申请同济大学工学硕士学位论文

”和欣”操作系统中基于数字水印技术安全构件模型的研究

（国家863“软件重大专项”项目　编号：　2001AA113400）

培养单位：电子与信息工程学院

一级学科：计算机科学与技术

二级学科：计算机应用

研 究 生：胡天华

指导教师：顾伟楠　教授

二〇〇六年二月

A dissertation submitted to
Tongji University in conformity with the requirements for
the degree of Master of Science

The Study of Security Component Model Based on Digital Watermarking Technology in "Elastos"

(Supported by National 863 High Tech Program, Grant No. 2001AA113400)

School/Department: School of Electronics and Information Engineering
Discipline: Computer Science and Technology
Major: Computer Application
Candidate: Tian-hua Hu
Supervisor: Prof. Wei-Nan Gu

February, 2006

和”作统基数水技安构模的究
”欣操系中于字印术全件型研

天华
济学
同大

学位论文版权使用授权书

本人完全了解同济大学关于收集、保存、使用学位论文的规定，同意如下各项内容：按照学校要求提交学位论文的印刷本和电子版本；学校有权保存学位论文的印刷本和电子版，并采用影印、缩印、扫描、数字化或其它手段保存论文；学校有权提供目录检索以及提供本学位论文全文或者部分的阅览服务；学校有权按有关规定向国家有关部门或者机构送交论文的复印件和电子版；在不以赢利为目的的前提下，学校可以适当复制论文的部分或全部内容用于学术活动。

学位论文作者签名：

年　月　日

经指导教师同意，本学位论文属于保密，在　　年解密后适用本授权书。

指导教师签名：　　学位论文作者签名：

年　月　日　　年　月　日

同济大学学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，进行研究工作所取得的成果。除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人创作的、已公开发表或者没有公开发表的作品的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。本学位论文原创性声明的法律责任由本人承担。

签名：

年　　月　　日

摘要

“和欣”操作系统是面向网络时代的嵌入式系统，能更有效地支持WEB服务，在体系结构上它是完全面向构件的网络操作系统。随着网络技术的发展，操作系统安全问题日益严重，建立安全防范体系的需求越来越强烈。“和欣”操作系统是国家863软件重大专项课题“基于中间件技术的因特网嵌入式操作系统及跨操作系统的中间件运行平台”的研究成果，本文在该成果上提出了一种基于数字水印技术的安全模型，模型的提出是与“和欣”操作系统紧密相关的，借鉴了很多“和欣”的构件化概念，并且利用“和欣”操作系统是面向构件的操作系统，容易开发并部署系统级的构件这一特点，设计了一个安全构件,并将安全构件封装为操作系统构件。所有期望得到运行环境的应用程序在被操作系统loader装载入内存空间之前，会被强制接受安全构件的“检查”，安全构件负责应用程序中水印的提取与验证，确保了只有合法程序才能被运行，是一种解决操作系统中安全问题的可行策略。

本文提出安全模型的同时，完成了安全构件的接口设计以及安全构件中使用的加密模块，实现了对应用程序中的水印的嵌入、提取与验证。设计的安全构件由操作系统构件管理模块统一管理，从而可以做到，不用更改操作系统内核就可以方便地实现系统功能的扩展；更重要的是安全构件与外部应用程序以及操作系统的交互完全通过接口进行，保证了构件的实现细节不会影响到使用构件的程序，必要时系统可以根据需要动态地置换和升级安全构件。

关键词：数字水印，中间件，安全构件

ABSTRACT

“Elastos” is a network age based embedded operation system . Web service is effectively supported by “Elastos”. It is a completely component based network operation system in system structure. As the development of network technology, the security problem of operation system is becoming more and more grievously. It is necessary to build a Safe guard system. “Elastos” is the research achievement of "Component oriented Embedded Networking Technology" which belongs to National 863 High Tech Program (863 Program). In this paper, the security model of “Elastos” based on the digital watermarking technology is presented. The presented model is closely related with “Elastos” and uses the component conception of “Elastos”. According to the feature that to develop and deploy a system component in “Elastos” is easy, the security component is designed and encapsulated as operation system level component. Before loaded to Memory space by operation system loader, the applications that want to achieve the running environment will be checked by security component. The applications’ watermarking is extracted and validated by security component, which ensure only the legal applications would be loaded. This is one of the feasible strategies to resolve security problem of operation system.

Following the presented security model, the interface and Encryption module of security component is accomplished. The embedding, extracting and validating of application’s watermarking are realized. The security component is managed by operation system component management module. So the expansion of system function can be realized without changing operation system kernel. More importantly, the security component communicates with operation system and exterior applications completely through the interface, which guarantees that the component’s realization detail will not affect the applications that are using the component. When necessity the system may dynamically replace and upgrade security component according to needs.

Key Words: digital watermarking, component technology, security component

目录

第一章 引言

1.1 课题来源及研究意义

本课题来源于国家 863 重大软件专项项目——“基于中间件技术的因特网嵌入式操作系统及跨操作系统的中间件运行平台”。“和欣”操作系统是一款面向嵌入式设备的操作系统，它区别与其他嵌入式操作系统的地方在于：该操作系统是面向网络时代的嵌入式设备，能更有效地支持 WEB 服务，在体系结构上是完全面向构件、中间件技术的网络操作系统，其精髓是科泰世纪公司原创的 CAR 构件技术。“和欣”操作系统的优势在于：该系统可实现“傻瓜”软件运行平台，面向消费市场，能够实现“用户零维护”；具有“瞬间启动”功能，各类应用软件、游戏和新闻均可通过图像浏览器实现“点击运行”；该系统具有“即插即用”功能，用户无需为各种硬件设备安装驱动软件，在配套软件方面，面对不同厂家的软件，可以以目标代码形式实现“无缝链接”。

本课题运用嵌入式和欣操作系统的灵活内核技术，结合构件技术，软件水印技术提出了安全构件模型，并将安全构件封装为操作系统构件。所有期望得到运行环境的应用程序在被操作系统 loader 装载入内存空间之前，会被强制接受安全构件的“检查”，安全构件负责应用程序中水印的提取与验证，确保了只有合法程序才能被运行，从而弥补了传统的操作系统安全策略的漏洞, 是一种解决操作系统中安全问题的可行策略。

从产品的角度来说，“和欣”操作系统的一个重要应用就是智能手机，作为未来智能手机操作系统，“和欣”要保证数据在传输过程中的安全性。随着手机上的数据传输量越来越大，更多的功能在手机上实现，包括：游戏下载，软件下载，在线播放视频和音频，email 服务，网上消费等等，大量的数据在网络上传输很容易让攻击者有机可乘，所以给“和欣”操作系统提供一套更完整的安全方案是至关重要的。因此，本课题对和欣操作系统下安全构件模型的研究，不仅具有重要的理论意义，还具有广阔的应用前景。

1.2 论文结构

本文的内容共分为七章:

第一章简要介绍了本课题的来源及研究意义。

第二章简单介绍了数字水印技术。

第三章主要介绍了“和欣”嵌入式操作系统及其构件运行平台。

第四章主要论述了和欣操作系统的CAR构件技术。

第五章具体论述了基于数字水印技术的安全构件模型并进行了分析。

第六章论述了安全构件的设计与实现。

第七章对目前的研究工作进行了总结，并提出了未来可能的研究方向和关键性问题。

第 2 章 数字水印技术

2．1 数字水印技术诞生的现实背景

计算机和网络的广泛应用，大大地方便了人们获取信息和交流信息。由于数字化信息以多种形式在网络上迅速便捷地传输，政府、企业及个人都逐渐把网络作为主要的通信手段，大量重要文件和个人信息以数字化形式存储和传输，电子商务则通过网络为我们提供了各种服务，网络与信息安全问题变得越来越重要。

传统的信息加密方法可以加密文本信息，保证其传输的安全，但如果要对图像、视频和声音等多媒体信息进行加密，则基于密码学的传统加密方法就比较困难了，随着计算机处理能力的快速提高，这种通过不断增加密钥长度来提高系统密级的方法变得越来越不可靠。特别是随着网络多媒体技术的发展，信息已经不仅仅局限于文本，许多信息是图形图像和视频格式，需要认证和版权保护的声像数据也越来越多。此外，在军事领域，人们可能需要将一幅作战地图隐藏在一幅艺术作品中。这些应用需求正是数字水印技术要解决的问题。

2．2 能够解决现实问题方案的提出—数字水印

数字水印 (digital watermarking)是实现版权保护的有效办法，因此如今已成为多媒体信息安全研究领域的一个热点，也是信息隐藏技术研究领域的重要分支。该技术即是通过在原始数据中嵌入秘密信息--水印 (watermark)来证实该数据的所有权。这种被嵌入的水印可以是一段文字、标识、序列号等，而且这种水印通常是不可见或不可察的，它与原始数据 (如图像、音频、视频数据)紧密结合并隐藏其中，并可以经历一些不破坏源数据使用价值或商用价值的操作而能保存下来。在数字水印系统中，隐藏信息的丢失，即意味着版权信息的丢失 ，从而也就失去了版权保护的功能，也就是说，这一系统就是失败的。

数字水印的定义：数字水印是永久镶嵌在其它数据（宿主数据）中具有可鉴别性的数字信号或模式，而且并不影响宿主数据的可用性。

2．3 数字水印介绍

数字水印的基本特性如下：

1）隐蔽性：在数字作品中嵌入数字水印不会引起明显的降质，并且不易被察觉。

- 隐藏位置的安全性：水印信息隐藏于数据而非文件头中，文件格式的变换不应导致水印数据的丢失。
- 鲁棒性：所谓鲁棒性是指在经历多种无意或有意的信号处理过程后，数字水印仍能保持完整性或仍能被准确鉴别。可能的信号处理过程包括信道噪声、滤波、数/模与模/数转换、重采样、剪切、位移、尺度变化以及有损压缩编码等。

下面介绍一下典型数字水印技术模型。如下图反映的是完成将水印信息加入到原始数据中的功能。如图 1.1。

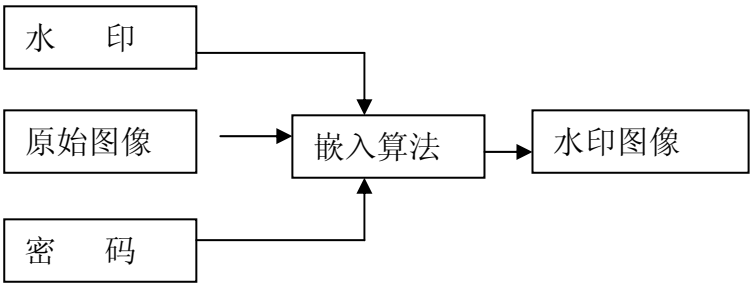


图 1.1 水印信号嵌入模型

如下图反映的是用以判断某一数据中是否含有指定的水印信号的功能。如图 1.2。

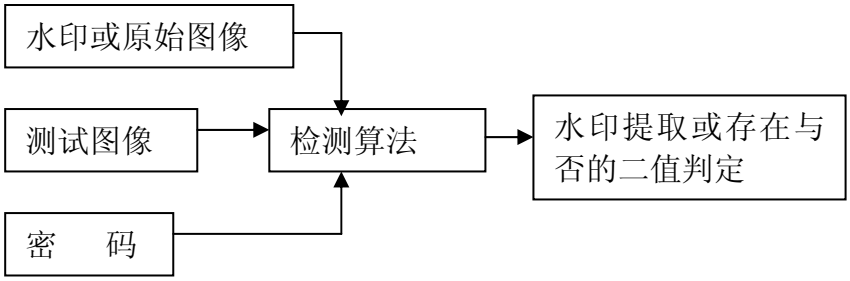


图 1.2：水印信号检测模型

再来看看数字水印的分类有哪些。

数字水印技术可以从不同的角度进行划分。

- **按特性划分**

按水印的特性可以将数字水印分为鲁棒数字水印和脆弱数字水印两类。鲁棒数字水印主要用于在数字作品中标识著作权信息，如作者、作品序号等，它要求嵌入的水印能够经受各种常用的编辑处理；脆弱数字水印主要用于完整性保护，与鲁棒水印的要求相反，脆弱水印必须对信号的改动很敏感，人们根据脆弱水印的状态就可以判断数据是否被篡改过

- **按水印所附载的媒体划分**

按水印所附载的媒体，我们可以将数字水印划分为图像水印、音频水印、视频水印、文本水印以及用于三维网格模型的网格水印等。随着数字技术的发展，

会有更多种类的数字媒体出现，同时也会产生相应的水印技术。

- **按检测过程划分**

按水印的检测过程可以将数字水印划分为明文水印和盲水印。明文水印在检测过程中需要原始数据，而盲水印的检测只需要密钥，不需要原始数据。一般来说，明文水印的鲁棒性比较强，但其应用受到存储成本的限制。目前学术界研究的数字水印大多数是盲水印。

- **按内容划分**

按数字水印的内容可以将水印划分为有意义水印和无意义水印。有意义水印是指水印本身也是某个数字图像（如商标图像）或数字音频片段的编码；无意义水印则只对应于一个序列号。有意义水印的优势在于，如果由于受到攻击或其他原因致使解码后的水印破损，人们仍然可以通过视觉观察确认是否有水印。但对于无意义水印来说，如果解码后的水印序列有若干码元错误，则只能通过统计决策来确定信号中是否含有水印。

- **按用途划分**

不同的应用需求造就了不同的水印技术。按水印的用途，我们可以将数字水印划分为票据防伪水印、版权保护水印、篡改提示水印和隐蔽标识水印。

票据防伪水印是一类比较特殊的水印，主要用于打印票据和电子票据的防伪。一般来说，伪币的制造者不可能对票据图像进行过多的修改，所以，诸如尺度变换等信号编辑操作是不用考虑的。但另一方面，人们必须考虑票据破损、图案模糊等情形，而且考虑到快速检测的要求，用于票据防伪的数字水印算法不能太复杂。

版权标识水印是目前研究最多的一类数字水印。数字作品既是商品又是知识作品，这种双重性决定了版权标识水印主要强调隐蔽性和鲁棒性，而对数据量的要求相对较小。

篡改提示水印是一种脆弱水印，其目的是标识宿主信号的完整性和真实性。

隐蔽标识水印的目的是将保密数据的重要标注隐藏起来，限制非法用户对保密数据的使用。

- **按水印隐藏的位置划分**

按数字水印的隐藏位置，我们可以将其划分为时（空）域数字水印、频域数字水印、时/频域数字水印和时间/尺度域数字水印。

时（空）域数字水印是直接在信号空间上叠加水印信息，而频域数字水印、时/频域数字水印和时间/尺度域数字水印则分别是在 DCT 变换域、时/ 频变换域和小波变换域上隐藏水印。

随着数字水印技术的发展，各种水印算法层出不穷，水印的隐藏位置也不再

局限于上述四种。应该说，只要构成一种信号变换，就有可能在其变换空间上隐藏水印。

2．4 数字水印的应用前景

多媒体技术的飞速发展和 Internet 的普及带来了一系列政治、经济、军事和文化问题，产生了许多新的研究热点，以下几个引起普遍关注的问题构成了数字水印的研究背景。

1） 字作品的知识产权保护

数字作品（如电脑美术、扫描图像、数字音乐、视频、三维动画）的版权保护是当前的热点问题。由于数字作品的拷贝、修改非常容易，而且可以做到与原作完全相同，所以原创者不得不采用一些严重损害作品质量的办法来加上版权标志，而这种明显可见的标志很容易被篡改。

“数字水印”利用数据隐藏原理使版权标志不可见或不可听，既不损害原作品，又达到了版权保护的目的。目前，用于版权保护的数字水印技术已经进入了初步实用化阶段，IBM 公司在其“数字图书馆”软件中就提供了数字水印功能，Adobe 公司也在其著名的 Photoshop 软件中集成了 Digimarc 公司的数字水印插件。然而实事求是地说，目前市场上的数字水印产品在技术上还不成熟，很容易被破坏或破解，距离真正的实用还有很长的路要走。

2）商务交易中的票据防伪

随着高质量图像输入输出设备的发展，特别是精度超过 1200dpi 的彩色喷墨、激光打印机和高精度彩色复印机的出现，使得货币、支票以及其他票据的伪造变得更加容易。

据美国官方报道，仅在 1997 年截获的价值 4000 万美元的假钞中，用高精度彩色打印机制造的小面额假钞就占 19％，这个数字是 1995 年的 9.05 倍。目前，美国、日本以及荷兰都已开始研究用于票据防伪的数字水印技术。其中麻省理工学院媒体实验室受美国财政部委托，已经开始研究在彩色打印机、复印机输出的每幅图像中加入唯一的、不可见的数字水印，在需要时可以实时地从扫描票据中判断水印的有无，快速辨识真伪。

另一方面，在从传统商务向电子商务转化的过程中，会出现大量过度性的电子文件，如各种纸质票据的扫描图像等。即使在网络安全技术成熟以后，各种电子票据也还需要一些非密码的认证方式。数字水印技术可以为各种票据提供不可见的认证标志，从而大大增加了伪造的难度。

3） 数据的隐藏标识和篡改提示

数据的标识信息往往比数据本身更具有保密价值，如遥感图像的拍摄日期、

经/纬度等。没有标识信息的数据有时甚至无法使用，但直接将这些重要信息标记在原始文件上又很危险。数字水印技术提供了一种隐藏标识的方法，标识信息在原始文件上是看不到的，只有通过特殊的阅读程序才可以读取。这种方法已经被国外一些公开的遥感图像数据库所采用。

此外，数据的篡改提示也是一项很重要的工作。现有的信号拼接和镶嵌技术可以做到“移花接木”而不为人知，因此，如何防范对图像、录音、录像数据的篡改攻击是重要的研究课题。基于数字水印的篡改提示是解决这一问题的理想技术途径，通过隐藏水印的状态可以判断声像信号是否被篡改。

4）隐蔽通信及其对抗

数字水印所依赖的信息隐藏技术不仅提供了非密码的安全途径，更引发了信息战，尤其是网络情报战的革命，产生了一系列新颖的作战方式，引起了许多国家的重视。

网络情报战是信息战的重要组成部分，其核心内容是利用公用网络进行保密数据传送。迄今为止，学术界在这方面的研究思路一直未能突破“文件加密”的思维模式，然而，经过加密的文件往往是混乱无序的，容易引起攻击者的注意。网络多媒体技术的广泛应用使得利用公用网络进行保密通信有了新的思路，利用数字化声像信号相对于人的视觉、听觉冗余，可以进行各种时（空）域和变换域的信息隐藏，从而实现隐蔽通信。

第 3 章 和欣嵌入式操作系统

3.1 和欣操作系统概述

3.1.1 和欣操作系统介绍

《和欣 1.1》是 32 位嵌入式操作系统。该操作系统可以从多个侧面进行描述：32 位嵌入式操作系统。操作系统基于微内核，具有多进程、多线程、抢占式、基于线程的多优先级任务调度等特性。提供 FAT 兼容的文件系统，可以从软盘、硬盘、FLASH ROM 启动，也可以通过网络启动。和欣操作系统体积小，速度快，适合网络时代的绝大部分嵌入式信息设备。

完全面向构件技术的操作系统。操作系统提供的功能模块全部基于 CAR（Carefree Application Run-Time）构件技术，因此是可拆卸的构件，应用系统可以按照需要剪裁组装，或在运行时动态加载必要的构件。

从传统的操作系统体系结构的角度来看，和欣操作系统可以看成是由微内核、构件支持模块、系统服务器组成的。

- 微内核：主要可分为 4 大部分：硬件抽象层（对硬件的抽象描述，为该层之上的软件模块提供统一的接口）；内存管理（规范化的内存管理接口，虚拟内存管理）；任务管理（进程管理的基本支持，支持多进程，多线程）；进程间通信（实现进程间通信的机制，是构件技术的基础设施）。
- 构件支持模块：提供了对 CAR 构件的支持，实现了构件运行环境。构件支持模块并不是独立于微内核单独存在的，微内核中的进程间通讯部分为其提供了必要的支持功能。
- 系统服务器：在微内核体系结构的操作系统中，文件系统、设备驱动、网络支持等系统服务是由系统服务器提供的。在和欣操作系统中，系统服务器都是以动态链接库的形式存在。

“和欣”操作系统可以广泛应用于工业装备、信息家电、汽车电子、手持设备、办公设备、商业电子、信息安全、军事国防等领域的各种嵌入式设备。

和欣 SDK 提供了应用软件的开发环境和工具。开发“和欣”应用软件的开发环境。如图 3.1。

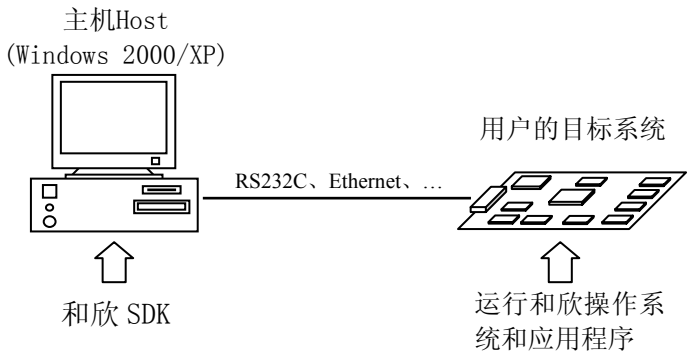


图 3.1 “和欣”应用软件的开发环境

开发“和欣”应用软件的过程，如图 3.2：

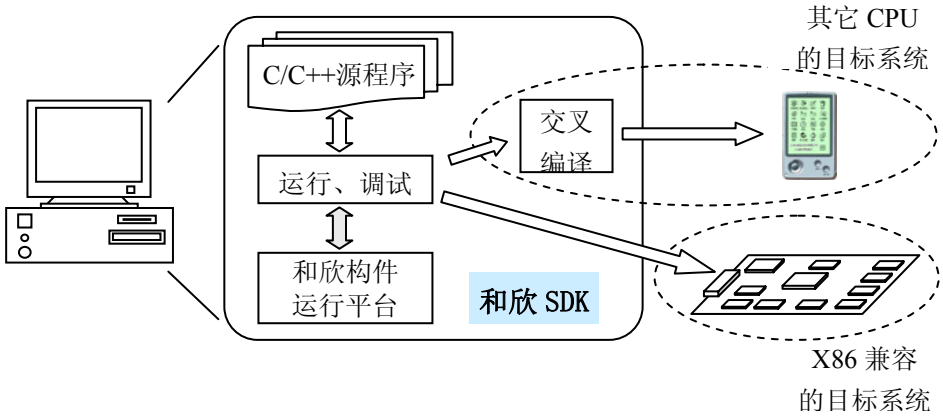


图 3.2 “和欣”应用软件的开发过程

3.1.2 和欣灵活内核简介

和欣操作系统的实现全面贯穿了 CAR 思想，CAR 构件可以运行于不同地址空间或不同的运行环境。我们可以把操作系统的内核地址区看成是一段特殊的地址空间，用户可以根据运行时的需求，自主选择将操作系统的某些系统服务构件、文件系统、图形系统、设备驱动构件等运行于内核地址空间或用户地址空间。与传统的操作系统的"大内核"、"微内核"体系结构相比，和欣操作系统内核里提供的系统服务，完全可以由用户依据系统自身的需求动态决定。因此我们称和欣操作系统内核为"灵活内核"（Agile Kernel）。

和欣灵活内核的体系结构，利用构件和中间件技术解决了长期以来困扰操作系统体系结构设计者的大内核和微内核在性能、效率与稳定性、安全性之间不能两全其美的矛盾。如图 3.3。

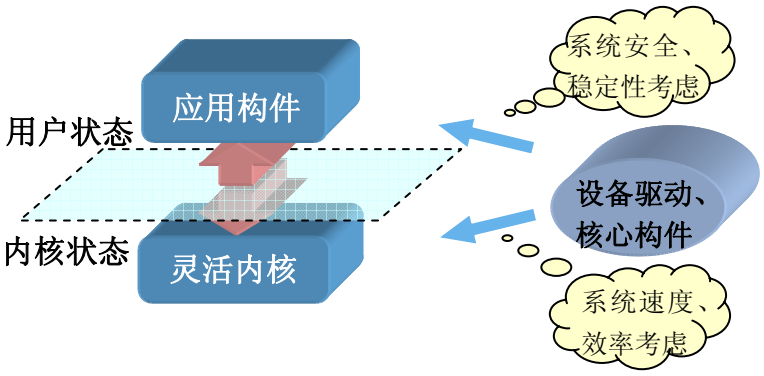


图 3.3 “和欣”灵活内核与系统构件和应用构件的关系

3.1.3 和欣操作系统提供的功能

从应用编程的角度看，和欣操作系统提供了一套完整的、符合 CAR 规范的系统服务构件及系统 API，为在各种嵌入式设备的硬件平台上运行 CAR 二进制构件提供了统一的编程环境。

和欣操作系统还提供了一组动态链接构件库，这些构件库通常是开发嵌入式应用系统时不可缺少的：

- 图形系统构件库（方便开发图形用户操作界面）；
- 设备驱动构件库（各种输入输出设备的驱动）；
- 文件系统构件库（FAT 兼容，包括对 FLASH 等的支持）；
- 网络系统构件库（TCP/IP 等网络协议支持）。

系统提供的构件库，以及用户开发的应用程序构件都是通过系统接口与内核交互，从这个意义上说，他们处于同样的地位。用户可以开发性能更好或者更符合需求的文件系统、网络系统等构件库，替换由科泰世纪公司提供的构件库，也可以开发并建立自己的应用程序构件库。这就是基于构件技术操作系统的优势之一。

此外，为了方便用户编程，在和欣 SDK 中还提供了以下函数库：

- 与微软 Win32 API 兼容的应用程序编程接口(elaw32 API)；
- 标准 C 运行库（libc）；
- 和欣提供的工具类函数（elautil）。

对程序员来说，和欣操作系统提供的用户编程接口与下一节将要介绍的和欣构件运行平台完全一样。所以，在相互兼容的硬件平台上，不管运行的是和欣操作系统还是 Windows 操作系统，应用程序可以不加区分地在其上运行。

和欣操作系统实现并支持系统构件及用户构件相互调用的机制，为 CAR 构件提供了运行环境。关于 CAR 构件的运行环境，其描述与“和欣构件运行平台”是一样的，在此从简。因此，我们可以把和欣操作系统看成是直接运行在硬件平台

上的“和欣构件运行平台”。可以用下图来表示和欣操作系统及其主要构成。如图 3.4。

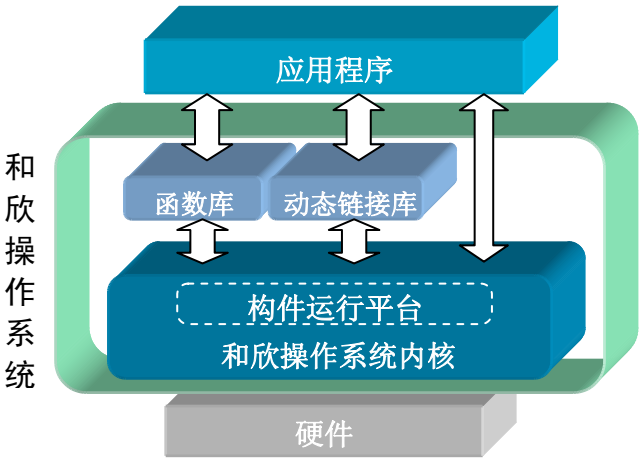


图 3.4“和欣”操作系统的系统结构图

3.4.4 和欣操作系统的优势

和欣操作系统的最大特点就是：

- 全面面向构件技术，在操作系统层提供了对构件运行环境的支持；
- 用构件技术实现了"灵活"的操作系统。

这是和欣操作系统区别于其他商用嵌入式操作系统产品的最大优势。

在新一代因特网应用中，越来越多的嵌入式产品需要支持 Web Service，而 Web Service 的提供一定是基于构件的。在这种应用中，用户通过网络获得服务程序，这个程序一定是带有自描述信息的构件，本地系统能够为这个程序建立运行环境，自动加载运行。这是新一代因特网应用的需要，是必然的发展方向。和欣操作系统就是应这种需要而开发，率先在面向嵌入式系统应用的操作系统中实现了面向构件的技术。

因此，构件化的和欣操作系统可以为嵌入式系统开发带来以下好处：

- 在嵌入式软件开发领域，导入先进的工程化软件开发技术。嵌入式软件一般用汇编语言、C 语言，在少数系统中已经支持了 C++开发，但是由于还没有一个嵌入式操作系统能够提供构件化的运行环境，可以说，嵌入式软件开发还是停留在手工作坊式的开发方式上。和欣操作系统使得嵌入式应用的软件开发能够实现工程化、工厂化生产，其原理请参考第四章 和欣 CAR 构件技术。
- 可以动态加载构件。动态加载构件是因特网时代嵌入式系统的必要功能。新一代 PDA 和移动电话等移动电子产品，不能再像以前那样由厂家将所有的功能都做好后固定在产品里，而要允许用户从网上获得自己感兴趣的程序。

- 随时和动态地实现软件升级。动态加载构件的功能，同样可以用于产品的软件升级，开发商不必为了添加了部分功能而向用户重新发布整套软件，只需要升级个别构件。
- 灵活的模块化结构，便于移植和剪裁。可以很容易定制成为针对不同硬件配置的紧凑高效的嵌入式操作系统。添加或删除某些功能模块也非常简单。
- 嵌入式软件开发商容易建立自己的构件库。在不同开发阶段开发的软件构件，其成果很容易被以后的开发所共享，保护软件开发投资。软件复用使得系列产品的开发更加容易，缩短新产品开发周期。
- 容易共享第三方软件开发商的成果。面向行业的构件库的建设，社会软件的丰富，使得设备厂家不必亲自开发所有的软件，可以充分利用现有的软件资源，充分发挥自己的专长为自己的产品增色。
- 跨操作系统平台兼容，降低软件移植的风险。在和欣开发环境上开发的软件所具有的跨平台特性，使得用户可以将同样的可执行文件不加修改地运行在和欣操作系统（嵌入式设备）与 Windows 2000/XP（PC）上。特别是对于需要将 Windows 上的软件移到嵌入式系统以降低产品成本的用户，这一特点不仅可以大大节约软件移植的费用，还可以避免因移植而带来的其他隐患。
- 功能完备的开发环境和方便的开发工具，帮助嵌入式开发人员学习和掌握先进的构件化软件编程技术，提高软件开发效率。应用软件可以在开发环境下开发调试，与硬件研制工作同时进行，缩短产品研制周期，从而减少成本。

3.2 和欣构件运行平台

3.2.1 平台简介

和欣构件运行平台提供了一套符合 CAR 规范的系统服务构件及支持构件相关编程的 API 函数，实现并支持系统构件及用户构件相互调用的机制，为 CAR 构件提供了编程运行环境。和欣运行平台在不同操作系统上有不同的实现，符合 CAR 编程规范的应用程序通过该平台实现二进制级跨操作系统平台兼容。

在和欣操作系统中，和欣构件运行平台与“和欣灵活内核”共同构成了完整的操作系统。

在 Windows 2000、WinCE、Linux 等其他操作系统上，和欣构件运行平台屏蔽了底层传统操作系统的具体特征，实现了一个构件化的虚拟操作系统。在和欣

构件运行平台上开发的应用程序，可以不经修改、不损失太多效率、以相同的二进制代码形式，运行于传统操作系统之上。

下图直观地显示了和欣构件运行平台在 Windows 2000/XP、和欣操作系统中的位置。如图 3.5。

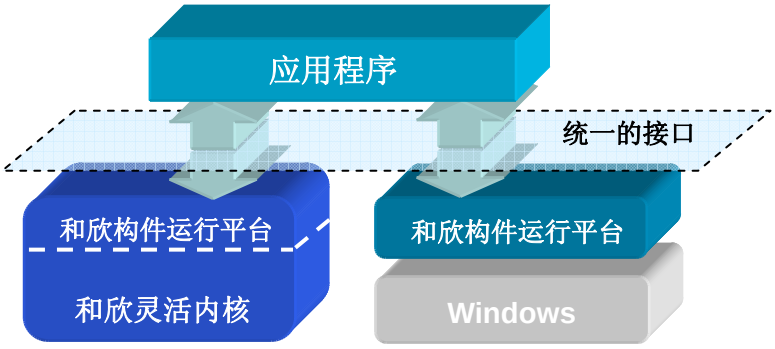


图 3.5 “和欣”构件运行平台与操作系统的关系

以下是和欣构件运行平台的设计目标和要求：

1)提供统一接口，屏蔽真实操作系统的具体特征

和欣构件运行平台作为操作系统与应用程序交互的唯一接口，为用户屏蔽真实操作系统的具体特征。

和欣构件运行平台提供了系统一级的 CAR 库的服务，提供中间件的自动生成机制。在“客户/中间件/服务器”(Client/Middleware/Server)三层体系结构中，当客户程序需要调用一个软件服务时，和欣构件运行平台可根据与应用相关的元数据动态生成代理构件，自动构造客户与服务相关的运行环境，客户与服务不再直接发生关系，所有的通信机制都通过和欣构件运行平台自动实现。和欣构件运行平台同时为用户提供一套完整的与系统服务相关的 API 函数及系统构件服务(如线程调度、内存分配、创建新的进程对象等)，对用户而言，“和欣构件运行平台”提供了操作系统能提供的所有服务。

2）目标代码是操作系统所能识别并执行的最终机器指令

基于和欣构件运行平台的目标代码是操作系统所能识别并加以执行的最终机器指令。基于和欣构件运行平台开发的应用程序的目标代码应该是二进制流的，是操作系统所能识别并加以执行的最终机器指令，无需类似 Java 虚拟机及 C#虚拟机解释执行，以便充分发挥系统的效率。

3）实现应用程序二进制级跨平台提供可能

和欣构件运行平台的物理表现必须为 DLL 动态链接库形式，“和欣”支持动态链接库，其软件产品遵守标准的 PE 文件格式，这确保了文件在使用上及物理存储上与 Windows 系列操作系统的一致， CAR 构件与微软 COM 构件二进制兼容为和欣构件运行平台实现应用程序二进制级跨平台奠定了坚实的基础。这样就可以在不同的操作系统上实现有相同功能的 CAR 构件运行环境和开发平台，从

而实现基于 CAR 构件技术的同一软件在“和欣”操作系统与其它通用操作系统平台如 Windows, Linux 上的二进制兼容。这一技术的实现将使 CAR 成为通用的构件、中间件开发平台，有利于不同类别操作系统的普及应用。

4）为实现灵活内核提供相关支持

基于 CAR 构件技术的和欣构件运行平台提供了安全控制以及应用服务器的运行、管理与调度机制，可以提供对用户透明的服务，比如可以根据不同的运行环境进行动态加载、更换、卸载与通信协议相关的网络服务构件、文件系统构件、驱动程序构件、以及分布式的事务处理构件等。为“和欣”操作系统的“灵活内核”(Agile Kernel)体系结构提供有力的支持。

和欣构件运行平台的主要功能设计：

实现一套内核代码映射引出(export)机制，该机制与 DLL 的引出函数调用相关，实现和欣构件运行平台所提供的 API 函数在用户态及内核态都能够调用，并且在所有用户空间只有一份相关代码的拷贝，最大程度上节省系统资源的消耗。

和欣构件运行平台提供一组与 CAR 服务相关的 API 函数(CAR 库函数)，提供对用户 CAR 的基本支持。

和欣构件运行平台提供基础 API 支持，为用户提供一些“和欣”操作系统所特有的、必需的函数调用。

提供获得标准系统内核构件对象接口指针的手段，实现对内核对象的访问及控制。

根据二进制构件的自描述信息自动生成构件的运行环境，动态加载构件。

提供构件之间的自动通信机制，构件间通信可以跨进程甚至跨网络。

构件的运行状态监控，错误报告等。

提供可干预构件运行状态的机制。

构件的生命周期管理，如进程延续(Persistence)控制、事务元(Transaction)控制等。

总的来说，在“和欣”操作系统上“和欣构件运行平台”与底层操作系统密不可分，在更大程度上“和欣构件运行平台”是底层操作系统的一个抽象层，和欣构件运行平台为客户程序与操作系统的交互界面，为客户程序提供相关的函数调用及构件服务，同时隐藏了底层操作系统的特征。

和欣构件运行平台代码映射机制是实现“和欣构件运行平台”代码共享的基本手段。

在“和欣”操作系统中，系统代码及用户代码都将被加载到共享地址空间(私有代码被加以保护，禁止用户进程直接访问)，和欣构件运行平台将系统的共享

代码索引表生成一个虚拟的动态链接库 zeesys.dll，通过 DLL 的引出机制，所有用户程序共享这些代码，操作系统在加载用户进程的初始化时通过自动在用户空间注册该虚拟的 zeesys. dll.。zeesys. dll 的引出函数表与和欣构件运行平台的共享代码索引表一一对应。

3.2.2 平台提供的功能

从和欣构件运行平台的定义，我们知道该平台为 CAR 提供了运行环境。从这个意义上，我们说的 CAR 技术也可以理解为在运行环境中对 CAR 规范提供支持的程序集合。

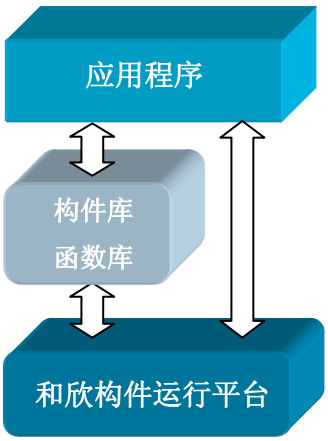
从编程的角度看，和欣构件运行平台提供了一套系统服务构件及系统 API（应用程序编程接口），这些是在该平台上开发应用程序的基础。

和欣操作系统提供的其他构件库也是通过这些系统服务构件及系统 API 实现的。系统提供的这些构件库为应用编程开发提供了方便：

- 图形系统构件库；
- 设备驱动构件库；
- 文件系统构件库；
- 网络系统构件库。

从和欣构件运行平台来看，这些构件和应用程序的构件是处于同样的地位。用户可以开发性能更好或者更符合需求的文件系统、网络系统等构件库，替换这些由科泰世纪公司提供的构件库，也可以开发并建立自己的应用程序构件库。

下图显示出和欣构件运行平台的功能及其与构件库、应用程序的关系。如图



3.6。

图 3.6 和欣构件运行平台功能图

从支持 CAR 构件的运行环境的角度看，和欣构件运行平台提供了以下功能：

- 根据二进制构件的自描述信息自动生成构件的运行环境，动态加载构件；

- 提供构件之间的自动通信机制，构件间通信可以跨进程甚至跨网络；
- 构件的运行状态监控，错误报告等；
- 提供可干预构件运行状态的机制，如负载均衡、线程同步、访问顺序控制、安全（容错）性控制、软件使用权的控制等；
- 构件的生命周期管理，如进程延续（Persistence）控制、事务元（Transaction）控制等；

总之，构件运行平台为 CAR 构件提供了对程序员完全透明的运行环境，构件可以运行在不同地址空间，不同环境，甚至跨网络。构件运行平台自动为构件运行提供支持，配置必要的网络协议、针对不同的输入输出设备的协议。程序员不必过多地去关心诸如网络协议转换及构件运行控制等与其他构件互操作时的协调问题，只需专注于自己需要解决的程序算法的实现。从而可以从繁杂庞大的应用环境体系中解放出来，大大提高编程的效率。

和欣构件运行平台直接运行二进制构件，而不是像 JAVA 和.NET 那样通过虚拟机在运行程序时解释执行中间代码。因此，与其他面向构件编程的系统相比，具有资源消耗小，运行效率高的优点。

3.2.3 平台的技术优势

和欣构件运行平台的主要技术优势列举如下：

- 开发跨操作系统平台的应用软件；
- 对程序员透明的 CAR 构件运行环境，提高编程的效率；
- 直接运行二进制构件代码，实现软件运行的高效率；
- 构件可替换，用户可建立自己的构件库。

需要说明的是，和欣构件运行平台实现的应用软件跨操作系统平台兼容是以具有同样的硬件体系结构为前提的。目前，和欣构件运行平台还不能支持不同指令系统的 CPU 间的“跨平台”兼容。

3.2.3 利用平台编程

对程序员来说，编写运行于和欣构件运行平台上的程序，运用 CAR 技术和跨平台技术的具体方法，体现在对构件库的接口方法、通用 API 函数的调用上。应用程序运行所需要的动态链接库，则是在程序运行时由和欣构件运行平台自动加载的。

下图简明地表示了编写运行于和欣构件运行平台上的应用程序所需的相关要素之间的关系的示意。如图 3.7。

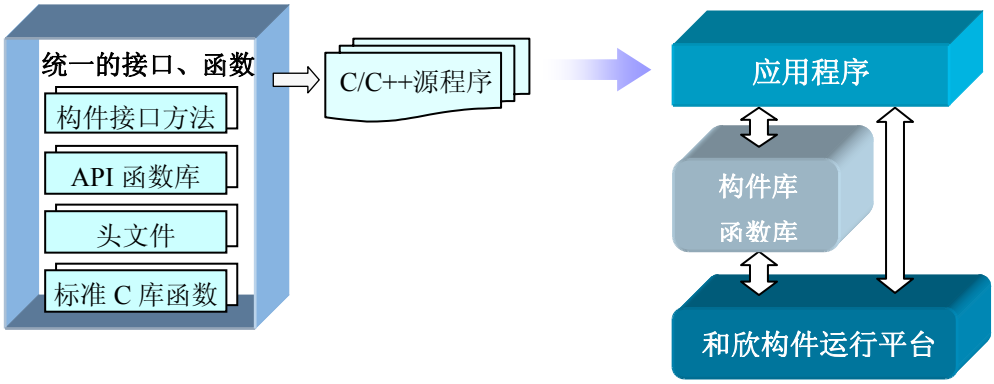


图 3.7 如何编写基于“和欣”构件运行平台的应用程序

第 4 章 和欣 CAR 构件技术

4.1 构件技术介绍

随着软件复杂度的与日俱增，传统的把整个软件的源程序拿来静态编译的方法显然不适合了。在这个前提下，产生了软件拼装模式，把软件分成个个相对独立的目标代码模块，称之为构件。软件开发人员只需要做和自己相关的构件，编译通过，就能够拿来和其他模块组装在一起使用了。通过装卸实现某个功能的构件，就可以实现对系统的灵活升级。如今，已经成熟且广为使用的构件技术有微软的 COM (Component Object Model), OMG 组织的 CORBA (Common Object Request Broker Architecture)等等，用它们生成的构件都是基于二进制目标代码的。现在大行其道的 Java 和.NET，虽然生成的程序都是基于中间代码的，但也处处体现着构件技术的思想。

面向对象技术实现了软件源代码层次的复用，提高了软件开发人员的生产率。构件技术是对面向对象技术的深化，实现了二进制层次上软件的复用，进一步提升了软件开发的效率。根据构件技术，软件系统可以拆分成相对独立的构件，构件之间通过约定的接口进行数据交换和信息传递。构件可以用不同的语言编写，只要符合一组二进制规范即可，这样大大提高了开发的灵活度。

4.2 CAR 构件技术

CAR 构件技术是面向构件编程的编程模型，它规定了一组构件间相互调用的标准，使得二进制构件能够自描述，能够在运行时动态链接。

CAR 兼容微软的 COM。但是和微软 COM 相比，CAR 删除了 COM 中过时的约定，禁止用户定义 COM 的非自描述接口；完备了构件及其接口的自描述功能，实现了对 COM 的扩展；对 COM 的用户界面进行了简化包装，易学易用。

从上面的定义中，可以说 CAR 是微软 COM 的一个子集。CAR 很大程度地借鉴了 COM 技术，保持了和 COM 的兼容性，同时对 COM 进行了重要的扩展。在和欣 SDK 工具的支持下，使得高深难懂的构件编程技术很容易被 C/C++程序员理解并掌握。

CAR 技术是在总结面向对象编程、面向构件编程技术的发展历史和经验，为更好地支持面向 Web Service (WEB 服务)的下一代网络应用软件开发而开发的。CAR 的编程思想是“和欣”技术的精髓，它贯穿于整个技术体系的实现中。

为了在资源有限的嵌入式系统中实现面向中间件编程技术，同时又能得到

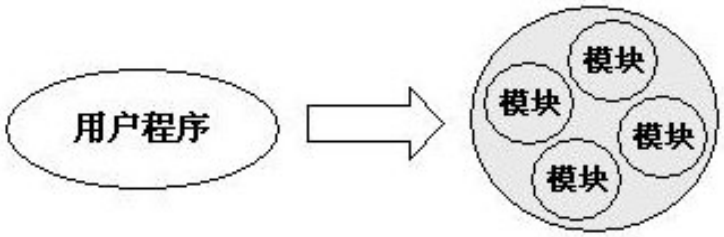
C/C++的运行效率，CAR 没有使用 JAVA 和.NET 的基于中间代码——虚拟机的机制，而是采用了用 C++编程，用和欣 SDK 提供的工具直接生成运行于和欣构件运行平台的二进制代码的机制。用 C++编程实现构件技术，使得更多的程序员能够充分运用自己熟悉的编程语言知识和开发经验，很容易掌握面向构件、中间件编程的技术。在不同操作系统上实现的和欣构件运行平台，使得 CAR 构件的二进制代码可以实现跨操作系统平台兼容。

4.3 CAR 构件的由来

80 年代以来，目标指向型软件编程技术有了很大的发展，为大规模的软件协同开发以及软件标准化、软件共享、软件运行安全机制等提供了理论基础。其发展可以大致分为以下几个阶段。

（1）面向对象编程

通过对软件模块的封装，使其相对独立，从而使复杂的问题简单化。面向对象编程强调的是对象的封装，但模块（对象）之间的关系在编译的时候被固定，模块之间的关系是静态的，在程序运行时不可改变模块之间的关系，就是说在运



行时不能换用零件。其代表是 C++语言所代表的面向对象编程。如图 4.1。

图 4.1 而向对象编程的运行模型，模块之间的关系固定

（2）面向构件编程

为了解决不同软件开发商提供的构件模块（软件对象）可以相互操作使用，构件之间的连接和调用要通过标准的协议来完成。构件化编程模型强调协议标准，需要提供各厂商都能遵守的协议体系。就像公制螺丝的标准一样，所有符合标准的螺丝和螺母都可以相互装配。构件化编程模型建立在面向对象技术的基础之上，是完全面向对象的，提供了动态构造部件模块（运行中可以构造部件）的机制。构件在运行时动态装入，是可换的。其代表是 MICROSOFT 的 COM 技术。如图 4.2。

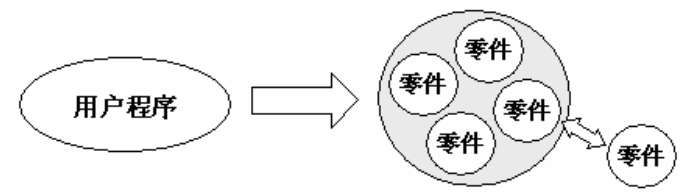


图 4.2 构件化程序的运行模型，运行时零件可替换

（3）面向中间件编程

由于因特网的普及，构件可来自于网络，系统要解决自动下载，安全等问题。因此，系统中需要根据构件的自描述信息自动生成构件的运行环境，生成代理构件即中间件，通过系统自动生成的中间件对构件的运行状态进行干预或控制，或自动提供针对不同网络协议、输入输出设备的服务（即运行环境）。中间件编程更加强调构件的自描述和构件运行环境的透明性，是网络时代编程的重要技术。其代表是 CAR、JAVA 和.NET（C#语言）。如图 4.3。

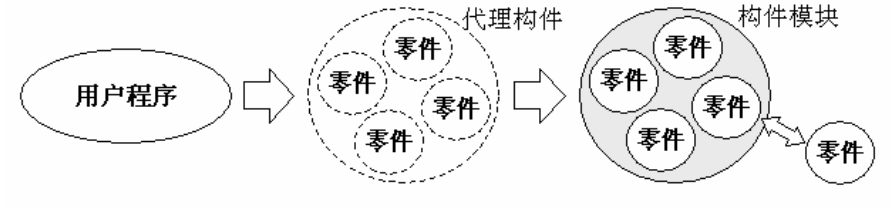


图 4.3 中间件运行环境的模型，动态生成代理构件

在这样的发展过程中，人们逐步深化了对大规模软件开发所需的科学模型、网络环境下软件运行必要机制的理解，使软件技术达到了更高的境界，实现了：

- 构件的相互操作性。不同软件开发商开发的具有独特功能的构件，可以确保与其他人开发的构件实现互操作。
- 软件升级的独立性。实现在对某一个构件进行升级时不会影响到系统中的其他构件。
- 编程语言的独立性。不同的编程语言实现的构件之间可以实现互操作。

构件运行环境的透明性。提供一个简单、统一的编程模型，使得构件可以在进程内、跨进程甚至于跨网络运行。

4.4 CAR 的实现

（1）CAR 对 COM 的封装

- 接口智能指针

在 MS COM 中，用户是直接得到 COM 对象的接口指针，然后通过这个接口指针访问该接口所定义的接口方法。

在 CAR 中，实现了接口智能指针，用户通过接口智能指针来访问这个接口所定义的方法，从这个角度讲，接口智能指针是对接口的封装。在这里，将具体

说明是怎样实现接口智能指针的，并提供一种接口智能指针的实现方法。

在 C++语言中，接口智能指针被定义为类，这个类只有一个成员变量，这个成员变量就是实际指向对象接口的接口指针。如图 4.4。

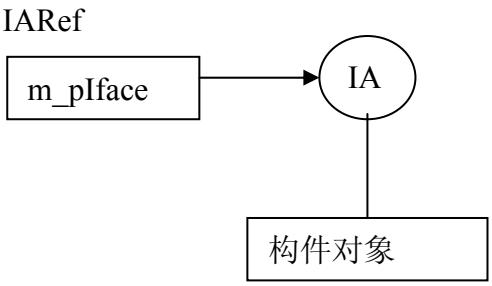
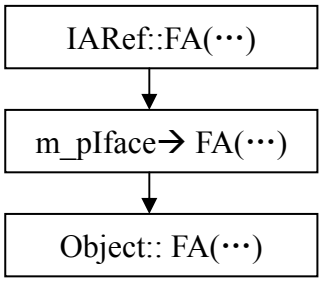


图 4.4 接口指针

图中 IARef 表示接口 IA 对应的接口智能指针，构件对象实现了接口 IA。

接口智能指针类实现了该接口的所有方法。这些接口方法的实现，都是通过



成员变量调用构件对象实现的对应方法。如图 4.5。

图 4.5 接口智能指针

CAR 的接口智能指针实现了智能管理引用计数，不需要用户自己去管理引用计数。

首先定义了对应于 IUnknow 接口(IUnknown 接口的解释参见 MSDN)的两个接口智能指针类，InterfaceRefArg 和 InterfaceRef, InterfaceRef 继承了 InterfaceRefArg。InterfaceRefArg 类中定义了一个成员变量 IUnknown * m_pIface。其它所有接口智能指针追根溯源都继承于此类，因此所有接口智能指针都具有该成员变量。正如所有的接口都继承于 IUnknown 一样，所有的接口智能指针都继承于 InterfaceRefArg。以 IButton 为例，它们继承关系图如下。如图 4.6。

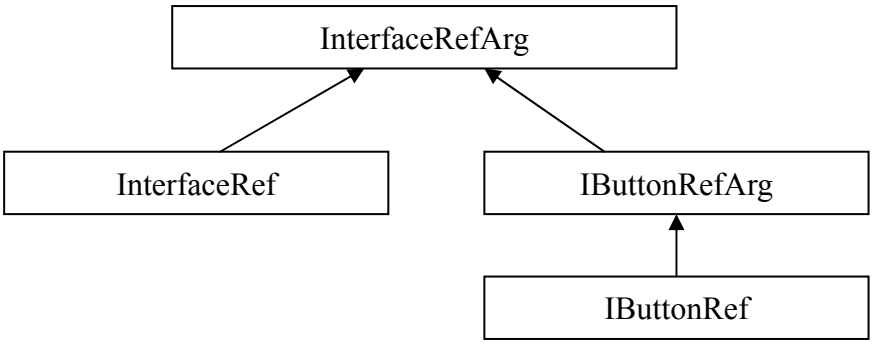


图 4.6 接口智能指针继承关系

通过接口智能指针可以创建出实现了该接口的构件对象，并使该智能指针的成员变量指向这个新创建出的构件对象。

- 类智能指针

在 COM 编程中，如果一个 CObject 对象实现了 IA, IB, IC 三个接口，IA 接口中有方法 FA，IB 接口中有方法 FB, IC 接口中有方法 FC。这样如果用户要调用 FA，FB,，FC 三个方法，那么需要写出下面的代码才能调用到 FA，FB，FC 三个方法(假定用户已经获得接口 IA 的接口指针 pIA)：

```
……
pIA→FA(…);
IB *pIB;
IC *Pic;
pIA→QueryInterface(IID_IB,&pIB);
pIA→QueryInterface(IID_IC,&pIC);
pIB→FB（…）;
pIC→FC（…）;
pIB→Release();
pIC→Release();
……
```

为了简单，这段代码没有考虑方法调用失败的情况。可以看出，只是简单地调用三个方法，却写了九行程序，代码过于麻烦。为了解决这个问题，发明了类智能指针。类智能指针是对构件类的封装，构件类指的是一个构件中定义的类，具体解释参见“CDL 语言”部分。假定已经有了指向上面的 CObject 对象的类智能指针变量 m_cObject，则调用以上三个方法的代码为：

```
m_cObject.FA(…);
m_cObject.FB(…);
m_cObject.FB(…);
```

可以看出，使用类智能指针，代码简单，易懂了很多。在 C++语言中，类智

能指针表现为类，这个类有若干个成员变量，每个成员变量用来指向对象的一个接口，成员变量的数目等于 CAR 对象实现的接口个数，成员变量和构件对象实现的接口一一对应。通过类智能指针，可以调用构件对象实现的所有接口方法，例如上例中，通过类智能指针 m_cObject，就可以调用接口 IA 的接口方法 FA，也可以调用接口 IB 的接口方法 FB，也可以调用接口 IC 的接口方法 FC。

采用类智能指针继承接口智能指针的方式，实现了类智能指针。如图 4.7。

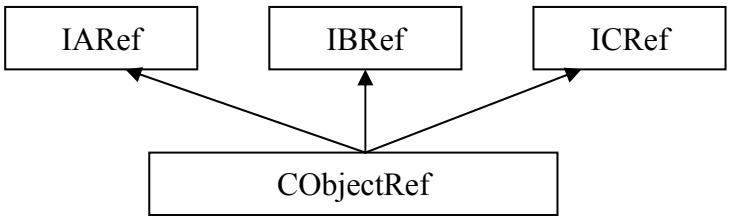


图 4.7 接口智能指针实现

在这种情况下，当用户调用 CObjectRef::FA(…)时，事实上它调用的就是 IARef::FA(…)。

通过类智能指针可以创建出这个类智能指针对应的构件对象，并且使这个类智能指针继承来的成员变量指向这个新创建出的对象。

- 类别智能指针

就像接口智能指针是对接口的封装一样，类别智能指针是对类别的封装物以类聚，具有相同特点的对象，我们可以把它们划分成一类。譬如，各种型号的声卡，它们是用于记录和播放声音的硬件，我们统称它们为声卡，那么声卡就是一个类别。

计算机上，每种型号的声卡都具有相应的驱动程序，但应用程序在使用驱动程序时，通常不关心正在使用的具体是什么声卡的驱动程序，而只希望使用系统中的可用的声卡。为解决这样的问题，在 CAR 中提出了类别的概念，CAR 类别是一组接口的集合，这组接口体现了该类别的共有特性。这样，CAR 允许构件客户在创建构件对象时通过指定具体的构件类来创建对象，也可以通过指定一个构件类别来创建对象。同样以声卡为例，可以把声卡的共用接口抽象出来，并定义为一个类别。当构件客户指定创建声卡类别的构件对象时，系统将自动为客户创建个当前可用的声卡驱动构件对象。

如前所述，一个类别是一个接口的集合，它可以被构件类和类别继承。所有继承这个类别的构件类，我们把它们划分成一类。继承这个类别的构件类将实现这个类别包含的所有接口。继承这个类别的类别，它的接口集合就变为两个接口集合的并集。

类别和构件类的区别是：类别是一个接口的集合，但不需要直接实现这些接口；构件类也是一个接口的集合，构件的开发者必须实现构件类所包含的所有接

口。同时，所有继承类别的构件类还必须实现该类别包括的所有接口。因此，也可以把构件类别看作是虚基类或超类。

类别智能指针的实现和类智能指针是基本相同的。但是它们某些方法的具体实现不同，比如它们创建构件对象的方法的实现是不同的。

（2）对创建构件对象的封装

在 MS COM 中， 是通过 CoCreateInstance 方法或者 CoCreateInstanceEx 方法创建构件对象的，关于 CoCreateInstance 和 CoCreateInstanceEx 的详细解释参见 MSDN。

- 通过接口智能指针创建构件对象的实现

通过接口智能指针创建构件对象，是通过调用接口智能指针提供 Instantiate 方法实现的，Instantiate 方法原型如下：

HRESULT Instantiate(REFCAR_CLSID rclsid , DWORD dwClsContext=CTX_DEFAULT);

HRESULT Instantiate(REFCAR_CLSID rclsid , COSERVERINFO *pServerInfo);

HRESULT Instantiate(REFCAR_CLSID rclsid , DWORD dwClsContext, COSERVERINFO *pServerInfo);

关于这三个方法的解释，可以参考“和欣”操作系统相关文档。

Instantiate 方法的实现步骤如下：

构造 CoCreateInstance 或者 CoCreateInstanceEx 方法所需的参数。

调用 CoCreateInstance 或者 CoCreateInstanceEx 方法来创建对象。

把 CoCreateInstance 或者 CoCreateInstanceEx 方法返回的接口指针赋值给成员变量。

以 CButton 构件对象实现 IButton 接口为例，使用接口智能指针 IButtonRef 来创建 CButton 构件对象，假设已经声明 IButtonRef 类型的变量 m_iButton。可以有如下几种写法：

```
m_iButton.Instantiate(CLSID_CButton);
m_iButton.Instantiate(CLSID_Cbutton, CTX_DIFF_DOMAIN);
m_iButton.Instantiate(CLSID_Cbutton, NULL);
m_iButton.Instantiate(CLSID_Cbutton, CTX_DIFF_DOMAIN,NULL);
```

关于这几行代码的解释，可以参考“和欣”操作系统相关文档。

- 通过类智能指针创建构件对象的实现

类智能指针创建构件对象的步骤和接口智能指针基本相同。有两点不同，其一，类智能指针对应于构件类，所以用类智能指针创建构件对象时不需要用户给入参数 rclsid，譬如 CLSiD_Cbutton；其二，类智能指针继承有多个接口智能指

针，创建完对象后，需要返回所有这些接口智能指针对应的接口指针。

类智能指针中提供如下几种 Instantiate 方法原型：

HRESULT Instantiate(DWORD dwClsContext=CTX_DEFAULT);

HRESULT Instantiate(COSERVERINFO *pServerInfo);

HRESULT Instantiate(DWORD dwClsContext, COSERVERINFO *pServerInfo);

关于这三个方法的解释，可以参考“和欣”操作系统相关文档。

Instantiate 方法的实现步骤如下：

构造 CoCreateInstanceEx 方法所需的参数。

调用 CoCreateInstanceEx 方法来创建对象。

把 CoCreateInstanceEx 方法返回的多个接口指针赋值给对应的成员变量。

同上例，假定已经声明了类智能指针 CButtonRef 的变量 m_cButton，那么通过类智能指针创建构件对象的代码如下：

```
m_cButton.Instantiate();
m_cButton.Instantiate(CTX_DIFF_DOMAIN);
m_cButton.Instantiate(NULL);
m_cButton.Instantiate(CTX_DIFF_DOMAIN,NULL);
```

关于这几行代码的解释，可以参考“和欣”操作系统相关文档。

- 通过类别智能指针创建构件对象的实现

类别智能指针的实现和类智能指针的实现基本相同。有两点不同，其一，利用类别智能指针来创建构件对象，被创建的构件对象是属于该类别的默认构件对象；其二，在类别智能指针 Instantiate 方法的实现中，没有调用 CoCreateInstanceEx 来创建构件对象，而是采用自己定义的方法 CoCreateCatInstanceEx 来创建构件对象。

需要强调的是，默认构件对象是可以配置的，用户可以指定某个构件对象是该类别中默认的构件对象。这就实现了动态创建不同构件对象，也就是实现了构件对象创建时的“多态”。

类别智能指针中提供的 Instantiate 方法原型如下：

HRESULT Instantiate(DWORD dwClsContext=CTX_DEFAULT);

HRESULT Instantiate(COSERVERINFO *pServerInfo);

HRESULT Instantiate(DWORD dwClsContext, COSERVERINFO *pServerInfo);

关于这三个方法的解释，可以参考“和欣”操作系统相关文档。

Instantiate 方法的实现步骤如下：

构造 CoCreateCatInstanceEx 方法所需的参数。

调用 CoCreateCatInstanceEx 方法来创建对象。

把 CoCreateCatInstanceEx 方法返回的多个接口指针赋值给对应的成员变量。

CoCreateCatInstanceEx 的原型如下：

```
HRESULT stdcall CoCreateCatInstanceEx(
/* [in]*/ REFCATID rcatid,
/* [in]*/ PUNKNOWN pUnkOuter,
/* [in]*/ DWORD dwClsContext,
/* [in]*/ UNIT uDevNo,
/* [in]*/ COSERVERINFO *pServerInfo,
/* [in]*/ ULONG cmq,
/* [in, out]*/ MULTI_QI *pResults);
```

它和 CoCreateInstanceEx 的区别是：①，第一个参数为 REFCATID rcatid，称之为类别 ID，而不是 CoCreateInstanceEx 中的 REFCLSIDrclsid；②，CoCreateCatInstanceEx 比 CoCreateInstanceEx 多了第四个参数 UINT uDevNo，称之为设备号，举个例子，计算机上有两块同样的声卡，可以用 uDevNo 来区分到底是驱动第一块声卡还是第二块声卡。

（3）CAR 对 COM 的扩展

CAR 在秉承了 COM 的核心精神的基础上，在构件的自描述封装及运行、构件类别、构件的自描述数据类型支持等等方面进行了扩展。由此实现了构件的无注册运行、支持构件对象创建的多态性、任意接口的远程化支持等特性。

对微软的 COM 进行了扩展主要体现在以下几个方面：

- 自描述数据结构

为支持构件化编程而设计的自描述数据结构：EzStr、EzByteBuf、EzStrBuf、EzArray。 EzStr 一般用来存储用户的常量字符串，它有一个定长的存储区，可以存储用户的字符串，它还保存该字符串的长度。EzByteBuf 提供存储字节的缓冲区，可以存放任何数据，EzStrBuf 中存放的是一个 EzStr 对象，EzArray 用来定义一个多维、定长、自描述数据类型的数组。(详见“和欣”文档)。

- 与创建、管理构件相关的定义与函数的实现

主要有：IID_INTERFACE_ INFO、C1assInfo、CoInitialize、CoInitializeEx。

- 构件类别(Category)与多态性

- IID_INTERFACE_ INFO

IID_INTERFACE_ INFO 是一个常量 IID，用以标识接口信息，调用任何一个接口的 QueryInterface(IID_INTERFACE_ INFO, (void**)ppv)方法时，返回的 PPV 中存放的值是这个接口的 IID。这是对 MS COM 的一个扩展。MS COM 中，

由于任何接口都继承了 IUnknown 接口，所以任何接口都可以映射到 IUnknown 接口，但是映射完后，进行反映射则就不行了。利用 IID_ INTERFACE_ INFO，我们就可以进行反映射。下面通过一个例子来说明这个扩展的意义。

在应用中，经常会用 EzVar (EzVar 的详细解释参见“和欣”文档)来远程传递一个接口指针。远程传递一个接口指针的主要步骤如下：

发送方应用程序把接口指针传递给操作系统。

操作系统列集接口指针的虚表，然后传送给接收方的操作系统。

接收方的操作系统散集接口指针的虚表。

把接口指针传递给接收方应用程序。

- C1assInfo (Class Information)

在 MS COM 中用户编写 ODL，然后用 MS 的工具 MIDL 或者 MkTypLib 产生 Type Library 信息，该信息是描述构件的元信息。

在 CAR 中，开发了自己的 CDL 语言，关于 CDL 语言，参照“和欣“中的“CDL 语言”部分，并且开发了自己的描述构件的元信息，称之为“C1assInfo"。同时开发了自己的编译器工具，使用该工具可以根据 CDL 文件得到 ClassInfo。

- CoInitialize 和 CoInitializeEx

在 MS COM 中，用户创建一个构件对象前，必须指定这个构件对象运行在什么类别的套间(apartment)中，有两种套间类别：single-threadedapartment 和 multithread apartment，简称为 STA 和 MTA。MS COM 通过调用 Colnitialize 或者 CoInitializeEx 来指定这个构件对象运行在什么类别的套间中。关于 apartment 的解释参见 MSDN。

可以看出，在 MS COM 中，构件对象运行在什么类别的套间中是由使用构件者决定的，这会造成如下两个问题：

构件对象编写者指定构件对象只能在一种套间类别中运行，但如果构件对象使用者指定构件对象在另一种套间类别中运行，会造成运行失败。

假设构件对象编写者指定构件对象可以在任何一种套间类别中运行。如下情况使用构件对象仍然会造成构件运行失败。譬如：

有一个 win32 应用程序，该应用程序中要使用构件 A，根据 MS COM 要求，在使用构件 A 前，应用程序必须调用 CoInitialize 或者 CoInitializeEx 来指定构件 A 运行在那个类别的套间中，假设为 STA。但如果在构件 A 的实现中，调用了一个 win32 API。而该 API 的实现中又使用了构件 A，根据 MS COM 规则，必须在使用构件 A 之前，指定构件 A 运行在那个类的套间中，假定为 MTA 。从以上描述中，可以看出 win32 应用程序中指定构件 A 运行在 STA, win32 API 指定构件 A 运行在 MTA。根据 MS COM 的规则，第二次指定失败，也就是 win32

API 使用构件 A 失败。

从以上分析可以看出，MS COM 中由构件使用者来指定构件对象运行在什么类别的套间中是不妥当的。

在“和欣”的 CAR 中，不使用套间，在使用构件前，不能也不需要调用 CoInitialize 和 ColnitializeEx 来指定构件对象运行的套间类别。

（4）构件类别

- 构件类别的原理简述

在 COM 中，所有的构件都是以类标识(CLSID)作为构件类的唯一标识〔以下简称 CLSID〕，每一个 CLSID 就对应了一个构件实现。

COM 规范声称，把虚接口抽象出来，就是实现了二进制的多态。但实际情况并不完全如此：构件客户端在使用构件时，还是要指定构件服务器的 CLSID 来创建构件对象，指定 CLSID 实际上就是指定了构件的实现。

所以我们认为，把接口抽象出来只是实现了构件方法调用的多态性，并未实现构件创建的多态性。而一个构件的使用总是要经过创建、调用、消亡这三个过程的，只有实现了构件对象创建的多态性，才算得上是完整的实现了构件使用的多态。

为了达到上述目的，我们引入了构件类别（category）的概念，一个构件类别包含了一组公用接口，某个构件类(class)要属于这个类别就必须继承这个类别并实现这个类别包含的所有接口。构件类别本身没有实现代码。

因此，构件类别也可以被称为抽象类或超类。

继承于构件类别(category)的所有构件类(class)都要实现构件类别中包含的所有接口，这也正是构件类别作为类别中所有构件类的公共入口点的技术基础。

- 构件类别中的缺省类标识(CLSID)的设定方法

构件类别中的缺省类标识的来源有三种：

使用 zmake 编译构件程序时，编译工具会自动对构件类别及其所属的构件类进行注册，最后一个注册到构件类别中的 CLSID 内该构件类别的缺省 CLSID。

使用 CAR 工具注册构件类别中的缺省类标识。

在没有注册文件的情况下，就使用定义构件类别的那个 DLL 文件。如果该文件中有一个以上的构件类属于该类别，将使用第一个属于该类别的构件类作为缺省类，否则创建对象过程将失败。

4.5 CAR 自描述模块封装及运行技术

（1）技术背景

在 COM (Component Object Model)技术中，强调构件的自描述，强调接口数

据类型的自描述，以便于从二进制级上把接口与实现分离，并达到接口可以跨地址空间(或者说可以远程化)的目的。

构件及接口数据的自描述是 COM 的理论基础及立足点之一。但在一些广泛采用的 COM 的具体设计和实现上，并未完全贯彻这种思想。比如微软的 MS COM 就是一个例子，其不足之处主要体现在以下几点：

在 MS COM 中，构件的一些相关运行信息都存放在系统的全局数据库——注册表中，构件在能够正确运行之前，必须进行注册。而构件的相关运行信息本身就应该是构件自描述的内容之一。

MS COM 对构件导出接口的描述方法之一是使用类型库(Type Library)元数据(Meta Data，用于描述构件信息的数据)，类型库本身是跟构件的 DLL (Dynamic Link Library)文件打包在一起的。但类型库信息却不是由构件自身来解释，而是靠系统程序 OLE32. DLL 来提取和解释，这也不符合构件的自描述思想。

大多数情况下，一个构件会使用到另一些构件的某种功能，也就是说构件之间存在相互的依存关系。MS COM 中，构件只有关于自身接口(或者说功能)的自描述，而缺少对构件依赖关系的自描述。在网络计算时代的今天，正确的构件依赖关系是构件滚动运行、动态升级的基础。

正是意识到 MS COM 中存在的种种问题，CAR 在继承了 COM 自描述思想的同时，针对上述问题，对 COM 的具体设计和实现进行了扩展和改进。

（2）基本思想

围绕着构件的自描述封装和运行，CAR 采用了如下的措施对 MS COM 进行改进和扩展：

CAR 把类信息(Class Info)作为描述构件的元数据，类信息所起的作用与 MS COM 的类型库相似，类信息由 CDL 文件编译而来，是 CDL 文件的二进制表述。与 MS COM 不同的是，MS COM 使用系统程序 OLE32.DLL 来取出并解释类型库信息；在 CAR 中，可以使用一个特殊的 CLSID 从构件中取出元数据信息，构件元数据的解释不依赖于其它的 DLL 文件。

在 CAR 的构件封装中，除了构件本身的类信息封装在构件内外，还对构件的依赖关系进行了封装。即把一个构件对其它构件的依赖关系也作为构件的元数据封装在构件中，我们把这种元数据称为构件的导入信息〔ImportInfo〕。

CAR 构件通过对 ClassInfo 和 ImportInfo 的封装，可以实现构件的无注册运行。并可以支持构件的动态升级和自滚动运行。

构件的类信息(ClassInfo)

构件类信息的生成：

构件的 ClassInfo 是 CDL (Component Definition Language)文件的编译结果，也就是 CDL 文件的符号化表示。在 MS COM 中，与 ClassInfo 相对应的是类型

库(TypeLib)信息，TypeLib 是 ODL (Object Definition Language)文件的编译结果，是 ODL 文件的符号化表示。

由于微软公司并未公开其 TypeLib 文件的格式，所以和欣操作系统采用自己定义的 ClassInfo 文件格式。

ClassInfo 被作为构件程序的元数据信息，用于描述构件导出的接口及方法列表。同时 ClassInfo 也是自动生成构件源程序的基础。

在最后编译构件程序时，C1assInfo 会作为资源数据被打包到构件 DLL 文件的资源段中。

构件类信息的获取

在根据 ClassInfo 自动生成的 C/C++构件源程序中，包括了标准构件导出函数 DllGetClassObject()的实现代码，DllGetClassObject 完成了标准的 COM 语义：取构件类对象(类对象：COM 术语，指用于创建构件对象的对象，也叫类厂)。在 CAR 对 DllGetClassObject 的自动实现中，支持使用一个特殊的 CLSID——CLSID_ ClassInfo 来获取构件的元数据，而不是类对象。

自动生成的 DllGetClassObject 的实现代码示例如下：

```
STDAPI D1lGetClassObject(REFCLSID c1sid, REFIID riid, void **ppv)
{
   if (clsid==CLSID_ ClassInfo)
   {
//取构件类信息，pCI_ClassInfo_指向构件类信息的起始地址
//参数 riid 被忽略，返回的地址存放在 ppv 中
*ppv=(void*)pCIClassInfo_;
return S_OK;
   }
   if (clsid==(REFCLSID)CLSID_ Csample)
   {
   //取构件 CSample 的类对象
      return    _g_ CSampleCF.QueryInterface(riid, ppv);
      }
      ……
      return CLASS_E_CLASSNOTAVAILABLE;
}
```

如示例：当输入参数指定的 CLSID 为 CLSID_ClassInfo 时，DllGetClassObject 返回的不再是某个类的类对象，而是构件的类信息的起始地址。

这样，当系统或构件客户需要使用 Classlnfo 元数据时，就可以用 CLSID_ Classlnfo 作为输入参数，调用构件的 DllGetClassObject 来取得。

获取构件元数据的示例代码如下：

```
IModuleRef.iModRef;
HRESULT (_stdcall * pDllGetClassObject) (REFCLSID, REFIID ,void**);
……
//装载构件 DLL 模块
hr=iProcessRef.LoadModule(ezsDllName, NULL, &iModRef);
if (FAILED(hr))
{
return hr;
}
//取构件 DllGetClassObject 函数的首址
hr=iModRef.GetProcAddress(
EZCSTR( "DllGetClassObject"),
0,
(ADDRESS)pDllGetClassObject);
//调用构件 DllGetClassObject 方法取构件的 CIassInfo
if (SUCCEEDED(hr))
{
hr=(*pDllGetClassObject)(
CLSID_ Classlnfo, riid, (void **)&pCIClassInfo_ );
}
……
```

从以上的代码可以看出，获取构件 ClassInfo 的过程不需要其它的系统程序的参与，直接调用构件 DLL 模块的 DllGetClassObject 就可以了。

（3）构件的导入信息(ImportInfo)

- 构件导入信息的必要性

在 CAR 构件的 Cl assInfo 或 COM 构件的 TypeLib 中，描述了构件自身支持的接口、方法等导出信息，这些接口方法可以被构件客户调用。另一方面，一个构件运行通常还依赖于其它构件，也就是说，构件程序本身也可能是其它构件程序的客户。

在 MS COM 中，构件程序没有关于自身的导入信息的描述，构件依赖关系是建立在全局的注册表之上，而注册表信息的增加和修改是通过构件的安装程序来完成的。这种做法的弊病是：

所有构件的运行都依赖于注册表，系统在能够运行前必须进行完整的安装，并且经过多个软件的反复安装、卸载后，系统注册表会变成非常庞大并且难以维护。

由于注册表是一个全局数据库，访问权限不容易控制，注册表容易成为病毒、黑客软件的入侵点。

构件间的依赖关系建立在完整安装的基础上，如：A 构件依赖于 B，当 B 没有安装到系统中时，由于系统没有任何关于 B 构件的信息，这时 A 构件也就不能正确运行了。

- 构件导入信息的构成

为生成构件导入信息，CAR 要求在 CDL 中定义的构件必须指定它的 URN(Uniform Resource Name), URN 是一个字符串，这个字符串类似于 URL(Uniform Resource Locator)，是关于构件 DLL 文件的网络定位信息，在 CDL 中对 URN 的定义示例如下：

```
[
version (1. 0)，
uuid(e363b985-8a3a-40a6-b88c-b2e10274fe54)，
urn(http://www.koretide.com/CAA/samples/hello.dll)
]
component Hello {
……
}
```

通过这种定义，URN 起到了唯一标识一个构件程序的作用，构件程序里的构件类由 CLSID 来标识。

为了对构件文件进行快速定位，CAR 对 COM 标准的 CLSID 进行了扩展，引入了 CAR_CLSID， CAR_CLSID 除了包含构件类的 CLSID 外，还包括构件程序的 URN。其 C/C++定义如下：

```
typedef struct CAR_CLSID
{
CLSID clsid;
WCHAR *urn.
} CAR_CLSID;
```

由于在 COM 的标准 API 及接口中，对 CLSID 的参数传递都以引用或指针方式传递，CAR 把 CLSID 放在 CAR_ CLSID 前，就保证了对标准 COM 的兼容性。也就是说，CAR_ CLSID 既可以在 CAR 平台上使用，也可以在 MS COM 平

台上使用。

除了 URN 外，对构件导入信息的描述还包括构件的版本号、C1assInfo 的版本号，最后修改日期，更新周期等，这些信息在构件升级及错误恢复时使用。

- 构件导入信息的生成

CAR 通过对构件 C/C++源程序的预处理生成构件导入信息。当构件客户使用一个构件时，必须通过#import 预处理语句导入构件的定义。如：

```
//client of component    -hello. dll
//
#include <stdio.h>
#import <hello.dll>
int _cdecl main()
{
……
}
```

#import 语句的预处理程序是 CAR 工具 mkimport. exe，在 CAR 编译环境里，调用 C/C++编译器编译 C/C++程序之前会先调用 mkimport.exe 对 C/C++程序进行预处理。

在经过对 C/C++源程序的预处理后，新生成了三种文件：

临时的 C/C++源程序文件。与最初源程序的区别是把#import<xxx. dll>语句替换成了#include <xxx.h>。因为 C/C++编译器并不能正确处理#import <xxx.dll>语句，生成的临时源程序是提供给 C/C++编译器真正编译的。

对源程序中的每一个#import <xxx. dll>语句，预处理器都生成了一个关于 xxx. dll 构件定义的头文件 xxx. h。头文件中声明了构件的 CLSID、URN、接口等等信息。

当前程序的导入信息文件。导入信息文件中记录了当前程序所使用到的所有构件的导入信息记录(URN、版本号、更新周期等等)。

由此，构件的导入信息分成了两部分，第一部分是 URN，基于效率上的考虑，URN 以 CAR_CLSID 的形式直接声明在生成的构件头文件中，每个 CLSID 的后面都跟了一个 URN 字符串，如：前面例举的构件类 CHello 的 CLSID 的声明如下：

```
const CAR_CLSID CLSID_ CHello=\
{{Ox3Dl9BC4C,OxB2C7, Ox4EA5, {0x84, 0x09, 0x63, OxDB, 0x93, OxOA,
0xD1,OxB7} },\
L”http://www.koretide.com/CAR/samples/hello.dll”};
```

导入信息的第二部分是关于构件的版本、最后修改日期、ClassInfo 版本、更新周期等信息。由于这些信息并不是经常使用，它们被写入到一个导入信息文件，并且同一可执行程序的所有 C/C++源程序的导入信息，在链接之前会被合并成一个。最后在进行链接时，程序的导入信息会作为资源数据链接到 DLL 或 EXE 文件的资源段中。

- 使用构件导入信息的 URN

构件客户运行时使用的 CLSID 实际是 CAR_CLSID，如创建 CHello 构件对象的程序：

```
……
hr=CoCreateInstance((REFCLSID)CLSID_CHello,\
NULL, CTX_SAME DOMAIN, IID_ IHello, &CHello);
if (FAILED(hr)){
……
}
……
```

在 CoCreateInstance 中传入的 CLSID_CHello 是一个 CAR_ CLSID，系统可以自动根据 CAR_CLSID 中的 URN 信息找到并装载正确的构件程序，并根据真正的 CLSID 找到相应的构件类。

通过 URN 机制的引入，构件程序不需要安装、注册过程也同样能够被构件客户端使用，并且由于 URN 是在编译时链接到构件客户程序的二进制代码中，如果结合数字签名等安全机制，即使是管理员也无法修改构件的 URN，对病毒和黑客程序能起到有效的防范作用。

（4）基于构件导入信息的构件缓存机制

在“和欣’网络操作系统上，所有的构件程序都存放在系统的构件缓存目录中，目录中的构件程序以 URN 为唯一标识(不是文件名)。

当客户指定的构件程序不在系统中时，“和欣”操作系统将自动根据构件的 URN 到网络上下载构件程序到系统构件缓存目录中。

（5）基于构件导入信息的构件自滚动运行及动态升级

正是通过 URN 等构件导入信息及构件缓存机制的引入，使得只要具备基本的构件运行环境，CAR 构件或构件客户程序就可以自滚动的运行。

如：构件 A 依赖于构件 B，构件 B 依赖于 C。在某系统中最初只安装了构件 A，在构件 A 运行时，构件 A 在创建构件 B 的构件对象时，通过 CAR_ CLSID 指定了构件 B 的 URN，系统就可以自动到网络上下载构件 B 的程序。同理，在没有事先安装构件 C 的情况下，构件 B 也能够得到正确运行。

这种自滚动运行机制给了软件的使用者极大的方便，软件的使用者根本不需

要了解除了他直接使用的软件之外的任何信息。软件的开发者也不再需要费心尽力的去为一个庞大而关系复杂的软件制作安装软件。

此外，“和欣”操作系统在启动后，一个构件第一次被装载时，系统会从构件的资源段取出构件的 ClassInfo，并根据 ClassInfo 中构件的更新周期判断构件是否需要升级，如果更新周期到了，系统会自动依据构件的 URN 到网络上去下载构件的更新版本。

4.6 CAR 技术对和欣平台的意义

对于面向 WEB 服务的应用软件开发，以及开发操作系统这样的大型系统软件而言，采用 CAR 构件技术具有以下的意义：

不同软件开发商开发的具有独特功能的构件，可以确保与其他人开发的构件实现互操作。

实现在对某一个构件进行升级时不会影响到系统中的其它构件。

不同的编程语言实现的构件之间可以实现互操作。

提供一个简单、统一的编程模型，使得构件可以在进程内、跨进程甚至于跨网络运行。

CAR 构件与微软的 COM 构件二进制兼容，但是 CAR 的开发工具自动实现构件的封装，简化了构件编程的复杂性，有利于构件化编程技术的推广普及。

CAR 构件技术是一个实现软件工厂化生产的先进技术，可以大大提升企业的软件开发技术水平，提高软件生产效率和软件产品质量。

软件工厂化生产需要有零件的标准，CAR 构件技术为建立软件标准提供了参考，有利于建立企业、行业的软件标准和构件库。

第 5 章 基于数字水印技术安全构件模型

5.1 安全构件模型的提出

基于数子水印技术的安全构件模型（以下简称安全构件）的提出是与“和欣”操作系统紧密相关的，借鉴了很多“和欣”的构件化概念，安全构件在“和欣”操作系统上更能够发挥其作用。

“和欣”操作系统是完全面向构件的操作系统，操作系统提供的功能模块全部基于构件技术，都是可拆卸的构件，应用系统可以按照需要剪裁组装，或在运行时动态加载必要的构件。因此基于面向构件的操作系统平台很容易开发并部署系统级的构件，以此来扩充操作系统的功能。安全构件在系统中的布局如下图所示。如图 5.1。

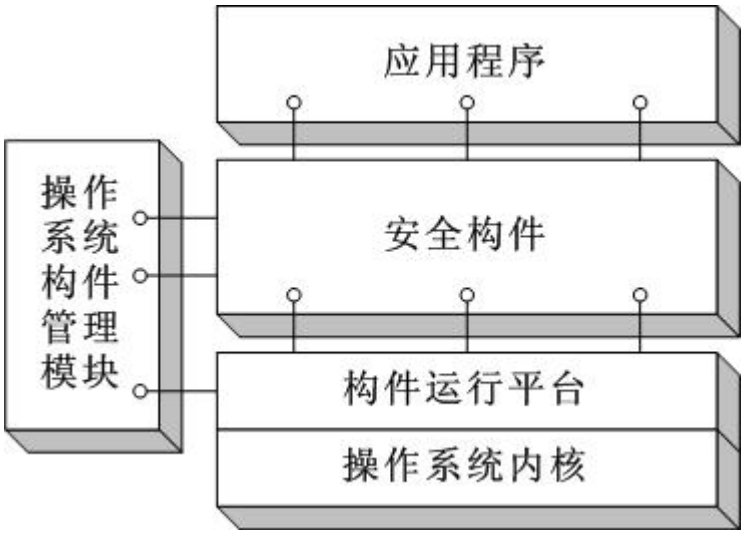


图 5.1 安全构件在系统中的布局

安全构件模型可从以下几个方面进行描述：

(1) 用 COM（component object model） 技术封装安全构件，安全构件提供两组接口，一组为系统功能接口；另一组为用户使用接口。安全构件同外部交互必须通过这些接口完成。

(2) 由操作系统提供构件的管理模块，负责控制安全构件对象的配置、初始化和启动终止等功能，并可随时检查构件的运行状态。

(3) 以独立的目标模块承载安全构件，这样的目标模块可以动态加载。

(4) 安全构件运行于独立的地址空间中，它的运行状态（例如发生运行期错误时）不会影响其它构件以及操作系统的稳定性。

5.2 安全构件模型的设计思想

“和欣”安全构件（Elastos Security Component 以下简称 ESC）的主要设计思想是：整个构件在设计上分为三层结构，分别为 Customer，Host，Provider。Customer 为客户，即所要调用安全构件的应用程序，能够使用安全构件中所提供的所有功能，也能选择其所需要的功能。Host 为中间层，一方面它可以加载所需要的 provider 和其上的接口，来得到 provider 提供的服务；另一方面它向用户提供那些它所得到的 provider 的接口，通过接口用户可以调用所要的功能。Provider 是安全构件中所有功能的具体实现，它提供了一系列的功能，通过接口提供给外部使用，这一层对于用户来说是透明的。这样设计的目的是为了让用户可以选择所要的功能，可以动态的加载所需模块，为嵌入式系统节省资源，同时也符合构件化的思想。如图 5.2。

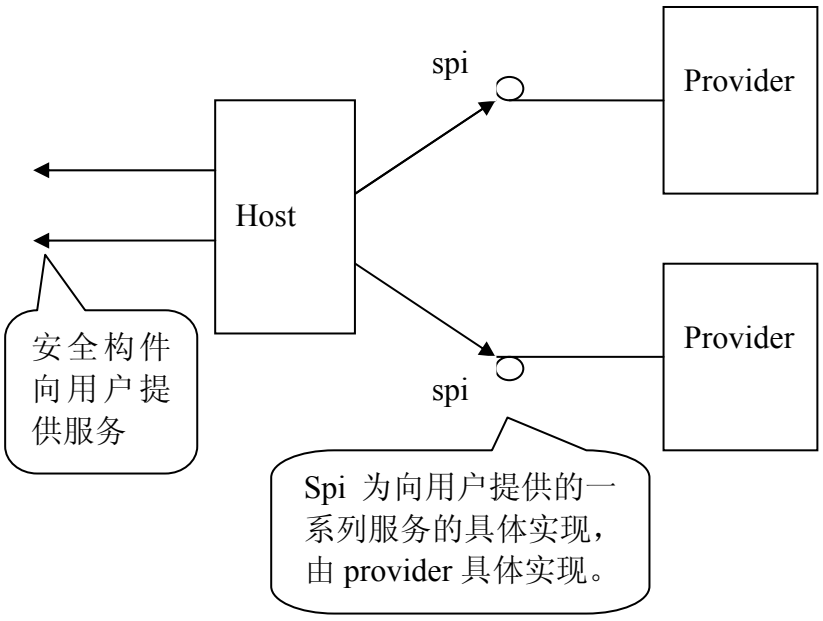


图 5.2 安全构件的三层结构

5.3 安全构件模型分析

“和欣”操作系统的安全构件，是一个用 COM 构件技术编写的，提供一系列安全服务的，可供“和欣”操作系统使用的安全构件。通过实现安全服务，包括加密、解密、验证等等，增强了操作系统的安全性，使得操作系统更加完善。

（1）下面为安全构件内部结构图。如图 5.3。

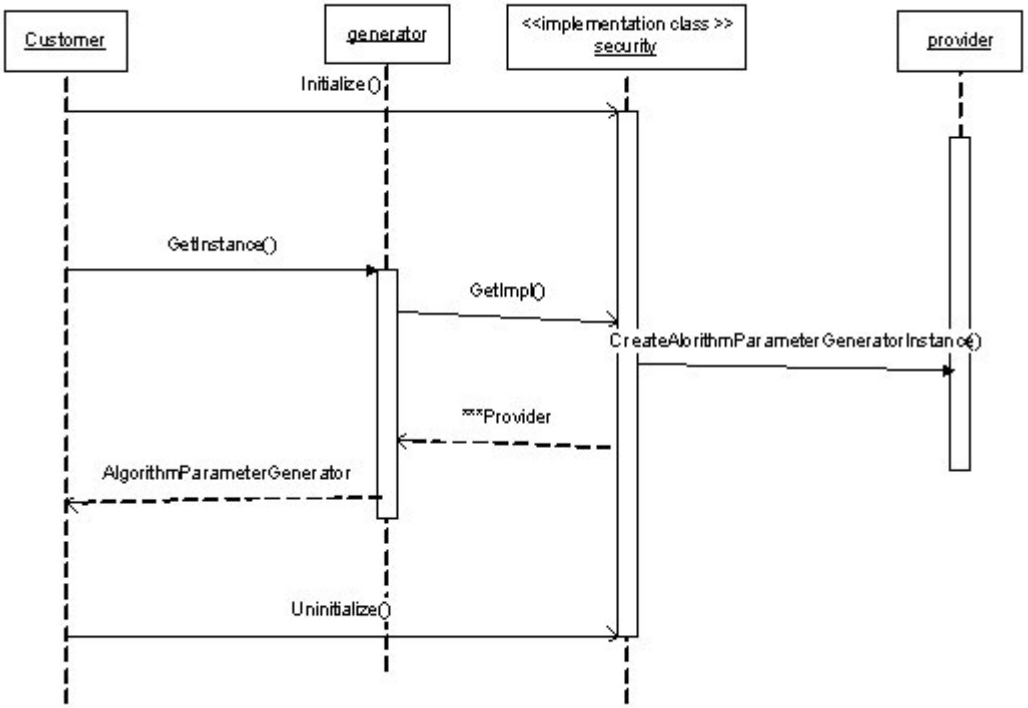


图 5.3 安全构件内部结构图

（2）各个主要类之间的关系图如图 5.4。

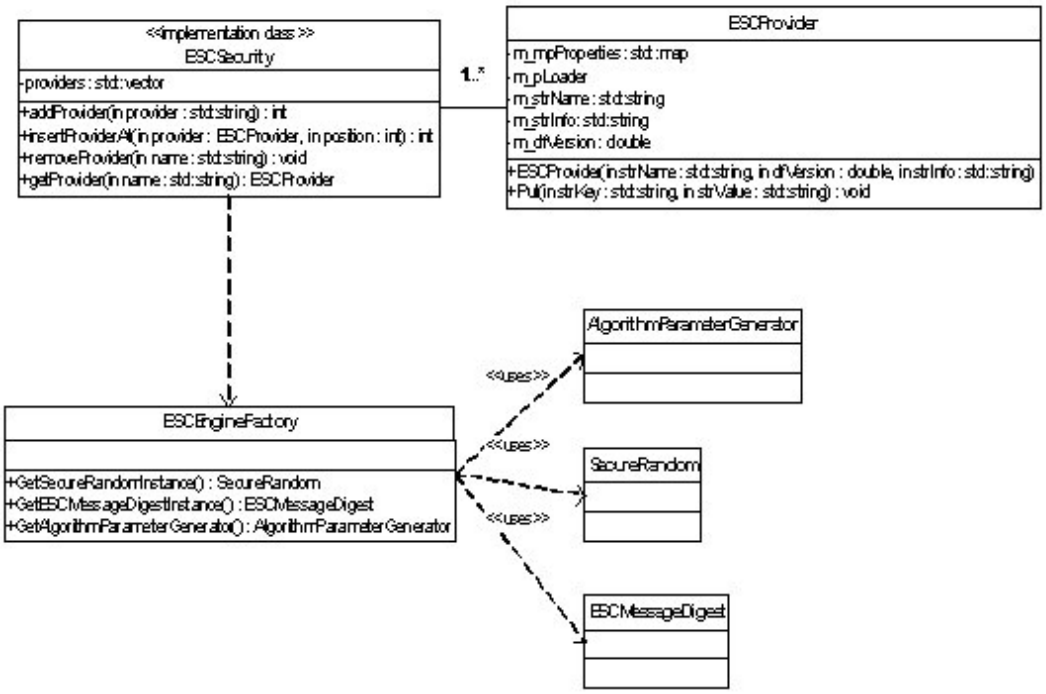


图 5.4 各个主要类之间的关系图

（3）提取主要类

- Engine 类

- Provider 类
- Security 类
- 各个算法类

类	主要功能
Engine 类	Security 类通过它来加载 Provider 类和各种安全算法。
Provider 类	所有安全算法的一个集合，通过它可以将特定的安全算法同具体的操作联系起来。
Security 类	建立了一个 Provider 类的清单，通过它可以知道 Provider 提供了哪里的安全算法。
各个算法类	实现了各个安全算法，一个算法类可能有一个或多个安全算法，和一系列的操作。被 Provider 类所装载。

从以上的分析可以看出，安全构件大致可分为三层：Customer 层，Host 层，Provider 层。

Customer 层：即用户层，使用安全构件的所有功能。可动态加载 Host，也就是上文提到的 Security 类。通过 Security 类来使用 Provider 类和各个算法类提供的功能和服务。

Host 层：这层分为两块，一块去 Customer 连接，一块去 Provider 连接。起关键作用的是 Security 类，它通过 Engine 类动态或者静态加载一个或多个 Provider，同时得到各个安全算法所提供的功能；另一方面，它将得到的功能提供给 Customer 来使用。

Provider 层：各个安全算法的集合，一个 Provider 存在多个接口，每个接口规范了一组操作，用户只有按照规范才能使用 Provider 提供的操作，从而实现安全算法。Provider 对于用户来说是透明，用户只有通过 Host 层才能使用到 Provider 提供的功能。以下是安全构件 ESC 的三层之间的关系图。如图 5.5。

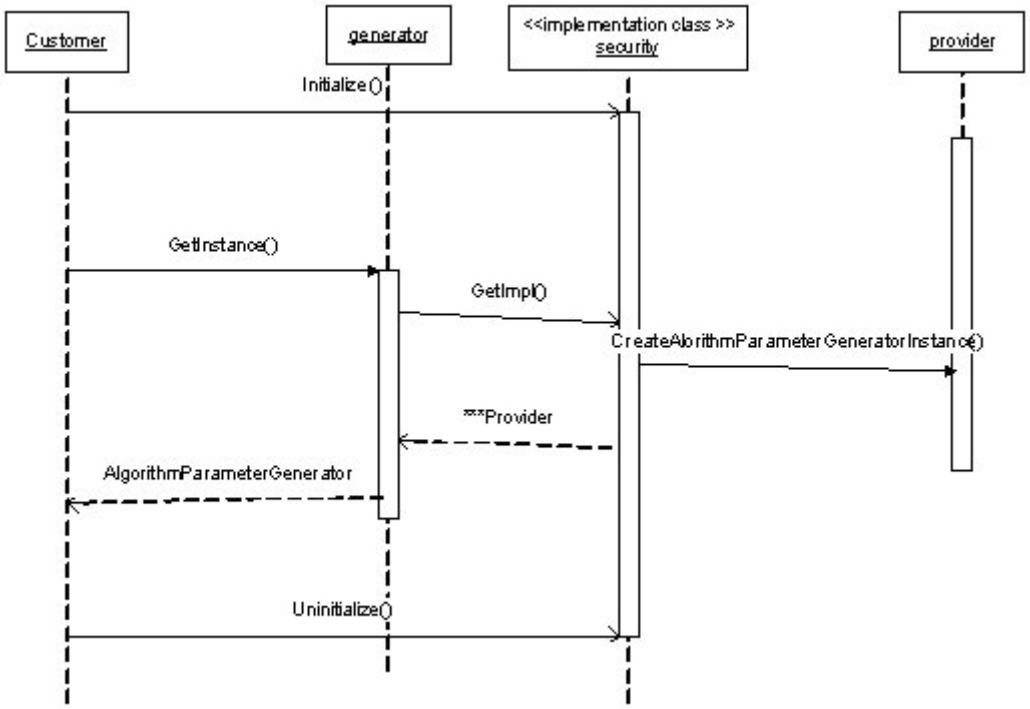


图 5.5 安全构件 ESC 的三层之间的关系图

5.4 安全构件模型的特点

将构件技术引入到安全构件模型的设计和实现中可以带来很多好处。首先，构件化安全构件模型能够充分地利用构件技术的高度模块化特征，实现二进制代码级的封装、继承与多态，既保证了代码的保密性又便于将来扩展功能；其次，安全构件与其他系统构件一样，由操作系统构件管理模块统一管理，从而可以做到，不用更改操作系统内核就可以方便地实现系统功能的扩展；更重要的是安全构件与外部应用程序以及操作系统的交互完全通过接口进行，保证了构件的实现细节不会影响到使用构件的程序，必要时系统可以根据需要动态地置换和升级安全构件。

第 6 章 安全构件的设计与实现

6.1 安全构件外部接口设计

如第 5 章种的图 5.1。安全构件对象的生命周期由操作系统提供构件的管理模块负责控制与管理。用 COM（component object model） 技术封装安全构件，安全构件提供两组接口，一组为系统功能接口；另一组为用户使用接口。安全构件同外部交互必须通过这些接口完成。

```
// ISecurityComponentUser 接口的全球唯一标识符
GUID IID_ ISecurityComponentUser;
// 安全构件的用户接口，提供对程序进行添加删除安全控制的方法
interface ISecurityComponentUser : IUnknown
{
HRESULT CreateSecurityComponent(
[in] String SecurityComponentInfo,
[out] SecurityComponent wmSecurityComponent);
HRESULT InsertSecurityComponent (
[in] SecurityComponent wmSecurityComponent,
[out] String szFileName);
HRESULT DeleteSecurityComponent (
[in] SecurityComponent wmSecurityComponent,
[in] String szFileName);
……
}
// ISecurityComponentManager 接口的全球唯一标识符
GUID IID_ ISecurityComponentManager;
// 安全构件的系统接口，提供检测程序并验证其合法性的方法
interface ISecurityComponentManager : IUnknown
{
HRESULT GetSecurityComponent(
[in] String szFileName,
[out] SecurityComponent wmSecurityComponent);
HRESULT VerifySecurityComponent (
```

[in] SecurityComponent wmSecurityComponent,

[out] bool bResult);

……

}

这里提出的安全构件的实现主要是从逻辑层面上说明构件化的安全模型，其中给出的接口设计符合实际需要。ISecurityComponentUser 接口和 ISecurityComponentManager 接口分别对应于用户接口和系统接口。

6.2 非对成加密算法模块

6.2.1 模块概述

非对称加密算法模块是安全构件 ESC 中一个非常重要的组成部分，提供使用非对称加密算法进行的加密和数字签名等功能。在此模块中使用的非对称加密算法为 RSA 算法，采用公开密钥加密标准 PKCS（Public-Key Cryptography Standard），提供了公钥和私钥的建立，以及加密和解密，签名和验证等等的操作。

安全构件中的非对称加密算法模块主要的用例模型可以归纳为以下几种：

（1）加密过程

用户可以对信息进行加密操作，通过得到公用密钥，将要加密的明文通过 RSA 算法加密成密文。

（2）解密过程

用户可以对信息进行解密操作，通过得到私人密钥，将要解密的密文通过 RSA 算法解密成明文。

（3）签名过程

用户可以对信息进行数字签名操作，签名者通过 RSA 算法得到一个私人密钥，用这个密钥对一段信息进行数字签名操作。

（4）验证签名过程

用户可以对有数字签名的信息进行验证，以检验签名的合法性，验证者通过 RSA 算法得到签名者的一个公共密钥，用此密钥来验证签名的合法性。

6.2.2 详细设计

根据以上的功能划分，可以提取出以下几个主要的类：

- RSA 公钥类
- RSA 私钥类
- RSA 加密类

- RSA 解密类
- RSA 签名类
- RSA 验证类

以上的类和一些接口组成了 RSA 算法的核心，RSA 所有的功能都是由这些类和接口共同完成的。以下是 RSA 算法实现过程中几个主要的类和接口的相互关系图。如图 6.1，6.2，6.3，6.4。

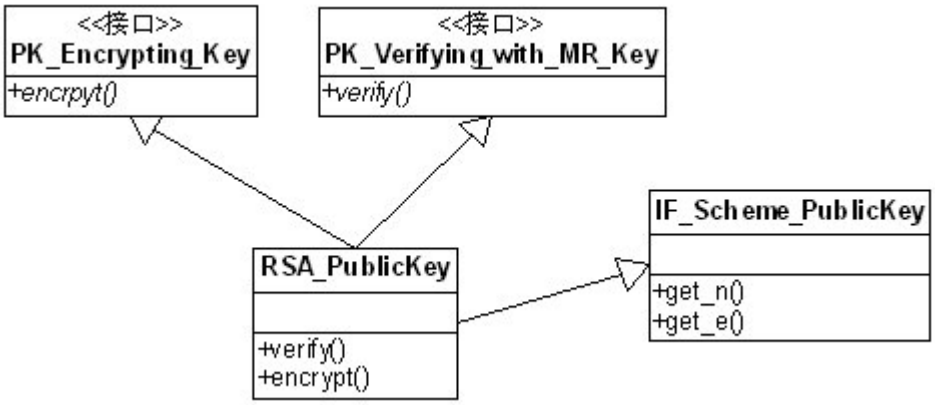


图 6.1 类和接口的相互关系图 1

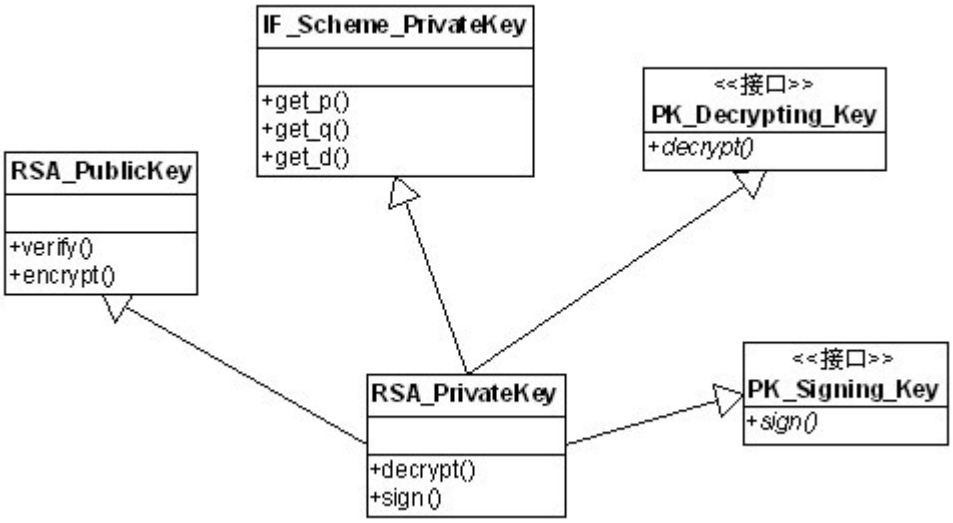


图 6.1 类和接口的相互关系图 2

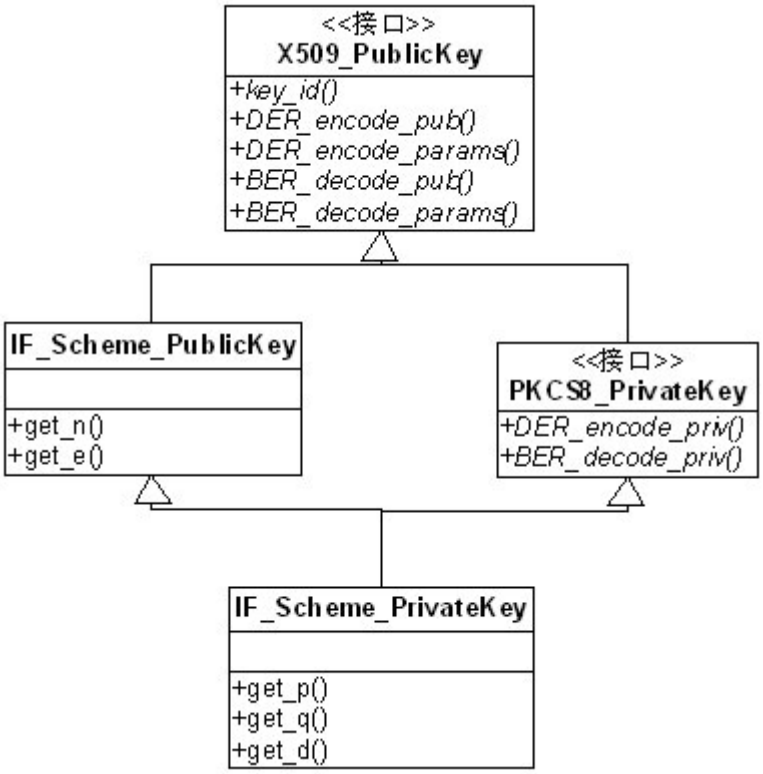


图 6.1 类和接口的相互关系图 3

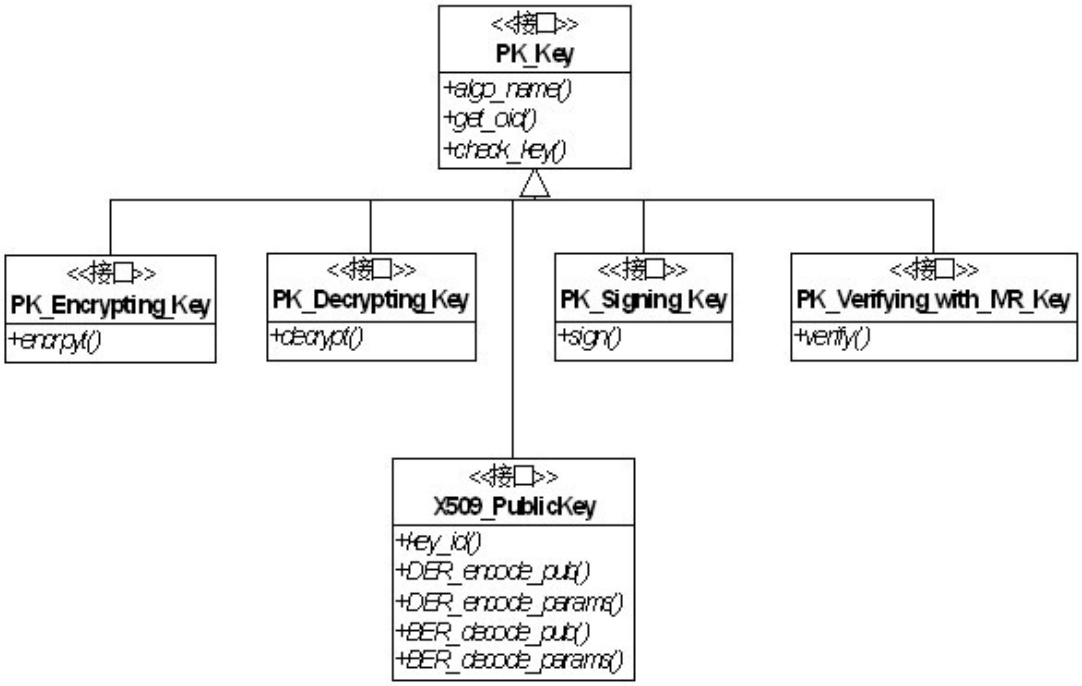


图 6.1 类和接口的相互关系图 4

6.2.3 类的具体描述

通过上述的分析和功能划分，在实现的过程中主要用到以下几个类和接口，

下面是对这些类和接口的具体分析和描述：

（1）RSA_PublicKey

描述

负责 RSA 中的公钥管理，产生公钥（n，e），并可用公钥来进行加密和验证的操作。

主要方法

RSA_PublicKey 方法

返回类型	构造函数（无返回）
参 数	BigInt& mod，BigInt& exp
功能说明	用 mod 和 exp 来初始化公钥(n,e)。

Encrypt 方法

返回类型	SecureVector<byte>
参数	byte in[]，u32bit len
功能说明	对信息进行加密，信息来源为 in，长度为 len。

Verify 方法

返回类型	SecureVector<byte>
参数	byte in[]，u32bit len
功能说明	对信息进行验证，信息来源为 in，长度为 len。

（2）RSA_PrivateKey

描述

负责 RSA 中的私钥管理，产生私钥（n，d），并可用私钥来进行解密和签名的操作。

主要方法

RSA_PrivateKey 方法

返回类型	构造函数（无返回）
参数	u32bit bits， u32bit exp
功能说明	用 bits 和 exp 来初始化私钥(n,d)。

RSA_PrivateKey 方法

返回类型	构造函数（无返回）

参数	BigInt& prime1, BigInt& prime2,BigInt& exp, BigInt& d_exp, BigInt& mod
功能说明	用 prime1,prime2,exp,d_exp,mod 来初始化私钥(n,d)。

Decrypt 方法

返回类型	SecureVector<byte>
参数	byte in[]，u32bit len
功能说明	对信息进行解密，信息来源为 in，长度为 len。

Sign 方法

返回类型	SecureVector<byte>
参数	byte in[]，u32bit len
功能说明	对信息进行签名，信息来源为 in，长度为 len。

Check_key 方法

返回类型	bool
参数	bool strong
功能说明	检验私钥的合法性。

（3）PK_Encrypting_Key

描述

这是一个接口，在这个接口中定义了加密函数，RSA_PublicKey 继承了这个接口，这个接口提供给外部的是加密算法，这个加密操作由 RSA_PublicKey 类加以实现。

主要方法

Encrypt 方法

返回类型	SecureVector<byte>
参数	byte[], u32bit
功能说明	提供加密操作接口。

（4）PK_Decrypting_Key

描述

这是一个接口，在这个接口中定义了解密函数，RSA_PrivateKey 继承了这个接口，这个接口提供给外部的是解密算法，这个解密操作由 RSA_PrivateKey

类加以实现。

主要方法

Decrypt 方法

返回类型	SecureVector<byte>
参数	byte[], u32bit
功能说明	提供解密操作接口。

（5）PK_Signing_Key

描述

这是一个接口，在这个接口中定义了签名函数，RSA_PrivateKey 继承了这个接口，这个接口提供给外部的是签名算法，这个签名操作由 RSA_PrivateKey 类加以实现。

主要方法

Sign 方法

返回类型	SecureVector<byte>
参数	byte[], u32bit
功能说明	提供签名操作接口。

（6）PK_Verifying_with_MR_Key

描述

这是一个接口，在这个接口中定义了验证函数，RSA_PublicKey 继承了这个接口，这个接口提供给外部的是验证算法，这个验证操作由 RSA_PublicKey 类加以实现。

主要方法

Verify 方法

返回类型	SecureVector<byte>
参数	byte[], u32bit
功能说明	提供验证操作接口。

（7）IF_Scheme_PublicKey

描述

这个类的主要作用是将公钥所用到的数 n，e 编码。

主要方法

DER_encode_pub 方法

返回类型	MemoryVector<byte>
参数	无

功能说明	用 DER 编码规则将 n，e 编码。

BER_decode_pub 方法

返回类型	void
参数	DataSource&
功能说明	用 BER 编码规则将 n，e 编码。

get_n 方法

返回类型	BigInt&
参数	无
功能说明	得到 n。

get_e 方法

返回类型	BigInt&
参数	无
功能说明	得到 e。

Check_key 方法

返回类型	bool
参数	bool
功能说明	检查 n 的合法性。

（8）IF_Scheme_PrivateKey

描述

这个类的主要作用是将私钥所用到的数 n，e，d，p，q，d1，d2，c 编码。

主要方法

DER_encode_priv 方法

返回类型	SecureVector<byte>
参数	无
功能说明	用 DER 编码规则将 n，e，d，p，q，d1，d2，c 编码。

BER_decode_priv 方法

返回类型	void
参数	DataSource&

功能说明	用 BER 编码规则将 n，e，d，p，q，d1，d2，c 编码。

get_p 方法

返回类型	BigInt&
参数	无
功能说明	得到 p。

get_q 方法

返回类型	BigInt&
参数	无
功能说明	得到 q。

get_d 方法

返回类型	BigInt&
参数	无
功能说明	得到 d。

Check_key 方法

返回类型	bool
参数	bool strong
功能说明	检查 n，e，d，p，q，d1，d2，c 的合法性。

6.2.4 随机数模块设计

非对称加密算法模块中使用的主要算法为 RSA 加密算法，该算法的核心内容就是两个大素数，所以要实现 RSA 算法，大素数是至关重要的。由此可见，在非对称加密模块中随机数的产生是 RSA 算法的前提条件。下面就具体研究和分析一下产生随机数这一模块的设计和实现。

在产生随机数模块中，运用了一个叫 Randpool 的技术，Randpool 有一个内部的状态叫做 pool，1024 个字节，所有的熵都加到这个 pool 里面，也从这里面来提取。这里的熵是当前系统的时间值。还存在一个缓冲 output，最后的的随机数输出就从这里产生。这个产生随机数的算法基于单向散列函数，这样做事为了所产生的随机数能更安全。

当每一次需要随机数的时候，先将一个计数器，一个时间值，和整个 pool

进行 hash 运算，这里采用 MD5 算法。再将 hash 函数得到的结果异或运算放入 output 缓冲中，每次产生一个 16 字节的 output。每当 8 个块产生以后（一个块就是 16 字节的 output 缓冲），就对 pool 进行一次混合操作，当有 265 个字节以上的熵被加入过以后，随机数就可以从 output 中产生了。

下面介绍一下 randpool 类中的属性和方法：

属性

ITERATIONS_BEFORE_RESEED	类型：u32bit
	描述：每次种种子的间隔。
POOL_BLOCKS	类型：u32bit
	描述：pool 中块的数量。
hash	类型：HashFunction*
	描述：用于进行 hash 函数。
pool	类型：SecureVector<byte>
	描述：内部状态，用来加入熵（entropy）。
output	类型：SecureVector<byte>
	描述：最后的输出缓冲，结果从中提取。
counter	类型：u32bit
	描述：计数器。

方法：

Randpoo 方法

返回类型	构造函数（无返回）
参数	
描述	初始化上述属性中的各个值。

Generate 方法

返回类型	void
参数	U64bit input
描述	先将一个计数器，一个时间值，和整个 pool 进行 hash 运算。再将 hash 函数得到的结果异或运算放入 output 缓冲中。每 8 个输出产生后进行一次 mix。

Mix_pool 方法

返回类型	void
参数	无
描述	将 pool 进行一次混合操作。

Randomize 方法

返回类型	void
参数	byte out[], u32bit length
描述	产生随机数，参数 out 得到一组随机数，length 是所要得到随机数的长度。

Is_Seeded 方法

返回类型	bool
参数	无
描述	判断熵是否大于 256 个字节。

Name 方法

返回类型	std::string
参数	无
描述	得到”Randpool”这个名字字符串。

Clear 方法

返回类型	void
参数	无
描述	清除 hash，output，pool，counter，entropy。

6.3 安全构件对软件水印的嵌入与抽取

软件水印是近年来出现的软件产品版权保护技术,可以用来标识作者、发行者、所有者及使用者等,并携带有版权保护信息和身份认证信息,可以鉴别出非法复制和盗用的软件产品。目前在软件版权保护方面，人们主要是通过加密的方式进行,比如:软件狗、Vbox、SoftSENTRY、SecuROM 和 SafeDISC 等。本文中的

软件水印则是另一种应用于操作系统安全策略的全新的软件加密技术，即所谓的软件水印就是把程序的版权信息和发行商身份信息嵌入到程序中，从而判断该程序的合法性。

6.3.1 软件水印

现在我们对软件水印做一个形式化的描述，同时对软件水印进行科学严谨的分类，并对涉及软件水印的术语进行说明，在此基础上来说明软件水印算法的有效性，科学性，实用性。为了能够合法地证明经过水印的程序的所有权，我们必须严格证明水印识别不具偶然性，水印嵌入具有隐秘性，同时还应具有抗攻击性。

定义 1：W 是一个软件水印的集合，P 是如下所述的一个谓词，P（w）表示：w ϵ W。

根据软件水印的特点，水印可以被嵌入到两个地方：程序的文本或当程序运行一个特定的输入后的所调用的语句集。同样，就攻击手段而言，也可以从这两方面着手。

定义 2：P 是一个程序的集合。p 则表示嵌入水印 w ϵ W 到 p ϵ p 之后所得到的程序。

dom(p)表示P所可以接受的输入序列的集合。out (p, I)表示p输入I后所产生的输出。S (p, I)表示p在处理了输入I后的所调用的语句(从应用程序的语句集(S_1, S_2 , S_3……，S_n))中抽取的，如(S_1, S_2 , S_3)。 | S (p , I)|表示输入I后运行p所调用语句集合的规模。

6.3.2 水印嵌入

水印嵌入，顾名思义，是指把信息 W 通过某种方式秘密地隐藏于需要保护的载体中，动态数据结构软件水印中已经提到 W 是一个整数，用一个比较复杂的数据结构来表示它。所以我们的水印嵌入的主要工作就是把这样的一个数据结构和所要保护的源程序融合，同时又要方便水印的抽取和尽可能抵抗现有的针对软件水印的攻击方式。下面以水印嵌入 JAVA 应用程序来说明水印是如何嵌入的。

假设安全构件的保护对象是 java 应用程序，水印嵌入的对象是 java 源程序(实际上，对于 java 字节码也是同样的原理，但软件水印工具还没实现 java 字节码水平的水印嵌入操作)。对于已经编译成 java 字节码形式的应用程序，如果要使用我们的软件水印工具，那么要先用反编译工具(这种工具比较多，如 JBuilder 中就集成有这个反编译工具)把它转化为 java 源程序的形式。下面将对水印嵌入算法作详细介绍，这个算法体现了动态嵌入的思想，即水印在运行时生成的，而不是在编译时。水印嵌入算法如下所示：

输入：

A，一个整数； //用户所要嵌入的信息

B，水印的代表形式；//用何种数据结构形式来表示整数

C，被保护的 java 源程序的包的路径；

D，被改造成结点类的类及其所属于的包；

E，发生水印的函数及其所属于的类和类所属于的包；

F，嵌入水印后的 Java 源程序输出的路径；

输出：嵌入水印的 java 源程序。

步骤：

1，根据包的路径寻找并调入所要保护的 java 源程序。

2，分析并改造结点类，即把源程序中的一个类改造成结点类。分析类的目的是寻找嵌入结点的位置，首先要对这个类的整个结构进行系统的分析。我们所分析的信息有：类所属的包，即 package；调用的包，即 import；所继承的类，即 extends 或所实现的接口，即 implements。同时为了使改造后的类不会产生语法语义错误，不违反 java 的规范，还要对这个类进行语法分析。一般而言，我们主要是把水印嵌在这个类的构造函数中。如果这个类没有构造函数，则要重新构造两个构造函数。之所以要构造两个构造函数，是因为 java 中的类如果没有构造函数，则会有其默认的构造函数，是一个无参数的构造函数。如果实例化这个类，就会调用这个构造函数，所以为了避免错误，则要先实现默认的构造函数。

3，根据所输入的水印的代表形式把一个大整数表示成一个复杂数据结构的形式——图结构。这里面的图结构的结点就是利用第二步中所改造的结点类。

4，分析并改造发生水印的函数。这里的函数可以是多个的，这样的目的是使攻击方更难以确定水印的存在性，因为到目前为止我们所改造的信息对源程序来说都是多余的，攻击方可能会通过对变量进行跟踪来发现这些无用的信息，多个函数调用是防止这种跟踪的方法之一。改造函数就是使得调用这个函数时，它的 S(P, I)包含有第三步中所述的嵌入的图结构。改造函数同样要对函数和函数所在的类进行系统的分析，分析的目的是为了防止嵌入的代码与原来的代码发生冲突，比如变量名等。

到现在为止，我们基本上完成了把水印嵌入源程序的工作。

6.3.3 水印抽取

水印抽取是整个安全的核心之一，水印抽取的结果应该是一个大整数 N，然后用户对它进行分解：N=P*Q，利用大数难以分解的特性，来证明用户的所有权。

水印嵌入和水印抽取是作为一个整体来考虑的，可以说水印抽取算法很大程度是由水印嵌入算法决定的，同时水印抽取算法的好坏对于安全构件的抗攻击程度的深浅是比较关键的。针对水印嵌入算法，我们通过检测应用程序的运行时的对象堆，即$S(p_w, I)$，来抽取水印。也就是说我们的水印算法是一种纯动态的算法，即水印识别程序的类型是$R(\Phi, S(p_w,I))$。当然水印抽取所用到的$S(p_w,I)$中的输入I必须是会调用在水印嵌入时所改造的函数，该函数会使得$S(p_w,I)$包含有前面所嵌入的图结构。

水印抽取工具所用到文本文件中的内容包括下面四项：

1， ROOT，作为$S(p_w, I)$的根，其格式是：

ROOT 地址(类型=<类型名>)；

2，CLS，即类，如果 CLS 没有基类且也没有定义静态变量，那么其格式是：

CLS 地址(name=类名，trace=路线序号)。

如果 CLS 是有基类，那么其格式还要加上：super 地址。

如果 CLS 是有静态变量的，那么就是：static 静态变量名地址。

3，OBJ，即实例化对象，其格式是：

OBJ 地址(sz=字节数，trace=路线序号，class=类型名@地址)。

此外还要加上实例化的域和地址，其格式是：域名 地址。

4， ARR，即 arrays，其格式是：

ARR 地址(sz=字节数，trace=l，nelems=7，elem type=类型名)。

为了更清楚地说明各个格式，典型的例子如下所示。我们的检测程序就是在这样的一个文本文件上进行抽取水印的操作。

ROOT 格式：ROOT 821ae8 (kind=<unknown>)

CLS 格式：CLS 8alcb60 (name java.lang.System, trace=l)

super 7fc568

static props 8a20768

OBJ 格式：OBJ gal cbc0 (sz=6, trace=l, class java.lang.String@7fc5c8)

value 8alcba0

ARR 格式：ARR gal bc80 (sz-8, trace=l, nelems=8, elem type=short)

水印抽取实际上就是重新生成所嵌入的数据结构。

整个抽取过程分成以下的三个步骤：

(1)抽取潜在的结点类，即检测潜在的 CLS。在这里可以把 java 运行时的标准类排除在外，因为在水印嵌入时，改造成结点类的类选用的是用户自建的类，而不是 java 标准函数库中的标准类。

(2)抽取潜在的结点对象，这一步是针对实例化对象，即 OBJ。抽取潜在的

结点对象也就是查看OBJ的<class=类型名@地址>中的类型名和地址是否与潜在的 CLS 的类名和地址是一样。如果是一样，那么就表示该结点对象是一个潜在的结点对象。

(3)判断潜在的边。检测潜在结点对象中的每一个域是否是连接到另一个潜在结点对象的边。OBJ 的每一个域都有它的地址，这个地址有可能是指向另一个潜在的结点。首先寻找与这个域有一样地址的 OBJ，如果 OBJ 是已经第二步中证明过是潜在的结点，那么这个域就是一条潜在的边；如果 OBJ 不是潜在的结点，那么这个域就不是一条潜在的边。通过上面三个步骤所做的工作，就已经建立的一个完整的所嵌入的图结构。根据用户所提供的图的大小(比如，结点数)，就可确定哪个是我们所要搜索的图结构。

(4)图搜索。在这一步中，对前三步中所建立的完整的所嵌入的图结构进行搜索，然后生成用户所输入的大整数。所谓搜索，就是对图的结点顺序进行确认。一般而言，这是一个子图同构问题，其算法复杂度相当于一个 NP 完全问题。但我们可根据所嵌入的图的特殊性来简化图搜索。

对于我们所举的两个构造图的方法，可以这样来搜索。对于基数-k 编码方式，在构造图的时候，设了一个首结点，这个首结点的其中一条边是指向其本身的。可以根据这样的特点来确定潜在的首结点，那么接下来图的结点顺序的确认就比较容易。

对于演化算法所构造的图，它的所有叶子结点的右子结点都是它自身。可以根据这个特点先确定一个叶子结点。然后顺着这个叶子结点对它的左结点进行循环，当循环到的结点的右子结点不是它自身时，那么这个结点就是首结点。

根据图搜索的结果，生成一个整数，就是我们的水印抽取工具所得到的结果。

6.4 安全构件对水印的验证

由安全构件从应用软件中抽取的水印证书信息，可以和有关发行商的证书信息进行对比，如果符合，则表示验证成功，可以执行，否则执行失败。而有关发行商的证书信息可以通过受信任的管理证书的权威认证机构 CA（certificate authority）中心来提供。本地操作系统也可以通过在线查询数据库或者下载部分证书库来进行离线查询。

6.5 安全构件实现操作系统安全策略

在基于安全构件的和欣操作系统安全策略中，所有期望得到运行环境的应用程序在被操作系统loader装载入内存空间之前，会被强制接受安全构件的“检

查”，安全构件负责应用程序中水印的提取与验证，检查的结果是每个应用程序都会被标注上相应的标签，3种基本标签分别是：

(1) 非法程序:在应用程序中未检测到任何水印信息或者由水印中提取到的发行商的签名证书已被系统标志为非法证书时，程序会得到“非法程序”的标签，此时程序将不会得到loader加载的机会。

(2) 安全程序: 在应用程序的水印中提取到的发行商的签名证书是受信任的证书时，程序会得到“安全程序”的标签，此时程序将会被loader加载到“安全域”中运行，在这个域中程序可以得到访问本地资源的权限。

(3) 试运行程序: 在应用程序的水印中提取到的发行商的签名证书是合法的，但操作系统的管理员不想提供过高的权限给该程序，此时程序就会得到“试运行程序”的标签，之后它将会被 loader 加载到“限制域”中运行，在这个域中程序对本地资源的访问将受到严格的限制，一旦程序企图越权访问本地资源将会被强行终止运行。如图 6.5。

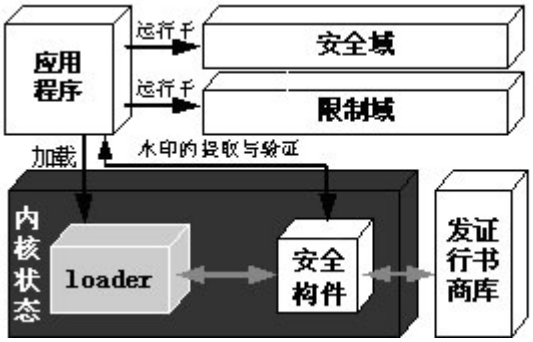


图 6.5 安全构件实现操作系统安全策略

基于安全构件实现的操作系统安全策略相对于传统的操作系统安全策略具有很多有益的特性。

(1) 是基于二进制代码的安全策略。与基于中间代码的安全策略不同，安全构件是通过对隐藏于二进制可执行代码中的水印信息的检测来实现安全策略的，因此既保证了应用程序的执行效率又实现了操作系统的安全保护，是一种两全其美的安全策略。

(2) 是基于构件技术的操作系统级的安全策略。将安全构件封装为操作系统构件，使安全构件运行于操作系统的“保护模式”下，一方面使安全构件自身的安全得到了保证，不会被恶意程序攻击，另一方面又可以方便的对安全构件进行动态地置换和升级。

(3) 是一种可定制的灵活多变的安全策略。安全构件通过与“发行商证书数据库”的交互来确定相应程序的受信任程度，并由此制定安全策略。本地操作系统可以通过维护一个该证书数据库的子集来自定义发行商的受信任程度，从而实

现了个性化的灵活多变的操作系统安全策略。

第七章 结论与展望

本文结合国家 863 课题“基于构件、中间件技术的因特网操作系统及跨操作系统的构件、中间件运行平台” ，首先对和欣操作系统及其应用前景进行了说明，然后提出了一种基于数字水印技术的安全构件模型，文中具体阐述了安全构件模型设计思想和该模型的特点，并给出了安全构件的设计方案，最后简要分析了该安全构件是如何确保操作系统的安全的。

随着无线通信技术的不断发展，尤其是 3G 通信技术的出现使得当今的手机功能日益强大，手机不仅仅再是传统的打电话工具。又因为手机操作系统的逐渐智能化，使得手机有能力承载越来越多的新功能。从产品的角度来说，“和欣”操作系统的一个重要应用就是智能手机，作为未来智能手机操作系统，“和欣”要保证数据在传输过程中的安全性以及下载软件运行的安全性。因此，本文对和欣操作系统下安全构件模型的研究，对和欣操作系统及其应用领域都具有一定的理论和实践意义。

致谢

在论文完成之际，我要衷心感谢所有关心和帮助过我的老师和同学们。

首先，我要感谢我的导师顾伟楠教授。本文是在顾老师的悉心指导和无微不至的关怀下顺利完成的。从论文的选题、资料搜集、理论分析到论文的撰写的全部过程中，都得到了顾老师的精心指导。两年半的研究生学习过程中，我得到了顾老师的极大帮助。顾老师严谨求实，一丝不苟的工作作风以及对学生的严格要求都使我受益匪浅。

其次，我要感谢“上海科泰世纪有限公司”的各位工程师和领导们，是他们为我提供了良好的试验环境，并且针对我研究过程中所遇到的困难提出了很多重要的建议，他们的工作热情和敬业精神都深深的感动了我。

最后我还要感谢同济大学基础软件工程中心的全体老师和同学们，是你们伴随着我度过了一段十分有意义的研究生生活。

在此，我向他们表示最衷心的感谢。

参考文献

[1] D. Batory and S. O'Malley, "The Design and Implementation of Hierarchical Software Systems with Reusable Components," ACM Trans. Software Eng. And Methodology, Oct. 1992

[2] Fox B.,Digital television comes down to earth. IEEE Spectrum, Volume:35, Issue: 10, Oct. 1998

[3] C. Szyperski, Component Software Beyond Object-Oriented Programming . Addison-Wesley, ADM Press, New York, 1998

[4] The Koala Component Model for Consumer Electronics Software IEEE 2000

[5] Steve Maillet, Using COM for Embedded Systems (Part II), Embedded SystemConference Session #425, 2000

[6] D. Stewart, "Software components for real time," Embedded Systems Programming, Dec 2000

[7] Evain J. P.,The multimedia home platform—an overview. E13U Technical Review, Spring 1998

[8] Michi Henning, Steve Vinoski,《Advanced CORBA® Programming with C++》Addison Wesley 1999

[9] 0verview of the CORBA Component Model,http://www. omg. org，1999

[10] M. Fayad and D.Schmidt, "Object-Oriented Application Frameworks", Comm. ACM, Oct.1997, pp. 32-38

[11] Evain J. P.,The multimedia home platform-an overview. E13U Technical Review, Spring 1998, pp. 4-10

[12] 上海科泰世纪有限公司，《和欣1.1》资料大全。2003

[13] DON BOX。《COM本质论》。中国电力出版社，2001

[14] 潘爱民。《COM原理与应用》。清华大学出版社，1999

[15] 张俊。数字电视软件平台——中间件及第三代机顶盒。中国广电技术论文，2004

[16] Chengyuan Peng, Pablo Cesar, and Petri Vuorimaa, "Intergration Of Applications Into Digital Television Environment" .the 7th International Conference on Distributed Multimedia systems,September 26-28, 2001

[17] G. Sivaraman, P. Cesar, and P. Vuorimaa, "System software for digital television applications". the IEEE International Conference on Multimedia and Expo, Tokyo, Japan, August 22-25, 2001

[18] P. Vuorimaa, "Digital television service architecture". Proceedings of IEEE International Conference on Multimedia and Expo, ICME2000, New York City,NY, USA, July 30—Aug. 2, 2000

[19] Jose C. Lopez-Ardao, Candido Lopez, Alberto Gil, Rebeca Diaz, Ana Femandez, Manuel Femandez, Andres Suarez, and Javier Munoz, "A MHP Receiver over RT-Linux for Digital TV". IEEE Region 8 International Symposium on Fideo/Image Processing and Multimedia Conmmunications,2002

[20] 张俊。中间件——数字电视软件平台。中国广电技术论文，2004

[21] 赵季伟。采用中间件平台开展数字电视交互业务的实践分析。中国广电技术论文，2004
[22] 王斌。浅谈互动电视中间件技术及其选择。中国广电技术论文，2004
[23] 陈焕经，徐朝晖。数字电视中间件技术发展浅析。中国广电技术论文，2004

个人简历　在读期间发表的学术论文与研究成果

个人简历：

胡天华，男，1979 年 8 月生。

1998 年 9 月～2002 年 7 月，同济大学计算机软件专业，获学士学位。

2003 年 9 月～2006 年 3 月，同济大学计算机科学与技术系，攻读硕士学位。

已发表论文：

［1］胡天华，顾伟楠. 基于“和欣”灵活内核的安全构件模. 计算机应用，2005 年 9 月