一种 Piccolo 加密算法硬件优化实现研究*

李 浪[1,2,3], 刘波涛[1,3], 余孝忠[1,3], 贺位位[1,3], 李仁发[2]
(1. 衡阳师范学院 计算机科学系, 湖南 衡阳 421002; 2. 湖南大学 信息科学与工程学院, 长沙 410082; 3. 聚落文化遗产数字化技术与应用湖南省重点实验室, 湖南 衡阳 421002)

摘 要: Piccolo 轻量级密码算法是近年来物联网领域提出的重要安全算法之一。研究了一种 Piccolo 加密算法硬件优化实现方法,相同的轮运算只实现一次;原始算法共有 r 轮运算,其中密钥为 80 和 128 位时,r 分别取值为 25 和 31。优化方法直接把 $r-1$ 轮重复调用变成 r 轮循环运算,同时在 r 轮循环运算完成后构造一个 RP^{-1} 轮置换函数。实验表明优化后的 Piccolo-80 密码算法在面积上少了 3227 个 Slices,优化效率达到 24.6%,有效节省了硬件实现面积,同时加密速率提高了 10%。

关键词: Piccolo; 轻量级密码算法; 优化; FPGA 实现
中图分类号: TP309.7 **文献标志码**: A **文章编号**: 1001-3695(2015)10-3056-04
doi:10.3969/j.issn.1001-3695.2015.10.041

Research on hardware optimization implementation of Piccolo

Li Lang[1,2,3], Liu Botao[1,3], Yu Xiaozhong[1,3], He Weiwei[1,3], Li Renfa[2]
(1. *Dept. of Computer Science, Hengyang Normal University, Hengyang Hunan* 421002, *China*; 2. *College of Information Science & Engineering, Hunan University, Changsha* 410082, *China*; 3. *Hunan Provincial Key Laboratory for Technology & Application of Cultural Heritage Digitalization, Hengyang Hunan* 421002, *China*)

Abstract: Piccolo was the one of important lightweight cryptographic algorithms as IoT cipher in the recent years. This paper researched the hardware optimization method of Piccolo encryption algorithm. The same round operation was achieved only once and repeated call. The original algorithm of Piccolo had r round operation. The key size of Piccolo was 80-bit/128-bit, the corresponding round number was 25/31. The optimal method made the $r-1$ round repeated calling into the r round operation repeated calling. It constructed a RP^{-1} function after the r round operation. The experimental results show that the optimal method of Piccolo-80 can save 3227 Slices for hardware implementation. Optimize efficiency reached 24.6%. Encryption rate increased by 10%.

Key words: Piccolo; lightweight cryptographic algorithms; optimization; FPGA implementation

物联网近几年越来越深入到人们的生活中,物联网安全也引起了人们的高度关注。Piccolo 密码算法是在密码硬件与嵌入式系统国际顶级会议(CHES 2011)上提出的[1],该密码算法主要是为物联网下资源受限的 RFID tags 等加密应用而研发[2~5]。Piccolo 加密算法是一种轻量级的对称分组密码算法。通过对其加密算法与结构的深入研究,发现原算法仍然在加密效率与实现面积上可以改进。本文研究了对 Piccolo 算法的合理优化,可以有效节省实现面积,使之更适合在资源约束的物联网芯片上实现。

1 Piccolo 算法分析

1.1 符号定义

$\|$:连接符;
r:迭代轮数;
RP:轮置换;
RP^{-1}:逆轮转换;
K:密钥;
rk:轮密钥;
c_{i+1}:用五位二进制数表示的十进制数$(i+1)$;
ARK:轮密钥加;
wk:白化密钥;
AWK:白化密钥加;
con:轮常量。

1.2 Piccolo 简介

Piccolo 算法是一种轻量级分组加密算法,分组长度 64 位,密钥长度有 80 位和 128 位两种。其中密钥为 80 位时记做 Piccolo-80,迭代轮数 r 为 25 轮;密钥 128 位记做 Piccolo-128,迭代轮数 r 为 31 轮。算法采用非平衡型 Feistel 结构;每轮中,加密数据同子密钥都进行异或运算(AddRoundKey, ARK 操作)、F 函数运算与 RP 轮置换函数运算(其中最后一轮没有

收稿日期: 2014-07-29; **修回日期**: 2014-09-09 **基金项目**: 国家自然科学基金资助项目(61572174);湖南省自然科学基金资助项目(2015JJ4011);衡阳师范学院产学研基金资助项目(12CXYZ01);湖南省科技厅科技计划项目(2013FJ3077,2013SK3178,2014FJ3061);衡阳师范学院大学生研究性学习和创新性实验计划资助项目(CX1531);湖南省"十二五"重点建设学科资助项目;聚落文化遗产数字化技术与应用湖南省重点实验室开放基金资助项目(J1401Z)

作者简介:李浪(1971-),男,教授,硕导,博士,主要研究方向为嵌入式系统与信息安全(lilang911@126.com);李仁发(1957-),男,教授,博导,主要研究方向为嵌入式计算与信息安全.

RP 轮置换函数)[6~8]。其运算结构如图 1 所示。

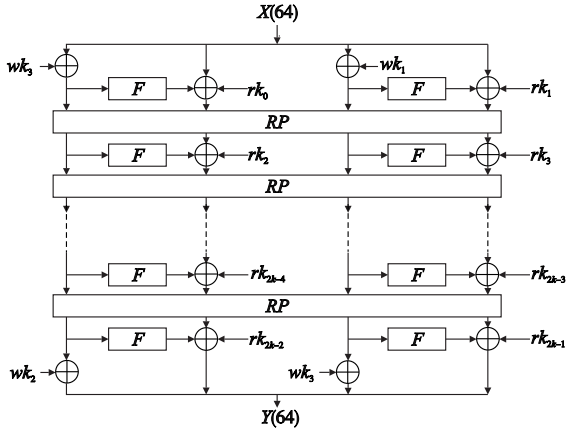

图1　Piccolo加密运算结构图

轮置换 *RP* 运算结构如图 2 所示。*RP* 函数置换运算将输入 64 位值划分为 8 Byte,然后进行字节的置换操作,将 *RP* 轮置换函数的 64 位输入数据从高到低依次划分为 8 Byte,进行图 2 所示运算输出 64 位数据。

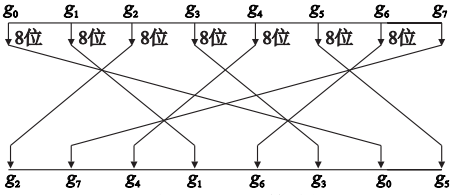

图2　轮置换*RP*运算结构图

Piccolo 密钥扩展分为 80 位和 128 位密钥长两种,分别叙述如下:

a)初始密钥 key 长度为 80 位时,

将初始密钥从高位开始按 16 位一组划分为五个部分。

白化密钥扩展表示为:$wk_0 \leftarrow k_0^L \mid k_1^R, wk_1 \leftarrow k_1^L \mid k_0^R, wk_2 \leftarrow k_4^L \mid k_3^R, wk_3 \leftarrow k_3^L \mid k_4^R$。

b)初始密钥 key 长度为 128 位时,

将初始密钥从高位开始按 16 位一组划分为八个部分,$i(0 \leqslant i < r)$。

白化密钥扩展:$wk_0 \leftarrow k_0^L \mid k_1^R, wk_1 \leftarrow k_1^L \mid k_0^R, wk_2 \leftarrow k_4^L \mid k_7^R, wk_3 \leftarrow k_7^L \mid k_4^R$。

当$(2i+2) \bmod 8 = 0$ 时,则按照$(k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7) \leftarrow (k_2, k_1, k_6, k_7, k_0, k_3, k_4, k_5)$进行轮密钥扩展;否则,依据$(k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7)$进行轮密钥扩展。

F 函数变换依次包括 S 盒变换、列混合变换及 S 盒变换,如图 3 所示。

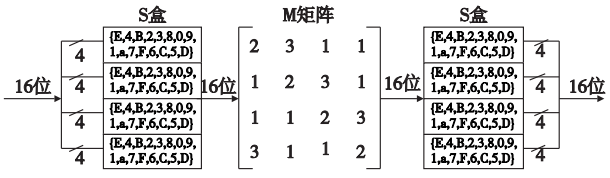

图3　*F*函数变换

Piccolo-80 算法是采用 24 + 1 轮运算,前 24 轮在轮函数硬件资源上可重复调用实现,最后一轮需要重新分配轮运算资源,因此迭代运算时不能连续 25 轮重复。Piccolo-128 算法,采用 30 + 1 轮运算,前 30 轮在轮函数硬件资源上可重复调用实现,最后一轮需要重新分配轮运算资源。Piccolo 原始算法中由于最后一轮运算需单独实现,这种方法不利于最大限度重复运行相同模块,从而增加芯片实现面积,同时占用实现资源。

1.3　Piccolo 存在的不足

Piccolo 密码算法面积优化实现最有效的方法是相同运算结构在硬件上只实现一次,然后进行重复调用。Piccolo-80 共有 25 轮运算,但只有 24 轮结构相同,最后一轮即第 25 轮结构与前面 24 轮不同,第 25 轮相比前 24 轮少一个轮置换(round permutation,RP),因此原始算法只能重复 24 轮,最后一轮要重新分配硬件资源,增加了实现面积,同时也增加了如存储空间资源占用[9]。

2　Piccolo 密码优化方法

不失一般性,以 Piccolo-80 为例进行优化方法的论述。

2.1　优化原理

首先直接进行 25 轮的重复调用,因此相对原算法就多了一个 *RP* 运算,然后在轮运算后增加一个 *RP* 逆运算进行结果的修正。由于 RP^{-1} 相比原来最后一轮的运算要简单得多,所以可节省实现面积,提高运行效率。优化后的 Piccolo 加密算法运算结构如图 4 所示。

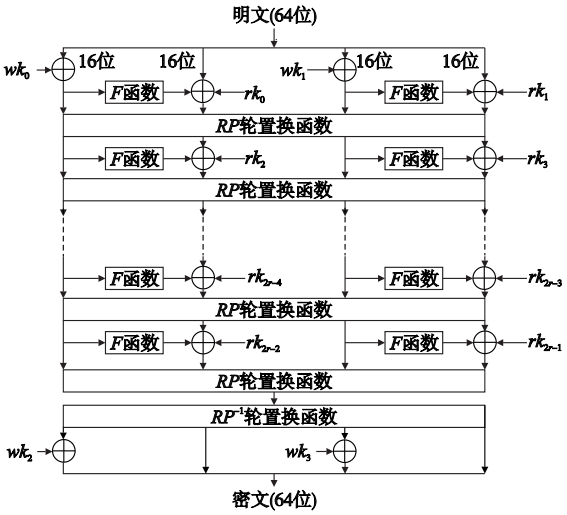

图4　Piccolo优化加密运算结构图

2.2　具体优化方法实现过程

Piccolo 加密运算包括以下模块:常数更新模块(updateConstant)、F 函数变换模块(function)、轮函数模块(PiccoloRound)、主控制模块(Piccolo)。

UpdateConstant 模块中包括四个端口,在 updateConstant 模块代码中:三个输入端口分别为初始密钥 key(80 位)、轮数 $i(0 \leqslant i < 25, i$ 为整数)模 5 取余数用 q(8 位)表示,常数 c_{i+1} 用 *c_i* 表示;一个输出端口为子密钥 *rk*。用连续赋值(assign)方式,通过常数 *c_i* 来构造 constant 参数,其中参数生成代码为

```
constant = {c_i[3:7],5'b00000,c_i[3:7],2'b00,c_i[3:7],
           5'b00000,c_i[3:7]}^32'hf1e2d3c
```

其中:5'b00000 表示 5 位宽的二进制数。将 q 的值作为选择初始密钥 key 的相应位与 constant 异或条件,得到最终结果 *rk*。

运算如下:

当 q 等于 0 或者 2 时:$rk = \{k_2, k_3\}$^constant;

当 q 等于 1 或者 4 时:$rk = \{k_0, k_1\}$^constant;

当 q 等于 3 时:$rk = \{k_4, k_4\}$^constant。

Function 模块(*F* 函数)中包括输入端口 in、输出端口 res,

在模块中声明 16 个宽为 4 位的寄存器:reg [0:3] sbox[0:15],在 initial 语句中初始化 S 盒(sbox)。将输入端口 in 每 4 位作 S 盒变换,通过连续赋值方式保存到线网型变量 t(16 位)矩阵中,接着作列混合变换,固定矩阵 M 与 t 关系如下:

$M=\{$
2,3,1,1,
1,2,3,1,
1,1,2,3,
3,1,1,2
$\}$

$t=\{t_0,t_1,t_2,t_3\}$

在有限域上,列混合变换是以 t 与转置矩阵 M^{T} 相乘实现,将列混合变换的结果每四位作 S 盒变换,结果赋给 res 输出端口。

RP 运算过程将输入结果相应位数作置换变换,但 *RP* 函数运算步骤包含在轮函数运算模块(PiccoloRound)中。

Piccolo-80 密码算法加密过程主要为 25 轮函数运算。在轮函数模块代码中,轮函数包括五个端口:res、state、key、q、计数器 count;在轮函数模块内部,将包含常数更新模块与 *F* 函数变换模块运算,利用 Verilog HDL 硬件描述语言的 assign 语句将常数更新模块与 *F* 函数变换模块并行,将常数更新模块得到的结果直接与 *F* 变换模块得到的结果进行运算;做到在不延迟加密时间的前提下,减少寄存器使用数量。将常数更新模块的输出子密钥信号记为 *rk*,*F* 函数变换模块的输出信号记为 $X[0]$ 与 $X[2]$,作如下运算步骤:

a) $X[1]$ = state[16:31] ^ $X[0]$ ^ rk[0:15];

b) $X[3]$ = state[48:63] ^ $X[2]$ ^ rk[16:31];

c)得到结果再作 *RP* 运算,将最终结果赋给输出端口信号 res。

主控制模块(Piccolo)主要运算包括计数器 count 控制 r(r 为 25)轮函数模块运算、RP^{-1} 轮置换函数运算、白化密钥生成运算和白化密钥异或运算。其中增加的 RP^{-1} 轮置换函数运算的运算结构如图 5 所示。

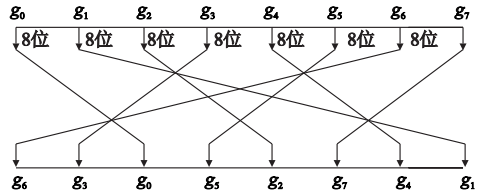

图5 轮置换函数逆运算(RP^{-1})

主控制模块(Piccolo)运算 Verilog HDL 关键代码描述如下:

```
always @(posedge clk) begin
        count <= (count^25)? count + 1:count;
          q <= (ready)? ((q^4)? q + 1:0):q;
        res <= ready? ((count)? t_res:{
        state[0:7]^key[0:7],state[8:15]^key[24:31],state[16:31],
        state[32:39]^key[16:23],state[40:47]^key[8:15],state[48:63]
        }):res;
        ready <= (count^25)? 1:0;
    end
    PiccoloRound PRound(t_res,res,key,count,q);
    assign result = {
        res[48:55]^key[64:71],res[24:31]^key[56:63],res[0:7],res[40:47],
        res[16:23]^key[48:55],res[56:63]^key[72:79],res[32:39],res[8:15]
        };
```

优化后的 Piccolo-80 算法运算流程如图 6 所示。

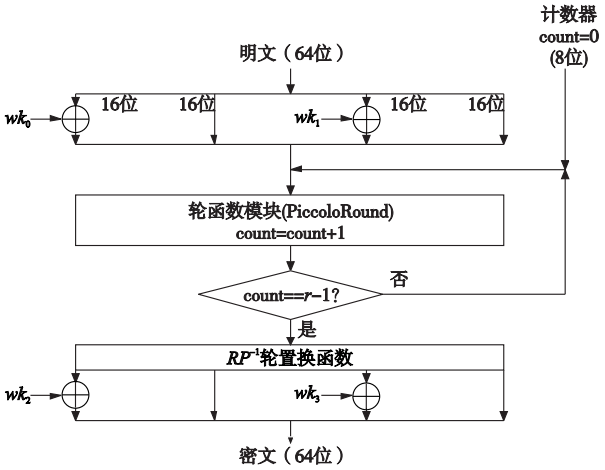

图6 Piccolo加密优化运算结构图

2.3 仿真实验

程序实现语言为 Verilog HDL 硬件描述语言,仿真软件为 ModelSim 6.1f,仿真结果截图如图 7 所示。

图7 优化后的Piccolo-80正确性验证结果

整个算法都是用连续赋值(assign)语句实现,用时钟信号控制计数器更新,完成整个加密只需要 25 个时钟周期,仿真运算步骤如下:

a)初始明文:plaintext = 64'h0123_4567_89AB_CDEF;初始密钥:key = 80'h0011223344_5566778899;时钟信号:clk = 1,跳到步骤 b)。

b)初始化计时器 count = 0,初始明文相应位与 wk_0、wk_1 异或,转步骤 c)。

c)调用 PiccoloRound 模块,计数器加 count = count + 1,转步骤 d)。

d)如果 count 等于 r = 25,转步骤 e);否则转步骤 c)。

e)PiccoloRound 模块输出结果作一次新构造 RP^{-1} 运算转步骤 f)。

f) RP^{-1} 运算输出结果相应位与 wk_2、wk_3 异或,转步骤 g)。

g)输出加密结果。

文献[1]在附录给出的测试向量截图如图 8 所示。原始文献[1]给出的测试向量是可以用来证明 Piccolo 改进的正确性。图 7 采用了原始文献提供的初始密钥和明文,在优化后的 Piccolo 算法中输入,得出的密文与原始文献[1]计算的密文相同,从而证明优化方法是正确的。

3 FPGA 硬件实现结果分析

将对优化前后的 Piccolo 密码算法通过 ISE 13.2 综合下载到 FPGA,进行硬件实现性能对比分析,其中 FPGA 型号为 Xil-

inx Virtex-5 LX50T。图 9 为未优化的 Piccolo 密码算法 FPGA 实现面积占用实验数据截图,图 10 是本文优化后的 Piccolo 密码算法下载到 FPGA 上的实验数据截图。

```
key         00112233 44556677 8899
plaintext   01234567 89abcdef
ciphertext  8d2bff99 35f84056
```

图8　原始文献提供的测试向量

```
Design Summary:
Number of errors:      0
Number of warnings:   66
Slice Logic Utilization:
  Number of Slice Registers:                5,312 out of  28,800   18%
    Number used as Flip Flops:              5,296
    Number used as Latch-thrus:                16
  Number of Slice LUTs:                     7,812 out of  28,800   27%
    Number used as logic:                   7,643 out of  28,800   26%
      Number using O6 output only:          7,395
      Number using O5 output only:             74
      Number using O5 and O6:                 174
    Number used as Memory:                    155 out of   7,680    2%
      Number used as Dual Port RAM:            64
        Number using O5 and O6:                64
      Number used as Shift Register:           91
        Number using O6 output only:           90
        Number using O5 output only:            1
    Number used as exclusive route-thru:       14
  Number of route-thrus:                       90
    Number using O6 output only:               85
    Number using O5 output only:                2
    Number using O5 and O6:                     3
```

图9　未优化Piccolo-80面积占用实验数据

```
Design Summary:
Number of errors:      0
Number of warnings:   66
Slice Logic Utilization:
  Number of Slice Registers:                5,268 out of  28,800   18%
    Number used as Flip Flops:              5,220
    Number used as Latches:                    32
    Number used as Latch-thrus:                16
  Number of Slice LUTs:                     4,629 out of  28,800   16%
    Number used as logic:                   4,461 out of  28,800   15%
      Number using O6 output only:          4,240
      Number using O5 output only:             87
      Number using O5 and O6:                 134
    Number used as Memory:                    155 out of   7,680    2%
      Number used as Dual Port RAM:            64
        Number using O5 and O6:                64
      Number used as Shift Register:           91
        Number using O6 output only:           90
        Number using O5 output only:            1
    Number used as exclusive route-thru:       13
  Number of route-thrus:                      103
    Number using O6 output only:               97
    Number using O5 output only:                4
    Number using O5 and O6:                     2
```

图10　优化后的Piccolo-80面积占用实验数据

根据图 9 和 10 实验数据得到,Piccolo-80 算法的实现面积占用从现有技术的 Size = 5312 + 7812 = 13124 Slices 优化到了 Size = 5268 + 4629 = 9897 Slices。其中 Slices 是 FPGA 中面积单元,可以得出优化后的 Piccolo-80 密码算法在面积上少了 3227 个 Slices,优化效率达到 24.6% 。

图 11 和 12 分别是下载到 FPGA 上硬件实现后的加密时钟频率,据此,可以计算出加密速率。

```
Design statistics:
   Minimum period:  13.424ns (Maximum frequency:  74.493MHz)
   Maximum path delay from/to any node:   4.628ns
   Maximum net delay:   0.805ns


Analysis completed Sat Feb 22 21:31:35 2014
--------------------------------------------------------------
```

图11　未优化Piccolo-80加密速率

从图 11 和 12 实验结果可以得到,Piccolo-80 算法所需加密速率从现有技术的 Minimum period = 13.424 ns 优化到了 Minimum period = 9.926 ns;可以得出优化后的 Piccolo-80 密码算法在加密速率上提高了 10% 。

```
Design statistics:
   Minimum period:   9.926ns (Maximum frequency: 100.746MHz)
   Maximum path delay from/to any node:   4.858ns
   Maximum net delay:   0.838ns


Analysis completed Sat Feb 22 14:15:58 2014
```

图12　优化的Piccolo-80加密速率

4　结束语

随着物联网的应用越来越深入,特别是目前的各种智能卡芯片越来越轻薄的情况下,如何更有效地减少加密算法在智能卡上的实现面积是一个重要研究课题。因此,如何对 Piccolo 密码算法进行硬件面积优化、同时又不降低加密性能是必要的。本文对 Piccolo 密码算法的结构进行优化,在面积减少的同时也保证了加密效率。本文的优化方法不但对 Piccolo 密码算法适应,也对类似轻量级密码算法设计与实现具有重要参考价值。进一步的工作是研究 Piccolo 密码算法在抗旁路攻击上的优化实现。

参考文献:

[1] Shibutani K, Isobe T, Hiwatari H. Piccolo:an ultra-lightweight block-cipher[C]//Proc of International Workshop of Cryptographic Hardware and Embedded Systems. 2011:326-341.

[2] Minier M. On the security of piccolo lightweight block cipher against related-Key impossible differentials[C]//Proc of the 14th International Conference on Cryptology. 2013:308-318.

[3] Jeong K. Security analysis of block cipher piccolo suitable for wireless sensor networks[J]. Peer-to-Peer Networking and Application, 2013,7(4):636-644.

[4] Jeong K. Cryptanalysis of cipher piccolo suitable for cloud computing [J]. Journal of Supercomputing,2013,66(2):829-840.

[5] Wang Yanfeng, Wu Wenling, Yu Xiaoli. Biclique cryptanalysis of reduced-round piccolo block cipher[C]//Proc of the 8th International Conference on Information Security Practice and Experience. 2012: 337-352.

[6] 赵光耀,李瑞林,孙兵,等. Piccolo 算法的差分故障攻击[J]. 计算机学报,2012,35(9):882-894.

[7] 赵新杰,郭世泽,王韬,等. Piccolo 密码代数故障分析研究[J]. 计算机学报,2013,36(4):1918-1926.

[8] 王晨旭,赵占锋,喻明艳,等. Piccolo 相关性功耗分析攻击技术研究[J]. 哈尔滨工业大学学报,2013,45(9):17-21.

[9] 李浪,李仁发,邹祎,等. PRESENT 密码硬件语言实现及其优化研究[J]. 小型微型计算机系统,2013,34(10):2272-2275.

(上接第 3055 页)

[8] 李丽. 基于双抽样的测量流长度分布的算法研究[D]. 大连:大连海事大学,2009.

[9] Kawahara R, Ishibashi K, Mori T, *et al*. Detection accuracy of network anomalies using sampled flow statistics[C]//Proc of Global Telecommunications Conference. [S. l.]:IEEE Press,2007:1959-1964.

[10] 刘渊,刘元珍,李小航. 一种新的基于 SCBF 的流抽样测量算法研究[J]. 计算机工程与应用,2007,43(29):140-142.

[11] Silveira F, Diot C, Taft N, *et al*. ASTUTE:detecting a different class of traffic anomalies[J]. ACM SIGCOMM Computer Communication Review,2010,40(4):267-278.

[12] Hohn N, Veitch D, Abry P. Cluster processes: a natural language for network traffic[J]. IEEE Trans on Signal Processing, 2003, 51 (8):2229-2244.

[13] Karagiannis T, Molle M, Faloutsos M, *et al*. A nonstationary Poisson view of Internet traffic[C]//Proc of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies. [S. l.]:IEEE Press,2004:1558-1569.

[14] 程光,唐永宁. 基于近似方法的抽样报文流数估计算法[J]. 软件学报,2013,24(2):255-265.

[15] 夏锋. OMNeT ++ 网络仿真[M]. 北京:清华大学出版社,2013.