寻找 Hash 碰撞，基于 [1]

方理楠

November 2020

1 Hash

哈希函数可以将任意比特长度的消息压缩为固定长度的比特串，该比特串通常称为散列值（Hash Value）。散列值生成过程可表示为

$$h = H(M)$$

其中，M 是一个变长消息，H 是 Hash 函数，h 是定长的散列值。哈希函数主要应用于消息认证和数字签名。

2 抗碰撞性

哈希函数应具有如下性质：

- 抗弱碰撞性
  对任何给定的消息 x, 找到满足 $y \neq x$且$H(x) = H(y)$ 的消息 y 在计算上是不可行的。
- 抗强碰撞性
  找到任何满足 $H(x) = H(y)$ 的偶对 x, y 在计算上是不可行的。

抗强碰撞性的哈希函数比抗弱碰撞性的哈希函数的安全性要高，即满足抗强碰撞性的哈希函数肯定满足抗弱碰撞性，反之不成立。

3 Hash 函数结构

Hash 函数的一般结构如图 1 所示 [2]，称为 MD 结构。其中，f 表示压缩函数，IV 是初始值，y_i 称为链接变量，m_i 是消息分组，即将一个消

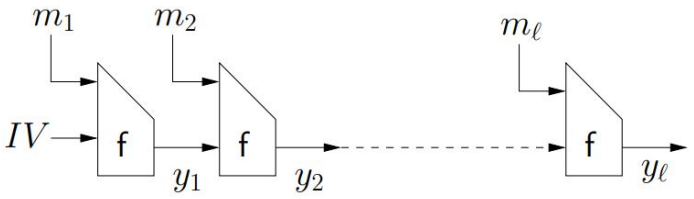

图 1: Merkle–Damgård construction

息划分为若干个等长的子消息，如果不能正好分割，则进行消息填充，使最后划分的消息分组都有相同的长度。f 有两个输入值，一个输出值，主流的哈希算法 f 中都有非线性函数。最后生成的链接变量 y_l 作为散列值。

4 碰撞构造方法

(1) 首先引入差分的概念，差分可以分为两类：

- 异或差分
  设 X 和 $X^{'}$ 为 32 位比特串，异或差分 $= X \bigoplus X^{'}$。例如 $X = 0\ldots011$，$X^{'} = 0\ldots010$，异或结果为 $0\ldots001$, 我们就说 $0\ldots001$ 为 X 和 $X^{'}$ 的异或差分。
- 模差分
  设 X 和 $X^{'}$ 为 32 位比特串，令 $result = (X - X^{'}) \bmod \mathrm{n}$，我们就说 $result$ 为 X 和 $X^{'}$ 的模差分，记为 ΔX。

(2) 对任意两个 l 比特长的消息 M 和 $M^{'}$，$M = (M_0, M_1, \ldots, M_{K-1})$,$M^{'} = (M_0^{'}, M_1^{'}, \ldots, M_{K-1}^{'})$，Hash 函数中引入差分如下：

$$\Delta H_0 \xrightarrow{(M_0, M_0^{'})} \Delta H_1 \xrightarrow{(M_1, M_1^{'})} \Delta H_2 \xrightarrow{(M_2, M_2^{'})} \ldots \Delta H_{K-1} \xrightarrow{(M_{K-1}, M_{K-1}^{'})} \Delta H$$

其中，ΔH_0 是初始值的模差分并且 $\Delta H_0 = 0$，ΔH_i 是第 i 轮后生成的链接变量的模差分。ΔH 是两个散列值的模差分，显然，如果 $\Delta H = 0$，我们就说 M 和 $M^{'}$ 产生了碰撞，称 $\{\Delta H_0, \Delta H_1, \ldots, \Delta H_{K-1}, \Delta H\}$ 为差分路径。通过消除输入的差分，得到 $\Delta H = 0$。

(3) 构造可以产生碰撞的消息 M 和 $M^{'}$[1]，步骤如下：

Step 1 寻找 ΔM（$\Delta M = M - M^1$）以及差分路径，使得 $P(\Delta H = 0)$接近于1，P 表示事件发生的概率。

Step 2 给定 ΔM，推算出可以使差分路径成立的充分条件。

Step 3 ΔM 不变，修改 M（$M^{'} = M + \Delta M$）使得推算出的充分条件有比较高的概率可以实现。

5 符号说明

令 x_i 表示第 i 轮产生的链接变量，$x_{i,j}(j \geq 1)$ 表示 x_i 的第 j 个比特值。$x_i[j]$ 表示 $x_{i,j}^{'} - x_{i,j} = 1$，$x_i[-j]$ 表示 $x_{i,j}^{'} - x_{i,j} = -1$。

6 step-2 方法概括 [3]

对于 ΔH_i，假设压缩函数有 64 步，那么从第 64 步开始倒推差分成立的充分条件。

(1) 构造充分条件

充分条件可以分成两类：

- 控制链接变量中借位 (carry) 的长度
  例如，对于 $\Delta x_2 = 2^6$，如果是 $x_2[7]$，则不会产生借位；如果是 $x_2[-7]$，即 $x_{2,7}^{'} = 0$，$x_{2,7} = 1$。那么第 7 位就会向 x 的更高位借位，这种借位对于模加运算没有影响，但会对压缩函数中的非线性函数产生影响。所以，如果想要阻止差分的扩散，便令 $x_2[7]$；相反，如果想要扩散差分，便令 $x_2[-7]$，并且可以扩散至任意位，如 $x_2[-7, -8, \ldots, +n]$。因此，构建充分条件时，首先停止差分扩散，当有条件互相矛盾时，选择扩散差分，具体扩散多少，视情况而定。
- 控制非线性函数 ϕ 的输出值 (常量不能控制)
  假设链接变量 x_1 的差分是 $x_1[-26, \pm 32], x_2$ 的差分是 $x_2[-26, \pm 32]$，x_3 的差分是 $x_3[\pm 32]$，$\phi(x_1, x_2, x_3) = (x_1 \vee \neg x_3) \bigoplus x_2$ 并且 $\Delta\phi = \pm 2^{31}$。首先我们要令 $\Delta\phi_{,26} = 0$, 所有的变化可由表 1 表示：从表 1 可以知道，$x_{3,26} = 1$ 时，可得 $\Delta\phi_{,26} = 0$。同理可得 $\Delta\phi_{,32} = \pm 1$ 的充分条件为 $x_{1,32} = x_{3,32}$。

x_1, x_2, x_3	$\phi(x_1, x_2, x_3)$	$\phi^{'} = \phi(\neg x_1, \neg x_2, x_3)$	$\Delta\phi$
0,0,0	1	0	-1
0,0,1	0	0	0
0,1,0	0	1	1
0,1,1	1	1	0
1,0,0	1	0	-1
1,0,1	1	1	0
1,1,0	0	1	1
1,1,1	0	0	0

表 1: $\Delta\phi_{,26}$

(2) 避免矛盾的产生

有两种方法来避免产生互相矛盾的充分条件:

- 扩展借位
  当需要 $\Delta\phi_{i,j} \neq 0$，但是输入 ϕ 的链接变量第 j 位都为 0 时，不可能生成所需要的差分。这时，需要从小于 j 并最接近 j 的差分处扩展借位直到 j。
  如 $\Delta\phi_i = -2^{19}$，输入的链接变量值为 $x_i[5]$ $y_i[-16]$ $z_i[10, 11, -12]$，最接近 -2^{19} 的是 $y_i[-16]$,所以进行借位扩展成 $y_i[16, 17, 18, 19, -20]$。
- 改变 $\Delta\phi$
  $\Delta\phi$ 的值是不变的，但是表示形式可以变化。例如 $\Delta\phi = 2^{20}$，可以表示成 $\Delta\phi = -2^{20} + 2^{21}$，或者 $\Delta\phi = -2^{20} - 2^{21} + 2^{22}$。

7 step-3 方法概括

消息修改通过修改 M 提前满足一些充分条件，减少搜索能产生碰撞的 M，$M^{'}(M^{'} = M + \Delta M)$ 的时间。如果一个随机消息不能满足所有条件，则重新选择一个随机消息，同样进行消息修改，使得提前满足一部分条件，然后观察修改后的消息是否满足所有充分条件。

消息修改方法有两种:

- 单个消息修改
  仅修改单个消息，以满足压缩函数的第一轮中的充分条件。例如，[1]

中，关于 $c1$ 有三个充分条件：$c_{1,8}=0, c_{1,12}=0, c_{1,20}=0$。$c_1$ 由下式计算得到：

$$c_1 = (d_1 + (c_0 + \phi(d_1, a_1, b_0) + m_2 + t) \lll 17) \bmod 2^{31}$$

以如下方式修改 m_2[4]:

a. 生成一个随机 32 比特消息 m_2^{old}，并计算 c_1^{old}。

b. 计算 c_1^{new}。

$$c_1^{new} \longleftarrow c_1^{old} - c_{1,7}^{old} \cdot 2^6 - c_{1,12}^{old} \cdot 2^{11} - c_{1,20}^{old} \cdot 2^{19}$$

c. 修改 m_2 使得 c_1^{new} 恒成立。

$$m_2^{new} \longleftarrow ((c_1^{new} - d_1) \ggg 17) - c_0 - \phi(d_1, a_1, b_0) - t$$

- 多消息修改
  多消息修改用于满足压缩函数第二轮中的充分条件，。[1] 中，a_5 有充分条件 $a_5^{32}=0$，如果 $a_{5,32}=1$，则需要进行修改，修改方式如下：

$$a_5 = (b_4 + (a_4 + \phi(b_4, c_4, d_4) + m_1 + t) \lll 5) \bmod 2^{31}$$

a. 因为循环左移 5 位，所以修改 $m_{1,27}$。

b. 因为 m_1 在第一轮中也同样参与计算，m_1 的改变会影响第一轮中后面的运算，所以还要做额外的修改。

修改细节如下：

1) 修改 m_1（step 2）
   $m_1 \longleftarrow m_1 + 2^{26}$
   那么：

$$d_1^{new} = (a_1 + (d_0 + \phi(a_1, b_0, c_0) + m_1 + t) \lll 12) \bmod 2^{31}$$

2) 修改 m_2（step 3）

$$c_1' \longleftarrow (d_1^{new} + (c_0 + \phi(d_1^{new}, a_1, b_0) + m_2 + t) \lll 17) \bmod 2^{31}$$

   要使 c_1 保持不变，则：

$$m_2 \longleftarrow ((c_1 - d_1^{new}) \ggg 17) - c_0 - \phi(d_1^{new}, a_1, b_0) - t$$

3) 修改 m_3, m_4, m_5（step 4，5，6）
同理，继续修改消息，直到 d_1^{new} 的影响消失。

$$m_3 \longleftarrow ((b_1 - c_1) \ggg 22) - b_0 - \phi(c_1, d_1^{new}, a_1) - t$$

$$m_4 \longleftarrow ((a_2 - b_1) \ggg 7) - a_1 - \phi(b_1, c_1, d_1^{new}) - t$$

$$m_5 \longleftarrow ((d_2 - a_2) \ggg 12) - d_1^{new} - \phi(a_2, b_1, c_1) - t$$

参考文献

[1] Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In *Advances in Cryptology –EUROCRYPT 2005*, volume 3494, pages 19–35. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. Series Title: Lecture Notes in Computer Science.

[2] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. In *Advances in Cryptology –CRYPTO 2005*, volume 3621, pages 430–448. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. Series Title: Lecture Notes in Computer Science.

[3] Yu Sasaki, Yusuke Naito, Jun Yajima, Takeshi Shimoyama, Noboru Kunihiro, and Kazuo Ohta. How to Construct Sufficient Condition in Searching Collisions of MD5. page 9.

[4] Yu Sasaki, Yusuke Naito, Noboru Kunihiro, and Kazuo Ohta. Improved collision attack on md5.