

# 软件加密与解密



[软件加密与解密\\_下载链接1](#)

著者:[美] Christian Collberg

出版者:人民邮电出版社

出版时间:2012-5-3

装帧:平装

isbn:9787115270757

内容简介:

对抗软件盗版、篡改和恶意逆向工程的理论、技巧和工具

近十年来，人们在软件防盗版和防篡改技术的研发上取得了重大进展。这些技术在保护软件开发人员的知识产权方面具有不可替代的作用。无论是研究人员、在校学生，还是开发人员，要了解这些技术及其能提供的安全级别和可能引发的性能开销，都可以从本书获得权威、全面的参考资料。

Christian Collberg和Jasvir

Nagra在书中详尽地介绍了相关技术，涵盖了计算机科学的各个相关领域，包括密码学、隐写术、水印、软件度量、逆向工程和编译优化等。本书通过大量的示例代码，向读者展示了代码混淆、软件水印、代码防篡改和“胎记”技术等保护算法的实现方式，并且从理论和实践两个角度探讨了这些技术的局限。

涵盖的内容

攻击者和防御者用来分析程序的各种主要方法

代码混淆技术，用于提高程序被分析和理解的难度

软件水印和指纹，用于标识软件开发者并追踪盗版

代码防篡改技术，用于检测和响应非法修改代码和数据的行为，从而保护软件

动态水印和动态混淆技术，用于阻止软件的非法复制

软件相似性分析和“胎记”算法，用于检测代码剽窃

硬件技术，用于保护软件及各类媒体免遭盗版和篡改

在分布式系统中，检测远端不可信平台上运行的软件是否被篡改

代码混淆技术在理论上的局限性

作者介绍:

作者简介:

Christian Collberg

瑞典隆德大学计算机科学博士，亚利桑那州立大学计算机科学系副教授，从事代码混淆、软件水印和“胎记”方面的基础性研究工作。他曾在新西兰奥克兰大学及中国科学院工作过。

Jasvir Nagra

专注于设计强壮的动态水印算法，曾致力于通过代码混淆和防篡改技术保护运行在远程不可信平台上的软件的完整性。目前，他任职于谷歌公司，在加利福尼亚州从事与基于编程语言的安全性有关的研究工作。

目录: 目录

第1章 什么是隐蔽软件 1

1.1 概述 1

1.2 攻击和防御 5

1.3 程序分析的方法 6

1.4 代码混淆 11

1.4.1	代码混淆的应用	13
1.4.2	混淆技术概述	17
1.4.3	被黑客们使用的代码混淆技术	21
1.5	防篡改技术	27
1.5.1	防篡改技术的应用	27
1.5.2	防篡改技术的例子	29
1.6	软件水印	30
1.6.1	软件水印的例子	32
1.6.2	攻击水印系统	34
1.7	软件相似性比对	36
1.7.1	代码剽窃	36
1.7.2	软件作者鉴别	37
1.7.3	软件“胎记”	38
1.7.4	软件“胎记”的案例	40
1.8	基于硬件的保护技术	41
1.8.1	把硬件加密锁和软件一起发售	42
1.8.2	把程序和CPU绑定在一起	43
1.8.3	确保软件在安全的环境中执行	43
1.8.4	加密可执行文件	44
1.8.5	增添物理防护	45
1.9	小结	46
1.9.1	使用软件保护技术的理由	46
1.9.2	不使用软件保护技术的理由	47
1.9.3	那我该怎么办呢	47
1.10	一些说明	48
第2章	攻击与防御的方法	49
2.1	攻击的策略	50
2.1.1	被破解对象的原型	50
2.1.2	破解者的动机	52
2.1.3	破解是如何进行的	54
2.1.4	破解者会用到的破解方法	55
2.1.5	破解者都使用哪些工具	58
2.1.6	破解者都会使用哪些技术	59
2.1.7	小结	69
2.2	防御方法	70
2.2.1	一点说明	71
2.2.2	遮掩	73
2.2.3	复制	75
2.2.4	分散与合并	78
2.2.5	重新排序	80
2.2.6	映射	81
2.2.7	指引	84
2.2.8	模仿	85
2.2.9	示形	87
2.2.10	条件—触发	88
2.2.11	运动	90
2.2.12	小结	91
2.3	结论	92
2.3.1	对攻击/防御模型有什么要求	92
2.3.2	该如何使用上述模型设计算法	93
第3章	分析程序的方法	94
3.1	静态分析	95
3.1.1	控制流分析	95
3.1.2	数据流分析	103

- 3.1.3 数据依赖分析 107
- 3.1.4 别名分析 109
- 3.1.5 切片 115
- 3.1.6 抽象解析 116
- 3.2 动态分析 118
  - 3.2.1 调试 118
  - 3.2.2 剖分 129
  - 3.2.3 trace 132
  - 3.2.4 模拟器 135
- 3.3 重构源码 137
  - 3.3.1 反汇编 139
  - 3.3.2 反编译 146
- 3.4 实用性分析 155
  - 3.4.1 编程风格度量 156
  - 3.4.2 软件复杂性度量 158
  - 3.4.3 软件可视化 159
- 3.5 小结 162
- 第4章 代码混淆 163
  - 4.1 保留语义的混淆转换 164
    - 4.1.1 算法OBFCE: 多样化转换 164
    - 4.1.2 算法OBFTP: 标识符重命名 170
    - 4.1.3 混淆的管理层 173
  - 4.2 定义 177
    - 4.2.1 可以实用的混淆转换 178
    - 4.2.2 混淆引发的开销 181
    - 4.2.3 隐蔽性 181
    - 4.2.4 其他定义 182
  - 4.3 复杂化控制流 183
    - 4.3.1 不透明表达式 183
    - 4.3.2 算法OBFWHKD: 压扁控制流 184
    - 4.3.3 使用别名 186
    - 4.3.4 算法OBFCTJbogus: 插入多余的控制流 191
    - 4.3.5 算法OBFLDK: 通过跳转函数执行无条件转移指令 195
    - 4.3.6 攻击 198
  - 4.4 不透明谓词 201
    - 4.4.1 算法OBFCTJpointer: 从指针别名中产生不透明谓词 202
    - 4.4.2 算法OBFWHKDopaque: 数组别名分析中的不透明值 204
    - 4.4.3 算法OBFCTJthread: 从并发中产生的不透明谓词 205
    - 4.4.4 攻击不透明谓词 207
  - 4.5 数据编码 211
    - 4.5.1 编码整型数 213
    - 4.5.2 混淆布尔型变量 217
    - 4.5.3 混淆常量数据 220
    - 4.5.4 混淆数组 222
  - 4.6 结构混淆 226
    - 4.6.1 算法OBFWCsig: 合并函数签名 226
    - 4.6.2 算法OBFCTJclass: 分解和合并类 229
    - 4.6.3 算法OBFDMRVSL: 摧毁高级结构 232
    - 4.6.4 算法OBFAJV: 修改指令编码方式 239
  - 4.7 小结 243
- 第5章 混淆理论 245
  - 5.1 定义 248
  - 5.2 可被证明是安全的混淆: 我们能做到吗 249
    - 5.2.1 图灵停机问题 250

- 5.2.2 算法REAA：对程序进行反混淆 252
- 5.3 可被证明是安全的混淆：有时我们能做到 254
  - 5.3.1 算法OBFLBS：混淆点函数 254
  - 5.3.2 算法OBFNS：对数据库进行混淆 261
  - 5.3.3 算法OBFPP：同态加密 263
  - 5.3.4 算法OBFCEJO：白盒DES加密 267
- 5.4 可被证明是安全的混淆：（有时是）不可能完成的任务 272
  - 5.4.1 通用混淆器 273
  - 5.4.2 混淆最简单的程序 276
  - 5.4.3 对混淆所有程序的不可能性的证明 277
  - 5.4.4 小结 278
- 5.5 可被证明为安全的混淆：这玩儿还能成吗 279
  - 5.5.1 跳出不可能性的阴霾 280
  - 5.5.2 重新审视定义：构造交互式的混淆方法 281
  - 5.5.3 重新审视定义：如果混淆不保留语义又当如何 283
- 5.6 小结 286
- 第6章 动态混淆 288
  - 6.1 定义 290
  - 6.2 代码迁徙 292
    - 6.2.1 算法OBFKMMN：替换指令 293
    - 6.2.2 算法OBFAGswap：自修改状态机 296
    - 6.2.3 算法OBFMAMDSB：动态代码合并 307
  - 6.3 加密技术 311
    - 6.3.1 算法OBFCKSP：把代码作为产生密钥的源泉 312
    - 6.3.2 算法OBFAGcrypt：结合自修改代码和加密 318
  - 6.4 小结 324
- 第7章 软件防篡改 325
  - 7.1 定义 327
    - 7.1.1 对篡改的监测 328
    - 7.1.2 对篡改的响应 331
    - 7.1.3 系统设计 332
  - 7.2 自监测 333
    - 7.2.1 算法TPCA：防护代码之网 335
    - 7.2.2 生成hash函数 338
    - 7.2.3 算法TPHMST：隐藏hash值 342
    - 7.2.4 Skype中使用的软件保护技术 349
    - 7.2.5 算法REWOS：攻击自hash算法 352
    - 7.2.6 讲评 356
  - 7.3 算法RETCJ：响应机制 357
  - 7.4 状态自检 360
    - 7.4.1 算法TPCVCPSJ：易遭忽视的hash函数 362
    - 7.4.2 算法TPJJV：重叠的指令 365
  - 7.5 远程防篡改 368
    - 7.5.1 分布式监测和响应机制 368
    - 7.5.2 解决方案 369
    - 7.5.3 算法TPZG：拆分函数 369
    - 7.5.4 算法TPSLSPDK：通过确保远程机器硬件配置来防篡改 372
    - 7.5.5 算法TPCNS：对代码进行持续的改变 375
  - 7.6 小结 376
- 第8章 软件水印 378
  - 8.1 历史和应用 378
    - 8.1.1 应用 379
    - 8.1.2 在音频中嵌入水印 382
    - 8.1.3 在图片中嵌入水印 383

- 8.1.4 在自然语言文本中嵌入水印 384
- 8.2 软件水印 387
- 8.3 定义 388
  - 8.3.1 水印的可靠性 389
  - 8.3.2 攻击 391
  - 8.3.3 水印与指纹 392
- 8.4 使用重新排序的方法嵌入水印 392
  - 8.4.1 算法WMDM: 重新排列基本块 394
  - 8.4.2 重新分配资源 396
  - 8.4.3 算法WMQP: 提高可靠性 397
- 8.5 防篡改水印 400
- 8.6 提高水印的抗干扰能力 403
- 8.7 提高隐蔽性 408
  - 8.7.1 算法WMMIMIT: 替换指令 409
  - 8.7.2 算法WMVVS: 在控制流图中嵌入水印 409
  - 8.7.3 算法WMCC: 抽象解析 416
- 8.8 用于隐写术的水印 421
- 8.9 把水印值分成几个片段 425
  - 8.9.1 把大水印分解成几个小片段 426
  - 8.9.2 相互冗余的水印片段 427
  - 8.9.3 使用稀疏编码提高水印的可靠性 432
- 8.10 图的编/解码器 432
  - 8.10.1 父指针导向树 433
  - 8.10.2 底数图 433
  - 8.10.3 排序图 434
  - 8.10.4 根延伸的平面三叉树枚举编码 434
  - 8.10.5 可归约排序图 435
- 8.11 讲评 436
  - 8.11.1 嵌入技术 437
  - 8.11.2 攻击模型 438
- 第9章 动态水印 439
  - 9.1 算法WMCT: 利用别名 443
    - 9.1.1 一个简单的例子 443
    - 9.1.2 水印识别中的问题 445
    - 9.1.3 增加数据嵌入率 447
    - 9.1.4 增加抵御攻击的抗干扰性能 452
    - 9.1.5 增加隐蔽性 455
    - 9.1.6 讲评 458
  - 9.2 算法WMNT: 利用并发 459
    - 9.2.1 嵌入水印的基础构件 462
    - 9.2.2 嵌入示例 467
    - 9.2.3 识别 469
    - 9.2.4 避免模式匹配攻击 470
    - 9.2.5 对构件进行防篡改处理 471
    - 9.2.6 讲评 473
  - 9.3 算法WMCCDKHLSpaths: 扩展执行路径 474
    - 9.3.1 水印的表示和嵌入 474
    - 9.3.2 识别 479
    - 9.3.3 讲评 480
  - 9.4 算法WMCCDKHLSbf: 防篡改的执行路径 481
    - 9.4.1 嵌入 481
    - 9.4.2 识别 484
    - 9.4.3 对跳转函数进行防篡改加固 484
    - 9.4.4 讲评 485

9.5 小结	486
第10章 软件相似性分析	489
10.1 应用	490
10.1.1 重复代码筛选	490
10.1.2 软件作者鉴别	492
10.1.3 剽窃检测	495
10.1.4 胎记检测	496
10.2 定义	497
10.3 基于k-gram的分析	501
10.3.1 算法SSSWawinnow: 有选择地记录k-gram hash	501
10.3.2 算法SSSWAMOSS: 软件剽窃检测	504
10.3.3 算法SSMCKgram: Java 字节码的k-gram “胎记”	507
10.4 基于API的分析	509
10.4.1 算法SSTNMM: 面向对象的“胎记”	510
10.4.2 算法SSTONMM: 动态函数调用“胎记”	512
10.4.3 算法SSSDL: 动态k-gram API “胎记”	513
10.5 基于树的分析	514
10.6 基于图的分析	518
10.6.1 算法SSKH: 基于PDG的重复代码筛选	518
10.6.2 算法SSLCHY: 基于PDG的剽窃检测	521
10.6.3 算法SSMCwpp: 整个程序的动态“胎记”	522
10.7 基于软件度量的分析方法	525
10.7.1 算法SSKK: 基于软件度量的重复代码筛选	525
10.7.2 算法SSLM: 基于度量的软件作者鉴别	527
10.8 小结	532
第11章 用硬件保护软件	534
11.1 使用发行的物理设备反盗版	535
11.1.1 对发行盘片的保护	536
11.1.2 软件狗和加密锁	541
11.2 通过可信平台模块完成认证启动	545
11.2.1 可信启动	546
11.2.2 产生评估结果	548
11.2.3 TPM	550
11.2.4 盘问式验证过程	551
11.2.5 社会可信性和隐私问题	553
11.2.6 应用和争议	555
11.3 加密的可执行文件	556
11.3.1 XOM体系结构	557
11.3.2 阻止重放攻击	560
11.3.3 修补有漏洞的地址总线	561
11.3.4 修补有漏洞的数据总线	564
11.3.5 讲评	565
11.4 攻击防篡改设备	565
11.4.1 监听总线——破解微软的XBOX	566
11.4.2 猜测指令——破解达拉斯半导体公司的DS5002FP微处理器	567
11.4.3 破解智能卡	570
11.4.4 非侵入式攻击	573
11.4.5 主板级的保护	574
11.5 小结	576
参考文献	578
• • • • •	<a href="#">(收起)</a>

## 标签

加密解密

计算机

安全

加密与解密

逆向工程

软件加密与解密

软件保护

编程

## 评论

:TP309.7

---

大部分算法看起来高深莫测，实际上只是伪码，不值得细看

---

很深的一本书，简单翻了一遍，要仔细看得花很长很长的时间，作者太用心了，认真分析了大量软件保护和攻击的方法。另外翻译也挺不容易的，近800页的书，内容还很深很偏。

-----  
不好不好

-----  
[软件加密与解密 下载链接1](#)

## 书评

以下是转述: 我是译者, 没办法, 这个书名根本就不是我起的。  
现如今我们这些译者, 译稿交上去之后, 就是砧板上的鱼肉, 比如, 你看见译者序里头  
“本书的英文名是Surreptitious Software,  
surreptitious一词的意思是“保密的, 隐瞒的”, 所以surreptitious  
software的意思就是...

-----  
[软件加密与解密 下载链接1](#)