单向 Hash 函数的一种构造方法*

王小云　　周大水

（山东大学数学系　济南　250100）　　（山东大学计算机系　济南　250100）

摘要　单向 Hash 函数已成为密码学的一个重要组成部分. 给定任一定长单向 Hash 函数 $f:\Sigma^m\to\Sigma^t, m>t$, 本文给出了利用 f 构造一单向 Hash 函数 F 的一种新方法, 该方法易于并行化.

关键词　定长单向 Hash 函数, 单向 Hash 函数, 碰撞.

单向 Hash 函数亦称为 MAC 码、信息检验和、信息摘要等, 它是数据完整性检测及数字签名的重要组成部分.[1,2]现实的数字签名过程一般是对单向 Hash 函数作用之后的信息(常称为信息的摘要)进行签名, 如 CCITT X.509, ISO/IEC 9796 及美国数字签名标准 DSS 等标准中规定的模式. 这就要求所使用的 Hash 函数 F 具有如下性质:

(F.1) F 能 Hash(杂凑)任意长度的信息, 亦即输入为任意长度的 0—1 序列.

(F.2) F 的输出是固定长度的 0—1 序列.

(F.3) 给定 F 与 x, $F(x)$ 是易于计算的.

(F.4) 给定 F 及随机选取的 x, 找出 $x'\neq x$, 使 $F(x)=F(x')$ 是计算上困难的. 这样的 x 和 x' 称为一个碰撞.

满足上述(F.1)～(F.4)条件的函数 F 称为弱单向 Hash 函数.

当条件(F.4)加强为如下条件:

(F.4)' 给定 F, 找出任何 x, x', 满足 $x\neq x'$, 但 $F(x)=F(x')$ 是计算上困难的.

这时称 F 为强单向 Hash 函数. 本文简称为单向 Hash 函数. 单向 Hash 函数的形式定义见文献[3].

单向 Hash 函数一般都是由定长单向 Hash 函数以某种方式迭代而成, 因此对单向 Hash 函数的研究大致可分为 2 个研究方向. 一是如何构造具体的定长单向 Hash 函数, 这些单向 Hash 函数通常由传统分组密码或者由具有单向运算性质的函数构造而来. 另一个研究方向是利用定长单向 Hash 函数来构造单向 Hash 函数. 本文的内容属于这后一方向.

目前, 由定长单向 Hash 函数来构造单向 Hash 函数的方法基本上都是顺序迭代法, 其中最具有代表性的有 Damgard[3]方法、Mata[4]方法、Noar—Yung[5]方法等. 本文的主要目的

---

* 本文研究得到信息安全国家重点实验室基金资助. 作者王小云, 女, 1966 年生, 副教授, 主要研究领域为数论与密码学. 周大水, 1963 年生, 讲师, 主要研究领域为计算机网络安全理论和技术.

本文通讯联系人: 王小云, 济南 250100, 山东大学数学系

本文 1995—07—11 收到修改稿

是给出构造单向 Hash 函数的另一种新方法，不同于以往的顺序迭代法，本文采用的是分组杂凑的方法，其最大的一个特点是适用于并行环境.

设 f 是任一定长单向 Hash 函数，$f:\Sigma^m\to\Sigma^t$，$m>t$，$\Sigma=\{0,1\}$. 本文将给出一种由 f 构造单向 Hash 函数 F 的新方法，并证明 F 的每一碰撞都可导致 f 的一个碰撞，因此，f 的单向性可遗传给 F.

本文出现的符号 x,x_i,y,s_i,u_i 等均表示二进制 0—1 序列，$x*y$ 表示 x 和 y 的连结，x 为 $x*y$ 的前缀. $|x|$ 表示 x 的长度，如 $|01101|=5$. 另外，还需明确如下 2 个术语：

(1)对报文 x 进行 l—分组是指将 x 表示成 $x=u_1*u_2*\cdots*u_k$，其中 $|u_i|=l,(i=1,2,\cdots,k-1),|u_k|\leqslant l$.

(2)对报文 x 进行 0—m 填充为 x^*，其中 $|x|\leqslant m$，是指在 x 之后填 0 得 $x^*=x*0*0*\cdots*0$，使 $|x^*|=m$.

1 构造方法

假设 f 是从 Σ^m 杂凑到 Σ^t 的定长单向 Hash 函数($m>t$)，根据参数 m 与 t 之间关系，我们分 2 种情形利用函数 f 来设计一个单向 Hash 函数 F. 另外，在本文中所采用的参数 s_0,s_1 为公开的初始值，且 $|s_0|=m-1,|s_1|=m$.

情形 1. 当 $m-t>1$ 时，对任一报文 x，我们用如下方法来杂凑 x.

对报文 s_0*x 进行($m-1$)—分组，$s_0*x=s_0*u_1*u_2*\cdots*u_k,|u_i|=m-1,1\leqslant i\leqslant k-1,|u_k|\leqslant m-1$. 利用定长单向 Hash 函数 f 对报文 s_0*x 进行第 1 次压缩得 $x_1=u_0'*u_1'*u_2'*\cdots*u_k'$，其中 $u_0'=f(s_0*1),u_i'=f(u_i*0),1\leqslant i\leqslant k-1$；当 $|u_k|=m-1$ 时，$u_k'=f(u_k*0)$，否则 $u_k'=u_k$. 称 x_1 为报文 x 进行第 1 次压缩后的报文.

对 x_1 再进行 1 次压缩得 x_2，所不同的是，假设 x_1 的($m-1$)—分组为 $x_1=u_{11}*u_{12}*\cdots*u_{1k_1}$，则 $x_2=u_{11}'*u_{12}'*\cdots*u'_{1k_1}$，其中 $u_{1i}'=f(u_{1i}*0),1\leqslant i\leqslant k_1-1$；当 $|u_{1k_1}|=m-1$ 时，$u'_{1k_1}=f(u_{1k_1}*0)$，否则 $u'_{1k_1}=u_{1k_1}$.

与对 x_1 的压缩规则相同，连续对 x 进行多次压缩，直到第 r 次压缩后报文 x_r 满足 $|x_r|\leqslant m-1$，对 x_r 的 0—m 补充 x_r^* 计算 $x_{r+1}=f(x_r^*)$.

令 $\bar{x}_{r+1}=x_{r+1}*\bar{u},\bar{u}$ 为 x_r^* 中补充 0 的个数 d 的长为 $m-t$ 的二进制表示，d 的长度不一定为 $m-t$，如不足，则在左边添加 0. 由于 $|s_0*x|\geqslant m-1$，知 $|x_i|\geqslant t(i=1,2\cdots,r)$，从而 $d\leqslant m-t$，因此 d 是可以用一个长为 $m-t$ 的二进制串来表示.

最后计算 $y=f(\bar{x}_{r+1})$，令 $F(x)=y$.

由上述描述知 $F(x)$ 将任一报文杂凑到长为 t 的二进制串，r 为压缩总次数.

以上计算可由下列算法表示：

算法 1.

输入：x,s_0.

第 1 步：对 s_0*x 进行($m-1$)—分组，$s_0*x=s_0*u_1*u_2*\cdots*u_k,|u_i|=m-1,1\leqslant i\leqslant k-1;|u_k|\leqslant m-1$.

$$u_0'\leftarrow f(s_0*1)\qquad u_i'\leftarrow f(u_i*0),i=1,2,\cdots,k-1$$

$$u_k'=\begin{cases}f(u_k*0), & |u_k|=m-1\\ u_k, & |u_k|<m-1\end{cases}\qquad X\leftarrow u_0'*u_1'*u_2'*\cdots*u_k'$$

$$n\leftarrow|X|$$

第2步：若$n<m-1$，则执行第4步.

第3步：对X进行$(m-1)$－分组，$X=u_0*u_1*u_2*\cdots*u_k$

$$u_i'\leftarrow f(u_i*0),1\leqslant i\leqslant k-1\qquad u_k'=\begin{cases}f(u_k*0), & |u_k|=m-1\\ u_k, & |u_k|<m-1\end{cases}$$

$$X\leftarrow u_0'*u_1'*u_2'*\cdots*u_k'\qquad n\leftarrow|X|$$

回到第2步.

第4步：令$y\leftarrow f(f(X^*)*\bar{\bar{u}})$，其中$X^*$为$X$的$0-m$补充，$\bar{\bar{u}}$表示$d$的$m-t$位二进制表示，$d$为$X^*$中补充0的个数.

第5步：输出y.

情形2.

当$m-t=1$时，则设计如下算法.

算法2.

输入：x,s_0,s_1.（$|s_0|=m-1,|s_1|=m$）

第1步：对x进行m－分组，$x=u_1*u_2*\cdots*u_k$.

$$s_0'\leftarrow f(s_0*1)\qquad s_1'\leftarrow f(s_1)$$

$$u_i'\leftarrow f(u_i),1\leqslant i\leqslant k-1\qquad u_k'=\begin{cases}f(u_k), & |u_k|=m\\ u_k, & |u_k|<m\end{cases}$$

$$X\leftarrow s_0'*s_1'*u_1'*u_2'*\cdots*u_k'\qquad n\leftarrow|X|$$

第2步：若$n<2m-1$，则执行第4步.

第3步：对X进行如下分组，$X=u_1*u_2*\cdots*u_k$，$|u_1|=m-1$，$|u_i|=m$，$2\leqslant i\leqslant k-1$，$|u_k|\leqslant m$.

$$u_1'\leftarrow f(u_1*0)\qquad u_i'\leftarrow f(u_i),2\leqslant i\leqslant k-1$$

$$u_k'=\begin{cases}f(u_k), & |u_k|=m\\ u_k, & |u_k|<m\end{cases}\qquad X\leftarrow u_1'*u_2'*\cdots*u_k'$$

$$n\leftarrow|X|$$

回到第2步.

第4步：当$m\leqslant n<2m-1$时，重复如下循环. 令$X=u_1*u_2$，$|u_1|=m$，$|u_2|\leqslant m-1$.

$$u_1'\leftarrow f(u_1)\qquad u_2'\leftarrow u_2$$

$$X\leftarrow u_1'*u_2'\qquad n\leftarrow|X|$$

输出：X.

将以上算法输出的X定义为$F(x)$.

综上所述，对任一将Σ^m杂凑到$\Sigma^t(m>t)$的定长单向Hash函数$f(x)$，我们可以利用$f(x)$设计一Hash函数$F(x)$，它将任一长度报文杂凑到长为t的二进制串.

2　Hash函数的单向性证明

上一节我们已经描述了Hash函数$F(x)$的设计方法，下面来证明$F(x)$为一单向Hash

函数. 显然 $F(x)$满足单向 Hash 函数的定义中的条件(F. 1)～(F. 3),因此只要证明条件(F. 4)′成立即可.

条件(F. 4)′可通过证明 $F(x)$的一个碰撞将导致 $f(x)$的一个碰撞来保证. 我们将按构造 F 的 2 种情形分别加以证明.

情形 1 的证明:

当 $m-t>1$ 时,假设对于某一报文 x,我们找到了另一报文 x',$x\neq x'$,使 $F(x)=F(x')$. 下面证明在多项式时间内可以找到 $f(x)$的一个碰撞.

设 r,r'分别为计算 $F(x),F(x')$所需压缩总次数,x_i,x_i'分别为进行第 i 次压缩后的报文. 由于 $F(x)=F(x')$,则知 $f(x_{r+1}*\bar{u})=f(x'_{r'+1}*\overline{u'})$,其中 $\bar{u},\overline{u'}$所表示整数分别为 x_r,$x'_{r'}$的 0−m 补充中 0 的个数,且 $|\bar{u}|=|\overline{u'}|=m-t$. 显然,当 $\bar{u}\neq\overline{u'}$时,我们找到了 $f(x)$的一个碰撞,从而结论得证.

当 $\bar{u}=\overline{u'}$时,下面我们分 2 种情况来讨论.

(1)当 $r=r'$时,可以推知 $|x_i|=|x'_i|(i=1,2,\cdots,r+1)$. 此时当 $x_{r+1}\neq x'_{r+1}$时,即可找到 $f(x)$的一个碰撞.

当 $x_{r+1}=x'_{r+1}$时,观察 x_r 与 x'_r. 由于 $x_r=x'_r$ 当且仅当 $x_r^*=(x'_r{}^*)$,则当 $x_r\neq x_r'$时,由 $x_{r+1}=f(x_r^*)=x'_{r+1}=f(x'_r{}^*)$可知找到 $f(x)$的一个碰撞.

当 $x_r=x'_r$ 时,观察 x_{r-1},x'_{r-1}. 依此类推,由于 $x\neq x'$,总存在 $i_0<r$,使 $x_{i_0}=x'_{i_0}$,但 $x_{i_0-1}\neq x'_{i_0-1}$,从而知存在 x_{i_0-1}的某一长为 $m-1$ 的报文块 u_i 与 x'_{i_0-1}相应的报文块 u'_i,满足 $u_i\neq u'_i$,但 $f(u_i*0)=f(u'_i*0)$,从而可找到 $f(x)$的一个碰撞.

(2)当 $r\neq r'$时,不妨假设 $r>r'$. 由于 $f(x_{r+1}*\bar{u})=f(x'_{r'+1}*\overline{u'})$且 $|\bar{u}|=|\overline{u'}|$,当 $x_{r+1}\neq x'_{r'+1}$时,即可找到 $f(x)$的一个碰撞.

当 $x_{r+1}=x'_{r'+1}$时,考虑 x_r 与 $x'_{r'}$,同情况(1)的讨论一样,假设存在一个 i_0 满足 $2\leqslant i_0\leqslant r'$,使 $x'_{i_0}=x_{i_0+(r-r')}$,但 $x_{i_0-1}\neq x_{i_0-1+(r-r')}$,那么一定存在 $x_{i_0-1+(r-r')}$的一长为 $m-1$ 报文块 u_i 与 x'_{i_0-1}相应报文块 u'_i,使 $u_i\neq u'_i$,但 $f(u_i*0)=f(u'_i*0)$,从而可找到 $f(x)$的一个碰撞.

若对任意的 $i_0,2\leqslant i_0\leqslant r'$,$x'_{i_0}=x_{i_0+(r-r')}$且 $x'_{i_0-1}=x_{i_0-1+(r-r')}$,则知 $x'_1=x_{1+(r-r')}$,因此 $f(u'_1*1)=f(u_1*0)$,其中 u'_1,u_1 分别为 s_0*x',$x_{r-r'}$的第一个长为 $m-1$ 的报文块,显然 u'_1*1,u_1*0 即为 $f(x)$的一个碰撞. 与 $f(x)$的单向性矛盾.

情形 2 的证明:

当 $m-t=1$ 时,我们仍假设存在 $x,x',x\neq x'$ 使 $F(x)=F(x')$. 根据算法 2 即知,在计算 $F(x),F(x')$过程中,如果 $f(x)$在第 4 步发生碰撞,则结论得证. 如果 $F(x)$在第 4 步未能发生碰撞,则知第 3 步运算结果相同. 类似于情形 1 的证明知 $f(x)$如果不在第 3 步发生碰撞,则一定在第 1 步发生碰撞. 与 $f(x)$的单向性矛盾. 从而 $F(x)$的单向性得证.

3 结束语

对任一定长度的单向 Hash 函数 $f(x)$,我们可根据 m 与 t 之间的关系利用 $f(x)$构造一单向 Hash 函数 $F(x)$. 算法中所引进的初始值并且在某些固定位置添加特殊比特的处理方法,目的是为了保证 $F(x)$的单向性. 另外,本文提出的方案是对报文采用分组杂凑的方法,

所以 $F(x)$的计算特别适合于并行处理，因此，在并行环境下可减少时间复杂性.

参考文献

1 Rompel J. One—way functions are necessary and sufficient for secure signatures. Proc. of the 22nd ACM symposium on Theory of Computing, 1990. 387～394.

2 Damgard I. Collision free hash functions and public key signature schemes. Proc. of Eurocrypt'87, 1987. 203～216.

3 Damgard I. A design principal for hash functions. Crypto'89. 1989. 416～427.

4 Merkle R C. One way hash function and DES. Crypt'89, 1989. 429～426.

5 Noar M, Yung M. Universal one—way hash functions and their cryptographic applications. Proc. of the 21th ACM Symposium on Theory of Computing, 1989. 35～43.

A METHOD FOR CONSTRUCTING ONE—WAY HASH FUNCTIONS

Wang Xiaoyun

(*Department of Mathematics Shandong University Ji'nan* 250100)

Zhou Dashui

(*Department of Computer Sciences Shandong University Ji'nan* 250100)

Abstract One—way hash functions is an important part of cryptography. In this paper, a new method for constructing one—way hash function F from any fixed size one—way hash function $f:\Sigma^m\to\Sigma^t, m>t$, is presented, the property of which is that it is easy to be parallelized.

Key words Fixed size one—way hash function, one—way hash function, collision.