Journal of Computer Applications
计算机应用,2013,33(11):3262-3266
ISSN 1001-9081
CODEN JYIIDU
2013-11-01
http://www.joca.cn

文章编号:1001-9081(2013)11-3262-05
doi:10.11772/j.issn.1001-9081.2013.11.3262

基本路径测试用例设计算法

王　敏[1*],陈少敏[1],陈亚光[2]
(1.武昌理工学院 信息工程学院,武汉 430223;　2.中南民族大学 生物医学工程学院,武汉 430074)
(*通信作者电子邮箱 kkmnet@yeah.net)

摘　要: McCabe 提出的基本路径测试法(McCABE T J. A complexity measure. IEEE Transactions on Software Engineering, 1976, SE-2(4): 308-320)是动态白盒测试技术中严谨而有效的方法,但存在测试用例设计效率较低的问题,影响了该方法在工程项目中的广泛应用。为了解决这一问题,从被测程序的基本结构出发,提出一种基于组合的基本路径测试用例设计方法。创建一种基于Z路径覆盖的基本单元图,构建由基本单元图组合形成控制流图的组合规则,以此为基础提出了基本路径组合算法,该算法只需一次扫描程序得到程序基本结构的路径集,将这些路径进行组合即可生成被测程序的基本路径集。该方法比 McCabe 所提出的方法构造过程简洁,能有效提高基本路径测试用例设计的效率。

关键词: 基本单元;基本路径测试;基本路径集;自动化测试;白盒测试
中图分类号: TP311.5　　**文献标志码:** A

Test case design algorithm for basic path test

WANG Min[1*], CHEN Shaomin[1], CHEN Yaguang[2]
(1. *College of Information Engineering, Wuchang University of Technology, Wuhan Hubei* 430223, *China*;
2. *College of Biomedical Engineering, South-Central University for Nationalities, Wuhan Hubei* 430074, *China*)

Abstract: McCabe's basic path testing method (McCABE T J. A complexity measure. IEEE Transactions on Software Engineering, 1976, SE-2(4): 308-320) is a more rigorous software testing one in dynamic white-box testing techniques, but the efficiency of McCabe method is lower. To solve this problem, this paper proposed an algorithm to design basic path test case according to the basic program structure. The algorithm first created a basic unit chart based on the Z-path coverage. Next, the rules that combined a control flow graph from basic unit were established. On this basis, the combination algorithm of the basic path was constructed. The algorithm collected the path set of basic program structure by scanning a program only once, and then generated the basic path set using those paths by combination ways. This method is more concise than the method proposed by McCabe, and it can improve the efficiency of the basic path test case design.

Key words: basic unit; basic path testing; basic path set; automated testing; white-box testing

0　引言

近年来,不少软件企业陷入了软件后期维护的危机之中。其主要原因是软件开发过程中测试不充分。据业界统计,软件测试过程中的单元测试可以发现大约80%的软件缺陷,同时,由于软件缺陷的放大效应,单元测试阶段的动态白盒测试对尽早发现软件缺陷、降低项目风险起着重要的作用。

由 McCabe 提出的基本路径测试法被认为是动态白盒测试技术中严谨而有效的测试方法[1],但是当程序逻辑结构较为复杂时,人工实现变得比较困难,进而对测试用例设计的效率和正确性都有较大的影响。科学工作者们一直在努力改变这种状况,针对基本路径集求解问题提出了一些算法[1-8],但迄今为止基本是拘泥于 McCabe 的基本思想所进行的一系列算法实现和算法改进,仍然不能圆满地对应各种程序结构。因算法的制约,目前尚没有面向基本路径测试的自动化测试工具问世。处于研究阶段的部分测试工具,由于其实用性和方便性欠佳,一直不能在工程实践中发挥作用。因此,寻找切实可行的基本路径集求解算法,实现基本路径测试过程的自动化依然是正在探索的问题。

本研究有别于 McCabe 的思路和方法,探索一种新的基本路径测试用例设计方法:首先通过语法分析获取被测程序的基本结构信息,这些信息以基本结构的路径字符串形式存放;然后采用路径字符串组合的算法求得被测程序的基本路径测试用例集。

1　控制流图组合法

基本路径测试用例集的设计主要是求解被测程序的基本路径集,McCabe 方法是通过被测程序的控制流图来寻找基本路径集,而被测程序可看作是3种基本结构:顺序结构、分支结构和循环结构按不同方式组合的产物;因此,分析3种基本结构的控制流图及其构成被测程序控制流图的规律,对基本路径集的求解具有重要意义。

1.1　基本单元图

按程序结构划分,McCabe[9]提出了5种基本控制结构对应的图形符号。由于全路径覆盖的路径数目太多,无论是人工或是自动化测试都难于实现,为了减少测试路径数,本研究借鉴于Z路径覆盖的思想[10],构建了4种基本图形符号,如图1所示,称之为基本单元图(简称基本单元)。

收稿日期:2013-05-20;**修回日期:**2013-07-20。　**基金项目:**国家自然科学基金资助项目(61072075)。
作者简介:王敏(1968-),女,四川眉山人,副教授,主要研究方向:软件测试;　陈少敏(1962-),男,四川成都人,讲师,主要研究方向:软件工程与方法;　陈亚光(1951-),男,安徽砀山人,教授,主要研究方向:信号处理。

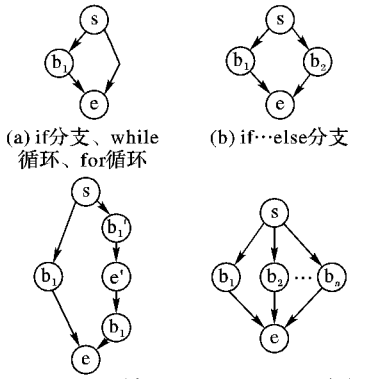


图 1　基本单元图的图形符号

其中:图 1(a)为 if 分支、while 或 for 循环结构演变成的一个二分支结构;图 1(b)表示选择结构,仍沿用 McCabe 的基本符号;do…while 循环可演变为一个顺序结构,能保证每条语句都能执行到,但为了达到所有判断分支覆盖的标准,在此,仍然将其演变成一个二分支结构,如图 1(c);图 1(d)表示多分支结构,仍沿用 McCabe 的基本符号。由于顺序结构可转化为一个节点融入各分支结构中,且不影响路径选择,在此不予考虑。

1.2　控制流图的组成

为了寻找基本单元组合形成控制流图的规律,在此将基本单元图的节点进行分类,现定义如下。

定义 1　假设被测程序由 m 个基本结构组成,对应的基本单元图记作 $g_i, 1 \leqslant i \leqslant m, g_i = (s_i, B_i, e_i)$。其中:$s_i$ 称为子源节点,对应基本单元的起始节点 s;B_i 称为子分支节点集,是基本单元 g_i 中直接与子源节点相连接的非汇节点的集合,对应基本单元图中的 $b_1, b_2, \cdots, b_n$;e_i 称为子汇节点,对应基本单元的终止节点 e。

除 do…while 循环结构的基本单元以外,其他结构均由上述 3 类节点构成,do…while 右分支中的节点 e′和 b_1′不属于上述 3 类节点,但它不影响后续讨论,因此不引入记号标识和定义。

定义 2　由被测程序的部分基本单元组合形成的局部控制流图叫作被测程序的子控制流图,记作 sg。基本单元图是子控制流图的特例,它只含一个基本单元。

分析程序及基本单元图结构可得如下结论:在基本单元图中,s_i 与 B_i 节点集中的各节点是直接相连的,即其间不可能插入其他子控制流图;B_i 节点集中的各节点与 e_i 之间可能插入子控制流图;e_i 后面也可能接续子控制流图。

下面以一段伪代码(例 1)来分析基本单元图组合形成被测程序控制流图的规律。

例 1:

```
1)    while (条件表达式 1)
2)    {
3)      语句 1
4)    }
5)    if (条件表达式 2)
6)    {
7)      if (条件表达式 3)
8)      {
9)        语句 2;
10)     }
11)     else
12)     {
13)       语句 3;
14)     }
15)   }
16)   else
17)   {
18)     语句 4
19)   }
```

例 1 中含一个循环结构和两个 if 分支结构,可得 3 个基本单元图如图 2(a)、(b)、(c),例 1 对应的控制流图如图 2(d)。这里采用行号作为基本单元的节点编号,其优点在于:程序行号易于获取,行号也是程序执行顺序的标志,同时在后续的研究中用行号表示的节点编号大小是决定基本单元组合及路径字符串组合的重要依据。

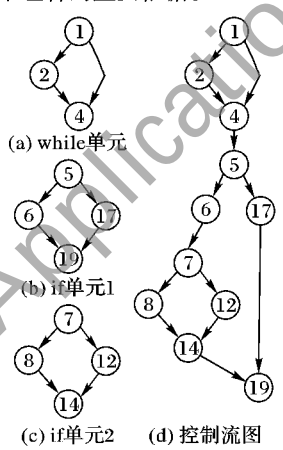


图 2　伪代码中所含基本单元图及控制流图

被测程序可看作是程序三大基本结构按顺序或嵌套方式组合的产物。由例 1 及其控制流图的构成分析发现:由基本单元图组合形成控制流图也包含两种方式,即顺序组合和嵌套组合。定义如下:

定义 3　若在子汇节点后面接续子控制流图,称这种组合方式为顺序组合;若在子分支节点(B_i 节点集中的各节点)与子汇节点之间插入子控制流图,称这种组合方式为嵌套组合。

由此可见,控制流图是由基本单元图按顺序或嵌套方式组合的产物。如何提取被测程序的基本单元,是进一步探索由基本单元组合生成控制流图方法的前提。

1.3　基本单元信息的提取算法

这里利用堆栈数据结构来实现基本单元信息的提取,即获取由子源节点、子分支节点和子汇节点构成的基本单元图,因此,首先分析如何捕获程序中的这 3 类节点信息,为了降低算法实现的复杂度,本研究要求被测程序代码编写满足一定规范(某些项目的编程规范中已包含这样的规范),即各分支结构不论语句行多少均用大括号包围,且左右大括号各占一行。以 C#语法为例,可按如下方法捕获各类节点。

1)子源节点。子源节点对应的代码行为分支和循环结构的开始行,扫描被测程序,通过判断读取的代码行是否含有分支和循环结构的关键字即可判断是否属于子源节点行。

2)子分支节点。子源节点代码行的下一行必为子分支节点行,对于二分支以上的分支结构,关键字 else、else if、case、default 的下一行必为子分支节点行。

3)子汇节点。子汇节点行必为“}”所在行,但不是所有的“}”所在行均为子汇节点行,如果“}”后续行为分支结构关键字 else、else if、case、default,则该行不是子汇节点行;否则即可判定为子汇节点行。

图 3 为基本单元信息提取算法的流程。

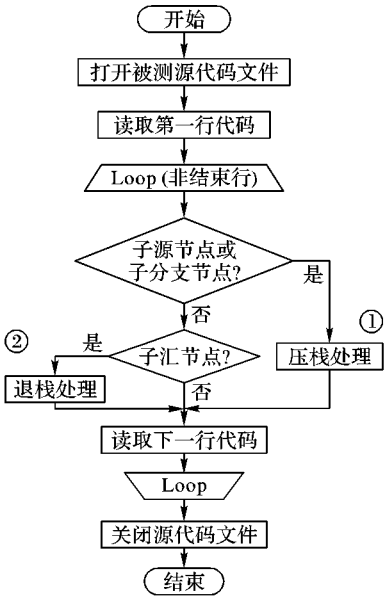
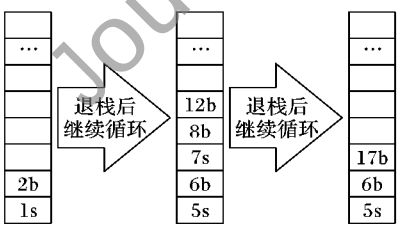


图 3 基本单元信息提取流程

过程①的处理将行号和节点类型标志(s 为子源节点,b 为子分支节点,e 为子汇节点)压入堆栈;过程②进行循环退栈,直到退栈信息为子源节点停止退栈,过程②得到如(ns,m_1b,m_2b,…,m_xb,pe)的结构信息,其中 n、m_i、p 分别为子源节点、子分支节点、子汇节点对应的行号,所得信息即为同一个基本单元的节点构成信息,根据这些信息即可得到一个基本单元图。按上面的流程对例 1 进行处理,示意图如图 4。处理过程为:逐行读取源代码,遇到子源节点(第 1 行)和子分支节点(第 2 行)进行压栈,到第 4 行为子汇节点行,生成节点信息 4e,此时堆栈元素构成如图 4(a),开始退栈,直到退出含“s”的元素 1s,得到基本单元信息{1s,2b,4e},由此可得到如图 2(a)的基本单元图;同理,图 4(b)、(c)退栈后得到基本单元信息{7s,8b,12b,14e}、{5s,6b,17b,19e},由此可得到如图 2(c)、(b)的基本单元图。



(a) 第1次退栈前 (b) 第2次退栈前 (c) 第3次退栈前

图 4 例 1 的处理流程

1.4 控制流图的组合规则

按上述算法可得到构成被测程序的所有基本单元,前面定义了基本单元组合生成控制流图包含顺序组合和嵌套组合两种方式,下面分析基本单元图组合形成控制流图的组合规则。

由于采用了程序的行号作为节点编号,因此可得如下两条结论:

结论 1 由基本单元图绘制的控制流图中,最小编号节点 s_{min} 即为被测程序控制流图的源节点;最大编号节点 e_{max} 即为被测程序控制流图的汇节点。

结论 2 除最小子源节点外,所有基本单元图的子源节点都以顺序组合或嵌套组合的方式与比其节点编号小且与其最近的子分支节点或子汇节点相连接。

由上述结论,可得出直接由基本单元图组合生成被测程序的控制流图的新方法,即基本单元组合法。组合规则如下:

1)组合节点对的选择规则。

由结论 2 得出组合对象的选择方法:假设 g_i 为任意一个不含最小子源节点的基本单元,其子源节点为 s_i,一定存在一个基本单元 g_j,其子分支节点 b_{jx} 或子汇节点 e_j 与 s_i 最近,则 g_i 与 g_j 为一对有组合关系的对象,存在组合节点对(s_i,e_j)或(s_i,b_{jx}),b_{jx} 属于子分支节点集 B_j。组合对象的选择即是寻找满足条件的组合节点对。

2)组合方式的选择规则。

对于组合对象 g_i 和 g_j,如果与 g_i 的子源节点最近的节点为 g_j 的子分支节点,则将 g_i 插入 g_j 的子分支节点与子汇节点之间,即为嵌套组合;如果与 g_i 的子源节点最近的节点为 g_j 的子汇节点,则将 g_i 接续到 g_j 的子汇节点之后,即为顺序组合。

3)组合顺序的确立规则。

各基本单元相互组合的先后顺序不影响组合形成的控制流图,但为了简化后续基本路径集的求解算法,采用先嵌套组合后顺序组合的顺序,按照组合节点对及其组合方式对所有基本单元进行组合,最后形成被测程序的控制流图。

2 基本路径组合法

基本单元组合得到的控制流图为测试人员提供了直观的图形化依据,便于辅助测试用例设计,但测试用例设计需要的是基本路径集。通过分析发现,利用基本单元的组合原理,将基本单元组合过程演变为基本路径的组合,这样能直接生成被测程序的基本路径集。与基本单元组合不同的是,基本路径组合首先从被测程序提取的不是基本单元,而是基本单元包含的路径。为便于描述,首先给出如下定义。

定义 4 设被测程序的第 i 个基本单元(g_i)包含 n 条基本路径,将这 n 条基本路径组成的集合称为基本子路径集。记作 Pg_i,$Pg_i=\{pg_{i1},pg_{i2},\cdots,pg_{in}\}$,$pg_{ij}(1\leqslant j\leqslant n)$ 为一条基本子路径。

如何从被测程序中提取出被测程序的基本子路径集是求解基本路径集的关键。

2.1 基本子路径集的求解算法

在 1.3 节的过程②中,增加对退栈信息的路径编辑处理,即可获得被测程序的基本子路径集。以例 1 为例,即根据基本单元信息{1s,2b,4e}、{7s,8b,12b,14e}、{5s,6b,17b,19e}可对应得到基本子路径集{1→4,1→2→4}、{7→8→14,7→12→14}、{5→6→19,5→17→19},增加路径编辑处理后,源代码扫描结束,生成的基本子路径集含 6 条子路径。

2.2 基本路径集的组合算法

求得基本子路径集后,可根据基本单元组合规则的原理

实现由基本子路径组合生成被测程序的基本路径集。下面讨论算法的实现,并结合例 1 进行验证。

1)组合节点对的选择。从基本子路径集中寻找最近的子源节点和子汇节点或最近的子源节点和子分支节点作为组合节点对。例 1 得到的子路径集中满足条件的组合节点对为(4,5)和(6,7)。

2)组合方式的判定。根据组合节点对的节点类型确定组合方式。如果节点对为子汇节点和子源节点,为顺序组合;如果节点对为子分支点和子源节点,为嵌套组合。例 1 中(4,5)为顺序组合节点对,(6,7)为嵌套组合节点对。

3)路径组合的实现。参照基本单元组合规则,先进行嵌套组合,再进行顺序组合。

a)嵌套组合的实现。对所有嵌套组合节点对,假设为(s_i,pb_j),由于此时还未进行路径顺序组合,故含 pb_j 的基本子路径只有一条,将含 pb_j 的子路径与含 s_i 节点的所有路径进行组合,组合方式为:将含 s_i 节点的所有子路径插入含 pb_j 子路径的子分支节点和子汇节点之间,得到组合形成的新子路径,然后将已参与组合的子路径删除。例 1 中的(6,7)节点对组合后得两条新子路径{5→6→7→8→14→19,5→6→7→12→14→19}。

嵌套组合包含一个特例(如图 1(c)),如果分支中含有 do…while 结构中的节点 b_1',它与同一条路径中 b_1 同时参与组合,随机从 Ps_i 中选一条路径进行组合,这不影响组合路径数且满足覆盖要求。

b)顺序组合的实现。顺序组合采用随机组合的方式,若组合节点对为(s_i,e_j),Ps_i 和 Pe_j 分别表示含 s_i 和 e_j 的基本子路径集合,$P(s_i)$和 $P(e_j)$分别为含 s_i 和 e_j 的子路径数,二者组合后的最小基本路径数为 $\max(P(s_i),P(e_j))$,假设 $P(s_i)>P(e_j)$,则将 Ps_i 中每一条路径与 Pe_j 中路径随机组合,保证 Pe_j 中每条路径至少参与组合一次,组合方式为:将含 s_i 节点的子路径接续到含 e_j 节点子路径的 e_j 节点之后,得到组合形成的新子路径,然后将已参与组合的子路径删除。例 1 中的(4,5)随机组合,P(4)=2,P(5)=3,故基本路径数为 max(P(4),P(5))=3,随机组合的结果不是唯一的,一个可能的结果是以下 3 条基本路径:

1→4→5→17→19

1→4→5→6→7→8→14→19

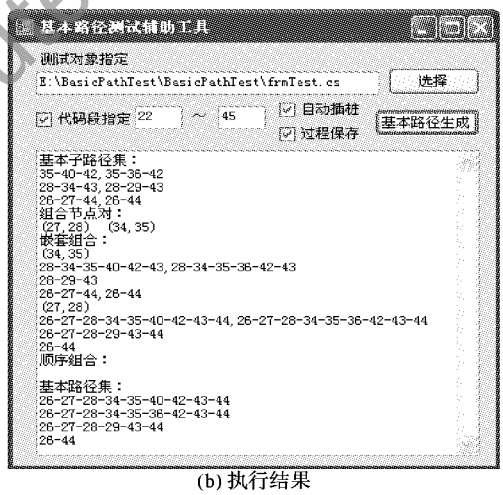
1→2→4→5→6→7→12→14→19

2.3　算法实现

根据上述算法的思想,本研究设计了一个 GUI 基本路径辅助测试工具,对图 5(a)所示的代码段求解基本路径集,工具执行结果如图 5(b)。工具实现了对指定代码文件某一区间代码的基本路径集求解,同时还可在扫描程序获得基本子路径的同时对源代码进行插桩,工具将路径组合过程输出到画面,也可勾选将组合过程信息保留到文本文件中供辅助测试用例设计参考。

```
22        private void sort(int iRecordNum, int itype)
23        {
24            int x = 0;
25            int y = 0;
26            while (iRecordNum > 0)
27            {
28                if (itype == 0)
29                {
30                    x = y + 2;
31                    iRecordNum = 0;
32                }
33                else
34                {
35                    if (itype == 1)
36                    {
37                        x=y+10;
38                    }
39                    else
40                    {
41                        x = y + 20;
42                    }
43                }
44            }
45        }
```

(a) 示例代码



(b) 执行结果

图 5　算法实现示例图

本文在 VS2005 平台采用 C#编程实现了该算法,并对嵌套和顺序组合混合的 C#伪代码进行了测试;处理 50 行 3 条件、100 行 6 条件、1 000 行 60 条件的伪代码,完成基本子路径集生成、基本路径集求解和生成插桩版程序的平均运行时间分别为 0.780 0 ms、1.268 7 ms、10.396 2 ms,设被测程序的规模(总代码行数及条件语句数)为 n,可得该算法的时间复杂度 $T(n)$ 为 $O(n)$。

2.4　基本路径组合法与 McCabe 法的对比分析

采用 McCabe 法和基本路径组合法分别对例 1 及图 5(a)所示的代码进行基本路径测试用例设计,二者所得结果是一致的,但两种方法存在较大的差别。主要体现在以下 3 个方面。

1)基本路径组合法可操作性好。

采用基本路径组合法设计测试用例,测试人员不必再绘制控制流图,减少了测试人员的工作量。

作为对比,McCabe 方法可操作性略显不足。因为 McCabe 法是根据程序控制流图确定基本路径集,而编程人员很少在编程同时给出程序控制流图,因此,测试用例设计人员的工作量较大。

2)基本路径组合法设计效率高。

基本路径组合法生成基本路径集只需要两步:

第 1 步　分析程序结构得基本子路径集;

第 2 步　对基本子路径进行组合生成基本路径集。

这两步均易于用算法实现,因此,能较大地提高测试用例设计效率。

而 McCabe 法生成基本路径集需要 3 个步骤:

第 1 步　分析程序结构绘制控制流图;

第 2 步　根据控制流图计算环路复杂度;

第 3 步　依据控制流图生成基本路径集。

对于较复杂的程序,需要花费较长的设计时间,设计效率比较低。

3)基本路径组合法算法实现简单。

在基本路径组合法中,采用堆栈数据结构即可实现第 1 步,即分析程序结构得基本子路径集。第 2 步实际上是按组合方式进行字符串的组合,因此,只需要采用基本的程序设计技术就能实现。

比较而言,McCabe 法算法实现较难,虽然文献[1-2]用算法实现了 McCabe 方法的 3 个步骤,但文献[1]不支持多分支结构,文献[2]采用了符号测试法选择基本路径集,执行效率低下,目前为止还没有算法能较完整地实现 McCabe 提出的方法。

3　结语

本文提出了一种基本路径测试用例设计的新算法,该算法有别于 McCabe 的思路,从被测程序的结构出发,只需要一次扫描被测程序得到其包含的基本子路径集,最后,按照组合规则将这些子路径进行组合,即可得到被测程序的基本路径集。目前,本研究实现了针对 C#代码的基本子路径集生成、基本路径集求解和基于基本路径测试的程序插桩版的生成。在后续的研究中,将借助于该算法的思想,进一步以工具的形式实现基本路径测试全过程的自动化,包括基本路径集的自动生成、测试数据的自动生成、测试用例的自动执行及测试报告的自动生成。本次讨论的对象没有考虑条件拆分的情况,也将在后续研究中完善。

参考文献:

[1] WIJAYASIRIWARDHANE T K, WIJAYARATHNA P G, KARUNARATHNA D D. An automated tool to generate test cases for performing basis path testing [C]// ICTer 2011: Proceedings of the 2011 International Conference on Advances in ICT for Emerging Regions. Piscataway: IEEE Press, 2011: 95 - 101.

[2] 严俊,郭涛,阮辉,等. JUTA: 一个 Java 自动化单元测试工具[J]. 计算机研究与发展,2010,47(10):170 - 178.

[3] 解圣霞. 基于基本路径测试的程序图自动生成的应用研究[J]. 通化师范学院学报,2009,30(12):38 - 41.

[4] ZHANG Z L, MEI L X. An improved method of acquiring basis path for software testing [C]// Proceedings of the 5th International Conference on Computer Science and Education. Piscataway: IEEE Press, 2010: 1891 - 1894.

[5] 王冠,景小宁,王彦军. 基本路径测试中的 McCabe 算法改进与应用[J]. 哈尔滨理工大学学报. 2010,15(1):48 - 51.

[6] 毛澄映,卢炎生. 分支测试中测试路径用例的简化生成方法[J]. 计算机研究与发展,2006,43(2):321 - 328.

[7] DU Q F, DONG X. An improved algorithm for basis path testing [C]// Proceedings of the 2011 International Conference on Business Management and Electronic Information. Piscataway: IEEE Press, 2011: 175 - 178.

[8] 杜庆峰,李娜. 白盒测试基本路径算法[J]. 计算机工程,2009,35(15):100 - 102.

[9] McCABE T J. A complexity measure [J]. IEEE Transactions on Software Engineering, 1976, SE-2(4): 308 - 320.

[10] 佟伟光. 软件测试[M]. 北京: 人民邮电出版社,2008:60 - 61.

[11] LUN L J, CHI X. Path numbers analysis of relationships on software architecture testing criteria [C]// Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering. Piscataway: IEEE Press, 2010: 118 - 122.

[12] 王培崇,钱旭. 基于改进鱼群算法的路径测试数据生成[J]. 计算机应用,2013,33(4):1139 - 1141.

[13] 施冬梅. 基本路径覆盖测试探针插桩技术研究[J]. 计算机工程与设计,2010,31(13):3025 - 3028.

[14] YANGU Y J, LUN L J, CHI X. Research on path generation for software architecture testing matrix transform-based [C]// CSSS 2011: Proceedings of the 2011 International Conference on Computer Science and Service System. Piscataway: IEEE Press, 2011: 2483 - 2486.

(上接第 3246 页)

参考文献:

[1] RIVEST R L, SHAMIR A, ADLEMAN L. A method for obtaining digital signatures and public key crypto systems [J]. Communications of the ACM, 1978, 21(2): 120 - 126.

[2] BONEH D. Twenty years of attacks on the RSA cryptosystem [J]. Notices of the AMS, 1999, 46(2): 203 - 212.

[3] LENSTRA A K. Integer factoring [J]. Designs Codes and Cryptography, 2000, 19: 101 - 128.

[4] SHAMA S, SHAMA P, DHAKAR R S. RSA algorithm using modified subset sum cryptosystem [C]// ICCCT-2011: Proceedings of the 2011 International Conference on Computer and Communication Technology. Piscataway: IEEE Press, 2011: 457 - 461.

[5] BELLARE M, NEVEN G. Identity based multi signatures from RSA [C]// Proceedings of the 7th Cryptographers' Track at the RSA Conference on Topics in Cryptology. Berlin: Springer-Verlag, 2007: 145 - 162.

[6] 司光东,杨加喜,谭示崇,等. RSA 算法中的代数结构[J]. 电子学报,2011,39(1):242 - 246.

[7] 裴东林,胡建军,李旭. RSA 算法中$\mathbf{Z}_{\varphi(n)}^{*}$的代数结构研究[J]. 计算机工程,2013,39(2):145 - 149.

[8] 闵嗣鹤,严士健. 初等数论[M]. 3 版. 北京: 高等教育出版社,2003.

[9] RIZOMILIOTIS P. On the resistance of Boolean functions against algebraic attacks using univariate polynomial representation [J]. IEEE Transactions on Information Theory, 2010, 56(8): 4014 - 4024.

[10] KNUDSEN L R, MIOLANE C V. Counting equations in algebraic attacks on block ciphers [J]. International Journal of Information Security, 2010, 9(2): 127 - 135.

[11] GHOSH S, DAS A. An improvement of linearization-based algebraic attacks [C]// Proceedings of the First International Conference on Security Aspects in Information Technology. Berlin: Springer-Verlag, 2011: 157 - 167.

[12] 谢佳,王天择. 寻找布尔函数的零化子[J]. 电子学报,2010,38(11):2686 - 2690.

[13] 李昕,林东岱. 对 Bivium 流密码的变元猜测代数攻击[J]. 电子学报,2011,39(8):1727 - 1732.