基于可信计算平台的可信动态客体监管系统的设计与实现*)

谭 良[1,2] 周明天[2]

(四川师范大学四川省软件重点实验室 成都 610066)[1]

(电子科技大学计算机科学与工程学院 成都 610054)[2]

摘 要 大部分安全操作系统在处理客体时,没有区分具体的客体类型,均采用统一的方法进标识,而且多数安全操作系统中采用传统访问控制方法保护的客体均是静态客体,忽略了对动态客体的保护,使得黑客或攻击者可以进行欺骗和中间人攻击,因此,安全操作系统中的动态客体并不可信。首先分析了操作系统中客体的类型,提出了可信动态客体的概念,并分析了其特点。为了防止动态客体泄露信息,提出了基于 TPM 的可信动态客体监管系统。一方面,该系统要求主体必须在 TPM 中注册,确保主体身份合法性,才能利用可信动态客体进行信息传递;另一方面,在 TPM 中必须存储创建该可信动态客体的属主的身份信息,以确保可信动态客体身份的合法性。最后进行了安全和性能分析,分析表明该可信动态客体监管系统可以阻止黑客利用动态客体进行欺骗和中间人攻击,防止信息泄露,为进一步建立可信计算环境提供了基础。

关键词 安全操作系统,客体,可信操作系统,可信动态客体,可信计算平台

Design and Implementation of the Monitor System for the Trusted Dynamic Object Based on the TPM

TAN Liang[1,2] ZHOU Ming-Tian[1]

(School of Comp. Sci. & Engn., Univ. of Electronic Sci. & Tech. of China, Chengdu 610054)[1]

(college of Electronic Engineering, Sichuan Normal University, Chengdu 610066)[2]

Abstract Most security operating system doesn't distinguish the sorts of the object, and labels them by the same method, moreover, the objects, which is protected by the traditional access control policies in most security operating system, are the static object, and ignore the dynamic object which can be utilized to cheat or man-in-the-middle attack by the hacker. So the dynamic object in the security operating system isn't trustworthy. The object types in the operating system, which are sorted into the static object and the dynamic object, are analyzed, and the conception of the trusted dynamic object is put forward. For prevent the dynamic object from leaking the information, the monitor system for the trusted dynamic object based on TPM (MSTDOBT) is presented. On the one hand, the MSTDOBT needs that the subject, who uses the trusted dynamic object to transfer the information, must register in TPM for authentication, on the other hand, the owner of the dynamic object must register in TPM for authentication too. Finally, we discuss the security of the MSTDOBT, it is shown that the MSTDOBT can prevent the dynamic object form being utilized to cheat or man-in-the-middle attack by hacker. All of these are the foundation forour future works.

Keywords Security operating system, Object, Trusted operating system, Trusted dynamic object, TPM

1 引言

随着信息技术的飞速发展,现实世界越来越依赖于计算机系统。一方面,人们享受着由此带来的巨大进步;另一方面,又不得不承受着越来越严重的信息安全威胁。

操作系统实质是个资源管理系统,管理处理机、存储器、设备、文件和作业等计算机资源,用户通过它获得对资源的访问。安全操作系统的目的是保证它所管理资源的安全性,包括:机密性、完整性等。信息的保密性是为了防止信息在非授权情况下的泄露;而信息的完整性是为了保护信息不被非法篡改或破坏。为了保证系统中信息的安全性,安全操作系统采用多种安全模型和控制框架对内容的访问实施控制,并在访问控制框架和安全模型方面均取得了丰硕的成果。在访问控制框架方面有:基于政策描述语言的 FAM(Flexible Authorization Manager)[1]和企业间多协调框架[2],基于安全属性的 GFAC 框架[3],基于统一模型的数据库 FMP[4]、RBAC[5]和 Flask[6]框架。在安全模型方面,最重要和最知名的安全模型包括:BLP[7], HRU[8], BIBA[9], Lattice Model of Information Flow[10]、Chinese Wall[11]、DTE[12]等等。但是,纵观安全操作系统将近 40 年的发展历史,可以发现安全操作系统的主要应用范围仍然是在国防和军事领域,在商用和民用领域尚未有成熟的安全操作系统出现。迄今为止,在整个国际上,安全操作系统的实际应用并不成功,在实际应用中发挥作用的操作系统绝大部分不是安全操作系统。究其本质,安全操作系统还存在不完善的地方,还存在诸多缺陷。另外,随着可信计算技术的兴起[13~19],可信操作系统逐渐成为研究热点。操作系统的可信不是凭空而来的。可信性的建立不仅需要对操作系统自身和可能降低系统可信性的执行代码进行一致性度量,需要对用户登录进行可信验证,对登录的内部合法用户行为进行监管,而且需要对操作系统中各种客体的可

*)基金项目:国家 863 宽带 VPN 项目 863-104-03-01 课题资助;2003 年度四川省科技攻关项目 03GG007-007 支持。**谭 良** 博士,主要研究方向为信息安全、中间件;**周明天** 教授,博士生导师,主要研究方向为网络计算,信息安全,分布并行处理。

信性进行检查和监督。否则，会影响用户对可信操作系统的信任。

目前，世界上已经设计和开发出来一系列的安全操作系统，典型的有 Multics、Mitre 安全核、UCLA 数据安全 Unix、KSOS 和 PSOS、LINUS IV、Xenix、System V/MLS、TUNIS、ASOS、DTOS（Distributed Trusted Operating System）、Flask7、SE-Linux8，以及国内的 SUNIX、COSIX V2.0、LIDS、SoftOS、SecLinux 等[20]。

值得注意的是，大部分安全操作系统在处理客体时，没有区分具体的客体类型。在计算机系统中，存在着许多客体，如文件、目录、共享内存、消息、信号量、管道、存储器、缓冲器、磁盘和外部设备等。不同类型的客体具有不同的特点。在安全操作系统中，将不同种类的客体采用同一方法进行标识，并采用相同的方法进行访问控制是有缺陷的。

实际上，多数安全操作系统中采用访问控制方法保护的客体均是静态客体。静态客体具有如下特征[21]：(1)客体仅仅是主体的行为对象，一旦主体拥有权限，客体只能完全"接受"主题的行为。(2)客体和主体之间不能验证身份。操作系统中的文件目录就属于静态客体。在安全操作系统中，采用一定的访问控制政策就可以保证这类客体安全，并且不会给操作系统带来新的脆弱性。但是对于进程、进程间通信(包括管道和 IPC 对象)这两类客体，如果采用传统的访问控制策略，即使能保证这类客体的安全性，也会给操作系统带来新的脆弱性。例如，不管是 Windows、Unix 还是 Linux，管道都是一种进程(主体)之间进行通信的有效机制。当主体 *A* 将用户名和密码通过管道客体传给主体 *B* 时，黑客或攻击者可以伪装为主体 *B* 接收主体 *A* 传过来的机密信息而不被察觉。如果管道客体在传递信息之前对主体的身份进行认证，那么黑客或攻击者就会被发现。在 Windows 中，黑客或攻击者通过 RunAs 服务进程利用命名管道来提升权限就属于这种情况的典型案例[22]，因此不能将这两类客体看成一般的静态客体。

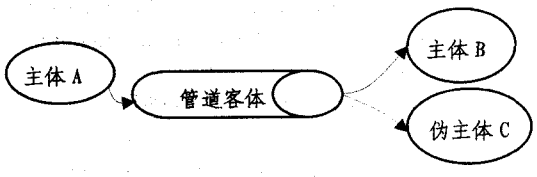

图 1　主体与伪主体之间的通信

本文将专注于讨论这两类客体。首先分析了操作系统中客体的类型，提出了可信动态客体的概念，并分析了其特点。为了防止动态客体泄露信息，提出了基于 TPM 的可信动态客体监管系统。一方面，该系统要求主体必须在 TPM 中注册，确保主体身份合法性，才能利用可信动态客体进行信息传递；另一方面，在 TPM 中必须存储创建该可信动态客体的属主的身份信息，以确保可信动态客体身份的合法性。最后进行了安全和性能分析，分析表明该可信动态客体监管系统可以阻止黑客利用动态客体进行欺骗和中间人攻击，防止信息泄露，为进一步建立可信计算环境提供了基础。

2　客体的分类

在操作系统中，存在着许多客体，如文件、目录、共享内存、消息、信号量、管道、存储器、缓冲器、磁盘和外部设备等。在安全操作系统中，为了不同安全政策的实施，可以对这些客体进行不同的分类。例如，在 OSR 模型[20]中，将除用户外的所有客体划分为五个种类：进程、文件目录、进程间通信、设备和系统控制数据等五个种类。"系统控制数据"包括系统时钟、主机名和域名等在整个系统范围内起作用的数据。"文件目录"表示系统中文件和目录的集合。因此，为了便于研究客体的可信性，我们将客体分为静态客体和动态客体。

定义 1　将在逻辑上只能作为主体行为对象的客体称为静态客体。

分析定义 1 可知，静态客体只能作为主体行为的受体。一旦主体拥有权限，客体只能完全"接受"主题的行为，客体没有根据自身的实际情况与主体的访问进行协商的权利，当然这类客体也不能验证主体的身份。显然，文件目录、设备和系统控制数据属于静态客体。

定义 2　将在逻辑上既可以作为主体的行为对象，也可以对其它客体或主体施以行为的客体，称为动态客体。

分析定义 2 可知，动态客体不仅作为行为的受体，而且也可以对其它主体或客体施以行为，因此，动态客体既具有静态客体的特征，又具有一般主体的特征。

显然，进程就属于动态客体。如：httped 服务进程，当客户端进程向 httped 服务进程发送请求时，httped 服务进程作为客户端进程的客体，接受该请求；为了处理这个请求，httped 服务进程将该请求转移给其它功能进程进行处理，此时，httped 服务进程作为其它功能进程的主体。与 httped 服务进程类似的进程如：telneted 服务进程、ftped 服务进程等。

进程间通信是不是也属于动态客体呢？当前，在 Windows 或 Linux 操作系统中，实现进程间通信的是管道和 IPC 对象，管道通信如图 1 所示；IPC 对象通信如图 2 所示。从具体实现上来看，要实现进程 A 和进程 B 的通信，需要进程 A 写 IPC 对象或管道，进程 B 去读 IPC 对象或管道来实现，此时的 IPC 对象或管道均是行为的受体，与一般的静态客体没有区别。但从逻辑上看，进程 A 将信息传递给 IPC 对象或管道，然后由 IPC 对象或管道将该信息传递给进程 B，此时的 IPC 对象或管道既作为进程 A 的客体，又作为进程 B 的主体。因此，IPC 对象或管道既是主体，又是客体。所以，从逻辑上看，IPC 对象或管道是动态客体。黑客或攻击者就是利用这一点进行攻击的。

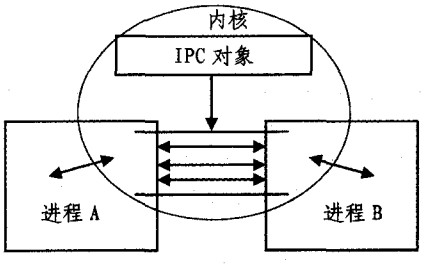

图 2　IPC 对象让无关进程交换数据成为可能

3　可信动态客体的基本概念及其特点

在操作系统中，需要大量动态客体。不管是进程、IPC 对象还是管道，它们都是信息传递的纽带，主体之间进行协作的有效手段。也是操作系统中容易出现脆弱性的地方。如果不加以保护，很容易被攻击者攻击。具体的攻击方式有：

1. 攻击者伪装主体 1 通过动态客体向主体 2 发送虚假消息，如图 3 所示。

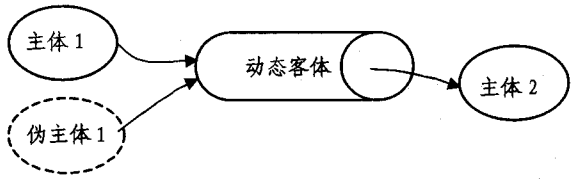

图 3　伪主体 1 通过动态客体发送虚假消息给主体 2

2. 攻击者伪装主体 2 接受通过动态客体传递过来的消息，导致重要信息的泄密，如图 4 所示。

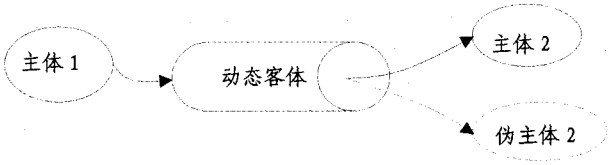

图 4　伪主体 2 接受通过动态客体传递过来的消息

3. 攻击者伪装动态客体将主体 1 的信息传递给伪主体 2，导致重要信息的泄密，如图 5 所示。

图 5　伪主体 2 接受通过伪动态客体传递过来的消息

因此，在操作系统中，必须对动态客体实施保护措施，进行监督与管理，才可能防止虚假信息的泛滥，重要信息的泄露。

定义 3　从逻辑上看，当动态客体作为行为的受体时，它既可以验证施以行为的主体身份，又可以保证内容是安全的、可信的；当作为行为的发起者时，它可以验证行为对象的身份。这样的动态客体称为可信动态客体。

分析定义 3 可知，可信动态客体具有如下特征：

1. 传递性：可信动态客体是活性客体，能够将某个主体的信息传递给另一个主体。

2. 真实性：可信动态客体不仅需要向其它主体表明自己身份的真实性，而且还必须验证其它主体身份的真实性。

从可信动态客体的特征可以看出，可信动态客体既可以作为主体之间信息共享和传递的通道，又可以防止欺骗和中间人攻击。当作为行为的受体时，它可以验证施以行为的主体身份；当作为行为的发起者时，它可以验证行为对象的身份。对于需要利用可信动态客体进行信息共享和传递的主体也可以验证该可信动态客体的身份。

4　基于 TPM 的可信动态客体的设计与实现

4.1　TPM 简介

TPM 是可信计算技术的核心，是一个含有密码运算部件和存储部件的小型片上系统。通过 LPC 总线与 PC 芯片集结合在一起，将重要的数据信号线或重要的存储区域严密保护起来，这样用人为的物理探头或一般的光探测技术就很难窥探到里面的数据。TPM 在封装时可用一信号探测的方式防拔除。如有人拔除则会触动一根预先埋好的信号线其上的信号将发生变化，从而激发一个硬中断，系统将直接跳到自毁灭中断处理程序中，从而将自己内部的数据全部销毁，整个系统也就自动毁灭，无法复原。因此，TPM 是一个具有较强的密码计算能力和存储能力、完全可信的“信任根”。

4.2　基于 TPM 的可信动态客体的逻辑结构

基于 TPM 的可信动态客体监管系统（the Monitor System for the Trusted Dynamic Object Based on TPM：MSTDOBT）将主体对可信动态客体的可信访问分为六部分：主体身份鉴别模块、可信动态客体身份鉴别模块、信息传输模块、加密签名模块、TPM 应用程序接口和 TPM，如图 6。

- 主体身份鉴别模块（Subject Authentication Module）：其功能是用来鉴别主体的身份；身份鉴别是访问可信动态客体的第一道关口。主体在访问可信动态客体之前，首先要经过身份认证系统识别身份。
- 可信动态客体身份鉴别模块（Trusted Dynamic Object Authentication Module）：其功能是用来验证可信动态客体的身份。验证可信动态客体的身份是指验证该可信动态客体属主的身份。如：对于管道，创建该管道的主体就是该管道的属主。在验证管道这个动态客体的时候，需要验证创建该管道属主的身份。
- 加密签名模块（Security Module）：其功能主要是为存储在可信动态客体中的信息提供加密、签名的功能。
- TPM 访问接口模块（TPM Access Interface Module：TPM_AIM）提供对 TPM 的访问接口，包括存储控制接口、密码运算接口等。
- TPM 用来存储可信静态客体的映像文件和提供密码运算。

值得注意的是，图 6 中，在主体 A 和可信动态客体之间，以及可信动态客体与主体 B 之间，均应进行主体身份鉴别和可信动态客体身份鉴别。为了图形直观，我们只在主体 A 和可信动态客体之间调用了主体身份鉴别模块，在可信动态客体与主体 B 之间调用了可信动态客体身份鉴别模块。

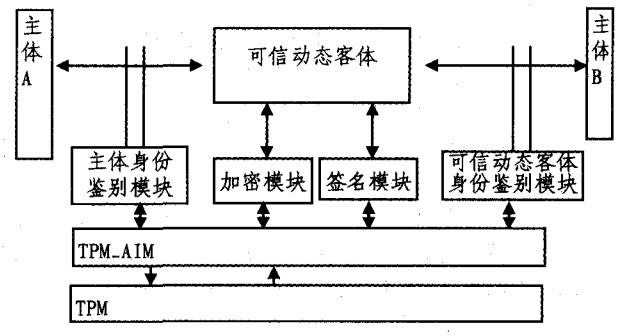

图 6　伪主体 2 接受通过伪动态客体传递过来的消息

4.3　基于 TPM 的可信动态客体监管系统的工作原理

4.3.1　对主体的要求

在操作系统中，主体要访问可信动态客体，它必须首先成为 TPM 的合法用户。主体在注册时，应将表 1 所示的用户身份信息、证书信息及私钥信息存放在 TPM 内部的 Flash 中，并防止非法用户对这类数据的读取。

表 1　用户信息表

用户名	用户 ID	证书	私钥
U_1	U_ID_1	C_1	PK_1
U_2	U_ID_2	C_2	PK_2
U_3	U_ID_3	C_3	PK_3

4.3.2 可信动态客体的可信标识

在操作系统中，主体对可信动态客体的访问，应该进行身份认证和访问控制，因此，系统应该对可信动态客体进行标识。但由于可信动态客体包含进程、管道和 IPC 对象这三类动态，因此，对三类可信动态客体的标识应该有所不同。

对于进程，如前面提到的 httped 服务进程、telneted 服务进程、ftped 服务进程等。主体在访问对这类动态客体时候，需要进行双向的身份认证并对传输的信息进行加密。在当前网络安全的解决方案中，这类产品或系统很多，如：AAA 服务器、Kerberbos 服务器等。应该说，对于进程这一类的动态客体，要使它们可信，只需要通过对现有安全系统或产品的集成，就可以满足要求，不需要特殊的标识。

对于管道，包括无名管道和有名管道，无名管道通常用在同族进程之间进行通信，由于同族进程的主体之间是父子、子孙关系，因此属于同一用户，而且同族进程拥有相同的逻辑地址空间，所以在利用无名管道进行通信的时候，不存在身份认证和信息泄露的问题。但对于有名管道，情况就不一样。多数操作系统在处理有名管道时，都把有名管道看是一个特殊的文件，存放在文件系统中。因此，对有名管道的处理，应参考安全操作系统对静态客体的处理方式。在安全操作系统中实现主体对文件客体的访问时，一种常用方法是访问控制列表(Access Control Lists)，又称为 ACLs。在这种实现方法中，每一个客体与一个 ACL 相对应：指明系统中的每一个主体获得对此客体访问的相应授权，如读、写、执行等等。为了识别这类可信动态客体，并启动可信动态客体监管系统，可以在 ACLs 中定义可信标识，使得原 ACL 项由一个三元组变成了一个四元组(*Type*, *ID*, *TC*, *Perm*)组成，分别描述如下：

- *Type*：用户或组的标志，用来表明此可信标识项是指定用户还是指定组的；
- *ID*：用户或组的 ID，用来表明此可信标识项所指定的用户或组的 ID 号；
- *TC*：可信标识，用来表明此可信标识项所指定的用户或组对此客体访问时是否需要可信验证系统的验证，如果 *TC*=1，表示需要；如果 *TC*=0，表示不需要。
- *Perm*：存取权限，用来表明此可信标识项所指定的用户或组的对此客体的访问权限。

对于 IPC 对象，即共享内存、消息和信号灯。主体在访问这类动态客体的时候，为了标识此类动态客体，需要改变动态客体的数据结构，以便主体和这类动态客体之间进行双向身份认证。如消息队列的数据结构为：

```
struct msg{…}
```

则可信消息队列的数据结构应为：

```
Struct Trusted_msg
        {
        struct msg M;
        int TC
        }
```

其中，TC 为可信标识，当 *TC*=1，表示为可信消息队列；如果 *TC*=0，表示普通消息队列。

值得注意的是，可信动态客体在被创建，该可信动态客体的属主必须首先在 TPM 中注册，将用户身份信息、证书信息及私钥信息存放在 TPM 内部的 Flash 中，以便对该可信动态客体的身份进行验证。

4.3.5 可信动态客体监管系统的工作流程

主体 1 通过可信动态客体将信息传递给主体 2，为了避免欺骗和中间人攻击，防止信息泄露，需要进行身份认证和访问控制，并保证信息传输的机密性。具体包括两个过程：一是主体 1 将信息传递给可信动态客体，二是可信动态客体将信息传递给主体 2。

对于主体 1 将信息传递给可信动态客体，具体过程如下：

(1)主体 1 发出对可信动态客体的访问请求。

(2)根据动态客体的可信标识，判断动态客体的类型。如果 *TC*=1，表明该动态客体是可信动态客体。转移到下一步；如果 *TC*=0，则该动态客体是普通动态客体。转移到(6)。

(3)通过 TPM_AIM 在 TPM 查询该动态客体属主，以验证该可信动态客体的身份。如果有，则表明该动态客体可信。如果没有，则退出访问。

(4)通过 TPM_AIM 在 TPM 查询主体的信息，以验证该可信动态客体的身份。如果有，则表明该主体可以访问该可信动态客体，如果没有，则退出访问。

(6)可信动态客体接受主体传输的消息。

(5)如果需要保密，则可调用安全模块，对可信动态客体的信息加密。

对于可信动态客体将信息传递给主体 2，其具体过程与前一过程基本相同，就不再赘述。

5 安全和性能性分析

(1)自身的安全性。用户信息、密钥或特征码正这些在鉴别过程中需要的重要信息存储在 TPM 中，防止了非法用户读取敏感信息。

(2)可信动态客体监管系统在主体和动态客体之间增加了双向认证，可以有效防止黑客进行欺骗和中间人攻击，减少了操作系统的脆弱性。另外，可信动态客体监管系统还可在主体和动态客体之间进行传递的信息进行加密和签名，保证信息的机密性和完整性。

(3)系统的工作效率。鉴于丰富的计算资源和较高的传输速率，MSTDOBT 的主要开销在于 TPM 内部的查询比较时间以及和 TPM 的通信时间。选用 SLE66 型号的嵌入式安全模块作为 TPM(通信速率为 38.4k/s)来实现 MSTDOBT 来计算时间花销(包含通信和查询比较时间)。由于查询比较过程在 SLE66 型号的嵌入式安全芯片内部，具有较高的安全性，并且查询比较一次的时间开销仅为几十毫秒，主体 1 通过可信动态客体将信息传递给主体 2 一共需要在 TPM 中进行 4 次查询比较，因此，一次信息传输所花的查询比较时间不会超过 400ms。另外，在 MSTDOBT 过程中，由于传输的数据量少，通信开销小，通信时间可以忽略，因此 MSTDOBT 过程所需要的时间开销不会超过 0.5ms。

总结 随着可信计算技术的兴起，可信操作系统逐渐成为人们关注的焦点。在可信操作系统中如何保证动态客体可信成为了一个紧迫的重要课题。本文首先分析了操作系统中客体的类型，提出了可信动态客体的概念，并分析了其特点。为了防止动态客体泄露信息，提出了基于 TPM 的可信动态客体监管系统。一方面，该系统要求主体必须在 TPM 中注册，确保主体身份合法性，才能利用可信动态客体进行信息传递的；另一方面，在 TPM 中必须存储创建该可信动态客体的属主的身份信息，以确保可信动态客体身份的合法性。最后进行了安全和性能分析，分析表明该可信动态客体监管系统

(下转封三)

对于加法运算，首先判断两个操作数的符号位，若为异号，加法转为减法运算；若为同号，用 new 申请一个最大数空间+2 的空间存放结果。为了满足基本类型中从个位开始相加的方法，首先将两个超长整型中的 vlstr 中的字符串转置后，从字符串的第一位开始相加，最后又将结果转置还原。比如对于运算“12345”+“456”，首先转置为“54321”+“654”，再从字符串的第一位相加，得到结果“10821”，最后转置还原为“12801”。

对于超长整型的乘法运算，利用一位数相乘的内部函数 multdigit(int) 和移位相结合，并累计求和。用 Visual C++ 编写如下代码：

```
friend verylong operator * (const verylong& m,const verylong& n)
 {verylong pprod("1"),tempsum("0");
 for(int j=0;j<n.vlen;j++)
 int digit=n.vlstr[j]-'0';
 pprod=m.multdigit(digit);
 pprod=pprod.mult10(j);
 tempsum+=pprod; }
 tempsum.vlsign=m.vlsign^n.vlsign;
 return tempsum;}
```

其思想是：首先将 n 的数据转置后，循环地从 n 左边第一位开始，取出每位数字 digit，乘以 m 后结果存于 pprod 中，再将 pprod 扩大 j 个 10 倍后累计求和存放于 tempsum。循环结束后，异或计算结果的符号位，返回结果。

其它函数的实现过程与上面介绍的加法、减法和乘法类似。为保证该程序的完整性，编写如下主程序：

```
#include "iostream.h"
#include "verylong.h"
void main()
{verylong x ,y;
cout<<"input two verylong datas:"<<endl;
cin>>x>>y;
cout<<"x+y 的和是"<<x+y<<endl;
cout<<"x*y 的积是"<<x*y<<endl;}
```

主函数 main 中定义两个 verylong 变量 x 和 y；从键盘上任意输入两个数，执行加法、乘法的结果如图 2 所示。

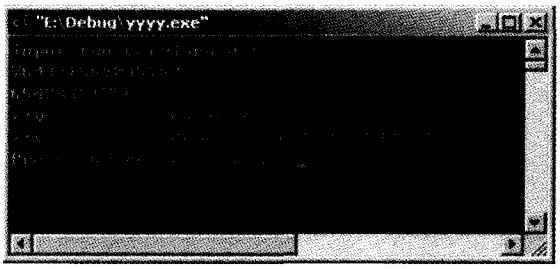

图 2　超长整数运算结果

结论　对 C++ 中超长整型 verylong 进行数据抽象和扩展，具有重要的现实意义，它的应用主要体现在：(1)计算机在处理长整型数据时可以表示出任何范围数据，是对 C++ 数据类型的扩展。(2)计算机处理数学问题的计算代数中，已明确需要该种数据类型的使用，并在 MAPLE 等数学软件[3,4] 中采用了该类型。(3)该数据类可继续推广为有理数类、复数类、多项式类等基础类型。这也正是为什么不构造一个实数的类型，而要构造整型数据类型的真正原因。

参 考 文 献

1　冯玉林，黄涛，倪斌. 对象技术导论[M]. 北京：科学出版社，1998
2　Pohl I. Objected-Oriented Programming UsingC++ [M]. 北京：电子工业出版社，2004
3　何青. 计算代数[M]. 北京：北京师范大学出版社，1997
4　Lkiviadis A, Akritas G. Elements of Computer Algebra with Applications [M]. Newyork: WILEY, 1996

(上接第 284 页)

可以阻止黑客利用动态客体进行欺骗和中间人攻击，防止信息泄露，为进一步建立可信计算环境提供了基础。我们的下一步工作是将 MSTDOBT 在 Linux 系统上实现。

参 考 文 献

1　Jajodia S, Samarati P, Subrahmanian V, Bertino E. A u nified framework for a nforcing m ultiple a ccess c ontrol policies. InS IGMOD'97, Tucson, AZ, May 1997. 474～485
2　Galiasso P, Hale O B J, Shenoi S, Ferraiolo V C, Hu V C. Policy Mediation for Multi-Enterprise Environments. ACSAC, 2000. 100～106
3　Abrams M, LaPadula L, Eggers K, Olson I. A Generalized Framework for Access Control: an Informal Description. In: Proceedings of the 13th National Computer Security Conference, Oct. 1990. 134～143
4　Bertino E, Jajodia S, Samarati P. Supporting Multiple Access Control Policies in Database Systems. In: IEEE Symposium on Security and Privacy, Oakland, 1996
5　Osborn S, Sandhu R, Munawer Q. Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies, ACM Transactions on Information and System Security, 2001,3(2):. 85～105
6　Secure Computing Corporation. DTOS Lessons Learned Report: [Technical Report DTOS CDRL A008]. Secure Computing Corporation, Secure Computing Corporation, 2675 Long Lake Road, Roseville, Minnesota 55113-2536 June 1997
7　Bell D E, Padula L J L. Secure Computer Systems: A Mathematical Model: [MTR 2547-II]. (AD 771 543), The MITRE Corporation, Bedford, Massachusetts, May 1973
8　Harrison M H, Ruzzo W L, Unman J D. Protection in operating systems. Communications of the ACM, 1976,19(8):461～471
9　Biba K J. Integrity considerations for secure computer systems: [Technical Report MTR 3153]. The Mitre Corporation, April 1977
10　Denning D E. A lattice model of secure information flow. Commun. ACM, 1976,19(5):236～242
11　Brewer D, Nash M. The Chinese Wall security policy. In: Proceedings of the 1989 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, May 1989. 206～214
12　Boebert W E, Kain R Y. A Pratical Alternative to Hierarchical Integrity Policies. In: Proceedings of 8th National Computing Security Conference, Gaithersburg, October 1985
13　侯方勇，王志英. 可信计算研究[J]. 计算机应用研究，2004(12)：1～4
14　刘鹏，刘欣. 可信计算概论[J]. 信息安全与通信保密，2003(7)：17～19
15　周明辉，梅宏. 可信计算初探[J]. 计算机科学，2004，31(7)：5～8
16　屈延文. 软件行为学[M]. 北京：电子工业出版社，2005
17　林闯，彭雪梅. 可信网络研究[J]. 计算机学报，2005，28(25)：751～758
18　谭良，周明天. CRL 分段-过量发布新模型[J]. 电子学报，2005，33(2)：227～230
19　谭良，周明天. CRL 增量-过量发布新模型[J]. 计算机科学，2005,32(4):133～136
20　Shan Zhiyony. Research on the Framework for Multi-Policies and Practice in Secure Operating System [D]. Institute of Software Chinese Academy of Sciences Beijing, P. R. China, 2002
21　屈延文. 软件行为学[M]. 北京：电子工业出版社，2005
22　胡建伟，汤建龙，杨绍全. 网络对抗原理[M]. 西安电子科技大学出版社，2004