一种面向应用需求的代码保护方法

徐 超[1,2] 何炎祥[1,3] 陈 勇[1,3] 吴 伟[1,3] 刘健博[1,3]
(武汉大学计算机学院 武汉 430072)[1] (徐州工业职业技术学院 徐州 221000)[2]
(武汉大学软件工程国家重点实验室 武汉 430072)[3]

摘 要 为了防止攻击者对编译后可执行代码进行读取分析和篡改,提出了一种基于编译器分析的软硬件相结合的保护框架。首先对具体应用需求进行分类(数据重要型和算法重要型),然后提取分类后的关键代码块,生成带标记的二进制代码,最后综合数字签名(RSA)和(AES)加解密算法对标记信息进行相应处理,并将其加载到FPGA进行校验运行。实验分析显示,该方法具有较好的可操作性和可维护性,既减小了软件保护的开销,降低了系统实现成本,又达到了保护目的。

关键词 代码安全保护,关键代码块,应用需求,数字签名,FPGA
中图法分类号 TP309.1 **文献标识码** A

Code Protection Method Oriented to Application Requirement

XU Chao[1,2] HE Yan-xiang[1,3] CHEN Yong[1,3] WU Wei[1,3] LIU Jian-bo[1,3]
(School of Computer, Wuhan University, Wuhan 430072, China)[1] (Xuzhou College of Industrial Technology, Xuzhou 221000, China)[2]
(State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China)[3]

Abstract Based on compiler analysis and combining handware and software, this paper put forward a protected framework in order to prevent attackers to read, analyse and tamper the compiled executable code when the process compiler compiles the source code. Firstly, classifying specific application requirement(data important and algorithm important), then extracting key code block and generating marked binary code after classification, synthesizing the advantage of digital signature(RSA) and (AES) encryption and decryption algorithm, employing digital signature and encryption to encrypt block marked information, finally loading it to FPGA to verify and run. The experimental analysis show that the method has good maneuverability and maintainability, can reduce the software protection costs and system implement cost, and reach the purpose of protection.

Keywords Code security, Key code block, Application requirement, Digital signature, FPGA

1 引言

随着软件规模的不断扩大,其内部结构越来越复杂;很多时候软件不以人们期望的方式工作,会发生各种各样的故障和失误,甚至失效,造成巨大损害,如"7.23"甬温线发生动车追尾事故,由于信号设备存在安全故障,造成中国铁路交通史上的一起重大伤亡事故。在人们对软件系统极为依赖的今天,软件的可信性研究对于信息安全、系统安全乃至整个国家、社会的安全都极为重要。影响软件可信性的主要因素包括来自软件内部的代码缺陷、代码错误、程序故障及来自软件外部的病毒、恶意代码等[1]。软件是由代码组成的,其可信性从根本上可表现为代码的可信性[2],因此从代码角度来保证软件的可信性是实现可信软件的重要途径之一。

众所周知,软件程序一般都需要经过编译器编译后方能执行,编译器在整个计算机软件系统中所处的"位置"非常特别——几乎所有的可执行软件都是通过编译器编译后产生的[3]。编译器在对程序源代码进行编译的过程中,很可能篡改其原本的语义,生成不安全的目标代码[4],因此,如果能在编译器对程序源代码编译的过程中进行保护,将大大提升软件的可信程度。

目前常用的目标代码保护方法主要分为硬件保护方法和软件保护方法两大类别[5],只采用某一种方法在安全强度、系统开销和实现成本之间难以找到平衡:纯软件保护方法(如水印、代码混淆等)虽降低了成本,但也减小了系统安全性,代码的执行效率也大幅度下降[7,8];纯硬件保护方法采用密码技术,安全强度很高,同时安全协处理器可大大提高代码的执行速度[6],但其对硬件的要求增加了系统的实现成本。

本文在现有的可执行代码保护方法的基础上提出了一种基于编译器分析的软硬件相结合的保护框架,即采用数字签名与AES加密算法相结合的方法进行保护,并在FPGA(Field-Programmable Gate Array)上进行了相应的实现和仿真[9,10],最终既实现了代码的高安全保护,又降低了系统实

到稿日期:2012-02-21 返修日期:2012-05-20 本文受国家自然科学基金重点项目(91118003),国家自然科学基金面上项目(61170022)资助。

徐 超(1980-),男,博士生,讲师,主要研究方向为可信软件、嵌入式系统等;何炎祥(1952-),男,教授,博士生导师,主要研究方向为可信软件等。

现成本。

本文第 2 节描述目标代码保护框架；第 3 节描述关键代码块提取；第 4 节详细介绍采用数字签名后加密方式对关键代码块进行保护，并在 FPGA 中保证其安全稳定运行；第 5 节介绍软件模拟的实验结果；最后是相关工作和总结。

2 目标代码保护框架

2.1 面向应用的保护

目标代码的保护内容根据程序具体应用需求而定，程序员通常可以采用手动方式设置安全级别，其安全级别分为 3 种：第一种是代码级（逐条进行审核保护），此方法比较复杂；第二种是块级别（程序块保护），此方法由于程序分支较多，容易出错；第三种是整体级别，此方法简单，但开销较大，效率低。

因此，本文提出在编译器对程序代码进行分析过程中，根据代码保护应用需求的不同，自动地针对关键代码块进行保护，从而将目标代码有效地保护起来。

2.2 面向应用的分类

数据和算法是程序的灵魂，因此程序按应用需求主要可分为数据重要型程序、算法重要型程序以及平衡型程序。数据重要型保护策略主要针对数据之间的逻辑关系、数据的存储方式和数据的运算 3 个方面进行保护；而算法重要型保护策略主要保护程序的执行流程，防止程序的主要设计思想被他人窃取[11-13]；平衡型程序则处于以上两种策略之间，该类程序保护内容主要是根据程序员指定的部分进行保护。由于本文讨论的主要是利用编译器以尽可能小的开销对程序代码进行自动保护，因此平衡型程序暂不考虑。具体保护方法如图 1 所示。

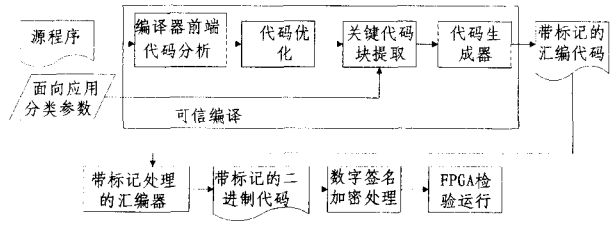

图 1 编译目标代码保护方案

源程序经过编译器前端分析后，根据具体的应用参数，确定程序是算法重要型还是数据重要型程序；然后选择相应的策略对关键代码块进行提取，生成带标记的汇编代码，又经过带标记处理的汇编器进行编译后，生成带标记的二进制代码；接着根据标记对带标记的二进制代码的 HASH 值进行数字签名和加密处理；最后将其加载到 FPGA 上校验运行。

3 关键代码块提取

3.1 查找关键代码块

为查找程序中的关键代码块，本文对数据重要型程序和算法重要型程序分别给出如下关键代码块的定义。

定义 1 数据重要型关键代码块，使用式(1)计算关键代码块的权重，权重越大，表明该代码块越重要。

$$W_d(I)=\sum_{i\in V(I)}(\alpha * p(i)+\beta * live(i)) \tag{1}$$

式中，$W_d(I)$表示代码 I 的数据重要型权重，$V(I)$表示代码 I 中使用和定义的变量集合，$p(i)$表示变量 i 在程序中使用的次数，$live(i)$表示变量 i 的生命期长度。

定义 2 算法重要型关键代码块，使用式(2)计算关键代码块的权重，权重越大，表明该代码块越重要。

$$W_c(B)=pre(B)+suc(B) \tag{2}$$

式中，$pre(B)$表示基本块 B 的前驱基本块的数目，$suc(B)$表示基本块 B 的后继基本块的数目。具体步骤如下：

(1)首先依次遍历每个函数，以基本块为单位构建控制流图；

(2)然后在控制流的基础上，构建函数调用关系图，图中每个节点代表一个函数，每条有向边代表函数调用，边上的权值代表调用的次数；

(3)对于递归调用，采用 3 次展开，删除循环边的方法，计算其调用次数；

(4)根据函数调用关系图，按每个节点入度的权值从大到小进行排序，选择前 $n/3$ 个节点作为关键节点；

(5)依次遍历每个关键节点。

a)数据重要型程序：根据控制流图计算该函数中变量的数据流信息，构造该变量的生命期，同时计算每个变量的使用次数，然后根据变量生命期的长短以及变量使用次数计算指令 I 的 $W_d(I)$，选择 $W_d(I)>\max(W_d(S))$，其中 $\max(W_d(S))$表示所有指令权值中的最大值。

b)算法重要型程序：根据控制流图计算每个基本块的前驱块和后继块的数目 $W_c(i)$，其中 m 表示基本块的总数目。然后根据式(3)选择关键基本块进行保护。

$$W_c(i)>\sum_{i=1}^{m}W_c(i)/m \tag{3}$$

3.2 标记的插入

标记信息是编译器提供给后端的有关源代码的关键代码块信息，它主要用于告诉后端的安全处理模块如何处理生成的目标代码，以尽可能减少保护目标代码的开销。标记插入方法如下：

(1)根据数据类型的不同，插入不同的标记伪指令：

a)数据重要型程序：由于标记是针对每条语句，因此只需要在关键指令前插入相应的标记指令即可，此处采用 SAFI 作为相应的标记伪指令，该指令不生成具体的代码。

b)算法重要型程序：依次遍历基本块，根据基本块的安全标记，在基本块的入口插入标记的中间表示 SAFE，在基本块的出口插入结束标记的中间表示 SEND，这两条指令均不产生具体代码。

(2)在生成目标代码时，对标记的中间表示使用专门的新增的标记指令：SAFI，SAFE，SEND。该指令是目标代码指令集以外的指令，用于分隔代码块与安全信息。因此要求目标指令集至少拥有 3 条可扩展的指令。

(3)在最后对目标代码进行执行时，根据新增的标记伪指令对目标代码加密和数字签名进行处理。

4 关键代码块保护策略

为对关键代码块进行相应保护，首先针对提取的关键代码块，采用数字签名与 AES 加密算法相结合的方法进行保护。具体方法为：先数字签名后用 AES 算法进行加密，带标记的二进制代码经 Hash 函数对关键代码块进行散列处理；然后用 RSA 私钥 K_d 对 Hash 值进行数字签名；最后采用 AES 算法对关键代码块进行加密，加密钥为 K[14]。具体方法

如图 2 所示。

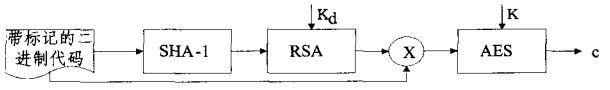

图 2　关键代码块的加密过程

其次，对带标记的二进制代码进行数字签名和加密之后，设计图 4 所示 FPGA 模块来保证其安全稳定地运行。具体分析过程如图 3 所示。

1. 对加密后的签名数据进行签名校验
2. If 校验不通过
　　停止运行
3. Else if 解密标记为 1
　　将当前指令交与解密部件解密
4. End if
5. If 该指令是标记指令
　　根据标记指令设置相应的解密标记位
6. Else
　　交与处理器直接运行
7. End if

图 3　基于 FPGA 代码块保护分析过程

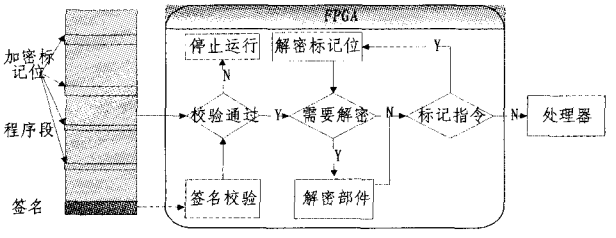

图 4　基于 FPGA 代码块保护运行过程

5　示例分析

为了验证该方法对目标代码保护的可行性和有效性，进行了软件模拟实验。实验过程主要包括以下几个步骤。

1. 首先构建源程序（见图 5）的控制流图（见图 6）；

```
F(a[],k)
{
  d=0;
  if(k>0)
  {
    c=getc();
    P=&c
    do{
      b+=*p
      If (b<256)
      {
        c+=a[d];
        c=getc();
        }
        else
        {
          d++;
          b=256-c;
        }
    }while(c! =0)
  }
  else
  {
    b=k*10;
  }
  return b
}
```

图 5　源程序结构

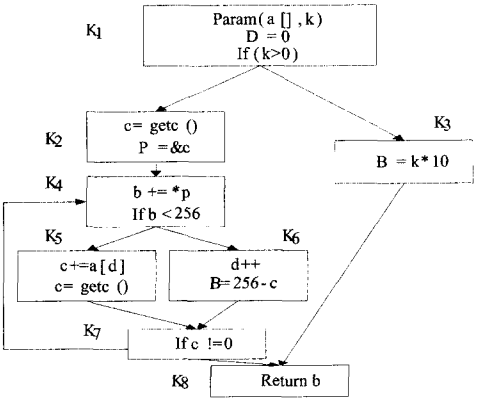

图 6　控制流图

2. 根据查找关键代码块的步骤，对图 6 所示代码块进行分析，获得表 1 所列相关数据。利用式(1)，结合表 1 的数据，获得关键代码块 K_4 和 K_7。

表 1　关键代码数据表

	前驱数目	后继数目	权值	阈值
K_1	0	2	2	
K_2	1	1	2	
K_3	1	1	2	
K_4	2	2	4	20/8=2.5
K_5	1	1	2	
K_6	1	1	2	
K_7	2	2	4	
K_8	0	2	2	

3. 根据标记插入算法，对图 6 中查找的关键代码块 K_4 和 K_7 插入相应的关键块标记，以供后续程序进行相应的处理，如图 7 所示。

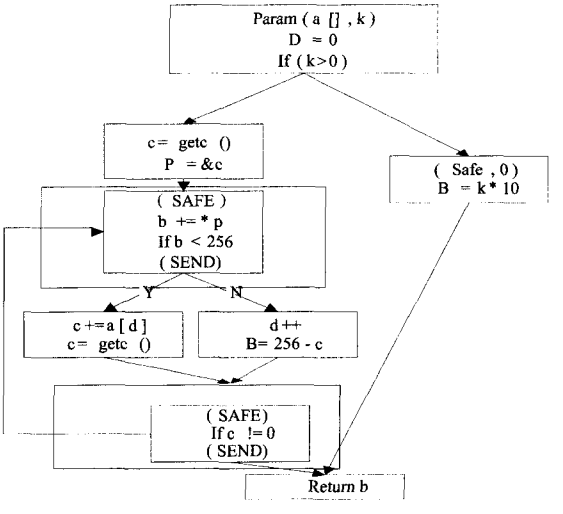

图 7　关键代码块标记图

4. 根据图 7 中的标记，对数据进行数字签名和加密保护，并将其加载到 FPGA 上运行。

6 实验及结果分析

为检验该保护框架的有效性，本文先利用 GCC4.6.3 编译器对源代码进行分析，分别对算法重要型程序和数据重要型程序插入对应的标记信息，然后使用 Quartus 仿真工具进行相应的 FPGA 模拟实验。图 8 和图 9 分别描述了算法重要型和数据重要型程序在加入标记信息前后代码大小的变化值。其中图的横坐标表示测试用例编号，纵坐标表示对应测试用例的数据量，其以字节为单位。

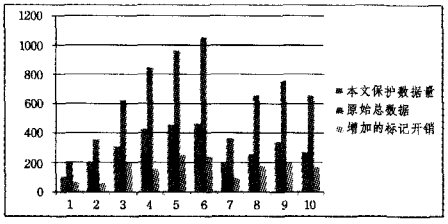

图 8 算法重要型实验结果

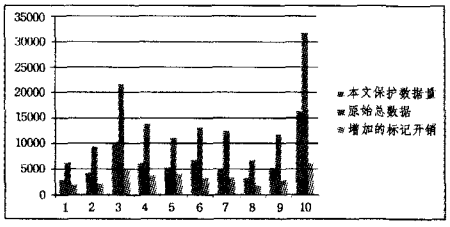

图 9 数据重要型实验结果

通过以上模拟实验可以看出，该方法无论是对数据重要型程序还是算法重要型程序，均能有效地减少保护的数据量（最高减少率可达到 33%，最低减少率也达到 15%，平均减少率为 27%），从而有效地减少了软件保护的开销，提高了软件运行的效率。

7 相关工作

目前，关于编译所生成目标代码的安全性问题研究可分为 3 个方面：代码安全漏洞防范、代码安全属性证明、可执行代码加密技术。可执行代码加密技术主要用于防止攻击者对编译后可执行代码进行读取分析和篡改。目前的主要方法有：(1)版权(水印)Copyright Notice(Watermark)，即在代码中嵌入版权信息，然后在运行时检测其正确性[15]。水印在跟踪和识别代码、防范代码的非法复制方面非常有用，但它并不能阻止攻击。(2)混淆代码(Obfuscation)，混淆编译器故意生成顺序混乱的代码，使得程序难以捉摸，从而达到隐藏重要算法和重要数据的目的[16]。在文献[17]中，C. Collberg 等人主要通过重新排列或变换代码，使得攻击者难以分析程序的控制流图。在文献[18]中，William Zhu 等人已证明分析混淆程序是 NP 难问题。混淆技术简单实用，不涉及任何加解密算法就可防止代码被攻击者分析，但其可保证的安全级别较低，且没有针对编译器具体应用需求进行分类保护。(3)签名(Signatures)，主要用来验证代码是否被更改，防止攻击者的恶意篡改行为[19]。在文献[20]中，B. Horne 等人提出了一种防篡改的自校验技术，该方法通过计算代码段的哈希值，并与正确值相比较，来确定代码是否被篡改。但该方法的可靠性依赖于校验和算法的保密程度，如果攻击者破解了校验和算法，那么验证系统将面临严重威胁。(4)加解密(Cryptographic Architectures)，通过编译器对程序代码进行加密，在安全协处理器中，程序可以加密的形式运行，这样代码就不会暴露于任何不可信的主存中，可防止攻击者分析或窃取。这种基于密码体系结构的技术需要硬件平台的支持，编译器生成代码时就立即对其加密。但造价昂贵的安全硬件平台限制了此类技术的发展。

因此，本文采用数字签名与 AES 加密算法相结合的方法进行保护，此方法不仅弥补了数字签名的依赖校验和算法的保密程度的不足，而且根据具体应用需求，只针对关键代码块进行保护，减少了软件保护的开销。最后将其加载到 FPGA 上稳定运行，形成一种有效的目标代码保护框架。

结束语 本文在结合硬件保护和软件保护优点的基础上，提出了一种目标代码保护框架。编译器在生成目标代码之前，根据具体应用的需求进行分类保护，提取关键代码块进行相应标记，再根据标记信息进行加密处理，以在保护开销和保护质量间获得平衡。同时，对整个提取加密过程进行了设计和仿真，结果表明，该方法无论在保护开销还是保护效果上均有较好的表现。

参 考 文 献

[1] Michael H, David L. Writing Secure Code [M]. Redmond, Washington, USA: Microsoft Press, 2005

[2] Yu Da-chuan, Shao Zhong. Verification of Safety Properties for Concurrent Assembly Code [C] // Proceedings of ICFP'04. Snowbird, Utah, USA, ACM, September 2004

[3] Fan Da-wei, Li Zhao-peng, Jiang Xin-yu. Code Optimization and Program Criterion Conversion in Certifying Compiler [J]. Micro computer system, 2011, 7: 1400-1405

[4] Kongubangaram V, Gelbart O, Simha R, et al. Compiler-Directed Region-Based Security for Low-Overhead Software Protection [C]// Third IEEE International Symposium on Dependable, Autonomic and Secure Computing (DASC). 2007: 47-54

[5] Luo Xiao-guang, Wang Sheng-yuan, et al. Research on Mobile Code Security Based on Technology of Confirmation Compiling [J]. Science Computer, 2007, 2: 267-269

[6] Wappler U, Fetzer C. Hardware Failure Virtualization via Software Encoded Processing [C]// Proc. of International Conference on Industrial Informatics. Paris, France: Hachette Livre Press, 2007: 977-982

[7] Madou M, et al. Software protection through dynamic code mutation [C]// Information Security Applications. 2006: 194-206

[8] Kanzaki Y, et al. Exploiting self-modification mechanism for program protection [C] // Computer Software and Applications Conference. 2003: 170-179

[9] Wang Qin, Sun Fu-ming, et al. Summarize of FPGA Designing Security [J]. Micro computer system, 2010, 7: 1333-1337

[10] Gelbart O, Ott P, Narahari B, et al. CODESSEAL: Compiler/FPGA Approach to Secure Applications [J]. Lecture Notes in Computer Science, 2005, 3495: 530-535

[11] Software Engineering Research and Practice [C]// SERP'05. Las Vegas, June 2005

[12] Lin Chun-xiao, Chen Yi-yun, Li Long, et al. Garbage collector verification for proof-carrying code [J]. Journal of Computer Science & Technology, 2007, 22(3): 426-437

[13] Kiyomoto, Andrew S. A Security Protocol Compiler Generating C Source Codes[C]// International Conference on Information Security and Assurance. Busan, Korea, April 2008:20-25

[14] Zhang Huan-guo, Wang Zhang-yi. Cryptography Introduction [M]. Wuhan: Wuhan University Press, 2009:23-25

[15] Sun Sheng-he, Lu Zhe-ming, Niu Xia-mu. Digital Watermarking Technology and Application[M]. Beijing: Science press, 2004: 36-37

[16] Wang C X. A security architecture for survivability mechanisms [D]. University of Virginia, Department of Computer Science, 2000

[17] Collberg C, Thomborson C. Watermarking, Tamper-Proofing, and Obfuscation-Tools for Software Protection[J]. IEEE Transactions on Software Engineering, 2002, 28(6): 735-746

[18] Zhu W, Thomborson C D, Wang Fei-yue. Applications of Homomorphic Functions to Software Obfuscation[C]// WISI. 2006: 152-153

[19] Han Xiao-xi, Wang Gui-lin. An Attack to Multisignature Schemes Based on Discrete Logarithm[J]. Chinese Journal of Computers, 2004, 8: 1147-1152

[20] Horne B, Matheson L, Sheehan C, et al. Dynamic self-checking techniques for improved tamper resistance[C]// Proc. 1st ACM Workshop on Digital Rights Management (DRM2001). Springer, 2002: 141-159

(上接第 61 页)

前所述。

综合命题 1—命题 3 的分析可知，协议能够满足公平性。

5 与改进前协议的比较

本文对文献[12]提出的协议进行了改进，方案主要是在原协议的交换轮次中增加了部分交换项，改进了原协议执行中的部分判断条件，同时删除了原协议中不影响协议安全属性的冗余消息及冗余子协议。通过 Kailar 逻辑证明了改进后的协议弥补了原协议存在的安全隐患。以下通过对协议的整体执行流程以及 3 个子协议的流程进行比较，分析改进后的协议与原协议的执行效率。

在协议的整体执行流程方面，改进后的协议并没有增加原协议的执行步骤，同时删除了由 A 发起的 Recover 子协议，故在协议执行流程方面，改进后的协议更为精简。在 3 个子协议中，改进后的协议增加的数据项包括：

1)在 Exchange 子协议的 b)中增加了一个签名运算，在 c)中增加了一个 Hash 函数运算；

2)在 Recover 子协议中的 a)中增加了 PR 等数据项，说明协议发起者是合法参与方，c)中增加了一个加密操作以满足邮件的保密性；

3)在 Abort 子协议中的 a)中增加了 PR 和 Hash 等数据项，说明协议发起者是合法参与方，同时在 b)和 c)中若 $R_i \in R''\_recovered$，则对签名内容进行更改。

以上增加的数据项对于协议安全属性的保证是必须的，且并未对原协议的执行效率和复杂度造成影响。同时由于删除了原协议的 Exchange 子协议中的 d)消息和由 A 发起的 Recover 子协议，因此改进后的协议并没有降低原协议执行效率。

结束语 本文分析了文献[12]提出的一个基于离线第三方的多方认证邮件协议，指出了该协议存在的安全隐患，并对原协议进行了改进。通过与文献[12]的比较，可以看出本文对协议所做的改进并没有降低原协议的执行效率，但消除了原协议存在的安全隐患。然后利用 Kailar 逻辑对改进后的协议进行了形式化分析。研究结果表明，在保证执行效率的基础上，改进后的协议具有多方认证邮件协议所必需的保密性、不可否认性及公平性的特性，而且具有抗篡改、重放、合谋等攻击的特点。

参 考 文 献

[1] González-Deleito N, Markowitch O. Exclusions and related trust relationships in multi-party fair exchange protocols [J]. Electronic Commerce Research and Applications, 2007, 6(3): 343-357

[2] Kremer S, Markowitch O, Zhou Jian-ying. An intensive survey of fair non-repudiation protocols [J]. Computer communications, 2002, 25(17): 1606-1621

[3] Franklin M, Tsudik G. Secure Group Barter: Multi-Party Fair Exchange With Semi-Trusted Neutral Parties [A]// Process of Financial Cryptography'98 [C]. LNCS1465. Anguilla, Springer, 1998: 90-102

[4] Kremer S, Markowitch O. A multi-party non-repudiation protocol [C]// The 15th International Conference on Information Security. Beijing, China, IFIP World Computer Congress, 2000: 271-280

[5] Markowitch O, Kremer S. A multi-party optimistic non-repudiation protocol [C]// Information Security and Cryptology ICISC 2000: Third International Conference. Seoul, Korea: Spinger-Verlag, 2001: 109-122

[6] Kremer S, Markowitch O. Fair Multi-Party Non-Repudiation protocols [J]. International Journal of Information Security, 2003, 1(4): 223-235

[7] 韩志耕，罗军舟. 一个公平的多方不可否认协议[J]. 计算机学报，2008，31(10)：1705-1715

[8] 韩志耕，罗军舟. 多方不可否认协议时限性分析与改进 [J]. 电子学报，2009，37(2)：377-381

[9] Ferrer-Gomila J L, PayerasCapell M, Huguetrotger L. A realistic protocol for multi-party certified electronic mail [C]// Proceedings of Information Security Conference. Sao Paulo: Springer, 2002: 210-219

[10] Zhou J. On the security of a multi-party certified e-mail protocol [C]// Proceedings of 2004 International Conference on Information and Communications Security. Malaga: Springer, 2004: 40-52

[11] Zhou J, Onieva J, Lopez J. Optimized multi-party certified email protocols [J]. Information Management and Computer Security, 2005, 13(5): 350-366

[12] 王彩芬，贾爱库，刘军龙，等. 基于签密的多方认证邮件协议[J]. 电子学报，2005，33(11)：2070-2073

[13] Kailar R. Accountability in electronic commerce protocols [J]. IEEE Trans. on Software Engineering, 1996, 22(5): 313-328

[14] 卿斯汉. 一种电子商务协议形式化分析方法[J]. 软件学报，2005，16(10)：1757-1765