软件保护新技术

向广利　朱平　钟忺　鲁晓成　著

WUHAN UNIVERSITY PRESS
武汉大学出版社

软件保护新技术

向广利　朱平　钟忺　鲁晓成　著

WUHAN UNIVERSITY PRESS
武汉大学出版社

图书在版编目(CIP)数据

软件保护新技术/向广利,朱平,钟忺,鲁晓成著. —武汉:武汉大学出版社,2012.9

ISBN 978-7-307-10001-5

Ⅰ.软… Ⅱ.①向… ②朱… ③钟… ④鲁… Ⅲ.软件—安全技术—高等学校—教材 Ⅳ.TP311.56

中国版本图书馆 CIP 数据核字(2012)第 162618 号

责任编辑:林 莉 责任校对:黄添生 版式设计:支 笛

出版发行:**武汉大学出版社** (430072 武昌 珞珈山)

(电子邮件:cbs22@whu.edu.cn 网址:www.wdp.com.cn)

印刷:湖北金海印务有限公司

开本:787×1092 1/16 印张:11.25 字数:282 千字 插页:1

版次:2012 年 9 月第 1 版 2012 年 9 月第 1 次印刷

ISBN 978-7-307-10001-5/TP·438 定价:25.00 元

前　言

软件保护是近几年来信息安全领域的一个新兴研究分支。在软件保护的研究中，需要借鉴计算机安全方面的技术，还会用到计算机科学其他领域的知识：密码学、软件水印、软件混淆、防篡改技术、密钥管理等。计算机软件可以简单地抽象为两个部分：数据和计算。本书主要是讨论软件中的数据保护和计算保护方法。从软件的版权保护角度，考虑到软件分发的途径和软件运行环境的多样性，本书也论述软件的版权保护和适合软件保护的密钥管理体系。

本书是一部关于软件保护理论与实践相结合的著作，全书共6章。第1章对软件保护的现状、应用领域进行综述性的说明；第2章介绍软件保护的基础知识，主要涉及：软件保护中常用的加密算法、HASH算法、签名算法和认证算法等；第3章论述软件中的数据保护方法，主要通过数据混淆来实现软件中的数据保护，提出基于同态加密的数据混淆方法；第4章论述软件中的计算保护方法，提出基于RSA算法和ElGamal算法的同态加密函数计算方法；第5章论述软件版权保护技术，重点介绍软件水印和软件防篡改技术，并对DRM系统做了简要的介绍；第6章论述软件保护中的密钥管理，考虑到软件的分发模式和运行环境的多样性，主要讨论密钥协商、密钥更新与密钥隔离等技术。

特别感谢武汉大学计算机学院的何炎祥教授和武汉理工大学计算机科学与技术学院的钟珞教授在本书撰写过程中的指导和帮助，没有何教授和钟教授的帮助与鼓励，本书很难面世。感谢实验室的徐光兴、黄亚妮、陈智明、姚琴、蔡郑、王海飞、熊信禄、陈雄、张夏、冯天文、张学佳、王智强、林川、黄小龙等同学的帮助。感谢武汉大学出版社参与本书编辑出版的各位同志，尤其要感谢林莉老师对本书出版的帮助。最后，感谢我的家人和朋友们，你们的鼓励与支持是我撰写本书的动力。

由于软件保护是一个新兴的、多学科交叉的研究领域，涉及的范围非常广泛，加之作者自身学识有限，书中错误和疏漏之处在所难免，敬请读者批评指正。

作　者

2012年8月

目 录

第 1 章　软件保护概述

1.1　引言

网络传播技术在给我们带来便利的同时，也带来了大量的版权纠纷，随之而来的版权保护问题也日渐突出。软件盗版、恶意篡改和逆向工程是软件安全面临的主要威胁，不容忽视其带来的经济损失。在《2011 年全球 PC 套装软件盗版研究》报告中被商业软件联盟 BSA 指出，亚太区去年的个人电脑软件盗版率为 60%，也就是说用户安装的每 5 个软件中就有 3 个是盗版的。2011 年，亚太区安装在 PC 上的盗版软件的商业价值创历史之最——比上年增长 12%，近 1,356 亿元。自 2003 年至今，中国 PC 软件盗版率却下降了 15 个百分点，降至 77%，尽管亚太区 PC 套装软件盗版率上升了 7 个百分点，这一进步在某种程度上与中国近年来一系列加强保护知识产权、推进软件正版化的举措密不可分。

在日益严峻的软件版权危机背景下，相应的立法被提上日程，相应法规相继出台，但难免存在立法漏洞并滞后于新生问题，因为计算机的发展历史短、技术更新快，一些技术手段的法律效力也难以界定：例如利用技术手段实施的盗版追踪和侵权取证是否具有法律效力，仍然缺乏明确的法律规定；因为在不同案例中它们所起的作用也许截然相反，逆向工程或反编译是否合法成为争议的焦点。软件的版权保护仅仅依靠法律规范和道德约束显然不够，应从法律规范、道德宣传和技术保护几方面入手，发展软件保护技术刻不容缓。本文介绍了常见的软件保护技术，分析了软件版权保护的进一步发展方向。

1.2　软件保护技术

软件保护是软件安全的一部分，在计算机安全系统中的地位如图 1-1 所示。

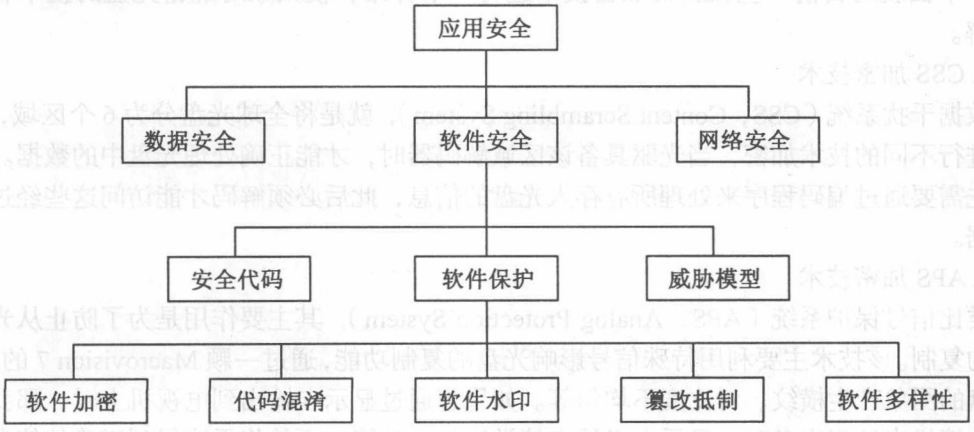

图 1-1

从广义的角度来说，软件保护技术包括计算机软件和系统的安全。如何防止合法用户和其数据被恶意客户端程序所攻击、设计和管理计算机系统来实现一个严密的安全系统，是目前大多数关于计算机的安全研究的重点[1-5]。例如可以限制在本地文件系统写文件操作。类似的技术还有监视客户端程序的软件故障隔离（Software Fault Isolation），这种技术能确保其不能够在它只能在赋予范围内进行写操作，此种技术在.NET 和 JAVA 中采用了。例如在 JAVA 安全模型中，不被信任的代码（例如 APPLET）将被禁止执行一些特定操作，即用户可以使用字节码校验来保证不被信任的客户端程序的类型安全。

从狭义的角度来说，软件保护技术就是在恶意环境下如何保护软件自身的数据和计算不受破坏和剽窃，软件能够在授权范围内正确的使用的相关技术。软件保护技术是一项综合的技术，目前一些供应商利用智能卡芯片本身具有很高的安全性，来误导软件开发商以为采用智能卡芯片的软件保护产品也一定具有同样的安全性，还有一些软件保护产品供应商在没有提供确切的数据和评测报告时，宣称自己的产品是不可破解的，往往是一种营销的策略。其实这些观点都是错误、片面的。软件保护产品不能够单一的由某个方面来以偏概全的断定其安全与否，它涉及从上层的应用软件、操作系统、驱动、硬件、网络等广泛的内容，所以是一个综合的技术范畴。

软件和其他安全产品相比具有一定的特殊性，下面基于硬件和软件分别介绍几种较常见的计算机软件保护技术。

1.2.1 基于硬件的保护方法

认证过程、数据加密、访问控制、唯一的系列号、密钥产生、可靠的数据传输和硬件识别，这些是基于硬件的软件保护策略包含的多种功能，有一些产品也支持许可证策略，这些策略的主要目的是防止硬件被复制。基于硬件的保护可以具有很好的安全性。主要包括以下几种典型方式：

1.2.1.1 加密光盘

越来越多的软件商使用加密方法来保护自己的软件，主要是为了防止盗版软件对软件市场的侵害。目前加密光盘的方法的主要原理是利用特殊的光盘母盘上的某些特征信息是不可再现的，而且这些特征信息在光盘复制时复制不到的地方，大多是光盘上非数据性的内容。下面就对目前一些较新的加密技术进行一下介绍，使大家对加密光盘的技术有一定的了解。

1. CSS 加密技术

数据干扰系统（CSS，Content Scrambling System），就是将全球光盘分为 6 个区域，并对各区进行不同的技术加密，当光驱具备该区域解码器时，才能正确处理光盘中的数据。该技术首先需要通过编码程序来处理所有存入光盘的信息，此后必须解码才能访问这些经过编码的数据。

2. APS 加密技术

类比信号保护系统（APS，Analog Protection System），其主要作用是为了防止从光盘到光盘的复制。该技术主要利用特殊信号影响光盘的复制功能，通过一颗 Macrovision 7 的芯片，使光盘的图象产生横纹、对比度不均匀等。如果想通过显示卡输出到电视机上时，那我们在使用计算机来访问光盘时，显示卡必须支持类比加密功能，不然将无法得到正确的信息，也就无法在电视上看到光盘影片的精美画面。

3. 光盘狗技术

一般的光盘加密技术实施起来费用高不说，而且花费的时间也不少，因为需要制作特殊的母盘，进而改动母盘机。我们若要自由选择光盘厂来压制光盘，可以采用光盘狗技术，光盘狗是专门加密光盘软件的优秀方案，并且通过了中国软件评测中心的加密性能和兼容性的测试。它不在母盘制造上动手脚，能通过识别光盘上的特征来区分是原版盘还是盗版盘。

该特征是在光盘压制生产时自然产生的，不同的母盘压制出的光盘即便盘上内容完全一样，盘上的特征也不一样，同一张母盘压出的光盘特征相同，这就使得盗版者翻制光盘过程中无法提取和复制的。

4. 外壳加密技术

“外壳”就是给可执行的文件加上一个外壳。这个外壳程序负责把控制权交还给解开后的真正的程序，将用户原来的程序在内存中解开压缩。用户根本不知道也不需要知道其运行过程，并且对执行速度没有什么影响，因为一切工作都是在内存中运行，用户最终执行的实际上是这个外壳的程序。在外壳程序中加入对软件锁或钥匙盘的验证部分就是我们所说的外壳加密。其实外壳加密的作用还很多，网络上许多程序是专门为加壳而设计的，如果你的程序不希望被人跟踪调试，如果你的算法程序不想被别人静态分析，如果你不希望你的程序代码被黑客修改，外壳程序就是很好的选择。它的主要特点在于保护你的程序数据的完整性，它对程序进行压缩或根本不压缩，反跟踪，加密代码和数据。

5. CGMS 技术

CGMS 技术也叫内容拷贝管理技术，它主要是通过生成管理系统对数字拷贝进行控制，它是通过存储于每一光盘上的有关信息来实现的，是用来防止光盘的非法拷贝的。CGMS 这一“串行”拷贝，生成管理系统既可阻止对其子版软件进行再拷贝，也可阻止母版软件进行拷贝。制作拷贝的设备必须遵守有关规则，即使是在被允许正常拷贝的情况下。为了使数字录音机能很方便地予以识别，数字拷贝信息可以经编码后送入视频信号。

6. DCPS 技术

数字拷贝保护系统技术，它的作用是让各部件之间进行数字连接，但不允许进行数字拷贝。以数字方式连接的设备，如 DVD 播放机和数字电视或数字录像机，就可以通过此技术交换鉴证密钥建立安全的通道。为了防止那些未鉴证的已连接设备窃取信号，DVD 播放机加密已编码的音频/视频信号，发给接收设备，接收设备接着进行解密。无需拷贝保护的内容则不进行加密。含有更新的密钥和列表（用来识别非认证设备）的新设备和新内容（如新的盘片或广播节目）也可获得安全特性。

7. CPPM 技术

预录媒介内容保护技术，该技术一般用于 DVD-Audio。它通过在盘片的导入区放置密钥来对光盘进行加密，但在 sector header 中没有 title 密钥，盘片密钥由“album identifier”取代，它取代了 CSS 加密技术。现有设备无需任何改动，因为该技术的鉴定方案与 CSS 相同。

8. CPRM 技术

录制媒介内容保护技术，它将媒介与录制相联系。即在每张空白的可录写光盘上的 BCA 处放置 64 比特盘片 ID，如果受保护的内容被刻录到盘片上时，它通过 ID 得到 56 位密码，然后进行加密。之后从 BCA 中读取盘片 ID，就可以生成盘片内容解密所需要的密钥，进而访问光盘信息。在盘片内容被复制到其他媒介的情况下，盘片 ID 会被丢失或出错，这样光盘数据也将无法得到解密。

现在软件市场上很多工具软件、设计软件、教学软件、杀毒软件、多媒体软件都进行了加密。通过使用这些加密技术，可以对软件的非法拷贝或非法使用造成障碍，软件的市场利益在一定程度上得到了保护。软件解密盗版可以说是不能被任何一种加密软件（硬件）所杜绝，只是解密盗版的难度不同。如果让盗版者在破解被保护的软件时，耗费极大的时间精力，付出巨大的代价，最终被迫放弃攻击，这就是一种好的加密效果。游戏软件经常采用的方式是，将被保护软件的部分密钥放在可移动的软盘或光盘当中，保护软件只有在软盘或光盘存在的时候才可以运行。这种方式的基本原理是在光盘的光轨上隐藏一个密钥，如 Macrovision SafeDisk 工具，而一般的光盘刻录机无法复制此密钥，这样光盘就无法复制，软盘也使用类似的技术。

存在的问题：用户在原盘被划坏或者毁坏的情况下就无法继续使用软件，并且黑客只需分析或跟踪找到判断代码处，修改可执行文件就能跳过此段代码，达到破解的目的。此种保护技术的安全性并不是很高，像 Elaborate Bytes 公司的 CloneCD 和 Padus 公司的 DiscJuggler 等拷贝工具，他们这种工作在原始模式（RAW MODE）的光盘拷贝程序可以原样拷贝加密光盘，但是有一些软件开发商考虑到价格优势，还是使用此种技术来保护自己的软件。

1.2.1.2 加密狗

加密狗又被称为加密锁，是一种智能性加密产品。它是一个硬件电路，可以安装在并口、串口或 USB 接口上的。加密狗的工作原理如图 1-2 所示。

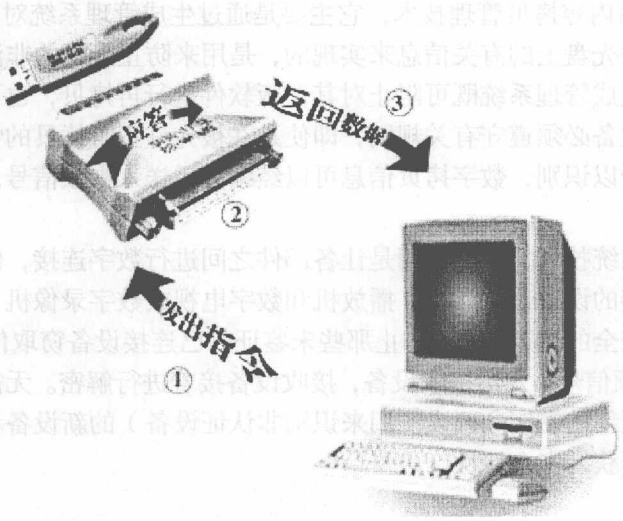

图 1-2

虽然在基于软件的保护方式和安全性上它具有更高的安全性，但是在易用性上存在一定问题，因为用户在使用被保护软件时，不得不在自己的机器上安装相应的保护硬件和驱动程序，易用性上存在一定问题，并且价格比基于软件的保护方式高。

加密狗的工作原理：

通过在软件执行过程中和加密狗交换数据，加密狗实现了加密。加密狗具有分析、判断及主动的反解密能力，因为加密狗内置单片机电路(也称 CPU)，“智能型”加密狗也由此得名。加密狗硬件不能被复制，因为专用于加密的算法软件内置于加密狗，该软件被写入单片机后，

就不能再被读出。并且加密算法是不可逆、不可预知的。一个数字或字符通过加密算法可以变成一个整数，如 DogConvert(A)=43565、DogConvert(1)=12345。我们通过下面的例子说明单片机算法的使用。如程序中有如下语句：A=Fx(3)。变量 A 的值需要通过常量 3 来得到。于是，原程序可以这样被改写：A=Fx(DogConvert(1)-12342)。DogConvert(1)-12342 就取代了原程序中的常量 3。只有软件编写者才知道实际调用的常量是 3。要是没有加密狗，算式 A=Fx(DogConvert(1)-12342)的结果也肯定不会正确，因为 DogConvert 函数就不能返回正确结果。这种加密方式使盗版用户得不到软件使用价值，相比一发现非法使用就警告、中止的加密方式，这种方式更温和、更隐蔽、更令解密者难以破解。

另外加密狗还有可以用作对加密狗内部的存储器的读写的读写函数。我们可以把上算式中的 12342 也写到加密狗的存储器中，这样 A 的值完全取决于 DogRead()和 DogConvert()函数的结果，令解密难上加难。由于解密者在触及加密狗的算法之前要面对许多难关，所以加密狗单片机的算法难度要低于一些公开的加密算法，如 DES 等。

1.2.1.3　基于硬件序列号加密

由于计算机软件是一种特殊的产品，为了保护软件开发商的利益，我们必须对软件进行加密保护，以此防止软件的非法复制、盗版。根据微机硬件参数给出该软件的序列号是采用基于硬盘号和 CPU 序列号的软件加密技术；软件提供商或开发商在接收到用户提供的序列号后，利用注册机（软件）产生该软件的注册号寄给用户即可。不同于以前的序列号的注册方法，它的注册信息与机器的硬件信息有关，进一步提高了软件的安全性[6-11]。

1. 硬盘号和 CPU 序列号

（1）硬盘的序列号分为逻辑序列号和物理序列号。物理序列号是在生产时由生产厂家写入的唯一存在的序列号。它是一个与操作系统无关的特性，不随硬盘的分区、格式化状态而改变，存在于硬盘的控制芯片内,像硬盘的扇区数、物理柱面数一样。用户主机的硬盘序列号不能用常规办法修改，只能用硬盘控制器的 I/O 指令读取。需要注意的是，将硬盘格式化成 FAT 或 FAT 32 后，分区引导扇区自动生成的逻辑序列号和硬盘的序列号有着根本的区别，后者是物理存在的。新的逻辑序列号会在每次格式化磁盘时产生。然而在实际的软件保护中，用户主机识别可通过硬盘的物理序列号和逻辑序列号。

我们可以使用物理序列号作为用户主机唯一性标志的硬件，因为硬盘物理序列号的唯一性和只读性的特点。一般是在软件安装到硬盘时读取该序列号,经过加密算法后生成的注册码保存起来(如写入注册表)。以后，安装到硬盘的软件可以比较安装时保存的注册码和当前的硬盘序列号，在不一致的情况下软件将不能运行，则说明该软件被非法拷贝到其他硬盘。逻辑序列号也时常被用来作为用户机器的标志来分配注册码。此时，我们使用的序列号，一般是逻辑盘符 C 盘的逻辑序列号。因为 C 盘下一般安装着用户的操作系统，用户就不会经常格式化 C 盘而导致序列号发生改变。并且在读取逻辑序列号时，C 盘肯定是存在的，而其他的分区盘符不一定存在。

（2）CPU 序列号可以用来识别每一个处理器，是一个建立在处理器内部的、唯一的、不能被修改的编号。它由 96 位数字组成，低 64 位每个处理器都不同，唯一地代表了该处理器，高 32 位是 CPU ID，用来识别 CPU 类型。Intel 为了适应这一新特征，在处理中增加了一个寄存器位（模式指定寄存器位：Model Specific Register–“MSR”）和两条指令（“读取”和“禁止”）。禁止指令可以禁止对处理器序列号的读取。MSR 位是为了配合 CPU 序列号的读取和禁止而设置。当“MSR”位为“1”时只能读取高 32 位（即 CPU ID），而低 64 位全为零；

当 MSR 位为“0”时可以读取 CPU 序列号。读取指令扩展了 CPUID 读取指令。96 位的处理器序列号可以通过执行读取指令得到。

2. 程序实现

加密方法：

通过应用程序取得 CPU 号和机器硬盘号，然后通过机密程序形成一个注册序列号，软件注册者接收到用户发过来的注册序列号后，按照预定的算法生成注册码，然后将其发给用户，通过注册形成合法用户。软件每次启动时，都到注册表或注册文件的相应位置读取注册码，并与软件生产的注册码比较，一致则是合法用户，否则是非法用户。非法用户即使知道注册序列号与注册码也无法使用，因为注册码具有唯一性，它与用户计算机的硬盘号与 CPU 号相关联。

实现过程：

（1）读取 CPU 号。

只能采用对硬盘控制器直接操作的方式进行读取硬盘的序列号，也就是说只能采用 CPU 的 I/O 指令操作硬盘控制器，采用在 DELPHI 嵌入汇编的方法读取 CPU 号。其读取方法如下：

MOV EAX.01H

如果 EDX 中低 18 位为 1，说明此 CPU 是支持序列号的。就是序列号的高 32 位 EAX。对同一型号的 CPU，32 位是一样的。

再执行：

MOV EAX.03H

此时序列号的第 64 位即 EDX:ECX。

（2）读取硬盘号

通过 CreateFile 函数读取硬盘号，CreateFile 可以打开物理设备和串口等，打开硬盘可以使用 CreateFile(‘\\.\PHYSICALDRIVEI’,)其中的 I 为 0~255，代表需要读取的硬盘。命令为：

hDevice:=CreateFile(‘\\.\PhysicalDrive0’,GENERIC_READ OR GENERIC_WRITE, FILE_SHARE_READ OR FILE_SHARE_WRITE, NIL, OPEN_EXISTING, 0, 0)

对打开的设备进行通信可以使用 DeviceIOControl 函数，发送指定命令，物理序列号和模型号可根据返回的 PSENDCMDOUTPARAMS 结构得到，并将其格式化为一定的格式输出。

（3）对注册表的操作。

可利用 TRegistry 对象在 Delphi 程序中来存取注册表文件中的信息。

创建和释放 TRegistry 对象

创建对象和释放内存可通过 Create 和 Destroy 来实现。

读取注册表中写入信息

读取注册表数据时，可采用函数 ReadString、ReadInteger、ReadBinaryData 来读取字符串、数值、二进制值。

向注册表中写入信息

信息通过 Write 系列方法转化为指定的类型，并写入注册表。

向注册表写入数据时，可采用 WriteString、WriteInteger、WriteBinaryData 等函数来写入字符串、数值、二进制值。

1.2.1.4 加密 CPU

CPU 的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，它是计算机的核心部分。通过 CPU 的私化来阻止一些常见的攻击可以提高计算机的安全性。CPU 私化是通过加密指令和数据，使其对外部以密文存在。所有加密均采用流加密，以保证安全性和效率。此外，防止恶意篡改的有效手段还包括对数据的认证，保护程序完整性。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。

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

1. CPU 私化的发展

嵌入式计算机伴随着计算机的发展已经进入到人们生活的各个方面。由于内存小，处理器能力低，这些嵌入式计算机往往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，但当被用于移动通信、金融事务、军事等领域时，就要求计算机的可靠性了。安全的计算机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），要能防止对计算机的任何破坏，阻挡恶意攻击。

CPU 的私化发展已经有很多年了，早在 30 年前，Best 首次提出了总线加密来保护计算机的安全，他假定 CPU 是安全的，CPU 内访问的所有数据和地址都是明文的，在 SoC 外部全都是以密文存储。

Taka-hashi 提出了在微处理器的芯片上嵌入一个安全的 DMA 控制器。所有 CPU 外部请求由 DMA 控制器管理，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，加密引擎使用了 DES 或 3—DES 加密，处理器和内存之间的总线通过加密引擎来加密。

XOM 工程提出了仅可执行的内存的硬件实现，将内存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。

实现 CPU 私化最直接的办法是，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。

由上可知，在具体实现时，CPU 的私化都是通过加密的方式来实现数据的保密性。因此，单独依靠特定 ISA 不可以保障安全性，但处理器的安全性可以依靠加密来提升。在 CPU 与外部交换数据的通路上如 CPU 与高速缓存之间，缓存与存储控制器之间等设置加密解密模块。考虑到性能问题，CPU 的私化将采用后者来实现。

2. CPU 私化的理论基础

（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，是流加密体制模型。如图 1-3 所示加密时只要产生同样的随机码流就可以解密数据了。

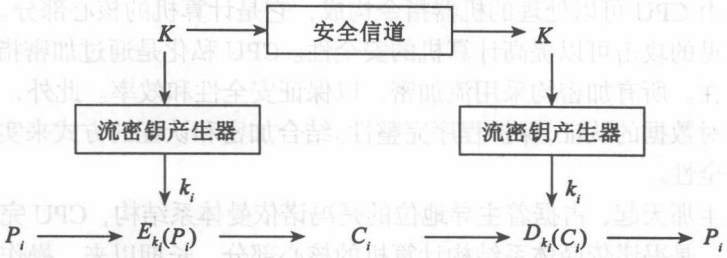

图 1-3 流加密模型

数据流加密定义如下:设 Key 为加密的密钥空间，那么序列 k1，k2，…∈ 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。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）Hash 认证原理。

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），这是 Hash 函数的基本思想。Hash 函数是单向的，在给定 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）。正是这些性质，使得它常被用来认证消息的真实性，因为它可以产生消息或其他数据块的“指纹”。SHA，MD5 等是 Hash 函数的常见应用算法，主要用于数据的完整性检查。

（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，如果在缓存中找到（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 1-4 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。

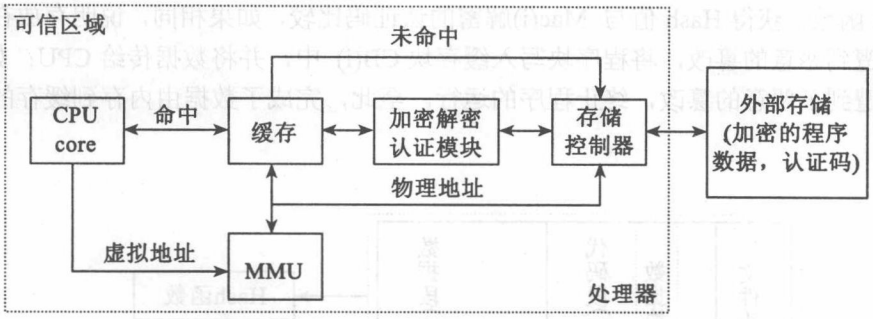

图 1-4　CPU 私化的原理模型

3．CPU 私化的设计实现

（1）流加密的密钥构造。

保护 CPU 的密钥的安全性也是很重要的，因为密钥是 CPU 私化的根本。CPU 的一个专有密钥寄存器中保存着 CPU 的密钥，外部连接着一个锂电池，随时给 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。主要用到了流加密的地方有: 应用程序的加密、内存到缓存的解密、缓存到内存的加密。初始种子的选取是流加密的关键，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。程序块的虚拟地址来确保同一应用程序的所有程序块的流加密的种子的不同; 靠应用程序 Hash 值来确保不同应用程序的程序块流加密所需要的产生的种子的不同性; CPU 的密钥 Key 来确保不同 CPU 私化的计算机的应用程序块的流加密的种子的不同性，同时由于 CPU 的密钥 Key 的可靠性是确保 CPU 私化的关键。最终这三个部分的流密钥加密的初始种子就是把各部分合并在一起经过 Hash 得到的 Hash 值就是。

（2）应用程序的加密。

通常由一个文件头(FileHeader) ，代码段 (Text Segment) ，数据段 (Data Seg-ment) 构成(为了简化，不考虑其他段)了一个可执行的应用程序。增加的一个 MAC 段(认证码段)来检查代码段和数据段的完整性是为了实现对应用程序的认证。

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) ，因此在对应用程序加解密和认证时，以缓存块大小为单位。如图 1-5 所示，加密的每一个块(代码段或数据段中) 在加密应用程序时都在MAC段中对应一个 MAC 块。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。

（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 1- 6 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。

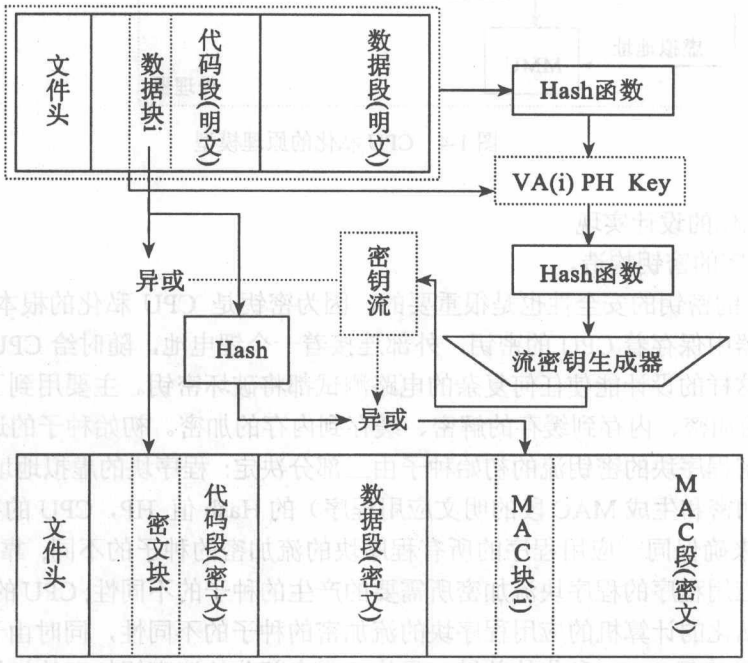

图 1-5　应用程序加密认证

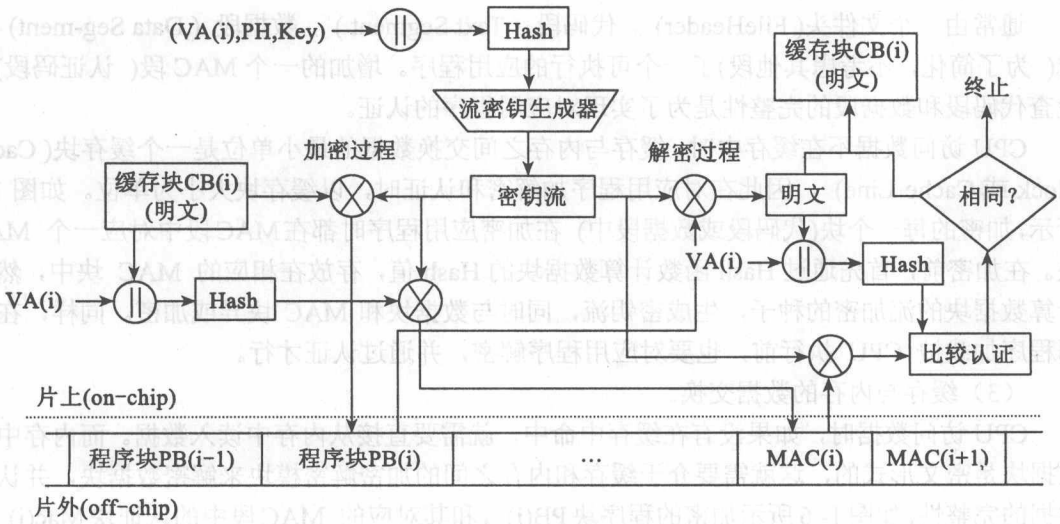

图 1-6　加密解密与认证模块