•信息工程•

DOI:10.15961/j.jsuese.202101150

本刊网刊

基于纠错码的SM3改进算法

郑明辉[1,2], 乔译萱[1], 朱小强[1], 陈　珩[1]

(1.湖北民族大学 智能科学与工程学院, 湖北 恩施 445000; 2.四川大学 网络空间安全学院, 四川 成都 610065)

摘　要: 目前, 密码分析者已经可以在较短的时间内有效找到MD5、SHA1等国际密码杂凑算法的碰撞, 通过熵增来增强杂凑值的随机性是提高密码杂凑算法抗碰撞性的有效途径, 因此提出一种将纠错码和SM3算法迭代结构融合的改进方案。首先, 基于纠错码的线性性质和最小汉明距离最大化原则, 选择拟阵理论所构建的二进制线性分组码, 计算出其系统形式的生成矩阵, 并通过循环移位来消除比特之间的规律, 计算最终产生的有效码字; 其次, 在线性分组码中遵循周期性原则选取最优码字来构建初始常量值, 并将其赋值于初始寄存器中, 同时在迭代结构中引入初始寄存器构成算法的压缩函数, 完成杂凑算法迭代结构的二次构建; 最后, 考虑杂凑值信息熵对算法混乱度的评估能力, 将提出的方案与现有公开的国际密码杂凑算法进行对比实验, 同时进行雪崩效应、算法效率、内存损耗测试, 并进行综合评价。实验结果表明, 本文方案在不改变运算效率的前提下具有稳定的雪崩效应, 运行过程中的内存损耗比SM3算法降低0.01～0.07 MB, 同时杂凑值的信息熵值高于其他对比算法。提出的基于纠错码的改进方案能够通过熵增证明杂凑值比特之间的随机性更高, 更好实现隐藏明文和杂凑值之间统计信息的目的, 提高了密码杂凑算法的安全性。

关键词: 杂凑算法; 信息熵; 纠错码; 雪崩效应

中图分类号: TN918.1　　**文献标志码:** A　　**文章编号:** 2096-3246(2023)03-0235-08

Improved SM3 Algorithm Based on Error-correcting Code

ZHENG Minghui[1,2], *QIAO Yixuan*[1], *ZHU Xiaoqiang*[1], *CHEN Heng*[1]

(1.College of Intelligent Systems Sci. and Eng., Hubei Minzu Univ., Enshi 445000, China;
2.School of Cyber Sci. and Eng., Sichuan Univ., Chengdu 610065, China)

Abstract: Cryptanalysts now can effectively find the collisions of MD5, SHA1 and other international hash algorithms in a short time. Increasing entropy to enhance the randomness of hash value is an effective way to improve the anti-collision performance of hash algorithm. Therefore, an improved scheme combining the iterative structure of error-correcting code and SM3 algorithm was proposed. Firstly, based on the linear properties of error-correcting codes and the maximization principle of minimum Hamming distance, the binary linear block codes constructed by matroid theory were selected to calculate their systematic form of generation matrix, the rules between bits were eliminated by cyclic shift, and the effective code words were calculated. Secondly, in the linear block code, an optimal code word was selected to construct the initial constant value according to the periodicity principle, and its value was assigned to the initial register. At the same time, a compression function of the initial register formation algorithm was introduced into the iterative structure to complete the second construction of the iterative structure of the hash algorithm. Finally, considering the evaluation ability of hash value information entropy on chaos degree of the algorithm, the proposed scheme was compared with existing international hash algorithms, and avalanche effect, the algorithm efficiency and memory loss were tested and comprehensively evaluated. Experimental results showed that the proposed scheme has stable avalanche effects without changing the computational efficiency, the memory loss during operation is 0.01~0.07 MB lower than that of SM3 algorithm, and the information entropy of the hash value is

收稿日期:2021 – 11 – 19
基金项目:国家自然科学基金项目(61772181)
作者简介:郑明辉(1972—), 男, 教授, 博士生导师, 博士. 研究方向: 信息安全. E-mail: mhzheng3@163.com
网络出版时间:2022 – 07 – 26 10 : 46 : 39　　**网络出版地址:**https://kns.cnki.net/kcms/detail/51.1773.tb.20220725.0948.005.html

higher than that of other comparison algorithms. The improved scheme based on error correction code can prove that the randomness between hash bits is higher through entropy increase, which can better achieve the purpose of hiding statistical information between plaintext and hash, value, and improve the security of hash algorithm.

Key words: hash algorithm; information entropy; error-correcting code; avalanche effect

在实际的通信过程中，数据保密性和数据完整性是传输数据的基本安全需求。密码杂凑算法作为基础的密码算法之一，主要功能是提供数据的完整性检验，即数据经过信道传输和存储过程未被未授权方修改。密码杂凑算法的实质是将任意长度的消息序列映射成固定长度的输出值（也称为杂凑值）。并且无法从杂凑值反推出原本的消息序列，称为杂凑函数的单向性。基于该特性，杂凑值可以构造“数据指纹”来进行数据的完整性检验，应用于身份认证、密钥推导、消息认证码、区块链等场景。典型的密码杂凑算法包括MD5、SHA1、SHA2、SM3等。其中，SM3算法[1]是由国家商用密码管理办公室于2010年公布的商用密码标准，2012年成为行业标准，并于2016年成为国家标准，2018年正式成为ISO/IEC国际标准[2]。北京大学密码学研究组开发维护的密码算法工具包OpenSSL分支GmSSL支持SM3算法[3]，并在之后的正式版本中添加了SM3算法的实现[4]。

Merkle-Damgard（记作MD）结构[5−6]是密码杂凑算法的经典迭代结构，基于该结构所构造的杂凑函数，如果压缩函数具有抗碰撞性，则该函数也具有抗碰撞性。SM3算法作为MD结构的典型代表，将任意长度的数据输入压缩成256 bit的输出，能够有效抵御穷举攻击，同时采用消息双字介入、P置换等方法构造具有更高复杂性的轮函数，使得对SM3算法构造原像攻击是比较困难的[7]。安全性方面，目前针对MD5、SHA-1、RIPEMD、HAVAL等杂凑函数已找到快速碰撞的方法[8]，同时在碰撞攻击、区分器攻击、原像攻击方式下，对SM3 算法的攻击难度相比其他传统算法更高[9]，能够在具有高安全性需求的应用场景下进行数据传递。

然而，由于MD结构是串行结构，在效率上很难突破，同时易遭受消息扩展攻击及二次碰撞攻击[10]，所以研究人员开始将目光投向杂凑算法的迭代结构。徐劲松等[11]提出基于并行扩展算法的杂凑函数，提升了算法安全强度，但不适合短消息的处理。Yang等[12]针对杂凑函数并行迭代结构的局部碰撞问题提出基于混沌映射的压缩函数，增强了算法的抗碰撞能力和运算效率，但由于基于格的并行迭代结构，导致运算开销没有显著降低。Halder等[13]利用2D-CA技术构建杂凑函数的迭代结构，使算法的随机性和扩散性有所提升，但算法的轮函数定义为35轮，运算复杂性不高。Liu等[14]构造具有更高随机性的3D-ECM来充当海绵函数，同时输出指定长度的杂凑值，减轻了侧信道攻击的威胁，但由于构造过程中增加字符转换操作，无法保证运算效率的提升。Todorova等[15]提出基于Zaslavsky混沌映射的杂凑算法，该算法具有更强的抗碰撞性，但由于迭代过程中使用较多异或操作，导致算法复杂性不高。王镇道等[16]将MD5算法迭代过程的64步运算设计为32级的流水线，在保持串行运算的前提下提高了算法的运算速度，但未曾考虑算法安全性。巫光福等[17]以MD5算法为例构造基于线性分组码的密码杂凑算法，提高了密码杂凑算法输出值的随机性，但产生的杂凑值为128 bit，抗穷举攻击能力较弱。

综上所述，目前针对密码杂凑算法迭代结构的研究方法主要包含并行计算和混沌映射。其中：并行计算达到了提高数据运算效率，但与串行结构相比增大了计算复杂度；混沌映射则是通过提升初值敏感度来增强杂凑算法的抗碰撞性，但未考虑优化杂凑值的比特混乱度，杂凑值混乱度的提升也是算法安全性的重要评估条件之一。针对构建具有更高随机性的密码杂凑算法，本文提出基于纠错码的SM3改进算法的设计方案，选用纠错能力更强的线性分组码并计算对应的生成矩阵；在生成的有效码字中选择8个最优码字串联赋值给初始寄存器，同时与每个512 bit消息分组进行迭代压缩运算，所产生的杂凑值为256 bit，若采用蛮力攻击，则需要执行2^{256}数量级的操作，保证算法的安全性。实验结果表明，本文算法满足雪崩效应，并在运算效率相近的情况下，产生的杂凑值随机性更高，同时算法内存消耗更少。

1 准备知识

在数字通信过程中不可避免发生差错，对于接收到的数据序列，纠错码的主要作用是在存储设备及通信中进行纠错和检错，被广泛用于密码学和通信系统中。纠错码主要分为分组码和卷积码。下面重点介绍分组码[18]及其相关知识[19−21]。

定义1（线性分组码） 有限域GF_q上的一个(n,k)线性分组码C是GF_q^n上的k维线性子空间，其中，n为分组码的码长，码C中向量称作码字，k为信息码元长度，k/n为码的信息率，$n-k$为码C的校验位或者监督位。

定义2（汉明距离） 两个不同码字之间的汉明

距离定义为两个序列之间对应不同的位数，记作d。

设在二元线性码中给定两个码字$\boldsymbol{c}^{(1)}$、$\boldsymbol{c}^{(2)}$，其中，$\boldsymbol{c}^{(1)}=\{c_1^{(1)},c_2^{(1)},\cdots,c_n^{(1)}\}$，$\boldsymbol{c}^{(2)}=\{c_1^{(2)},c_2^{(2)},\cdots,c_n^{(2)}\}$，则$\boldsymbol{c}^{(1)}$和$\boldsymbol{c}^{(2)}$的汉明距离为：

$$d(\boldsymbol{c}^{(1)},\boldsymbol{c}^{(2)})=\sum_{i=1}^{n}(c_i^{(1)}\oplus c_i^{(2)}) \tag{1}$$

式中，$\oplus$为异或运算。

码字间的距离表示码字之间差异程度的大小。当存在干扰时，距离越大，一个码字变成另一个码字的可能性越小。

定义3（最小汉明距离）　码C的最小汉明距离$d_{\min}$定义为所有任意两个码字汉明距离的最小值。对于线性分组码，记为$(n,k,d_{\min})$。

定义4（生成矩阵）　对于(n,k,d)线性分组码，有下列关系式：

$$C=\boldsymbol{m}\boldsymbol{G} \tag{2}$$

式中：$\boldsymbol{m}$为k维矢量；$k\times n$的矩阵$\boldsymbol{G}$称为线性分组码C的生成矩阵，$\boldsymbol{G}$的行向量构成码C的一组基。

对于给定信息码元长度k、码长n的线性分组码C，其生成矩阵不唯一，并且可以通过初等行变换相互进行转换。其中，无论生成矩阵如何初等变换，码字都是唯一确定的，任意生成矩阵可产生相同的2^k个码字。

2　基于纠错码的SM3算法构造

SM3算法是基于分组迭代的国际密码杂凑算法，该算法比其他国际杂凑算法标准设计更复杂，具体表现在Merkle Damgard迭代结构中每一轮压缩都使用2个消息字，以及消息拓展过程中每一轮都使用5个消息字，并且将不同群运算结合，使明文消息非线性迅速扩散。下面给出基于纠错码的SM3改进算法具体构造过程。

单个消息分组处理过程主要利用纠错码构建初始常量并将其嵌入到初始缓存器中，再进行64步迭代操作。具体分为初始常量构造、消息预处理、消息扩展、迭代压缩、杂凑值输出5个步骤。其中，输入是最大长度为$2^{64}-1$ bit的消息，输出是长度为256 bit的消息杂凑值，处理单元是512 bit消息分组。

2.1　初始常量构造

在构造基于纠错码的SM3算法过程中，需要选择合适的线性分组码C，一个最优$(n,k,d_{\min})$分组码C必须满足以下条件。

1）确定n和k，使$d_{\min}$尽量最大化，则构造出的码C可以提高纠错能力。

2）确定n和$d_{\min}$，使k尽量最大化，则构造出的码C可以提高传输速率。

3）确定k和$d_{\min}$，使n尽量最小化，则构造出的码C可以提高传输速率。

综上所述，构造性能良好的纠错码，需要考虑信息k、码长n、最小汉明距离$d_{\min}$这3个参数的相互制约问题，达到传输效率及纠错能力的平衡。

根据可变拟阵搜索算法和拟阵联接度的定义[22]，可以构建一类最优的二进制线性$(n,k,d_{\min})$码及它的生成矩阵$\boldsymbol{G}_{a\times b}$[23]。因为本文所构建的加法常数表需8个32 bit串联而成，所以选择构建SM3的线性分组码为(32,6,16)，并求得该线性分组码的生成矩阵$\boldsymbol{G}_{6\times32}$。为了使加法常数能达到随机化和无规律性最大化的效果，需要有效降低比特之间的规律性，经过测试，选择将生成矩阵$\boldsymbol{G}_{6\times32}$进行循环左移6位，最终产生的杂凑值的熵值达到预期设计要求，即杂凑算法的随机性更高。生成码字集合$U=\{\boldsymbol{u}_1,\boldsymbol{u}_2,\cdots,\boldsymbol{u}_{2^k}\}$，$k$=6。

基于SM3算法的初始常量及迭代过程的特征，应选择8个码字串联来构建本文算法的初始常量，其中初始常量集合B^0以下列方式选取：

$$B^0=\{\boldsymbol{u}_8,\boldsymbol{u}_{16},\cdots,\boldsymbol{u}_{8i}\},1\le i\le 8 \tag{3}$$

重新构造的初始常量应满足两个要求：

1）初始常量二进制表示中，1、0的数量比趋近于1。

2）初始常量二进制表示中，最长1游程小于10，最长0游程小于8。

2.2　消息预处理

假设输入消息m的长度为l bit，首先，在消息的末尾先添加比特“1”；再在后面添加k个“0”，k满足$l+1+k\equiv 488 \bmod 512$；再添加64 bit的比特串来表示输入消息的长度$l$，得到填充后的消息$m'$，长度为512×$n$ bit；最后，将消息m'按512 bit进行分组：

$$m'=m'_0m'_1\cdots m'_{n-1} \tag{4}$$

式中，$n=(l+k+65)/512$。

2.3　消息扩展

将第i个消息分组m'_i扩展生成132个字$W_0,W_1,\cdots,W_{67},W'_0,W'_1,\cdots,W'_{63}$，具体操作如下：

1）将m'_i分为16个32 bit的比特串$W_0,W_1,\cdots,W_{15}$。

2）$W_{16},W_{17},\cdots,W_{67}$以下列规则进行扩展：

$$\begin{aligned}W_j=&P_1(W_{j-16}\oplus W_{j-9}\oplus(W_{j-3}<<<15))\oplus\\&(W_{j-13}<<<7)\oplus W_{j-6},16\le j\le 67\end{aligned} \tag{5}$$

式中，W_j为消息扩展的第j个字，$<<<k$表示循环左移k比特运算，固定公式$P_1(\cdot)$定义为：

$$P_1(X)=X\oplus(X<<<15)\oplus(X<<<23) \tag{6}$$

3）$W_{16},W_{17},\cdots,W_{67}$以下列规则进行扩展：

$$W'_j=W_j\oplus W_{j+4},0\le j\le 63 \tag{7}$$

式中，W'_j为消息扩展的第$j+69$个字。

2.4 迭代压缩

将初始常量集合B^0中8个码字分别赋值于寄存器A、B、C、D、E、F、G、H中，对第i个消息分组m_i'以下列方式迭代：

$$B^{i+1}=CF(B^i,m_i'),0\le i\le n-1 \tag{8}$$

式中，CF压缩函数由64步迭代运算组成，B^i为第i次迭代输入的集合。将B^i赋值于A、B、C、D、E、F、G、H作为初始寄存器，同时添加中间变量$SS1$、$SS2$、$TT1$、$TT2$进行左向赋值操作，具体过程描述如式(9)～(12)：

$$SS1\leftarrow((A<<<12)+E+(T_j<<<j))<<<7 \tag{9}$$

$$SS2\leftarrow SS1\oplus A<<<12 \tag{10}$$

$$TT1\leftarrow FF_j(A,B,C)+D+SS2+W_j' \tag{11}$$

$$TT2\leftarrow GG_j(E,F,G)+H+SS1+W_j \tag{12}$$

式中，←表示左向赋值运算符，T_j为32 bit常量，$FF_j(\cdot)$、$GG_j(\cdot)$为定义好的布尔函数[1]，$0\le j\le 63$。

将更新后的中间变量$TT1$、$TT2$与寄存器A、B、C、D、E、F、G、H进行状态更新，过程描述如下：

1)将初始寄存器A、C、E、G分别赋值于寄存器B、D、F、H，然后将中间变量$TT1$赋值于寄存器A。

2)对中间变量$TT2$进行公式运算后赋值于E，具体运算过程如下：

$$E\leftarrow P_0(TT2) \tag{13}$$

式中，固定公式$P_0(\cdot)$定义为：

$$P_0(X)=X\oplus(X<<<9)\oplus(X<<<17) \tag{14}$$

3)将更新后的寄存器B、F进行循环左移操作后赋值于寄存器C、G，描述如下：

$$C\leftarrow B<<<9 \tag{15}$$

$$G\leftarrow F<<<19 \tag{16}$$

每轮迭代的输入都是上一轮迭代的输出再与512 bit的分组消息进行运算的结果。

2.5 输出杂凑值

最终所有消息分组处理完毕之后，最后一个512 bit的输出即为算法杂凑值。

密码杂凑算法的安全水平是由它的输出长度决定的[24]，本文所构造的杂凑函数输出长度为512 bit，与128 bit的输出相比更能抵抗原像攻击、第二原像攻击和碰撞攻击。

3 实验分析与对比

第2节所构造的密码杂凑函数是针对SM3算法的改进算法，下面从雪崩效应、信息熵方面进行本文算法的安全性分析，同时通过仿真实验进行算法运算效率和内存损耗性能分析与讨论。

3.1 安全性分析

3.1.1 雪崩效应分析

密码学中约定密码杂凑算法应满足雪崩效应，即输入消息微小的改变会引起杂凑值至少一半以上的位数发生变化，以达到更好的混淆效果，利用式(17)～(19)对本文改进算法的雪崩效应进行稳定性评估。

$$\overline{B}=\frac{1}{N}\sum_{i=1}^{N}B_i \tag{17}$$

$$P=\frac{\overline{B}}{n}\times100\% \tag{18}$$

$$\Delta P=\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{B_i}{n}-P\right)^2}\times100\% \tag{19}$$

式(17)～(19)中，B_i为第i次测试的杂凑值改变的比特数，P为平均雪崩系数，N为测试的总次数，n为杂凑值比特数，$\overline{B}$为平均变化比特数。理想状态下P为50%，表明密码杂凑算法具有良好的雪崩效应，均方差ΔP数值越小，则密码杂凑算法的稳定性越好[25]。

为了进一步验证改进算法性能，选择传统SHA-256算法和SM3算法，以及改进的MD5算法[17](记为Wu-MD5算法)进行雪崩效应测试。随机选择明文消息并计算生成的杂凑值，任意改变消息中的1 bit，同时计算新生成的杂凑值。由于杂凑值长度不同，所以仅针对雪崩系数P及均方差ΔP进行数值比较，结果如表1所示。值得说明的是，雪崩效应仅是杂凑算法扩散效应的指标之一，其结果无法直观地进行4种算法混淆性的优劣比较，仅能够进行雪崩效应的稳定性评估。

表 1 不同测试次数下4种算法的雪崩特性统计

Tab. 1 Avalanche characteristics statistics of four algorithms under different test times

算法	1 000次		5 000次		10 000次		50 000 次	
	P/%	ΔP/%	P/%	ΔP/%	P/%	ΔP/%	P/%	ΔP/%
SM3算法	50.01	3.11	50.11	3.19	50.02	3.10	50.02	3.13
SHA-256算法	49.94	3.13	50.07	3.16	49.99	3.18	50.04	3.14
Wu-MD5算法	50.15	4.40	50.15	4.24	49.97	4.41	50.02	4.41
本文算法	49.98	3.05	49.95	3.14	50.01	3.11	50.01	3.11

表1结果表明：在测试总次数N分别为1 000、5 000、10 000和50 000的情况下，本文算法与其他3种算法的雪崩系数P均接近50%，达到了杂凑函数雪崩效应的理想状态，说明本文算法拥有良好的混淆和扩散性；同时，本文算法的均方差ΔP的数值偏小，充分说明本文算法具有稳定的雪崩效应。

3.1.2　信息熵分析

熵反映了信息源的平均不确定性，在密码学领域内，信息熵也是用于衡量信息序列随机性的一项重要指标。信息序列的随机性越大，熵值越大；信息序列的随机性越小，熵值越小[26]。一般熵值的大小也与攻击者分析杂凑算法的规律性所需要的时间成正比。利用熵值的大小来度量纠错码构建的改进密码杂凑算法的安全性。熵的计算公式为：

$$H(x)=\sum_{i=1}^{n}p(x_i)\mathrm{lb}\ p(x_i) \qquad (20)$$

式中，$H(x)$为消息x的信息熵，$p(x_i)$为消息中x_i出现的概率。

针对第2节中生成矩阵$\boldsymbol{G}_{6\times32}$分别进行循环左移5、6、7位操作，输入长度为20～500 byte的样本数据，计算在输入数据相同时的杂凑值信息熵，如图1所示。由图1可以看出，循环左移6位时信息熵数值更高，相对于循环左移5、7位稳定性更好，而循环左移操作的目的是增大比特之间的无规律性，说明构造的加法常数值符合算法设计最初指标，即达到随机性和无规律性最大化，提升了杂凑值随机性。

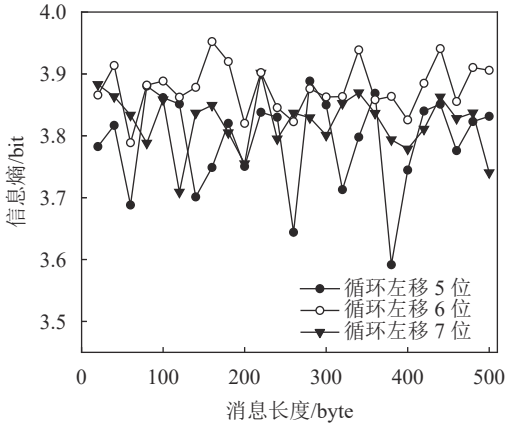

图 1　本文算法中不同循环左移位数的信息熵比较

Fig. 1　Information entropy comparison of different cyclic left shift numbers in the proposed algorithm

为了评估本文算法的杂凑值随机性，在明文样本相同的情况下，选择典型的密码杂凑算法进行杂凑值的信息熵对比实验，具体标准为：1）本文算法是基于SM3算法的改进杂凑算法，所以选择SM3算法进行对比；2）在杂凑值长度都为256 bit的情况下，选择SHA-256算法进行对比。不同算法的杂凑值信息熵对比实验结果如表2所示。

表 2　本文算法、SM3算法及SHA-256算法的杂凑值信息熵比较

Tab. 2　Comparison of information entropy between the proposed algorithm, SM3 algorithm and SHA-256 algorithm

消息输入	算法	杂凑值	信息熵/bit
qyx	SHA-256	07226DDCAA020C4B5862B37BEE3A5E6177F577CE62E44FE6B10F7C82723A149B	3.851 378 73
	SM3	949349E7BAA8D077B8CE799CFCA9409EA77FF42B5661F43A06128674DEFBBED5	3.868 541 05
	本文算法	4E86BF0FF80E3F32C30ABBCC2BEF18DF5FA55D3D90733C60157742E4A7CDA709	3.902 148 56
return	SHA-256	7187F0675EB3827939741ACF7342BA78836ECEC21A31ECF3F34A55309D3BEE8A	3.831 061 05
	SM3	C8086AED8C2AE28AE1933D053C9DBED12A80DDD1EB3C58F08197AACBB777E702	3.750 168 85
	本文算法	AB70B060AA594FA8DD67E0D7EDE562A19F5F96467C106A36A20F8C8AE5374998	3.855 769 21
message	SHA-256	AB530A13E45914982B79F9B7E3FBA994CFD1F3FB22F71CEA1AFBF02B460C6D1D	3.840 868 61
	SM3	1756AC517F85FFDA751DCDEBF3C89575272FC56904F9BAAD983EC44C36FEAC7B	3.851 338 50
	本文算法	0D4ABC3516F0D37306BCFAB3CEC169D3A68B8908D41F7224D02408A977948110	3.886 245 77
security	SHA-256	5D2D3CEB7ABE552344276D47D36A8175B7AEB250A9BF0BF00E850CD23ECF2E43	3.842 242 32
	SM3	3A4818F8F42A5764C86A8BA0E17765C6037EA086DC9393E1500C4CDC56E783B2	3.875 414 50
	本文算法	84284E2AB42D28B785C1EB7C61650804A9016BA8432FD9CC5791370F8A3D3A14	3.920 046 20
123456789	SHA-256	15E2B0D3C33891EBB0F1EF609EC419420C20E320CE94C65FBC8C3312448EB225	3.669 052 73
	SM3	C7AE0AEC3D2F9BEB84DC1885AA7A576BAA7A07B38060AFC64C5600F93A5456B5	3.783 981 12
	本文算法	9379307AB36EA8F6B1B44A9BE6AD74992F8806618565021D391D600B03C8D6F8	3.841 762 55
hashfunction	SHA-256	031B551581A47C95C3D48AA97C7ED953B517AFDCCDC69F818A31BCF8122D24ED	3.833 066 92
	SM3	32DAD4C10CA66CD876BBAD5AB64C0F6CF5F824967EC7C3015697BA6955B80831	3.869 276 37
	本文算法	F19119067D5A48CA8B3508453DB8C3027D0CEF212E860B6DDD11E3BFDAAC4A85	3.909 536 08
errorcorrectingcode	SHA-256	333649280D0F13862D76933A111FD5B543E8B9A516A9B693A593E661DE477441	3.748 536 92
	SM3	8223A465B592304D7F586D3DB505F3E5E55B95839919C10874D4F57DF4BEFF32	3.789 960 63
	本文算法	1F56EB0ADEE878B6AEBFEB80A1C5A8F8C832434B74F4187292D4BE9F7E30F5BB	3.838 385 64

表2表明：在迭代结构相似的情况下，本文算法因为选择纠错更强的最优码(n,k,d_{min})来构建加法常数表，杂凑值的熵值相对于 SM3算法有所增加；在杂凑值长度相同的情况下，由于本文算法迭代结构包含消息双字介入等方法，使得轮函数复杂性提升，消息能够快速扩散，从而导致最终生成杂凑值的信息熵值更高，随机性更强。以上结果表明，本文所构造的SM3改进算法利用纠错码技术有效平衡了杂凑值长度和迭代结构两个方面，使得算法杂凑值随机性有所提升，能够更好地隐藏明文消息和杂凑值之间的关联性。

3.2 性能分析

一般来说，密码杂凑算法的运算效率及内存损耗是需要研究人员考虑的重要方面，在配置为Intel Core i5–9400 2.90 GHz、16 GB RAM的计算机上，进行改进算法与传统SM3算法的时间效率及内存损耗的对比实验，并进行分析。

3.2.1 运算效率分析

利用 Java 1.8.0_291进行算法运算效率测试。针对不同明文消息长度的输入，分别选择本文算法与SM3算法进行效率分析，如图2所示。

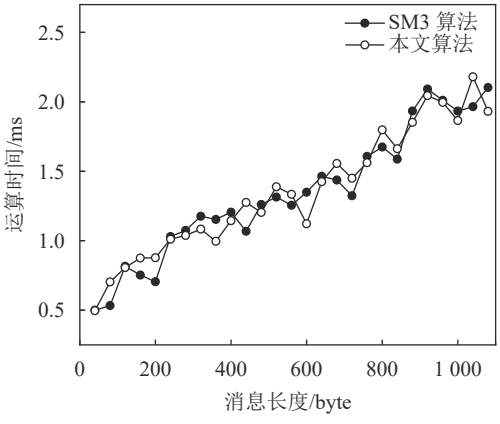

图 2 本文算法与SM3算法时间复杂度比较

Fig. 2 Comparison of time complexity between the proposed algorithm and SM3 algorithm

由于迭代结构的串行特性，明文消息长度和运行时间成正比，图2结果表明，本文算法可以支持算法快速运算，在输入明文消息长度为40～1 080 byte的条件下，1 s内可以进行450～2 000次运算，与SM3算法的运算效率基本一致。

3.2.2 内存消耗分析

JProfiler作为商业授权的Java性能剖析工具，具有对被分析对象影响较小、针对内存（memory）分析功能强大等特点，专用于分析Java SE、Java EE应用程序。利用JProfiler分析工具，针对不同长度的明文输入，选用30个明文样本集，对本文改进算法及SM3算法分别进行相同明文输入的内存损耗测试对比，结果如图3所示。

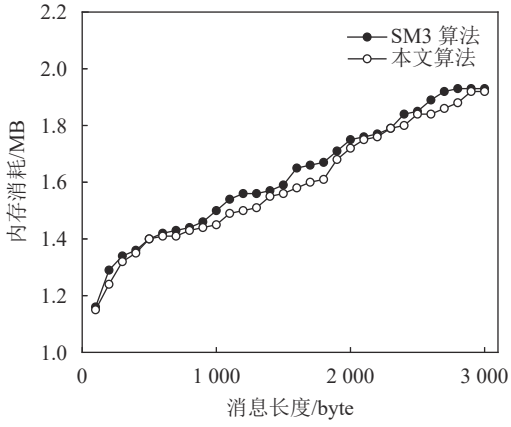

图 3 本文算法与SM3算法内存损耗比较

Fig. 3 Comparison of memory loss between the proposed algorithm and SM3 algorithm

由图3可知：本文算法的内存损耗并不高，在明文输入长度为100～3 000 byte的条件下，随着消息长度的增加，计算量增大，导致内存损耗呈线性增长趋势；同时，由于本文利用纠错码来构造加法常数表，导致在消息长度相同的情况下，本文算法相比于SM3算法内存损耗降低0.01～0.07 MB，实现了性能优化。

基于纠错码的SM3算法是在传统SM3算法的基础上进行的改进。改进后的SM3算法不仅具有较强的雪崩效应，且具有更强的随机性，使攻击者更难找到其中规律。上述算法效率及内存损耗实验表明，本文改进算法在效率相同的情况下内存损耗有所下降，为高效率、低内存需求的应用场景提供技术参考。

4 结 论

为了提高杂凑值的随机性，本文利用纠错码可以在数字通信过程中提高可靠度的特性，提出一种对SM3算法的改进方案。该方案选择拟阵理论构建的线性分组码(32,6,16)，通过生成矩阵$\boldsymbol{G}_{6\times32}$计算有效码字，同时利用周期性原则选择8个码字构建初始常量值，并嵌入到迭代结构中进行64轮运算；从信息熵值和雪崩效应两个角度进行杂凑值稳定性和随机性评估，同时测试本文算法在运算效率和内存损耗的全局性能并进行分析。实验结果表明，本文算法具有理想雪崩效应的特性，生成的杂凑值相比其他密码杂凑算法混乱度有明显提高，使攻击者更难逆推出明文消息，具备更高的算法安全性。另外，本文算法在保留传统密码杂凑算法的串行迭代结构的前提下，能够在1 s内进行450～2 000次高速运算，且内存损耗与SM3算法相比降低0.01～0.07 MB，为信息安全领

域的应用开发提供理论参考。在未来的研究工作中，将重点研究密码杂凑算法的迭代结构，尝试采用并行结构进行数据运算效率的优化，并且在区块链技术的底层架构来验证优化后算法的适用性。

参考文献：

[1] Wang Xiaoyun,Yu Hongbo.SM3 cryptographic hash algorithm[J].Journal of Information Security Research,2016,2(11):983–994.[王小云,于红波.SM3密码杂凑算法[J].信息安全研究,2016,2(11):983–994.]

[2] 国家密码管理局.我国SM2/3/9密码算法正式成为ISO/IEC国际标准[EB/OL].(2018–11–22)[2021–07–22].https://www.oscca.gov.cn/sca/xwdt/2018-11/22/content_103 9430.shtml.

[3] Guan Z.GmSSL[CP/OL].(2018–10–25)[2021–07–22].https://github.com/guanzhi/GmSSL/.

[4] OpenSSL.Changelog[EB/OL].(2018–09–11)[2021–07–22].https://www.openssl.org/news/changlog.html#x3.

[5] Merkle R C.One way hash functions and DES[M]//Advances in Cryptology—CRYPTO'89 Proceedings.New York:Springer,1990:428–446.

[6] Damgård I B.A design principle for hash functions[M]//Advances in Cryptology—CRYPTO'89 Proceedings.New York:Springer,1990:416–427.

[7] Zou Jian,Dong Le.Improved preimage and pseudo-collision attacks on SM3 hash function[J].Journal on Communications,2018,39(1):46–55.[邹剑,董乐.对缩减轮数SM3散列函数改进的原像与伪碰撞攻击[J].通信学报,2018,39(1):46–55.]

[8] Wang Xiaoyun,Yu Hongbo.How to break MD5 and other hash functions[C]//Advances in Cryptology—EUROCRYPT 2005.Berlin:Springer,2005:19–35.

[9] Yao Jian.Domestic commercial cryptographic algorithm and its performance analysis[J].Computer Applications and Software,2019,36(6):327–333.[姚键.国产商用密码算法研究及性能分析[J].计算机应用与软件,2019,36(6):327–333.]

[10] Yang Yijun,Chen Fei,Zhang Xiaomei,et al.Research on the hash function structures and its application[J].Wireless Personal Communications,2017,94(4):2969–2985.

[11] Xu Jinsong,Zhang Minxuan,Chen Shiwei,et al.Parallel algorithm for extending Merkle-Damgard Hash construction[J].Journal of National University of Defense Technology,2017,39(6):59–63.[徐劲松,张民选,陈士伟,等.Merkle-Damgard Hash结构并行扩展算法[J].国防科技大学学报,2017,39(6):59–63.]

[12] Yang Yijun,Chen Fei,Sun Zhiwei,et al.Secure and efficient parallel hash function construction and its application on cloud audit[J].Soft Computing,2019,23(18):8907–8925.

[13] Haldar T,Chowdhury D R.Design of hash function using two dimensional cellular automata[M]//Proceedings of the Fifth International Conference on Mathematics and Computing.Singapore:Springer,2021:33–45.

[14] Liu Hongjun,Wang Xingyuan,Kadir A.Constructing chaos-based hash function via parallel impulse perturbation[J].Soft Computing,2021,25(16):11077–11086.

[15] Todorova M,Stoyanov B.Novel hash function using Zaslavsky map[J].AIP Conference Proceedings,2021,2333(1):070005.

[16] Wang Zhendao,Li Ni.An optimized MD5 algorithm and hardware implementation[J].Journal of Hunan University (Natural Sciences),2022,49(2):106–110.[王镇道,李妮.一种优化的MD5算法与硬件实现[J].湖南大学学报(自然科学版),2022,49(2):106–110.]

[17] Wu Guangfu,Zeng Xianwen,Liu Juan,et al.Design and analysis of Hash function based on error correcting code[J].Netinfo Security,2018(1):67–72.[巫光福,曾宪文,刘娟,等.基于纠错码的Hash函数的设计与分析[J].信息网络安全,2018(1):67–72.]

[18] Zhang Lingyu,Chen Deyuan.The large capacity embedding algorithm for H.264/AVC intra-prediction mode video steganography based on linear block code over Z4[J].Multimedia Tools and Applications,2020,79(17/18):12659–12677.

[19] Zhang Zhuoran,Zhang Huang,Zhang Fangguo.Survey on applications of list decoding to cryptography[J].Journal of Electronics & Information Technology,2020,42(5):1049–1060.[张卓然,张煌,张方国.列表译码在密码中的应用综述[J].电子与信息学报,2020,42(5):1049–1060.]

[20] Chara M,Podestá R,Toledano R.Block transitive codes attaining the Tsfasman–Vladut–Zink bound[J].Designs,Codes and Cryptography,2020,88(6):1227–1253.

[21] Prasad S,Pal A K.Hamming code and logistic-map based pixel-level active forgery detection scheme using fragile watermarking[J].Multimedia Tools and Applications,2020,79(29/30):20897–20928.

[22] Zhang Shuiping,Lin Pingping,Wu Guangfu,et al.Construct the systematic binary quasi-cyclic codes with rate $1/p$ based on variable matroid search algorithm[J].Journal of Electronics & Information Technology,2016,38(11):2916–2921.[张

水平,林平平,巫光福,等.基于可变拟阵搜索算法构造码率为1/*p*的二进制系统准循环码[J].电子与信息学报,2016,38(11):2916–2921.]

[23] Zhang Shuiping,Lin Pingping,Wang Keke,et al.Construction of binary shift dual codes[J].Journal of Jiangxi University of Science and Technology,2016,37(1):74–79.[张水平,林平平,王柯柯,等.二进制移位对偶码的构造[J].江西理工大学学报,2016,37(1):74–79.]

[24] Wei Hongru,Huang Jingyi.SOTS:A hash function-based shorter post-quantum digital signature scheme[J].Journal of Computer Research and Development,2021,58(10):2300–2309.[卫宏儒,黄靖怡.SOTS:一个基于哈希函数更短的后量子数字签名方案[J].计算机研究与发展,2021,58(10):2300–2309.]

[25] He Wenqi,Chen Jiayu,Zhang Lianbin,et al.Optical Hash function based on multiple scattering media[J].Acta Physica Sinica,2021,70(5):054203.[何文奇,陈嘉誉,张莲彬,等.一种基于多重散射的光学Hash函数[J].物理学报,2021,70(5):054203.]

[26] Zhang Lei,Chen Chuan,Tan Qiyun,et al.An image encryption algorithm combining S-box and chaotic mapping[J].Journal of Beijing University of Posts and Telecommunications,2021,44(6):40–47.[张雷,陈川,谭淇匀,等.结合S盒与混沌映射的图像加密算法[J].北京邮电大学学报,2021,44(6):40–47.]

（编辑　赵　婧）

引用格式:Zheng Minghui,Qiao Yixuan,Zhu Xiaoqiang,et al.Improved SM3 algorithm based on error-correcting code[J].Advanced Engineering Sciences,2023,55(3):235–242.[郑明辉,乔译萱,朱小强,等.基于纠错码的SM3改进算法[J].工程科学与技术,2023,55(3):235–242.]