哈希函数的一个设计原理

Ivan Bjerre Damgård

摘要

我们证明，如果存在一个从 m 比特到 t 比特（其中 m > t）的计算上抗碰撞的函数 f，那么存在一个将任意多项式长度的消息映射到 t 比特串的计算上抗碰撞的函数 h。

令 n 为消息的长度。h 的构造可以有两种方式：一种是使其能够在 1 个处理器上以线性于 n 的时间被求值，另一种是使其能够在 O(n) 个处理器上以 O(log(n)) 的时间被求值（其中对 f 的求值算作一步）。最后，对于任意常数 k 和较大的 n，使用 k 个处理器可以获得相对于第一种构造 k 倍的加速。

除了提出了一个普遍合理的哈希函数设计原理，我们的结果为几个先前提出的、看似不相关的哈希函数构造提供了一个统一的视角。它也提示了对其他已提出构造的修改，以使安全性证明可能更容易。

我们给出了三个具体的构造示例，分别基于模平方、基于 Wolfram 的伪随机比特生成器 [Wo] 以及基于背包问题。

1 引言及相关工作

一个哈希函数 h 被称为抗碰撞的，如果它将任意长度的消息映射到某个固定长度的串，但要找到满足 h(x) = h(y) 的 x, y 是一个困难问题。注意，我们这里专注于公开可计算的哈希函数，即不受密钥控制的函数。

在确保数据完整性以及用于数字签名方案时，对这类函数的需求是众所周知的——例如参见 [Da], [De], [DP]。

已经提出了多种哈希函数构造，例如基于 DES [Wi], [DP] 或基于 RSA [DP], [Gir]。[Da] 中的构造是第一个可以证明其抗碰撞性的构造，前提是其所基于的原子操作是安全的——在那种情况下是模平方的单向性，更一般地，是无爪置换对的存在性。后来的一个例子是 [Gi]。不幸的是，这个良好的理论特性意味着与其他方案相比效率有所降低：这些函数所需的时间大致相当于将 RSA 应用于整个消息。

因此，构造可证明抗碰撞且快速的哈希函数的问题仍然开放。

为已知构造提供证明的许多困难源于这样一个事实：随着哈希处理的消息长度增加，情况似乎变得更加复杂。另一方面，如果我们不能对任意长度的消息进行哈希，哈希函数就毫无用处。

在本文中，我们证明，如果使用正确的构造，这个困难是可以被消除的：结果表明，能够以抗碰撞的方式仅从消息长度中“切掉”1 比特，就意味着能够对任意长度的消息进行哈希。该证明提出了一个基本合理的设计原理，可作为设计新哈希函数和修订现有哈希函数的指导。

我们的构造与 Merkle 独立发现的“元方法”非常相似 [Me]；相比之下，本构造包含几个额外的要素，使得无需对函数参数进行任何额外假设即可进行形式化证明。

与 Naor 和 Yung 独立工作的 [NaYu] 中使用的方法也有相似之处。他们也证明了固定大小的哈希函数可以组合起来以实现对任意多项式长度消息的压缩。这既针对我们这种类型的哈希函数，也针对具有稍弱属性的哈希函数：允许敌方选择 x，然后给予他从哈希函数族中随机选择的一个实例。现在他无法在多项式时间内找到另一个满足 f(x) = f(y) 的 y。Naor 和 Yung 从任何单向置换构造了这种类型的函数，并用它们来构建数字签名方案。

从实用的角度来看，我们的构造更高效和直接。这是因为 [NaYu] 为了让他们的构造适用于具有较弱属性的哈希函数，必须为要处理的每个消息块选择一个新的、独立的固定长度哈希函数实例。

最后，应该提到 Impagliazzo 和 Naor 最近的一些独立工作 [ImNa]。他们证明，如果所使用的背包诱导出一个单向函数，那么以与本文第 4.3 节相同方式从背包问题构造的哈希函数在 [NaYu] 的意义上是安全的。

2 预备知识

我们首先定义抗碰撞函数族的概念：

定义 2.1

一个固定大小的抗碰撞哈希函数族 $\mathcal{F}$ 是一个无限的有限集族 $\{F_m\}_{m=1}^{\infty}$，以及一个函数 $t: \mathbf{N} \to \mathbf{N}$，使得对所有 $m \in \mathbf{N}$ 有 $t(m) < m$。

F_{m} 的一个成员是一个函数 $f: \{0,1\}^{m} \to \{0,1\}^{t(m)}$，称为 $\mathcal{F}$ 的大小为 m 的实例。

$\mathcal{F}$ 必须满足以下条件：

1. 存在一个（关于 m 的）概率多项式时间算法 Θ，给定一个 m 值，随机选择 $\mathcal{F}$ 的一个大小为 m 的实例。
2. 对于任何实例 $f \in F_m$ 和 $x \in \{0,1\}^m$，$f(x)$ 可以在多项式时间内计算。
3. 给定一个如 1) 中随机选择的实例 $f \in \mathcal{F}$，很难找到 $x, y \in \{0,1\}^m$，使得 $f(x) = f(y)$ 且 $x \neq y$。

更形式化地：对于任何（关于 m 的）概率多项式时间算法 Δ，以及任何多项式 P，考虑大小为 m 的实例 f 的子集，对于这些实例，Δ 以至少 $1 / P(m)$ 的概率输出 $x \neq y$ 使得 $f(x) = f(y)$。令 $\epsilon(m)$ 为 Θ 选择这些实例之一的概率。那么作为 m 的函数，$\epsilon(m)$ 比任何多项式分数消失得更快。

条件 1) 和 2) 说明 $\mathcal{F}$ 在实践中是有用的：可以高效地选择实例和计算函数值。3) 陈述了抗碰撞属性。

3) 的一个基本含义是 $\mathcal{F}$ 中的函数具有某种单向性：

引理 2.1

令 $\mathcal{F}$ 为一个抗碰撞函数族，f 为一个大小为 m 的实例。令 P_{f} 为通过在 $\{0,1\}^m$ 中随机均匀选择 x 并输出 $f(x)$ 而在 $\{0,1\}^{t(m)}$ 上生成的概率分布。

那么，任何对根据 P_f 选择的像进行 f 求逆的算法，其成功概率都不大于 $1/2 + 1/P(m)$，对于任何多项式 P。

如果 P_f 是 f 的像上的均匀分布，或者如果 $m - t$ 是 $O(m)$，那么任何求逆算法的成功概率都不大于 $1 / P(m)$。

证明

假设引理为假。给定一个算法 Δ，它以至少 $1/2 + 1/P(m)$ 的概率对 f 求逆，我们均匀选择 x 并将 $f(x)$ 作为输入给 Δ。如果 Δ 成功，我们得到一个 y，使得 $f(x) = f(y)$。令 A 为 $f(x)$ 的原像大小至少为 2 的事件。$\{0,1\}^m$ 至少是 f 的像的两倍大，因此 A 以大于 $1/2$ 的概率发生。因此，根据对 Δ 的假设，当 A 发生时，它以至少 $1/P(m)$ 的概率成功。显然，Δ 选择给出 $f(x)$ 的原像中的哪个元素与 x 的选择（在给定 $f(x)$ 的情况下）不相关。因此，给定 A 发生且 Δ 成功，$x \neq y$ 的概率至少为 $1/2$。

对于第二个陈述，注意如果 P_f 是均匀分布，那么 Δ 的成功与 A 的发生不相关。并且如果 $m - t = O(m)$，那么 A 以压倒性概率发生。在任何一种情况下，Δ 基本上以 $1 / P(m)$ 的概率给我们一个碰撞。

最后，我们定义抗碰撞哈希函数族的概念：

定义 2.2

一个抗碰撞哈希函数族 $\mathcal{H}$ 是一个无限的有限集族 $\{H_m\}_{m=1}^{\infty}$，以及一个多项式有界函数 $t: \mathbf{N} \to \mathbf{N}$。

H_{m} 的一个成员是一个函数 $h: \{0,1\}^{*} \to \{0,1\}^{t(m)}$，称为 $\mathcal{H}$ 的大小为 m 的实例。

$\mathcal{H}$ 必须满足以下条件：

1. 给定一个 m 值，存在一个（关于 m 的）概率多项式时间算法 Θ，在输入 m 时随机选择 $\mathcal{H}$ 的一个大小为 m 的实例。

2. 对于任何实例 $h \in H_m$ 和 $x \in \{0,1\}^*$，$h(x)$ 易于计算，即可以在 m 和 $|x|$ 的多项式时间内计算。
3. 给定一个如 1) 中随机选择的实例 $h \in \mathcal{H}$，很难找到 $x, y \in \{0,1\}^*$，使得 $h(x) = h(y)$ 且 $x \neq y$。

更形式化地：对于任何概率多项式时间算法 Δ，以及任何多项式 P，考虑大小为 m 的实例 h 的子集，对于这些实例，Δ 以至少 $1 / P(m)$ 的概率输出 $x \neq y$ 使得 $h(x) = h(y)$。令 $\epsilon(m)$ 为 Θ 选择这些实例之一的概率。那么作为 m 的函数，$\epsilon(m)$ 比任何多项式分数消失得更快。

注意定义 2.1 和 2.2 之间的本质区别在于，在 2.2 中，我们对函数输入的长度没有任何限制，除了从明显的事实——多项式时间算法不能哈希超多项式长度的消息——得出的结论。

3 基本构造

本节的主要结果是，可以从固定大小的抗碰撞哈希函数族构造出抗碰撞哈希函数族：

定理 3.1

令 $\mathcal{F}$ 为一个将 m 比特映射到 $t(m)$ 比特的固定大小抗碰撞哈希函数族。那么存在一个将任意长度的字符串映射到 $t(m)$ 比特串的抗碰撞哈希函数族 $\mathcal{H}$。

令 h 为 $\mathcal{H}$ 中大小为 m 的一个实例。那么在长度为 n 的输入上求值 h 最多可以使用 1 个处理器在 $n / (m - t(m) + 1) + 1$ 步内完成（我们将 $\mathcal{F}$ 中函数的求值算作 1 步）。

证明

对于 $\mathcal{F}$ 中每个大小为 m 的实例 f，我们将构造 $\mathcal{H}$ 中一个大小为 m 的实例 h。令 $t = t(m)$。对于比特串 a, b，我们用 $a \parallel b$ 表示 a 和 b 的串联。

构造将分为两种情况：首先讨论 $m - t > 1$ 的情况，稍后再处理 $m - t = 1$ 的情况。

我们描述如何计算 h 在输入 $x \in \{0,1\}^*$ 上的值：

将 x 分割成大小为 $m - t - 1$ 比特的块。如果最后一块不完整，则用 0 填充。令 d 为所需 0 的个数。令这些块表示为 $x_1, x_2, \ldots, x_{n/(m-t-1)}$，其中 $n = |x|$（填充后的长度）。

我们在这个序列后附加一个额外的 $m - t - 1$ 比特块 $x_{n / (m - t - 1) + 1}$，其中包含 d 的二进制表示，前面加上适当数量的 0。

然后定义一系列 t 比特块 $h_0, h_1, \ldots$：

最后，令 $h(x) = h_{n / (m - t) + 1}$。

检查 $\mathcal{H}$ 满足定义 2.2 中的条件 1 和 2 是容易的。对于条件 3，假设反证我们有一个算法 Δ，它能找到 $x \neq x'$ 使得 $h(x) = h(x')$。

令 h_i, x_i（分别为 h_i', x_i'）为计算 $h(x)$（分别为 $h(x')$）时的中间结果。

如果 $|x| \neq |x'| \mod (m - t)$，那么必然有 $x_{n / (m - t) + 1} \neq x_{n' / (m - t) + 1}'$，所以 $h(x) = h(x')$ 立即给我们一个 f 的碰撞。因此我们现在可以假设 $|x| = |x'| \mod (m - t)$，并且不失一般性地假设 $|x'| \geq |x|$。

现在考虑等式

如果 $h_{n/(m-t)} \mid x_{n/(m-t)+1} \neq h_{n'/(m-t)}' \mid x_{n'/(m-t)+1}'$，我们就有 f 的一个碰撞，完成。如果不是，我们可以转而考虑等式

并重复这个论证。显然，这个过程必须停止，要么通过产生一个 f 的碰撞，要么（因为 $x \neq x'$）通过建立等式

这显然是不可能的。

总结一下，我们现在有一个将 Δ 转化为一个寻找 f 碰撞的算法的归约。假设 Δ 最多需要 $T(m)$ 次比特操作。那么 x 和 x' 的长度必须小于 $T(m)$。

因此，整个归约需要 $O(T(m)F(m))$ 的时间，其中 $F(m)$ 是计算一个 f 值所需的时间，特别地，如果 T 是多项式，那么整个归约也是多项式时间。这最终与定义 2.1 中的条件 3 矛盾。

最后，我们讨论 $m - t = 1$ 的情况。一个简单但效率不高的解决方案是在所有消息被哈希之前对其进行前缀无关编码，然后使用类似于上面的构造，将 h 的定义改为：

这里，x_{i} 当然是 1 比特块。这可以用与上面大致相同的方式证明是安全的。

如果 f 满足引理 2.1 中的第二个条件，有一个更高效的解决方案：我们均匀选择一个 t 比特串 y_0，并定义

这次不对 x 进行前缀无关编码。上面的论证现在将表明，h 的一个碰撞要么给我们一个 f 的碰撞，要么给我们一个 y_0 的原像。但根据引理 2.1，我们就完成了。 $\square$

备注

- 使用引理 2.1 的构造的最后一个版本不仅适用于 $m - t = 1$ 的情况，而且适用于 P_f 是（接近）均匀分布或 $m - t = O(m)$ 的情况。这值得注意，因为此版本允许每次应用 f 时比通用构造多哈希 1 比特，因此效率稍高。
- 上述证明中附加一个额外块到消息的技巧，只是为了确保我们能够区分需要用 d 个 0 填充的消息，与在填充前恰好以 d 个 0 结尾的消息。在许多应用中，忽略最后一个块中的末尾 0 是完全可接受的，在这种情况下可以跳过构造的这一部分。

让我们看看定理 3.1 与先前已知的哈希函数之间的联系。在 [Da] 中，基于无爪置换对构造了哈希函数，即具有相同定义域 D 的置换对 (f_0, f_1)，找到满足 $f_0(x) = f_1(y)$ 的 $x \neq y$ 是困难问题。我们可以从这对 (f_0, f_1) 构造一个抗碰撞函数族的实例，通过定义函数 $f: D \times \{0, 1\} \to D$ 为：

对于 $x \in D$ 和 $b \in \{0,1\}$。在由此定义的函数族上应用定理 3.1，将得到与 [Da] 中提出的完全相同的哈希函数，只是我们消除了那里使用的消息前缀无关编码的需要（在这种情况下 P_f 是均匀的）。

作为第二个例子，考虑从传统密码构造哈希函数的最初想法之一，归功于 Rabin：令 E 是一个加密算法，使用 k 比特的密钥加密 t 比特的消息。令 $m = t + k$。我们将消息 x 分割成 k 比特块 $x_1, x_2, \ldots, x_n$。然后我们随机选择一个固定的 t 比特块 h_0 并令 $h_{i+1} = E_{x_{i+1}}(h_i)$。$h(x)$ 定义为 h_{n+1}。

这可以通过令 $f(a, b) = E_{\alpha}(b)$ 对于一个 t 比特块 b 和一个 k 比特块 a，纳入定理 3.1 的框架。不幸的是，这个 f 不是抗碰撞的：用任意密钥加密任何消息并用不同密钥解密，将有很大概率产生 f 的碰撞。这不一定意味着该函数是弱的。但这确实意味着，安全性证明不能仅基于 f 本身的属性，而必须依赖于函数的全局结构。

然而，对于 f 的具体例子，已经发现了弱点：如果使用 DES 作为 E，使得 t 仅为 64，众所周知，按上述构造的函数 h 允许给定 x 和 $h(x)$ 的敌方利用“生日悖论”找到一个 $y \neq x$ 使得 $h(x) = h(y)$。这实际上是一个比说哈希函数不抗碰撞更强的陈述。

在 ISO 内，目前提议标准化该方案的一个修改版本，其中 f 定义为 $f(a, b) = E_{\mathfrak{a}}(b) \oplus b$。对于这个版本的 f，事实上我们有望使用定理 3.1 来通过仅查看 f 证明整个函数的抗碰撞性：给定 c，如果 E 是一个强加密算法，则不易求解 $f(a, b) = c$ 得到 a 和 b，因此有很好的理由相信这个版本的 f 是抗碰撞的。

3.1 并行化哈希函数

基于定理 3.1，我们可以给出允许并行计算哈希值的替代构造：

定理 3.2

令 $\mathcal{F}$ 为一个将 m 比特映射到 $t(m)$ 比特的抗碰撞函数族。那么存在一个将任意字符串映射到 $t(m)$ 比特串的抗碰撞哈希函数族 $\mathcal{H}$，具有以下属性：

令 h 为 $\mathcal{H}$ 中大小为 m 的一个实例。令 $t = t(m)$。那么在长度为 n 的输入上求值 h 可以使用 $n / 2t$ 个处理器在 $O(\log_2(n / t)t / (m - t))$ 步内完成（我们将 $\mathcal{F}$ 中函数的求值算作 1 步）。

证明

我们给定 $\mathcal{F}$ 中一个大小为 m 的实例 f。根据定理 3.1，我们可以构造一个哈希函数 h'，它在 $t / (m - t)$ 步内将 $2t$ 比特映射到 t 比特。注意，由于输入长度是固定的，我们在构造 h' 时不需要附加额外的输入块。

然后我们构造 $\mathcal{H}$ 中的一个实例 h 如下：

给定一个长度为 n 的消息 x。我们用若干个 0 填充 x，使得得到的比特串 x_0 的长度等于某个 j 的 $2^j t$。现在通过如下方式定义 x_{i+1} 来构造序列 $x_0, x_1, \ldots, x_j$：将 x_i 分割成长度为 $2t$ 的块，对每个块应用 h'，并串联结果得到 x_{i+1}。

序列停止于 x_{j}，其长度为 t。然后我们使用定理 3.1 对 x 的长度 n 的二进制表示进行哈希，得到一个 t 比特块 len_{x}。最后我们令

关于计算 h 所需时间和处理器的陈述很容易验证。

至于抗碰撞性，假设我们能够在期望多项式时间内产生 $x \neq x'$ 使得 $h(x) = h(x')$。

如果 $x_{j'}' \mid \text{len}_{x'} \neq x_j \mid \text{len}_x$，我们就有 h' 的一个碰撞，与定理 3.1 矛盾。因此我们也可以假设 $n = n'$，因为否则 $\text{len}_{x'} = \text{len}_x$ 将意味着碰撞。

现在，$x \neq x'$ 意味着我们可以选择一个 i 使得 $x_i \neq x_i'$，但 $x_{i+1} = x_{i+1}'$。这显然意味着 h' 的一个碰撞。$\square$

最后，也很容易看出如何构造一个允许 c 个处理器协作计算哈希值的方案，实现对长消息 c 倍的加速。粗略地说，我们将消息分成大小大致相同的 c 部分，使用定理 3.1 并行哈希每个部分，最后再次使用定理 3.1 哈希这 c 个输出块。将其形式化并证明抗碰撞性留给读者。

4 具体构造

下面我们提出三个具有固定输入大小的抗碰撞函数的具体构造。然后可以通过直接应用定理 3.1 将这些函数转化为哈希函数。

4.1 基于模平方

我们首先给出一个基于提取具有两个大素数因子的大数的平方根的困难性的构造。该构造与 [Gir] 中考虑的函数有一些相似之处，但根本区别在于 [Gir] 中的函数不允许应用定理 3.1。

令 $n = pq$，其中 p 和 q 是大素数。令 n 的长度为 s 比特。具体而言，可以考虑 $s = 512$。接下来，选择 I，它是数字 $1, 2, \ldots, s$ 的一个真子集。对于任意 s 比特串 $y = y_1, y_2, \ldots, y_s$，令 $f_I(y)$ 是所有满足 $j \in I$ 的 y_j 的串联。

最后，我们可以通过设置 $m = s - 8$，$t = |I|$，并定义

来定义我们候选的抗碰撞函数 f，从 m 比特到 t 比特，其中 $3F|x$ 表示字节 $3F$（十六进制）和 x 的串联。这个串联意味着模平方的所有输入都小于 $n/2$，但又足够大以保证模约简总是发生。因此，我们防止了由 $x^2 = (-x)^2 \mod n$ 所暗示的平凡碰撞，也防止了通过选择例如小的 2 的幂作为输入来寻找碰撞的尝试。

我们还需要指定 I 的选择，以使 f 安全且高效。寻找 f 碰撞的问题可以重新表述为：找到数字 $x \neq y$，使得它们模 n 的平方在由 I 指定的位置上匹配。Girault [Gir] 展示了如果 I 指定了数量合理较少的（小于 64）最低有效位或最高有效位，如何做到这一点。

这表明我们应该选择均匀分布在 s 个可能位置上的位置，因为这样 Girault 的方法和相关方法 [GTV] 将失效。另一方面，有充分的实践理由不使用完全随机的位置，而至少将它们分组到字节中。此外，$|I|$ 不应选择得太小，以防止“生日碰撞”。

具体来说，如果 $s = 512$，上面的建议是选择 $|I| = 128$，并让 f_{I} 提取每第 4 个字节。

该函数每次模平方可以哈希最多 376 比特，这在例如 IBM P/S II 型号 80 上，速度约为 100 Kbits/秒。专用硬件将提供 Mbit/秒区域的速度。

4.2 基于 Wolfram 的伪随机比特生成器

第二个建议基于 Wolfram 提出的伪随机比特生成器 [Wo]。一般来说，伪随机比特生成器是一种算法，它将一个短的、真正随机的种子作为输入，并将其扩展为一个长的、看似随机的输出字符串。直观上很清楚，这样的生成器在某种意义上必须实现从其种子到其输出的单向函数：如果给定输出的某些部分很容易找到种子，那么整个输出就可以被预测，从而被识别为非随机的。

然而，这种单向性通常并不意味着直接从生成器构造的函数具有抗碰撞性——因此非常需要对具体实例进行分析。

让我们看看 Wolfram 建议的算法：我们定义从 n 比特串到 n 比特串的函数 g：令 $x = x_0, x_2, \ldots, x_{n-1}$，那么 $g(x)$ 的第 i 比特是

其中 1 的加减法是模 n 的。可以将其视为一个寄存器 R，其中比特通过设置 $R := g(R)$ 并行更新。这被称为一维元胞自动机。

为了将其用于伪随机比特生成，执行以下操作：

1. 随机选择 x。
2. $x\coloneqq g(x)$
3. 输出 x_0。转到 2。

在 [Wo] 中，分析了这个伪随机比特生成器，并给出了大量统计测试的结果。证明了其对抗限于某些计算类型的敌方的安全性，但其对抗任意多项式时间敌方的能力

仍然是一个猜想。然而，所有已知的证据都表明，该生成器实际上非常强大。

令算法在输入 x 上产生的比特表示为 $b_{1}(x), b_{2}(x), \ldots$。

利用这个构造抗碰撞函数的自然方法是选择两个自然数 $c < d$，并让函数 f_0 定义为

这个想法有两个可能的缺陷，必须处理：

首先，对此类函数的一个自然要求是所有输出比特依赖于所有输入比特。很容易看出，改变 x 的 1 比特最终在多次执行 $x := g(x)$ 后会影响 x 的所有比特——但效果通过寄存器传播缓慢：每次应用 g 向右传播 1 比特，向左传播约 0.25 比特 [Wo]。因此，选择太小的 c 显然是危险的。c 的一个自然最小值将是保证所有输出比特依赖于所有输入比特的值。

第二个问题是 g 本身不是抗碰撞的：例如，$g(1^n) = g(0^n) = 0^n$。显然，$g(x) = g(y)$ 意味着 $f_0(x) = f_0(y)$。更一般地，如果对于小的 v 有 $g^v(x) = g^v(y)$，那么至少存在不可忽略的机会使得 $f_0(x) = f_0(y)$。

消除这个困难的一个自然方法是将 f 限制在 n 比特串的一个子集上，从而降低上述形式的对 x, y 存在的概率，或者至少使它们更难找到。一个具体的可能性是限制为形式为 $x \mid z|$ 的串，其中 z 是一个在 $\{0,1\}^r$ 中随机选择的常数串，$r < n$。

因此，我们将定义我们最终的候选函数 f，使其将一个 $n - r$ 比特串 x 映射为

下面的引理为这种方法提供了支持：

引理 5.1

如果 $g^{\nu}(x) = g^{\nu}(y) = z$，并且 x 和 y 的 2ν 个连续比特相等，那么 $x = y$。

证明

令 j 为我们知道相等的 $2v$ 比特紧右侧的位置索引。z 的每个比特最多依赖于 x（或 y）的 $2v + 1$ 个比特。选择 l 使得 z_l 是 x（或 y）的比特 $j, \ldots, j + 2v$ 的函数。现在，由于反转 x_j 将反转 z_l，我们的假设意味着 $x_j = y_j$。我们现在可以将相同的论证“滑动”一个位置到右边。

这提供了良好的证据，表明选择相对较大的 r 将使“平凡”碰撞更加稀疏且更难找到。

作为一个具体例子，假设我们选择 $n = 512$，$r = 256$，$c = 257$ 和 $d = 384$。得到的 f 将把 256 比特串映射到 128 比特串，因此通过定理 3.1 从 f 构造的哈希函数将以 128 比特的块处理消息，并产生 128 比特输出。

该函数非常适合硬件实现：在 VLSI 中，我们可以通过并行更新所有比特来计算 g，因此一次 g 的应用只需要 1 或 2 个时钟周期，与 n 无关。即使时钟速度相当适中，这也能提供 Mbit/秒区域的速度。此外，这样的硬件实现将非常容易且廉价。

4.3 基于背包问题

尽管背包问题是 NP 完全的，因此在最坏情况下可能非常困难，但在密码学中利用这种困难并不容易，正如许多基于该问题的公钥系统的命运所示。

然而，困难主要源于加密函数必须是可逆的，因此使用的背包必须具有某种内置结构，这在许多情况下被密码分析者利用。另一方面，哈希函数永远不需要求逆，因此可以使用完全随机生成的背包。

天真的做法是在区间 $1 \ldots M$ 中随机选择数字 $a_1, \ldots, a_s$，其中 s 是预期消息的最大长度。然后我们可以将二进制消息 $m_1, m_2, \ldots, m_s$ 哈希为

如 [GC] 所示，这对于大的 s 是完全不安全的，更准确地说，当 $M \leq s^{\log(s)/4}$ 时。为了解决这个问题，我们建议将 s 固定为相对于 M 合理较小的值，并使用定理 3.1 来构造实际的哈希函数。例如，可以选择 s 约为 $2\log(M)$，这意味着 f 将 s 比特块压缩为 $s/2$ 比特块，并且 [GC] 的条件远未满足。

具体选择可以是 $s = 256$ 和 $M = 2^{120} - 1$。这将使最终哈希函数的输出长度为 128 比特。在 IBM P/S II 型号 80 上，此版本将以大约 250 Kbits/秒的速度运行。

要指定该函数，需要指定大约 4 Kbytes 的数据。在例如具有 640K RAM 的 PC 上，这似乎并不算过多。但在内存较小的情况下，可以牺牲时间换取内存，并伪随机生成 $\pmb{a}$ 而不是记住它们。

[ImNa] 的结果有力地表明该函数确实是抗碰撞的，尽管证明的安全属性比我们这里需要的要弱（见第 1 节）。

该函数强度的另一个迹象是，判断给定背包是否诱导单射映射的问题在一般情况下是 co-NP 完全的。碰撞当然是非单射性的见证（如果背包压缩其输入，决策问题显然是平凡的，但这并不意味着见证容易找到）。

我们注意到，即使是稍微扩展其输入的背包（因此不能被 [ImNa] 使用）也可以用来构建在 $[\mathrm{NaYu}]$ 意义上安全的哈希函数，如果愿意假设它们诱导抗碰撞映射的话。考虑到所涉及问题的 co-NP 完全性以及决策问题在这种情况下非平凡的事实，这似乎是合理的。可以通过调整 $[\mathrm{NaYu}]$ 的技术获得构造和证明。

参考文献

[Da] Damgård: "Collision Free Hash Functions and Public Key Signature Schemes", Proceedings of EuroCrypt 87, Springer.

[De] D. Denning: "Digital Signatures with RSA and other Public Key Cryptosystems", CACM, vol.27, 1984, pp.441-448.

[DP] Davis and Price: "The Application of Digital Signatures Based on Public Key Crypto-Systems", Proc. of CompCon 1980, pp.525-530.

[GC] Godlewski and Camion: “Manipulation and Errors, Localization and Detection”, Proceedings of EuroCrypt 88, Springer.

[Gi] Gibson: “A Collision Free Hash Function and the Discrete Logarithm Problem for a Composite Modulus”, Manuscript, 1/10/88, London, England.

[Gir] Girault: "Hash Functions Using Modulo-n Operations", Proceedings of Euro-Crypt 87, Springer.

[GTV] Girault, Toffin and Vallee: “Computation of Approximate L-th Roots Modulo n and Application to Cryptography”, Proceedings of Crypto 88, Springer.

[ImNa] Impagliazzo and Naor: "Efficient Cryptographic Schemes Provably as Secure as Subset Sum", Proc. of FOCS 89.

[Me] Merkle: "One Way Hash Functions and DES", these proceedings.

[NaYu] Naor and Yung: “Universal One-Way Hash Functions”, Proc. of STOC 89.

[Wi] Winternitz: "Producing a one-way Hash Function from DES", Proceedings of Crypto 83, Springer.

[Wo] Wolfram: “Random Sequence Generation by Cellular Automata”, Adv. Appl. Math., vol 7, 123-169, 1986.