高性能 BLAKE 算法研究及其 FPGA 实现*

许文龙[a]，王 奕[a,b]，陈 佐[a,b]，李仁发[a,b]，宋 倩[a]
（湖南大学 a. 嵌入式系统及网络实验室；b. 网络与信息安全湖南省重点实验室，长沙 410082）

摘 要：BLAKE 算法是基于 ChaCha 流密码和采用标准 HAIFA 迭代模式的 SHA-3 候选算法之一。针对现有 BLAKE 算法对于循环单元中 G 函数模型的研究少，且没有考虑硬件代价及实现结果等问题，提出可重构的设计思想，在 FPGA 上实现了 BLAKE 算法循环单元的三种 G 函数模式。在 Xilinx Virtex-5 FPGA 上的实现结果表明，在不影响性能的前提下，本方案可重构后的面积比分别实现的面积总和减少了 58%。

关键词：安全哈希算法；可重构；现场可编程门阵列；G 函数；BLAKE
中图分类号：TP311 **文献标志码**：A **文章编号**：1001-3695(2012)06-2098-04
doi:10.3969/j.issn.1001-3695.2012.06.024

Research of high-performance BLAKE algorithm on FPGA platform

XU Wen-long[a], WANG Yi[a,b], CHEN Zuo[a,b], LI Ren-fa[a,b], SONG Qian[a]
(*a. Embedded System & Network Laboratory, b. Hunan Province Key Laboratory of Network & Information Security, Hunan University, Changsha* 410082, *China*)

Abstract: BLAKE algorithm is one of SHA-3 finalist which is based on ChaCha and take use of standard HAIFA iterative mode. Previous research work had been done on hardware implementation of BLAKE algorithm, but the disadvantage of their implementations lay on few research on the G_function module of the round unit of BLAKE without considering hardware costs and results. In order to resolve the above shortage, this paper proposed a new reconfigurable architecture which could support three different parameters of G-function module of round unit of BLAKE algorithms on Xilinx Virtex-5 FPGA. The experimental results show that the proposed design has smaller size that is 58% smaller than others without affecting performance.
Key words: secure hash algorithm(SHA); reconfigurability; field-programmable gate array(FPGA); G-function; BLAKE

密码学中的哈希函数是将任意长度的消息块压缩为固定长度摘要的函数。SHA 是一种数据加密算法。1993 年 NIST (National Institute of Standards and Technology)公布了 SHA-0 算法，目前已有 SHA-1 和 SHA-2。2004 年，一直在国际上广泛使用的两大密码算法 MD5、SHA-1 被 Wang 等人[1]成功破解，于是 NIST 于 2007 年发起了更加安全的 hash 函数新标准的征集活动——SHA-3 竞赛[2]，防止 SHA-2 算法被破解带来的危害。BLAKE 凭借其运算速度快等优越性在第二轮中脱颖而出，成为五个候选算法之一。BLAKE 算法是由 Aumasson 等人[3]共同设计的，根据消息摘要位宽的不同，可以分成 BLAKE-28、BLAKE-32、BLAKE-48、BLAKE-64 四种版本。

相比传统的哈希函数，BLAKE 算法运算速度快，且安全性高。目前已经有研究者在硬件上实现了 BLAKE 算法，并给出了 BLAKE 算法的硬件性能。Baldwin 等人[4]提出了一个硬件接口，分别实现了 BLAKE 算法的四个参数版本，并且带有消息填充单元。Aumasson 等人对 BLAKE 进行详细的研究，分别实现了 BLAKE 的四种版本，并实现了 BLAKE-32 的三种 G 函数组合模式[3](1G、4G、8G)，其吞吐量分别为 253 Mbps、4 153 Mbps、5 295 Mbps。Matsuo 等人[5]实现了带有统一接口的 BLAKE-32 算法，并在 SASEBO-GII FPGA 上测试了其频率、吞吐量、面积以及功耗等，吞吐量达到 2 676 Mbps，面积为 1 660 slices，其吞吐量是目前最好的，但是占用的面积较大。

综上所述，目前 BLAKE 的研究存在以下两点问题：a) BLAKE 的实现版本单一，只实现了 BLAKE 的一个版本(如 BLAKE-32)或者分别实现四个版本，没有综合考虑硬件代价、吞吐量、速度等性能指标；b)对于 BLAKE-32 的几种 G 函数模型研究得较少，仅有 Aumasson 等人作了相关研究，只分别实现了这几种 G 函数模型，没有综合考虑硬件代价和速度。

针对上述问题，本文分别实现 BLAKE-32 算法的三种 G 组合模式(1G、4G、8G)，提出三种 G 模式的可重构结构图，并将 BLAKE-32 的三种不同 G 函数模式以可重构结构实现在统一的硬件资源上，为使用者提供了可进行灵活选择的可重构结构。可重构结构实现的占用面积比三种模式分别实现的面积总和减少了 58%。可以看出，本文提出的可重构结构比文献中的设计硬件资源占用更少、吞吐量更大。

1 BLAKE 介绍

BLAKE 是 SHA-3 的一种候选算法，满足 NIST 制定的所有标准，提供了理论和实践上的安全保证，它完全依赖于预分析部件。BLAKE 算法是基于 Bernstein 的 ChaCha 流密码(一种较快的流密码)，采用的是 LAKE(一种带宽管道结构的哈希函

收稿日期：2011-11-26；**修回日期**：2011-12-30 **基金项目**：国家自然科学基金资助项目(60873074,60673061)；长沙市科技计划资助项目(K1003028-11)；中央高校基本科研业务费资助项目

作者简介：许文龙(1988-)，男，山东滕州人，硕士研究生，主要研究方向为嵌入式系统(276714926@qq.com)；王奕(1977-)，女，讲师，博士，主要研究方向为计算机系统结构、嵌入式安全；陈佐(1979-)，男，讲师，博士，主要研究方向为复杂网络、数据挖掘、流量建模；李仁发(1956-)，男，教授，博士，主要研究方向为嵌入式系统；宋倩(1986-)，女，硕士研究生，主要研究方向为嵌入式系统.

数)的算法思想,使用标准的 HAIFA 迭代模式,并且在 ChaCha 核心函数上建立它自己的压缩函数。HAIFA 迭代模式在压缩函数中混入 salt 和 counter,引入随机哈希来克服迭代结构的弱点。BLAKE 主要抵抗一般的次原像攻击、长延期、旁路攻击[3]。

BLAKE 根据消息摘要的位宽不同可分为 BLAKE-28、BLAKE-32、BLAKE-48、BLAKE-64 四种。BLAKE-28 和 BLAKE-32 运行 32 bit 的字,输出 224 bit、256 bit 的固定长度串;BLAKE-48 和 BLAKE-64 运行 64 bit 的字,输出 384 bit 和 512 bit 的固定长度串。这几种算法有很多相似的地方,但具体操作各有不同,如输入位数等不同。本文主要针对 BLAKE-32 进行研究,进而可以将设计扩展到其他几种哈希函数。

BLAKE 压缩函数的结构是 LAKE 的一个继承。首先初始化 4×4 矩阵状态值,输入包括初始串值、salt、counter;其次,矩阵经过十次信息块循环迭代赋值;最后,压缩返回一个新的串值。图 1 是一次循环的结构图。

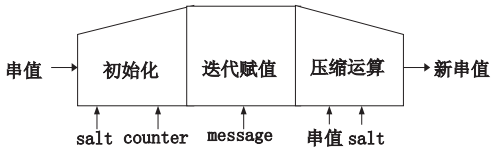

图1　BLAKE压缩函数结构图

BLAKE 压缩函数的步骤:

1)压缩函数的初始化

BLAKE-32 运行 32 bit 的字,并且返回一个 32 bit 的哈希值,它和 SHA-256 有相同的初始值[3]:

$$IV_0 = 6A09E667 \quad IV_1 = BB67AE85$$
$$IV_2 = 3C6EF372 \quad IV_3 = A54FF53A$$
$$IV_4 = 510E527F \quad IV_5 = 9B05688C$$
$$IV_6 = 1F83D9AB \quad IV_7 = 5BE0CD19$$

BLAKE-32 有 16 个参数值[3]:

$$C_0 = 243F6A88 \quad C_1 = 85A308D3$$
$$C_2 = 13198A2E \quad C_3 = 03707344$$
$$C_4 = A4093822 \quad C_5 = 299F31D0$$
$$C_6 = 082EFA98 \quad C_7 = EC4E6C89$$
$$C_8 = 452821E6 \quad C_9 = 38D01377$$
$$C_{10} = BE5466CF \quad C_{11} = 34E90C6C$$
$$C_{12} = C0AC29B7 \quad C_{13} = C97C50DD$$
$$C_{14} = 3F84D5B5 \quad C_{15} = B5470917$$

BLAKE-32 的压缩函数有四个输入值(本文中 salt 的值为 0):

串值 $h = h_0, \cdots, h_7$;

信息块 $m = m_0, \cdots, m_{15}$;

salt $s = s_0, \cdots, s_3$;

计数器值 $t = t_0, \cdots, t_1$。

压缩函数:输入 h, m, t, s 四个参数后,会产生一个新的串值 h',$h' = \text{compress}(h, m, s, t)$。

初始化主要是对一个 16 个字的状态值($V_0 \sim V_{15}$)进行初始化赋值,这个状态值是以一个 4×4 的矩阵形式给出的,如下所示:

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix} \leftarrow \begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 & h_7 \\ s_0 \oplus c_0 & s_1 \oplus c_1 & s_2 \oplus c_2 & s_3 \oplus c_3 \\ t_0 \oplus c_4 & t_0 \oplus c_5 & t_1 \oplus c_6 & t_1 \oplus c_7 \end{pmatrix}$$

2)迭代赋值

一旦状态值被初始化后,压缩函数就会迭代地执行 10 次循环,每一次循环都会改变状态值 V 的值,循环计算如下:

$$G_0(v_0, v_4, v_8, v_{12}) \quad G_1(v_1, v_5, v_9, v_{13}) \quad G_2(v_2, v_6, v_{10}, v_{14})$$
$$G_3(v_3, v_7, v_{11}, v_{15}) \quad G_4(v_0, v_5, v_{10}, v_{15}) \quad G_5(v_1, v_6, v_{11}, v_{12})$$
$$G_6(v_2, v_7, v_8, v_{13}) \quad G_7(v_3, v_4, v_9, v_{14})$$

在每一次循环 r 中(r 表示第几轮循环),函数 $G_i(a, b, c, d)$ 被定义为

$$a \leftarrow a + b + (m_{\sigma r(2i)} \oplus c_{\sigma r(2i+1)})$$
$$d \leftarrow (d \oplus a) >>> 16$$
$$c \leftarrow c + d$$
$$b \leftarrow (b \oplus c) >>> 12$$
$$a \leftarrow a + b + (m_{\sigma r(2i+1)} \oplus c_{\sigma r(2i)})$$
$$d \leftarrow (d \oplus a) >>> 8$$
$$c \leftarrow c + d$$
$$b \leftarrow (b \oplus c) >>> 7$$

BLAKE-32 共执行 10 轮的置换运算,消息字 m 及常量 c 在不同的轮运算中运用不同的置换 $\sigma_{r(2i)}$ 和 $\sigma_{r(2i+1)}$。置换规则如表 1 所示。

表 1　置换规则表

σ_0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
σ_1	14	10	4	8	9	15	13	6	1	12	0	2	11	7	5	3
σ_2	11	8	12	0	5	2	15	13	10	14	3	6	7	1	9	4
σ_3	7	9	3	1	13	12	11	14	2	6	5	10	4	0	15	8
σ_4	9	0	5	7	2	4	10	15	14	1	11	12	6	8	3	13
σ_5	2	12	6	10	0	11	8	3	4	13	7	5	15	14	1	9
σ_6	12	5	1	15	14	13	4	10	0	7	6	3	9	2	8	11
σ_7	13	11	7	14	12	1	3	9	5	0	15	4	8	6	2	10
σ_8	6	15	14	9	11	3	0	8	12	2	13	7	1	4	10	5
σ_9	10	2	8	4	7	6	1	5	15	11	9	14	3	12	13	0

在刚开始的四个 G 函数 $G_0 \cdots G_3$ 可以并行运算,因为它们单独执行矩阵中的一列,相互不干扰,本文把 $G_0 \cdots G_3$ 的计算步骤叫做列步骤。同样,最后四个函数 $G_4 \cdots G_7$ 执行的是矩阵中不同对角线上的参数,所以称之为斜步骤。图 2 给出了 G 函数的运算过程,图 3 给出了列步骤和斜步骤的计算规则。

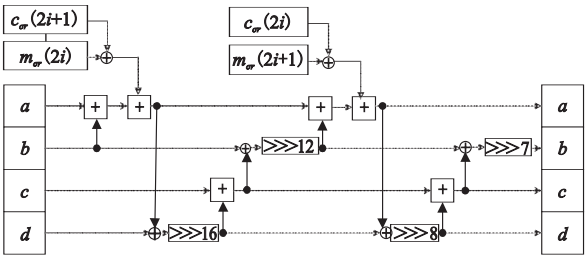

图2　G函数的运算过程

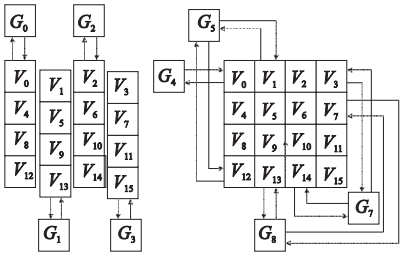

图3　列步骤和斜步骤计算规则

3)压缩运算

循环函数完成后,最终的 $v_0 \cdots v_{15}$ 作为输入,与初始化的 $h_0 \cdots h_7$ 和 $s_0 \cdots s_3$ 进行相应位的异或运算,产生一串新的值——最终消息摘要值 $h_0' \cdots h_7'$:

$$h_0' \leftarrow h_0 \oplus s_0 \oplus v_0 \oplus v_8$$
$$h_1' \leftarrow h_1 \oplus s_1 \oplus v_1 \oplus v_9$$
$$h_2' \leftarrow h_2 \oplus s_2 \oplus v_2 \oplus v_{10}$$
$$h_3' \leftarrow h_3 \oplus s_3 \oplus v_3 \oplus v_{11}$$
$$h_4' \leftarrow h_4 \oplus s_0 \oplus v_4 \oplus v_{12}$$
$$h_5' \leftarrow h_5 \oplus s_1 \oplus v_5 \oplus v_{13}$$
$$h_6' \leftarrow h_6 \oplus s_2 \oplus v_6 \oplus v_{14}$$

2 硬件的设计与实现

2.1 可重构 BLAKE 算法设计

BLAKE-32 三种 G 模块的实现是基于 G 函数组合模块的，分别调用 1、4 和 8 个 G 函数，每个 G 组合模块调用的函数结构相同，都是四个输入经过一系列的运算输出四个新值，不同的只是 G 函数的组合个数。考虑到 BLAKE 的三种 G 函数版本除了调用个数不同外，其余结构基本一致，因此，为了提高硬件的利用率，本文提出了 BLAKE 三种 G 函数组合模块的可重构方案。图 4 为 BLAKE 三种 G 函数模块可重构方案的硬件体系图。

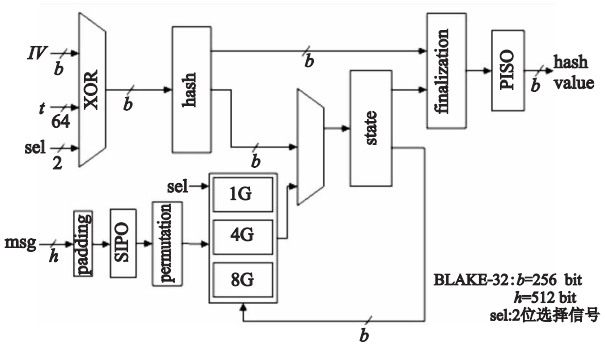

图4 BLAKE可重构方案的硬件体系结构

在图 4 中，msg 代表的是信息块，sel 选择信号用来选择 1G、4G 和 8G 组合模块。表 2 给出了循环函数的不同 G 组合模块对应的 sel 选择信号的值。

可重构 BLAKE 算法的状态设计实现如图 5 所示，包括 IDLE、LOAD、LOAD_RED、PERMUTE 以及 OUT 五个状态。IDLE 是空闲状态，表示没有消息需要处理。当有消息需要处理的时候，进入 LOAD 状态，取数完成后进入 LOAD_RED 准备状态，然后直接进入 PERMUTE 状态进行十次迭代，round_r 为 10 表示循环结束，进入 OUT 输出状态，当 count 为 15 时，输出完成进入 IDLE 空闲状态。

表2 G组合模块对应的sel选择值表

G model	sel
8G	sel=00
4G	sel=01
1G	sel=10

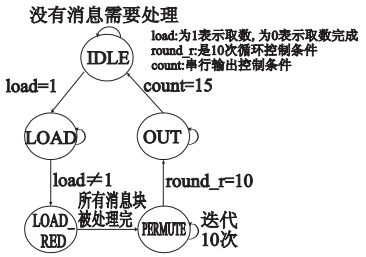

图5 可重构BLAKE算法的状态设计图

2.2 关键部件的设计

迭代赋值是 BLAKE 算法的核心部件，也是 BLAKE 算法硬件实现的关键部件，在很大程度上决定了 BLAKE 算法硬件实现的性能，现就以 BLAKE-32 的 4G 模块为例给出该部件的说明。图 6 给出了迭代赋值函数的示意图，图中只是给出了半个循环的演示，4 个 G_function 就是代表的列步骤（$G_0 \cdots G_3$），C 和 M 分别代表常量和消息字，经过 G 运算以后得出新的 $v_0 \cdots v_{15}$，经过 LVL2 和 LVL1 重新赋给矩阵，以备下半个循环（斜步骤）使用。

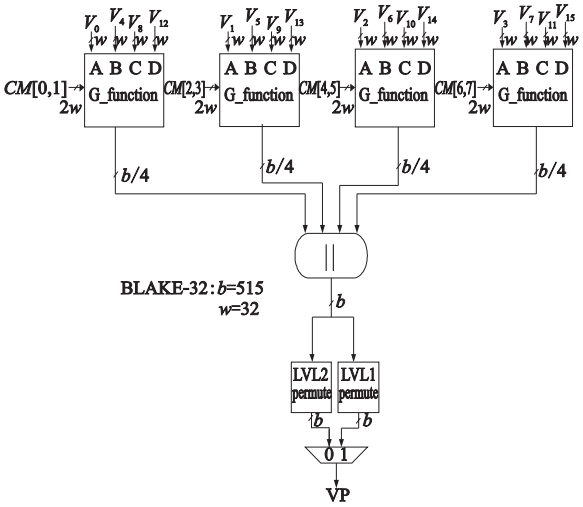

图6 迭代赋值函数结构

LVL2 置换表将上半个循环的矩阵值（4 个列步骤的输出）赋给一个新的矩阵，以适用于下半个循环（斜步骤）。LVL1 置换表是 LVL2 的逆置换表，如表 3 所示。

表 3 LVL2、LVL1 置换表

LVL1	0	1	2	3	5	6	7	4	10	11	8	9	15	12	13	14
LVL2	0	1	2	3	7	4	5	6	10	11	8	9	13	14	15	12

本文提到了 1G_BLAKE-32、4G_BLAKE-32、8G_BLAKE-32 的结构，其中 1G、4G 和 8G 就是 G 函数模块调用的个数。在其他文献中 4G_BLAKE-32 比较常见，就是调用 4 个 G 函数，在一次循环中先并发执行列步骤 $G_0 \cdots G_3$，然后再并发执行斜步骤 $G_4 \cdots G_7$，对这 4 个 G 重复使用了两次。函数 1G_BLAKE-32 就是只调用 1 个 G 函数，先利用它依次执行列步骤 $G_0 \cdots G_3$，然后再依次执行斜步骤 $G_4 \cdots G_7$，这 1 个 G 函数在一个循环内重复使用了八次，占用了八个周期。同理，8G_BLAKE-32 就是调用 8 个 G 函数，其中 4 个 G 函数先执行列步骤，然后另外 4 个执行斜步骤，8 个 G 函数只用了一次。

3 结果比较

本文分别实现了 BLAKE-32 三种 G 组合模式，并设计实现了这三种组合模式的可重构结构，采用 Verilog 硬件描述语言，用 Xilinx ISE 11.1 和 Quartus 9.0 将设计综合实现在 Xinlinx Virtex-5（xc5vlx220）FPGA 上，对 BLAKE 硬件实现的频率、吞吐量和面积等几个关键参数进行了相关比较。表 4 ~ 6 给出了本设计与其他文献的结果比较。通过实验结果，本文可重构的设计比三种分开设计的面积总和小了 58%。

表 4 给出了 BLAKE 的 1G 组合模式在 Xilinx Virtex-5（xc5vlx220）上的硬件实现结果。从表中可以看出，目前性能结果最好的是 Aumasson 等人[3]的设计。所以，本文的设计与其比较，可重构的设计相对于文献[3]，速度提高了 38%，吞吐量提高了 37%。

表 4 1G_BLAKE-32 模型在 Xilinx Virtex-5（xc5vlx220）上的实现结果比较

备注	面积/slices	速度/MHz	时钟/cycles	吞吐量/Mbps
文献[3]	390	91	81	575
文献[8]	56	372	844	225
本文可重构	2 149	126	82	785

表 5 给出了 BLAKE 的 4G 组合模式在 Xilinx Virtex-5 (xc5vlx220) 上的硬件实现结果比较。从表中可以看出,目前性能最好的是 Matsuo 等人的实验结果,本文可重构的设计速度相对文献[5]的结果提高了 10%,吞吐量提高了 10%。

表 5　4G-BLAKE-32 模型在 Xilinx Virtex-5(xc5vlx220)上的实现结果比较

备注	面积/slices	速度/MHz	时钟/cycles	吞吐量/Mbps
文献[6]	1 660	115	22	2 676
文献[5]	1 660	115	22	2 676
文献[7]	1 851	117	23	2 611
文献[3]	1 217	100	21	2 438
文献[4]	1 118	118	56	1 079
本文可重构	2 149	126	22	2 927

表 6 给出的是 BLAKE 的 8G 组合模式在 Xilinx Virtex-5 (xc5vlx220)上的硬件实现结果比较。本文可重构设计吞吐量为 2 927 Mbps,虽然比文献[3]的结果小 5.7%,但是速度比它的设计快 1.88 倍。

表 6　8G_BLAKE-32 模型在 Xilinx Virtex-5(xc5vlx220)上的实现结果比较

备注	面积/slices	速度/MHz	时钟/cycles	吞吐量/Mbps
文献[3]	1 694	67	11	3 103
本文可重构	2 149	126	22	2 927

4　结束语

本文通过分析指出现有 BLAKE 算法在硬件实现上的不足,设计了改进的 BLAKE 算法即三种 G 函数组合模式硬件实现方案。从本文的数据对比结果中可以看出,本方案同时支持 BLAKE 算法的三种 G 函数组合模式,与已有的设计相比,提供了一种面积小、时钟频率高和灵活性强的 BLAKE 硬件实现方案。

参考文献:

[1] WANG Xiao-yun, YU Hong-bo. How to break MD5 and other hash functions[C]//Lecture Notes in Computer Science, vol 3494. Berlin: Springer, 2005: 19-35.

[2] National Institute of Standards and Technology. Cryptographic hash algorithm competition[EB/OL]. (2008-11-10)[2011-02-10]. http://csrc.nist.gov/groups/ST/hash/timeline.html.

[3] AUMASSON J P, HENZEN L, MEIER W, *et al*. SHA-3 proposal BLAKE[EB/OL]. (2008-10-31)[2011-02-12]. http://www.cs.rit.edu/~ark/20090927/Round2Candidates/BLAKE.pdf.

[4] BALDWIN B, HANLEY N, HAMILTON M, *et al*. FPGA implementations of the round two SHA-3 candidates[EB/OL]. (2010-04-26)[2011-03-12]. http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/Aug2010/documents/papers.

[5] MATSUO S, KNEZEVIE M, SCHAUMONT P, *et al*. How can we conduct "fair and consistent" hard-ware evaluation for SHA-3 candidate[EB/OL]. (2010-08-15)[2011-04-13]. http://csrc.nist.Gov/groups/ST/Hash/sha-3/Round2/Aug2010/documents/papers.

[6] KOBAYASHI K, IKEGAMI J, MATSUO S, *et al*. Evaluation of hardware performance for the SHA-3 candidates using SASEBO-GII [EB/OL]. (2010-12-10)[2011-3-20]. http://eprint.iacr.org/.

[7] HOMSIRIKAMOL E, ROGAWSKI M, GAJ K. Comparing hardware performance of fourteen round two SHA-3 candidates using FPGAs [EB/OL]. (2009-04-26)[2011-04-20]. http://eprint.iacr.org/.

[8] BEUCHAT J L, OKAMOTO E, YAMAZAKI T. Compact implementations of BLAKE-32 and BLAKE-64 on FPGA[EB/OL]. (2010-04-26)[2011-04-13]. http://eprint.iacr.org/2010/173.pdf.