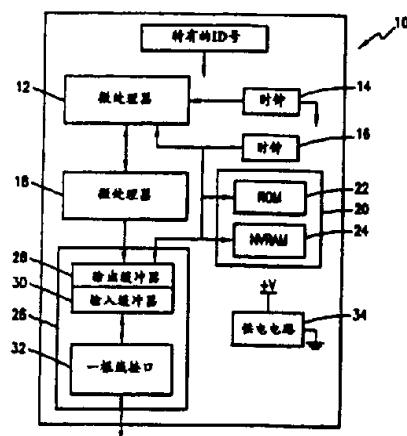
[72]发明人 斯蒂芬·M·柯里 唐纳德·W·鲁密斯  
克里斯多夫·W·福克斯

权利要求书 3 页 说明书 41 页 附图页数 8 页

[54]发明名称 保密交易的方法、设备、系统和固件

[57]摘要

本发明涉及一种用于进行保密交易的电子模块。具体地说，该电子模块能够通过一保密的加密技术与服务供应设备来往传递信息，从而可以用电子方式保密地传送货币和其它有价值的数据。模块可以被编程，对实时保持跟踪，记录交易供以后查询，并且能够产生加密密钥对。



# 权 利 要 求 书

1. 一种用于保密交易的电子模块，其特征在于，包括：  
输入/输出电路，用于与数据处理电路连通；  
数学协处理器电路，它与所述输入/输出电路电气连接；  
微处理器电路，它与所述输入/输出电路电气连接；和  
存储器电路，它与所述微处理器电路电气连接，所述电子模块是可编程的，用以在所述电子模块和所述数据处理电路之间进行保密的加密数据传输。
2. 如权利要求 1 所述的电子模块，其特征在于，所述数据处理电路是另一个电子模块。
3. 如权利要求 1 所述的电子模块，其特征在于，还包括一个单线接口，它与所述输入/输出电路相连。
4. 如权利要求 1 所述的电子模块，其特征在于，所述存储器电路适于存储在所述电子模块与所述数据处理电路之间进行加密数据传输期间使用的秘密加密/解密密钥。
5. 如权利要求 1 所述的电子模块，其特征在于，所述加密交易是作了时间标记的。
6. 一种用于保密交易通信的系统，其特征在于，包括：  
第一模块，它包括：  
输入/输出电路；  
随机数据发生装置，用于产生随机数；和  
第一交易组，用于请求所述随机数发生装置产生所述随机数，并且用于向所述输入/输出电路提供所述随机数；和  
服务供应设备，它包括：  
用于从所述第一模块的所述输入/输出电路读取所述随机数的装置；  
用于把所述随机数与第一数据合并并且用秘密密钥对所述随机数与所述第一数据的合并加密以产生第一证明的装置，由此所述第一模块的所述输入/输出电路适于接收所述第一证明。
7. 如权利要求 6 所述的系统，其特征在于，所述服务供应设备包括第二模块。
8. 如权利要求 6 所述的系统，其特征在于，所述第一模块还包括一个标识

符，用于识别所述第一模块，并且所述第一交易组将所述标识符提供给所述输入/输出电路。

9. 如权利要求 8 所述的系统，其特征在于，所述用于读的装置还用于从所述第一模块的所述输入/输出电路中读取所述标识符。

10. 如权利要求 6 所述的系统，其特征在于，所述第一模块还包括第二交易组。

11. 如权利要求 6 所述的系统，其特征在于，所述模块还包括用于对完整交易进行时间标记的装置。

12. 一种在模块和服务供应设备之间传递加密信息的方法，其特征在于，包括以下步骤：

- a) 在所述模块中创建第一随机数；
- b) 把所述随机数传送给所述服务供应设备；
- c) 用所述服务供应设备中的秘密密钥至少对所述随机数加密，从而产生一个证明；
- d) 至少将所述证明传送给所述模块；
- e) 用所述模块中的公开密钥对所述证明解密；
- f) 将所述第一随机数与在步骤 e) 的经解密的第一证明中找到的数进行比较，确定两个数是否匹配。

13. 如权利要求 12 所述的方法，其特征在于，步骤 b) 还包括将模块标识符传送给所述服务供应设备的步骤。

14. 如权利要求 12 所述的方法，其特征在于，所述服务供应设备是另一个模块。

15. 如权利要求 12 所述的方法，其特征在于，所述方法包括单线总线。

16. 如权利要求 15 所述的方法，其特征在于，所述单线总线基本上是一根线总线。

17. 一种在模块和服务供应设备之间传递加密信息的方法，其特征在于，包括以下步骤：

- a) 在所述服务供应设备中创建第一随机数；
- b) 把所述随机数传送给所述模块；
- c) 用所述模块中的秘密密钥至少对所述随机数加密，从而产生一个证明；

d)至少将所述第一证明传送给所述服务供应设备;  
e)用所述服务供应设备中的公开密钥对所述第一证明解密;  
f)将所述第一随机数与在步骤 e)的经解密第一证明中找到的数进行比较，确定两个数是否匹配。

18. 如权利要求 17 所述的方法，其特征在于，所述服务供应设备是另一个模块。

19. 如权利要求 17 所述的方法，其特征在于，所述方法包括单线总线。

20. 如权利要求 18 所述的方法，其特征在于，所述单线总线基本上是一根线总线。

21.一种用模块对加密数据解密的方法，其特征在于，包括以下步骤：

接收第一加密数据和第二加密数据；

用存储在所述模块中的秘密密钥对所述第一加密数据解密，从而产生第一解密密钥；

将所述第一解密密钥提供给一电子系统；

通过所述电子系统用所述第一解密密钥对所述第二加密数据解密，从而产生一个有用的信息。

22.如权利要求 21 所述的方法，其特征在于，所述加密数据是一个电子邮件信息。

# 说 明 书

保密交易的方法、设备、系统和固件

## 相关申请

本申请要求保护 1995 年 9 月 29 日提交的美国临时申请 60/004,510。

以下申请具有共同的受让人，它们具有相关的主题，并且因此通过引用包括在此：

序号 08/595,014，于 1996 年 1 月 31 日提交，发明名称为“用于传递价格单位的方法、设备和系统”；

序号 08/594,975，于 1996 年 1 月 31 日提交，发明名称为“在保密模块和另一模块之间传递有价值的信息”

## 发明背景

### 本发明的技术领域

本发明涉及一种保密交易用的方法、设备和固件。尤其，在基于电子模块的系统中，可以将模块构造成至少提供保密数据传输、数字签名，或者授权进行货币交易。

### 相关技术的描述

目前，具有相关磁条的信用卡是在市场上较佳的货币交易媒体。信用卡用户可以将卡拿到自动取款机、本地商店或银行，并进行货币交易。在许多情况下，可以通过电话接口用卡进行货币兑换。磁条卡用来帮助识别卡和卡的使用者。卡为传递提供了程度相当低的保密性。不管怎样，卡可使卡的持有者购买产品、支付债款，并在独立的银行帐户之间进行货币兑换。

已对磁条卡作了改进。已出现用微电路代替磁条的卡。一般与磁条一样，微电路被用来使读卡机进行交易。

## 发明内容

本发明是用于在一较佳的便携式模块和一服务供应者设备之间传递加密信

息的设备、系统和方法。本发明包括一模块，它具有专用识别，能够产生随机数(例如 SALT)，并将随机数与例如兑换货币的请求一起传送给服务供应者的设备。作为回复，服务供应者的设备可以(根据交易的类型)用秘密或公开密钥对随机数和其它信息进行加密，并将加密信息作为经签名的证明送回模块。收到经签名的证明后，模块将根据交易的类型用公共或秘密密钥对该证明解密，并将解密后的数值与原始的随机数进行比较。另外，如果数值相同，那么可以认为被请求的交易是保密的，从而进行交易。模块能够对交易信息作时间标记，并将交易信息存储在存储器中，以便事后查看。

## 附图概述

结合附图阅读以下详细描述，可以更全面地理解本发明的方法和设备，其中：

- 图 1 是一模块实施例的方框图；
- 图 2 例示了创建一交易组的过程；
- 图 3 例示了接收一电子邮件(E-mail)消息的技术；
- 图 4 例示了为公证函数准备模块的技术；
- 图 5 例示了把模块用作公证的技术；
- 图 6 例示了准备一模块以进行货币交易的技术；
- 图 7 例示了用一模块进行货币交易的技术；
- 图 8 例示了用一模块进行货币交易的技术；
- 图 9 例示了用一模块进行货币交易的技术；
- 图 10 例示了通过网络传递数据的技术；
- 图 11 例示了模块内软件和固件的结构；
- 图 12 例示了模块内软件和固件的结构。

## 目前较佳实施例的详细描述

图 1 例出了体现本发明一实施例的模块 10 的方框图。模块电路是单一的集成电路。可以理解，模块 10 也可以是组合在一起的多个集成或离散的元件电路。模块 10 包括微处理器 12、实时时钟 14、控制电路 16、数学协处理器 18、存储电路 20、输入/输出电路 26 以及能量电路。

可以将模块做得足够小，以便将其包含在各种物品中，包括(但不局限于)纪念品、卡、戒子、计算机、钱包、钥匙链、徽章、宝石、邮票或者使用者可以实际抓握或挂钩的任何物品。

微处理器 12 最好是 8 位微处理器，但也可以是 16、32、64 或任何可操作的位数。时钟 14 为模块电路提供计时。也可以用分立的时钟电路 14，提供连接运行的实时时钟。

数学协处理器电路 18 设计用来处理非常大的数。具体地说，协处理器将处理 RSA 加密和解密的复杂数学。

存储电路 20 可以包含只读存储器和非易失性随机存取存储器。另外，本领域的普通技术人员应该理解，可以用易失性存储器、EPROM、SRAM 和各种其它类型的存储电路建立一个等效的装置。

控制电路 16 为整个电路提供定时、锁定和各种必要的控制功能。

输入/输出电路 26 能与模块 10 进行双向通信。输入/输出电路 26 最好至少包括一个输出缓冲器 28 和一个输入缓冲器。为通过单线总线进行通信，可以输入/输出电路 26 中包括单线接口电路 32。

对于维持存储电路 20 工作和/或为模块 10 中的其它电路提供辅助电能，能量电路 34 是必须的。能量电路 34 可以包括电池、电容器、R/C 电路、光电池或任何其它等效的产生能量的电路或装置。

现在将讨论保密交易模块一实施例的固件结构和使用模块 10 的一系列应用举例。这些例子试图说明模块 10 的一组较佳特性，并解释该模块提供的服务。这些应用决不限制本发明的能力，而是举例揭示它的能力。

## I. 关于较佳模块及其固件设计的概述

模块 10 最好包括通用 8051 兼容微控制器 12 或合理的相似产品；连续运行的实时时钟 14；对大整数进行操作的高速模乘幂加速器(high-speed modular exponentiation accelerator)(数学协处理器)18；具有单线接口 32 的用于发送和接收数据的输入和输出缓冲器 28、30；具有预编程固件的 32 千字节的 ROM 存储器 22；用于存储关键数据的 8 千字节的 NVRAM(非易失性 RAM)；以及控制电路 16，该电路能使微控制器 12 加上电源，从而解释位于输入电路 26 中的数据，并对该数据起作用。模块 10 从单线线路汲取其工作电源。最好将微控制器 12、

时钟 14、存储器 20、缓冲器 28 和 30、单线前端 32、模乘幂加速器 18 以及控制电路 16 集成在单个硅片上，并用封装技术将它们组装在一个不锈钢的微型罐中，其中所用的安装技术实质上不可能在探测 NVRAM 24 中的数据时不破坏该数据。最初，大多数的 NVRAM 24 都可用来支持诸如下述的应用。普通的技术人员将理解，存在许多可对比的模块设计变化。例如，可以使用易失性存储器，或者使用接口而不用单线。可以将硅片安装在信用卡、戒子等物品中。

模块 10 最好首先由服务供应者(Service Provider)使用，服务供应者将数据装入模块 10，使其实现有用的功能，其次由端点用户(End User)使用，端点用户向模块 10 发出命令，从而为了端点用户的利益以服务供应者的名义进行操作。为此，模块 10 提供功能，以支持服务供应者为一计划中的应用建立模块。它还提供功能，以允许端点用户调用服务供应者提供的服务。

每个服务供应者可以通过创建一个交易组 40(参见图 11 和 12)而保留一块 NVRAM 存储器，支持它的服务。交易组 40 只是一组由服务供应者定义的对象(object)42。这些对象 42 包括数据对象(加密密钥、交易计数、钱款数额、日期/时间标志等)和交易脚本(transaction script)44，其中交易脚本规定了如何以有用的方式组合数据对象。每个服务供应者创建其自己的独立于每个其它交易组的交易组 40。因此，多个服务供应者可以在同一个模块 10 中提供不同的服务。可以支持的服务供应者的数量依赖于每个交易组 40 所定义的对象 42 的数量和复杂性。以下例举了一些可在交易组 40 内定义的对象 42：

RSA 模数	时钟偏移
RSA 指数	随机 SALT
交易脚本	配置数据
交易计数器	输入数据
货币寄存器	输出数据
破坏器	

在每个交易组 40 内，模块 10 将最初接受某些具有不可撤消效果的命令。一旦在某个交易组 40 中执行了这些不可撤消命令中的任何一些，它们将保持有效，直至模块的可用寿命终止或者直至从模块 10 中删除应用的交易组 40。另外，存在一些命令，它们具有不可撤消的效果直至模块寿命终止或者直至发出主擦除，

以擦除模块 10 的整个内容。以下将对这些命令作进一步的讨论。这些命令是服务供应者对端点用户执行的操作进行必要控制所必须的。一些不可撤消命令的例子有：

Privatize Object	Lock Object
Lock Transaction Group	Lock Micro-In-A-Can™

由于模块的许多作用集中在其保密的能力上，所以 Privatize 命令是一个非常重要的不可撤消的命令。

一旦将模块 10 作为一个整体锁定，便将剩余的 NVRAM 24 存储器分配给一个循环缓冲器，用于支持对在先交易的审计跟踪。每个交易由交易组的标号、规定组内交易脚本 40 的标号以及日期/时间标志来识别。

由固件实现的基本概念是服务供应者可以将交易脚本 44 存储在一个交易组 40 中，以便只执行诸对象中他希望端点用户能够执行的那些操作。服务供应者还可以存储和秘密化 RSA 密钥或者允许模块 10 以服务供应者的名义对交易“签名”的密钥(加密密钥)，并将这些密钥秘密化，从而保证它们的可靠性。通过使交易组 40 中的一个或多个对象 42 秘密化并/或将其锁定，服务供应者保持对允许模块 10 以其名义做什么进行控制。端点用户不能添加新的交易脚本 44，并因此被限制于对可以用服务供应者编程的交易脚本 44 执行的对象 42 进行操作。

## II. 模块的使用模型

本节说明模块 10 的一系列实际应用，涉及最简单的至最复杂的。这些应用中的每一个都将作详细描述，以便清楚为什么模块 10 是该应用的主要实现技术。

### A. 保密电子邮件的背景

在本节中，我们提供了一例如何使用模块 10 以允许任何人在任何地方保密地接收他或她自己的电子邮件。

#### 1. 标准的电子邮件

在标准的电子邮件系统中，用户的计算机与 Internet 服务供应者相连，并且当对供应者计算机查询新邮件时，用户的计算机要提供一个邮件口令。邮件以明

文形式驻留在供应者的计算机中，可被在那里工作的任何人阅读。另外，当邮件从它的源地传送时，它会经过许多计算机，并且也暴露在这些位置上。如果用户通过局域网络从他的供应者那里接收到他的邮件，那么同一网络上的其它任何人都可以获得并阅读该邮件。终于，利用许多不要求用户输入口令的邮件系统，坐在用户计算机旁的任何人都可以检索和阅读他的邮件，因为当查询供应者的计算机时，他的计算机自动提供口令。

还经常可以从用户计算机的配置文件中复制口令，并用它从不同计算机中阅读到他的邮件。作为这种广泛分布的明文形式的邮件以及较弱的口令保护的结果，标准的电子邮件被认为是非常不保密的。

为了克服该问题，已设计出称为 P.G.P.(Pretty Good Privacy)的保密系统。为了使用 P.G.P.，用户产生一个包含公共部分和秘密部分的完整的 RSA 密钥集。通过将公开密钥放在他所有电子邮件消息的签名块中并按排在 P.G.P. 公开密钥的可以公共访问的目录中公布，用户使其公开密钥可被广泛使用。他将其秘密密钥存储在他自己的个人计算机上，也许以口令保护的形式。当有人希望向该用户发送秘密电子邮件时，他产生一个随机 IDEA 加密密钥，并用 IDEA 加密算法对整个消息加密。然后，他用预期的接收者所提供的公开密钥对 IDEA 密钥本身加密。他以电子邮件形式将用 IDEA 加密的消息和用用户公开密钥加密的 IDEA 密钥发送给该用户。由于用 IDEA 对消息加密以及用预期接收者的公开密钥对 IDEA 密钥加密，所以除非是预期的接收者，其它看见这次传输的任何人都不能读出它。接收者的计算机包含相应的秘密密钥，并因此可以对 IDEA 密钥解密，以及用经解密的 IDEA 密钥对消息解密。这针对那些试图远程阅读用户邮件的人采取了保密措施，但当其他人可以访问该用户的计算机时，因用户计算机本身包含秘密密钥而使效力便降低了。即使秘密密钥是用口令保护的，但经常容易猜出用户口令或者当用户输入口令时对他进行偷看，所以用户计算机几乎不能保密。另外，因为用户的秘密密钥存储在其自己的计算机上且其它地方没有，所以接收者只能在该计算机上接收保密的电子邮。因此，P.G.P. 的弱点在于它被严格地局限于存有秘密密钥的用户计算机。

## 2. 保护模块的电子邮件

利用用来保护电子邮件的例示模块 10，用户可以将他的电子邮件转寄至他

前往的任何地方，无需害怕他人读取或者他的 PC 机是危及其邮件安全性的弱链路。保护模块的电子邮件系统类似于 P.G.P. 系统，只是将用来对 IDEA 密钥解密的秘密密钥存储在模块 10 的某交易组的一个秘密化对象中，而不是存储在 PC 机中。保护模块的电子邮件系统操作如下：

a. 参照图 2、11 和 12，在 S1，用户创建一个交易组 40，在 S2，产生一个 RSA 密钥集，并且将其装入交易组 40 的三个对象 42(一个 RSA 模数对象 N，和两个 RSA 指数对象 E 和 D)中。然后在 S3，他将解密指数 D 秘密化。最后在 S4，他创建一个交易脚本 44，以便取出位于输入数据对象中的数据，用模数 N 对其加密并将指数 D 秘密化，并且将结果放在输出数据对象中。在 S5，他锁定组，防止添加任何其它的交易脚本 44。他“忘记”了 D 的值，并在公共目录和他的电子邮件消息的签名块中公开 E 和 N 的值。由于他已经忘记了 D 并且已经使 D 的指数对象秘密化，所以任何人都没有办法找到 D 的值。

b. 参照图 3，为了把保密的电子邮件发送给用户，使用 P.G.P. 系统。当在 A1 处用户接收到保密电子邮件时，在 A2，他把经加密的 IDEA 密钥输送到交易组 40 的输入数据对象中，然后在 A3，他调用交易脚本 44，对该密钥解密，并在 A4 将经解密的结果放在输出数据对象中。然后他从输出数据对象中读取经解密的 IDEA 密钥，并在 A5，用它对他的邮件解密。注意，现在任何人，包括用户，都不可能在非实际拥有模块 10 的情况下阅读任何新的邮件。因此由于阅读邮件的计算机上必须实际存在模块 10，所以没有用户的知识就无法读取他的邮件。用户可以把他的模块 10 带到他去的任何地方，并能在任何地方用它阅读他的转寄邮件。他家中的计算机不是保密系统的弱点。

上述保密电子邮件是最简单可行的模块应用，它只要求一个 RSA 密钥和一个交易脚本 44。它甚至不必将公开密钥 E 存储在模块 10 中，而由于公开密钥被认为是公众可以得到的，所以这样做是个好办法。通过将 E 存储在一个指数对象中并且不把该对象或模数对象 N 秘密化，用户可以确保总能从模块 10 中读到公开密钥。由于不会要求模块 10 进行加密操作，所以交易脚本 44 都不包含 E。

## B. 数字公证业务

本节描述了使用模块 10 的较佳公证业务。

### 1. 标准公证服务的背景

传统的公证服务供应者接收并检查来自端点用户的文件，然后在文件上提供防伪标记，表示该文件是在某日送交公证人的，等等。这类公证业务的一种应用可以是记录对新发明的揭示，从而如果必要，以后可以在法院建立发明的优先权。在该情况下，公证员提供的最重要的服务是证明发明人在某日拥有该发明。(用来建立优先权的传统方法是使用一个 lab 记事本，发明人和公证员在记事本上签名，并记录揭示重要发明的日期。)

## 2. 使用模块的电子公证业务

有家公司(以下称为服务供应者)决定从商，为其客户(以下称为端点用户)提供公证业务(严格地说，是优先权证明业务)。服务供应者通过把模块 10 用作它的“代理”来选择做这项工作，并授权它们以他的名义对文件进行认证(记录日期并签名)。以下是该系统的较佳工作方式：

- a. 参照图 4、11 和 12，在 B1，服务供应者创建一个交易组 40，以便在模块 10 的一个“注册点(registered lot)”中实现电子公证功能。
- b. 在 B2，服务供应者使用保密计算设备，生成一个 RSA 密钥集并将其编程到每个模块 10 中，成为由三个对象 42 构成的组，该三个对象是一个模数对象和两个指数对象。尽可能使密钥集中的公共部分广泛地公知，并且服务供应者完全忘记秘密部分。使秘密指数对象秘密化，以防止从模块 10 中读回它。
- c. 服务供应者从每个模块 10 中阅读实时时钟 14，并创建一个时钟偏移对象，该对象包含实时时钟 14 之读数与一些方便的参照时间(例如，1970 年 1 月 1 日上午 12:00)之间的差。然后，在 B3，通过对实时时钟加上时钟偏移对象的值，从任何模块 10 中获得实际的时间。
- d. 在 B4，服务供应者创建一个交易序列计数器对象(transaction sequence counter object)，该对象初始为零。
- e. 服务供应者创建一个交易脚本 44，该交易脚本把输入数据对象的内容加至实际时间(实际时间是实时时钟 14 与时钟偏移对象之值的和)后，随后是交易计数器的值，再后是特有的激光登记号(unique lasered registration number)。然后，交易脚本 44 规定用秘密密钥对所有这一日期加密，并将所有日期放在输出数据对象中。在 B5，将进行该操作的指令作为交易脚本对象存储在交易组 40 中。
- f. 在 B6，将其它任何对象 42 秘密化，它不希望直接可读或可写。
- g. 在 B7，服务供应者锁定交易组 40，防止增加任何附加的交易脚本 44。

h. 参照图 5，现在服务供应者将模块分配给付款客户(端点用户)，供公证业务使用。当端点用户希望公证某一文件时，端点用户执行保密散列算法(规定于保密散列标准，联邦信息处理标准 Pub. 180)，以便将整个文件缩小至 20 字节的消息摘要。然后，在 C1，端点用户将 20 字节的消息摘要发送给输入数据对象，并且在 C2，访问交易脚本 44，将消息摘要与实际时间、交易计数以及特有的激光序列号汇集在一起，并用秘密密钥对所得的数据包签名。

i. 在 C3，端点用户通过用公开密钥对证明解密并检查消息摘要和实际时间标志等，检查该证明，以确保它们是正确的。然后，在 C4，端点用户以数字形式存储该数字证明以及文件的原始复印件。服务供应者将证明由其模块提供的证明的真实性。

j. 在服务供应者规定的一段时间之后，用户返回他的模块 10，付款，并获得一个包含新的秘密密钥的新模块。旧模块可以通过擦除整个交易组并且对它们重新编程而被循环使用。服务供应者保留一个它所用过的所有公开密钥的存档，以便在需要时证明旧证明的真实性。

### C. 数字投币机(cash dispenser)

该例对模块的使用是将模块 1 用作一个存款箱，由此对货物或服务进行付款。(为了便于讨论，推迟讨论用现金重新装满模块 10 的主题)。在该情况下，服务供应者是银行或其它金融机构，端点用户是银行的储户，希望用模块 10 进行购物，并且商家是被购货物或服务的供应商。以下将详细说明服务供应者、商家以及端点用户在这些交易中的作用。

如在模块 10 中实现的数字钱包的基本概念是，模块 10 最初包括一个含给定款值的锁定的货币对象，并且模块 10 可以经请求生成证明，这些证明基本上是签过名的文件，可以证明曾从货币对象的值中减去过所要求款额的事实。由于这些签名文件证明了曾将内部货币对象的值减去过对应于证明值的数值的事实，所以它们等效于现金。商家可以通过把它们还给服务供应者，而用这些证明赎回现金。

当处理代表现金的数字证明时，“重放”或复制是一个基本的问题。由于很容易复制和再传输数字数据，所以它不同于普通硬币或纸币，由于在制造硬币和纸币过程中使用了特殊的技术，所以重造钱币是很困难的。为此，收款人必须采

取特殊的步骤，以确保他所收到的数字证明不是对一些早期发放的证明的重放。通过收款人生成一个随机的“SALT”，这是一个询问数(challenge number)，并将它提供给付款人，便可解决这个问题。

SALT 是一种防止重放的方法。随机数在一种询问/回答的模式下发送和使用。要求另一方返回该随机数，作为他们回答的一部分。

付款人构造一份签名证明，该证明包括钱款数额和收款人的 SALT。当收款人收到该证明时，他用公开密钥对其进行解密，检查钱款数额，然后确认 SALT 与他提供的相同。通过使证明成为收款人秘密的，付款人向收款人保证该证明不是复印件或重放件，因些它是可靠的。无论模块 10 是否为付款人或收款人，都可以使用该方法。

另一个必须解决的问题是不能否认。这意味着交易的双方都应该不能争辩他实际上没有参与交易。交易记录(钱款证明)应该包含项目，以证明交易的每一方都是自愿参加者。

### 1. 传统现金交易的背景

在传统的现多交易中，端点用户首先接收来自银行的联邦储备券(Federal Reserve Note)，并且银行从他帐户的余额中减去等价的钱款数额。端点用户利用“公开密钥”可以证明联邦储备券的可靠性，包括：

- a. 可被一磁体吸引的磁性油墨；
- b. 埋在纸中的红蓝细线；
- c. 雕像周围的微细印刷；
- d. 与 USA 和券的面额一起印刷的嵌条。

该系统的“秘密密钥”是如何获得用于印刷钱币的原材料以及如何实际印刷钱币的细节。该信息由政府保存，不会泄露。端点用户把这些券带给商家，与其交换货物或服务。商家也用储备券的“公开密钥”证明他们是合法的。

最后，商家把储备券带到银行，出纳员再次检查“公开密钥”。如果储备券是合法的，那么商家银行帐户的余额就会增加这些储备券的面值。

该交易的最终结果是，端点用户的银行余额减少，商家的银行余额增加相同的数额，货物或服务从商家转移到端点用户，并且联邦储备券可以再用于其它的交易。

## 2.用模块进行货币交易的举例

由于数字数据与联邦储备券不同，容易对它们拷贝和复制，所以用模块 10 和数字证明进行货币交易有些复杂。然而，使用“SALT”和交易序号可以保证数字证明的可靠性。（在以下讨论中，假设交易的每一方都有自己的带秘密密钥的 RSA 密钥组，能够保密。）

a.参照图 6，服务供应者(银行)通过创建一个交易组 40 制备了模块 10，其中交易组 40 包含表示存储在模块 10 中货币值的货币对象。在 D1，服务供应者还创建一个交易计数器对象、一个模数对象和一个指数对象，并将供应者的秘密密钥存储在指数对象中。在 D2，他将密钥秘密化，使它不能被读取。接着，在 D3 和 D4，他将交易脚本 44 存储在交易组 40 中以便进行货币交易，并且锁定交易组，以至于不能再建立其它对象。（以下将进一步详细描述该交易脚本干什么。）最后在 D5，他广泛公开相应的公开密钥，使任何人都可以获得它。

b.端点用户接收来自服务供应者的模块 10，并且把存储在模块 10 中的数额记入端点用户银行帐户的借方。利用 PC 机或手提式计算机，端点用户可以质询模块 10，以便证明余额是正确的。

c.参照图 7，当在 E1，端点用户希望向商家购买一些物品或服务时，在 E2 和 E3，商家读取模块特有的激光登记号，并将其与随机的 SALT 一起放在一个数据包中。然后在 E4，商家用商人自己的秘密密钥对该数据包签名，并且在 E5，将所得的经加密的数据包与购买金额一起发送给交易组 40 的输入数据对象。

d.然后，商家调用由服务供应者编入模块 10 的交易脚本。在 E6，该交易脚本 44 从货币对象中减去购买金额，在 E7，将交易计数器对象的值加到输入数据对象的内容上，并且在 E8，用秘密密钥对所得的数据包签名并将结果放入在输出数据对象中。

e.然后，在 E9，商家从输出数据对象中读取结果，并用服务供应者的公开密钥对其解密。然后，在 E10，他确认购买的数额是正确的，并且剩余数据与他在步骤 c 中签名的数据包相同。

f.在 E11，在确认了模块 10 提供的证明可靠且为原件(非复印件)之后，商家发送货物或服务。之后，商家把数字证明送到银行。

g.在 E12，银行用服务供应者的公开密钥对证明解密，取出购买金额和交易计数，并在 E14 用商家的公开密钥对剩余数据解密，以揭示模块的特有激光登记

号。然后，银行在一数据库中用特有的激光登记号对模块 10 进行查寻，以便确认以前没有提交过该交易的交易计数。当测试通过时，在 E15，银行把交易计数值加到数据库中，然后使商家的银行余额增加购买的金额。已由模块 10 和商家对证明各部分签名的事实确认了交易是商家和模块 10 两方自愿同意的。

注意，存在许多不同的方式对交易计数值、特有的激光登记号、收款人提供的随机 SALT 以及用模块秘密密钥、或商家秘密密钥或两者加密的购买金额进行数据组合。许多组合也能为特有性、可靠性和不可否认性提供令人满意的保证，并且对固件的设计使服务供应者适于书写交易脚本 44，以满足他的特殊需要。

#### D.数字现金补充

以上在第 II.C.节中对数字现金钱包的讨论没有解决现金补充的问题。如第 II.C.节所讨论的，服务供应者可以简单地通过添加另一个含服务供应者公开密钥的模数对象和指数对象、一个随机 SALT 对象以及一个交易脚本，为模块 10 增加现金补充的能力，从而将钱款添加到余额中。服务供应者可以亲自或者通过网络远距离地对模块增加钱款。增加钱款的过程如下：

1. 参照图 8，在 F1 和 F2，服务供应者读取模块的特有激光登记号(ID 号)，并访问交易脚本 44，以便返回随机 SALT 对象的值。在 F3，模块 10 由先前值和随机数发生器计算一个新的随机 SALT 值，并将其返回至服务供应者。

2. 在 F4，服务供应者将模块 10 返回的随机 SALT 与将要增加的钱款数额和模块 10 特有的激光登记号一起放在一个数据包中，并用服务供应者的秘密密钥对所得的数据包加密。然后，把该经加密的数据包写回到交易组 40 的输入数据对象中。

3. 服务供应者调用交易脚本 44，该交易脚本用服务供应者的公开密钥对输入数据对象的内容解密，然后对照原来提供的数据，检查特有的激光登记号和随机 SALT 的值。在 F5，如果 SALT 匹配，那么从数据包中取出钱款金额，并将其加到模块中货币对象的值上。

注意，包括特有的激光登记号并不是严格必须的，但包括它可以确保服务供应者正确了解哪个模块正在接收资金。

## E. 举例描述在模块之间直接传递资金

以上第 II.C.2.g. 节揭示了当商家将数字证明返回他的银行归于他的帐户时所产生的问题。商家银行必须将证明送回服务供应者以便履行，或者访问一数据库中服务供应者的记录从而可以确定交易计数对象的值是否是特有的。这不方便，且要求有基础设施。由于商家银行必须把用过的证明号记录到一数据库中，以防止重复使用，所以它还防止进行任何匿名的交易(因为如果使用现金，交易就会匿名)。通过在模块之间进行资金转移可以消除所有这些问题。另外，在模块之间实现资金转移所需的步骤比第 II.C.2. 节中描述的步骤简单得多。

在以下讨论中，假设商家也有一个模块，用于收集从端点用户(客户)那里接收到的资金。端点用户拥有的模块将被称为付款器，而商家拥有的模块将被称为收款器。以下是执行资金转移的步骤：

1. 参照图 9、11 和 12，商家用他的计算机访问收款器中的交易脚本 44，以便提供随机 SALT。他从交易组 40 的输出对象中读出该 SALT。

2. 在 G1，商家将 SALT 和端点用户的购买金额拷贝到付款器的输入数据对象中，然后在 G2，商家访问付款器中的交易脚本 44，以便从余额中减去购买金额，，将收款器的 SALT 与购买金额组合到一数据包中，用服务供应者的秘密密钥对所得的数据包进行加密，并将其返回到输出数据对象中。

3. 然后，在 G3，商家读取该数据包并将其拷贝到收款器的输入数据对象中，然后访问收款器中的交易脚本 44，以便用服务供应者的公开密钥对数据包解密，并且在 G4，参照收款器原始产生的 SALT，检查 SALT。如果它们一致，那么收款器将购买金额加到它的余额上。

这就完成了资金转移。注意该交易将购买金额从付款器有效地转移到收款器，并且交易步骤比 II.C.2. 中描述的三向交易要简单得多。商家可以通过一次类似的交易将余额转移到他的银行帐户上，在所述交易中，银行将一个 SALT 提供给商家的模块，而商家模块制备一份将传送给银行的有关余额的证明。商家用模块收集资金简化了交易，避免了用数据库确认特有性的需要，并且保持了现金交易经常引起的端点用户的匿名性。

## F. 通过网络用模块交易的举例

还可以通过网络进行上述第 II.C.2.、II.D. 和 II.E. 节描述的交易，商家、端点

用户和模块之间可以存在物理间距。但是，由于与模块 10 的通信中有一个不加密，并且该通信会因此而受到伪造，所以这会引发潜在的问题。为了避免该问题，两方都必须产生一个 SALT，以便另一方可以证明它能够用服务供应者的秘密密钥对该 SALT 加密的能力，并由此证明了可靠性。由于该协议的操作涉及在模块之间转移资金(以上的第 II.E.节)，所以以下将描述该协议的操作。该方法可用来使上述任何交易在网络上进行。显然，它能够通过 Internet 网进行保密的电子贸易。

1. 参照图 10、11 和 12，在 H1，付款器产生一个随机 SALT，并通过网络将其发送给收款器。

2. 在 H2，收款器将购买金额加到付款器的 SALT 后，随后接是收款器随机产生的 SALT。然后，收款器用服务供应者的秘密密钥对该数据包加密，并将它送回付款器。

3. 在 H3，付款器用服务供应者的公开密钥对数据包解密，取出付款器的 SALT，并将其与付款器在步骤 1 中提供的 SALT 进行比较。如果它们一致，那么在 H4，付款器从它的余额中减去购买金额，并在 H5，产生一个由购买金额和收款器 SALT 组成的证明，其中用服务供应者的秘密密钥对证明加密，并且将其返回收款器。

4. 在 H6，收款器用服务供应者的公开密钥对数据包解密，取出收款器 SALT，并将其与收款器步骤 2 中提供的 SALT 进行比较。在 H7，如果它们一致，那么收款器将购买金额加到它的余额上。

SALT 的交换可使每个模块确认它正在与另一个模块通信，并且确认所要求的资金转移因此是合法的。步骤 3 中描述的 SALT 比较可使付款器在提款之前确认收款器是一合法的模块，并且步骤 4 中描述的比较可使收款器在存款之前确认付款器是一合法的模块。上述交易在加密数据包中提供了最少的必要信息，确认资金正从一个模块 10 转移到另一个模块。可以以匿名为代价包括诸如特有激光登记号等其它信息，以便提供附加信息并对交易进行更多的控制。

## G. 软件授权和应用计数(usage metering)技术的举例

模块 10 非常适用于在综合软件系统使能够提供特殊软件特点的任务，并且适用于统计这些特点的应用。(该用法模型与上述用于从模块 10 中提款的模型相

同。)

### 1.准备

参照图 11 和 12，服务供应者创建一个交易组 40 并将一配置对象存储在该组中，细述了端点用户可以使用模块 10 中的哪个软件。服务供应者还创建一个包含准用信贷(它可以以时间为单位而不是实际的美元数)的货币对象，并且存储和秘密化一对 RSA 秘密密钥，以用于鉴定。存储一交易脚本 44，用于接收 SALT 以及从端点用户提取的金额，用提款金额减余额，并输出包含提款金额、销售量以及配置对象值的 RSA 经签名的证明。

### 2.应用

在使用模块 10 内软件的周期间隔中，PC 程序生成一个随机 SALT 以及使用模块 10 的收费金额，并将该信息发送给模块 10。模块 10 减少余额并返回证明。PC 机对证明解密并确认 SALT 是相同的，提款金额是正确的，以及存储在配置对象中的信息授权使用模块 10 内的软件。如果所有这些测试都成功，那么模块 10 在向模块 10 索要另一个证明前执行一段规定的时间，或者执行给定数目的操作。

该应用模型存在许多可能的变化。例如，交易脚本 44 还可以将实际时间汇集到证明中，以便 PC 机上运行的应用程序可以保证其执行时间精确测量了的。(这可以要求服务供应者在初始化期间创建一个时钟偏移对象，以便为测量时间提供基准。)

## H.对交易接触式存储器的模拟

该应用模型描述了如何用模块 10 模拟更简单的交易接触式存储器<sup>TM</sup>(Transation Touch Memory，以下称“TTM”)(DS 1962)，或者可以以接近等效或类似的方式进行工作的任何类似的装置或替代物的行为。TTM 的主要特点是存在一个与存储块相关的计数器，该计数器以这样的方式工作，即当改变存储块的内容时自动使计数器增值。

### 1.准备

通过创建一个配置对象，一个交易计数器对象和一个交易脚本对象，可以将该简单的特点编入模块 10 中，其中交易脚本对象将输入对象的内容与交易计数器对象的值合并，并将它们放在配置对象中，在该过程中自动对计数器增值。锁定所有这三个对象 42，但不将它们秘密化。

## 2. 应用

为了增加或提取钱款，端点用户直接读取配置对象和交易计数器对象的值，然后对配置对象解密并参照计数器对象的值检查来自解密数据包的交易计数。端点用户还参照模块 10 的登记号检查来自加密数据包的特有激光登记号。如果两者一致，那么认为余额有效。对余额加上或减去一金额，使交易计数增 1，并且再次对数据包加密且将其存储在输入数据对象中。然后，调用交易脚本 44，将数据和交易计数器的值移至配置对象，在该过程中自动增加计数器的值。(交易脚本 44 保证当改变配置对象中的数据时，将增加计数器对象的值。)

由于模块 10 本身不必进行任何加密，所以这一简单的操作可以进行的相当快。但是，与利用 TTM 一样，端点用户现在必须用一个保密的计算设备进行加密和解密操作。因此与那些依赖于模块加密能力的应用相比，该应用受到的保护较弱。

### I. 邮政计费业务的技术举例

该应用模型描述了用模块 10 分发邮资证明的应用。构成证明的数字信息以二维条形码的形式被印在信封上，服务供应者(U.S.P.S.)可以阅读和鉴别条形码。可以结合模块 10 使用在接至激光打印机的普通 PC 机上运行的计算机程序，打印邮资证明。

#### 1. 准备

服务供应者创建一个组包含货币寄存器、每个模块公用的 RSA 秘密密钥(指数对象和模数对象)和交易脚本 44。脚本 44 将 SALT 和(端点用户计算机提供的)要提的金额与特有的激光登记号合并在一起，用秘密密钥对该数据包加密，从余额中减去提款，并将加密证明放在 PC 机可以阅读的输出对象中。

服务供应者用规定钱款额对余额初始化，锁定余额和脚本 44，将 RSA 密钥对象秘密化，并且锁定该交易组，以便不能增加更多的脚本。然后，可以在柜台上销售用该方式准备的模块，以便与基于 PC 机的邮资计费程序一起使用。

#### 2. 应用

当要打印第一张信封时，PC 程序通过计算日期的单向散列(例如，保密散列标准，FIBS PUB 180)和该部分的特有的激光登记号，准备第一 SALT。将该信息连同要提取的邮资金额一起传送给模块 10。将所得的证明以及散列生成数(对于

第一散列为 1)、特有的激光登记号、标志的明文命名、日期和希望用来识别端点用户的其它信息印刷成二维条形码。通过对先前的 SALT 再进行单向散列并增加散列生成数，可以生成后续的 SALT。

当服务供应者收到信封时，大多数看面值，并不阅读数字条形码。但是，可以阅读条形码的统计样本，并且用公开密钥对提供的信息解密，以及核实信息。检查差异，并在现有法律下起诉欺骗行为。由于服务供应者可以由特有的激光登记号、日期以及散列生成号重新创建 SALT，并由此核实交易不仅是现时的而且与特定的模块 10 链接，所以可以进行核实。

注意，上述存在许多可能的变化，它们导致类似的结果。大多数相似的欺骗是复制品，在该行为中，用户捕获发送给打印机的数字信息，产生邮资证明，并对相同的证明制作许多复印件。服务供应者简单通过阅读散列生成数和特有的登记号并在一数据库对其进行查找以确信该用户不在复制同一证明，便可以容易地探明这一现象。（该检查比要求 RSA 解密的完整证明核查更频繁。）

#### J. 订阅信息业务

该应用模型描述了这样一种应用，即服务供应者通过互连网以加密形式为同意对信息付款的用户制作可用信息。该应用的工作方式与上述第 A 节中描述的保密电子邮件应用模型正好相同，只是服务供应者就其以电子邮件方式发给用户的加密信息给用户开帐单。帐单信息是从 RSA 的公开密钥登记中获得的，该登记允许服务供应者根据他的公开密钥或他的模块 10 的特有激光序号进行识别并给用户开帐单。

#### K. 担保秘密密钥保密的登记

为了向商家独立确认端点用户的身份，服务供应者希望保留一个登记，该登记包含特定模块 10 的公开密钥以及姓名、地址和模块 10 发向的个人的其它识别信息。为此，服务供应者必须确保登记中的公开密钥与只有模块 10 知道的秘密密钥对应。为了保证这一要求，当从模块 10 中取出公开密钥并将其放在登记中时，模块 10 必须由服务供应者拥有。在登记中记录下该信息后，服务供应者可以将模块 10 传送给登记中命名的端点用户。

对于端点用户来说，当他接收到模块 10 时，能确认服务供应者或服务供应

者雇员不知道秘密密钥也是很重要的。由于理想的登记系统应该不需要任何一方信任另一方，所以这是重要的。只有当能够使第一方确信另一方不可能知道秘密密钥，系统工作才能使每个人满意。

实现这一要求的方法是，服务供应者向模块 10 发一命令，使它产生用随机数生成一个完整的 RSA 密钥集，然后自动使指数中的一个秘密化，致使没有人能够发现秘密密钥的值。该密钥集具有特定的类型，不同于服务供应者编入罐中的密钥集，因此用模块直接做生意的任何人都能自己确定只有模块 10 知道秘密密钥。

### 1.准备

为该应用创建一个保护口令的交易组 40，然后在该组中创建一个由模块生成的 RSA 密钥集。(生成密钥集之后，自动锁定模数和一个指数，而模块 10 的固件自动将第二指数秘密化)。然后，服务供应者创建一个交易脚本 44，它将用秘密密钥对来自输入对象的数据加密，并将加密结果放在输出对象中。为了满足应用的任何附加目的，交易脚本 44 可以有选择地将附加信息(例如，交易计数器)加到来自输入对象的数据上。还可以以服务供应者的判断力增加其它对象 42 和交易脚本 44。当完成时服务供应者锁定交易组 40。

接着，服务供应者从交易组 40 中阅读 RSA 模数和公共指数，并将它们与识别端点用户的信息一起记录在登记中。最后，服务供应者将模块 10 运给端点用户，而后再将可用来访问交易组 40 的口令传送给端点用户。

### 2.应用

当商家希望通过 Internet 或其它网络获得对端点用户的肯定的识别时，商家会生成一个专用的数据包，并将它发送给端点用户，然后端点用户把数据传送到输入对象中，并调用交易脚本 44，用模块 10 生成的密钥对数据加密。将所得的加密数据包送回商家。然后，商家访问服务供应者提供的数据库，以获得属于端点用户的公开密钥，并试图用端点用户的公开密钥对加密数据包解密。如果解密成功，那么商家已经证明在网络远程连接的地方实际存在端点用户的模块 10。通过保证在远地存在端点用户的模块 10，该识别结果使数据包的内容以及因此由该数据包内容代表的任何金融交易(可以是端点用户要求的)有效并合法化。

在这里描述的模型中，进行金融交易的权力来自服务供应者保存的登记。因此，该信息必须是准确的，并且模块 10 的秘密密钥必须对各方保密。由于每个

模块 10 都有它自己特有的 RSA 密钥集, 所以该模型没有对模块 10 进行表示货币独立于服务供应者保存的登记的规定. 而是使登记与模块能够用其秘密密钥签名的能力一同起确定作用, 以识别对于任何其它方来说为远距离的端点用户.

#### L. 对交易量征税

此应用适用于一种交易模型, 在该模型中, 服务供应者试图向端点用户收取服务费用, 该费用是模块 10 传递货币总量的一个百分数. 模型类似于上述第 C、D、E 和 F 节描述的模型, 但附加了一个破坏者对象, 该对象能够使任何特定的交易脚本 44 在预定的日期和时间截止. 该模型还要求使用一附加的货币对象, 用一合适的交易脚本 44 对其编程, 以累加流出模块 10 的所有货币的总值.

1. 服务供应者创建一个交易组 40, 它包含如上第 D 和 E 节描述的货币对象等. 服务供应者还创建一个附加的货币对象, 起量累加器的作用. 服务供应者还创建交易脚本 44, 如在 D 和 E 中一样, 用于提款或存款, 只是用于向模块 10 加钱的交易脚本包括一个破坏者对象集, 以便在将来的某一预定时刻截止, 并且用于提款的交易脚本 44 包括一个一条将提款金额加到货币对象上的命令, 货币对象起量累加器的作用. 然后, 服务供应者锁定交易组并将模块运送给端点用户.

#### 2. 应用

如以上第 D 和 E 节所述, 端点用户用模块 10 存款和提款. 在使用模块 10 期间, 在起量累加器作用的货币对象中, 累加由模块 10 化费的所有货币的累积总额. 当时限截止时, 端点用户不再能够向他的模块 10 加钱了, 尽管需要时在还有存款之前他可以继续提款. 然后, 端点用户将模块 10 还给服务供应者进行恢复. 服务供应者读取剩余的钱款以及量累加器中记录的钱款. 服务供应者向端点用户开服务费用的帐单, 该服务费用是量累加器中数额的一个百分数. 如果端点用户愿意支付这笔款项, 继续他的服务, 那么破坏并重建交易组 40, 然后将端点用户返还时留在模块 10 中的钱款金额编回交易组 40 的货币对象中. 然后, 只要端点用户支付服务费用, 服务供应者就把经恢复的模块还给端点用户.

上述系统可使服务供应者定期收取服务费用, 不必监视和关心端点用户进行的每项金融交易. 如量寄存器的内容所确定的, 费用基于实际应用.

## 与模块一起使用的固件定义举例

<b>对象</b>	模块固件接收并运行的最基本的数据结构。下节提供了有效对象以其定义的表。
<b>组</b>	对象的一个自含集。一个对象的作用域限于它作为成员的组。
<b>组 ID</b>	一个数，最好在 0 和 255 之间，表示一特定的组。
<b>对象 ID</b>	一个数，最发在 0 和 255 之间，表示一特定组内的一个特殊对象。
<b>对象类型</b>	最好是一个字节的分类符，它描述了一个特定的对象。
<b>PIN</b>	一个由字母和数字表示的个人身份识别号，其长度最好为 8 个字节。
<b>公共 PIN</b>	对诸如审计跟踪访问等共享资源进行控制的 PIN。还用来控制主机创建和删除组的能力。
<b>组 PIN</b>	对访问某一组内对象之特殊操作进行控制的 PIN。
<b>审计跟踪</b>	对发生在锁定模块之后的交易的记录。
<b>锁定对象</b>	通过执行锁定对象命令已锁定的对象。一旦对象锁定，就不能直接阅读了。
<b>秘密对象</b>	通过执行秘密化对象命令而秘密化的对象。一旦对象秘密化，就不能直接读或写了。
<b>锁定组</b>	用锁定组的命令已锁定的组。锁定组后，不能创建对象。

**复合对象** 若干对象的组合。个体对象继承了复合对象的属性。

### 对象定义举例

**RSA 模数** 一大的整数，最好其长度最多为 1024 位。它是两个大的素数的乘积，每个素数的位数长度约为所需模数的一半。RSA 模数在以下等式中使用，用于对消息 M 加密和解密：

(1) 加密：  $C = M^e \pmod{N}$

(2) 解密：  $M = C^d \pmod{N}$

其中，C 是密文，d 和 e 是 RSA 指数(参见下文)，而 N 是 RSA 模数。

**RSA 指数** (上述等式 1 和 2 中所示的)的 e 和 d 都是 RSA 的指数。它们一般是较大的数，但小于模数(N)。RSA 指数可以是秘密的，或公开的。当在模块中创建 RSA 指数时，可以声明它们是其中的一种。一经创建，指数可以由公开指数转换成秘密指数。但指数秘密化之后，将保持秘密化，它所属的交易组 40 受到破坏。

**交易脚本** 交易脚本是模块执行的一系列指令。当调用交易脚本时，模块固件解释脚本中的指令，并将结果放在输出数据对象中(参见下文)。实际脚本只是一个对象表。对象排列的次序规定了对对象进行的操作。交易脚本 44 最好只要 128 字节。

**交易计数器** 交易计数器对象的长度最好为 4 个字节，并且当创建时通常初始值为零。每次调用引用该对象的交易脚本时，交易计数器增 1。一旦锁定交易计数器，就只能进行阅读，并且成为一个不能撤消的计数器。

**货币寄存器** 货币寄存器对象的长度最好为 4 个字节，并且可用来表示货币

或一些其它形式的信贷。一旦已经创建该对象，就必须将其锁定，以防用户窜改它的值。一旦锁定，只有通过调用交易脚本才能改变该对象的值。进行货币交易的一般交易组 40 可以具有一个从货币寄存器中提款的脚本和一个向货币寄存器存款的脚本。

**时钟偏移** 该对象最好为 4 个字节数，它包含模块实时时钟读数与一些方便时刻(例如 1970 年 1 月 1 日上午 12:00)的差。然后，通过把时钟偏移的值与实时时钟相加，从模块中获得真实的时间。

**SALT** SALT 对象的长度最好为 20 个字节，并且当创建时应该用随机数据对其初始化。当主机发出生成随机 SALT 的命令时，模块将先前的 SALT 与模块的随机数(最好由随机发生加电时产生)合并，以产生一新的随机数。如果没有将 SALT 对象秘密化，那么以后可以通过发出一个阅读对象命令进行阅读。

**配置数据** 这是一个用户定义的结构，其最大长度最好为 128 个字节。该对象一般用来存储其交易组 40 特定的配置信息。例如，可用配置数据对象规定货币寄存器对象的格式(即它表示的货币类型)。由于该对象没有预定的结构，所以交易对象决不能使用它。

**输入数据** 输入数据对象只是一个输入缓冲器，其最大长度最好为 128 个字节。一个交易组可以具有多个输入对象。主机用输入数据对象存储交易脚本 44 将处理的数据。

**输出数据** 交易脚本将输出数据对象用作输出缓冲器。当创建交易组时，自动创建该对象。其长度最好为 512 字节，并且继承了其交易组的口令保护。

- 随机填充** 当脚本解释器遇到该类型的对象时，它自动填充当前信息，致使它的长度比先前模数的长度短 1 位。当创建交易组时，自动创建该对象的句柄。它是一个秘密对象并且不可以用阅读对象命令阅读。
- 工作寄存器** 脚本解释器将该对象用作工作空间，并且该对象可以在交易脚本中使用。当创建交易组时，自动创建该对象的句柄。它是一个秘密对象并且不可以用阅读对象命令阅读。
- ROM 数据** 当创建交易组时，自动创建该对象。它是一个锁定对象，并且可以用写对象命令改变它。该对象的长度为 8 字节并且其内容等同于  $8 \times$  Micro-In-A-Can<sup>TM</sup> 的 ROM 数据

### 较佳的模块固件命令集

#### 设置公共的 PIN(01H)

发送(至模块)

01H, 旧 PIN, 新 PIN, PIN 选择字节

接收数据

如果成功, CSB(命令状态字节) = 0, 否则为合适的差错代码

输出长度 = 0

输出数据 = 0

注意:

PIN 选项字节可以是对以下任何值的逐位“或”操作:

PIN\_TO\_ERASE 00000001b(要求 PIN 作主擦除)

PIN\_TO\_CREATE 00000010b(要示 PIN 作组创建)

最初模块的 PIN(个人身份识别号)为 0(空), 选项字节为 0。一旦建立了 PIN, 只有通过提供旧的 PIN 或通过主擦除才有改变它。但是, 如果在选项字节中设置了 PIN\_TO\_ERASE, 那么只有通过设置公共 PIN 命令才能改变 PIN。

设置公共 PIN 命令的差错代码可能有:

ERR\_BAD\_COMMON\_PIN (公共 PIN 匹配失败)

ERR\_BAD\_PIN\_LENGTH (新的 PIN 长度>8 字节)

ERR\_BAD\_OPTION\_BYTE (不认识的选项字节)

对于本节描述的所有命令，主机接收到的数据将是返回数据包的形式。返回数据包具有以下结构:

命令状态字节 (如果命令成功, 为 0, 否则为差错代码, 1 字节)

输出数据长度 (命令输出长度, 2 字节)

输出数据 (命令输出, 以上规定的长度)

### 主擦除(02H)

发送数据

02H, 公共 PIN

接收数据

如果命令成功, 则 CSB = 0, 否则为 ERR\_BAD\_COMMON\_PIN

输出长度 = 0

输出数据 = 0

注意:

如果 PIN 选项的 LSB(最低有效位)清零(即, 不要求 PIN 作主擦除), 那么, 为公共 PIN 的值发送 0。一般, 该文本总假定需要 PIN。如果没有建立 PIN, 应该把零发成 PIN。这对公共 PIN 和组 PIN 都是适用的(见下文)。如果 PIN 正确, 那么固件删除所有的组(见下文)以及组内的所有对象。将公共 PIN 和公共 PIN 选项字节重新设置为零。

擦除每样东西后, 模块发送返回数据包。CSB 如以上所述。将输出数据长度和输出数据字段设置为零。

### 创建组(03H)

发送数据

03H, 公共 PIN, 组名, 组 PIN

## 接收数据

如果命令成功，则 CSB = 0，否则为适当的差错代码

如果成功，则输出长度=1，否则为 0

如果成功，则输出数据 = 组 ID，否则为 0

注意：

组名的最大长度为 16 字节，并且最大的 PIN 长度为 8 字节。如果在公共 PIN 选项字节中设置 PIN\_TO\_CREATE 的位，并且发送 PIN 与公共 PIN 不匹配，那么模块将把 OSC 设置成 ERR\_BAD\_COMMON-PIN。

创建组命令的差错返回代码可能有：

ERR\_BAD\_COMMON\_PIN (不正确的公共 PIN)

ERR\_BAD\_NAME\_LENGTH (如果组名长度>16 字节)

ERR\_BAD\_PIN\_LENGTH (如果组 PIN 的长度>8 字节)

ERR\_MIAC\_LOCKED (已锁定模块)

ERR\_INSUFFICIENT\_RAM (没有足够的存储区供新组使用)

## 设置组 PIN(04H)

### 发送数据

04H，组 ID，旧的 GPIN，新的 GPIN

### 接收数据

如果命令成功，CSB = 0，否则为适当的差错代码

输出长度 = 0

输出数据 = 0

注意：

组 PIN 只限制访问由命令数据包中发送的组 ID 确定的组内的对象。

设置组 PIN 命令的差错代码可能有：

ERR\_BAD\_GROUP\_PIN (组 PIN 匹配失败)

ERR\_BAD\_PIN\_LENGTH (新的组 PIN 的长度>8 字节)

## 创建对象(05H)

发送数据

05H, 组 ID, 组 PIN, 对象类型, 对象属性, 对象数据

接收数据

如果命令成功, CSB = 0, 否则为适当的差错代码

如果成功, 输出长度 = 1, 否则为 0

如果成功, 输出数据 = 对象 ID, 否则为 0

注意:

如果创建对象命令成功, 那么模块固件返回组 ID 确定的组内的对象 ID。如果主机提供的 PIN 不正确, 或者锁定组命令(以下描述)已锁定了组, 那么返回 CSB 中的差错代码。如果对象因任何原因而失效, 那么对象创建也失败。例如, 如果正在创建的对象是一个 RSA 模数(0 类型), 并且它的长度大于 1024 位。如果它服从所有的交易脚本规则, 那么交易脚本创建将成功。

创建对象命令的差错返回代码可能有:

ERR_BAD_GROUP_PIN	(不正确的组 PIN)
ERR_GROUP_LOCKED	(组已锁定)
ERR_MIAC_LOCKED	(模块已锁定)
ERR_INVALID_TYPE	(规定的对象类型失效)
ERR_BAD_SIZE	(对象长度失效)
ERR_INSUFFICIENT_RAM	(没有足够的存储空间供新对象使用)

对象类型:	RSA 模数	0
	RSA 指数	1
	货币寄存器	2
	交易计数器	3
	交易脚本	4
	时钟偏移	5
	随机 SALT	6

# LOCKED/UNLOCKED

配置对象	7
输入数据对象	8
输出数据对象	9

对象属性： 锁定的 00000001b  
秘密化的 00000010b

也可以在创建之后，通过使用以下描述的锁定对象和秘密化对象的命令，锁定对象并使对象秘密化。

## 锁定对象(06H)

发送数据

06H, 组 ID, 组 PIN, 对象 ID

接收数据

如果命令成功, CSB = 0, 否则为适当的差错代码

输出长度 = 0

输出数据 = 0

注意：

如果组 ID, 组 PIN 和对象 ID 都正确, 那么模块将锁定规定的对象。锁定对象是一个不可撤消的操作。

锁定对象命令的差错返回代码可能有：

ERR\_BAD\_GROUP\_PIN (不正确的组 PIN)

ERR\_GROUP\_LOCKED (组已锁定)

ERR\_MIAC\_LOCKED (模块已锁定)

ERR\_BAD\_GROUP\_ID (规定的组不存在)

ERR\_BAD\_OBJECT\_ID (规定的对象不存在)

## 秘密化对象(07H)

发送数据

07H, 组 ID, 组 PIN, 对象 ID

接收数据

如果成功，CSB = 0，否则为适当的差错代码

注意：

如果组 ID，组 PIN 和对象 ID 有效，那么将该对象秘密化，秘密化的对象共享锁定对象的所有性质，但它们是不可读的。只有通过交易脚本才能修改秘密化的对象。注意锁定一个秘密化的对象是合法的，但由于对象秘密化是一个比对象锁定更强的操作，所以这是没有意义的。将对象秘密化是一个不可撤消的操作。

秘密化对象命令的差错返回代码可能有：

ERR\_BAD\_GROUP\_PIN (不正确的组 PIN)

ERR\_GROUP\_LOCKED (组已锁定)

ERR\_MIAC\_LOCKED (模块已锁定)

ERR\_BAD\_GROUP\_ID (规定的组不存在)

ERR\_BAD\_OBJECT\_ID (规定的对象不存在)

### 使对象可破坏(08H)

发送数据

08H, 组 ID, 组 PIN, 对象 ID

接收数据

如果成功，CSB = 0，否则为适当的差错代码

注意：

如果组 ID，组 PIN 和对象 ID 有效，那么使对象可破坏。如果某一对象可破坏，那么在组破坏器起作用后，交易脚本不能使用该对象。如果交易组内不存在破坏器的对象，那么可破坏对象的属性位不起作用。使对象可破坏是一个不可撤消的操作。

使对象可破坏命令的差错返回代码可能有：

ERR\_BAD\_GROUP\_PIN (不正确的组 PIN)

ERR\_GROUP\_LOCKED (组已锁定)

ERR\_MIAC\_LOCKED (模块已锁定)

ERR\_BAD\_GROUP\_ID (规定的组不存在)

ERR\_BAD\_OBJECT\_ID (规定的对象不存在)

## 锁定模块(09H)

发送数据

09H , 公共 PIN

接收数据

如果成功, CSB = 0 , 否则为适当的差错代码

如果成功, 输出长度 = 2 , 否则为 0

如果成功, 输出数据 = 审计跟踪, 否则为 0

注意:

如果主机提供的公共 PIN 正确, 并且先前没有锁定模块, 那么命令将成功。当锁定模块时, 它将不接收任何新的组或对象。这意味着自动锁定所有的组。系统或诸组不用的 RAM 将被用来进行审计跟踪。模块未被成功锁定, 就不进行审计跟踪。

审计跟踪记录的长度为 6 字节, 并且具有以下结构:

组 ID | 对象 ID | 日期/时间标志

一旦建立了审计跟踪, 每次执行交易脚本, 就把上述形式的记录存储在第一个字节大小可用的位置上。注意, 由于模块必须在审计跟踪开始之前锁定, 所以组 ID 和对象 ID 都不能变化。这总是允许这样的应用, 它处理审计跟踪, 以便唯一识别被执行的交易脚本。一旦审计跟踪用完了所有可用的存储空间, 它将把新的交易记录存储在最旧的交易记录上。

锁定模块命令的差错代码可能有:

ERR\_BAD\_COMMON\_PIN (提供的公共 PIN 不正确)

ERR\_MIAC\_LOCKED (模块已锁定)

## 锁定组(0AH)

发送数据

0AH , 组 ID , 组 PIN

接收数据

如果命令成功, CSB = 0 , 否则为适当的差错代码

输出长度 = 0

输出数据 = 0

**注意:**

如果提供的组 PIN 是正确的，那么模块 BIOS 将不允许在规定的组内创建更多的对象。由于组完全是自含式实体，所以通过执行删除组命令(以下描述)可以删除它们。

锁定组命令的差错返回代码可能有：

- ERR\_BAD\_GROUP\_PIN (不正确的组 PIN)
- ERR\_GROUP\_LOCKED (组已锁定)
- ERR\_MIAC\_LOCKED (模块已锁定)
- ERR\_BAD\_GROUP\_ID (规定的组不存在)

### 调用交易脚本(0BH)

发送数据

0BH, 组 ID, 组 PIN, 对象 ID

接收数据

如果命令成功, CSB = 0, 否则为适当的差错代码

如果成功, 输出长度 = 1, 否则为 0

输出数据 = 估计的完成时间

**注意:**

模块返回的时间估计是六分之一秒。如果返回 CSB 中的差错代码，那么时间估计将为 0。

执行交易脚本命令的差错返回代码可能有：

- ERR\_BAD\_GROUP\_PIN (不正确的组 PIN)
- ERR\_BAD\_GROUP\_ID (规定的组不存在)
- ERR\_BAD\_OBJECT\_ID (组中不存在脚本对象)

### 阅读对象(0CH)

发送数据

0CH, 组 ID, 组 PIN, 对象 ID

接收数据

如果命令成功, CSB = 0, 否则为适当的差错代码

如果成功，输出长度 = 对象长度，否则为 0

如果成功，输出数据 = 对象数据，否则为 0

注意：

如果组 ID，组 PIN 和对象 ID 都正确，那么模块检查指定对象的属性字节。如果对象没有秘密化，那么模块把对象数据发送组主机。如果组 PIN 无效，或者对象已秘密化，那么模块返回输出长度中的 0，以及返回数据包的数据字段。

阅读对象命令的差错代码可能有：

ERR\_BAD\_GROUP\_PIN (不正确的组 PIN)

ERR\_BAD\_GROUP\_ID (规定的组不存在)

ERR\_BAD\_OBJECT\_ID (组中不存在对象)

ERR\_OBJECT\_PRIVATIZED (对象已秘密化)

## 写对象(0DH)

发送数据

0DH，组 ID，组 PIN，对象 ID，对象大小，对象数据

接收数据

如果成功，CSB = 0，否则为适当的差错代码

输出长度 = 0

输出数据 = 0

注意：

如果组 ID，组 PIN 和对象 ID 都正确，那么模块检查指定对象的属性字节。如果对象没有锁定或者没有秘密化，那么模块将对象先前的大小和数据清零，并用新的对象数据代替。注意，对象类型和属性字节不受影响。

写对象命令的差错返回代码可能有：

ERR\_BAD\_GROUP\_PIN (不正确的组 PIN)

ERR\_BAD\_GROUP\_ID (规定的组不存在)

ERR\_BAD\_OBJECT\_ID (组中不存在对象)

ERR\_BAD\_OBJECT\_SIZE (规定了不合法的对象大小)

ERR\_OBJECT\_LOCKED (对象已锁定)

ERR\_OBJECT\_PRIVATIZED (对象已秘密化)

# 命令 - 读组名

## 读组名(0EH)

发送数据

0EH , 组 ID

接收数据

CSB = 0

输出长度 = 组名的长度

输出数据 = 组名

注意:

组名长度最大值为 16 字节。在组名中，所有的字节值都是合法的。

## 删除组(0FH)

发送数据

0FH , 组 ID , 组 PIN

接收数据

如果成功， CSB = 0， 否则为适当的差错代码

输出长度 = 0

输出数据 = 0

注意:

如果组 PIN 和组 ID 正确，那么模块将删除指定的组。删除一个组会自动破坏该组内的所有对象。如果已经锁定模块，删除组命令将失效。

删除组命令的差错代码可能有:

ERR\_BAD\_GROUP\_PIN (不正确的组 PIN)

ERR\_BAD\_GROUP\_ID (指定的组不存在)

ERR\_MIAC\_LOCKED (模块已锁定)

## 获取命令状态信息(10H)

发送数据

10H

接收数据

00000001

CSB = 0

输出长度 = 6

输出数据 = 模块状态结构(见下文)

注意:

该操作不需要 PIN，并且永远不会失败。状态结构定义如下:

最后执行的命令 (1 字节)

最后命令的状态 (1 字节)

接收到的时间命令 (4 字节)

### 获取模块配置信息(11H)

发送数据

11H

接收数据

CSB = 0

输出长度 = 4

输出数据 = 模块配置结构

注意:

该操作不需要 PIN，并且永远不会失败。配置结构定义如下:

组数 (1 字节)

标志字节(见下文) (1 字节)

审计跟踪的大小/空 RAM (2 字节)

标志字节是对任何以下值的逐位“或”操作:

00000001b (锁定模块)

00000010b (需要访问的公共 PIN)

### 读审计跟踪信息(12H)

发送数据

12H， 公共 PIN

接收数据

如果命令成功，CSB = 0，否则为合适的差错代码  
如果成功，输出长度 = 审计跟踪结构大小(5)，否则为 0  
如果成功，输出数据 = 审计跟踪信息结构，否则为 0

注意：

如果发送的公共 PIN 有效，并且模块已锁定，那么它返回以下审计跟踪配置信息：

所用的交易记录数	(2 字节)
空的交易记录数	(2 字节)
规定审计跟踪是否因先前的读命令而卷动的布尔符号	(1 字节)
读审计跟踪信息命令的差错代码可能有：	
ERR_BAD_COMMON_PIN	(公共 PIN 不正确)
ERR_MIAC_NOT_LOCKED	(不锁定模块)

### 读审计跟踪(13H)

发送数据

13H，公共 PIN

接收数据

如果命令成功，CSB = 0，否则为合适的差错代码  
如果成功，输出长度 = 新记录的 # \*6，否则为 0  
输出数据 = 新的审计跟踪记录

注意：

如果发送的公共 PIN 有效，并且模块已锁定，那么它将把所有新的交易记录传送给主机。

读审计跟踪命令的差错代码可能有：

ERR_BAD_COMMON_PIN	(公共 PIN 不正确)
ERR_MIAC_NOT_LOCKED	(不锁定模块)

### 读组审计跟踪(14H)

发送数据

14H，组 ID，组 PIN

# 读组审计跟踪记录

## 接收数据

如果命令成功，CSB = 0，否则为合适的差错代码

如果成功，输出长度 = 对组的记录或 # \* 6，否则为 0

如果成功，输出数据 = 对组的审计跟踪记录

## 注意：

该命令与读审计跟踪命令相同，只是仅将包含有发送数据中指定的组 ID 的记录返回给主机。这允许交易组记录跟踪它们自己的活动，不需要参照其它组的记录。

读组审计跟踪命令的差错代码可能有：

ERR\_BAD\_GROUP\_ID (组 ID 不存在)

ERR\_BAD\_GROUP\_PIN (公共 PIN 不正确)

ERR\_MIAC\_NOT\_LOCKED (不锁定模块)

## 读实时时钟(15H)

### 发送数据

15H，公共 PIN

### 接收数据

如果公共 PIN 匹配，CSB = 0，否则 ERR\_BAD\_COMMON\_PIN

输出长度 = 4

输出数据 = 实时时钟的 4 个最高字节

## 注意：

该值不用时钟偏移调整。该命令通常由服务供应者使用，以便在交易组创建期间计算时钟偏移。

## 读经调整的实时时钟(16H)

### 发送数据

16H，组 ID，组 PIN，偏移对象的 ID

### 接收数据

如果成功，CSB = 0，否则为合适的差错代码

如果成功，输出长度 = 4，否则为 0

输出数据 = 实时时钟 + 时钟偏移 ID

注意：

如果组 ID 和组 PIN 有效，并且对象 ID 是时钟偏移的 ID，那么该命令成功，模块将时钟偏移加到 RTC 的 4 个最高字节的当前值上，并将该值返回在输出数据字段中。注意，可以写一个交易脚本，进行相同的任务，并把结果放在输出数据对象中。

读经调整的实时时钟命令的差错代码可能有：

ERR\_BAD\_GROUP\_PIN (组 PIN 不正确)

ERR\_BAD\_GROUP\_ID (指定的组不存在)

ERR\_BAD\_OBJECT\_TYPE (对象 ID 不是时钟偏移)

### 获取随机数据(17H)

发送数据

17H，长度(L)

接收数据

如果成功，CSB = 0，否则为合适的差错代码

如果成功，输出长度 = L，否则为 0

如果成功，输出数据 = 随机数据的 L 字节

注意：

该命令提供了一个良好的保密有用的随机数源。

获取随机数据命令的差错代码可能有：

ERR\_BAD\_SIZE (需要的字节数>128)

### 获取固件版本 ID(18H)

发送数据

18H

接收数据

CSB = 0

输出长度 = 固件版本 ID 串的长度

输出数据 = 固件版本 ID 串

注意:

该命令把固件版本 ID 作为一个 Pascal 类型的串(长度 + 数据)返回。

### 获取空的 RAM(19H)

发送数据

19H

接收数据

CSB = 0

输出长度 = 2

输出数据 = 包含空的 RAM 数量的 2 字节值

注意:

如果模块已锁定, 输出数据字节将均为 0, 表示所有未被交易组使用的存储空间都保留下用于审计跟踪。

### 改变组名(1AH)

发送数据

1AH, 组 ID, 组 PIN, 新的组名

接收数据

如果成功, CSB = 0, 否则为合适的差错代码

输出长度 = 0

输出数据 = 0

注意:

如果模块中存在指定的组 ID, 并且提供的 PIN 是正确的, 那么用主机提供的新的组名替代交易组名。如果提供组 ID 为 0, 那么发送的 PIN 必须是公共 PIN。如果它是正确的, 那么用主机提供的新的名字替代模块名。

改变组名命令的差错代码可能有:

ERR\_BAD\_GROUP\_PIN (组 PIN 不正确)

ERR\_BAD\_GROUP\_ID (指定的组不存在)

ERR\_BAD\_NAME\_LENGTH (新的组名>16 字节)

## 差错代码定义

### ERR\_BAD\_COMMAND(80H)

当模块固件不认识主机刚发送的命令时，产生该差错代码。

### ERR\_BAD\_COMMON\_PIN(81H)

当命令需要一公共 PIN 并且提供的 PIN 与模块的公共 PIN 不匹配时，并返回该差错代码。最初，将公共 PIN 设置为 0。

### ERR\_BAD\_GROUP\_PIN(82H)

交易组可以有它们自己的 PIN，图 11。如果已设置该 PIN(由设置组 PIN 命令设置)，那么必须提供它来访问组内的任何对象。如果提供的组 PIN 与实际的组 PIN 不匹配，那么模块将返回 ERR\_BAD\_GROUP\_PIN 差错代码。

### ERR\_BAD\_PIN\_LENGTH(83H)

有 2 个命令可以改变 PIN 值。它们是设置组 PIN 命令和设置公共 PIN 命令。这两个命令都需要新的 PIN 和旧的 PIN。如果提供的旧的 PIN 是正确的，但新的 PIN 的长度大于 8 个字符，那么将返回 ERR\_BAD\_PIN\_LENGTH 差错代码。

### ERR\_BAD\_OPTION\_BYTE(84H)

选项字节只应用于公共 PIN。当执行设置公共 PIN 命令时，主机提供的最后字节是选项字节(命令章节中有描述)。如果模块不认识该字节，那么它将返回 ERR\_BAD\_OPTION\_BYTE 差错代码。

### ERR\_BAD\_NAME\_LENGTH(85H)

当创建交易组命令时，主机提供的一个数据结构是组名。组名的长度不可以超过 16 个字符。如果提供的名字长于 16 字符，那么返回 ERR\_BAD\_NAME\_LENGTH 差错代码。

### ERR\_INSUFFICIENT\_RAM(86H)

当模块中没有足够的堆时，创建交易组命令和创建对象命令返回该差错代码。

#### ERR\_MIAC\_LOCKED(87H)

当已锁定模块时，不能创建或破坏任何组或对象。创建或删除对象的任何尝试都将产生 ERR\_MIAC\_LOCKED 差错代码。

#### ERR\_MIAC\_NOT\_LOCKED(88H)

如果已锁定模块，那么不存在审计跟踪。如果执行一个审计跟踪命令，那么将返回该差错代码。

#### ERR\_GROUP\_LOCKED(89H)

一旦锁定了某个交易组，就不可能在该组内创建对象。对象属性和类型也被冻结。创建对象或改变它们属性或类型字节的任何尝试都将产生 ERR\_GROUP\_LOCKED 差错代码。

#### ERR\_BAD\_OBJECT\_TYPE(8AH)

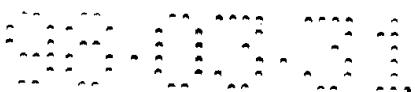
当主机向模块发出创建对象命令时，它提供的一个参数是对象类型(见命令章节)。如果固件不认识该对象类型，那么它将返回 ERR\_BAD\_OBJECT\_TYPE 差错代码。

#### ERR\_BAD\_OBJECT\_ATTR(8BH)

当主机向模块发出创建对象命令时，它提供的一个参数是对象属性字节(见命令章节)。如果固件不认识该对象属性类型，那么它将返回 ERR\_BAD\_OBJECT\_ATTR 差错代码。

#### ERR\_BAD\_SIZE(8CH)

当创建或写一个对象时，常会产生 ERR\_BAD\_SIZE 差错代码。只有当主机提供的对象数据具有无效长度时，才会发生。



#### ERR\_BAD\_GROUP\_ID(8DH)

在交易组层次上操作的所有命令都要求在命令数据包中提供组 ID。如果模块中不存在指定的组 ID，那么它将产生 ERR\_BAD\_GROUP\_ID 差错代码。

#### ERR\_BAD\_OBJECT\_ID(8EH)

在对象层次上操作的所有命令都要求在命令数据包中提供对象 ID。如果特定交易组(交易组也在命令数据包中指定)内不存在指定的对象 ID，那么它将产生 ERR\_BAD\_OBJECT\_ID 差错代码。

#### ERR\_INSUFFICIENT\_FUNDS(8FH)

如果调用执行金融交易的脚本对象，并且货币寄存器的值小于需要提取的金额，那么返回 ERR\_INSUFFICIENT\_FUNDS 差错代码。

#### ERR\_OBJECT\_LOCKED (90H)

锁定对象是只读的。如果试图执行写对象命令，并且它规定了锁定对象的对象 ID，那么模块将返回 ERR\_OBJECT\_LOCKED 差错代码。

#### ERR\_OBJECT\_PRIVATE (91H)

秘密对象是不能直接读或写的。如果试图执行读对象命令或写对象命令，并且它规定了秘密对象的对象 ID，那么模块将返回 ERR\_OBJECT\_PRIVATE 差错代码。

#### ERR\_OBJECT\_DESTRUCTED (92H)

如果对象是可破坏的，并且交易组的破坏器在工作，那么脚本不可以使用该对象。如果调用的脚本使用已被破坏的对象，那么模块将返回 ERR\_OBJECT\_DESTRUCTED 差错代码。

最好将本发明的实施例放在耐用的硬币式不锈钢罐中。应该理解，可以将所示的模块放在任何实质上带关节的物品中。带关节的物品例子包括信用卡、戒子、手表、皮夹、钱包、项链、珠宝、ID 徽章、钢笔、书写板等。

模块最好是单个芯片的“信托计算机”。用术语“信托”表示计算机是极其

保密的，未获承认的方法不能窜改。模块包含一数字协处理器，被优化用来进行数学加强的加密。BIOS 最好是不能改变的，并且特别是为非常保密的交易设计的。

每个模块具有一个随机“种子”发生器，该发生器具有创建秘密/公开密钥集的能力。秘密密钥永远不会离开模块，并且只有模块知道。另外，当错误进入模块时通过积极进行自破坏，防止秘密密钥被发现。可以用一个个人身份识别号(PIN)将模块与用户联系在一起。

当用模块进行交易时，由模块和与模块通信的系统这两者之一或两者创建授权证明。证明可以包含各种信息。特别是，证明可以包含：

- 1)谁是模块用户，借助于专用的登记号。
- 2)交易何时发生，借助于交易的真实时间标志。
- 3)交易发生在何地，借助于经登记的模块接口位置标识。
- 4)保密信息，借助于消息摘要上的独特排序的交易和数字签名。
- 5)模块状态，表示成有效、丢失、或截止。

尽管附图中示出并且在以上详细说明中描述了本发明方法和设备的一个较佳实施例，但应该理解，本发明不限于所揭示的实施例，本发明可以不脱离以下权利要求书叙述和限定的本发明的精神，进行众多的重新布置、变化和替代。

## 说 明 书 附 图

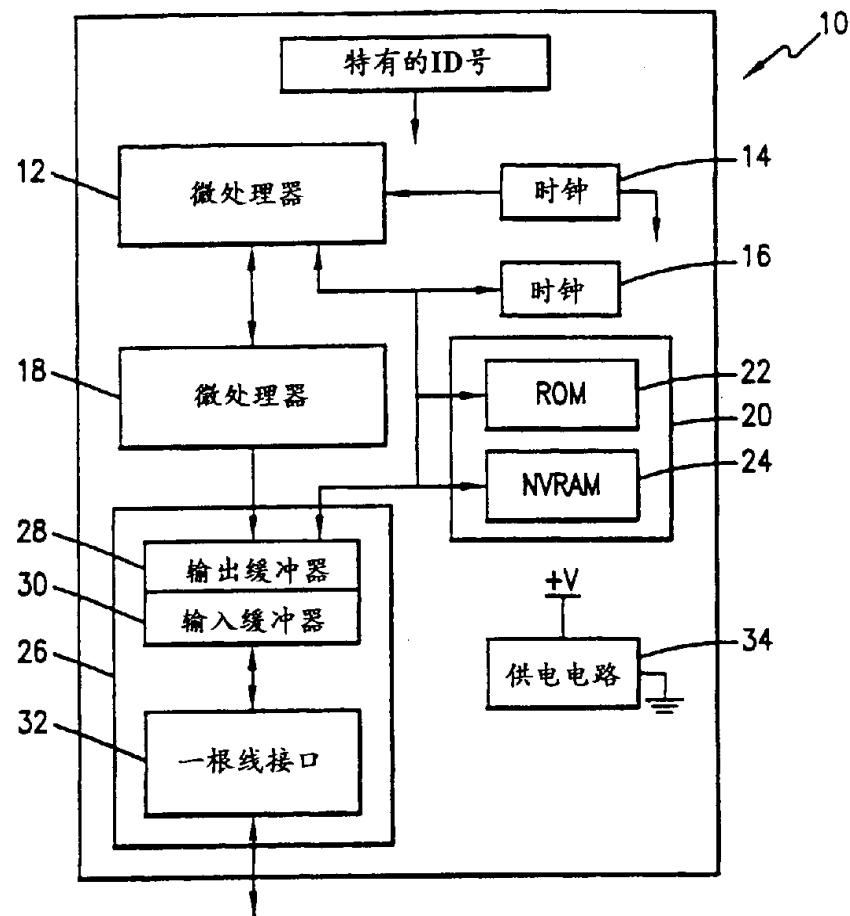


图 1

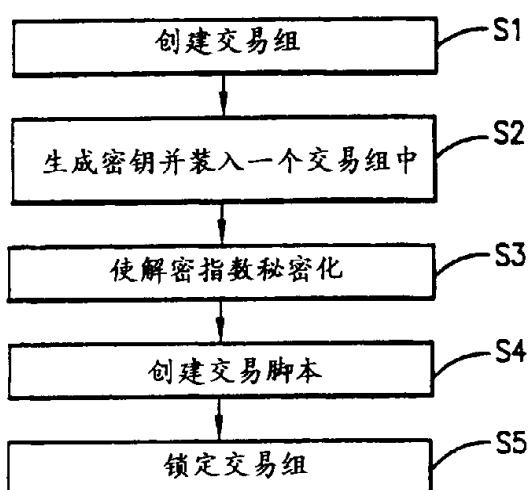
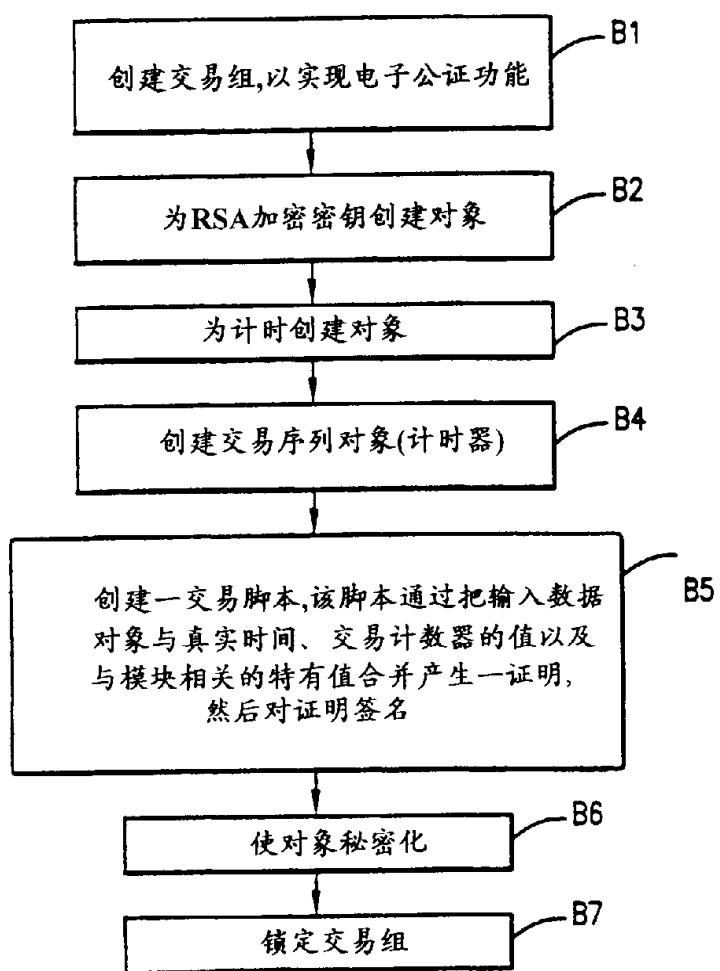
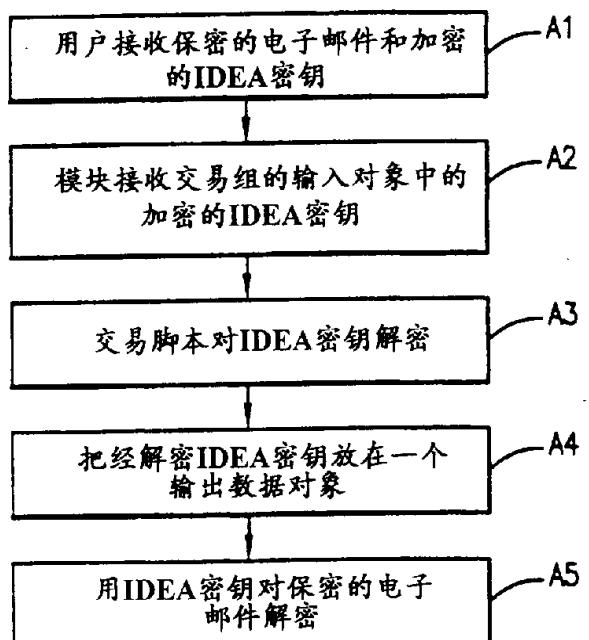


图 2



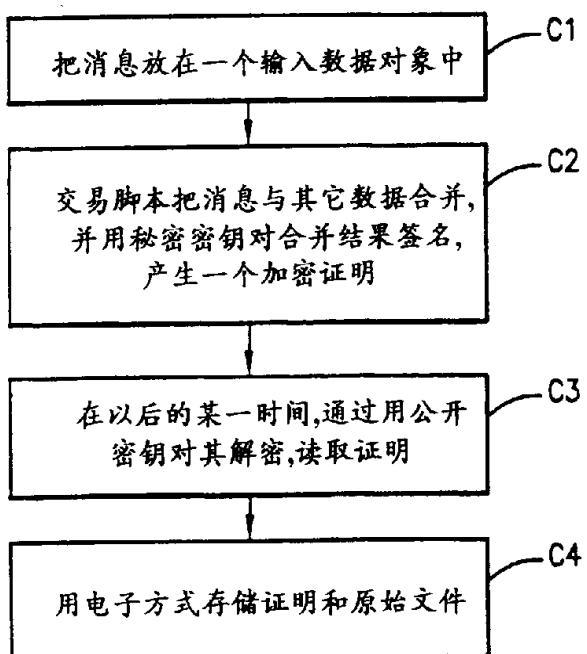


图 5

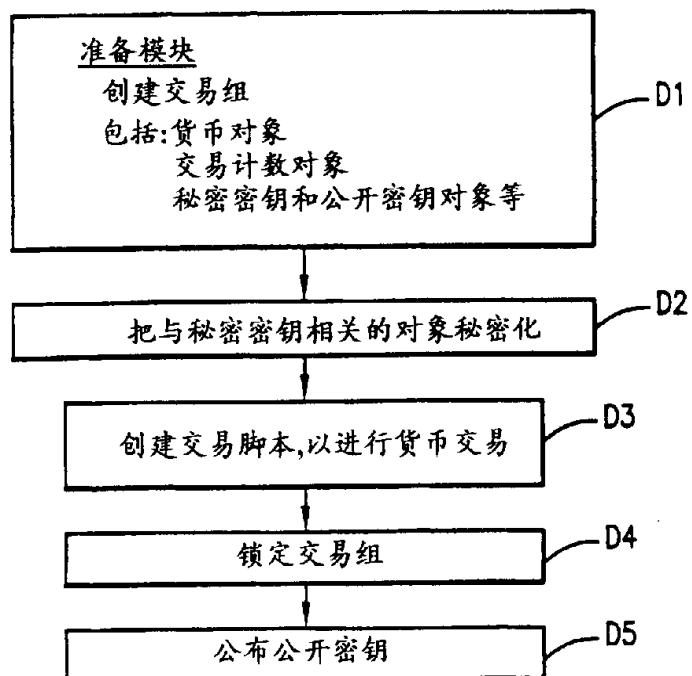


图 6

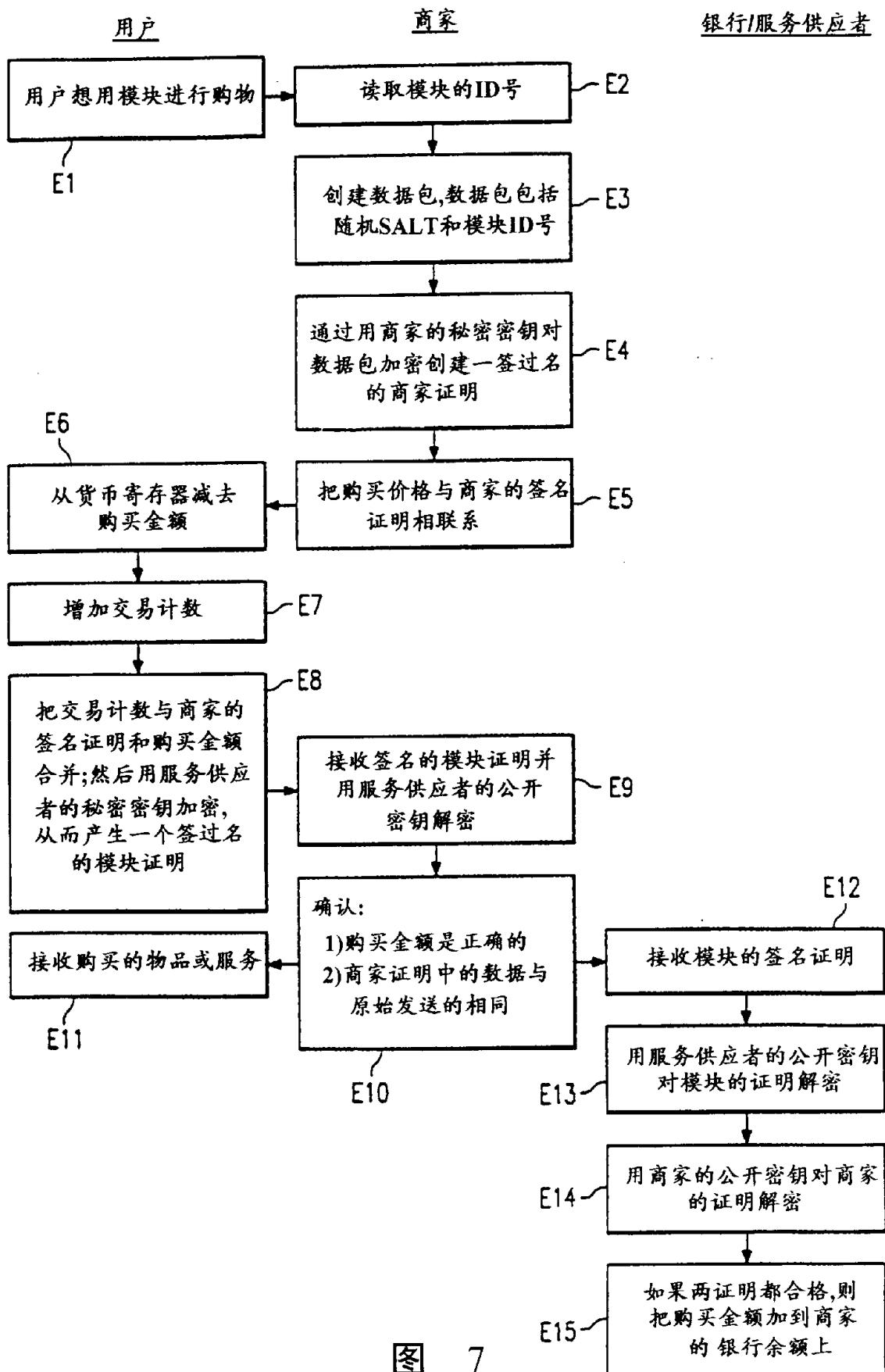


图 7

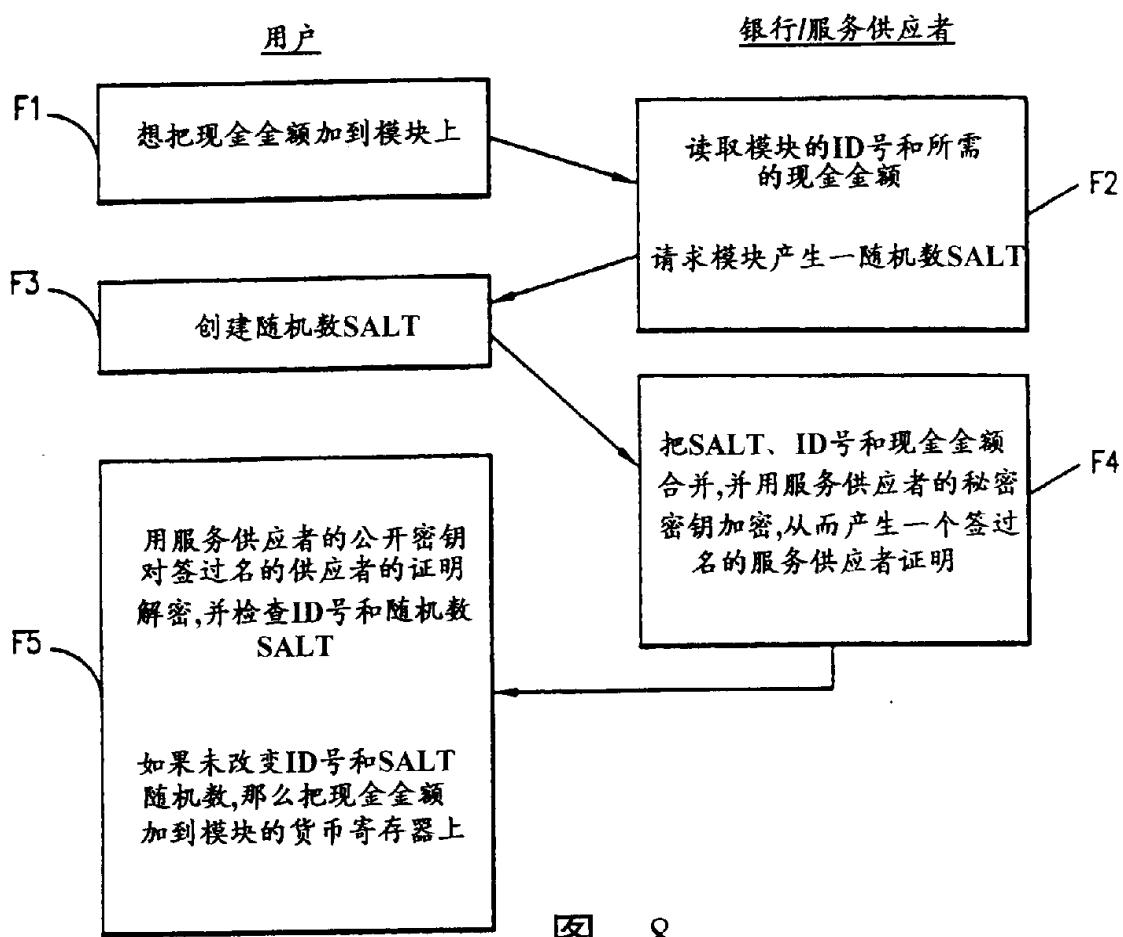


图 8

从用户模块至商家模块传送的例子

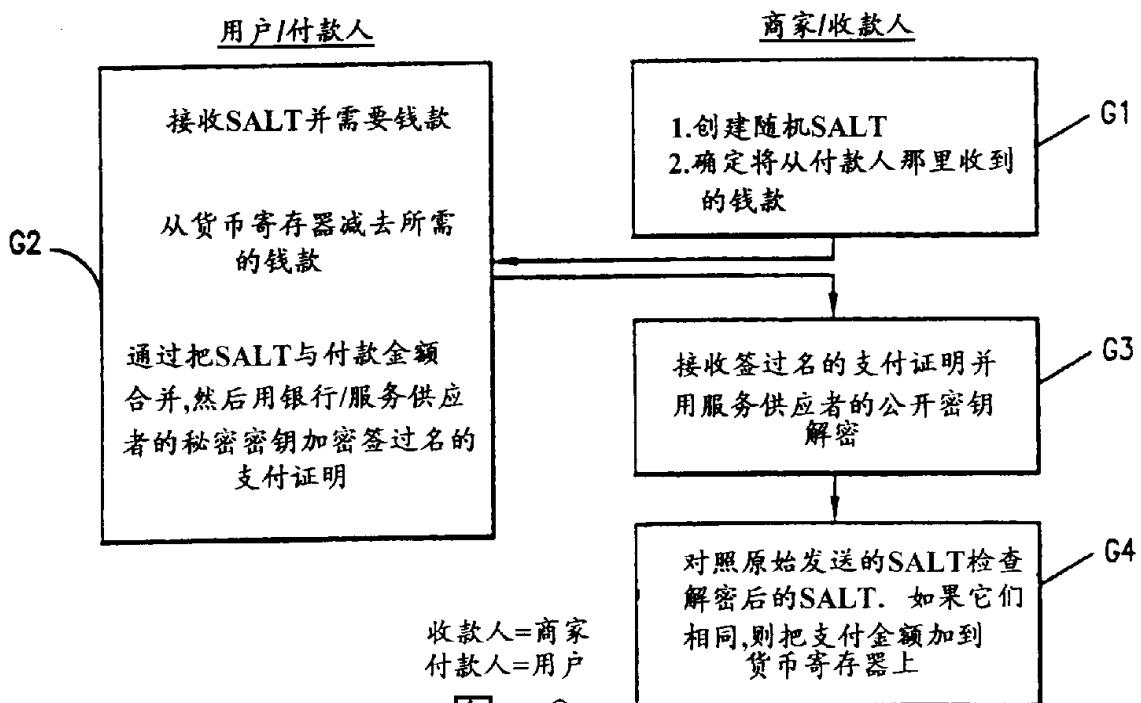


图 9

用模块通过网络端交易

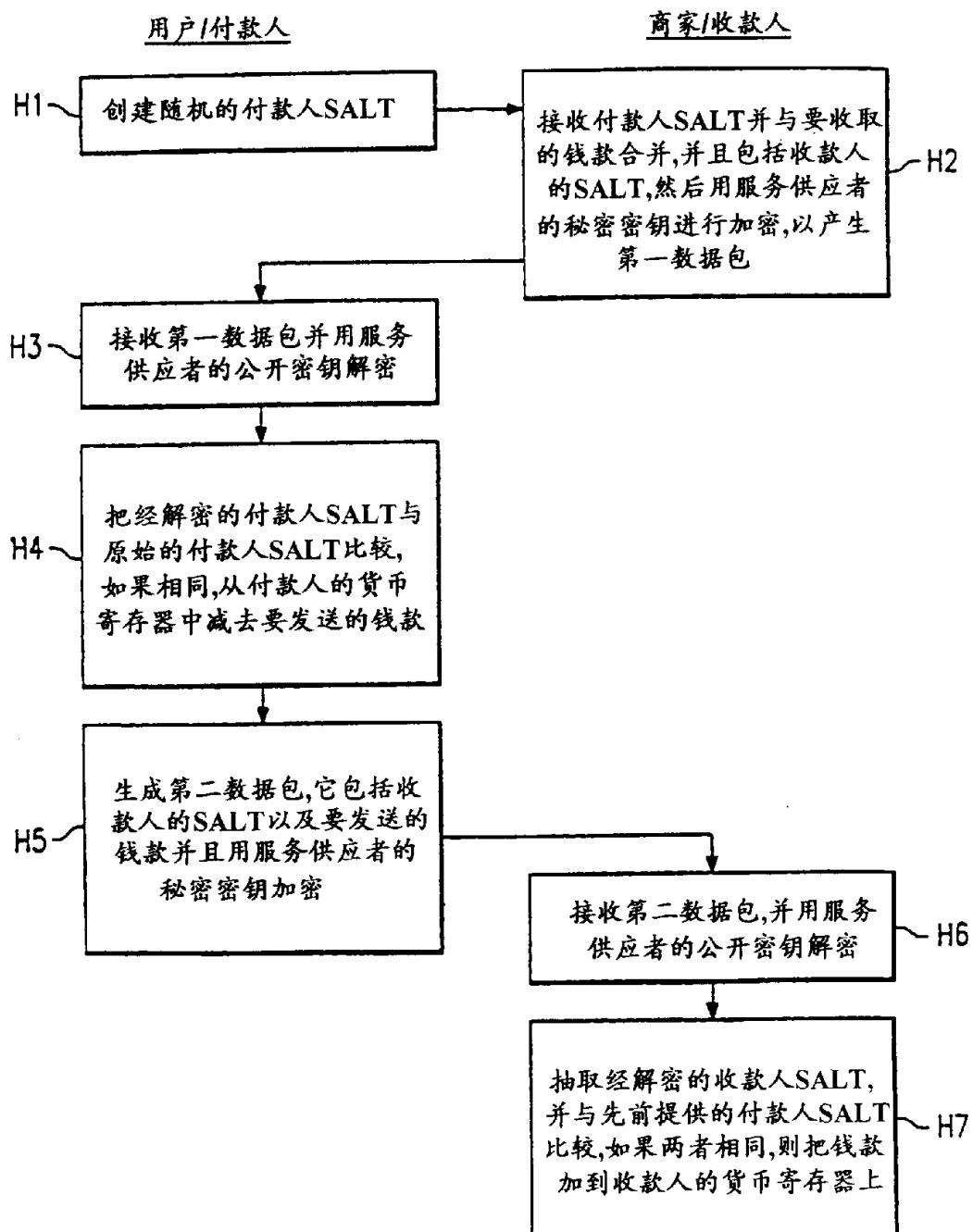


图 10

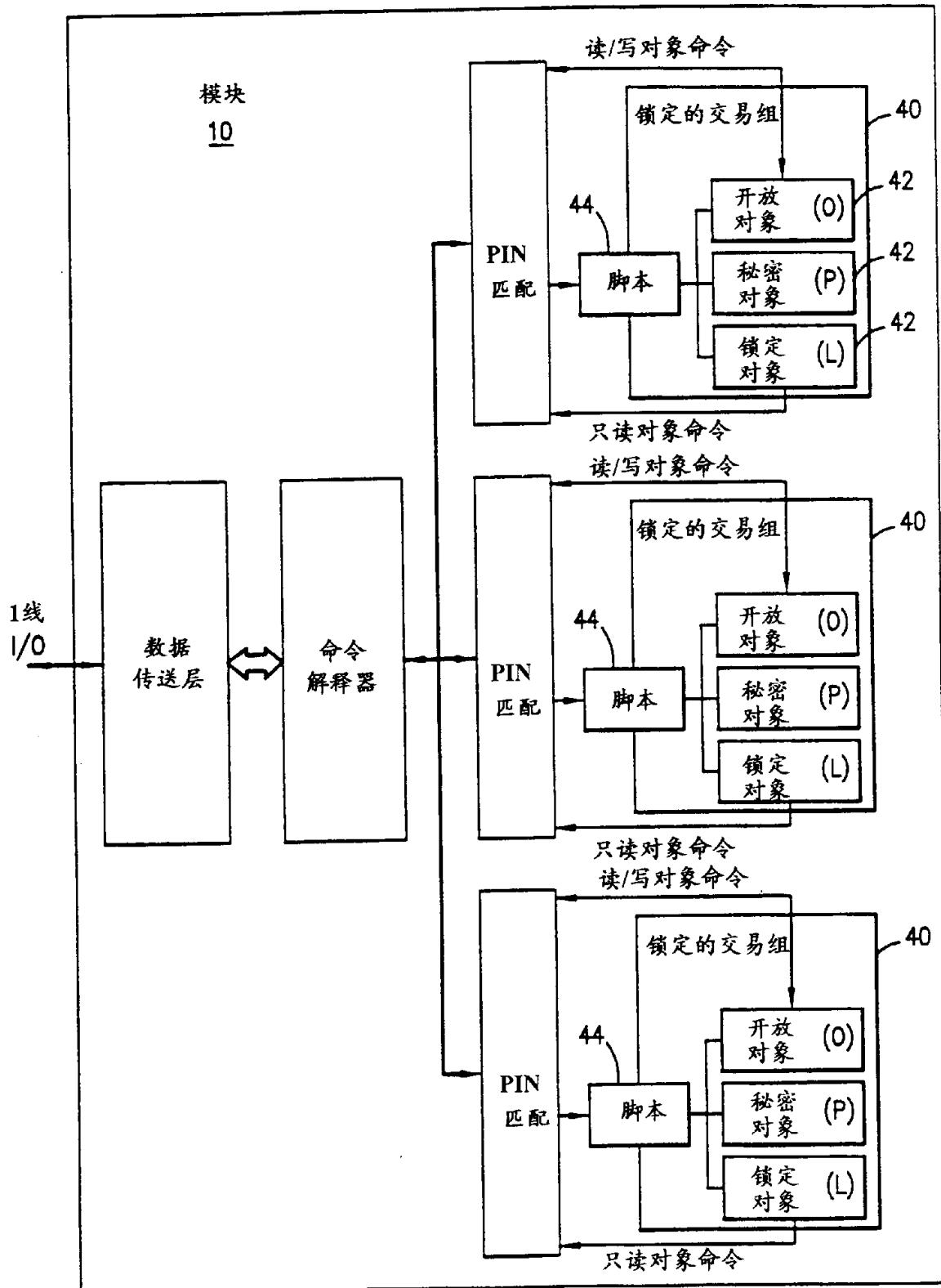


图 11

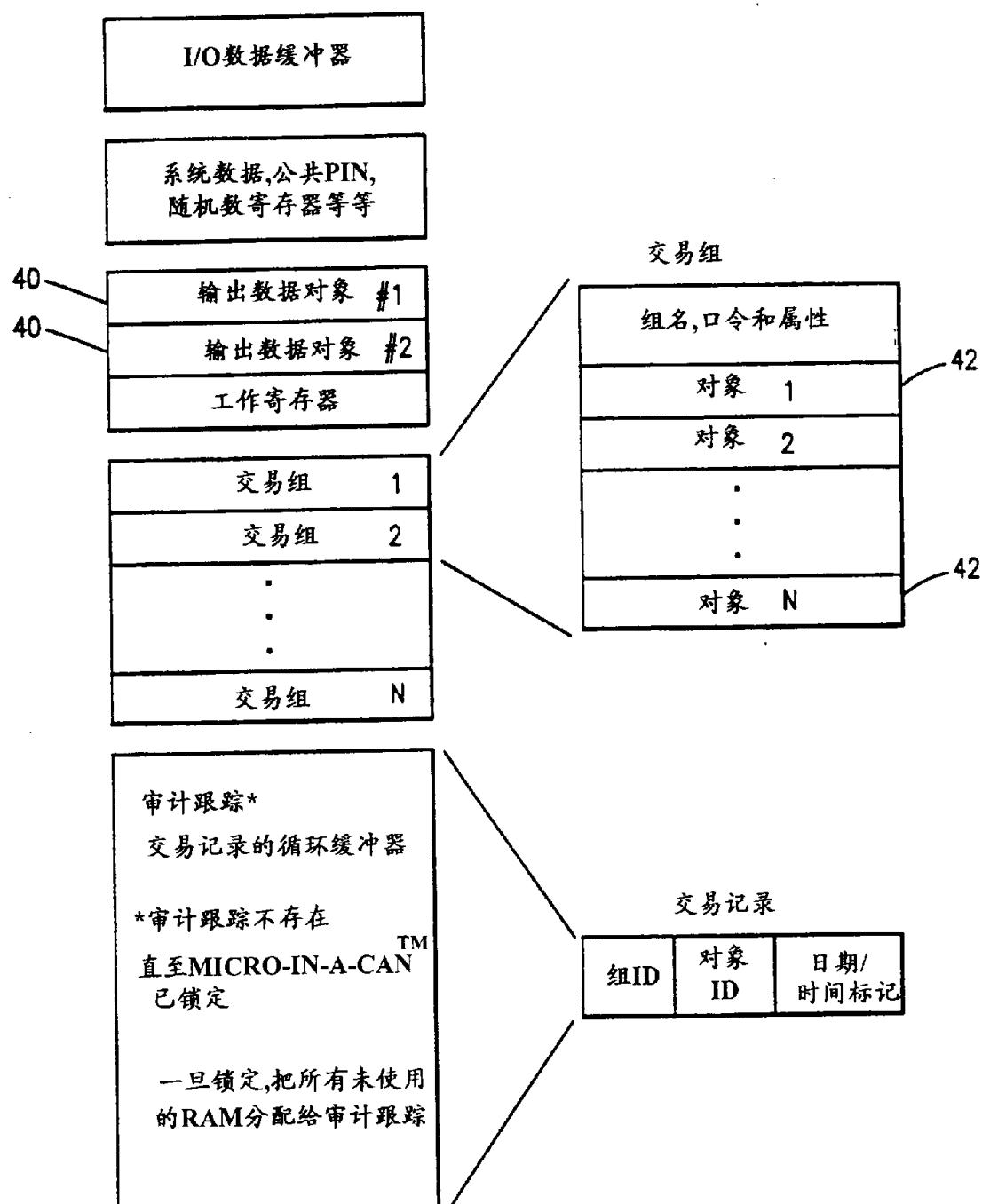


图 12