中国科学: 信息科学 2010 年 第 40 卷 第 10 期: 1299～1311
www.scichina.com info.scichina.com
《中国科学》杂志社
SCIENCE CHINA PRESS

论 文

多变量 Hash 函数的构造理论与方法

王后珍[①②], 张焕国[①*], 伍前红[①], 张雨[①], 李春雷[①], 张欣雨[①]

① 武汉大学计算机学院空天信息安全与可信计算教育部重点实验室, 武汉 430072
② 中国科学院数学机械化重点实验室, 北京 100080
* 通信作者. E-mail: liss@whu.edu.cn

收稿日期: 2009–08–18; 接受日期: 2010–04–20
国家自然科学基金 (批准号: 60673071, 60970115, 60970116)、国家高技术研究发展计划 (批准号: 2007AA01Z411) 和中国科学院数学机械化重点实验室开放课题 (批准号: KLMM0903) 资助项目

摘要 文中基于多变量非线性多项式方程组的难解性问题, 并结合 HAIFA 迭代框架, 提出了一种新的 Hash 算法, 它与目前广泛使用的 Hash 算法相比具有下列优点: 安全性基于一个公认的数学难题; 输出 Hash 值长度可变; 设计自动化, 用户可根据实际需求构造满足其特定要求的 Hash 函数. 同时还详细讨论了新算法的安全性、效率和性能, 通过理论分析, 选取适当参数的新算法, 在其难解性问题的假设前提下, 可达到理想 Hash 函数的安全性强度; 实验结果表明, 新算法在效率和性能方面与其他 Hash 函数具有可比性. 此外, 由于新算法具有良好的随机性, 还可将其作为一种伪随机数生成器.

关键词
密码学
Hash 函数
MQ 问题
多变量多项式

1 引言

Hash 函数在数字签名、完整性检验、身份认证和动态口令鉴别等现代信息安全技术中被广泛应用, 它是一类特殊的单向函数, 对于输入的任意长度消息序列, 输出固定长度的 Hash 函数值. 目前使用最广泛的 Hash 函数是 MD 系列和 SHA 系列, 它们都是基于逻辑运算直接构造的, 近年来, 对于这类 Hash 函数的研究取得了重大突破, MD5 已不安全 [1], 虽然还未找到 SHA-1 的碰撞, 但其计算复杂度已经约减到 2^{63} 次 [2]. 文献 [3, 4, 5] 中还对带密钥的 Hash 函数的安全性进行了详细的分析. 美国国家标准和技术研究所在 2005 年 10 月和 2006 年 8 月举办了两次研讨会, 评估了当前 Hash 函数的使用状况, 决定公开征集新一代 Hash 函数标准 SHA-3[6]. 2009 年 7 月第 1 轮评选结束, 共有 14 个候选算法进入第 2 轮评选, 预计这一计划将在 2012 年完成.

Hash 函数的设计主要分为 3 类方法: (1) 直接构造法. 它不基于任何假设及密码体制, 而是采用大量的逻辑运算来确保 Hash 函数的安全性. MD 系列和 SHA 系列的 Hash 函数均是采用这种方法设计的, 但文献 [1, 2] 的分析结果使人们对这种构造方法本身的安全性产生了严重质疑. (2) 基于分组密码算法的 Hash 函数, 其安全性依赖于分组密码的安全性, 文献 [7] 分析了基于分组密码设计的 Hash 函数的安全性. (3) 基于难解性问题的构造法. 利用一些难解性问题诸如离散对数、因子分解等可构造 Hash 函数, 在合理的假设下是可证明安全的, 但一般来讲其效率低, 缺乏实用性.

引用格式: 王后珍, 张焕国, 伍前红, 等. 多变量 Hash 函数的构造理论与方法. 中国科学: 信息科学, 2010, 40: 1299–1311

本文基于多变量非线性多项式方程组的难解性问题构造了一种新的 Hash 函数, 与基于其他困难性问题设计的 Hash 函数相比, 它的效率主要取决多变量多项式的稀疏程度, 我们可以权衡安全性和实用性来控制多变量多项式方程组的稠密度, 以构造出满足用户特定需求的多变量 Hash 函数.

2 多变量 Hash 函数的构造

2.1 MQ 问题与 Hash 函数

有限域 F_q 上 n 个变元的二次多项式方程组形式为

$$p_i(x_1,\ldots,x_n)=\sum_{1\leqslant j\leqslant k\leqslant n}a_{i,j,k}x_jx_k+\sum_{1\leqslant j\leqslant n}b_{i,j}x_j+c_i, \tag{1}$$

其中, $1\leqslant i\leqslant m, a_{i,j,k}, b_{i,j}, c_i\in F_q$.

定义 1 随机给定一个 F_q 上 n 个变量的二次方程组 $P=(p_1,\ldots,p_m)$, $m\leqslant n$. 对于已知元素 $y\in F_q^m$, 求解一个元素 $x\in F_q^n$, 使得 $y=(p_1(x),\ldots,p_m(x))$, 称为 MQ 问题.

文献 [8] 中指出 MQ 问题是一个 NP 难解性问题. 同时也指出, 当上述方程组的次数大于 2 时, 它的求解问题仍然是一个 NP 难解性问题, 我们常把这类求解问题统称为 MQ 问题.

定理 1 对于给定的任意值 $\delta=(\delta_1,\ldots,\delta_n)$, 可以在多项式时间 $O(mn^2)$ 内, 寻找到一对变量 $x=(x_1,\ldots,x_n)$, $y=(y_1,\ldots,y_m)$ 满足 $y-x=\delta$, 使得 $P(x)=P(y)$.

证明 设 $L_\delta: t\mapsto P(t+\delta)-P(t)$, 对于给定的 δ, 显然差分方程 $L_\delta(t)=0$ 是关于变量 t 的线性方程组, 因此能在 $O(mn^2)$ 内求解 t, 从而得到 $P(t+\delta)=P(t)$.

定理 1 表明, 利用二次 MQ 问题直接构造多变量 Hash 函数是不安全的, 不能抵御多变量差分攻击 [9]. 因此本文采用有限域上高次 (最高项次数大于 2) 多变量多项式方程组来构造新 Hash 函数的压缩函数, 并将这种新的多变量 Hash 函数称之为 MPH(multivariate polynomails for Hash). 结合安全性和效率, 文中采用有限域上的多元三次多项式构造 MPH 的压缩函数.

2.2 MPH 的结构

定义 2 MPH 的压缩函数 $\mathrm{CF}(x)$ 定义为 $\mathrm{CF}: F_q^n\mapsto F_q^m$,

$$\mathrm{CF}(x)=(f_1(x_1,\ldots,x_n),\ldots,f_m(x_1,\ldots,x_n)),$$

这里 $x=(x_1,\ldots,x_n)\in F_q^n$, $q=2^s, m, s\in\mathbb{Z}^+$, $n=2m$, 函数 f_i 是在有限域 F_q 上随机选择的 n 元三次多项式, 其形式如下:

$$f_i(x)=\sum_{1\leqslant j\leqslant k\leqslant t\leqslant n}a_{i,j,k,t}x_jx_kx_t+\sum_{1\leqslant j\leqslant k\leqslant n}b_{i,j,k}x_jx_k+\sum_{1\leqslant j\leqslant n}c_{i,j}x_j+d_i, \tag{2}$$

其中, $1\leqslant i\leqslant m, a_{i,j,k,t}, b_{i,j,k}, c_{i,j}, d_i\in F_q$.

压缩函数 $\mathrm{CF}(x)$ 多项式方程组的项数最大值为 $n(n+1)(n+2)(n+3)/12$, 其中方程组的三次项个数为 $n^2(n+1)(n+2)/12$, 二次项的个数为 $n^2(n+1)/4$, 一次项的个数为 $n^2/2$, 常数项的个数为 $n/2$. 以 $x\in F_{256}^{64}$ 为例, 此时 Hash 值长度为 256 bit, 则压缩函数 $\mathrm{CF}(x)$ 大约需要 1.5 MB 的存储空间, 效率也很低, 缺乏实用性. 因此, 本文压缩函数 $\mathrm{CF}(x)$ 采用指定基本域 F_q、变量个数 n 和三次项项数 Terms 的随机稀疏多项式, 同时保留全部低次项. 对于任意一个三次项 $ax_ix_jx_k$ 而言, 我们需要存储其

系数 $a \in F_q$ 以及相应的下标 $i, j, k \in [1, n]$, 而对于所有的二次项 bx_ix_j、一次项 cx_i 以及常数项 d 而言, 由于没有稀疏化处理, 故按一定规则排序后, 存储其相应系数 $b, c, d(\in F_q)$ 即可.

由于传统的 Merkle-Damgåard 结构安全存在缺陷, 本文采用改进的 Merkle-Damgård 结构 HAIFA[10] 来构造多变量 Hash 函数 MPH. 算法 1 描述 MPH 对任意消息的具体杂凑过程.

算法 1 MPH 算法

Input: 任意报文 M, MPH 的基本域 F_q 参数 $q = 2^s$, $s \in \mathbb{Z}^+$

Output: h 比特 Hash 值 $H(M)$.

填充报文. 数据填充方式与 SHA-x 系列类似. 填充后的报文分组为 $M_1, \ldots, M_L$, 其中 M_i 的长度为 h 比特, $h = ns/2$. 再将每个分组 M_i 编码为基本域 F_q 上一个 $n/2$ 维向量;

初始化 $n/2$ 维向量 IV, 盐变量 $salt$ 和计数器 counter 均是 $n/2$ 维向量, 且其初值均为 0;

CV_0=IV;

for $i = 1; i \leqslant L; i++$ **do**

  $CV_i = CF((CV_{i-1} \wedge salt_{i-1})||(M_i \wedge counter_{i-1}))$;

  for $j = 1; j \leqslant n/2; j++$ **do**

    if $j \leqslant n/4)$ **then**

      $\mathrm{salt}_i[j] = \mathrm{CV}_{i-1}[j] \wedge \mathrm{CV}_i[n/2+1-j]$;

      else

        $\mathrm{salt}_i[j] = \mathrm{CV}_{i-1}[j] \wedge \mathrm{CV}_i[j-n/4]$;

      end

    end

    $\mathrm{counter}_i[n/2+1-j] = (i >> 4(j-1))\&0\mathrm{x}0\mathrm{F}$;

  end

end

$H(M)$ =encodebits(CV_L);

return 返回 h 比特位 Hash 值 $H(M)$.

算法 1 给出了新 Hash 算法的具体实现方法, 这里符号 || 表示将有限域 F_q 上的两个 $n/2$ 维向量级联成一个 n 维向量, encodebits(CV_L) 表示将链变量 CV_L 编码为 h 位二进制序列, 符号 $\wedge$、& 和 $\gg$ 分别表示逐位逻辑异或、逻辑与和右移运算.

3 安全性分析

3.1 原像攻击

由于 $z = \mathrm{CF}(x)$ 为 n 个变元 $n/2$ 个方程的多变量 3 次方程组, 属于 MQ 问题. 已知消息摘要 z 要求解 x 使得 $z = \mathrm{CF}(x)$, 一般采用固定值方法，即将 $\mathrm{CF}(x)$ 中 n 个变量随机固定 $n/2$ 个, 将其转化为置换方程组. 然后利用 F5[11] 等常规算法求解. 因此只要选取合适的参数, 便能到达足够的安全性.

这里需要说明的是虽然求解有限域上次数大于 2 的方程组也是 NP 困难性问题, 但目前讨论其求解复杂度的文献较少, 因此本文在讨论高次方程组求解问题的复杂度时, 总是将其当作二次 MQ 问题来分析, 显然这是合理的. 当采用将压缩函数的三次项稀疏化处理后, 由于低次项仍然是稠密的, 因此

即使丢掉所有的三次项, 此时就原像攻击而言, 仍需求解 MQ 问题, 故本文稀疏化高次项的 MPH 能抗原像攻击.

3.2 第二原像攻击和碰撞攻击

由于第二原像攻击和碰撞攻击有包含的关系, 下面我们只讨论 $\mathrm{CF}(x)$ 的抗碰撞攻击, 假设碰撞差分为 $\Delta x \in F_q^n$, 寻找 $\mathrm{CF}(x)$ 的碰撞等价于解差分方程 $\mathrm{CF}(x+\Delta x)-\mathrm{CF}(x)=0$, 只要 Δx 不为零, 差分方程的最高次数为 2, 根据定义 1 上述求解过程为 MQ 问题, 与原像攻击具有相同复杂度. 因此, MPH 的压缩函数 $\mathrm{CF}(x)$ 是强无碰撞的.

下面讨论稀疏 $\mathrm{CF}(x)$ 的抗碰撞性, 通过上面的分析, 它等价于求解稀疏 MQ 问题, 我们先介绍有关这一方面的研究进展. 求解非线性多元多项式方程组的传统方法是构造 Gröbner 基的 Buchberger 算法及其变型 F4、F5 算法. 它的基本思想是先把单项式排序, 然后根据适当的多项式系数合并两个方程来约去最高位的单项式. 重复此过程直到所有方程中的同一个变量被约去, 最后解余下改变过的多项式方程组. 但遗憾的是, 实际操作中剩余单项式的次数在消元过程中急剧增加, 因此即使对于一个中等数量的未知量, 此算法的时间复杂度都巨大而不可行. 在最坏的运行情况下, 可知 Buchberger 算法要运行两倍指数时间. Shamir 等人在文献 [12] 中证明了当 MQ 方程是超定时, 解方程的复杂度大大降低, 并提出了求解超定 MQ 问题的 XL 算法, 文献 [13] 中提出了改进的 XL 系列算法如 XSL 算法等, 文献 [14] 给出了求解超定 MQ 的复杂度. 文献 [13, 15] 中指出如果 MQ 是稀疏的、且具有一个规则的结构, 那么问题就很容易解决, 但对于一般的稀疏 MQ 问题并不十分有效. 文献 [16] 中引入了 Dixon 结式, 并提出了一种新的求解 MQ 问题的 DR 算法, 与文献 [13, 15] 类似, 只对特殊的稀疏方程有效. 文献 [17] 较为详细的讨论了稀疏 MQ 问题, 并在大型工作站作了测试, 实验结果表明, 目前求解 MQ 问题的方法对其方程组的项数不太敏感. 从目前的研究进展来看, 有限域 $F_q(q>2)$ 上随机稀疏 MQ 问题仍然是难解的, 这也是本文多变量 Hash 函数 MPH 结构安全性的理论依据.

由于差分攻击的原因, 三次多项式压缩函数 $\mathrm{CF}(x)$ 抗碰撞能力等价于求解其差分二次方程组的复杂性, 如果 $\mathrm{CF}(x)$ 过于稀疏, 则其差分函数也可能为稀疏多项式, 通过上面的分析, 虽然目前没有证据表明稀疏 MQ 问题比稠密 MQ 问题易解, 但在选取压缩函数 $\mathrm{CF}(x)$ 时, 应该尽量保证其差分函数的稠密性. 接下来我们将分析压缩函数 $\mathrm{CF}(x)$ 中每个方程的三次项项数 Terms 与其二次差分方程组的稠密度之间的关系.

定义 3 设压缩函数 $\mathrm{CF}(x)$ 的差分函数为 $\mathrm{DCF}(x)=\mathrm{CF}(x+\Delta x)-\mathrm{CF}(x)$, 方程中第 i 个多项式中 d 次项的稠密度定义为 $\rho(i,d)=\mathrm{Terms}(i,d)/N(d)$; n 元压缩函数 $\mathrm{CF}(x)$ 及其相应差分函数 $\mathrm{DCF}(x)$ 的稠密度分别定义为

$$\rho_3=\frac{2}{n}\sum_{i=1}^{n/2}\rho(i,3), \rho_2=\frac{2}{n}\sum_{i=1}^{n/2}\rho(i,2).$$

这里 $\rho(i,d)$ 表示多项式中 d 次项的实际项数与理论上的最大项数之比.

由于我们只对压缩函数 $\mathrm{CF}(x)$ 的三次项进行了稀疏处理, 而低次项是稠密的, 显然差分函数 $\mathrm{DCF}(x)$ 的低次项也是稠密的, 因此本文在度量函数的稠密性时只考虑函数中高次项的稠密度. 利用前面的压缩函数构造方法随机生成指定三次项项数 Terms 的压缩函数, 并统计其相应差分方程的稠密度, 实验结果如图 1 所示.

图 1 给出了 $\mathrm{GF}(2^8)$ 上 MPH-x 系列 $\mathrm{CF}(x)$ 的稠密度 ρ_3 与 $\mathrm{DCF}(x)$ 的稠密度 ρ_2 之间的关系, 其中, x 轴表示压缩函数 $\mathrm{CF}(x)$ 的稠密度 ρ_3, y 轴表示相应差分函数 $\mathrm{DCF}(x)$ 的稠密度 ρ_2. 从图上可看

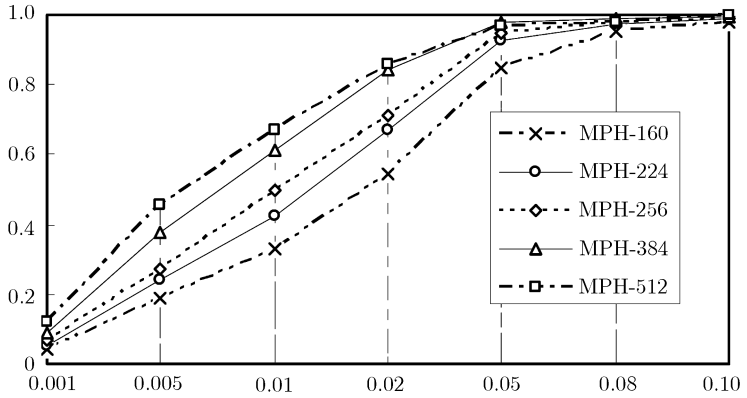

图 1 GF(2^8) 上 MPH-x 系列 CF(x) 的稠密度 ρ_3 与 DCF(x) 的稠密度 ρ_2 之间的关系

出, 当 $\rho_3 \to 0.1$ 时, $\rho_2 \to 1$, 其次, 当 ρ_3 取固定值时, 压缩函数的变量个数 n 越大 (即 Hash 长度值越大), 其相应差分函数的稠密度 ρ_2 越大, 这意味着对于较长 Hash 值的 MPH, 其压缩函数可以更为稀疏; 图 1 还表明适当稀疏化处理的压缩函数, 也能保证其差分函数的稠密性, 这给我们具体构造压缩函数提供了指导, 同时也说明本文的构造方法是合理的. 通过上述实验和安全性分析, 可以得到如下结论: 当压缩函数 CF(x) 的稠密度 $\rho_3 \geqslant 0.1$ 时, 若假设 MQ 问题是不可计算的, 则 MPH-x 系列算法可以认为是绝对安全的.

3.3 基于传统 Merkle-Damgård 结构的攻击

自 MDx 系列 Hash 函数受到攻击后, Merkle-Damgård 迭代结构暴露出了诸多安全缺陷, 例如长度扩展攻击、多碰撞攻击、Herding 攻击等. 近年来涌现出了很多新的 Hash 函数构造结构, 它们大多都是传统 Merkle-Damgård 结构的变形, 例如 AsiaCrypto2005 上 Lucks 给出的宽管道 (wide-pipe) 和双管道 (double-pipe) 两种迭代结构 [18] 以及 Biham 和 Dunkelman 在文献 [10] 中提出的 HAIFA(Hash iterative framework) 迭代框架等, 这些改进结构被广泛应用于设计新的 Hash 算法, 如 2009 年 7 月进入 SHA-3 第二轮评选的候选算法 BLAKE 和 ECHO 就是采用 HAIFA 迭代框架. 本文我们也采用 HAIFA 结构来构造多变量 Hash 函数 MPH, 故本文 MPH 算法也就继承了 HAIFA 的一些优点, 能抵抗对于传统 Merkle-Damgård 结构一些常见攻击方法, 诸如多碰撞 (multicollision)[19]、长度扩展 (length extension)、固定点 (fixed-point)[20]、Herding[21] 等攻击方法.

4 仿真实验及性能评估

4.1 MPH 的存储空间

MPH 的存储包括两个部分: 指定有限域上的乘法表和压缩函数. 乘法表的大小取决于基本域 F_q, 压缩函数的存储空间与其变量的个数 n 、及其每个方程的三次项项数 Terms 有关. 这里我们只讨论 MPH 压缩函数的存储, 其存储空间 S(单位:bit) 可表示为

$$S(n, q, \text{Terms}) = \frac{n}{2}\left[(3\log_2^n + \log_2^q)\text{Terms} + \frac{(n+1)(n+2)}{2}\log_2^q\right]. \tag{3}$$

我们为优化压缩函数 CF(x) 的存储空间, 设计了 CF(x) 的一种生成方法 (见算法 2).

算法 2 压缩函数 CF(x) 的生成算法

Input: 有限域 F_q 的参数 q, Hash 值长度 h 及三次项项数 Terms.

Output: 压缩函数 CF(x).

验证参数 q, h, Terms 是否满足 $q = 2^s$, $s|h$ 以及 Terms$\in [1, N]$. 这里 $s, h \in \mathbb{Z}^+$, $N = n(n+1)(n+2)/6$, $s, h \in \mathbb{Z}^+$, $n = 2h/s$;

生成 $n/2-1$ 个随机因子 RF[i] $\in (1, N)$, 且满足 GCD($RF[i], N) = 1$, GCD(RF[i]$\%q, q) = 1$, $RF[i]\%q$ 不等于零以及数组 $RF, RF\%q$ 的中没有重复元素;

for $j = 1; j \leqslant$ Terms; $j++$ **do**

  Term3[1][j] =get_random()$\%N$;

end

for $j = 1; j \leqslant (n+1)(n+2)/2; j++$ **do**

  Term210[1][j] =get_random()$\%q$;

end

for $i = 2; i \leqslant n/2; i++$ do

  for $j = 1; j \leqslant$ Terms; $j++$ **do**

    Term3[i][j] = (Terms[j] $\times$ RF[j])$\%N$;

  end

  for $j = 1; j \leqslant (n+1)(n+2)/2; j++$ **do**

    Term210[i][j] = (Term210[1][$(n+1)(n+2)/2 - j + 1$] $\times$ RF[j])$\%q$;

  end

end

Return: CF(x)={Term3,Term210};

算法 2 中 get_random() 表示随机数生成函数, 二维数组 Term3 用于存储压缩函数 CF(x) 的三次项, 三次项相应的系数 $C[i][j]$ = Term3$[n/2-i+1][j]\%q$, 而二维数组 Term210 用于存储压缩函数 CF(x) 的二次项、一次项及常数项. 压缩函数的后 $n/2-1$ 个方程由第一个方程通过随机因子 RF 导出. 因此我们在存储 CF(x) 时只需存储它的第一个方程即可.

根据算法 2, 压缩函数 CF(x) 的存储空间为 S'(单位:bit) 可表示为

$$S'(n, q, \text{Terms}) = \text{Terms} \cdot \log_2 \frac{n(n+1)(n+2)}{6} + \frac{(n+1)(n+2)}{2} \log_2^q . \tag{4}$$

与 (3) 式相比至少节省 $n/2$ 倍的存储空间, 例如 GF(2^8) 上三次项项数 Terms=500 的压缩函数 CF(x) 构成的 Hash 函数 MPH-256, 采用 (3) 式存储需 120.64 KB, 而采用 (4) 式则只需 3.11 KB. 由于压缩函数 CF(x) 的第一个多项式是随机选取的, 且后面的多项式是由第一个多项式通过随机因子导出的, 因此压缩函数 CF(x) 具有整体随机性.

Term3 中元素的生成原理: 每个方程的三次项项数理论值为 $N = n(n+1)(n+2)/6$ 项, 我们将其按一定规则排序后, 则任意一项 $x_i x_j x_k$ 与 $R \in [0, N)$ 一一对应, 将整数 R 转化为分量下标 i, j, k 的方法见算法 3.

为了更好的说明本文构造算法的性能, 我们利用算法 2、3 给出了一个 GF(2^8) 上的具体实例 MPH-160(见附录), 下文简称之为 MPH-160 实例, 并对其进行相关性能测试.

算法 3 压缩函数 CF(x) 三次项的解码算法

Input: 参数 $R \in [0, N)$.

Output: 三次项 $x_i x_j x_k$ 的分量 $\{i, j, k\}$.

定义函数 $N[y] = y(y+1)(y+2)/6$; $t = n-1$;
while $N[n] - N[t] < R$ **do**
  $i = n - t - 1$; $M = N[n] - N[i]$; $R = R - M - 1$; $t--$;
end
$r = R\%(n-i)$; $l = (R-r)/(n-i)$;
if $r < \ell$ **then**
  $j = n - \ell; k = n - r - 1$;
else
  $j = \ell + i; k = r + i$;
end
Return: 返回 $x_i x_j x_k$ 的下标分量 $\{i, j, k\}$.

4.2 MPH 的效率分析

本文新 Hash 函数的基本域 F_q 取 $q = 2^8$, 压缩过程中主要运算为上的乘法运算和加法运算, 因此实现效率较高, 也易于硬件实现. 由于基本域 F_q 较小, 因此可预运算并存储其乘法表, 与 AES 相仿乘法运算可通过查表完成. 压缩函数中一个 d 次项需要 d 次查表运算, 从而新 Hash 函数的效率主要取决于其压缩函数的项数, 项数越少效率越高, 同时, 当压缩函数的项数一定时, 高次项的比重越小, 其运算效率越高.

表 1 给出了 MPH 算法压缩函数 CF(x) 的复杂性, 实现效率与具体的实现平台有关, 其中 q 为有限域参数, Terms 为压缩函数的三次项项数, n 为变量的个数, 本文实验环境为 Inter Core2 P8600 2.4 GHz CPU、2 G 内存的 PC 机, 实现平台为 Windows XP Visual C++6.0. 在我们的平台上 SHA-1 的效率为 69.3577 cycles/byte, 而附录 MPH-160 实例为 1456.5 cycles/byte, 即实例大约比 SHA-1 慢 21 倍; 从前面的安全性分析可看出, MPH 压缩函数的安全性主要基于其差分函数的难解性, 因此差分函数的稠密度 ρ_2 是本文 MPH 算法安全性度量的重要指标. 表 2 给出了当指定差分函数的稠密度 ρ_2 时, 相应压缩函数稠密度 ρ_3 的大致取值、以及新算法 MPH 与相应 SHA 算法的效率比较, 其中 η 表示相同 Hash 值长度的 MPH-x 与 SHA-x 之间的效率之比.

表 2 给出了 MPH 算法安全性与效率之间的关系, 在具体构造其压缩函数时, 可根据表 2 做出效率与安全性的权衡. 通过 3.2 小节的分析, 目前仍然没有求解稀疏 MQ 问题的有效方法, 即便如此, 我们仍然推荐, 在实际应用中应保证差分函数稠密度 ρ_2 至少为 0.5, 另外, 当差分函数稠密度 $\rho_2 \geqslant (q-1)/q$ 时, 我们认为差分函数是稠密的, 由于压缩函数 CF(x) 是随机选取的, 这显然是合理的, 本文有限域取 GF(2^8), 因此当 $\rho_2 \geqslant (256-1)/256 = 95.86$ 时, 我们认为 MPH 算法是绝对安全的, 从表 2 中可看出满足这种绝对安全的 MPH 算法 (如 160、256、384、512 位), 比相应长度的 SHA 算法分别慢 42、121、231、699 倍.

4.3 随机性统计分析

对于任意长度的消息比特序列, Hash 函数都会将其变成某一固定长度的摘要, 从安全性角度来说,

表 1　MPH 算法的复杂性

运算类型	运算次数
查表运算	$n(3\text{Terms}+n^2+2n)/2$
逐位异或运算 “∧”	$(\text{Terms}+n^2+3n+12)n\log_2^q/4$
逻辑与运算 “&”	$n\log_2^q/2$
右移运算 “≫”	$n\log_2^q/2$

表 2　MPH-x 与 SHA-x 系列算法的效率比较

	MPH-160		MPH-224		MPH-256		MPH-384		MPH-512	
ρ_2	ρ_3	η	ρ_3	η	ρ_3	η	ρ_3	η	ρ_3	η
0.50	0.018	21	0.013	–	0.011	61	0.007	97	0.006	238
0.60	0.024	24	0.017	–	0.015	66	0.010	112	0.008	259
0.70	0.031	27	0.022	–	0.018	72	0.013	128	0.011	286
0.80	0.042	29	0.030	–	0.026	80	0.017	143	0.015	345
0.90	0.065	34	0.045	–	0.037	98	0.024	171	0.022	426
>0.96	0.092	42	0.076	–	0.055	121	0.046	231	0.043	699

摘要序列应该具有良好的随机性. 美国国家技术与标准局 (NIST) 推出的标准测试套件 STS(statistical test suite)[22] 是目前伪随机测试中最具权威的工具之一.

STS 套件中的每种检测方法对于某一组测试序列均会产生相应的 P-value. 若 P-value$\geqslant\alpha$, 则认为该序列通过该项测试. 其中显著性水平均为 $\alpha=0.01$. NIST 采用两种方法解释测试结果, 即通过率和 P-value 分布的均匀性.

通过率. 测试算法独立生成组待测序列, 根据各次测试的 P-value 是否大于 $\alpha=0.01$ 计算通过率. 先计算

$$\hat{p}\pm 3\sqrt{\frac{\hat{p}(1-\hat{p})}{m}},\hat{p}=1-\alpha. \tag{5}$$

若各次测试通过率落入可信区间 $(\hat{p}-3\sqrt{\hat{p}(1-\hat{p})/m},\hat{p}+3\sqrt{\hat{p}(1-\hat{p})/m})$(其中 $m\geqslant 1000$), 则可认为所测算法为信任度高的随机序列.

P-value 分布的均匀性. 测试算法独立生成 m 组待测序列, 根据各次测试的 P-value, 计算

$$\sum_{i=1}^{10}\frac{(F_i-m/10)^2}{m/10},$$

其中, F_i 是在区间 $[(i-1)\times 0.1,i\times 0.1)$ 的 P-value 的个数, m 为测试序列的组数. 然后再计算 $P\text{-value}_\tau=\text{igamc}(\frac{9}{2},\frac{\chi^2}{2})$, 其中 $\text{igamc}(n,x)$, 是不完全 Gamma 函数.

若 $P\text{-value}_\tau\geqslant 0.0001$, 则可以认为测试序列是均匀分布的.

本文采用 NIST STS(Version 1.8) 测试多变量 Hash 函数 MPH, 它共由 15 项核心测试组成, 文献 [22] 中给出了每种测试方法的详尽描述. 表 3 中前 3 列分别列出了每一项的测试编号、测试名称和每一项的测试次数.

根据 STS 的测试条件, 生成一个 1 G 比特的待测二进制序列, 以 ASCII 码 0 和 1 分别表示 1 比特的 0 和 1, 并存储为一个 1 G 字节的测试数据文件, 其生成方法与文献 [22] 中的 G-SHA1 算法类似, MPH-160 实例的测试结果如表 3.

表 3 MPH-160 实例的随机性测试结果

序号	测试项目	测试次数	P-value$_\tau$ 最小值	最小通过率
1	Rrequency	1	0.125200	0.9910
2	Block-frequency	1	0.259616	0.9910
3	Cumulative-sums	2	0.267573	0.9900
4	Runs	1	0.030197	0.9920
5	Longest-run	1	0.911413	0.9880
6	Rank	1	0.564639	0.9900
7	DFT	1	0.036592	0.9920
8	Nonperiodic-templates	148	0.000991	0.9810
9	Overlapping-templates	1	0.733899	0.9890
10	Universal	1	0.542101	0.9880
11	Approximate entropy	1	0.990138	0.9870
12	Random-excursions	8	0.226559	0.9822
13	Random-excursions-variant	18	0.053823	0.9838
14	Serial	2	0.684890	0.9910
15	Linear-complexity	1	0.440975	0.9910

根据 (5) 式可得算法通过率的可信区间为 $[0.9805607, 0.9994392]$, 其次 P-value 的均匀性参数 P-value$_\tau$ 应大于等于 0.0001, 从表 3 结果可看出 15 个测试项目全部通过测试, 这说明本文提出的多变量 Hash 函数 MPH 算法完全符合随机性要求. 同时说明本文的 MPH 算法还可用于构造伪随机数发生器.

4.4 雪崩性质测试分析

为了隐藏明文消息的冗余度, Shannon 提出了混乱与散布的概念, 加密体制中要求充分且均匀地利用密文空间, Hash 函数同样如此, 要尽量做到相应消息串与对应的 Hash 值不相关, 而对于结果的二进制表示, 每 1 bit 只有 0 或 1 两种可能, 因此理想 Hash 的散布效果应该是初值的细微变化将导致结果的每比特都以 50% 的概率变化. 考察 Hash 算法在消息串发生 1 bit 变化的情况下, 引起 Hash 结果的变化比特数为 B.

定义 4 设平均变化比特数 $\bar{B}$、平均变化概率 P、变化比特数 B 的均方差 ΔB 和 P 的均方差 ΔP 分别为

$$\bar{B}=\frac{1}{N}\sum_{i=1}^{N}B_i,\quad P=\frac{\bar{B}}{h},\quad \Delta B=\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(B_i-\bar{B})^2},\quad \Delta P=\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\frac{B_i}{h}-P)^2},$$

其中 N 为统计次数, B_i 为第 i 次测试变化的比特数, h 为 Hash 值的长度.

每次测试方法为在消息空间中随机选取一段消息进行新的 Hash 函数压缩, 然后改变消息串 1 bit 得到另一 Hash 结果, 比较两个结果得到变化的比特数为 B_i. 表 4 给出附录 MPH-160 算法的雪崩统计结果, 其中统计次数 N=1000.

由表 4 可知, MPH-160 算法的 $\bar{B}$ 和 P 非常接近理想状况下 80 和 50% 的变化概率, 相当充分均匀地利用了消息空间, 消息的任何扰动, 都使得 Hash 结果在统计上产生接近等概率的均匀分布, 从统

表 4　GF(256) 上 MPH-160 的雪崩性测试结果

三次项项数	$\bar{B}$	ΔB	P	ΔP
10	79.828000	6.893035	0.498925	0.499175
50	79.921000	6.712214	0.499506	0.499756
100	79.795000	6.729656	0.498719	0.498968
200(实例)	80.007100	6.901999	0.500416	0.500612
500	80.041000	6.831121	0.500507	0.501157
1000	79.145000	6.887264	0.500969	0.501219
全选	80.081200	6.665199	0.501020	0.503121

计效果上看, 保证了攻击者在已知一些消息 Hash 值对的情况下无法得到任何 Hash 值分布的有用信息. 而 ΔB、尤其 ΔP 越接近于零, 则表明算法对消息的混淆与扩散性质的稳定性越好, 表 4 显示 MPH 算法具有质良好的雪崩性质.

5　参数选取说明

本文采用三次多项式来构造 MPH 的压缩函数 CF(x), 主要是从算法效率角度考量, 理论上讲也可采用高次多项式来设计, 但目前研究有限域上高次多项式方程组求解复杂性的文献较少, 其具体复杂性难以度量, 因此权衡安全性和效率我们采用三次多项式.

由于本文 MPH 算法的主要运算为有限域的乘法运算, 适于通过在小域上的查表操作完成压缩算法, 一般推荐 GF(2^8), 这样与目前的密码设计主流相吻合, 如 AES 等均是以字节为处理单位以便于工程实现. 但文献 [23] 中指出采用 GF(2) 是不安全的. 因此选取有限域时除了 GF(2) 外均可.

为提高压缩函数的效率必须对其稀疏化处理, 我们采取保留全部低次项, 稀疏化高次项的方法, 这主要是因为对于一般的多项式, 高次项占绝对比重, 例如假如 MPH-160 算法采用稠密三次多项式构造压缩函数, 则三次项项数为 11480 项, 低次项共 861 项 (包括系数为 0 的项). 显然这种稀疏处理是合理的; 另外一个更为重要的原因是, 如果直接采用整体稀疏化处理, 会以很高的概率出现一类特殊的多项式, 即齐次多项式或缺少某个变量的多项式, 很明显, 此时存在变量 x 满足碰撞 $\mathrm{CF}(0,\ldots,0)=\mathrm{CF}(0,\ldots,x=a,\ldots,0)$, 其中 a 为固定值, 文献 [23] 中将 x 称为 “孤立” 变量, 并认为文献 [17] 中的多项式整体稀疏化方法存在安全缺陷, 通过 3.1、3.2 小节的安全性分析, 只要保留所有低次项, 便能防止这类特殊多项式的出现.

本文的有限域采用 GF(2^8), 故压缩函数的变量 n 一般取 8 的倍数, 方程的个数等于 $n/2$, 此时 MPH 的输出值为 $4n$ bit, 只要相应的变量 n 分别取 40、56、64、128，根据算法 1、2 且适当的初始化便能得到 MPH-160、MPH-224、MPH-256、MPH-384、MPH-512. 相应压缩函数稠密度 ρ_3 的选取参见 3.2 节, 根据文献 [24] 中的安全性分析, 并结合 MQ 公钥密码体制 [8, 25] 的设计理念, 一般推荐取 $\rho_3>0.02$, 亦即相应差分函数的稠密度 $\rho_2>0.5$. 假如在安全性要求相对较低的领域如智能卡上使用, 压缩函数还可取得相对稀疏. 安全性与效率的关系参见表 1, 2.

另外, 为了更好地阐述本文新算法的各项性能, 附录中给出了一个实例, 限于篇幅, 我们以 MPH-160 为例, 其压缩函数稠密度仅为 0.02, 差分函数的稠密度为 0.52, 通过本文的安全性分析, 我们仍然认为它是安全的, 有兴趣的读者可以对其进行各项性能测试及分析. 上述实例仅为说明本文新 Hash 算法的相关性能, 实际应用中推荐 MPH-256 或输出长度值更大的 MPH 算法.

6 结论及进一步工作

本文提出了一种基于有限域上非线性方程组难解性问题构造 Hash 函数的方法, 与目前广泛使用的 Hash 算法如 SHA 系列相比具有下列特点: 安全性基于一个公认的数学难题, 碰撞分析等价于解有限域上的非线性方程组; Hash 函数的输出长度可变; 引入了整体随机性, 即从一族 Hash 函数中随机选择 Hash 函数; 设计自动化, 可根据用户的实际需求演化出满足其特定需求的多变量 Hash 函数. 通过本文的理论分析和实验结果表明, 新 Hash 算法在安全性和性能方面与其他 Hash 函数具有可比性, 易于软硬件实现, 且压缩函数的每个多项式及项数之间的独立性, 尤其易于并行计算; 另外, 由于本文 MPH 算法良好的随机性, 也可用于伪随机数生成. 其次, 由于本文算法的效率主要取决压缩函数的稀疏性, 对于 n 个变量的压缩函数, 共有 $\mathcal{O}(n^4)$ 项, 随着变量 n 的增加, 算法效率的变慢程度增快, 因此, 我们下一步的工作是: 在设计 Hash 值较长的 MPH 算法 (如 MPH-512) 时, 考虑选取较大的有限域, 如 GF(2^{16}) 或 GF(2^{32}), 从而降低压缩函数的规模, 同时设计有限域上元素乘法的高效算法, 以提高 MPH 算法的效率. 总之，我们相信经过进一步的研究完善, 它完全可能成为一种新的实用的 Hash 函数.

参考文献

1 Wang X Y, Yu H B. How to break MD5 and other hash functions. In: Proceedings of EUROCRYPT 2005. LNCS, Vol 3494. Berlin: Springer-Verlag, 2005. 19–35

2 Wang X Y, Yao A, Yao F. Finding collisions in the full SHA-1. In: Proceedings of CRYPTO 2005. Berlin: Springer-Verlag, 2005. 17–36

3 Wang X Y, Yu H B, Wang W, et al. Cryptanalysis on HMAC/NMAC-MD5 and MD5-MAC. In: Proceedings of EUROCRYPT 2009. Berlin: Springer-Verlag, 2009. 121–133

4 Yu H B, Wang X Y. Distinguishing attack on the secret-prefix MAC based on the 39-Step SHA-256. In: Proceedings of ACISP 2009. Berlin: Springer-Verlag, 2009. 185–201

5 Zhang H N, Li L, Wang X Y. Fast correlation attack on stream cipher ABC v3. Sci China Ser F-Inf Sci, 2008, 51: 936–947

6 Xue Y, Wu W L, Wang Z Y. Survey on the candidate algorithms of SHA-3 Hash function family. J Graduate School Chinese Acad Sci, 2009, 26: 577–586

7 Yuan Z, Wang W, Jia K T, et al. New birthday attacks on some MACs based on block ciphers. In: Proceedings of CRYPTO 2009. Berlin: Springer-Verlag, 2009. 209–230

8 Ding J T. Multivariate Public Key Cryptosystems. Berlin: Springer-Verlag, 2006. 11–190

9 Biham E, Dunkelman O. A framework for iterative hash functions-HAIFA. In: Proceedings of Second NIST Cryptographic Hash Workshop. 2006. http://csrc.nist.gov/groups/ST/hash/documents/DUNKELMAN_NIST3.pdf

10 Faugère J C. A new efficient algorithm for computing Grobner bases without reduction to zero(F5). In: Proceedings of ISSAC2002. ACM Press, 2002. 75–83

11 Shamir A, Patarin J, Courtois N, et al. Efficient algorithms for solving overdefined systems of multivariate polynomial equation. In: Proceedings of Eurocrypt2000. LNCS, Vol 1807. Berlin: Springer-Verlag, 2000. 392–407

12 Courtois N, Pieprzyk J. Cryptanalysis of Block ciphers with overdefined systems of equations. In: Proceedings of Asiacrypt 2002. LNCS, Vol 2501. Berlin: Springer-Verlag, 2002. 267–287

13 Bardet M, Faugère J C, Salvy B. On the complexity of Grobner basis computation of semi-regular overdetermined algebraic equations. In: Proceedings of ICPSS 2004. Berlin: Springer-Verlag, 2004. 71–74

14 Raddum H V, Semaev I. New technique for solving sparse equation systems. Cryptology ePrint Archive, 2006. http://eprint.iacr.org/2006/475.pdf

15 Tang X J, Feng Y. Applying dixon resultants in cryptography. J Softw, 2007, 18: 1738–1745

16 Ding J, Yang B Y. Multivariates polynomials for hashing. Cryptology ePrint Archive, 2007. http://eprint.iacr.org/2007/137.pdf
17 Lucks. A failure-friendly design principle for hash functions. In: Proceedings of ASIACRYPT2005. LNCS, Vol 3788. Berlin: Springer-Verlag, 2005. 474-494
18 Joux A. Multicollisions in iterated Hash functions. Application to cascaded constructions. In: Proceedings of Crypto 2004. LNCS Vol 3152. Berlin: Springer-Verlag, 2004. 306–316
19 Kelsey J, Schneier B. Second preimages on n-Bit Hash functions for much less than 2^n. In: Proceedings of EUROCRYPT 2005. LNCS, Vol 3494. Berlin: Springer-Verlag, 2005. 474–490
20 Kelsey J, Kohno T. Herding Hash functions and the nostradamus attack. In: Proceedings of EUROCRYPT2006. LNCS, Vol 4004. Berlin: Springer-Verlag, 2006. 183–200
21 Rukbin A, Soto J, Nechvatal J, et al. AStatistical test suit for random pseudorandom number generator for cryptographic applications. NIST Special Publication 800-22, 2001. http://csrc.nist.gov/rng/SP800-22b.pdf
22 Aumasson J, Meier W. Analysis of multivariate Hash functions. In: Proceedings of ICISC 2007. LNCS, Vol 4817. Berlin: Springer-Verlag, 2007. 309–323
23 Bettale L, Faugère J C, Perret L. Security analysis of multivariate polynomial for hashing. In: Proceedings of Inscrypt 2008. LNCS, Vol 347. Berlin: Springer-Verlag, 2009. 115–124
24 Wang H Z, Zhang H G, Guan H M, et al. A new perturbation algorithm and enhancing security of SFLASH signature scheme. Sci China Ser F-Inf Sci, 2010, 53: 760–768

附录 A　MPH-160 实例

三次项: 共 200 项, 每个值 $R \in [0, n(n+1)(n+2)/6)$, 其中 $n = 40$

25A3 2981 0555 0A05 0700 2599 1AB0 0E00 201F 1532 2C0A 0FC2 1AB0 0578 1F25 2AE0 1CBB 1E42 0253 1856 1214 2B9E 16CC 1DFC 09F4 0CD0 1258 1A08 2927 0328 238C 2300 1A60 2828 26B8 1D9C 0402 12A0 1234 21EE 094C 0732 2A85 2520 289E 26EA 1018 03B1 2BF6 1266 1F80 12BD 041F 12D0 1F95 0956 2B25 2BF1 1ECA 1004 1D48 1B93 13EF 0190 26E9 1230 2AE0 23C4 276A 1F95 04F6 0B50 0D7C 022F 2C06 1A98 1CB6 1689 1037 0F19 1D56 1D5A 146C 2B2A 03D8 03BD 1A4E 04B4 20B8 27B0 0428 09B8 23CD 05AC 11C3 24A8 0320 2A64 1919 2437 05CC 2495 06F1 1628 05EF 2BC0 0408 2B48 10C3 0771 1428 2858 244A 0715 21CF 0292 24B5 2255 2120 1A7C 0228 25CC 2AFD 2CAE 29F8 0444 10C2 2C92 0520 1432 1432 1029 2BC1 04F7 2CA7 17F0 1320 2494 0C04 106C 1554 0A7B 1BE4 1EDF 2258 171C 19AC 19EA 149A 15A0 260A 1EF1 2149 2774 109D 06CE 0EA0 0ECE 109A 0C9E 09C4 15EE 0D9D 1170 1F70 05C8 16A8 067C 159C 1B7C 074C 014B 24EA 0C0C 155E 1AB8 0508 1318 0547 0852 08F8 223B 242C 1E30 25A9 1B53 25C0 22D6 2229 2C14 17B0 0720 0D0A 1E78 0D20 0660 26D9 2872 0BF6 1EDC

二次项系数: 共 820 项, 每个值 $R \in [0, 255]$

D8 F0 0D BA 1A 6F 02 C8 68 5F E2 53 88 FF 46 9E 7F CD 01 41 58 3D 03 7D 20 B1 7D CD 35 A6 46 BD C1 2B 7E FF 81 98 71 BC CB 7C BD 28 CF D7 B8 59 1D 8A 0C 59 67 0B A4 7A 88 5D CC E9 C6 40 93 86 54 A5 74 F3 1B 64 36 EB BB 31 11 D6 CC CD 12 9A 69 79 C1 B3 EF B0 00 86 02 15 36 F2 D1 AB AD 29 9B 44 39 BE 73 3C 1B FF 40 66 24 84 07 47 1E 08 6D 82 67 78 77 94 DF 4A 95 C1 04 10 DF 4E 5C 4D 9E F1 14 87 0E 86 E6 9D 62 03 40 5B 09 AE A2 4C 30 8E 47 CF 88 1F 09 6D 49 4A 7F 6A 77 93 68 9A 64 95 4C 76 76 AB 60 6F 2A F1 AB 83 59 6B 67 0D BA 83 AA E1 A9 B9 45 98 26 96 EC 29 21 E1 98 BA F5 16 2A B5 32 14 51 1D 9F 16 AD 23 7F E4 35 F4 14 4B D6 D6 62 CA 5C 73 35 8E 96 74 65 A1 B5 52 59 5E F6 69 CA 28 81 4F C9 5B 90 63 04 83 D3 A9 45 CC 14 D5 3B BA 17 A5 7D 1A C4 74 CC 39 87 0D 98 53 F8 BD C6 4E B7 AC B1 A8 9A 6C 6F AF 0B D3 55 F0 AB 06 BD F1 3C D7 06 23 BD 51 DD 12 D6 90 BB DF 77 D7 AF F0 BD DC 5E E6 15 66 48 CF A3 1A 4D 9D E5 92 6F D8 93 B0 D9 8F CA 32 21 94 6A 86 CA E9 BD DF 0D 75 59 88 D8 FB 47 79

18 4D 64 28 34 BD 08 A1 D9 D5 E6 D9 FD 48 33 3F 1E 27 0B 95 CB 58 10 FE 66 45 54 58 28 D1 79 CB 65 90 16 A2 12 37 5D 5A 84 51 FB F9 7A 34 40 A2 5E 34 80 65 C8 16 B9 5B 44 C8 75 F3 55 4F F2 87 BA 00 81 33 EF F3 2C E6 F7 B5 9B B9 59 3D 39 80 69 21 92 9F C8 0A FF 20 2E 66 86 02 67 5C FC D3 9C 05 6C 81 D4 D9 BD 6D BA 67 8E C1 A1 4A 65 04 C5 58 09 EB D4 1C 2C 72 AD 87 4D 2C 67 E2 DD D7 95 07 A1 34 C8 D1 57 16 56 D1 9C B8 5A 42 0D 53 FC 3F AD F0 F3 34 87 7B 48 93 94 1A 5C C8 DE BD 2C 6E E6 F5 D4 C3 43 09 52 59 8D 47 8D 0F 79 98 96 40 45 57 2D 3A 57 63 87 F2 21 73 4E F6 47 AA EA A2 04 6C FF 97 CA 6F 36 69 28 15 41 98 F1 F9 19 C2 9B C7 8A 16 E5 52 F2 0C BC E0 46 53 60 C8 55 0B C3 40 51 35 32 70 8B 68 36 CB 7E C5 BD 9E 0F 2D 75 E9 13 B1 79 6F 12 49 2F 09 4A C9 70 3F F7 12 EA 19 D2 85 C3 33 D8 BE 7F 10 4C 7E 24 B0 FF E8 9C 65 CF F1 5A E2 6E 12 41 95 65 3F C1 37 58 1D 43 A1 8B 70 C7 E1 A5 D2 CB 8D B4 AB 6F 57 72 5D D8 E2 C7 BF D2 D4 8E CD B9 2D 9C 9E 99 D7 FE 96 94 7D 83 DC 84 A1 7B 0E E2 E9 BD 34 E7 BA EF F3 F0 08 01 04 27 6D FB E4 60 78 F9 CC 41 47 73 E3 A6 58 C2 B3 43 9C E0 D9 8C CD 70 01 D1 02 FE 12 AE 81 87 A6 A2 D4 3C BD FE 1F 83 B2 01 B0 3F 6E 41 D8 51 DB 4C F9 5F 9B 90 E1 D3 FB 78 56 27 22 D8 F2 61 8F 8B 32 D9 C2 5B C9 42 38 20 3A E8 9D 2C A5 37 3E E7 E1 7D BD B7 CC F1 08 23 DF F3 8B 37 06 95 A5 2A AE 5B 59 3E 1F 3F 47 E8 0C F8 DA 31 68 7C C2 5C 7C C4 A5 C9 37 CF 11 19 C5 E9 B6 93 63 2D D2 C6 71 49 EC 70 C8 81 C5 9F 4F F7 8C 16 C9 29 51 F2 F1

一次项系数: 共 40 项, $R \in [0, 255]$

BB BC 8E C9 77 FF A3 2F 92 09 B4 B1 30 FC A9 A2 2B E2 BA 6D 10 9C 8E 2A 6B 1A A4 6F 45 D2 F3 69 EA 6A CD 2B F0 B8 09 45

常数项: 74

随机因子: 共 19 个, 每个值 $R \in [0, n(n+1)(n+2)/6)$, 其中 $n = 40$, 用于生成压缩函数的后 19 个多项式.

2415 02E1 182B 0017 0A75 0337 2BE7 0057 0481 0295 15DF 000D 0781 038B 03F1 010F 1DB1 0049 01DD

注: 上述数据均由十六进制表示, 根据算法 2、3 可将其解码成压缩函数 $\mathrm{CF}(x)$, 如三次项第一个值 25A3, 根据算法 3 解码后对应于 $(\alpha^7+\alpha^6+\alpha^3+\alpha^2+\alpha+1)x_{18}x_{32}x_{37}$, α 是有限域 $\mathrm{GF}(2)[x]/(x^8+x^4+x^3+x+1)$ 的生成元.