控制流图上支配关系计算方法的分析与实现

马红途[1,2] 赵荣彩[1] 苏彦兵[2]
(信息工程大学信息工程学院 郑州 450002)[1] (信息工程技术研究所 北京 102249)[2]

摘 要 支配关系在数据流分析和静态单赋值等程序分析和优化中应用很广泛。采用位向量表示支配结点集合,描述了采用迭代法计算控制流图上支配结点集合的算法,在支配结点集合的基础上讨论了对直接支配结点、支配边界结点的计算方法,并在NPB和SPEC2000测试集上进行了测试。测试结果表明:控制流图的构建占用了过程内支配关系计算的几乎一半时间;对于不包含goto语句的结构化程序,迭代算法一般只需迭代2次。

关键词 控制流图,迭代算法,位向量,支配关系

Analysis and Implementation of the Computation of Dominator in CFG

MA Hong-tu[1,2] ZHAO Rong-cai[1] SU Yan-bing[2]
(College of Information Engineering, University of Information Engineering, Zhengzhou 450002, China) [1]
(Institute of Information Engineering, Beijing 102249, China) [2]

Abstract Compilers use dominance information extensively during program analysis and optimization, such as data-flow analysis and the computation of static single-assignment forms. The iterative method to compute dominators was discussed in detail, in which dominator relation is represented by bit vector. Based on the result of dominators, the method to compute immediate dominator and dominator frontier was given. At last, experimental results of NPB and SPEC2000 were presented, which show that construction of intraprocedual CFG takes almost half of the total time; if goto statement is not contained in structural program, the iterative algorithm will finish in two iterations.

Keywords CFG, Iterative formulation, Bit vector, Dominator

1 引言

1959年,Prosser[5]给出了支配关系的定义:

箱子i支配箱子j是指在流图上从入口到出口的路径上,每条通过箱子j的路径必定经过箱子i。

支配关系信息在程序的分析和优化过程中应用很广泛,例如,数据流分析[1]、循环检测、结构化分析[6]、调度[7]、计算依赖关系图及静态单赋值形式[4]、检测VLSI电路中对等线的错误对[8]等。支配集的计算方法主要有图的可达性方法及迭代法。

图的可达性方法就是考虑从入口点到结点b的所有路径,不断从这些路径上移除某个结点,如果移除某个结点后,b变得不可达,那么该结点就在b的支配结点集合中。Lowry和Medlock[9]基于图的可达性给出了计算支配关系的算法,算法的时间复杂性至少是$O(N^2)$。Purdom和Moore[16]提出的改进方法是在图上构建一棵生成树,然后不断移除结点,如果树中的某个孩子不可达,那么这个结点就是孩子结点的支配结点,算法的时间复杂性也是$O(N^2)$。

迭代法把支配关系用数据流等式来表示,然后基于控制流图求最大不动点的解。Allen[10]采用数据流方法来求解支配集,并对支配集问题给出了数据流等式,并提出了一个迭代的方法来求解支配关系集合[17],其时间复杂度为$O(N^2)$。Hecht和Ullman[11]提出一种基于逆向后序遍历顺序进行迭代求解的算法,在可化简的图上,该迭代方法可以在线性时间内求解。

迭代算法中比较著名的算法是Tarjan[12]提出的一个使用深度优先搜索和union-find的方法,该算法基于如下观察结论:一个结点的支配者在深度优先生成树上则一定在该结点之上,算法的复杂度为$O(N\log N+E)$,算法的效率依赖于union-find操作的效率。

Buchsbaum[18]基于Lengauer-Tarjan的算法提出了一个线性时间的分而治之方法,也就是把深度优先搜索树上的叶结点组合成子树,通过对子树求解,可以在线性时间进行union-find操作。

Georgiadis[3]比较了Cooper[2]提出的基于树的迭代算法,Lengauer-Tarjan算法和一个混合的SEMI-NCI方法。测试结果表明这些算法的性能比较类似,但是当图比较大时,Lengauer-Tarjan算法和混合方法比较快。

本文的主要贡献在于:1)采用位向量实现了支配结点、直接支配结点及支配边界结点的计算方法;2)验证了迭代算法的有效性,并依据测试结果提出控制流图的化简对于减少过程内支配关系算法的时间至关重要。本文第1部分给出了控

到稿日期:2008-04-16 本文受国家863计划资助项目(2006AA01Z408)资助。
马红途(1976-),男,博士研究生,工程师,主要研究方向为并行与高性能计算等,E-mail:mahongtoo@hotmail.com;**赵荣彩** 博士生导师,教授,主要研究方向为并行与高性能计算和计算机网络安全等;**苏彦兵** 博士,高级工程师,主要研究方向为信息通信等。

制流图的构建和化简算法；第 2 部分对支配结点集合算法、直接支配集合算法及支配边界结点集合算法进行了描述，并分析了算法的复杂性；第 3 部分针对实例描述了支配结点的计算过程；第 4 部分对 NPB 和 SPEC2000 测试集进行了测试，试验结果表明迭代算法一般只需迭代 2 次即可求得支配结点集合；第 5 部分针对结构化程序的支配关系计算提出了改进意见，并介绍下一步工作。

2 控制流图的构建及化简

一个过程内的控制流图是一个有向图，可以用 $CFG_p=(N_p,E_p,Entry(p),Exit(p))$来表示，其中，$N_p$ 表示结点的集合，与程序中语句组成的基本块相对应，E_p 表示结点间控制流的有向边，$Entry(p)$表示图的入口结点，$Exit(p)$表示图的出口结点。在控制流图中任何结点 $n\in N_p$ 都有唯一的一个入口和出口，因此本文假设程序中没有跨过程的跳转语句。

控制流图的构建算法在一些论文[10,13-15,19]中有详细的论述，这里只讨论本文的控制流图构建过程中需注意的地方：

1. 函数调用点包含两个结点，调用点和返回点；

2. 跳转指令的处理，在构建过程中若遇到 label 指令，则把相应指令的结点指针附加到 label 符号上，整个图构建完成后，再依据跳转指令目的 label，取得相应的结点，添加边；

3. 入口点 Entry(p) 到出口点 Exit(p)之间添加相应的边；

4. 每个结点都有相应的顺序编号，表示其在控制流图上的先后关系。

控制流图构建完成后，应消除不可达结点，其算法为：

不可达结点的消除算法 unreachable_visit：

```
输入：控制流图结点 n，位向量 mark
输出：位向量 mark
在位向量 mark 中标记 n 可达；
遍历当前结点 n 的后继链；
    取得 n 的后继 succ；
    如果 succ 未遍历，调用 unreachable_visit(succ,mark)；
返回；
```

该算法第一次调用时 mark 被清零，mark 的每一个比特表示控制流图的一个结点，对函数入口点调用 unreachable_visit，当整个调用返回后，在 mark 中标记遍历过的结点即为可达结点，否则就可移除相应的结点。

3 支配关系及计算方法

3.1 支配结点集合

如果从入口点到结点 v 上的所有路径都通过 u，那么 u 就在 v 的支配结点集合中。如果 $u\neq v$，那么就称 u 严格支配 v。用 $dom(v)$ 表示 v 的支配结点集合，计算等式形如式(1)[2]。

$$dom(n0)=\{n_0\}$$

$$dom(v)=(\bigcap_{u\in pred(v)} dom(u))\cup\{v\},\forall v\in V-r \qquad (1)$$

迭代算法求解支配关系的算法思想：入口结点的支配结点集合 $dom(n_0)$为其自身，以此为迭代的初始条件，任何其他非入口结点的支配结点集合 $dom(v)$的初始支配集合为所有结点，再在后序遍历生成结点链上不断运用式(1)求解，如果在某次迭代过程中任何结点的支配结点集合都没有改变，则求解完成。

在描述迭代法求支配结点算法之前，定义三个数据结构：控制流图进行深度优先遍历生成控制流图结点链表；支配关系集合 $N\times N$ 的矩阵 D，矩阵 D 每一行表示了对应编号结点的支配结点集合；当前结点 X 的支配关系集合 T。

迭代法求解支配结点集合算法

```
把当前结点 X 置为入口结点；
D[X]←ϕ；
D[X]←X；
do {
    changed = FALSE；
    for 深度优先生成结点链上的每个结点 X；
        if X 为入口结点 then continue；
          初始化 X 的当前支配结点集合 T 为所有结点；
          for X 的前驱结点链上每个结点 P
              T = T 与 D[P]的交；
          end for
            把当前结点 X 添加到 T 中；
            if D[X]不等于 T then，
            D[X] = T
            changed = TRUE；
        end if
    end for
} while (changed)；
```

算法复杂性分析：在最好情况下，如果在控制流图中无环，那么两次迭代就可计算得到不动点信息，因此最好时间复杂度为 O($2N$)。若每次迭代只收敛一个结点，那么算法的最坏时间复杂度为 O(N^2)。

定理 在深度优先生成树上，n 层顶点至多在 $n+1$ 次迭代之前收敛。

证明：在深度优先生成树上，根据支配关系定义，入口顶点的后继结点的支配结点集合一定只包含入口顶点及其自身，那么在第一次迭代的时候入口顶点即可收敛。假设在深度优先生成树上，$n-1$ 层顶点会在 $n-1$ 次收敛。据支配关系定义，在第 n 次迭代中，n 层顶点的所有前驱的支配结点集合中都包含了收敛的 $n-1$ 层顶点的支配关系，所以在第 n 次迭代中，n 层顶点一定收敛。证毕。

因此，最坏情况下，一次只收敛一个顶点，最坏时间复杂性得证。

3.2 直接支配结点集合

$idom(v)$，顶点 v 的直接支配结点集合，也就是存在顶点 w，$w\neq v$，且 w 支配顶点 v，并且被 $dom(v)-v$ 中的顶点支配，那么 w 存在于 v 的直接支配结点集合中。

直接支配结点的基本思想是：如果在支配结点集合中移除结点 v 本身，遍历该结点的支配结点，如果某个支配结点的支配集合等于 $dom(v)-v$，则该支配结点为 v 的直接支配结点。

主要的数据结构有 N 元向量 ID，$ID[v]$表示结点 v 的直接支配结点。

直接支配结点算法

```
for 控制流图上的每个结点 X；
    if X 为入口结点 then continue；
      在 D[X]集合中移除 X；
    for X 的支配结点集合中的每个结点 P
      if D[X] == D[P] then
```

```
            ID[X]←P;
          End if
        end for
          把 X 添加到 D[X]集合中;
    end for
```

该算法的时间复杂度为 $O(N)$。

3.3 支配边界结点集合

$df(v)$，v 结点的支配边界结点集合，v 支配某个结点，但是不支配该结点的后继，则这个结点就是支配边界结点。其形如式(2)[4]：

$$df(x)=\{y|(\exists p\in Pred(y))((x\geqslant p) \text{ and } (x\not\geqslant y))\} \quad (2)$$

如果自上向下遍历控制流图，支配边界结点是控制流的汇合结点。

需定义的数据结构有 $N\times N$ 的矩阵 DF，矩阵 DF 每一维对应了相应结点的顺序编号，也就是每一维表示了其对应编号结点的支配结点集合。

支配边界结点算法 dom_frontier

```
输入：当前结点 X
输出：当前结点的支配边界结点集合
for 控制流图上的每个结点 Y;
    if X 是 Y 的直接支配结点 then
        dom_frontier(Y);
    endif
    end for
DF[X]←ϕ;
    for X 的后继结点链中的每个结点 S
        if X 不是 S 的直接支配结点 then
          把 S 添加到 DF[X]集合中
        end if
    end for
    for 控制流图上的每个结点 Y;
        if X 是 Y 的直接支配结点 并且 Y 的 DF 不空 then,
        for 控制流图上的每个结点 Z;
                if Y 的 DF 包含结点 Z,并且 X 不是 Z 的直接支配结点 then,
                        把 Z 添加到 DF[X]集合中
                end if
            end for
        end if
end for
```

与 cytron [4] 所提算法中的深度优先生成树类似，本算法通过在直接支配结点上的递归实现在支配树上的深度优先遍历。算法的时间复杂度为 $O(E+N^2)$。

4 实例分析

本节将通过例子来说明迭代分析法求支配结点集合的计算过程。设控制流图的孩子结点以从左至右的顺序存储，则深度优先生成树如图 1(b)所示。

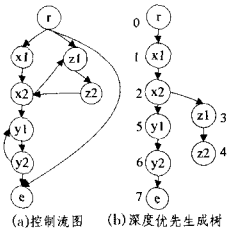

图 1 控制流图及其深度优先生成树

图 1(a)描述了控制流图中，结点支配关系的计算过程是：

第一次迭代：

第一步，初始结点状态，只包含了根结点的支配顶点关系：

$$\begin{array}{c} \\ r \\ x1 \\ x2 \\ z1 \\ z2 \\ y1 \\ y2 \\ e \end{array}\begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \end{bmatrix}$$

第二步，$x1$ 的支配结点集合为：

$$\begin{array}{c} \\ r \\ x1 \\ x2 \\ z1 \\ z2 \\ y1 \\ y2 \\ e \end{array}\begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0&0 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \end{bmatrix}$$

第三步，$x2$ 的支配结点集合为：

$$\begin{array}{c} \\ r \\ x1 \\ x2 \\ z1 \\ z2 \\ y1 \\ y2 \\ e \end{array}\begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0&0 \\ 1&1&1&0&0&0&0&0 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \end{bmatrix}$$

依此类推，第一次迭代结束后，整个控制流图的支配结点集合为：

$$\begin{array}{c} \\ r \\ x1 \\ x2 \\ z1 \\ z2 \\ y1 \\ y2 \\ e \end{array}\begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0&0 \\ 1&1&1&0&0&0&0&0 \\ 1&0&0&1&0&0&0&0 \\ 1&0&0&1&1&0&0&0 \\ 1&1&1&0&0&1&0&0 \\ 1&1&1&0&0&1&1&0 \\ 1&0&0&0&0&0&0&1 \end{bmatrix}$$

在第一次迭代中，$x2$ 的支配关系集合计算不精确，因此 $x2$ 的支配关系集合需要在下一次迭代中进一步精确。

第二次迭代后整个控制流图的支配结点集合为：

$$\begin{array}{c} \\ r \\ x1 \\ x2 \\ z1 \\ z2 \\ y1 \\ y2 \\ e \end{array}\begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0&0 \\ 1&0&1&0&0&0&0&0 \\ 1&0&0&1&0&0&0&0 \\ 1&0&0&1&1&0&0&0 \\ 1&0&1&0&0&1&0&0 \\ 1&0&1&0&0&1&1&0 \\ 1&0&0&0&0&0&0&1 \end{bmatrix}$$

据支配结点集合定义知，第二次迭代后即求得整个控制流图的支配顶点集合，但是整个算法需迭代 3 次才能终止。

5 测试结果

本文所述算法已经基于 KAP 系统进行了实现，为了验证和分析本文所述算法，针对 NPB-omp-C 2.3 版本和 SPEC2000 进行了测试。测试平台为 Intel Pentium IV 2.4 GHZ，256MB 内存，操作系统为 RedHat Linux 8.0，内核版本号为(2.4.18-14)。

在测试过程中，分别记录了控制流图构建时间，支配结点集合、直接支配结点集合平均计算时间，迭代算法的平均迭代次数和最大迭代次数，以及支配边界结点集合平均计算时间，测试结果如表 1 所示。

表 1 测试数据

测试实例	图的个数	平均结点个数	图平均构建时间(ms)	支配和直接支配计算			支配边界计算	总的平均运行时间
				平均迭代次数	最多迭代次数	平均时间(ms)	平均时间(ms)	(ms)
NPB-omp-C 2.3								
BT	28	101	1.64	2	2	0.84	0.95	3.86
CG	7	85	0.93	2	2	0.43	0.59	2.16
EP	1	229	0.0027	2	2	0.0023	0.0022	0.0079
FT	17	47	0.42	2	2	0.18	0.19	0.89
IS	5	59	0.6	2	2	0.26	0.26	1.28
LU	18	137	2.0	2	2	1.03	1.28	4.86
MG	15	67	0.77	2	2	0.33	0.42	1.73
SP	22	140	2.12	2	2	1.07	1.42	5.24
SPEC2000-INT								
164.gzip	106	48	0.45	2	2	0.23	0.26	1.02
175.vpr	300	41	0.39	2	2	0.18	0.20	0.87
176.gcc	2244	89	2.38	2	3	1.49	2.41	7.03
181.mcf	26	47	0.39	2	2	0.18	0.17	0.83
197.parser	324	37	0.38	2	2	0.20	0.21	0.89
255.votex	923	51	0.49	2	2	0.27	0.36	1.30
256.bzip2	74	43	0.46	2	2	0.26	0.28	1.12
300.twolf	191	115	2.51	2	2	1.28	1.80	6.59

在表 1 中所测数据没有考虑过程间的控制流图，因此针对每个函数构建一个控制流图，表中所列数据都是整个程序的运算时间在这些所有控制流图上的平均。观测实验结果数据可以发现两点现象：1)控制流图构建及化简时间几乎占用了整个运算时间的一半；2)迭代法计算支配关系集合的平均迭代次数为 2 次。

结束语 本文基于前人关于支配关系算法的研究成果，对支配关系的迭代算法进行了分析和实验。通过实验得出两点结论：1)由于现在的大部分程序采用结构化设计方法，其控制流图比较简单，一般通过一次迭代即可求得支配关系集合；2)分析结构化程序过程中，可以尽量简化控制流图，减少结点数目，以期减少构建和计算时间。Cooper[2] 提到采用位向量计算支配关系集合的交操作比较占用时间，但是他并没有提及控制流图构建和化简所占用的时间比。在下一步工作中，作者将主要考虑在过程间的支配结点集合的计算[1]。由于过程间结点比较多，控制流图的化简和 Cooper 所提及的交操作的优化将成为重点研究问题。

参 考 文 献

[1] Sathyanathan P W. Interprocedural Dataflow Analysis - Alias Analysis. Ph. D thesis. Stanford University, 2001

[2] Cooper K D, Harvey T J, et al. A Simple, Fast Dominance Algorithm. Http://www.cs.rice.edu/~keith/EMBED/dom14.pdf, 1999

[3] Georgiadis L, Werneck R F, et al. Finding Dominators in Practice. www.cs.princeton.edu/~lgeorgia/dominators_esa04.pdf, 2004

[4] Cytron R, Ferrante J, Rosen B K, et al. Efficiently Computing static single assignment form and the control dependence graph. ACM Transactions on Programming Languages and Systems, 1991, 13(4): 451-490

[5] Prosser R. Applications of Boolean matrices to the analysis of flow diagrams // Proceedings of the Eastern Joint Computer Conference. Spartan Books, NY, USA, Dec. 1959: 133-138

[6] Sharir M. Structural analysis: A new approach to flow analysis in optimizing compilers, 1980, 5: 141-153

[7] Sweany P H, Beaty S J. Dominator - path scheduling: A global scheduling method // Proceedings of the 25th International Symposium on Microarchitectrue. 1992: 260-263

[8] Amyeen M E, Fuchs W K, et al. Fault equivalence identification using redundancy information and static and dynamic extraction // Proceedings of the 19th IEEE VLSI Test Symposium. March 2001

[9] Lowry E, Medlock C. Object code optimization. Communications of the ACM, Jan. 1969: 13-22

[10] Allen F E. Control flow analysis. SIGPLAN Notices, 1970, 5(7): 1-19

[11] Hecht M S, Ullman J D. A simple algorithm for global data flow analysis problem. SIAM J. Comput., 1975, 4(4): 519-532

[12] Tarjan R E. Testing flow graph reducibility. J. Comput. Syst. Sci., 1974, 9: 355-365

[13] Sinha S, Harrold M J, et al. Interprocedural control dependence. ACM SIGSOFT, 2001, 10(2): 209 - 254

[14] Aho A V, Sethi R, et al. Compilers: Principles, Techniques, and Tools. Addison-Wesley, 1986

(下转第 77 页)

与树的结构密切相关。如同 Zhiguo Wan 等人对 nPAKE+协议的执行效率分析，该协议的幂计算代价也在 O(log$n$)。协议的计算成本如表 1 所示。

表 1　计算成本

	域 A 中客户	KDC_A	KDC_B	域 B 中客户
指数运算	$n(7+\log n)$	$4n+2$	$3m+2$	$m(5+\log m)$

结束语　本文所提出的协议将 Zhiguo Wan 等学者提出的适用单域中的 nPAKE$^+$ 协议扩展到两个域中，该协议是一个基于不同口令认证的跨域间的组密钥交换协议，实现了两个域中的客户组在域服务器的协助下建立域间共享的组会话密钥的过程，其中每个用户都与一个可信任的服务器共享一个独立的密码。该协议在组密钥生成和协商时使用了 Diffie-Hellman 密钥树，因此在计算和通信代价方面具有更高的效率。通过对该协议的安全分析和执行效率的分析，表明该协议是安全高效的。

参 考 文 献

[1] Bellovin S, Merrit M. Encrypted key exchange: password based protocols secure against dictionary attacks // Proceedings of the Symposium on Security and Privacy. IEEE, 1992: 72-84

[2] Steiner M, Tsudik G, Waider M. Refinement and extension of encrypted key exchange // ACM Operation Sys. Review, 1995, 29 (3): 22-30

[3] Ding Y, Horster P. Undetectable on-line password guessing attacks. ACM Operating Systems Review, 1995, 29(4): 77-86

[4] Lin C, Sun H, Hwang T. Three-party encrypted key exchange: attacks and a solution. ACM Operating Systems Review, 2000, 34(4): 12-20

[5] Lin C, Sun H, Steiner M, et al. Three-party Encrypted Key Exchange Without Server Public-Keys. IEEE Communications Letters, IEEE Press, 2001, 5(12): 497-499

[6] Byun J W, Jeong I R, Lee D H, et al. Password-authenticated key exchange between clients with different passwords // Proceedings of ICICS'02. LNCS Vol. 2513. Springer-Verlag, 2002: 134-146

[7] Chen L. A Weakness of the Password - Authenticated Key Agreement between Clients with Different Passwords Scheme // The document was being circulated for consideration at the 27th the SC27/WG2 meeting. Paris, France, 2003-10-20/24

[8] Kim J Y, Kim S J, Kwak J, et al. Cryptanalysis and Improvement of Password Authenticated Key Exchange between Clients with Different Passwords // ICCSA 2004. LNCS 3043. 2004: 895-902

[9] Denning D, Sacco G. Timestamps in key distribution protocols. Communications of the ACM, 1981, 24 (8): 533-536

[10] YoonEn-J, Yoo K-Y. A Secure Password - Authenticated Key Exchange Between Clients with Different Passwords // APWeb Workshops 2006. LNCS 3842. Berlin Heidelberg: Springer-Verlag, 2006: 659-663

[11] Byun J W, Lee D H, Lim J. EC2C-PAKA: An efficient client-to-client password-authenticated key agreement

[12] Gang Y, Dengguo F, Xiaoxi H. Improved Client-to-Client Password-Authenticated Key Exchange Protocol // IEEE ARES 2007. 2007: 564-574

[13] Phan R C W, Goi B M. Cryptanalysis of an Improved Client-to-Client Password-Authenticated Key Exchange (C2C-PAKE) Scheme // Ioannidis J, Keromytis A D, Yung M, eds. ACNS 2005. LNCS, vol. 3531. Heidelberg: Springer, 2005: 33-39

[14] Yoneyama K, Ota H, Ohta K. Secure Cross-Realm Client-to-Client Password-Based Authenticated Key Exchange Against Undetectable On-Line Dictionary Attacks // AAECC 2007. LNCS 4851. Berlin Heidelberg: Springer-Verlag, 2007: 257-266

[15] Bresson E, Chevassut O, Pointcheval D. Group Diffie - Hellman Key Exchange Secure against Dictionary Attacks // Zheng Y, ed. ASIACRYPT 2002. LNCS, vol. 2501. Heidelberg: Springer, 2002

[16] Byun J W, Lee D H. N-Party Encrypted Diffie-Hellman Key Exchange Using Different Passwords // Ioannidis J, Keromytis A D, Yung M, eds. ACNS 2005. LNCS, vol. 3531. Heidelberg: Springer, 2005: 75-90

[17] Tang Q, Chen L. Weaknesses in two group Diffie-Hellman Key Exchange Protocols. Cryptology ePrint Archive, 2005, 197

[18] Byun J W, Lee D H, Lim J. Password - based Group Key Exchange Secure Against Insider Guessing Attacks // Proceedings of CIS'05. LNAI Vol. 3802. Springer-Verlag, 2005: 143-148

[19] Wan Zhiguo, Deng R H, Bao Feng, et al. nPAKE$^+$: A Hierarchical Group Password-Authenticated Key Exchange Protocol Using Different Passwords // ICICS 2007. LNCS 4861. Berlin Heidelberg: Springer-Verlag, 2007: 31-43

[20] Bresson E, Chevassut O, Pointcheval D, et al. Provably authenticated group diffie-hellman key exchange // Proceedings of 8th ACM Conference on Computer and Communications Security. 2001: 255-264

(上接第 57 页)

[15] Holloway G, Smith M D. The Machine - SUIF Control Flow Graph Library. http://www.eecs.harvard.edu/hube/software/nci/cfg.pdf, 2002

[16] Paul J, Purdom W, Moore E F. Immediate predominators in a directed graph. Communications of the ACM, 1972, 15(8): 777-778

[17] Allen F E, Cocke J. Graph-theoretic constructs for program flow analysis. Technical Report RC 3923 (17789). IBM Thomas J. Watson Research Center, July 1972

[18] Buchsbaum A L, Kaplan H, et al. Linear - time pointer - machine algorichms for least common ancestors, mst Verification, and dominztors // Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing. 1998: 279-288

[19] Zhao Kejia. GCC 基本块与控制流程图的数据结构分析. Technical Report. Creative Compiler Research Group, 2002