SHA-3候选算法Keccak的Matlab设计与实现

刘 花 包小敏

(西南大学数学与统计学院 重庆 400715)

摘 要 由NIST发起的新一代Hash函数标准SHA-3的全球公开征集过程目前已进入最后一轮筛选,Keccak是进入最后一轮的5个候选者之一。介绍了Keccak算法及特点,给出了一种基于Matlab、带有图形界面GUI的Keccak程序设计与实现过程。本程序既可用于实际的Keccak Hash值运算,最重要的是为Keccak的教学与研究提供了一个方便直观的工具。

关键词 Hash,SHA-3,Keccak,Sponge,Matlab

中图法分类号 TP309 **文献标识码** A

Design and Implementation of the SHA-3 Candidate Algorithm Keccak Based on Matlab

LIU Hua BAO Xiao-min

(School of Mathematics and Statistics, Southwest University, Chongqing 400715, China)

Abstract Currently the final round of SHA-3(Secure Hash Algorithm-3) competition launched by NIST is under way. The hash algorithm Keccak is one of the five candidates for the competition at the last round. This paper described the design and implementation of Keccak based on Matlab. Our program with a GUI(Graphic User Interface) can be used both in practical Keccak Hash value calculation and Keccak teaching or experiments.

Keywords Hash,SHA-3,Keccak,Sponge,Matlab

1 引言

Hash函数又称哈希函数、杂凑函数或散列函数,在现代密码学中扮演着重要的角色。Hash函数接受一个任意有限长度的消息作为输入,产生一个固定长度的消息摘要作为输出,其基本思想是把消息摘要作为输入信息的一个紧凑表示像,用于惟一地识别消息。Hash函数被广泛地应用在数据完整性检测、数字签名、信息认证以及文件校验等方面,是密码学的一个基本工具。

早期较成熟的Hash函数是MD5。1993年,美国国家标准与技术研究所(NIST)公布了密码学上第一代安全Hash标准(Secure Hsah Standard,SHS)SHA-0,随后,基于安全性考虑,NIST先后于1995年、2002年更新了Hash函数为SHA-1,SHA-2。目前,Hash函数的设计与分析已成为信息安全领域的热点问题,近年来取得的突破性进展,使得现行的标准Hash函数中的MDx系列和SHA系列的安全性都不再满足实际的要求。因此NIST于2007年发起了Hash函数新标准的征集活动,即SHA-3竞赛。

截止于2008年10月31日NIST收到了来自世界各个密码组织或个人提交的64份候选提案,Keccak算法即为其中之一。经历了2008年的初选、2009年的第一轮筛选及2010年的第二轮筛选后,Keccak成功进入SHA-3的最后一轮(final round),成为5个最终候选Hash算法之一。目前,针对这5个算法,各国相关的学者都在积极地研究,以便在2012年能从这5个算法中选出一个作为SHA-3标准。因此,对Keccak的研究除了其本身的理论意义之外,还有重要的现实意义。

Keccak是由STMicroelectronics的G. Bertoni、J. Daemen、G. Assche和NXP Semiconductors的M. Peeters等人共同研究设计的。与进入SHA-3最终决赛的其余4个算法相比较,Keccak使用了不同于传统Merkle-Damg{\aa}rd结构的sponge结构,从而减小了被MD结构常用攻击攻破的风险。鉴于sponge结构的新颖与独特性,对Keccak算法的学习与研究更具有重要的理论意义。

2 Keccak算法描述

自算法提交参赛起,Keccak算法设计者就不断对算法进行更新改进。时至最终决赛,Keccak算法在padding规则、迭代轮数、算法参数等方面均有不同程度的改进,本文以下的讨论均基于Keccak算法进入第三轮决赛时的最新版。

2.1 Keccak算法总体描述

Keccak是基于海绵构造(sponge construction),增添特殊的padding算法后得到的Hash函数族,定义为 $Keccak[r,c]$,其参数 r 与 c 分别称为码率(bitrate)和容量(capacity),$r+c$ 决定了Keccak中所使用的基本函数 $Keccak\text{-}f[b]$ 置换宽度 $b\in\{25,50,100,200,400,800,1600\}$。图1展示了其总体

刘 花(1987—),女,硕士生,主要研究方向为密码学,E-mail:yngjyzlh@swu.edu.cn;包小敏(1959—),男,博士,教授,主要研究方向为密码学、组合数学。

构造，图中的函数 f 即为 *Keccak-f*[b]。

如图 1 所示，任意长度的信息 M 输入后首先进行 Pad 处理，使其处理后长度为 r 的整数倍(假设为 k 倍)，并按每组 r 比特将其分为 k 组。之后进入 sponge 结构的 absorbing 部分：在 absorbing 部分算法共进行 k 次 *Keccak-f*[b]置换，置换宽度 b 即为每次置换输入变量长度(b 比特)。其中第 i 次置换输入变量的前 r 比特由信息 M 经 Pad 处理并分组后的第 i 组与第 $i-1$ 次置换后输出变量的前 r 比特进行异或运算得到，后 $b-r=c$ 比特则由全 0 常量与第 $i-1$ 次置换后输出变量的后 c 比特进行异或运算得到。如此进行 k 次迭代置换后进入 spong 结构 squeezing 部分：算法 Keccak 的最终输出变量由 squeezing 部分每一次 *Keccak-f*[b]置换输出变量的前 r 比特依次并联得到，因此所需进行的置换次数由 Keccak 算法输出长度的具体要求决定。

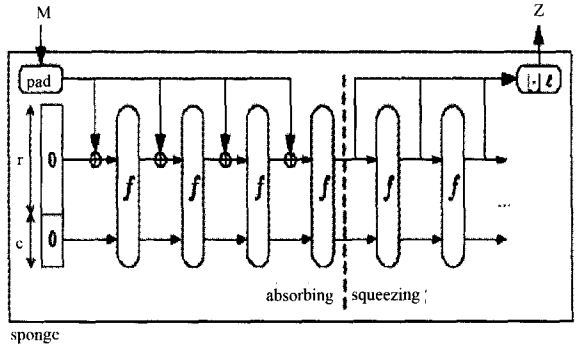

图 1 The sponge construction Keccak-f[b]

分析不难看出，sponge 结构中“吸收(absorb)”含义源于 absorbing 部分每次置换输入对信息 M 的“吸纳作用”，而“挤压(squeez)”含义则源于 squeezing 部分每次置换输出“挤压”对 Keccak 算法最终输出信息的“挤压作用”。

在文献[1]中，NIST 要求参赛候选算法必须支持至少 4 种不同的输出长度 $n\in\{224,256,384,512\}$，因此，Keccak 设计者定义了如下 4 种指定输出长度的函数。

$n=224$:⌊ *Keccak*[$r=1152,c=448$]⌋$_{224}$

$n=256$:⌊ *Keccak*[$r=1088,c=512$]⌋$_{256}$

$n=384$:⌊ *Keccak*[$r=832,c=768$]⌋$_{384}$

$n=512$:⌊ *Keccak*[$r=576,c=1024$]⌋$_{512}$

当然，对应于其他有效的 r,c 取值，Keccak 有另外的应用，此处我们不做进一步的说明，更多信息可查阅文献[2]。

2.2 Keccak 算法细则描述

1) Padding 规则的进一步改进

进入第三轮竞赛后，设计者将 Keccak 算法的 Padding 规则确定为 Muti-rate padding—pad10 * 1，即将信息 M 首先并联一个‘1’，继而并联若干个‘0’，最后再并联一个‘1’，其中‘0’的个数是使得 M 并联后长度为 r 的倍数的最小正整数。文献[3]中证明了此规则既满足与前两轮规则相同的功效，又不影响 Keccak 算法的安全性，同时还将 Padding 的最小位数从原来的 25 位降低至 2 位，较前两轮 Padding 规则简单有效。

2) 基本函数 *Keccak-f*[b]描述

Keccak-f[b]为迭代置换函数，是 Keccak 算法的基本核心函数。根据置换宽度 b 的不同取值，共有 7 种不同的迭代置换，其中 $b\in\{25,50,100,200,400,800,1600\}$。

迭代轮数 n_r 由 b 唯一决定：$n_r=12+2l$，其中 $2^l=b/25$。提交给 SHA-3 筛选的 *Keccak-f*[b]函数，b 的值为 1600，故迭代轮数为 24 轮，相应地 24 个轮常数[4]参与轮迭代。

从 Keccak 算法的 sponge 整体结构来看，状态更新可视为作用在长度为 b 的一维数组 s 上的迭代排列，但 *Keccak-f*[b]置换内部状态更新使用的却是作用在 3 维数组 A 上的迭代排列，两者对应关系如下：

$s[\omega(5y+x)]=A[x][y][z],\omega=b/25$

作用在 3 维数组上，*Keccak-f*[b]的每一轮置换均由 $\theta,\rho,\pi,\chi,\iota$ 这 5 个步骤组成，文献[5]详细给出了 5 个步骤的算法分析，并给出了每个置换步骤对算法安全性的影响说明。

3) formal bit-reordering 规则说明

在 Keccak 内部，输入信息 M 遵循有效位由低到高排列的规则，即第 0 个比特为最低有效位(LSB)，最后一个比特为最高有效位(MSB)。而这与 NIST 的约定恰好相反，为使 Keccak 内部约定与 API 外部约定兼容，设计者给出信息进入算法前的处理规则，即为 formal bit-reordering[2]。

在实际操作中，对于含有 8 比特的字节，采用 formal bit-reordering 进行两次操作后信息将复原，因此事实上我们仅需对最后一个所含比特数 $\rho<8$ 比特的字节进行操作即可。值得注意的是，该规则在 Keccak 算法的程序实现上不容忽视。

3 Keccak 算法的 Matlab 设计与实现

从高等教育教学的角度出发，无论是 Hash 函数的重要性，还是 Keccak 算法所使用 sponge 结构的独特性，都决定了设计一个带有图形用户界面(Graphic User Interface，GUI)的 Keccak 进行研究和学习是很有必要的。与现仅有的 Keccak 算法 C 语言程序实现相比，Matlab 的图形用户界面更易于学生操作，且数据观察更为直观。值得一提的是，本文基于 Matlab 所设计的 Keccak 在实现 Hash 值输出的同时，增添了中间数据的输出，对算法的学习和研究起到重要的辅助作用。

3.1 主要内部函数

程序的设计大致可分为用户选项选择与检测、Keccak 算法运行以及输出与存盘三大板块，其中涉及到如下几个重要的内部函数：

1) Keccak 算法主函数

Keccak 算法状态更新使用的是 3 维数组上的迭代排列，且所涉及的中间数据指标较多，因此我们主要使用 cell 型数据结构进行运算。

程序中，函数[*HashValue*, *PerInter*, *state*, *absorb*, *max*] = *KeccakMain*(M,r,c)为核心函数，实现 *Keccak* 算法在参数 r,c 时对十六进制字符表示信息 M 的作用，各输出变量相关说明如下：

• *absorb* 记录在算法 Absorbing 部分共经历的 *Keccak-f*[b]置换次数。

• *max* 记录程序实现中所经历的 *Keccak-f*[b]置换总次数。

• *HashValue* 即为 M 的信息摘要(Hash 值)，以十六进

制输出。当用户选择 Hash 值以二进制输出时，主程序使用函数 [*BinString*] = *HexString2BinString* (*HexString*, *Lengh*)进行转换。

• *PerInter* 以 cell 型矩阵储存了算法中所有 *Keccak-f* [*b*]置换的 24 轮中间数据。1×max 型 cell 矩阵，每一 *PerInter*{*i*}为 1×24 型 cell 矩阵(1≤*i*≤max)，储存第 *i* 次置换的 24 轮中间数据。由于 *Keccak-f*[*b*]置换的每一轮迭代均有 5 个置换步骤，因此我们设计 *PerInter*{*i*}{*j*}为 1×6cell 矩阵(1≤*i*≤max,1≤*j*≤24)。其中 *PerInter*{*i*}{*j*}{1}表示进入第 *i* 次置换、第 *j* 轮迭代时的初始状态；当 2≤*k*≤6 时，*PerInter*{*i*}{*j*}{*k*}则表示经过第 *k*－1 步置换后的状态量，储存于 5×5cell 矩阵。

• *state* 以 cell 型矩阵记录每一次 *Keccak-f*[*b*]置换作用后的状态量。

2) 数据处理函数

在程序设计中，我们尽可能充分考虑到各种数据类型及输入格式的需要，通过以下几个函数进行数据处理，增强程序功能。

• 十六进制与二进制字符型数据的相互转换。

[*BinString*] = *HexString2BinString* (*HexString*, *Lengh*)实现十六进制字符串向二进制字符串的转换；

[*HexString*] = *BinString2HexString* (*BinString*, *Lengh*)实现二进制字符串向十六进制字符串的转换。

• 信息输入格式处理。

outArray=*rowDeblank*(*inArray*)去除字符型数组中含有的空格。

3) 使用帮助与用户提示

为更高程度满足用户的研究使用需求，程序用户选项设置较细，因而增加了对程序使用提示及报错功能的要求。多点控制开关与报错函数的配合使用以及帮助提示函数的引入巧妙地克服了该问题。

3.2 主要功能实现与使用

程序运行后，首先显示出如图 2 所示的图形用户界面，用户在数据控制面板上选择相关参数，输入待处理数据，再点击控制面板中的执行按钮，运行结果将在相应窗口显示。程序实现的基本功能是给定信息在指定 Keccak 参数 *r*,*c* 下的 Hash 值运算与输出，其扩展功能是中间数据的输出。与此同时，为满足用户需要，我们添加了若干辅助功能，具体实现与使用如下：

1) Hash 值的运算与输出

如图 2 所示，图形用户界面上方为 SHA-3 规格选择，参照 NIST 对 SHA-3 的要求，共有 4 种规格选择，当用户用下拉菜单选择了 SHA-3 规格后，界面将在 KECCAK 参数显示区，自动显示对应参数 *r*,*c* 值，并在程序内部自动赋值(人为输入 *r*,*c* 无效)。SHA-3 规格选定的 KECCAK 参数既定，此时只需要在信息显示窗口输入待处理信息，选择 Hash 值输出类型，最后点击执行按钮即可，Hash 值将在 Hash 值输出窗口显示。需要指出的是，输入信息类型与所选择的“信息数据类型”需保持一致，否则程序将显示错误提示。

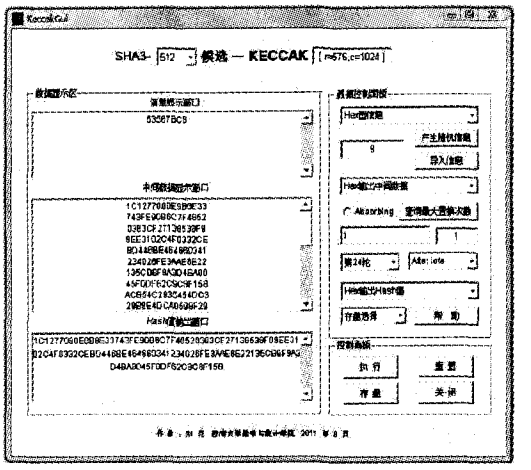

图 2 Keccak 图形用户界面 GUI

2) 中间数据输出

Keccak 算法状态量使用 3 维数组存储，分析算法后我们将中间数据细化到 Absorbing/Squeezing 过程的每一次 *Keccak-f*[*b*]置换中每一轮、每一步置换步骤后的状态量，因而用户需要选择的中间数据选项较多，我们使用“帮助”提示用户使用。具体的中间数据选项操作步骤如图 3 所示。此外，用户使用时若所有中间数据选项均未改动(保持默认值)，程序将认为用户不需要输出任何中间数据。反之，任一中间数据选项被更改，说明用户需要中间数据输出，则用户需要依次完成所有的中间数据选项程序方可正常运行，否则，程序将显示报错提示，指导用户完成选项。

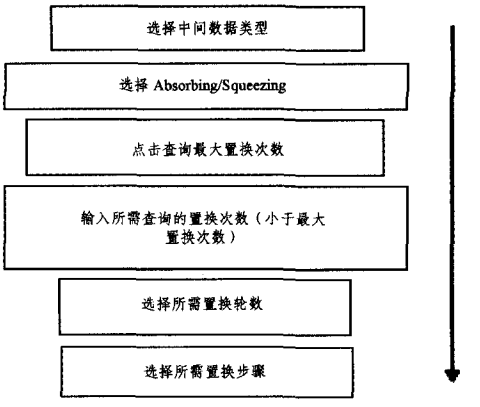

图 3 Keccak 中间数据选项操作步骤

3) 其他辅助功能

为实际研究需要，在程序中添加了若干辅助功能：

• 信息获取功能扩展。

产生随机信息：需要首先选择信息数据类型、输入所需信息长度。为今后的算法随机性分析数据来源打下基础。

导入信息：便于用户对已有信息进行处理，同时适用于大量数据处理输入。

键盘输入：信息量较少情况下提供方便。

• 窗口数据存盘。

程序设计了存盘功能，用户可用下拉菜单选择存盘选项，将所需存储的数据进行存盘。

结束语 为制定新的密码 Hash 算法标准 SHA-3，全世界的专家学者都积极参与到 NIST 组织的 SHA-3 竞赛中。

进入最终决赛的5个候选算法由世界各国的密码学家精心设计，代表了当前Hash函数研究领域最新的研究成果，和高级加密标准(AES)一样，也会成为密码学发展的一个里程碑。

本文在全球公开征集新一代安全Hash标准SHA-3的最后阶段，对可能成为SHA-3标准的Keccak算法进行了初步的研究，并利用Matlab中的Guide，设计开发了带有图形用户界面的Keccak。该程序可以根据需要输出Keccak运行的Hash值及各中间数据，对Keccak的教学及其深入研究都有着积极的意义。

下一步，我们希望对Keccak算法的核心函数*Keccak-f*[*b*]进行深入分析，并尝试对Keccak算法进行随机性检测，更加深入地探讨Keccak算法的安全性。

参考文献

[1] Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm(SHA-3) family[J]. Federal Register Notices,2007,72(212):62212-62220

[2] Bertoni G,Daemen J,Peeters M,et al. The KECCAK SHA-3 submission[EB/OL]. http://keccak.noekeon.org/,2011-01-14

[3] Bertoni G,Daemen J,Peeters M,et al. Keccak implementation overview[EB/OL]. http://keccak.noekeon.org/,2011-01-14

[4] Bertoni G,Daemen J,Peeters M,et al. The Keccak reference. http://keccak.noekeon.org/,2011-01-14

(上接第397页)

频率U较大时，长信号的数据传输量远远低于短信号传输量。

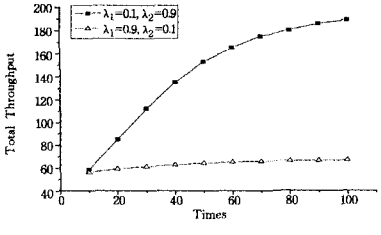

图4 长信号的数据传输量

由图5可见，在权值选取$\lambda_1=0.9,\lambda_2=0.1$的条件下，传输短信号时的频谱利用率高于长信号。反之取$\lambda_1=0.1,\lambda_2=0.9$时，长信号的频谱利用率则高于短信号。由于长信号对频谱可用时长Q的要求较高，导致在频谱可用时长Q较小时，长信号对频谱的利用率大大降低。

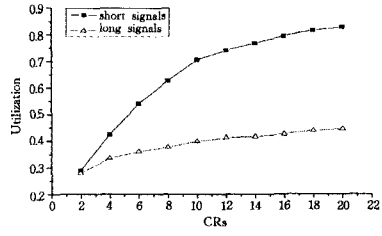

图5 $\lambda_1=0.9,\lambda_2=0.1$时的信道利用率

而短信号对频谱可用时长Q并不敏感，所以在频谱可用时长Q较小的情况下，短信号对频谱的利用率也不会有大幅地减少，如图6所示。

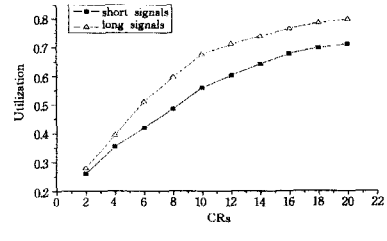

图6 $\lambda_1=0.1,\lambda_2=0.9$时的信道利用率

结束语 本文从信道可用性的角度，同时考虑两种不同的频谱特性，引入了两个不同的特征参数来描述频谱资源可用性，该方法丰富了资源的描述特征，对提高频谱利用率具有一定作用。针对不同用户对信道长短要求不一的特点，文中根据频谱状态更新频率和可用时长两个特征对频谱可用性同时进行描述，用户可根据自身需求自适应来调整两个参数的权值。当业务为长信号时，设置较大的频谱平均空闲概率以获得适当的频谱；短信号业务则可以主要考虑频谱可用率。该方法同时可用于研究时延性质的场景，能接受时延的服务可以设置较大的频谱可用性，以提高频谱利用率；不能接受时延的业务则主要考虑频谱平均空闲概率，以满足业务要求。综上所述，本文提出的方法具有较高的普遍适用性，同时仿真结果也表明，该方法可以有效地提高频谱利用率。

参考文献

[1] Haykin S. Cognitive radio: brain-empowered wireless communications[J]. IEEE Journal on Selected Areas in Communications, 2005,23(2):201-220

[2] Wild B,Ramchandran K. Detecting primary receivers for cognitive radio applications[C]// Proceedings of 1st IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN'05). Baltimore, MD, USA, Piscataway,NJ,USA:IEEE,2005:124-130

[3] Haykin S, Huber K, Chen Z. Bayesian sequential state estimation for MIMO wireless communications[J]. Proceedings of the IEEE,2004,92(3):439-454

[4] 刘勤. 认知无线通信系统的频谱资源管理[J]. 中兴通讯技术, 2009(2)

[5] Weiss T A, Jondral F K. Spectrum pooling an innovative strategy for the enhancement of spectrum efficiency[C]// IEEE Radio Communications. 2004:8-14

[6] Guo Cai-li,Zeng Zhi-min,Feng Chun-yan,et al. Novel proactive spectrum selection algorithm for cognitive radio networks[J]. Journal of Xidian University: National Edition,2008(6):1121-1126