SHA-1充分条件自动化求解算法

胡云山 申 意 曾 光 韩文报
(解放军信息工程大学 郑州 450001) (数学工程与先进计算国家重点实验室 无锡 214125)

摘 要 充分条件的求解是模差分攻击的重要步骤之一。将充分条件的求解转化为F_2上线性方程组的构造过程,利用线性方程组解的判定定理判断每步所求得充分条件的正确性,提出了针对SHA-1模差分攻击的充分条件自动化求解算法。文中算法做适当变形后,同样适用于MD5、SHA-0等与SHA-1结构相似的Hash函数充分条件的自动化求解。

关键词 密码学,Hash函数,SHA-1,充分条件,碰撞攻击
中图法分类号 TN918.1 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.8.026

New Algorithm for Automatic Deriving Sufficient Conditions of SHA-1

HU Yun-shan SHEN Yi ZENG Guang HAN Wen-bao
(The PLA Information Engineering University, Zhengzhou 450001, China)
(State Key Laboratory of Mathematical Engineering and Advanced Computing, Wuxi 214125, China)

Abstract Deriving sufficient conditions is one of the important technologies in the differential mode attacking. In this paper, turning the problem of deriving sufficient conditions into structure of linear equations in F_2, using the judgment theorem of linear equations to determine the correctness of the sufficient conditions derived by each step, a new algorithm for automatic deriving sufficient conditions of SHA-1 hash function was proposed. This algorithm is equally applicable to derive sufficient conditions in SHA-0 which has similar structure with SHA-1 after appropriate deformation.

Keywords Cryptology, Hash function, SHA-1, Sufficient conditions, Collision attacks

1 引言

SHA-1算法是美国国家标准局于1995年公布的Hash算法,该算法被美国政府核准作为Hash函数标准算法。作为标准Hash算法,SHA-1算法在数字签名、公钥密码系统等信息安全领域有着非常广泛的应用。自SHA-1算法被提出以来,SHA-1算法的安全性研究受到了很多密码分析者的关注。

2005年,王小云教授利用部分碰撞和多消息修改技术,把对SHA-1碰撞攻击的复杂度降到了2^{69},奠定了Hash算法随机碰撞攻击方法中模差分攻击的主流地位[1]。2009年,Cameron McDonald结合飞去来器技术,给出了复杂度为2^{52}的攻击结果[2]。2011年,Rafi Chen给出了复杂度为2^{58}的SHA-1随机碰撞攻击[3]。2013年,Marc Stevens基于对部分碰撞性质的分析,给出了攻击复杂度为$2^{57.5}$的攻击结果[4,5]。2015年,Eli Biham利用中立比特等技术给出了SHA-0全轮及SHA-1减轮的碰撞数据[6]。针对SHA-1减轮碰撞的研究也不断深入[7,8],2010年,Grechnikov给出了复杂度为$2^{52.9}$的72轮和复杂度为2^{56}的73轮的SHA-1减轮碰撞攻击[9]。2011年,Grechnikov等人得出了75轮的SHA-1减轮碰撞消息对[10]。2012年,Adinetz等人也得出了一组75轮的SHA-1碰撞消息对[11]。此外,Grobner基[12]、线性码[13]以及飞去来器[14]等密码学理论工具也被应用到寻找SHA-1随机碰撞对的研究中。

由差分路径求解充分条件是模差分攻击技术的关键步骤之一。目前由差分路径求解充分条件普遍采用手动推导的方法,还没有有效的自动化求解算法。本文将充分条件用F_2上线性方程组的形式表示,利用线性方程组解的判定定理判断每步所求得的充分条件是否符合差分路径,首次提出了针对SHA-1模差分攻击的充分条件自动化求解算法。算法的有效性经编程实验得到验证。值得注意的是,本文所提出的是一个通用框架,其作适当变形后,同样适用于具有与SHA-1结构相似的MD5、SHA-0等Hash函数充分条件的自动化求解。本文仅选取SHA-1算法作为典型代表进行描述。

本文第2节简要介绍了SHA-1算法以及针对SHA-1算法的模差分攻击技术;第3节提出了针对SHA-1模差分攻击的充分条件自动化求解算法;最后结束全文。

到稿日期:2015-07-06 返修日期:2015-09-28 本文受国家自然科学基金项目(61003291),数学工程与先进计算国家重点实验室开放课题(2013A03,2013A10)资助。

胡云山(1990—),男,硕士生,CCF会员,主要研究方向为信息安全,E-mail:huyunshan@foxmail.com;申 意(1990—),男,硕士生,主要研究方向为公钥密码;曾 光(1980—),男,博士,副教授,主要研究方向为网络密码;韩文报(1963—),男,教授,博士生导师,主要研究方向为密码理论、网络安全。

2 SHA-1 及模差分攻击技术简介

2.1 SHA-1 算法简介

SHA-1 算法将任意长度的输入消息进行压缩处理后，输出长度为 160bit 的消息摘要值。算法主要包括预处理和主循环两个部分。

在预处理部分，SHA-1 算法将消息填充为 512bit 的整数倍后，每 512bit 的消息块划分为 16 个 32bit 的消息字（m_0 — m_{15}），由这 16 个 32bit 的消息字扩展至 80 个 32bit 的消息字（m_0 — m_{79}），消息扩展方式为：$m_i=(m_{i-3}\oplus m_{i-8}\oplus m_{i-14}\oplus m_{i-16})<<<1$。

SHA-1 算法包括 5 个寄存器，分别记为 a,b,c,d,e，寄存器的初始值为算法常量。在 SHA-1 算法的主循环部分，算法对这 5 个寄存器做 4 轮 80 步操作，每步操作为：$a_i=(a_{i-1}<<<5)+f_i(b_{i-1},c_{i-1},d_{i-1})+e_{i-1}+m_{i-1}+K_i$，$b_i=a_{i-1}$，$c_i=b_{i-1}<<<30$，$d_i=c_{i-1}$，$e_i=d_{i-1}$；

其中，a_i,b_i,c_i,d_i,e_i 分别表示 5 个寄存器在第 i 步的值。根据寄存器比特之间的关系，SHA-1 算法的每步操作也可以表示为：

$$a_i=(a_{i-1}<<<5)+f_i(a_{i-2},a_{i-3}>>>2,a_{i-4}>>>2)+(a_{i-5}>>>2)+m_{i-1}+K_i \quad (1)$$

在上述描述中，“$<<<$”表示循环移位，K_i 是算法常数，$f_i(b,c,d)$表示算法使用的逻辑函数，分别为：

IF：$f_i(b,c,d)=(b\wedge c)\vee(\neg b\wedge d)$

XOR：$f_i(b,c,d)=b\oplus c\oplus d$

MAJ：$f_i(b,c,d)=(b\wedge c)\vee(b\wedge d)\vee(c\wedge d)$

在 80 步操作之后，$(a_0+a_{80},b_0+b_{80},c_0+c_{80},d_0+d_{80},e_0+e_{80})$作为下一个 512 bit 分组寄存器的输入值，用下一组 512 bit 分组继续执行算法，最后一个分组$(a_{80},b_{80},c_{80},d_{80},e_{80})$的级联值为最后输出的消息摘要值。

2.2 模差分攻击技术简介

模差分随机碰撞攻击的技术基础是部分碰撞。王小云教授提出的针对 SHA-1 算法的模差分攻击主要分为以下 6 个步骤：

1）确定扰动向量；

2）由扰动向量构造高概率的差分路径；

3）由差分路径得到消息比特和寄存器比特需要满足的充分条件；

4）对第一块明文消息对进行消息修改，使得明文消息对满足尽可能多的充分条件；

5）搜索 SHA-1 近似碰撞消息对；

6）利用近似随机碰撞块构造 SHA-1 体制的随机碰撞。

在攻击过程中，充分利用 SHA-1 体制的加法运算和减差分的性质，对消息差分和寄存器差分做进位扩展操作来构造符合需要的差分路径与充分条件。进位扩展，即对于寄存器或逻辑函数第 j 比特位的减差分，在 SHA-1 体制的加法运算中减差分$(2^j,2^{j+1},\cdots,2^{j+k-1},2^{-(j+k)})$与减差分 2^{-j} 等价，减差分$(2^{-j},2^{-(j+1)},\cdots,2^{-(j+k-1)},2^{j+k})$与减差分 2^j 等价，其中 k 称为由第 j 比特位向第 $j+k$ 比特位差分进位扩展的位数。将减差分$(2^{-j},2^{-(j+1)},\cdots,2^{-(j+k-1)},2^{j+k})$或$(2^j,2^{j+1},\cdots,2^{j+k-1},2^{-(j+k)})$转化为 2^j 和 2^{-j} 的表示形式，则称将第 j 比特位的减差分转化为无进位扩展的形式。

3 SHA-1 充分条件自动化求解算法

充分条件的求解是模差分攻击中的关键步骤之一。充分条件有多种分类方法，本文将充分条件分为 3 类：消息条件、进位条件以及逻辑函数条件。消息条件是指为保证消息差分符合差分路径而必须满足的充分条件；进位条件是指在 SHA-1 体制的加法运算中为控制减差分进位扩展位数而必须满足的充分条件；逻辑函数条件则是指为使每步操作中逻辑函数的差分值符合差分路径而必须满足的充分条件。由式(1)可知，3 类充分条件均可由消息比特和寄存器 a 中的比特来表示，且表达式均为线性关系式，因此充分条件均可转换为 F_2 上线性方程的形式。令：

$$A=(a_{0,0},\cdots,a_{0,31},\cdots,e_{0,0},\cdots,e_{0,31},a_{1,0},\cdots,a_{80,31})$$

$$M=(m_{0,0},\cdots,m_{0,31},\cdots,m_{79,0},\cdots,m_{79,31})$$

其中，$a_{0,j},b_{0,j},c_{0,j},d_{0,j},e_{0,j}(0\leqslant j\leqslant 31)$分别表示 SHA-1 算法中 5 个寄存器初始值的第 j 比特，$a_{i,j},m_{i-1,j}(1\leqslant i\leqslant 80,0\leqslant j\leqslant 31)$分别表示寄存器 a 与消息 m 在第 i 步中的第 j 比特。因此，对于任意一个充分条件，均可构造一个含有 5281 个元素的向量 $S_i=(s_0,\cdots,s_{5280})$，以 S_i 为系数，构造如下 F_2 上的线性方程：

$$S_i\cdot[A\quad M\quad 1]^{\mathrm{T}}=0 \quad (2)$$

其中，$s_i\in\{0,1\}$，s_i 为 1 表示对应位置的自变量在 S_i 所代表的充分条件中存在，s_i 为 0 则表示不存在。易知该线性方程与 S_i 所对应的充分条件等价。

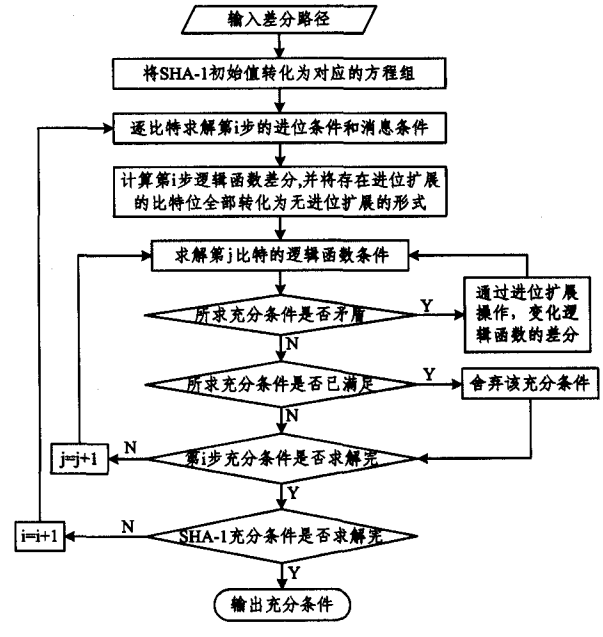

图 1 SHA-1 充分条件求解算法框架

本文算法第一步先将 SHA-1 算法中寄存器的初始值逐比特转换为对应的线性方程组，然后逐比特构造 3 类充分条件所对应的线性方程。在构造逻辑函数条件所对应的线性方程中，为减少充分条件数量，先将每步所求得逻辑函数的减差分全部转化为无进位扩展的形式，而后再构造线性方程，并且每求得一组新的逻辑函数条件，都需根据线性方程组解的情况判断该组逻辑函数条件是否与已求得的条件矛盾或重复。若存在矛盾，则对该比特位差分做进位扩展操作，而后重新求解；若已求得的充分条件已使得该逻辑函数条件成立，则舍弃该条件。待线性方程组构造完毕后，将最后所得方程组转换

为对应的充分条件并输出。基于以上描述，图 1 示出 SHA-1 碰撞充分条件自动化求解算法的框架。下面从符号定义、进位和消息条件的求解、逻辑函数条件的求解、逻辑函数条件的判定 4 个方面对算法框架进行描述。

3.1 相关符号定义

为表述方便，先给出本文部分符号说明，如表 1 所列。

表 1 相关符号定义

符号	符号说明
m, m'	SHA-1 近似碰撞或随机碰撞的明文消息对
$m_{i-1,j}, m'_{i-1,j}$	明文消息对 m，m′第 i 步的第 j 比特
$\Delta m_{i-1,j}$	明文消息对第 i 步第 j 比特的减差分
$f_{i,j}, f'_{i,j}$	逻辑函数第 i 步中的第 j 比特
$\Delta f_{i,j}$	逻辑函数第 i 步第 j 比特的减差分
$a'_{i,j}, b'_{i,j}, c'_{i,j}, d'_{i,j}$	计算 m′时，寄存器 a，b，c，d 第 i 步的第 j 比特
$a_{i,j}, b_{i,j}, c_{i,j}, d_{i,j}$	计算 m 时，寄存器 a，b，c，d 第 i 步的第 j 比特
$\Delta A_{i,j}, \Delta B_{i,j}, \Delta C_{i,j}, \Delta D_{i,j}$	寄存器 a，b，c，d 第 i 步第 j 比特的减差分
DET(H)	矩阵 H 的秩
S_t	$S_t=[s_0,\cdots,s_{5280}]^T$，$s_i$ 表示式(2)中对应的系数
S'_t	$S'_t=[s_0,\cdots,s_{5279}]^T$，$s_i$ 表示式(2)中对应的系数
D_t	系数矩阵 $D_t=[S'_1 \ \cdots \ S'_t]^T$
$\overline{D_t}$	增广矩阵 $\overline{D_t}=[S_1 \ \cdots \ S_t]^T$
X	称矩阵 X 为自变量矩阵，其中 $X=[A \ M \ 1]^T$

在下文算法描述中，$Xa_{i,j}, Xb_{i,j}, Xc_{i,j}, Xd_{i,j}, Xm_{i,j}$ 分别表示 $a_{i,j}, b_{i,j}, c_{i,j}, d_{i,j}, m_{i,j}$ 在自变量矩阵 X 中所对应的自变量。

3.2 进位条件和消息条件的求解

在充分条件的求解过程中，进位条件与消息条件是逐步、逐比特求解的。当差分路径已知时，前 16 步消息的减差分也是确定的，而 16 步之后扩展消息对存在差分的位置确定但减差分的符号不定。因此，需要构造消息条件来保证 16 步之后的消息差分符合差分路径。针对消息条件，本文算法构造如下线性方程：

$$\begin{cases} Xm_{i-1,j}=0, & \text{若 } \Delta m_{i-1,j}=1 \\ & (17\leqslant i\leqslant 80, 0\leqslant j\leqslant 31) \\ Xm_{i-1,j}\oplus 1=0, & \text{若 } \Delta m_{i-1,j}=-1 \\ & (17\leqslant i\leqslant 80, 0\leqslant j\leqslant 31) \end{cases}$$

在 SHA-1 体制的加法运算中，相同的减差分经过进位扩展操作之后可以得到不同的运算结果，因此需要构造进位条件来控制差分的进位扩展位数。针对进位条件，本文算法构造如下线性方程：

$$\begin{cases} Xa_{i,j}=0, & \text{若 } \Delta A_{i,j}=1 (1\leqslant i\leqslant 80, 0\leqslant j\leqslant 31) \\ Xa_{i,j}\oplus 1=0, & \text{若 } \Delta A_{i,j}=-1 (1\leqslant i\leqslant 80, 0\leqslant j\leqslant 31) \end{cases}$$

根据以上描述，进位条件和消息条件求解算法如算法 1 所示。

算法 1 进位条件和消息条件求解算法

输入：明文消息与寄存器 a 在第 i 步第 j 比特的减差分 $\Delta m_{i-1,j}, \Delta A_{i,j}$

输出：第 i 步第 j 比特的进位条件与消息条件所对应的线性方程

步骤 1：若 $17\leqslant i\leqslant 80$，构造消息条件所对应的线性方程如下：

a）若 $\Delta m_{i-1,j}=1$，构造方程 $Xm_{i-1,j}=0$；

b）若 $\Delta m_{i-1,j}=-1$，构造方程 $Xm_{i-1,j}\oplus 1=0$。

步骤 2：构造进位条件所对应的线性方程如下：

a）若 $\Delta A_{i,j}=1$，构造方程 $Xa_{i,j}=0$；

b）若 $\Delta A_{i,j}=-1$，构造方程 $Xa_{i,j}\oplus 1=0$。

3.3 逻辑函数条件的求解

由差分路径计算逻辑函数的减差分值是容易的。在 SHA-1 体制的加法运算中，对减差分做进位扩展操作前后的两组减差分仍然等价。因此，为减少充分条件数量，本文算法在求解逻辑函数条件之前，先将所求得的逻辑函数差分全部转化为无进位扩展的形式，而后逐比特求解。

对于逻辑函数第 i 步第 j 比特的输入参数 $b_{i,j}, b'_{i,j}$，可对表达式作如下变化：

$$b'_{i,j}=(1-\Delta B_{i,j}^2)\cdot Xb_{i,j}+\frac{(\Delta B_{i,j}+1)\cdot \Delta B_{i,j}}{2}$$

$$b_{i,j}=(1-\Delta B_{i,j}^2)\cdot Xb_{i,j}+\frac{(\Delta B_{i,j}-1)\cdot \Delta B_{i,j}}{2}$$

由上式不难得出如下结论：

$$\begin{cases} b'_{i,j}=1, b_{i,j}=0, & \text{当 } \Delta B_{i,j}=1 \\ b'_{i,j}=0, b_{i,j}=1, & \text{当 } \Delta B_{i,j}=-1 \\ b'_{i,j}=b_{i,j}=Xb_{i,j}=1 \text{ or } 0, & \text{当 } \Delta B_{i,j}=0 \end{cases}$$

因此，$b'_{i,j}$ 与 $b_{i,j}$ 可由关于 $Xb_{i,j}$ 的表达式表示。同理，$c'_{i,j}$ 与 $c_{i,j}$ 可由关于 $Xc_{i,j}$ 的表达式表示，$d'_{i,j}$ 与 $d_{i,j}$ 可由关于 $Xd_{i,j}$ 的表达式表示。不妨令：

$$Tb_{i,j}=(1-\Delta B_{i,j}^2)\cdot Xb_{i,j}+\frac{(\Delta B_{i,j}-1)\cdot \Delta B_{i,j}}{2}$$

$$Tb'_{i,j}=(1-\Delta B_{i,j}^2)\cdot Xb_{i,j}+\frac{(\Delta B_{i,j}+1)\cdot \Delta B_{i,j}}{2}$$

$$Tc_{i,j}=(1-\Delta C_{i,j}^2)\cdot Xc_{i,j}+\frac{(\Delta C_{i,j}-1)\cdot \Delta C_{i,j}}{2}$$

$$Tc'_{i,j}=(1-\Delta C_{i,j}^2)\cdot Xc_{i,j}+\frac{(\Delta C_{i,j}+1)\cdot \Delta C_{i,j}}{2}$$

$$Td_{i,j}=(1-\Delta D_{i,j}^2)\cdot Xd_{i,j}+\frac{(\Delta D_{i,j}-1)\cdot \Delta D_{i,j}}{2}$$

$$Td'_{i,j}=(1-\Delta D_{i,j}^2)\cdot Xd_{i,j}+\frac{(\Delta D_{i,j}+1)\cdot \Delta D_{i,j}}{2}$$

根据上述分析，由逻辑函数的输入参数可知：

$b_{i,j}=Tb_{i,j}, c_{i,j}=Tc_{i,j}, d_{i,j}=Td_{i,j}$

$b'_{i,j}=Tb'_{i,j}, c'_{i,j}=Tc'_{i,j}, d'_{i,j}=Td'_{i,j}$

由此可知，SHA-1 算法逻辑函数的 3 个输入变量均可由以 $Xb_{i,j}, Xc_{i,j}, Xd_{i,j}$ 为参数的线性表达式表示。逻辑函数条件求解算法先对逻辑函数的表达式做适当变化，用关于 $Xb_{i,j}, Xc_{i,j}, Xd_{i,j}$ 的表达式表示逻辑函数的减差分值，然后基于上述表示形式构造线性方程组，最后通过逻辑函数条件判定算法判断所求得的逻辑函数条件是否符合要求。本节将详细描述每轮逻辑函数条件所对应线性方程的构造方法。

3.3.1 第一轮逻辑函数条件的求解

对于 SHA-1 算法第一轮逻辑函数 IF，其可表示为：$f_{i,j}=(b_{i,j}\cdot c_{i,j})\oplus[(1-b_{i,j})\cdot d_{i,j}]$。由此可得如下等价表达式：

$f'_{i,j}=(Tb'_{i,j}\cdot Tc'_{i,j})\oplus[(1-Tb'_{i,j})\cdot Td'_{i,j}]$

$f_{i,j}=(Tb_{i,j}\cdot Tc_{i,j})\oplus[(1-Tb_{i,j})\cdot Td_{i,j}]$

分析上式可知，对于第一轮逻辑函数条件的求解，可按 $\Delta f_{i,j}, \Delta B_{i,j}, \Delta C_{i,j}, \Delta D_{i,j}$ 的取值分以下 3 种情况分别构造线性方程求解：

a）$\Delta f_{i,j}=0$；

b）$\Delta f_{i,j}=1$ or -1，$\Delta B_{i,j}$ 与 $\Delta C_{i,j}\cdot \Delta D_{i,j}$ 至少有一个不为 0；

c）$\Delta f_{i,j}=1$ or -1，$\Delta B_{i,j}=0$ 且 $\Delta C_{i,j}\cdot \Delta D_{i,j}=0$。

对于情况 a），即 $\Delta f_{i,j}=0$ 时，则 $f'_{i,j}=f_{i,j}$，故等价于下式成立：

$$(Tb_{i,j}\cdot Tc_{i,j})\oplus[(1-Tb_{i,j})\cdot Td_{i,j}]=(Tb'_{i,j}\cdot Tc'_{i,j})\oplus[(1-Tb'_{i,j})\cdot Td'_{i,j}] \quad (3)$$

分析可知，式(3)为线性表达式。故可直接构造等式(3)作为情况 a)的充分条件所对应的线性方程。

对于情况 b)，即 $\Delta f_{i,j}=1$ 或 -1，等价于下式成立：

$$f'_{i,j}=(Tb'_{i,j}\cdot Tc'_{i,j})\oplus[(1-Tb'_{i,j})\cdot Td'_{i,j}]=\frac{(1+\Delta f_{i,j})}{2} \tag{4}$$

$$f_{i,j}=(Tb_{i,j}\cdot Tc_{i,j})\oplus[(1-Tb_{i,j})\cdot Td_{i,j}]=\frac{(1-\Delta f_{i,j})}{2} \tag{5}$$

分析式(4)、式(5)可知，由于 $\Delta B_{i,j}$ 与 $\Delta C_{i,j}\cdot\Delta D_{i,j}$ 至少有一个不为 0，因此式(4)、式(5)均为线性表达式。故可直接构造式(4)、式(5)作为情况 b)的充分条件所对应的线性方程组。

对于情况 c)，即 $\Delta f_{i,j}=1$ 或 -1 时，仍然等价于式(4)、式(5)成立，但由于 $\Delta B_{i,j}=0$ 且 $\Delta C_{i,j}\cdot\Delta D_{i,j}=0$，式(4)、式(5)中至少有一个为非线性表达式。故对于情况 c)，需分以下 3 种条件分别构造线性方程：

1)若 $\Delta C_{i,j}=\Delta f_{i,j}$，则充分条件为 $b_{i,j}=1$，故构造线性方程 $Xb_{i,j}\oplus 1=0$；

2)若 $\Delta D_{i,j}=\Delta f_{i,j}$，则充分条件为 $b_{i,j}=0$，故构造线性方程 $Xb_{i,j}=0$；

3)若 $\Delta C_{i,j}\neq\Delta f_{i,j}$ 且 $\Delta D_{i,j}\neq\Delta f_{i,j}$，则无逻辑函数条件可以满足该比特的减差分值。

对于情况 c)，按上述 3 种分类构造线性方程。

3.3.2 第二轮和第四轮逻辑函数条件的求解

对于 SHA-1 第二轮和第四轮逻辑函数 XOR：$f_{i,j}=b_{i,j}\oplus c_{i,j}\oplus d_{i,j}$，则可对表达式做如下变化：

$$f'_{i,j}=Tb'_{i,j}\oplus Tc'_{i,j}\oplus Td'_{i,j} \tag{6}$$

$$f_{i,j}=Tb_{i,j}\oplus Tc_{i,j}\oplus Td_{i,j} \tag{7}$$

对于第二和第四轮逻辑函数条件的求解，可按 $\Delta f_{i,j}$ 的取值分以下两种情况进行：

a)$\Delta f_{i,j}=0$；

b)$\Delta f_{i,j}=1$ or -1。

对于情况 a)，由式(6)、式(7)知，$\Delta f_{i,j}=0$ 等价于如下表达式成立：

$$Tb_{i,j}\oplus Tc_{i,j}\oplus Td_{i,j}=Tb'_{i,j}\oplus Tc'_{i,j}\oplus Td'_{i,j} \tag{8}$$

化简式(8)，其等价于下式成立：

$\Delta B^2_{i,j}\oplus\Delta C^2_{i,j}\oplus\Delta D^2_{i,j}=0$

因此，对于情况 a)，若 $\Delta B^2_{i,j}\oplus\Delta C^2_{i,j}\oplus\Delta D^2_{i,j}=0$，则 $\Delta f_{i,j}=0$ 恒成立，该比特位无需构造充分条件；若 $\Delta B^2_{i,j}\oplus\Delta C^2_{i,j}\oplus\Delta D^2_{i,j}\neq 0$，则无充分条件可以满足逻辑函数在该比特的差分值。

对于情况 b)，由式(6)、式(7)知，$\Delta f_{i,j}=1$ 或 -1 等价于如下表达式成立：

$$Tb'_{i,j}\oplus Tc'_{i,j}\oplus Td'_{i,j}=\frac{(1+\Delta f_{i,j})}{2} \tag{9}$$

$$Tb_{i,j}\oplus Tc_{i,j}\oplus Td_{i,j}=\frac{(1-\Delta f_{i,j})}{2} \tag{10}$$

易知，式(9)、式(10)均为线性表达式。因此，可直接构造式(9)、式(10)作为情况 b)充分条件所对应的线性方程组。

3.3.3 第三轮逻辑函数条件的求解

对于 SHA-1 第三轮逻辑函数 MAJ：$f_{i,j}=(b_{i,j}\wedge c_{i,j})\vee(b_{i,j}\wedge d_{i,j})\vee(c_{i,j}\wedge d_{i,j})$，可按 $\Delta B_{i,j}$，$\Delta C_{i,j}$，$\Delta D_{i,j}$ 的取值分以下 4 种情况求解：

a)$\Delta B^2_{i,j}+\Delta C^2_{i,j}+\Delta D^2_{i,j}=0$；

b)$\Delta B^2_{i,j}+\Delta C^2_{i,j}+\Delta D^2_{i,j}=1$；

c)$\Delta B^2_{i,j}+\Delta C^2_{i,j}+\Delta D^2_{i,j}=2$；

d)$\Delta B^2_{i,j}+\Delta C^2_{i,j}+\Delta D^2_{i,j}=3$。

对于情况 a)，即 $\Delta B^2_{i,j}+\Delta C^2_{i,j}+\Delta D^2_{i,j}=0$ 时，则易知 $\Delta f_{i,j}=0$ 恒成立。故若 $\Delta f_{i,j}=0$，则无需构造充分条件；若 $\Delta f_{i,j}\neq 0$，则无充分条件可以满足该比特位的差分值。

针对情况 b)，即 $\Delta B^2_{i,j}+\Delta C^2_{i,j}+\Delta D^2_{i,j}=1$ 时，易知逻辑函数 3 个输入参数中仅有一个差分值不为 0。当 $\Delta B^2_{i,j}=1$，$\Delta C^2_{i,j}=0$，$\Delta D^2_{i,j}=0$ 时，则 $\Delta B_{i,j}$，$\Delta f_{i,j}$ 与充分条件三者的关系如表 2 所列。

表 2 $\Delta B_{i,j}$，$\Delta f_{i,j}$ 与充分条件三者的关系

$\Delta B_{i,j}$	$\Delta f_{i,j}$	充分条件	$\Delta B_{i,j}$	$\Delta f_{i,j}$	充分条件
1	1	$c_{i,j}\neq d_{i,j}$	-1	1	不存在
1	-1	不存在	-1	-1	$c_{i,j}\neq d_{i,j}$
1	0	$c_{i,j}=d_{i,j}$	-1	0	$c_{i,j}=d_{i,j}$

观察表 2 易知，当 $\Delta B^2_{i,j}=1$，$\Delta C^2_{i,j}=0$，$\Delta D^2_{i,j}=0$ 时，逻辑函数条件对应的线性方程如下：

$$(H_b\cdot Xc_{i,j})\oplus(H_b\cdot Xd_{i,j})=\Delta f^2_{i,j} \tag{11}$$

其中，$H_b=1+\Delta f^2\varphi_{i,j}\cdot(\frac{3+\Delta B_{i,j}\cdot\Delta f_{i,j}}{2})$。若方程(11)无解，则表示无充分条件可以满足该比特位的差分值。

同理可得：当 $\Delta B^2_{i,j}=0$，$\Delta C^2_{i,j}=1$，$\Delta D^2_{i,j}=0$ 时，逻辑函数条件对应的线性方程为：

$$(H_c\cdot Xb_{i,j})\oplus(H_c\cdot Xd_{i,j})=\Delta f^2_{i,j} \tag{12}$$

其中，$H_c=1+\Delta f^2_{i,j}\cdot(\frac{3+\Delta C_{i,j}\cdot\Delta f_{i,j}}{2})$。

当 $\Delta B^2_{i,j}=0$，$\Delta C^2_{i,j}=0$，$\Delta D^2_{i,j}=1$ 时，逻辑函数条件对应的线性方程为：

$$(H_d\cdot Xb_{i,j})\oplus(H_d\cdot Xc_{i,j})=\Delta f^2_{i,j} \tag{13}$$

其中，$H_d=1+\Delta f^2_{i,j}\cdot(\frac{3+\Delta D_{i,j}\cdot\Delta f_{i,j}}{2})$。若方程(12)或(13)无解，则表示无充分条件可以满足该比特位的差分值。

综合式(11)－式(13)可得出结论：对于情况 b)，构造如下线性方程：

$$(H_1\cdot Xb_{i,j})\oplus(H_2\cdot Xc_{i,j})\oplus(H_3\cdot Xd_{i,j})=\Delta f^2_{i,j} \tag{14}$$

其中，$H_1=\Delta C^2_{i,j}\cdot H_c+\Delta D^2_{i,j}\cdot H_d$；$H_2=\Delta B^2_{i,j}\cdot H_b+\Delta D^2_{i,j}\cdot H_d$；$H_3=\Delta B^2_{i,j}\cdot H_b+\Delta C^2_{i,j}\cdot H_c$。

针对情况 c)，即 $\Delta B^2_{i,j}+\Delta C^2_{i,j}+\Delta D^2_{i,j}=2$ 时，则逻辑函数的 3 个输入参数中有两个差分值不为 0。当 $\Delta B^2_{i,j}=1$，$\Delta C^2_{i,j}=1$，$\Delta D^2_{i,j}=0$ 时，则 $\Delta f_{i,j}$，$\Delta B_{i,j}+\Delta C_{i,j}$ 与充分条件的关系如表 3 所列。

表 3 $\Delta B_{i,j}+\Delta C_{i,j}$，$\Delta f_{i,j}$ 与充分条件三者的关系

$\Delta B_{i,j}+\Delta C_{i,j}$	$\Delta f_{i,j}$	充分条件	$\Delta B_{i,j}+\Delta C_{i,j}$	$\Delta f_{i,j}$	充分条件
2	1	恒成立	0	-1	不存在
2	0	不存在	-2	1	不存在
2	-1	不存在	-2	0	不存在
0	1	不存在	-2	-1	恒成立
0	0	恒成立			

观察表 3 不难看出，表中充分条件由 $\Delta f_{i,j}$ 与 $\Delta B_{i,j}+\Delta C_{i,j}$ 共同决定。令：$S1_{i,j}=\frac{\Delta B_{i,j}+\Delta C_{i,j}}{2}$，$S2_{i,j}=\frac{\Delta B_{i,j}+\Delta D_{i,j}}{2}$，

$S3_{i,j}=\frac{\Delta C_{i,j}+\Delta D_{i,j}}{2}$。

由表 3 可以得出结论：当 $\Delta B_{i,j}^2=1,\Delta C_{i,j}^2=1,\Delta D_{i,j}^2=0$ 时，若 $\Delta f_{i,j}=S1_{i,j}$，则无需构造充分条件；若 $\Delta f_{i,j}\neq S1_{i,j}$，则不存在符合要求的充分条件。同理可得：当 $\Delta B_{i,j}^2=1,\Delta C_{i,j}^2=0,\Delta D_{i,j}^2=1$ 时，若 $\Delta f_{i,j}=S2_{i,j}$，则无需构造充分条件；若 $\Delta f_{i,j}\neq S2_{i,j}$，则不存在符合要求的充分条件。当 $\Delta B_{i,j}^2=0,\Delta C_{i,j}^2=1,\Delta D_{i,j}^2=1$ 时，若 $\Delta f_{i,j}=S3_{i,j}$，则无需构造充分条件；若 $\Delta f_{i,j}\neq S3_{i,j}$，则不存在符合要求的充分条件。

基于上述描述，可得出结论：对于情况 c)，若 $\Delta f_{i,j}=J1_{i,j}$，则无需构造充分条件；若 $\Delta f_{i,j}\neq J1_{i,j}$，则无充分条件可以满足该比特位的差分值。其中：

$$J1_{i,j}=\frac{\Delta B_{i,j}+\Delta C_{i,j}+\Delta D_{i,j}}{2}$$

针对情况 d)，即 $\Delta B_{i,j}^2+\Delta C_{i,j}^2+\Delta D_{i,j}^2=3$ 时，逻辑函数输入参数的差分值均不为 0。充分条件、$\Delta f_{i,j}$ 与 $\Delta B_{i,j}+\Delta C_{i,j}+\Delta D_{i,j}$ 的关系如表 4 所列。

表 4　$\Delta B_{i,j}+\Delta C_{i,j}+\Delta D_{i,j}$，$\Delta f_{i,j}$ 与充分条件三者的关系

$\Delta B_{i,j}+\Delta C_{i,j}+\Delta D_{i,j}$	$\Delta f_{i,j}$	充分条件	$\Delta B_{i,j}+\Delta C_{i,j}+\Delta D_{i,j}$	$\Delta f_{i,j}$	充分条件
3	1	恒成立	−1	1	不存在
3	0	不存在	−1	0	不存在
3	−1	不存在	−1	−1	恒成立
1	1	恒成立	−3	1	不存在
1	0	不存在	−3	0	不存在
1	−1	不存在	−3	−1	恒成立

由表 4 可得出结论：对于情况 d)，若 $\Delta f_{i,j}=J2_{i,j}$，则无需构造充分条件；若 $\Delta f_{i,j}\neq J2_{i,j}$，则无充分条件可以满足该比特位的差分值。其中：

$$J2_{i,j}=\frac{(\Delta B_{i,j}+\Delta C_{i,j}+\Delta D_{i,j})}{\sqrt{(\Delta B_{i,j}+\Delta C_{i,j}+\Delta D_{i,j})^2}}$$

3.4　逻辑函数条件的判定

在求解逻辑函数条件的过程中，针对每一比特位，每求得一组逻辑函数条件，都需要判断是否与已求得的充分条件矛盾或重复。本文算法将所有的充分条件转化为 F_2 上线性方程组的形式，对于某一比特位的逻辑函数条件，若方程组无解，则表示所求得的充分条件之间存在矛盾，需要对该步逻辑函数的减差分值做进位扩展操作以改变逻辑函数的差分，进而求得一组新的逻辑函数条件，反之则说明不存在矛盾；若该比特位及其之前求得的线性方程组与上一比特位及其之前的线性方程组等价，则表示该逻辑函数条件与已求得的某些充分条件等价，故需删除该条件，反之则保留。因此，逻辑函数条件的判定转变为了线性方程组解的判定问题。

令 $n(i,j)$ 表示第 i 步第 j 比特及其之前构造的方程组，根据线性方程组解的判定定理，对第 i 步第 j 比特求得的逻辑函数条件做如下两次判定：如果 $DET(D_{n(i,j)})\neq DET(\overline{D_{n(i,j)}})$，则表示该比特位的逻辑函数条件与已求得的充分条件矛盾，反之则不矛盾。当 $j\neq 0$ 时，$DET(\overline{D_{n(i,j-1)}})=DET(\overline{D_{n(i,j)}})$，或当 $j=0$ 时，$DET(\overline{D_{n(i,0)}})=DET(\overline{D_{n(i-1,31)}})$，则表示该比特位的逻辑函数条件已满足，需舍弃该条件，反之则保留。

结合逻辑函数条件判定算法，逻辑函数条件求解过程如算法 2 所示。

算法 2　逻辑函数条件求解算法

输入：逻辑函数在第 i 步第 j 比特位输入参数的减差分 $\Delta B_{i,j}$，$\Delta C_{i,j}$，$\Delta D_{i,j}$

输出：第 i 步第 j 比特逻辑函数条件对应的线性方程

步骤 1：如果为第一轮逻辑函数 IF，即 $1\leqslant i\leqslant 20$：

a)若 $\Delta f_{i,j}=0$，则构造线性方程(3)；

b)若 $\Delta f_{i,j}=1$ or -1，且 $\Delta B_{i,j}$ 与 $\Delta C_{i,j}\cdot\Delta D_{i,j}$ 至少有一个不为 0，构造线性方程组(4)、(5)；

c)若 $\Delta f_{i,j}=1$ or -1，$\Delta B_{i,j}=0$，$\Delta C_{i,j}\cdot\Delta D_{i,j}=0$，则按 3.3.1 节中描述，分 3 种情况判断，构造线性方程或返回无解。

步骤 2：如果为第二轮或第四轮逻辑函数 XOR，即 $21\leqslant i\leqslant 40$ 或 $61\leqslant i\leqslant 80$：

a)若 $\Delta f_{i,j}=0$，则当 $\Delta B_{i,j}^2\oplus\Delta C_{i,j}^2\oplus\Delta D_{i,j}^2\neq 0$ 时，返回无解；

b)若 $\Delta f_{i,j}=1$ or -1，则构造方程组(9)、(10)。

步骤 3：如果为第三轮逻辑函数 MAJ，即 $41\leqslant i\leqslant 60$：

a)若 $\Delta B_{i,j}^2+\Delta C_{i,j}^2+\Delta D_{i,j}^2=0$，则当 $\Delta f_{i,j}\neq 0$ 时，返回无解；

b)若 $\Delta B_{i,j}^2+\Delta C_{i,j}^2+\Delta D_{i,j}^2=1$，则构造方程(14)；

c)若 $\Delta B_{i,j}^2+\Delta C_{i,j}^2+\Delta D_{i,j}^2=2$，则当 $\Delta f_{i,j}\neq J1_{i,j}$ 时，返回无解；

d)若 $\Delta B_{i,j}^2+\Delta C_{i,j}^2+\Delta D_{i,j}^2=3$，则当 $\Delta f_{i,j}\neq J2_{i,j}$ 时，返回无解。

步骤 4：若前 3 步返回无解，则对逻辑函数差分做进位扩展操作。返回步骤 1，重新求解该比特的逻辑函数条件。

步骤 5：若 $DET(D_{n(i,j)})\neq DET(\overline{D_{n(i,j)}})$，则对逻辑函数差分做进位扩展操作，返回步骤 1，重新求解该比特的逻辑函数条件。

步骤 6：若 $DET(\overline{D_{n(i,j-1)}})=DET(\overline{D_{n(i,j)}})$(当 j=0 时，若 $DET(\overline{D_{n(i,0)}})=DET(\overline{D_{n(i-1,31)}})$)，则舍弃该逻辑函数条件。

步骤 7：输出第 i 步第 j 比特逻辑函数条件对应的线性方程。

结束语　由差分路径求解充分条件是模差分攻击技术的关键步骤之一。本文通过将充分条件的求解转化为 F_2 上线性方程组的构造过程，利用线性方程组解的判定定理判断每步充分条件求解的正确性，首次提出了针对 SHA-1 模差分攻击的充分条件自动化求解算法。针对王小云在文献[1]中所给出的差分路径，本文利用充分条件自动化求解算法对该差分路径的充分条件进行了求解，算法运行速度较快，且运行结果与手动推导结果相一致，实验结果验证了本文算法的正确性。本文算法做适当变形后，同样适用于与 SHA-1 结构相似的 Hash 函数充分条件的自动化求解。

参 考 文 献

[1] Wang Xiao-yun, L Yi-qun, Yu Hong-bo. Finding collisions in the full SHA-1[M]// Advance in Cryptology-CRYPTO 2005. Berlin Heidelberg: Springer-Verlag, 2005: 17-36

[2] McDonald C, Hawkes P, Pieprzyk J. Differential Path for SHA-1 with complexity O(2^{52}): Report 2009, 259[R/OL]. Cryptology ePrint Archive, http://eprint.iacr.org/2009/259

[3] Chen R. New Techniques for Cryptanalysis of Cryptographic Hash Functions [D]. Technion: Technion-Israel Institute of Technology, 2011

[4] Stevens M. Attacks on Hash Functions and Applications [D]. Holland: Leiden University, 2012

[5] Stevens M. New Collision Attacks on SHA-1 Based on Optimal Joint Local-Collision Analysis [M] // Advance in Cryptology-CRYPTO 2005. Berlin Heidelberg: Springer-Verlag, 2013: 245-261

[6] Biham E, Chen R, Joux A. Cryptanalysis of SHA-0 and Reduced SHA-1[J]. Journal of Cryptology, 2014, 28(1): 110-160

(下转第 147 页)

加，且查询时间趋于平稳。当 $k=10$ 时，查询 1min 内的 Top-K 关键字需要 1.2s，查询 8min 内的关键字则需要 1.4s。查询时间先是增加而后逐渐平稳，原因在于算法执行初期，生成的检查点数量较多，查询时需要合并的代价较大，但随着检查点数量的平稳，查询的时间也趋于稳定。

结束语 本文研究分布式环境下流数据 Top-K 关键字查询的问题，由于海量的数据和未知庞大的关键字集合，使得现有的算法精度较低。在 Spark Streaming 框架的基础上提出了分布式 Top-K 关键字查询算法 TSTop-K，该算法采用动态 Summary 的更新策略，通过设置检查点的方式，提出相应的合并方法，使得在关键字集合未知时 Top-K 查询结果具有较高的精度。实验表明 TSTop-K 算法能很好地支持实时性强、精度要求高的 Top-K 关键字查询，具有实用价值。在性能对比上，TSTop-K 算法能动态调整存储空间的大小，使之性能和精度高于经典的算法。下一步工作将研究 Top-K 时空关键字查询算法。

参 考 文 献

[1] Chen L, Cong G, Cao X, et al. Temporal spatial-keyword top-k publish/subscribe[C]// 2015 IEEE 31st International Conference on Data Engineering. IEEE, 2015: 255-266

[2] Zheng K, Su H, Zheng B, et al. Interactive top-k spatial keyword queries[C]// 2015 IEEE 31st International Conference on Data Engineering. IEEE, 2015: 423-434

[3] Charikar M, Chen K, Farach-Colton M. Finding Frequent Items in Data Streams[J]. Theoretical Computer Science, 2004, 312(1): 1530-1541

[4] Metwally A, Agrawal D, Abbadi A E. Efficient Computation of Frequent and Top-k Elements in Data Streams[C]// International Conference on Database Theory. Springer-Verlag, 2005: 398-412

[5] Zaharia M, Das T, Li H, et al. Discretized streams: fault-tolerant streaming computation at scale[C]// Twenty-Fourth ACM Symposium on Operating Systems Principles. 2013: 423-438

[6] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Commun. ACM (CACM), 2008, 51(1): 107-113

[7] Ci Xiang, Ma You-zhong, Meng Xiao-feng. Method for top-K query on big data in cloud[J]. Journal of Software, 2014, 25(4): 813-825(in Chinese)
慈祥，马友忠，孟小峰. 一种云环境下的大数据 Top-K 查询方法[J]. 软件学报，2014，25(4)：813-825

[8] Song Jie, Hao Wen-ning, Chen Gang, et al. Research of Distributed ETL Architecture Based on MapReduce[J]. Computer Science, 2013, 40(6): 152-154(in Chinese)
宋杰，郝文宁，陈刚，等. 基于 MapReduce 的分布式 ETL 体系结构研究[J]. 计算机科学，2013，40(6)：152-154

[9] Manku G, Motwani R. Approximate frequency counts over data streams[C]// Proceedings of the 28th International Conference on Very Large Data Bases. 2002: 346-357

[10] Demaine E D, Lopez-Ortiz A, Munro J I. Frequency estimation of internet packet streams with limited space[C]// Proceedings of the 10th Annual European Symposium on Algorithms. 2002: 348-360

[11] Cormode G, Muthukrishnan S. What's hot and what's not: tracking most frequent items dynamically[J]. TODS, 2005, 30(1): 249-278

[12] Estan C, Varghese G. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice[J]. ACM Trans. Comput. Syst., 2003, 21(3): 270-313

[13] Manerikar N, Palpanas T. Frequent items in streaming data: An experimental evaluation of the state-of-the-art[J]. DKE, 2009, 68(4): 415-430

[14] Agarwal P K, Cormode G, Huang Z, et al. Mergeable summaries[C]// PODS. 2012: 23-34

[15] Frequent Itemset Mining Dataset Repository, University of Helsinki, 2008[OL]. http://fimi.cs.helsinki.fi/data

[16] Kim J M, Park Y T. Scalable OWL-Horst ontology reasoning using SPARK[C]// International Conference on Big Data and Smart Computing. IEEE, 2015: 79-86

[17] Armbrust M, Xin R S, Lian C, et al. Spark sql: Relational data processing in spark[C]// Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015: 1383-1394

[18] Lin C Y, Tsai C H, Lee C P, et al. Large-scale logistic regression and linear support vector machines using Spark[C]// 2014 IEEE International Conference on Big Data. IEEE, 2014: 519-528

[19] Akgün B. Streaming Linear Regression on Spark MLlib and MOA[C]// Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015. ACM, 2015: 1244-1247

(上接第 127 页)

[7] Cannière C D, Rechberger C. Finding SHA-1 Characteristics: General Results and Applications [M]// Advances in Cryptology-ASIACRYPT 2006. Berlin Heidelberg: Springer-Verlag, 2006: 1-20

[8] Cannière C D, Mendel F, Rechberger C. Collisions for 70-Step SHA-1: On the Full Cost of Collision Search [M]// Selected Areas in Cryptography. Berlin Heidelberg: Springer-Verlag, 2007: 56-73

[9] Grechnikov E A. Collisions for 72-step and 73-step SHA-1: Improvements in the Method of Characteristics: Report 2010, 413 [R/OL]. Cryptology ePrint Archive, http://eprint.iacr.org/2010/413.pdf

[10] Grechnikov E A, Adinetz A V. Collision for 75-step SHA-1: Intensive Parallelization with GPU: Report 2011, 641 [R/OL]. Cryptology ePrint Archive, http://eprint.iacr.org/2011/641

[11] Adinetz A V, Grechnikov E A. Building a collision for 75-round reduced SHA-1 Using GPU Clusters[M]// Euro-Par 2012 Parallel Processing. Berlin Heidelberg: Springer-Verlag, 2012: 933-944

[12] Sugita M, Kawazoe M, Perret L, et al. Algebraic Cryptanalysis of 58-Round SHA-1[M]// Fast Software Encryption. Berlin Heidelberg: Springer-Verlag, 2007: 349-365

[13] Pramstaller N, Rechberger C, Rijimen V. Exploiting Coding Theory for Collision Attacks on SHA-1[M]// Cryptography and Coding. Berlin Heidelberg: Springer-Verlag, 2005: 78-95

[14] Joux A, Peyrin T. Hash Functions and the (Amplified) Boomerang Attack[M]// Advances in Cryptology-CRYPTO 2007. Berlin Heidelberg: Springer-Verlag, 2007: 244-263