具有可变散列输出的安全 Hash 算法*)

白恩健　王井刚　肖国镇

(西安电子科技大学综合业务网国家重点实验室　西安710071)

摘　要　本文给出了一个具有可变散列输出的安全 Hash 算法，该算法可以把任意长的消息散列成128、160、192、224或256比特的消息摘要。算法的安全性与有效性都能满足实际的需求；设计简单，没有大的数据结构和复杂程序；运行速度快，适用于高速软硬件实现，特别适用于32-位的计算机处理；安全性方面比 SHA 算法更具优势，算法不仅能抗所有已知攻击，而且还提供了充分大的安全空间来抗将来的未知攻击。文章给出了算法的具体描述，设计诱因和安全性分析，并且将算法与其它同类算法进行比较。

关键词　摘要，Hash 算法，SHA，生日攻击

A One-Way Hashing Algorithm with Variable Length of Output

BAI En-Jian　WANG Jing-Gang　XIAO Guo-Zhen

(P. O. Box 119 Key Lab. On ISN, Xidian Univ., Xi'an 710071)

Abstract　A One-way hashing algorithm with variable length of output is presented in the paper. The algorithm compresses a message of arbitrary length into a digest of 128, 160, 192, 224, 256 bits. The advantages of the algorithm are simple in design, fast in speed, very efficient and particularly suited for 32-bit computer which predominates the current workstation market. The algorithm is prior to SHA in secure, it can thwart not only the known attacks, but also future unknown attacks. The specifications, design rationale and security analysis are given in the paper. In addition, the compared results to other algorithms are listed at the end of the paper.

Keywords　Digest, Hash algorithm, SHA, Birthday attack

1　引言

单向散列算法 H 是将一任意长度的消息压缩成固定长度的散列值，该散列值被称为消息摘要。单向散列算法有一个非常重要的性质被称为抗碰撞(collision-resistance) 条件，即找到两个随机消息 M 和 M′满足 H(M)＝H(M′)在计算上是不可行的。单向散列算法在随机数产生，公钥算法，认证特别是数字签名等方面有广泛的应用。自从公钥密码体制诞生以来，安全 Hash 算法的研究取得了很大的进展，理论方面的成果见文[1]。最近的工作主要是集中在算法的设计方面，比较著名的算法象 MD 系列(包括 MD2, MD4, MD5)，SHA 算法，FFT 算法等[2]，所有这些算法的输出长度都是固定的，其中 MD 系列算法和 FFT 算法的散列值为128比特，SHA 算法的散列值为160比特。Yuliang Zheng 等人提出了一个具有可变散列输出的算法 HAVAL[3]，该算法能够满足不同的安全性需求。

本文借鉴了 MD 系列算法和 HAVAL 算法的思想，设计了一个新的具有可变散列输出的安全 Hash 算法。仿真结果表明算法在处理小数据方面与 SHA 算法具有相当的速度，处理大型数据时稍慢于 SHA 算法，但仍能满足实际的需求。算法的突出优点是不仅能够抵抗所有已知的攻击，而且在抗将来的未知攻击方面也存在一定的优势。在这里，我们给出算法的设计目标：

(1)安全性。找到两个具有相同散列值的消息在计算上不可行，不存在比穷举攻击更有效的攻击方法。

(2)直接安全性。算法的安全性不基于任何困难问题假设。

(3)速度。算法要适用于高速软件实现，适合于32-位计算机处理。

(4)简单性。算法尽可能简单，没有大的数据结构和复杂程序。

2　算法描述

首先给出算法的一般性描述，然后给出算法的详细描述。

算法以512-位的分组作为处理输入文本，每一分组又划分为16个32-位字。设 m 是待散列的消息，首先进行消息填充，填充后的消息长度为512的倍数。然后对每一个512-位的分组进行消息扩展，把消息分组从16个32-位字扩展成80个32-位字。算法有8个32-位初始变量，分别记为 A, B, C, D, E, F, G, H，这些变量和其它常数都来自于自然对数底 e。接着进行算法的主循环，循环的次数是消息中512-位分组的数目。主循环有四轮，每轮20次操作，同一轮中使用相同的轮常数。每次操作对上述8个变量进行一次线性变换和上述8个变量中的6个进行一次非线性运算。相邻的两轮中间插入一个 FL 函数，该函数作用于上一轮的输出，并把结果输入到下一轮。所有的80次操作完成之后，再用下一数据分组继续运行算法，最

*)基金项目：国家"十五"国防预研基金项目 (41001040102)；国家重点基础研究发展规划973资助项目 (G1999035804)。白恩健　博士生，主要研究兴趣为网络安全与信息保密。王井刚　硕士生，主要研究兴趣为分组密码与流密码。肖国镇　教授，博士生导师，研究方向为信息论，密码学和编码学。

后的输出是 A,B,C,D,E,F,G,H 的级联。图1、图2分别给出了算法的一次操作和主循环。

2.1 消息填充

消息填充包括三部分：原始消息、散列长度和原始消息的长度。首先在原始消息 m 后面添加一个1，然后添加 d 个零，使得新消息的长度为439mod512。接下来的9个比特填充要输出的散列长度 h 的二进制表示。最后的64比特填充原始消息的长度 l，$l=64$。即：$M=m\|1\|0^d\|h\|l$ 。

2.2 消息扩展

算法把填充后的消息 M 进行扩展。用下面的方法将512-位消息分组从16个32-位字（M_0-M_{15}）扩展成80个32-位字（M_0-M_{79}）。

前16个字保持不变，其余的字为：

$M_t=M_{t-16}\oplus M_{t-1}$，$16\leqslant t\leqslant 79$，$t$ 不是16的倍数；

$M_t=M_{t-16}\oplus S[\mathrm{Rotl}(M_{t-1})]$，$16\leqslant t\leqslant 79$，$t$ 为16的倍数。

其中 $\mathrm{Rotl}(M_{t-1})$ 表示将 M_{t-1} 左循环移位一次。S 是非线性部分，设 $M_{t-1}=\mathrm{Rotl}(M_{t-1})$，把 M_{t-1} 表示成左右相等的两半部分，分别记为 $M_{(t-1)L}$，$M_{(t-1)R}$，把 M_{t-16} 也表示成左右相等的两半部分 $M_{(t-16)L}$，$M_{(t-16)R}$，然后做下列运算：

$M_{tR}=[(M_{(t-1)L}\ \mathrm{I}\ M_{(t-16)L})<<<1]\oplus M_{(t-1)R}$；

$M_{tL}=[M_{tR}\ \mathrm{U}\ M_{(t-16)R}]\oplus M_{(t-1)RL}$。

则 $S[\mathrm{Rotl}(M_{t-1})]=M_{tL}\|M_{tR}$。

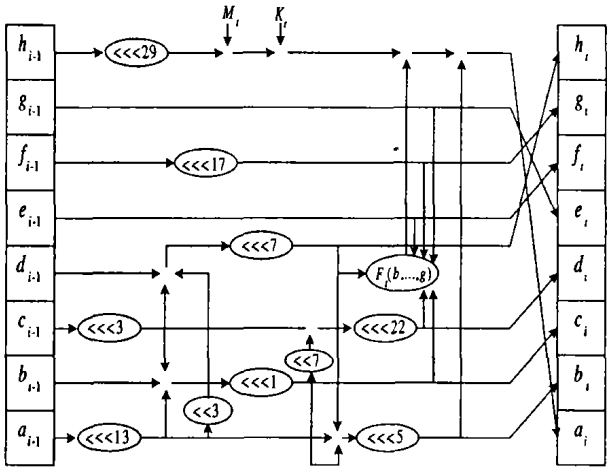

图1 算法的一次操作

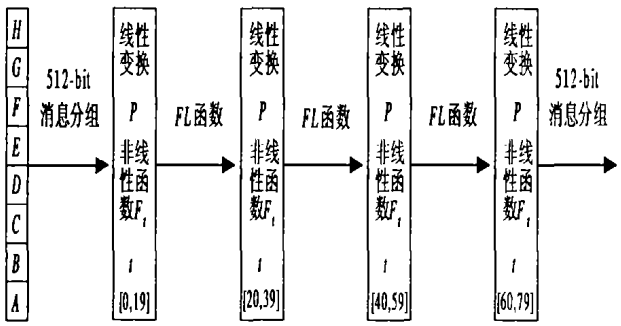

图2 算法主循环

2.3 初始变量与轮常数

算法共需要12个32-位变量。其中8个为初始变量，4个为轮常数。8个初始变量对应自然对数底 e 的小数部分的前256比特，4个轮常数对应接下来的128比特。下面列出了算法使用的12个32-位变量。

$A=0\mathrm{xb7e15162}$ $B=0\mathrm{x8aed2a6a}$ $C=0\mathrm{x8aed2a6a}$

$D=0\mathrm{x9cf4f3c7}$ $E=0\mathrm{x62e7160f}$ $F=0\mathrm{x38b4da56}$

$G=0\mathrm{x78d904}$ $H=0\mathrm{x5190cfef}$

$K_t=0\mathrm{x324e7738}$，$t\in[0,19]$；$K_t=0\mathrm{x926cfbc5}$，$t\in[20,39]$；

$K_t=0\mathrm{xf4bf8d8d}$，$t\in[40,59]$；$K_t=0\mathrm{x8c31d7b3}$，$t\in[60,79]$

具体取法是：设 $\overline{A}$ 表示 $(e-2)\times 2^{32}$ 的整数部分，$\underline{A}$ 表示 $(e-2)\times 2^{32}$ 的小数部分，A 就是 $\overline{A}$ 的十六进制表示。然后用 $\overline{B}$ 表示 $\underline{A}\times 2^{32}$ 的整数部分，$\underline{B}$ 表示 $\underline{A}\times 2^{32}$ 的小数部分，B 就是 $\overline{B}$ 的十六进制表示。同样的，再用 $\underline{B}\times 2^{32}$ 分别取整和取小数部分，得到 C，一直做下去直到得到全部12个变量。

我们用下面的程序产生这些变量，数组 a 表示存放 e 后面的小数部分（十进制），数组 b 表示存放二进制数，js 为 e 小数部分的总个数。

```
k=0;
while(1)
{
   for(js1=0,i=js-1;i≥0;i--)
   {
      c=(a[i]*2+js1)%10;
      if((a[i]*2+js1)>9) js1=1;
      else js1=0;
      a[i]=c;
   }
   b[k++]=js1;
   if(k>2000) break;
}
```

2.4 非线性函数 F

在算法主循环的四轮中采用不同的非线性函数，分别为：

$F_t(x_0,x_1,x_2,x_3,x_4,x_5)=x_0\oplus x_1\oplus x_0x_2\oplus x_1x_3\oplus x_4\oplus x_5$，$t\in[0,19]$；

$F_t(x_0,x_1,x_2,x_3,x_4,x_5)=x_0\oplus x_1\oplus x_0x_2\oplus x_1x_3\oplus x_0x_1x_2\oplus x_4\oplus x_5$，$t\in[20,39]$；

$F_t(x_0,x_1,x_2,x_3,x_4,x_5)=x_0x_1\oplus x_0x_2\oplus x_1x_3\oplus x_4\oplus x_5$，$t\in[40,59]$；

$F_t(x_0,x_1,x_2,x_3,x_4,x_5)=x_0x_1\oplus x_0x_2\oplus x_1x_3\oplus x_0x_1x_2x_3\oplus x_4\oplus x_5$，$t\in[60,79]$。

上述四个非线性函数具有良好的性质：平衡性、扩散特性、互不等价、具有高的非线性度。

2.5 线性变换 P

算法的每一轮采用不同的线性变换，但是线性变换的结构是类似的。先把8个初始变量赋值到另外的变量中，$A\to a$，$B\to b$，$C\to c$，$D\to d$，$E\to e$，$F\to f$，$G\to g$，$H\to h$。

```
对 t∈[0,19]                    对 t∈[20,39]
h:=h<<<29;                     h:=h<<<29;
a:=a<<<13;                     a:=a<<<13;
c:=c<<<3;                      d:=d<<<3;
b:=b⊕a⊕c;                      c:=c⊕a⊕d;
d:=d⊕c⊕(a<<<3);                e:=e⊕d⊕(a<<<3);
b:=b<<<1;                      c:=c<<<1;
d:=d<<<7;                      e:=e<<<7;
a:=a⊕b⊕d;                      a:=a⊕c⊕e;
c:=c⊕d⊕(b<<<7);                d:=d⊕e⊕(c<<<7);
a:=a<<<5;                      a:=a<<<5;
c:=c<<<22;                     d:=d<<<22;
temp=a⊕h⊕M_t⊕K_t               temp=a⊕h⊕M_t⊕K_t
    ⊕F_t(b,c,d,,f,g);              ⊕F_t(b,c,d,,f,g);
h:h=d;                         h:h=e;
d:d=c;                         e:e=d;
c:c=b;                         d:d=c;
b:b=a;                         c:c=a;
a:a=temp;                      a:a=temp;
temp1=g;                       temp1=g;
g:=f<<<17;                     g:=f<<<17;
f:f=e;                         f:f=b;
e:e=temp1。                    b:b=temp1。
对 t∈[40,59]，                 对 t∈[60,79]，
h:=h<<<29;                     h:=h<<<29;
a:=a<<<13;                     a:=a<<<13;
e:=e<<<3;                      f:=f<<<3;
d:=d⊕a⊕e                       e:=e⊕a⊕f;
f:=f⊕e⊕(a<<<3);                g:=g⊕f⊕(a<<<3);
d:=d<<<1;                      e:=e<<<1;
```

$f_i:=f<<<7$;	$g_i:=g<<<7$;
$a_i:=a\oplus d\oplus f$;	$a_i:=a\oplus e\oplus g$;
$e_i:=e\oplus f\oplus(d<<7)$;	$f_i:=f\oplus g\oplus(e<<7)$;
$a_i:=a<<<5$;	$a_i:=a<<<5$;
$e_i:=e<<<22$;	$f_i:=f<<<22$;
temp$=a\oplus h\oplus M_i\oplus K_i$ $\oplus F_i(b,c,d,,f,g)$	temp$=a\oplus h\oplus M_i\oplus K_i$ $\oplus F_i(b,c,d,,f,g)$;
$h_i:h=f$;	$h_i:h=g$;
$f_i:f=e$;	$g_i:g=f$;
$e_i:e=d$;	$f_i:f=e$;
$d_i:d=a$;	$e_i:e=a$;
$a_i:a=$temp;	$a_i:a=$temp;
templ$=g$;	templ$=d$;
$g_i:=c<<<17$;	$d_i:=c<<<17$;
$c_i:c=b$;	$c_i:c=b$;
$b_i:b=$templ。	$b_i:b=$templ。

其中<<<n表示左循环移位n位，<<n表示左移n位。

2.6 *FL*函数

在算法主循环的四轮中相邻两轮之间插入一个*FL*函数(图3)。

先将8个32-位变量级联，然后分成左右相等的两部分，每部分128比特。设第i轮的输出为H_i，$H_0=a\|b\|c\|d\|e\|f\|g\|h$。*FL*函数的输入为$H'_{i-1}$，$H_i$，输出为$H'_i$。

$H'_{iR}=[(H_{iL}\ \mathrm{I}\ H'_{(i-1)L})<<<1]\oplus H_{iR}$，

$H'_{iL}=[(H'_{iR}\ \mathrm{U}\ H'_{(i-1)R})]\oplus H_{iL}$，

$H'_i=H'_{iL}\|H'_{iR}$。

然后再将256比特划分为8个32-位变量，继续做算法运算。

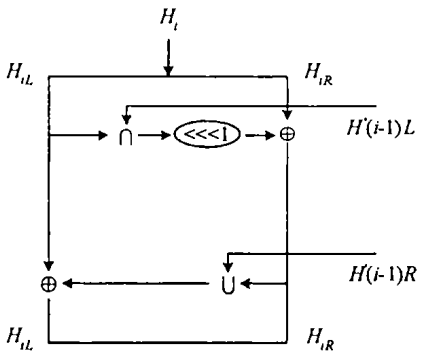

图3 FL函数

2.7 算法散列输出

对消息的所有分组进行主循环运算，最后的输出结果为：(*)$A\|B\|C\|D\|E\|F\|G\|H$。

如果要求的散列值为256比特，则(*)即为输出结果。否则，用X_i^j表示x_i的一个j比特串。

情况1(128比特散列值)：把E,F,G,H分解为

$H=H_1^8H_2^8H_3^8H_4^8$，$G=G_1^8G_2^8G_3^8G_4^8$

$F=F_1^8F_2^8F_3^8F_4^8$，$E=E_1^8E_2^8E_3^8E_4^8$。

128比特散列值为$Y_3Y_2Y_1Y_0$，这里：

$Y_3=D\oplus(H_1^8G_2^8F_3^8E_4^8)$，$Y_2=C\oplus(H_2^8G_3^8F_4^8E_1^8)$，

$Y_1=B\oplus(H_3^8G_4^8F_1^8E_2^8)$，$Y_0=A\oplus(H_4^8G_1^8F_2^8E_3^8)$。

情况2(160比特散列值)：把F,G,H分解为

$H=H_1^7H_2^7H_3^7H_4^6H_5^6$，$G=G_1^7G_2^6G_3^7G_4^6G_5^6$，$F=F_1^7F_2^6F_3^7F_4^6F_5^6$。

160比特散列值为$Y_4Y_3Y_2Y_1Y_0$，这里：

$Y_4=E\oplus(H_1^7G_2^6F_3^7)$，$Y_3=D\oplus(H_2^6G_3^7F_4^6)$，

$Y_2=C\oplus(H_3^7G_4^6F_5^6)$，$Y_1=B\oplus(H_4^6G_5^6F_1^7)$，

$Y_0=A\oplus(H_5^6G_1^7F_5^6)$。

情况3(192比特散列值)：把分G,H解为

$H=H_1^6H_2^5H_3^5H_4^6H_5^5H_6^5$，$G=G_1^6G_2^5G_3^5G_4^6G_5^5G_6^5$。

192比特散列值为$Y_5Y_4Y_3Y_2Y_1Y_0$，这里：

$Y_5=F\oplus(H_1^6G_2^5)$，$Y_4=E\oplus(H_2^5G_3^5)$，

$Y_3=D\oplus(H_3^5G_4^6)$，$Y_2=C\oplus(H_4^6G_5^5)$，

$Y_1=B\oplus(H_5^5G_6^5)$，$Y_0=A\oplus(H_6^6G_1^6)$。

情况4(224比特散列值)：把H分解为

$H=H_1^5H_2^5H_3^4H_4^5H_5^4H_6^5H_7^4$

224比特散列值为$Y_6Y_5Y_4Y_3Y_2Y_1Y_0$，这里：

$Y_6=G\oplus H_7^4$，$Y_5=F\oplus H_6^5$，

$Y_4=E\oplus H_5^4$，$Y_3=D\oplus H_4^5$，

$Y_2=C\oplus H_3^4$，$Y_1=B\oplus H_2^5$，

$Y_0=A\oplus H_1^5$。

3 设计诱因

3.1 非线性函数*F*

算法主循环所用的非线性函数是整个Hash算法的最主要部分，它们的性质决定了整个算法的安全性和有效性。我们的目标是设计四个六变元的布尔函数，这些函数具有如下性质：

(1)0-1平衡；(2)具有高的非线性度；(3)满足严格雪崩特性(SAC)；(4)结构上不等价；(5)有效性，所需指令数目尽可能少。

定义1 我们称非线性函数f和g在结构上等价，是指利用一个线性变换可以把f变为g。即存在一个可逆矩阵A和向量B满足$f(xA\oplus B)=g(x)$或者$f(xA\oplus B)\oplus 1=g(x)$，这里$A,B\in GF(2)$。否则称$f$和$g$是结构上不等价的。一个非线性函数集合被称为是线性不等价的，如果这个集合中的任意两个非线性函数都是结构上不等价的。

我们这样构造满足上述性质的非线性函数：首先，选取两个二元布尔函数，$f=x_0\oplus x_1$，$g=x_0x_1$。然后构造四元Bent函数$F=x_0\oplus x_1\oplus x_0x_2\oplus x_1x_3$，$G=x_0x_1\oplus x_0x_2\oplus x_1x_3$。由$F,G$可以构造出四个互不等价的Bent函数：

$F_1=x_0\oplus x_1\oplus x_0x_2\oplus x_1x_3$，

$F_2=x_0\oplus x_1\oplus x_0x_2\oplus x_1x_3\oplus x_0x_1x_2$，

$G_1=x_0x_1\oplus x_0x_2\oplus x_1x_3$，

$G_2=x_0x_1\oplus x_0x_2\oplus x_1x_3\oplus x_0x_1x_2x_3$。

从而就可以构造出满足性质要求的六元非线性函数：

$F_1(x_0,x_1,x_2,x_3,x_4,x_5)=x_0\oplus x_1\oplus x_0x_2\oplus x_1x_3\oplus x_4\oplus x_5$，

$F_2(x_0,x_1,x_2,x_3,x_4,x_5)=x_0\oplus x_1\oplus x_0x_2\oplus x_1x_3\oplus x_0x_1x_2\oplus x_4\oplus x_5$，

$F_3(x_0,x_1,x_2,x_3,x_4,x_5)=x_0x_1\oplus x_0x_2\oplus x_1x_3\oplus x_4\oplus x_5$，

$F_4(x_0,x_1,x_2,x_3,x_4,x_5)=x_0x_1\oplus x_0x_2\oplus x_1x_3\oplus x_0x_1x_2x_3\oplus x_4\oplus x_5$。

根据参考文[4]中的已知结果(定理3.9.5，定理4.1.9，定理4.1.10)，上述四个六元非线性函数满足0-1平衡性，*SAC*，结构上不等价并且具有高的非线性度为24(六元非线性函数的最高非线性度为28)。同时，有效性是显然的。

3.2 线性变换*P*

算法主循环四轮采用了四个结构相同的线性变换，该线性变换的选取原因是为了极大化雪崩效果和易于在处理器中实现。由于差集{0,1,3,5,7,13,17,22,29}模32没有公因子，这表明一个单比特变化在两次或更多次操作后将产生最大数目的比特变化[5]。

3.3 *FL*函数

算法主循环中两轮之间插入了一个*FL*函数，这样设计的目的是为了抵抗将来的未知攻击。

(下转第197页)

方式下，软件开发人员不必进行程序编写，只需要将获取的组件进行集成组装，最终得到一个 GIS 应用系统。这种方式下的 GIS 应用系统开发过程如图3所示。

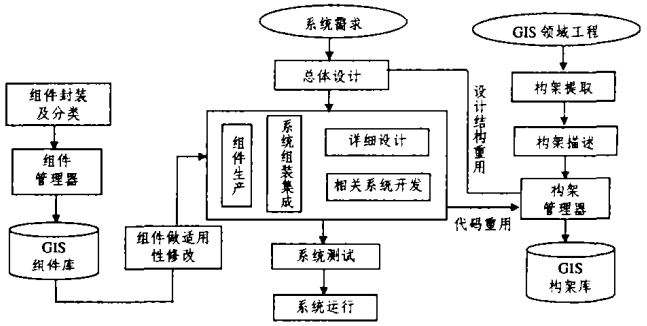

图3 组件式 GIS 应用系统开发过程

结束结 根据国内外 GIS 技术的发展现状，本文较为详细地提出了一种组件式 GIS 系统的设计方案，目前笔者主持开发的组件式 GIS 系统已初具规模，希望笔者的研究能对建立具有自主知识产权的地理信息系统产生积极的促进作用。

参 考 文 献

1 黄杏元，汤勤. 地理信息系统概论[M]. 北京：高教出版社，1998

2 周心铁，刘毓华. 组件技术与 GIS 的发展[J]. 计算机世界，1998，3

3 宋关福，钟耳顺. 组件式地理信息系统研究与开发[J]. 中国图像图形学报，1998，5

（上接第191页）

算法中使用的函数与 MISTY[6]中使用的函数具有相同的设计指标，不同之处在于前者加了一个左循环移位，这个改变并没有影响软硬件的规模和速度，但使得攻击者更难分析。FL 函数的设计指标是当函数的输入 H_{i-1} 固定时，函数必须是线性的，这样就不会增加算法的平均线性和差分概率。而且由于函数的结构都是一些逻辑运算 AND，OR，XOR 和循环移位，使得软硬件实现的速度非常快。

4 安全性分析

4.1 生日攻击

这种攻击方法不涉及算法的结构，可以攻击任何 Hash 算法。由于算法具有可变的散列长度，生日攻击方法所需的操作数分别为 2^{64}，2^{80}，2^{96}，2^{112}，2^{128}。Preneel[7]指出用生日攻击的方法获得具有160比特消息散列值的一对消息碰撞至少要20年的时间。Pieprzyk 和 Sadeghiyan[8]建议用128比特的消息散列值来抵抗这种攻击方法。

4.2 随机攻击

随机攻击就是攻击者随机选取一个消息或消息的一部分，并且期望该消息的散列值与给定消息的散列值相同。这种攻击成功的概率为 $1/2^r$，r 表示散列长度。消息散列值为64比特就可以抵抗这种攻击。

4.3 中间相遇攻击

这是一种选择明文/密文的攻击，是生日攻击方法的一种变形。用于攻击使用轮函数的 Hash 算法，其成功的概率与生日攻击相同。我们的算法可以抵抗这种攻击。

4.4 差分密码分析

Tom Berson 用差分密码分析成功攻击了 MD5的单轮[9]，但此种攻击对全部四轮无效。相比之下，Bert den Boer 和 Antoon Bosselaers 的攻击成功得多，他们使用 MD5中的压缩函数产生碰撞[10]。但是，他们的攻击方法对我们的算法无效。其原因与 SHA 算法抗这种攻击的原因相同，在算法中增加了变量，没有使用 F_t 已经使用过的 b,c,d,e,f,g。而且，由于引入了 FL 函数使得攻击算法的全部四轮更加困难。

5 算法比较

与 MD 系列，SHA 和 HAVAL 算法相比，我们的算法既继承了他们的优点，又做了相当的改变。主要的改变是添加了 FL 函数和线性变换。

（1）同 SHA 算法一样，增加了第四轮。但是已经证明 MD 系列和 SHA 使用的布尔函数具有等价结构。我们重新设计了每轮使用的非线性函数，这些布尔函数具有良好的性质。

（2）与 HAVAL 算法一样，算法的输出也是可变的。但是我们对消息进行了扩展变换，变换的方法与 SHA 不同。而 HAVAL 算法和 MD 系列都是采用相同的消息，只是每一轮使用的次序不同。

（3）算法保留了 MD4和 SHA 的方案，每隔20轮重复使用相同的轮常数。

（4）添加了线性变换，这将达到最佳的雪崩效应。

（5）增加了 FL 函数，可以抵抗将来的未知攻击。

（6）速度。我们对算法进行仿真（Inter Celeron Processor 500MHZ，C 语言编程），用160比特的散列输出与 SHA 算法比较，前者平均速度为3.70Mbyte/s，后者为6.64Mbyte/s。虽然算法的运行速度比 SHA 算法慢了近45%，但是对于实际的需求来说，算法的速度已经足够了。

参 考 文 献

1 Damgard I. A design principle for hash functions. In: G. Brassard, ed. Advances in Cryptology-Proceedings of Crypto'89, Lecture Notes in Computer Science, Springer-Verlag, 1990, 435: 416～427

2 Schneier B. Applied Cryptography: protocols, algorithms, and source code in C, 2nd Rev edition. John Wiley, 1996

3 Zheng Y, Pieprzyk J, Seberry J. HAVAL-A one-way hashing algorithm with variable length of output. Advances in Cryptology-AUSCRYPT'92 Proceedings, Springer-Verlag, 1993. 83～104

4 温巧燕，等. 现代密码学中的布尔函数. 科学出版社，2000

5 Anderaon R, Biham E, Knudaen I. Serpent: A proposal for the advanced encryption standard. NIST AES Proposal, 1998

6 Matsui M. New block encryption algorithm MISTY. Fast Software Encryption, FSE'97, LNCS 1267, Springer-Verlag, 1997. 54～68

7 Preneel B. Analysis and design of cryptographic hash functions. PhD thesis, Katholieke University Leuven, Jan. 1993

8 Pieprzyk J, Sadeghiyan B. Design of hashing algorithms. Springer-Verlag, 1993

9 Berson T. Differential cryptanalysis Modwith applications to MD5. Advances in Cryptology-EUROCRYPT'92 Proceedings, 1992. 71～80

10 den Boer B, Bosselaers A. Collisions for the compression function of MD5. Advances in Cryptology-EUROCRYPT'93 Proceedings, Springer-Verlag, 1994. 293～304