基于流加密的CPU私化的研究

邓程方,王 铮
(重庆大学 计算机学院 计算机软件与理论系,重庆 400044)

摘 要:CPU是计算机的核心部分,它的主要功能就是解释并执行计算机指令以及处理计算机中的数据。运行在计算机中的所有应用程序都是由CPU可以处理的机器指令构成。为了提高计算机的安全性,提出了通过CPU的私化来阻止一些常见的攻击。CPU私化是通过加密指令和数据,使其对外部以密文存在,来达到私化的目的。处于安全性和效率的考虑,所有加密均采用流加密。此外,对数据的认证,保护程序完整性也是防止恶意篡改的有效手段。文中结合加密和认证的方式来实现CPU的私化,提高计算机的安全性。

关键词:CPU;流加密;私化;MAC;Hash认证
中图分类号:TP309.7 **文献标识码**:A **文章编号**:1673-629X(2011)05-0134-05

Research of Privatization of CPU Based on Stream Cipher

DENG Cheng-fang, WANG Zheng
(Dept. of Computer Software and Theory, Institute of Computer, Chongqing University, Chongqing 400044, China)

Abstract: *The central processing unit (CPU) is the core portion of a computer system that carries out the instructions of a computer program, and is the primary element carrying out the computer's functions. All programs that is executed by computer consist of the machine code that can be recognized by CPU. In order to enhance the security of computer, proposes a method to deter some common attack through the privatization of CPU. The privatization of CPU is to cipher the instructioin and data, store only encypted data in external memory out of CPU. In consideration of security and performence, uses stream cipher for encryption. Moreover, also proposes the authentication of data to deter the malicious tamper of adversary, and provide the protection of integerity of programs. Make a research on the privatization of CPU through encyption and authention, and enhance the security of computer.*

Key words: *CPU; stream cipher; privatization; MAC; Hash authentication*

0 引 言

从计算机诞生那天起,冯诺依曼体系结构占据着主导地位,CPU则是冯诺依曼体系结构计算机的核心部分,完成计算机指令和处理计算机数据。长期以来,计算机的安全任务都交给了操作系统,操作系统控制着整个计算机,阻击病毒入侵和恶意攻击。然而多年过去了,操作系统越来越大,安全策略也越来越复杂,但病毒入侵和恶意攻击却从来没有停止过。越来越多的计算机通过硬件来实现计算机的安全,比如对内存的控制、对总线的控制等。这些实现无非是在操作系统不能控制的地方,通过硬件来提供更底层的保护。CPU作为整个计算机的核心,实际控制着计算机的运行,对CPU的保护可能是对计算机保护的最后一道防线。这里通过CPU仅执行授权的可信任的应用程序,拒绝任何未授权的恶意程序来达到CPU私化的目的。

收稿日期:2010-08-04;**修回日期**:2010-11-05
作者简介:邓程方(1985-),男,硕士研究生,研究方向为分布式实时嵌入系统;王 铮,硕士,副教授,研究方向为网络安全、嵌入式实时操作系统。

1 CPU私化的发展

随着计算机的发展,嵌入式计算机已经进入到人们生活的各个方面。这些嵌入式计算机往往由于内存小,处理器能力低,无法安装安全复杂的操作系统。简单的运用可能对其安全性要求不高,但当被用于军事、金融事务、移动通讯等,就要求计算机的可靠性了。安全的计算机要能阻挡恶意攻击,防止对计算机的任何破坏;要保护计算机内数据的安全,不被窃取(如用户的资料,应用程序的知识产权等)。

CPU的私化发展了很多年了,早在30年前,Best[1]首次提出了总线加密来保护计算机的安全,Best假定CPU是安全的,CPU内访问的所有数据和地址都是明文的,但在SoC外部全都是以密文存储。Takahashi[2]提出了在微处理器的芯片上嵌入一个安全的DMA控制器。DMA控制器管理所有CPU外部请求,同时在外部与内部存储之间传输数据时,使用加密解密引擎。Dallas半导体公司生产的DS5240和DS5250

集成了分组加密引擎来加密处理器和内存之间的总线,加密引擎使用了 DES 或 3-DES 加密[3]。XOM 工程提出了仅可执行的内存的硬件实现,将内存被分为不同的部分来保护安全的过程,使用对称加密(如 AES)对硬件总线加密[4]。

要实现 CPU 私化,最直接的办法就是为 CPU 指定一种特定 ISA(instruction set architecture),使其编译的程序只能在特定 ISA 硬件上运行[5]。在前面提到的 CPU 私化的具体实现中,可以看出它们都是通过加密的方式来实现数据的保密性。所以,仅靠特定 ISA 不能保证安全性的需求,但可以依靠加密来提升处理器的安全性[6]。在 CPU 与外部交换数据的通路上设置加密解密模块,如 CPU 与高速缓存之间,缓存与存储控制器之间等。出于对性能的考虑,文中将采用后者来实现 CPU 的私化。

2　CPU 私化的理论基础

2.1　流加密的原理

流加密(Stream Cipher)就是用算法和初始密钥一起产生一个随机码流,再和数据流异或(XOR)一起产生加密后的数据流的过程,图 1 就是流加密体制模型[7]。解密时只要产生同样的随机码流就可以解密数

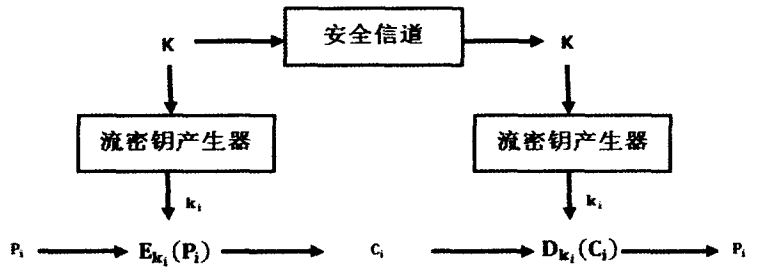

图 1　流加密体制模型

据了。数据流加密定义如下:设 Key 为加密的密钥空间,那么序列 $k_1, k_2, \cdots \in$ Key 被称为密钥流序列。密钥流既可以随机选择或由密钥流产生器生成。生成密钥流需要一个初始输入密钥 K,被称为种子(Seed)。密钥流产生器生成的都是伪随机序列,也就是说经过一个周期会出现重复生成。理论上讲,如果这个周期足够大,就是说一个周期的密钥流足够的长,当大于加密明文的长度时,可以认为是一次一密(One-Time Pad),一次一密理论上是不可破译的[8]。所以,流密钥加密安全性高,实现简单,速度快,得到了广泛的应用,常用的流加密有 RC4,A5 等。

2.2　Hash 认证原理

Hash 函数的基本思想就是对任意长度的数据分组,都能生成一个固定长度的消息摘要(message digest)。Hash 函数是单向的,即给定 Hash 值要找到对应的初始值计算不可行。任何对初始值的细微改动都将使 Hash 值发生很大的改变(雪崩性)[9]。正是 Hash 函数的这些性质,使其常被用来产生消息或其他数据块的"指纹",认证消息的真实性[10]。Hash 函数的常见应用算法有 SHA,MD5 等,主要用于数据的完整性检查。文中使用的认证就是通过 Hash 函数来生成消息认证码(MAC)。

2.3　CPU 私化的工作原理

当 CPU 要读取一个数据时,首先会以虚拟地址(Virtual Address)为索引从缓存(Cache)中查找,如果在缓存总找到(称为命中 Cache Hit),就直接读取缓存并送给 CPU 核处理;如果在 Cache 中没有找到该数据(称为未命中 Cache Miss),则由通过存储管理单元(MMU)获得该虚拟地址对应的物理地址(Physical address),CPU 直接从该物理地址的物理内存中读取数据并缓存到 Cache 中,同时将要覆盖的缓存数据写回内存。

在 Best 提出总线加密时,提到的一些规则沿用至今:SoC 是可信任的,加密单元和密钥保存在片上,而且所有的硬件加密单元位于缓存和存储控制器之间。文中对 CPU 的私化同样要求:CPU 是可信任的,CPU 以外都是不可信任的,对 CPU 的逆向工程是困难的,攻击者不能访问缓存数据。

文中所做的 CPU 私化也沿用了 Best 的模型,就是在缓存和存储控制器之间增加一个加密解密认证模块,如图 2 所示。程序和数据在 CPU 的缓存中以明文的形式存储,在 CPU 的外部以密文的形式存储。相应的,如果 CPU 访问缓存未命中,就要从内存中读取数据,就要把内存中的数据解密并通过认证,返回给 CPU 读取和写到缓存中去。缓存

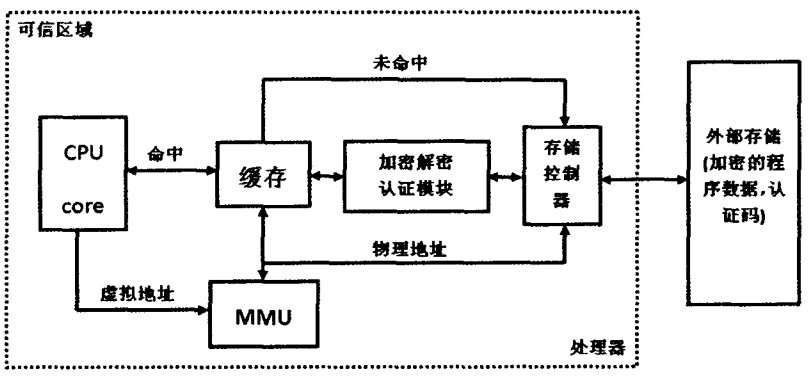

图 2　CPU 私化的原理模型

和内存交换数据的过程就增加了加密解密和认证过程,其整个过程如下:

1)从内存中读取数据块和认证块;

2)解密数据块和认证块,并认证数据块的完整性;

3)执行的过程中修改了缓存的数据块,并且要让出缓存空间时;

4)生成数据块的认证块,并加密数据块和认证块;

5)写回到内存中。

3 CPU 私化的设计实现

3.1 流加密的密钥构造

密钥是 CPU 私化的根本,保护 CPU 的密钥的保护安全性也是很重要的。CPU 的密钥被保存在 CPU 的一个专有密钥寄存器中,并且外接一个锂电池给 CPU 供电,以便密钥长期保存。这样的设计能使任何复杂的电路测试都将破坏密钥。文中在三处都用到了流加密:应用程序的加密,内存到缓存的解密,及缓存到内存的加密。流加密的关键是初始种子的选取,构造加密程序块的密钥流的初始种子由三部分决定:程序块的虚拟地址 VA(i),初始应用程序(未加密和生成 MAC 段的明文应用程序)的 Hash 值 HP,CPU 的密钥 Key。程序块的虚拟地址能确保同一应用程序的所有程序块的流加密的种子不同;应用程序 Hash 值能确保不同应用程序的程序块流加密所需要的产生的种子不相同;CPU 的密钥 Key 是确保不同 CPU 私化的计算机的应用程序块的流加密的种子不同,同时由于 CPU 的密钥 Key 的可靠性是确保 CPU 私化的关键。而最终的流密钥加密的初始种子就是把这三个部分合并在一起经过 Hash 得到的 Hash 值。

3.2 应用程序的加密

一个可执行的应用程序,通常由一个文件头(File Header),代码段(Text Segment),数据段(Data Segment)构成(为了简化,不考虑其他段)[11]。为了实现对应用程序的认证,增加了一个 MAC 段(认证码段)来检查代码段和数据段的完整性。由于 CPU 访问数据不在缓存中时,缓存与内存之间交换数据的最小单位是一个缓存块(Cache Block 或 Cache Line),所以在对应用程序的加密解密和认证时,以缓存块大小为单位。加密应用程序时,加密的每一个块(代码段或数据段中)都在 MAC 段中对应一个 MAC 块,如图 3 所示。在加密前,首先通过 Hash 函数计算数据块的 Hash 值,存放在相应的 MAC 块中,然后计算数据块的流加密的种子,生成密钥流,同时与数据块和 MAC 块异或加密。同样,在应用程序加载被 CPU 执行前,也要对应用程序解密,并通过认证才行。

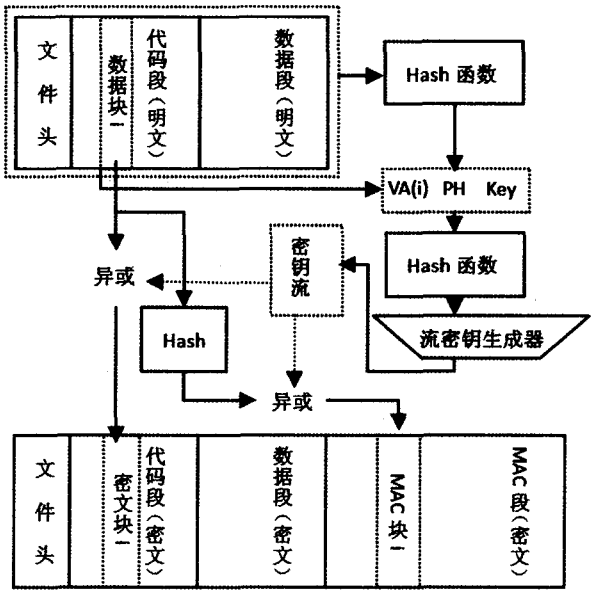

图 3 应用程序加密认证

3.3 缓存与内存的数据交换

CPU 访问数据时,如果没有在缓存中命中,就需要直接从内存中读入数据。而内存中的数据块是密文形式的,这就需要介于缓存和内存之间的加密解密模块来解密数据块,并认证数据的完整性。如图 4 所示加密的程序块 PB(i),和其对应的 MAC 段中的认证块 Mac(i)被同时经由流密钥解密,解密的密钥流由程序块 PB(i)的虚拟地址和 CPU 中的密钥合并作为流加密的种子所产生的密钥流。密文的程序块 PB(i)经解密后,得到明文数据块 Plaintext,接下来就是对程序块的完整性的认证。将明文 Plaintext 与程序块 PB(i)的虚拟地址合并,经过 Hash 函数,获得 Hash 值与 Mac(i)解密的认证码比较,如果相同,说明存放在内存中的程序块未遭到恶意的篡改,将程序块写入缓存块 CB(i)中,并将数据传给 CPU;如果不同,说明程序遭到了恶意的篡改,终止程序的运行。至此,完成了数据由内存到缓存的解密和认证过程。

在向缓存块写入数据的同时,如果要覆盖缓存中先前读入的内存数据,就需要先将缓存块中的数据写回内存再进行写入。数据经由缓存写回内存,这就需要将明文的加密成密文,由于缓存的数据与内存原先的数据相比可能更新了,那么它的认证码也需要更新,这样才能在下次再访问的缓存数据块时,通过数据完整性的认证。所以如图 4 所示,以缓存块 CB(i)的虚拟地址 VAi 和 CPU 密钥 Key 合并作为流密钥的种子,生成密钥流,用于缓存块 CB(i)的加密,生成密文的 PB(i)写入到内存中。同时将缓存块 CB(i)与它的虚拟地址 VAi 合并,经由 Hash 函数,计算消息认证码,在经由前面生成的密钥流加密消息认证码 Mac(i),写入

到内存 MAC 段中。

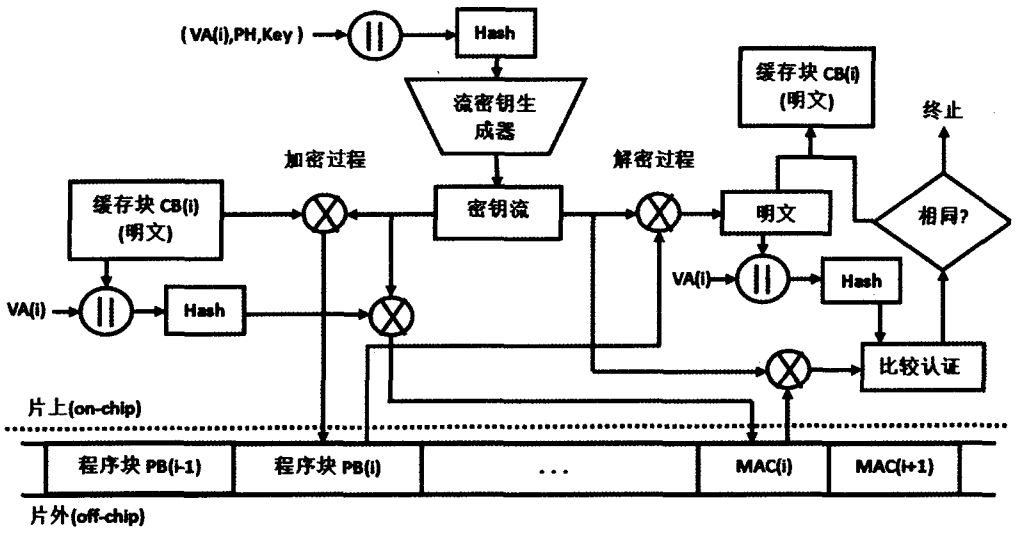

图 4　加密解密与认证模块

4　CPU 私化评估

如前面所说,CPU 私化的目的就是提高计算机的安全性,要能阻挡恶意攻击,防止对计算机的任何破坏;同时保护计算机内数据的安全,不被窃取。本次的设计做到了:

(1)保密性和安全性。

通过对应用程序分块进行流加密来确保应用程序的数据的保密性。首先,由于文中用到的流加密是对程序块进行的,所以为了提高其安全性,对不同的程序块使用不同的种子来生成不同的密钥流。为了确保程序块的流加密的种子的不同,由程序块的虚拟地址,初始程序的 Hash 值和 CPU 私有的密钥 Key 来一起来生成程序块的流加密的种子。这就保证了不同程序块的加密密钥流不一样,使其具有一次一密(One-Time Pad)的安全性。由于最终用于程序块加密的密钥流是由前面所说的三部分的 Hash 值作为种子的生成的,所以即使破解了一个程序块的密钥流及其 Hash 值,也是无法知道 CPU 的密钥的。再则,一开始就假设了 CPU 的密钥是不可被窃取的,确保了整个加密解密及 MAC 认证过程的安全性。

同时,由于要求仅在 CPU 的缓存中以明文形式存储,在 CPU 以外的任何地方都以密文形式存储,所以即使应用程序和数据被外部获得,那也是加密了的程序和数据。不能运行,对其的逆向工程也不可行,并能保护应用程序的知识产权和数据不被窃取。

对于外来病毒和木马程序,由于其未经过 CPU 的密钥加密和 MAC 认证,对 CPU 来说就如同一对无用的数据,经过解密后就成了混乱的指令,由于其不能通过 MAC 认证,所以更不可执行。

(2)防恶意篡改。

攻击者常常通过篡改目标计算机中可信程序,来达到攻击的目的。在没有对程序数据进行认证的情况下,程序就被 CPU 执行是很危险的事。而文中在应用程序的生成时,就附带了 MAC 段,并对 MAC 段也进行了加密。对于直接的恶意篡改,是不能通过 MAC 段认证的;而要想篡改程序的同时伪造 MAC 段也是不可行的,由于 MAC 段和数据段一样都经过了流加密。这样就对应用程序形成了有效的认证,任何对应用程序本身的篡改,都将造成程序通不过认证而终止执行。除了对程序本身的篡改,对内存的篡改也是一样,不能经过 CPU 合法的流加密,同样不能伪造认证码,不能通过认证,最终将被从内存中清除。有了这样的实现,数据的完整性首先得到了保证,把任何篡改应用程序的攻击者拒之门外,也使常见的寄生在应用程序中的病毒无法存活。

(3)减少了性能损失。

在设计加密解密及认证模块时,并没有将其放在缓存和 CPU 之间,就是出于对性能的考虑。在考虑性能的问题上,首先用了流加密,正是由于流加密的实现简单,加密比一般的分组加密的速度快,其次,将数据块和认证块使用相同的密钥流加密解密,使得两者的加密解密过程能同时进行,节约了时间。再次,加密解密及认证模块放在缓存和内存之间,正是由于 CPU 访问内存的延迟较 CPU 访问缓存的延迟高得多,如果在 CPU 访问数据的同时,就通过 MMU 计算出物理地址来对内存中的数据块,进行解密及认证,那么当 CPU 命不中时,要求访问内存时,可以从加密解密及认证模块获得需要访问的值,这样就能尽量减少 CPU 的等待。再加上加密解密及认证模块是通过硬件实现,速度可以得到保证。

(4)可行性。

文中提到的 CPU 的私化相对于未私化的 CPU 仅仅多了一个加密解密认证模块,同时在内存模型中增加了 MAC 段,在执行程序上与未私化 CPU 并无不同。加密解密认证模块都有其硬件实现;流加密实现简单,

很容易做成硬件，如 RC4 就有硬件的实现[12]；认证用的 Hash 函数，就更容易做成硬件了。认证用的 MAC 段是与程序数据段和代码段构造的一种映射关系的附加段，并没有打乱原有的应用程序的结构。所以，文中提到的 CPU 私化是在不改变原有的体系结构的基础上，在其中加入加密解密认证模块来实现的，是可行的。

当然这样的设计也不能解决所有的安全问题，这跟 CPU 私化的目的有关，CPU 私化的目的就是在计算机内部构件一个硬件或软件的安全模型，保证计算机内部的安全，而非与外界隔离。计算机内部的所有合法程序都认为是可信的，外来程序都是非法的，都不能获得 CPU 的执行。这就决定了，如果内部程序设计缺陷同样可能招致一些攻击（如溢出攻击等）。而在计算机对外部提供服务时，也同样不能阻止来自外部的 DOS 攻击。此外，内存模型中增加 MAC 段，同样造成了一定程度的内存消耗，但换来的是数据的安全性，是值得的。

5 结束语

提出使用流加密来实现 CPU 的私化，在计算机内部构建一个安全可信的机制，为 CPU 提供一个私有的可信的运行环境。这样的 CPU 私化能防范恶意的逆向工程，病毒程序的恶意攻击；同时，通过 Hash 认证来阻止攻击者对应用程序的恶意修改，保证了数据的完整性。随着计算机对安全性的要求越来越高，特别是在嵌入式在一些关键领域的大量运用，CPU 的私化必将有更长远的发展。

参考文献：

[1] Best R M. Preventing Software Privacy with Crypto-Microprocessors[C] // IEEE Spring COMPCON 80. San Francisco, California:[s. n.],1980:466-469.

[2] Takahashi, Heer D N. Secure memory management unit for microprocessor[P]. U. S. Patent. US5825878,1998.

[3] Kuhn M G. Cipher Instruction Search Attack on the Bus-Encryption Security Microcontroller DS5002FP[J]. IEEE Transaction on Computer,1998,47:1153-1157.

[4] Lie D, Thekkath C, Mitchell M, et al. Architectural support for copy and tamper resistant software[C] // International Conference on Architectural Support for Programming Languages and Operating Systems. [s. l.]:[s. n.],2000:168-177.

[5] 夏 冰，白永红，郑秋生. 一种基于嵌入式系统的安全处理器研究与实现[J]. 中原工学院学报，2009(4):20-22.

[6] 韩 林，韩 军，曾晓洋. 一种专用指令集安全处理器的架构设计与实现[J]. 小型微型计算机系统，2009(4):746-751.

[7] 段云所，魏仕民，唐礼勇. 信息安全概论[M]. 北京：高等教育出版社，2003.

[8] 邢书宝，李 刚，薛惠锋. 一次一密加密系统设计与实现[J]. 计算机技术与发展，2007,17(3):150-152.

[9] Schneier B. 应用密码学[M]. 吴世忠，等译. 北京：机械工业出版社，2000.

[10] 肖皇培，张国基. 基于 Hash 函数的报文鉴别方法[J]. 计算机工程，2007(6):101-103.

[11] 俞甲子，石 凡，潘爱民. 程序员的自我修养：链接、装载与库[M]. 北京：电子工业出版社，2009.

[12] 杨 梅，张耀文. RC4 流密码原理与硬件实现[J]. 信息通信，2009(6):40-43.

（上接第 130 页）

并利用虚粒子对障碍物进行模拟。然而实际中的水流运动是很复杂的，特别是存在障碍物的情况。如何使复杂水流动画的模拟更加逼真，是未来研究的主要方向。

参考文献：

[1] 湛永松，杨明浩，刘文兆，等. 基于 SPH 的中国水墨画风格烟气动画系统[J]. 系统仿真学报，2009,21(18):5770-5774.

[2] 李大鸣，陈海舟，张建伟，等. 基于 SPH 法的二维矩形液舱晃荡研究[J]. 计算力学学报，2010,37(2):369-374.

[3] 李同飞，刘瑛琦，张晨明，等. 基于光滑质点流体动力学方法数值波浪水槽研究[J]. 海洋工程，2009,127(11):96-100.

[4] 张锁春. 光滑质点流体动力学(SPH)方法(综述)[J]. 计算物理，1996,13(4):385-397.

[5] Liu G R, Liu M B. Smoothed Particle Hydrodynamic a meshfree particle method[M]. [s. l.]:World Scientific Publishing Co. Pte. Ltd. ,2003.

[6] Liu M B, Liu G R, Lam K Y. Numerical simulation of incompressible flows by SPH[C]//International Conference on Scientific & Engineering Computing. Beijing:[s. n.],2001.

[7] Liu M B, Liu G R, Lam K Y. Investigations into water mitigations using a meshless Particle method[J]. Shock Waves, 2002,12(3):181-195.

[8] 陈前华，邓建松，陈发来. 滴水涟漪的计算机动画模拟[J]. 计算机研究与发展，2001,38(5):524-528.

[9] 李付鹏，汪继文. 一种复合型数值方法的改进与算法实现[J]. 微机发展，2003,13(12):122-124.

[10] 张姝慧，汪继文. 求解浅水波方程的光滑粒子流体动力学法[D]. 合肥：安徽大学计算机科学与技术学院，2007.

[11] Liu G R, LiuM B. 光滑粒子流体动力学——一种无网格粒子法[M]. 韩 旭，杨 刚，强洪夫译. 长沙：湖南大学出版社，2005.

[12] 张 飞，汪继文. 一维溃坝问题的间断 Galerkin 方法[J]. 安徽大学学报(自然科学版)，2005,29(2):5-8.