摘 要

Hash 函数是密码学中最基本的模块之一，被广泛应用于数字签名、消息鉴别、模拟 random oracle 和伪随机数生成器等领域，是近几年密码学研究的热点领域。本文对 Hash 函数的研究现状和存在的问题做了全面的介绍，比较和分析了 Hash 函数的各种设计思想和分析方法，并对若干问题进行了探讨。MD 迭代方式是目前最常用的设计方法，通常为了保证 Hash 函数的抗碰撞性需要压缩函数具有抗自由碰撞性。本文进一步研究了 MD 迭代方式构造的 Hash 函数和压缩函数之间的关系；在黑盒子模型下证明了在压缩函数不能抵抗某些自由碰撞攻击的条件下，MD 迭代方式构造的 Hash 函数仍然具有抗碰撞性。此结果放宽了对压缩函数的限制，更有利于压缩函数的设计。平衡度是衡量 Hash 函数安全性的新指标，本文重点研究了压缩函数平衡度在 MD 迭代方式下的保持问题。在原定义模型下，MD 迭代方式不能保持压缩函数的平衡性，从而给压缩函数的设计要求带来了困难。鉴于此本文提出了局部平衡度的概念，并证明当压缩函数满足局部平衡性要求时，MD 迭代方式构造的 Hash 函数的平衡度要大于等于压缩函数的平衡度。最后从实际应用角度出发，讨论了 Hash 函数针对目前攻击方法的预防措施，通过对算法简单升级的办法来增加其抵抗攻击的能力。

关键词：Hash 函数，抗碰撞性，MD 构造方式，Hash 函数平衡度，差分攻击。

Abstract

Hash function is one of the most important fundamental modules of cryptography. It is widely used in many cryptographic applications, such as digital signature, message authentication, random oracle simulating and pseudo random number generator. During recent years, hash function has attracted more and more attention in cryptographic field. In this thesis we describe the state of the art for cryptographic hash functions and some open problems, compare and analyze design and attack methods of hash function and discuss issures related to our work. MD iterative structure is the most popular method in real design. In order to contruct collision-resistant hash function we need to design free-start collision-resistant compression function. We did further research work on the relation between the security properties of hash function and compression function and prove that, under certain circumstance, hash function can be collision resistant even though compression function is not against free-start collision attack. This result can relax the security requirement of compression function in design. Hash function balance is a newly introduced security criterion of hash function. We discuss the problem of balance preservation in MD transform. In original model MD transform can not keep the balance of compression function, which may make the design of compression function much trickier. We introduce partial-balance conception to solve this problem. It can be proved that the balance of hash function can be equal or bigger than that of compression function when compression function is partial balance. Finally we summarize kinds of attack technique and some latest attack results. The way to prevent these attacks from the practical perspective is also discussed. Combining other's techniques we put forward several possible methods.

Keywords: Hash function, collision-resistant, MD-strenghned structure, balance , differential attack.

第一章　引言

随着计算机技术和网络技术的迅速发展，信息化和网络化成为当今世界经济与社会发展的大趋势。而计算机网络的开放性和共享性使得对信息安全的需要更加强烈。信息安全的主要需求包括机密性和可鉴别性两个部分。信息的机密性是指未被授权的主体不能得到信息的内容。信息的可鉴别性包括两个方面，一是消息来源的鉴别（数据源可鉴别），二是消息未被篡改过（数据完整性）。

过去，对消息的机密性保护是通过物理的安全和信任来得到的，比如档案被密封起来并放到绝对安全的地方。对消息可鉴别性的保护依赖于文件或者签名的难以伪造。而在电子时代，信件、合同及其他文件被二进制数据流所替代。电子文件具有易复制和易修改的特性，特别是在开放的网络环境，电子文件将更容易被获取、复制和修改，从而带来更大的不安全性。

交流对话的保护同样存在这个问题。对于面对面的交谈，人们很容易找到一个防止窃听的环境。并且，交流双方的身份鉴别也很容易实现。而在网络中，人们的身份只是一些代码或信息流，攻击者很容易通过截获或者伪造这些信息来冒充合法用户。并且，窃听这些对话也变得非常容易。

密码技术可以很好的解决上述问题。很多世纪以来，密码技术被用来保护军事和外交秘密。这些技术的大部分是加密方案，即通过密钥将消息转换成密文。这些密码对于未授权的用户是不可鉴别的，只有拥有密钥的用户才能解密它们。密码研究者曾经认为只要保证了信息的机密性同时也就保证了信息的可鉴别性。其理由如下：如果某个密文经过解密后得到了有意义的消息，那么这个密码一定是来源于拥有密钥的实体。但事实上，构造一个假消息的密文并不一定要知道密钥：完整性保护取决于加密算法和算法的模式。

密码学曾经只是少数人研究的艺术，但是随着 DES 的公布以及公钥体制的提出，密码学已经成为一个科学研究领域。新提出的概念和定义清楚地分开了机密性和可鉴别性问题，并且使得鉴别算法成为一个重要的研究课题[1]。经过近些年的发展，已经提出了大量的鉴别方案。这些方案在实际中已经有了广泛的商业应用，比如为电子商务和无线通信系统做安全性保护等等。

鉴别算法主要包括两个部分，一部分是没有密钥的 Hash 函数，另一部分是

有密钥的 MAC。本论文的主要内容是 Hash 函数的性质和安全性分析。简单的说 Hash 函数就是把任意长度的输入消息映射成固定长度的消息，而且一般情况输出长度均比输入长度要短。Hash 函数最常见的应用是数字签名。在数字签名体制里，签名算法通常只对消息的摘要进行签名而不是原消息。Hash 函数也用于生成 MAC（一般是有密钥的 hash 函数）、密码保护、零知识证明、密钥推导算法、模拟 Random Oracle 等。

作为最基础的密码模块，Hash 函数的历史并不算长，属于密码技术中比较年轻的分支。最著名 Hash 函数是产生于上世纪九十年代初的 MD 系列，包括 MD4、MD5 和 RIPEMD 等。随后基于 MD 系列，NSA 又设计了 SHA 系列 Hash 函数，包括 SHA0、SHA1 和 SHA2。其中，MD5 和 SHA1 是在实际中应用最广泛的算法。另外，随着密码设计和分析技术的发展，又有一些新的输出长度更长、安全性更高的算法出现，如 Tiger 和 Whirlpool 等。与其它密码算法的研究类似，Hash 函数的研究也主要有两个方向，一个是设计原理和新算法的设计，另一个是安全性分析。

近些年，Hash 函数的设计技术并没有得到很大的发展。设计主要包括压缩函数设计和模式构造两部分。压缩函数设计有直接构造和基于分组密码构造等方式。直接构造的算法仍是采用的 MD 系列的设计思想和部分分组密码的设计理念，并没有新的突破。基于分组密码的构造方式由于效率的问题至今也没有很好的解决方案。同样，模式构造方面最近的研究成果也不多。目前使用较多的仍是 Damgard 等人提出的 MD 迭代构造模式。本文针对 MD 迭代模式的某些性质做了一些探讨，研究了在该模式下 Hash 函数和压缩函数的安全性之间的关系，以及平衡度的保持问题。通过分析证明，我们可以在保持 Hash 函数安全性的前提下降低对压缩函数安全性的要求，从而有利于压缩函数的设计。

与设计方法相反，Hash 函数的分析技术在近些年取得了巨大的进步。各种分析方法不断发现和完善，其中最突出的是差分分析方法。差分分析方法原是用于分组密码的攻击方法，但对于 Hash 函数也是非常有效的。早在 1998 年，H. Dobbertin 就使用差分分析方法成功的找到了 MD4 算法的碰撞对。这种攻击非常有效，甚至可以找到有意义的碰撞对。但是，对于 MD5 和 SHA 系列等算法，由于算法本身的复杂性，差分分析不能很容易的找到碰撞对。因此，除了改善差

分分析技术本身以外，密码分析者还提出了多种其它补充的分析方法。Joux 等人在对 SHA0 的分析过程中提出了近似碰撞和多块消息连接的分析方法，并用此方法攻破了 SHA0 算法。分析技术最大的突破是在 2004 年，王小云宣布攻破了 MD5 算法。该攻击采用的也是差分分析，并结合了近似碰撞和多块消息连接攻击。该攻击最有效的手段是提出了消息块修正分析方法，此方法配合差分分析方法可以很大地降低攻击的复杂度。这种结合各种分析方法的攻击对于直接设计的 Hash 函数非常有效。王小云利用这种方法先后攻击了 MD4、RIPEMD、HAVAL 和 SHA0，并且于 2005 年在理论上攻破了 SHA-1 算法。由于 MD5 和 SHA1 等都是实际中常用的 Hash 函数，这些攻击成果给实际应用带来了巨大的隐患。面对这一状况，我们有两种解决办法：一个是重新设计新的 Hash 算法，另一个是采用某种简单的方式将算法升级以抵抗新的攻击方法。设计新的算法不是短时间内可以完成的，并且将实际中的所有算法同时替换也是不现实的。而升级算法简单易行并且十分适合实际应用的需要。本文针对这一途径进行了探讨，在结合其它改进方式的基础上提出了新的解决方式。另外，Joux 等人发现了针对 MD 迭代结构的多碰撞攻击和前像攻击。本文也对此进行了讨论，并通过对 MD 结构稍做修改来抵抗这种攻击。

本文的前三章包括了 Hash 函数的基本性质介绍，Hash 函数的各种设计方法的比较和分析。第四章是关于压缩函数与 Hash 函数安全性质关系的一些分析研究；第五章是 Hash 函数平衡度的分析，其中重点是压缩函数平衡度在 MD 构造方式下的保持问题；第六章着重介绍了各种攻击方法，并且针对这些攻击方法对目前的 Hash 函数结构做了改进以增强其安全性。最后总结全文，并对 Hash 函数未来的发展方向做了探讨。

第二章　Hash 函数的基本概念

Hash 函数是把任意长度的二进制串映射到特定长度的二进制串的函数，是最基本的密码学工具之一，广泛应用于数字签名、消息鉴别、模拟 Random Oracle 和伪随机发生器等领域。

1．1　Hash 函数的定义

定义 1.1 ：我们称函数 H：{0，1}*→{0，1}^{n}为 Hash 函数，其中{0，1}*代表任意长度的比特串集合，{0，1}^{n}代表长度为的 n 比特的二进制串集合。其作用是用于计算任意长度的消息的 n 比特摘要。

事实上，构造一个可以有任意长度输入的函数是非常困难的，因此，大部分的 Hash 函数的设计是从某个输入长度固定的压缩函数入手，然后采用某种结构将压缩函数扩展生成输入长度为任意值的 Hash 函数。这种扩展方式有迭代、树型和图型等结构。为了方便介绍 Hash 函数的安全性要求，在这里介绍最常用的迭代结构 Merkle－Damgard 加强结构，简称为 MD 方式[2]。

MD 方式首先对消息填充，以满足压缩函数输入长度的要求和保证其安全性。填充方式是在消息末尾首先填充一比特“1”，然后填充足够比特的“0”，最后将消息的长度填充到末尾。填充后消息长度正好为压缩函数输入长度的整数倍。填充好的消息被分成固定大小的消息块$M_1,...,M_l$，初始变量设成某个固定值，然后进行压缩函数的迭代处理。设压缩函数为 f，则 Hash 的处理过程总结如下：[2]

- --填充原始消息并将消息分成固定长度的块$M_1,...,M_l$
- --将初始变量 H_0 设成固定值 IV
- --设 i 从 1 到 L，计算$H_i = f(H_{i-1}, M_i)$
- --输出 $H(M) = H_l$

1．2　Hash 函数的安全性质

安全性是对可能存在的攻击而言的，因此在定义压缩函数和 Hash 函数的安全性之前，先介绍可能存在的攻击方式。设 H 是 Hash 函数 $\{0,1\}^* \times \{0,1\}^n \to \{0,1\}^n$，f 是压缩函数 $\{0,1\}^m \times \{0,1\}^n \to \{0,1\}^n$。M 是任意长度的消息，m 是单块长度消息，h 是 f 的输入或输出中间变量。

- Hash 函数的攻击方式：
  - **前像攻击**：给定任意值 $Y \in \{0,1\}^n$，找到消息 M 使得 $H(M) = Y$；
  - **二次前像攻击**：给定任意消息 M，找到消息 $M' \neq M$ 使得 $H(M) = H(M')$；
  - **碰撞攻击**：找到两个消息 $M' \neq M$ 使得 $H(M) = H(M')$；
- 压缩函数的攻击方式

  因为压缩函数有两个变化的输入变量，因此对于压缩函数的攻击可以分为固定输入中间变量和自由输入中间变量攻击。如果固定输入中间变量，那么对压缩函数的攻击等同于对 Hash 函数的攻击。对于自由输入中间变量的攻击，有如下攻击方式：
  - **自由前像攻击**：给定任意值 $Y \in \{0,1\}^n$，找到 $m \in \{0,1\}^m, h \in \{0,1\}^n$ 使得 $f(h,m) = Y$；
  - **自由二次前像攻击**：给定消息 m，找到 $m' \in \{0,1\}^m, h \in \{0,1\}^n$ 使得 $m' \neq m$ 且 $f(h,m) = f(h,m')$；
  - **自由碰撞攻击**：找到两对消息和中间变量 $m, m' \in \{0,1\}^m, h, h' \in \{0,1\}^n$ 使得 $(h,m) \neq (h',m')$ 且 $f(h,m) = f(h',m')$；
  - **半自由碰撞攻击**：找到两个消息块和一个中间变量 $m, m' \in \{0,1\}^m, h \in \{0,1\}^n$ 使得 $(h,m) \neq (h',m')$ 且 $f(h,m) = f(h,m')$；[3]

我们用对这些攻击的抵抗性来定义 Hash 函数或压缩函数的安全性要求。对

于某种攻击方式，如果对 Hash 函数最好的攻击方法是强力攻击，那么就说该 Hash 函数是安全的。用攻击中调用压缩函数的次数来衡量攻击的复杂度，定义如下：

- **抗碰撞性**：Hash 函数 H（或者压缩函数 f）是抗碰撞的，如果最好的碰撞攻击的计算复杂度为 $2^{n/2}$。
- **抗前像（二次前像）性**：Hash 函数 H（或者压缩函数 f）是抗前像（二次前像）的，如果最好的前像（二次前像）攻击的计算复杂度为 2^n。
- **抗自由碰撞性**：压缩函数是抗自由碰撞的，如果最好的自由碰撞攻击复杂度为 $2^{n/2}$。
- **抗自由前像（二次前像）性**：Hash 函数 H（或者压缩函数 f）是抗自由前像（二次前像）的，如果最好的自由前像（二次前像）攻击的计算复杂度为 2^n。
- **抗半自由碰撞性**：压缩函数是抗半自由碰撞的，如果最好的半自由碰撞攻击的计算复杂度 为 $2^{n/2}$。

显然，抗自由碰撞的函数一定是抗碰撞的，抗二次前像攻击的函数一定是抗前像攻击的。但是，抗碰撞性和抗前像性之间并不存在直接的关系。虽然对于安全的 Hash 函数来说，前像攻击的复杂度要高于碰撞攻击，但是这两种安全性不能相互替代，也不存在严格递增的关系[3]。抗前像性并不能保证抗碰撞性，同样有抗碰撞性的 Hash 函数也不能保证其有抗前像性。因此，安全的 Hash 函数必须同时满足抗碰撞性和抗前像性。但是，通常对 Hash 函数安全性的研究更关注抗碰撞性。这是因为相对来说单向性更容易满足，并且前像攻击的复杂度也更高。

1.3 Hash 函数的分类

广义的分，Hash 函数可以分为普通 Hash 函数和带密钥的 Hash 函数（MAC）。普通 Hash 函数又有如下分类：

- **单向 Hash 函数（one-way hash function）**：

  单向 Hash 函数是满足如下条件的函数：

  a) 函数的输入 x 可以是任意长度，输出 $h(x)$ 是固定长度 n。

b) 给定函数 h 和输入 x，$h(x)$ 的计算应该很简单

c) 任意给定的 h 的输出 y，找到一个消息 x 满足 $h(x)=y$ 在计算上是不可行的。任意给定 x 和 $h(x)$，找到另一个消息 x_1 使得 $x_1 \neq x, h(x_1)=h(x)$ 在计算上也是不可行的。

- **抗碰撞 Hash 函数（collision－resistant hash function）：**

抗碰撞 Hash 函数是满足如下条件的函数：

a) 函数的输入 x 可以是任意长度，输出 $h(x)$ 是固定长度 n。

b) 给定函数 h 和输入 x，$h(x)$ 的计算应该很简单

c) 满足不可逆性和抗二次碰撞性

d) 满足抗碰撞性

- **泛单向 Hash 函数（universal one-way hash function)**

泛单向 Hash 函数是满足如下条件的函数

a) 输入 x 可以是任意长度，并且函数有一个随机参数 s，s 是随机选取的，选定之后即可以公开，输出值 $h(x,s)$ 具有固定长度 n。

b) 给定 h、s，和输入 x，$h(x,s)$ 的计算简单。

c) 在给定 x 和 $h(x,s)$ 之后，找到 x' 使得，$x \neq x', h(x,s)=h(x',s)$ 在计算上是不可行的。

- **MAC**

MAC 是满足如下条件的函数：

a) 输入 x 可以是任意长度，并且函数有另一个长度为 k 的称为密钥的输入，输出值 $h(x,k)$ 具有固定长度 n。

b) 给定 h，k，和输入 x，$h(x,k)$ 的计算简单。

c) 给定 x，在不知道 k 的情况下，计算 $h(x,k)$ 是不可行的。即使在得到很多消息和验证码对的情况下，确定 k 或者计算一个新消息的 MAC 值也是不可行的。

1．4　输出长度的要求

设某个 Hash 函数的输出空间为 n 比特的字符串，即 $R=\{0,1\}^n$。如果该函数可以满足以下条件就称该函数是完全安全的：

- 寻找前像或者二次前像的复杂度为 2^n
- 寻找碰撞的复杂度为 $2^{n/2}$

由生日攻击的性质知道，前像生日攻击的复杂度为 O（2^n），碰撞生日攻击复杂度为 O（$2^{n/2}$）。显然，定义中的复杂度等于生日攻击的复杂度，也就是说对完全安全的 Hash 函数最有效的攻击是生日攻击。

对于具有完全安全性的 Hash 函数来说，现在的问题是输出长度 n 至少取何值才能使生日攻击是在实际中是不可行的。1995 年，n＝128 被认为是足够安全的长度，所以 MD5 算法的输出长度为 128 位。但是随着计算技术的发展，128 位已经不够安全。最新的研究认为 2^{80} 是较为安全的界限，所以 Hash 函数的输出应该至少应该大于等于 160 位。单向函数的输出长度应该大于等于 80 位。如果考虑到较为长期的安全性，Hash 函数应该选择更大的输出长度，如 192 位，256 位和 512 位等。目前最新的算法如 tiger 和 whirlpool 等的输出长度都是 512 位。至于单向函数，由于攻击者可以同时搜寻大量不同的值的前像，所以其长度更应该取较大的值。

1．5　几种常见 Hash 函数介绍

目前实际应用中比较常见的 Hash 算法是 MD5、SHA1 和 SHA2 系列等算法，另外一些新设计的算法有 tiger 和 whirlpool 等。这里简要的介绍一下 MD5 和 SHA1 算法的内部结构。

MD5 和 SHA1 都是采用的 MD 迭代式构造方式，这里省略了消息的填充和分块部分，只介绍他们的压缩函数的内部构造。

1.5.1. MD5

MD5 的输出长度为 128 位，消息的分组长度为 512 比特。512 比特首先被分成 16 个 32 比特的字。压缩函数有四轮，每轮有 16 步操作。连续的四步操作

有如下形式：

$$a = b + ((a + \Phi_i(b,c,d) + w_i + t_i)) <<< s_i),$$
$$d = a + ((d + \Phi_{i+1}(a,b,c) + w_{i+1} + t_{i+1}) <<< s_{i+1}),$$
$$c = d + ((c + \Phi_{i+2}(d,a,b) + w_{i+2} + t_{i+2}) <<< s_{i+2}),$$
$$b = c + ((b + \Phi_{i+3}(b,d,a) + w_{i+3} + t_{i+3}) <<< s_{i+3}).$$

其中，a、b、c 和 d 是四个中间变量寄存器，初始值为 MD5 的初始向量。t_{i+j}，$s_{i+j}(j = 0,1,2,3)$是与步数相关的常量，w_{i+j}是该步的消息字，$<<< s_{i+j}$是循环左移s_{i+j}位，“+”是模 32 加。消息字的顺序和常量的取值这里不祥述。

每一轮都选取了不同的非线性轮函数，如下所示：

$$\Phi_i(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z), 0 \le i \le 15,$$
$$\Phi_i(X,Y,Z) = (X \wedge Z) \vee (y \wedge \neg Z), 16 \le i \le 31,$$
$$\Phi_i(X,Y,Z) = X \oplus Y \oplus Z, 32 \le i \le 47,$$
$$\Phi_i(X,Y,Z) = Y \oplus (X \vee \neg Z), 48 \le i \le 63,$$

其中，X，Y 和 Z 都是 32 比特的字[5]。

1.5.2. SHA-1

SHA-1 的输出长度为 160 比特，消息块长度为 512 比特。SHA-1 的轮函数与 MD5 类似，不同之处在于消息字处理顺序的编排过程。MD5 只是在每一轮中对消息字的顺序做简单变换，而 SHA-1 采用线性函数生成后续消息字，其生成函数如下所示[6]：

$$m_i = (m_{i-3} \oplus m_{i-8} \oplus m_{i-14} \oplus m_{i-16}) << 1, i = 16,...,79;$$

扩展后的消息经过四轮处理，每轮为 20 步，算法描述为：

For i = 1，2，…，80

$$a_i = (a_{i-1} << 5) + f_i(b_{i-1} + c_{i-1}, d_{i-1}) + e_{i-1} + m_{i-1} + k_i$$
$$b_i = a_{i-1}$$
$$c_i = b_{i-1} << 30$$
$$d_i = c_{i-1}$$
$$e_i = d_{i-1}$$

$IV = (a_0, b_0, c_0, d_0, e_0)$ 为初始变量。其中每一轮的非线性函数如下表所示：

$$\Phi_i(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z), 0 \le i \le 19,$$
$$\Phi_i(X,Y,Z) = X \oplus Y \oplus Z, 20 \le i \le 39,$$
$$\Phi_i(X,Y,Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z), 40 \le i \le 59,$$
$$\Phi_i(X,Y,Z) = X \oplus Y \oplus Z, 60 \le i \le 79.$$

这两种算法都是基于非线性函数多轮处理的结构。这种结构因过于简单因而对差分攻击的抵抗力非常弱，在第六章将具体介绍对它们的差分攻击。

第三章 Hash 函数的构造方法

由 Hash 函数的定义可知，Hash 函数的输入要求是任意长度的。但事实上，构造一个有任意长度输入的函数是非常困难的。因此，通常的做法是首先设计某些有固定长度输入的压缩函数，然后采用某种结构将其连接来构造有任意输入长度的 Hash 函数。因此，Hash 函数构造方法的介绍分为两个部分，第一部分介绍由压缩函数构造 Hash 函数的几种模式，第二部分介绍目前设计压缩函数的主要方法。

3.1 Hash 函数构造模式

构造模式除了要满足输入长度的要求外，更重要的一点是要保证安全性。即在压缩函数是完全安全的前提下，该构造模式构造的 Hash 函数也要是完全安全的。更好的构造模式甚至可以降低对压缩函数安全性的要求仍能达到完全安全性。在此基础上，还要考虑实现的简单性和效率问题。目前，安全的构造模式并不多，主要有迭代式、树型方式以及可自由扩展方式等等。其中，实际应用最广泛的是 MD 迭代式构造模式。下面介绍这几种构造模式的具体结构。

3.1.1 MD 线性迭代构造模式

MD 构造模式是由 Merkle 和 Damgard 分别提出的一种构造模式，它是第一个具有可证明安全的构造模式。MD 是一种迭代式结构。在定理 3.1 中来对这种方式进行描述并证明其正确性[7]。

定理 3.1 设 F 是一个将长度 m 比特串映射到长度为 t（m）比特串的抗碰撞的函数族，那么存在一个抗碰撞的 Hash 函数族 H 可以将任意长度的串映射到 t（m）长度的串。

设 h 是 H 的一个实例，那么对于长度为 n 比特的消息串在一个处理器的情况下可以在 $n/(m-t(m)+1)+1$ 步内完成压缩过程。

证明：对于每一个 $f \in F$，构造一个 $h \in H$。设 $t = t(m)$，a||b 表示 a 和 b 的连接。

将构造方式分为两种情况：首先讨论 $m-t>1$ 的情况，随后来看 $m-t=1$ 的情况。

对于给定的任意长度输入 x，h 的计算过程为：

将 x 分成长度为 $m-t-1$ 大小的块。如果最后一块长度不足，那么在其后面补充足够的 0。设 d 是需要补充的 0 的个数。用 $x_1, x_2, ...x_{n/(m-t-1)}$ 来表示消息块，其中 n=|x|。在这些消息块之后附加一个新块 $x_{n/(m-t-1)+1}$。这个新的消息块是 d 的二进制表示。

按如下方式定义 t 比特长度的 $h_0, h_1, ..$：

$$h_1 = f(0^{t+1} \| x_1)$$
$$h_{i+1} = f(h_i \| 1 \| x_{i+1})$$

最后，令 $h(x) = h_{n/(m-t)+1}$。

下面来证明 h 的抗碰撞性。设 $h_i, x_i (resp. h'_i, x'_i)$ 是计算 $h(x), h(x')$ 过程中的中间变量。如果 $|x| \neq |x'| \bmod (m-t)$ 那么显然有 $x_{n/(m-t-1)+1} \neq x'_{n'/(m-t-1)+1}$，因此 $h(x) = h(x')$ 意味着找到了 f 的一个碰撞。假设 $|x| = |x'| \bmod (m-t)$，不失一般性，假设 $|x| \geq |x'|$。

考虑下面等式 $h(x) = f(h_{n/(m-t)} \| x_{n/(m-t-1)+1}) = f(h'_{n'/(m-t)} \| x'_{n'/(m-t-1)+1}) = h(x')$，如果 $h_{n/(m-t)} \| x_{n/(m-t-1)+1}) \neq h'_{n'/(m-t)} \| x'_{n'/(m-t-1)+1}$，那么就得到了 f 的碰撞对。否则，可以考虑下面的等式 $f(h_{n/(m-t)} \| x_{n/(m-t-1)+1}) = f(h'_{n'/(m-t)} \| x'_{n'/(m-t-1)+1})$ 并依次往前类推。显然，这个过程停止的情况要么是找到 f 的一个碰撞，要么有下面等式成立

$$0^{t+1} \| x_1 = h'_i \| 1 \| x'_{i+1}$$

显然这个等式是不可能成立的。

在证明过程中，将对 Hash 函数的碰撞攻击归约为对压缩函数的碰撞攻击，规约的复杂度变换是多项式级别的。由此证明，抗碰撞的压缩函数在这种迭代构造方式下是能够构造任意输入长度的抗碰撞的 Hash 函数的。

最后来讨论 $m-t=1$ 的情况。如果 f 满足抗前像攻击，那么可以如下构造 h。

首先选取 t 比特长度的 y_0，按如下方式定义 h：

$$h_1 = f(y_0 \| x_1)$$
$$h_{i+1} = f(h_i \| x_{i+1})$$

证明方法与前一种情况类似，h 的碰撞要么产生一个 f 的碰撞，要么会生成 y_0 的一个前像。而前面定义 f 是抗前像攻击的。

证明完毕

3．1．2　MD 并行迭代模式

线性迭代模式虽然具有可证明安全，但是这种结构不能够并行，效率比较低。在定理 3.1 的基础上，下面来构造可并行的迭代方式。这种方式其实是一种树型结构。我们在定理 3.2 中描述了并行构造方式并证明了其安全性。

定理 3.2　设 F 是一个将长度 m 比特串映射到长度为 $t(m)$ 比特串的抗碰撞的函数族，那么存在一个抗碰撞的 Hash 函数族 H 可以将任意长度的串映射到 $t(m)$ 长度的串。并且 H 满足如下性质：

设 h 是 H 的一个实例，那么对于长度为 n 比特的消息串在 $n/2t$ 个处理器的情况下可以在 $O(\log_2(n/t)t/m-t)$ 步内完成压缩过程。

证明：设 $f \in F$ 是一个 F 的一个实例。由定理 3.1，构造 Hash 函数 h'，h' 在 $t/(m-t)$ 步内将 2t 比特映射到 t 比特。注意，既然现在输入的长度已经是确定的了，那么就没有必要在 h' 的后面添加长度块了。

按如下方式构造 $h \in H$：

设消息 x 的长度为 n。用 0 来填充 x 使得 x 的长度等于 $2^j t$，j 是某个整数。现在构造序列 $x_1, x_2, ...x_j$，其中 x_{i+1} 由 x_i 来定义：将 x_i 分割成长度为 2t 的块，使用 h' 来计算每一块并把结果连接作为 x_{i+1}。

当 x_j 的长度等于 t 时，停止计算。然后将 x 的长度 n 的二进制串使用定理 3.1 得到一个 t 比特的块 len_x。最终有

$$h(x) = h'(x_j \| len_x)。$$

对于这种并行方式的抗碰撞性的证明与前一定理证明类似。如果找到 h 的碰撞对，我们有：如果 $x'_{j'} \| len_{x'} \neq x_j \| len_x$，那么就得到 h' 的碰撞，这与定理 3.1 矛盾。因此，假设 $n = n'$。现在，$x \neq x'$ 意味着可以找到 i 使得 $x_i \neq x'_i$ 但是 $x_{i+1} \neq x'_{i+1}$。显然，这是 h' 的一个碰撞。

MD 构造方式是目前应用最多的模式，各种常用的 Hash 函数都采用了 MD 方式。但是，MD 方式并不是完美的。第七章将会介绍对 MD 方式的攻击方法。

3．1．3 可扩展 Hash 构造方式

首先介绍可扩展性的概念。可扩展性是指这样一种性质，在函数作用于一个特定的消息之后，当消息被修改时可以很容易得到新消息的 Hash 值，而不需要重新进行计算。具体来讲，设对消息 M 做了 Hash，且其 Hash 值为 x。具有可扩展性的 Hash 函数 H 的修改函数 F 可以以比直接重新计算 M' 更短的时间内得到更新的 Hash 值 x'。更新所需要的时间与消息的长度无关，而只与修改的消息长度有关[8]。

可扩展的构造方式适用于对经常做改动或者只有少量差异的消息集合做 Hash 的情况。比如，要对硬盘上的文件系统做 Hash，当某个文件被改动时，我们并不希望重新对整个硬盘做 Hash 来得到新的 Hash 值。如果 Hash 函数具有可扩展性，那么就可以做一个很简单的更新操作来得到新的 Hash 值。

显然，MD 线性迭代方式因为迭代的原因不可能具有可扩展性。对于并行的树型方式，如果要具有可扩展性就必须保留全部的中间变量并且计算并不简单。Bellare 等人对此问题做了研究，并给出了一个可扩展 Hash 函数的构造方案。下面来看具体的方案：

本方案上层的结构非常简单。设 x 是一个消息块序列，$x = x_1, x_2, ... x_n$，其中每个消息块的长度为 b 比特，其中 b 是任意选取的一个参数。首先，使用函数 h 来处理每一块 x_i 并得到输出 y_i。（特别的，$y_i = h(<i> \| x_i)$），其中 i 是块索引的二进制表示）。随后，这些输出以某种方式混和来生成最终的 Hash 值 $y = y_1 \otimes y_2 \otimes ... \otimes y_n$，其中 $\otimes$ 表示混和运算。

这里，称 h 为随机变换函数。实际中，h 可以从某些标准 Hash 函数如 SHA-1 等导出的，将其看成完全安全的 Hash 函数或者随机函数。混和函数⊗是一个典型的群操作，意味着将 $y_1 y_2 ... y_n$ 看做交换群 G 的的元素，且该群的运算为⊗。

这个方案称为随机－混和方案。其结构如下图所示：

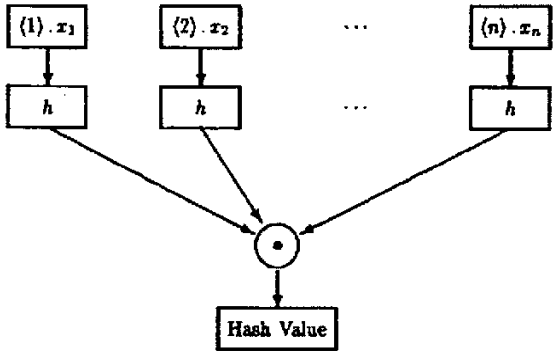

图 3.1

简单的表达式为：

$$HASH^{h}_{<G>}(x_1...x_n) = \otimes^{n}_{i=1} h(<i>\| x_i)\ ;$$

显然，这种方案最大的优点是可扩展性。比如某个消息块 x_i 变成了 x'_i，那么新的 Hash 值就是 $y \otimes h(<i>\| x_i)^{-1} \otimes h(<i>\| x'_i)$，其中 $()^{-1}$ 表示群的逆运算，y 是原来的 Hash 值，即 x 的 Hash 值。另外因为群运算是可结合可交换的，这个方案同时也是可并行的。

在介绍具体的方案之前，先来看各组成部分的安全性要求。函数 h 的输入是固定长度的，因此可以将其看成压缩函数。为了保证方案的安全性，必须保证 h 的抗碰撞性。否则，整个方案则无安全性可言。在消息前面添加索引的原因是为了防止重排攻击，如果没有索引号，那么任意交换两块消息的位置就构成一个碰撞对。至于混和函数，同样也要保证其抗碰撞性。

通过选择不同的群，可以得到不同的可扩展抗碰撞 Hash 函数。这里只介绍两个比较典型的算法：

◆ **MuHASH**

MuHASH是乘法Hash的简称。这里将混和运算设成群G的乘法运算，其中

群G中离散对数问题是困难的。比如，取$G = Z_p^*$，其中p为某个合适的质数。计算公式为：

$$MuHASH_{G,h}(x_1,...x_n) = \prod_i h(< i > \| x_i)$$

这种方案的计算时间主要集中在乘法阶段，虽然乘法运算的效率可能不高，但可以通过某些方法使得其效率在可接受的范围内。

主要来看它的安全性。可以证明，MuHASH的抗碰撞性等价于G中的离散对数问题。事实上，如果可以构造出MuHASH的碰撞对，那么就可以找到解决G中离散对数问题的方法。也就是说，MuHASH的安全性是基于离散对数问题的，与攻击者的手段无关。在实际中，MuHASH的安全性可能比理论上的更高。这是因为，至今为止即使在能解决某些离散对数问题的情况下，也还没有找到攻击MuHASH的方法。

◆ **AdHASH**

AdHASH，即加法Hash，它采用大整数M的模加运算来作为混和运算。其表达式为：

$$AdHASH_{M,h}(x_1,...x_n) = \sum_i h(< i > \| x_i) \bmod M$$

使用加法来代替乘法显然会带来效率上的提高。Hash过程只包括n个模加运算，扩展只需要两个模加运算。事实上，AdHASH的速度接近标准Hash函数，并具有可扩展和可并行的优势。

同样，AdHASH的安全性也是可以证明的。它的可碰撞性等价于背包问题。另外，只要h是抗前像攻击的，那么可以证明AdHASH同样是抗前像攻击的。

总的来说，AdHASH不管是在效率上还是安全性方面，都是非常有吸引力的。

3．2　压缩函数的设计

有了安全的构造模式之后，Hash 函数的研究工作就可以主要集中在压缩函数的设计上。如果能设计出足够安全的压缩函数，那么安全的 Hash 函数就可以基于上面的模式构造出来。目前，压缩函数的设计方式主要有三种：由分组密码

构造，直接设计，基于代数结构构造。

3．2．1　由分组密码构造

由分组密码构造 Hash 函数有很多原因。首先是历史的原因，DES 是最早被广泛接受的商用密码基础模块标准，所以在此基础上构造 Hash 函数就成为很自然的事。其次是为了设计和实现的方便，高效而且安全的分组密码和 Hash 函数的设计都是很困难的。另外，现存分组密码的软件和硬件实现也可以重新利用，这样又可以降低成本。最重要的是，分组密码经过了大量的研究，我们对其安全性有了一定的了解，这样我们对 Hash 函数安全性的信任可以建立在对分组密码的信任基础上。但是，这种方法也有其不足之处。最主要的是，相对于直接设计的 Hash 函数，这种方式的效率很低，因为分组密码每轮都需要密钥变换。还有就是有些本身是安全的分组密码在用于 Hash 函数时常会表现出意想不到的问题。

下面讨论基于分组密码构造 Hash 的各种方式，在讨论中，将加密操作写成 $Y=E_K(x)$。其中，x 表示明文，Y 表示密文，K 表示密钥。明文和密文的长度是分组密码的分块长度用 b 表示。密钥长度用 k 表示。这些参数的典型值如 des：b＝64，k=56，AES：b=128，k=128。用 n 表示 Hash 函数的输出长度。建立在分组密码基础上的 Hash 函数有一个 Hash 比率的概念，即每次加密操作可以处理的消息块的个数。

按照 n＝b，n=2b,n>2b 来对构造方式进行分类。这样分类的原因是大多数的分组密码的分组只有 64 位或 128 位，新的如 AES 则有 128 位，这个长度对于 Hash 函数来说是远远不够的，为了抗碰撞一般至少要两倍的分组长度。

3．2．1．1　单倍长度构造

单倍长度构造的 Hash 函数其输出长度等于分组密码的分组长度，也就是说 Hash 函数的输出长度 n＝b。这种构造方式是将压缩函数的输入做线性变换作为分组密码的输入，并通过不同的组合来得到多种构造方式。

设压缩函数为 f，输入变量为 H_{i-1}，消息块为 X_i，输出为 H_i，则有

$H_i = (X_i, H_{i-1})$。

设分组密码的输入变量为明文 P 和密钥 K。这两个变量可以从以下四个值中取值：$(X_i, H_{i-1}, X_i \oplus H_{i-1}, V)$（V 是常数）。另外，可以修改分组密码的结构，加入一个新变量前像反馈 FF 与原输出做异或运算，这样就可以得到 $4^3=64$ 种不同的方案。其结构如图所示：

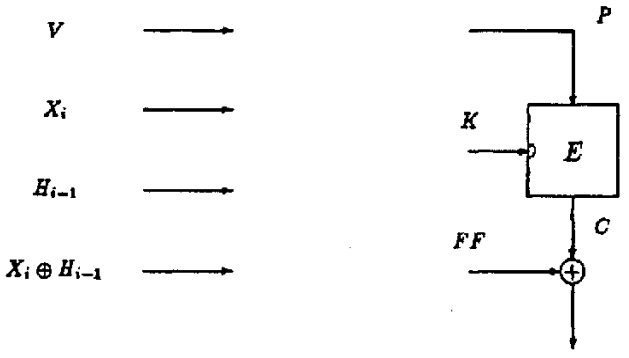

图 3.2

在文献[9]中，作者对这 64 种方案的安全性做了分析。文章没有对这些方案做安全性证明，而是分析了这些方案对几种已知的攻击方法的抵抗能力。这些攻击方法包括前像攻击、后向攻击、置换攻击和直接攻击等。经过分析，其中的 12 种方案能够抵抗所有这些攻击方法，而其它的方案都不能抵抗其中的至少一种攻击。因此，作者认为这 12 种方案是安全的构造方式。

在文献[10]中，作者基于黑盒子模型对这 12 种方案的安全性进行了证明。并同时证明了对于另外 8 种只容易受到后向攻击的方案也是具有抗碰撞安全性的，但它们不具有抗前像攻击安全性。

3．2．1．2　双倍长度构造

双倍长度构造的 Hash 函数其输出长度两倍于分组密码的分组长度，也就是说 Hash 函数的输出长度 n=2b。将双倍长度的构造方式按其 Hash 比率进行分类讨论。

- **Hash 比率为 1 的构造方式**

设 $H_i=(H_i^1,H_i^2),M_i=(M_i^1,M_i^2)$ 则 Hash 比率等于 1 的压缩函数可表示为：

$$\begin{cases}H_i^1=f^1(H_{i-1}^1,H_{i-1}^2,M_i^1,M_i^2)\\H_i^1=f^2(,H_{i-1}^1,H_{i-1}^2,M_i^1,M_i^2)\end{cases}$$

或

$$\begin{cases}H_i^1=f^1(H_{i-1}^1,H_{i-1}^2,M_i^1,M_i^2)\\H_i^1=f^2(H_i^1,H_{i-1}^1,H_{i-1}^2,M_i^1,M_i^2)\end{cases}$$

其中，f^1,f^2 分别包含一次加密操作，第一种方式称为并行方式，第二种方式称为串行方式。

当采用的分组密码的密钥长度等于分组长度时[11, 12]，可以证明比率为 1 的构造方式中安全性最高的前像攻击的复杂度为 $2^{n3/2}$，碰撞攻击的复杂度为 $2^{n3/4}$。

当采用的分组密码的密钥长度等于分组长度两倍时[13]，攻击方法与长度相等时略有不同，但是同样可以证明其前像和二次前像攻击的复杂度应该小于等于 $2^{m3/2}$，碰撞攻击的复杂度应该小于等于 $2^{n3/4}$。

目前存在的分组密码大都属于以上的两种形式，因此，以上结果说明完全安全的 Hash 比率为 1 的构造方式是不存在的。要构造完全安全的 Hash 函数必须降低 Hash 比率或者采用密钥长度与分组长度比率更大的的分组密码。当然，如果 $2^{m3/4}$ 的复杂度在实际应用中可以接收的话，也可以考虑这种构造方式。

- **Hash 比率小于 1 的构造方式**

Hash 比率小于 1 的构造方式有很多，但由于效率太低，在实际应用中并没有很大的前景，因此大部分的构造方式只有理论价值[14~17]。

以这种方式设计的方案有 MDC-2 和 MDC-4，它们是由 B.Brachtl 等人设计[18,19]。MDC-2 用了两个并行的 MMO 方案。C_L,C_R 分别代表值 C 的左半部分和右半部分值，H 表示中间变量，X 表示消息块。MDC-2 的压缩函数可以描述成：

$$H_{i+1}\|\tilde{H}_{i+1}=f(H_i\|\tilde{H}_i,X_i),$$

其具体计算过程如下：

$$C_{i+1}=E_{z(H_i)}(X_i)\oplus X_i,$$

$$\tilde{C}_{i+1} == E_{z(H_i)}(X_i) \oplus X_i, \| \tilde{H}_{i+1} = f(H_i \| \tilde{H}_i, X_i),$$

$$H_{i+1} = C_{i+1}^{L} \| \tilde{C}_{i+1}^{R},$$

$$\tilde{H}_{i+1} = \tilde{C}_{i+1}^{L} \| C_{i+1}^{R}$$

注意到，如果省略掉左右两半部分的交换，那么两条链将是互相独立的，攻击者可以各个击破，从而失去了加倍长度的优势。最后没有输出变换，输出长度等于中间链变量的长度。

MDC-4的压缩函数包含两个顺序的MDC-2压缩函数。对于第二个MDC-2压缩函数，密钥由第一个压缩函数的链变量输出获得，明文输入是前一个MDC-4压缩函数相反一边的输出。

对MDC-2和MDC-4已知的最好的前像攻击复杂度分别是$2^{3n/4}$,2^n，碰撞攻击复杂度为$2^{n/2}$。对MDC-2的前像攻击说明这种构造并不具有完全的安全性。另外，MDC-2的伪前像攻击和伪碰撞的复杂度分别是$2^{n/2}$和$2^{n/4}$。MDC-4的伪碰撞攻击复杂度是$2^{3n/8}$，对于以DES为基础构造的MDC-2和MDC-4，攻击更加容易。MDC-2和MDC-4的Hash比率分别是1/2和1/4。

3．2．1．3　高效的基于分组密码构造方式

注意到基于分组密码构造的Hash函数每一次迭代的密钥都是不同的。分组密码通常都包含一个密钥编排算法，该算法将密钥转化成一系列的子密钥用于每一轮加密操作中。当分组密码用于加密时，只有在更换新的加密密钥时才需要做一次密钥编排。但当分组密码用于Hash函数时，每一次算法调用都要重新进行一次密钥编排，这样就大大降低了算法的效率，特别对那些密钥编排算法很慢的分组密码更是一个问题。

如果可以在对消息做Hash计算过程中，保持密钥不变或者只采用较小的密钥集合，那么密钥编排工作就可以事先完成，从而可以很大程度的提高运算速度。我们称这种固定密钥并且每个分组只调用一次加密操作的构造方式为高效的基于分组密码hash构造方式。

在文献[20]中，作者对这种构造方式进行了全面的讨论并得出结论：在理想分组密码模型下，构造可证明安全的迭代式高效 hash 函数是不可能的。下面简单介绍证明过程。

将高效的基于分组密码构造的压缩函数的通用模型定义为：

$$f(h_{i-1},m_i)=f_2(h_{i-1},m_i,E_K(f_1(h_{i-1},m_i)))$$

其中，$f_1:\{0,1\}^n\times\{0,1\}^n\to\{0,1\}^n$，$f_2:\{0,1\}^n\times\{0,1\}^n\times\{0,1\}^n\to\{0,1\}^n$是任意的简单变换，K 取自一个固定的密钥集合。

作者证明如上述结构的压缩函数不能构造可证明安全的迭代函数。对于任意的函数 f_1 和 f_2，攻击者可以至多$|K|(n+\lceil\lg(n)\rceil)$次加密操作找到函数 H_f的碰撞。特别的，对于某些一般的函数，比如异或，可以以 2|K|次加密操作找到碰撞。

攻击算法主要利用了函数 f_1的性质，因为加密只需要加密 f_1的结果，而 f_1是 $2n\to n$的简单函数，因此值域里的每个值都会有若干个前像，并且这些前像是很容易计算的。这样，在 f_1映射下有相同值的（h，m）利用同一次加密，在通过 f_2之后可以得到多个不同的 Hash 值，从而降低了加密运算次数。具体过程是使用贪心算法构造一颗 Hash 树。将压缩函数的中间向量作为树节点，由一个中间向量到另一个中间向量的消息作为树枝。从 h_0 节点开始，每次选择可以增加最多节点的加密值来做加密操作，并且每次加密操作使得树的深度增加一层，可以证明，当树的深度达到$n+\lceil\lg(n)\rceil$时，必然存在某一层节点的数目大于等于 2^n。这样，根据生日攻击原理，就可以以很高的概率找到碰撞对。

这个攻击并不是一个实际可操作的攻击，因为攻击算法其实是需要 2^n 的操作次数和存储空间的。意义在于说明基于分组密码的高效 hash 函数构造方式在黑盒子模型下不是可证明安全的。

针对这种攻击我们可以提出一种近似的高效构造方式，其定义仍采用上面的模型，不同的是密钥 K 的选择算法，我们重新定义 K 的选择算法 g:

设密钥$K\in\{0,1\}^k$，$S\in\{0,1\}^*$是某个状态变量，计数器$n\in[0,...,(n+\lceil\lg(n)\rceil)]$，n 的初始值为 0，则算法 g 的描述为：

$$S \times \{0,1\}^n \times \{0,1\}^n \to S \times K$$
$$n \leftarrow (n+1) \bmod (n + \lceil \lg(n) \rceil)$$

即算法每隔 $n+\lceil \lg(n) \rceil$ 次，重新做一次密钥编排，而不是采用 K 集合里的值。这样，上述攻击算法就不能成功进行，从而也是可证明安全的了。但是这种构造方式的效率将低于原来的构造方式。

另外，虽然这种高效的构造方式在黑盒子模型下是非可证明安全的，但因为不存在实际的攻击，因此也可能作为一种构造方式考虑。

3．2．2 直接设计

直接设计顾名思义是从头开始专门设计 Hash 函数。这就意味着可以摆脱利用分组密码或者某些数学问题所带来的限制，从而可以设计更高效的算法。

在实际运用中最受关注的直接设计的 Hash 函数是 MD 家族。MD 系列采用的是多轮非线性函数构造方式。这种构造方式曾经一度很受推崇，但是最近的一系列攻击说明，它并不能达到预期的目的。特别是对于差分攻击，这种方式的抵抗力非常弱。虽然可以通过增加轮数、增强非线性函数的非线性性、改变消息编排等来增加其强度，但实践证明这并不是一个好的设计结构。

针对 MD 系列的缺陷，Biham 等人认为 Hash 函数的设计应该尽量采用分组密码的设计理念和准则，比如 S 盒、线性变换、变化的循环移位、快速有效的混和等等。另外，分组密码的安全分析技术也应该用于 Hash 函数的安全分析，比如对线性攻击、代数攻击的抵抗能力等。又因为 Hash 函数有其自身的特点，还要注意某些对于分组密码并不构成足够威胁的攻击方法，比如相关密钥攻击等等。目前，采用这种方式设计的 Hash 函数比较少，有 Whirlpool 和 tigher 等；tiger 是针对 64 位处理器的，有 512 比特的输出长度[21]。Whirlpool[22]是欧洲的建议标准，输出长度是 512 比位，内部结构的设计思想类似 AES 的设计理念。这两种算法目前还没有有效的攻击算法。

3．2．3 基于某些代数结构的 Hash 函数

- 基于模运算的 Hash 函数

这种 Hash 函数的设计是利用数字签名算法中用到的模运算硬件来实现 Hash 函数。其中的一些设计方法可以将其安全性建立在某些数学难题上，并且有很好的可扩展性。不足之处是原始函数有丰富的数学结构，攻击者可以利用同态的结构和指数模运算的不动点来攻击函数。另外，还必须要保证不能有太小的输入。这种设计方法可以分为两类，一类是安全性不归约到某些数学难题上，这种方法通常有很高的效率，但是其安全性不能保证，很多类似的算法都被攻破了，另一类是可以归约到某些数学难题上，但这种方法往往效率很低。

- 没有归约的设计方案

大部分的这种设计都用到一个两个素数乘积 N 作为模，N 的规模一般在 512 到 1024 比特之间。这种 Hash 函数与 RSA 组合可以设计成很好的数组签名算法。但是，这种方案存在一个实际的问题：产生模数 N 的人知道 N 的分解，那么相对于其他的 Hash 函数的用户来说，他就有潜在的优势可以利用。如果可以设计一种算法，知道模数分解对攻击该算法并没有帮助，那么就可以解决这个问题，但这种算法往往很难设计。另一个解决办法是让可信任的第三方来生成模数，或者利用多方安全计算来生成，但这样又会增加 Hash 函数应用的复杂性。

其中效率最高的一个方案是建立在模平方基础上，形式如下：

$$H_i = (x_i \oplus H_{i-1})^2 \bmod N \oplus H_{i-1},$$

这种方案对输入消息一定要加入冗余才能保证其安全性。加入冗余的比较成功的算法是 MASH-1，形式如下：

$$H_i = ((x_i \oplus H_{i-1}) \vee A)^2 (\bmod N) \oplus H_{i-1}$$

其中 A＝0xF000..00，输入 x_i 的每一字节的最高四位都被设成 1111，平方运算的输出被削减到 n 比特。随后加入了一个复杂的输出变换，其中包括很多压缩函数的应用，目的是尽量破坏其中的数学结构。最后的输出结果最多为 n/2 比特。对 MASH-1 最有效的前像象攻击和碰撞攻击的复杂度分别为 $2^{n/2}$ 和 $2^{n/4}$。MASH-2 是 MASH-1 的变种，其指数是 2^8+1，安全性有所提高[23][24]。

第四章　MD 构造方式中压缩函数与 Hash 函数抗碰撞性的关系

第三章已经介绍了 MD 构造方式，并简单的讨论了它的某些特性。这里，我们着重分析 MD 构造方式中压缩函数的抗碰撞性与 Hash 函数抗碰撞性之间的关系。

由第二章 Hash 函数性质介绍可知，压缩函数的抗碰撞性可分为三种，包括抗自由碰撞性、抗半自由碰撞性和抗碰撞性。对于 Hash 函数，由于其初始向量取固定值，因此只有抗碰撞性的要求。本章主要讨论在 MD 模式下构造抗碰撞的 Hash 函数对压缩函数安全性质的要求。

在具体内容之前先对符号做规定。设 f 为压缩函数，f 的输出长度为 n 比特，H 是 f 以 MD 迭代方式构造的压缩函数。另外，在本章中 Hash 函数默认为以迭代结构 MD 构造方式构造的 Hash 函数

4．1　压缩函数的抗半自由碰撞性与 Hash 函数的抗碰撞性

引理 4.1：对于压缩函数 f，如果存在一个复杂度为 2^s 的半自由碰撞攻击，那么 Hash 函数 H 就存在一个复杂度为 $2\times 2^{\frac{s}{2}+\frac{n}{4}}$ 的碰撞攻击。

证明： 采用中间攻击的方法找到两块长度的消息 $M=m_1m_2$ 和 $M'=m_1m'_2$ 使得 $H(M)=H(M')$。设 H 的初始向量为 h_0。

首先，随机选取 $2^{\frac{n+s}{2}}$ 个 m_1，并计算 $s_1=f(h_0,m_1)$；然后找到 $2^{\frac{n-s}{2}}$ 个碰撞对 (s_1',m_2) (s_1',m_2') 和 $f(s'_1,m_2)=f(s_1',m_2')$，其中 s_1' 是随机的。如果找到 $s_1=s_1'$，那么攻击成功有 $H(M)=H(M')$。根据生日攻击原理，这个攻击成功的概率为 0.63。因为两步的复杂度都是 $2^{\frac{n+s}{2}}$，因此整个攻击的复杂度为 $2\times 2^{\frac{n+s}{2}}$。

由引理可知，对于存在半自由碰撞的压缩函数，由其构造的 Hash 函数不具有抗碰撞性。因此，为了构造安全的 Hash 函数，必须要求压缩函数具有抗半自由碰撞性。

4．2　压缩函数抗自由碰撞性与 Hash 函数的抗碰撞性

对于抗碰撞性，Merkle 和 Damgard 分别证明了如果存在一个对 Hash 函数的碰撞攻击，那么就可以构造一个对压缩函数的自由碰撞攻击；并且后者的复杂度等于前者。因此，可以认为，如果不存在有效的对压缩函数的自由碰撞攻击，那么就不存在由此压缩函数构造的 Hash 函数的碰撞攻击。但是，反过来，如果压缩函数不具有抗自由碰撞性，由它构造的 Hash 是否有可能有抗碰撞性呢？如果可能，压缩函数是要满足什么样的性质呢？表面上看，对压缩函数的自由碰撞攻击似乎并不能直接应用于对 Hash 函数的碰撞攻击。在文献[10]中，J.Black 证明了一组基于分组密码构造的 Hash 函数具有抗碰撞性，虽然他们的压缩函数是不能抵抗自由碰撞攻击的。我们的研究发现：如果对压缩函数的自由攻击受限于某些特定的条件，那么由此构造的 Hash 函数是能够抵抗碰撞攻击的。我们的证明通过对这两个攻击成功概率的归约来说明，对压缩函数的自由碰撞攻击并不能给 Hash 函数的碰撞攻击带来很多用处。

设 A 是对压缩函数 f 的自由碰撞攻击算法，A 可以在没有任何输入的条件下输出一组碰撞对 (h,m) 和 (h',m')，即有 $f(h,m)=f(h',m')$。我们不关心 A 的具体攻击过程。用 f 的计算次数来定义 A 攻击过程的复杂度。

首先假设 A 输出的碰撞对的中间变量都是随机的。这个假设是合理的，因为压缩函数通常都是被认为是伪随机或者包含伪随机构件的，并且攻击者的能力是有限的。差分攻击就是满足这种假设的攻击算法。在差分攻击中，攻击者只能找到两个具有某个特定差分的碰撞对，但并不能任意选择具有指定中间变量的碰撞。基于对 A 的假设，有下面的定理：

定理 4.1：假设压缩函数 f 不具有抗自由碰撞性并且对其最有效的自由碰撞攻击为算法 A。如果能以 2^s 的复杂度找到碰撞对（碰撞对的中间变量是随机的），那么基于算法 A 找到由 f 构造的 Hash 函数的碰撞对的复杂度至少为 $2^{\frac{n+2s-1}{2}}$。

证明：设算法 B 是对 Hash 函数 H 的碰撞攻击，B 利用算法 A 来实现其攻击。设 h_0 是 H 的初始变量。定义一个直接图 $G=(V_G,E_G)$，G 的点集为

$V_G = \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n$，每个点记为 (h,x,y)。边属于 E_G 当且仅当两点 (h,x,y)，(h',x',y') 有 $y = h'$。

在算法 B 调用 A 的执行过程中，按如下规则向图 G 中加入着色的点：

-- 初始情况下，G 是空集。

-- 当 B 调用 A，A 返回碰撞对 (h,x,y) 和 $(h,'x',y')$ 时，如果这两个点没有出现在 G 中，那么将其添加到 G 中。如果 $h = h_0$，那么将点 (h,x,y) 着成红色；否则将其着成黑色。如果新加入的点能够连接到其它点，那么将这条边加入到 G 中。

下面给出另外的几个定义：对于两点 (h,x,y) 和 $(h,'x',y')$，如果 $y = y'$，则称这两点是碰撞的。两条路径 P和P'，如果它们都开始于红色点并且结束于碰撞的点，那么称这两条路径是碰撞的。如果 B 找到两条碰撞的路径，则算法输出这两条路径。设 C 表示算法 B 找到两条碰撞路径的事件。攻击过程如下所示：

Algorithm $B(H^f, A)$

$G \leftarrow \varepsilon$

While there is no colliding path (P,P') in G

(h,x,y) , $(h,'x',y')$ $\leftarrow B(f)$

if (h,x,y), $(h,'x',y')$ does not exists in G, add into G

if $h = h_0$ then color it red else color it black

if $h' = h_0$ then color it red else color it black

for all vertices (p,q,g) in G

if $p = g$ then add arcs $(h,x,y) \rightarrow (p,q,g), (h',x',y) \rightarrow (p,q,g)$

if $q = g$ then add arc $(p,q,g) \rightarrow (h,x,y)$

if $h' = g$ then add arc $(p,q,g) \rightarrow (h',x',y)$

end for

end while

return (P,P')

注意到 B 输出的两条碰撞路径并不一定是相同长度的，这就意味着在 MD 加强的构造方式下，B 的输出并不一定是真正的碰撞。但是，我们知道找到相同长度的碰撞的复杂度至少大于等于不同长度的碰撞；因此，碰撞攻击的复杂度应该大于等于 B 的复杂度。

下面证明如果 B 可以调用 A 至多 q 次，那么 B 成功的概率最多为 $\frac{2q^2}{2^n}$。

设 C_i 为 事件 C 在第 i 次调用 A 时发生，并设 C_0 为空事件。那么有 $\Pr[C]=\sum_{i=1}^{q}\Pr[C_i \mid \bar{C}_{i-1}\wedge\ldots\wedge\bar{C}_0]$。给定事件 $\bar{C}_{i-1}\wedge\ldots\wedge\bar{C}_0$ ，事件 C_i 可能在四种情况下发生。为了更清楚的讨论这些情况，我们定义另外几个概念。设 $Arc(i,j)$ 表示事件 G 中存在点 (h_i,x_i,y_i) 和 (h_j,x_j,y_j)，并且有 $y_i=h_i$；设 $\mathrm{Re}d(i)$ 表示事件 (h_i,x_i,y_i) 被着成红色； 设 $Collide(i,j)$ 表示事件 G 中存在点 (h_i,x_i,y_i) 和 (h_j,x_j,y_j)，并且 $y_i=y_j$。

Case 1: 在 A 的第 i 次调用后加入了红色点 (h_i,x_i,y_i) ，并且 G 中存在弧 $(h_i,x_i,y_i)\rightarrow(h_j,x_j,y_j)$，其中 (h_j,x_j,y_j) 是在 A 的第 j 次调用时加入 G 的，$j<i$。根据对算法 A 的限制，$h\neq h'$，那么每次只能往 G 中加一个点。这样发生事件 C 就必须要有事件 $Arc(i,j)$ 和 $\mathrm{Re}d(i)$ 成立。用如下的式子计算 C 的概率：

$$\Pr[Arc(i,j)\wedge\mathrm{Re}d(i)]=\Pr[Arc(i,j)\mid\mathrm{Re}d(i)]\Pr[\mathrm{Re}d(i)]$$

$$\leq\Pr[\mathrm{Re}d(i)]\leq\frac{2}{2^n}$$

第二个不等式成立是因为 A 输出的中间变量是随机的，并且需要其中的一个满足条件 $h=h_0$。

Case 2: 在 A 的第 i 次调用后加入了两个黑色点 (h_i,x_i,y_i) 和 (h_i',x_i',y_i)，并且 G 中存在弧 $(h_j,x_j,y_j)\rightarrow(h_i,x_i,y_i),(h_r,x_r,y_r)\rightarrow(h_i',x_i',y_i)$，其中 (h_j,x_j,y_j) 和 (h_r,x_r,y_r) 分别是在第 i 次和第 j 次调用 A 时添加进去的， $j,r<i$ 。显然，

新加入的两个点是碰撞对，并且他们分别是两条碰撞路径的末端点。这种情况需要有事件 $Arc(i,j)$ 和 $Arc(i',r)$ 成立，按如下方式计算 C 的概率：

$$\Pr[Arc(i,j)\wedge Arc(i',r)]=\Pr[Arc(i,j)\mid Arc(i',r)]\Pr[Arc(i',r)]$$

$$\le \Pr[Arc(i',r)]\le \frac{i-1}{2^n}$$

第二个不等式成立的原因是在第 i 次调用 A 之前，G 中最多存在 $i-1$ 个不同的中间变量。

Case 3: 在第 i 次调用 A 后至少加入了一个黑色点，并且图 G 中存在弧 $(h_j,x_j,y_j)\rightarrow(h_i,x_i,y_i)$ 和点 (h_r,x_r,y_r)，其中 (h_j,x_j,y_j) 和 (h_r,x_r,y_r) 分别是在第 i 次和第 j 次调用后加入的点，$j,r<i$。这种情况需要事件 $Arc(i,j)$ 和 $Collide(i,r)$ 成立，按如下方式计算 C 的概率：

$$\Pr[Arc(i,j)\wedge Collide(i,r)]=\Pr[Arc(i,j)\mid Collide(i,r)]\Pr[Collide(i,r)]$$

$$\le \Pr[Collide(i,r)]\le \frac{i-1}{2^n}$$

第二个不等式成立的原因是在第 i 次调用 A 之前，G 中最多存在个 $i-1$ 不同的中间变量。

Case 4: 在第 i 次调用 A 后至少加入了一个黑色点，并且图 G 中存在弧 $(h_j,x_j,y_j)\rightarrow(h_i,x_i,y_i)$ 和 $(h_i,x_i,y_i)\rightarrow(h_r,x_r,y_r)$，其中 (h_j,x_j,y_j)，分别是在第 i 次和第 j 次调用后加入的点，$j,r<i$。这种情况需要事件 $Arc(i,j)$ 和 $Arc(i,r)$ 成立，按如下方式计算 C 的概率：

$$\Pr[Arc(j,i)\wedge Arc(i,r)]=\Pr[Arc(i,r)\mid Arc(j,i)]\Pr[Arc(j,i)]$$

$$\le \Pr[Arc(i,j)]\le \frac{2(i-1)}{2^n}$$

第二个不等式成立的原因是在第 i 次调用 A 之前，G 中最多存在个 $i-1$ 不同的中间变量，并且在第 i 次调用 A 时 A 输出了两个不同的中间变量 h 和 h'。

我们忽略了一次调用加入的两个点分别连接两条碰撞路径的情况，因为这种情况发生的概率相比上述情况是非常小的。

总结所有上述情况，有

$$\Pr[c] \leqslant \sum_{i=1}^{q} \frac{2}{2^h} + \frac{4(i-1)}{2^n} = \frac{2q^2}{2^n}$$

另外，每输出一个碰撞对 A 需要进行 2^s 次压缩函数 f 的计算，如果 B 只能做 q 次 f 计算，那么 B 成功的概率至多为 $\frac{2q^2}{2^n 2^{2s}}$。因此，B 的攻击复杂度至少为 $2^{\frac{n+2s-1}{2}}$

证明完毕。

下面来讨论当 S 取不同值时 B 的复杂度的取值情况。

如果 $s \geq 1$，那么 B 的复杂度要高于强力攻击，这是没有意义的。在这种情况下，自由碰撞攻击算法 A 并不能给对 Hash 函数的碰撞攻击带来方便。也就是说，如果某个压缩函数存在这种类型的自由碰撞攻击，并不会影响由其构造的 Hash 函数的抗碰撞性。

如果 $s = 0$，此时 A 可以以一次 f 计算找到自由碰撞对，这时 B 的复杂度只有常数量级的减少。这种情况的例子是单块分组密码构造 Hash 中的一类受到后向攻击的构造方式。

如果 $s < 0$，此时每计算一次 f 可以得到 2^s 个碰撞对，这时 B 的复杂度就减少了 2^s，B 利用算法 A 确实能降低攻击复杂度。这种情况的例子是双倍长度的分组密码构造 Hash 并且 Hash 比率为 1 的一些构造方式。在这些构造方式里，利用输入直接的线性关系，一次计算可以得到多个自由碰撞对。

由上面的讨论可以看到，除非算法 A 有足够大的能力，计算 f 一次得到多个自由碰撞对，否则 A 并不会给 Hash 函数的抗碰撞性带来影响。因此，对于压缩函数的安全性，可以不要求它一定具备抗自由碰撞性。

假设 A 输出的两个中间变量都是随机的可能会太强了，下面来降低对算法 A 的假设，增加其攻击能力，然后在新的假设下重新计算 B 的复杂度。

假设 1：A 可以自由选择碰撞对中的其中一个中间变量。让 A 可以同时自由选择两个自由变量是没有意义的。因为，如果这样，A 就可以选择让两个中间变量同时等于 h_0 那么就得到了 H 的碰撞。

把算法 B 的过程稍微做一下改动。当 B 在任意循环中调用 A 时，它给 A 一个指定的中间变量。也就是说将在 while 循环里的第一句改成 $(h_B,x,y),(h',x',y)\leftarrow A(f,h_B)$。看起来这个改动可以给 B 很大的帮助，因为现在 B 可以选择它需要的中间变量。但事实并非如此，下面可以证明，B 的复杂度只有常数级的减少。

推论 4. 1 在假设 1 条件下，如果 A 输出 f 的自由碰撞对的复杂度为 2^s，则利用 A 对 Hash 函数 H 的碰撞攻击的复杂度至少为 $2^{\frac{n+2s-2}{2}}$。

证明：Case 1: 为了提高成功概率， B 可以选择 $h_B=h_0$。但是 B 不能控制压缩函数结果值 y。因此这种情况下要求事件 $Arc(i,j)$ 成立，有：

$$\Pr[Arc(i,j)]\leq \frac{2(i-1)}{2^n}$$

Case 2: B 可以选择 h_i 或者 h_r，但是因为它不能选择另一个中间变量，因此，这种情况下要求事件 $Arc(j,i)$ 或者 $Arc(r,j)$ 成立，有：

$$\Pr[Arc(i,j)]\leq \frac{2(i-1)}{2^n}$$

Case 3: 如 case1

$$\Pr[Arc(i,j)]\leq \frac{2(i-1)}{2^n}$$

Case 4: 如 case3

$$\Pr[Arc(i,j)]\leq \frac{2(i-1)}{2^n}$$

总结所有情况，有

$$\Pr[C]\leq \sum_{i=1}^{q} 4\times\frac{2(i-1)}{2^n}=\frac{4q(q-1)}{2^n}$$

假设 2: 假设 A 有更强的攻击能力，给定任意的 $y\in\{0,1\}^n$，A 可以输出两个随机对 $(h,m),(h',m')$，有 $f(h,m)=f(h',m')=y$。

同样，也需要把算法 B 的过程稍微做一下改动。当 B 在任意循环中调用 A 时，它给 A 一个指定的 hash 结果值 y。也就是说将在 while 循环里的第一句改成 $(h_B,x,y),(h',x',y)\leftarrow A(f,y)$。下面来分析这种情况下，B 的攻击复杂度的变化。

推论 4.2 在假设2条件下，如果A输出f的自由碰撞对的复杂度为2^s，则利用A对Hash函数H的碰撞攻击的复杂度至少为$2^{\frac{n+2s-1}{2}}$。

证明：Case 1: 这种情况下B没有选择开始向量的能力，因此要求事件$\mathrm{Re}d(i)$成立：

$$\Pr[\mathrm{Re}d(i)] \le \frac{2}{2^n}$$

Case 2: A增加的攻击能力在这种情况下并没有用处，因此成功概率没有变化：

$$\Pr[Arc(i,j) \wedge Arc(i',r)] \le \frac{i-1}{2^n}$$

Case 3: B可以选择y使得y与G中的某个点碰撞，但是B不能预测开始向量。因此要求事件$Arc(i,j)$成立：

$$\Pr[Arc(i,j)] \le \frac{2(i-1)}{2^n}$$

Case 4: 与case3类同：

$$\Pr[Arc(i,j)] \le \frac{2(i-1)}{2^n}.$$

总结上述所有的情况，有

$$\Pr[c] \le \sum_{i=1}^{q} \frac{2}{2^h} + \frac{5(i-1)}{2^n} = \frac{5q^2-q}{2^{n+1}}$$

注意在这种情况下，Hash函数不是抗前像攻击的，因为其压缩函数不是抗自由前像攻击的。

也可以考虑其它情况，比如A可以选择消息或者一个消息和一个中间变量。但是，除非A可以同时选择输入和输出中间变量，否则B的复杂度不会有很大的变化。但是如果压缩函数存在这样攻击算法，说明这个压缩函数的强度太弱，在实际中是不可能用这样的压缩函数来构造Hash的。

由上面的定理和推论可以得出这样的结论：为了构造抗碰撞的Hash函数，压缩函数并不需要具有抗自由碰撞性。但是，不排除自由碰撞攻击可能用于前像攻击或者二次前像攻击的可能。

综上所述，在MD迭代构造方式下，为了使Hash函数具有抗碰撞性，压缩函数必须满足抗半自由碰撞性，但不必要满足抗自由碰撞性。这个结果可以放宽设计压缩函数的安全性要求，从而有利于压缩函数的设计。

第五章 Hash 函数的平衡度

Hash 函数的平衡度是 2004 年提出的一个新概念，它是衡量 hash 函数输出分布的一个指标。研究表明，生日攻击的复杂度与 hash 函数的平衡度有密切的关系：随着平衡度的降低，对 hash 函数可碰撞性的攻击复杂度将大大降低。因而，平衡度将成为 hash 函数安全性的一个新的指标。由于现有的 Hash 函数的构造方式以 MD 方式为主，为了使 Hash 函数有较高的平衡度，压缩函数应该满足怎样的条件？本章对这个问题进行了探讨，提出了局部平衡度的概念，并利用此概念解决了压缩函数局部平衡度与 Hash 函数平衡度的关系问题。

5. 1 平衡度对生日攻击的影响

通常的教科书都认为，根据生日攻击寻找 Hash 函数的碰撞对的复杂度大约是 $r^{1/2}$，其中 r 为 Hash 函数的值域的大小。特别的，对于输出长度为 m 位的 Hash 函数，寻找碰撞对的复杂度是 $2^{m/2}$。这个估计是选择 hash 函数输出长度的主要依据，即应该使 Hash 函数的输出长度足够长，使得 $2^{m/2}$ 在计算上是不可行的。但是，这种对生日攻击的分析并不准确，因为这种分析假定 Hash 函数是规则的，即在 Hash 函数的映射下，值域的每一个点都有相同数目的原像，但事实并非如此。在文献[25，26]中，Bellare， G. Laccetti, G. Schmid 分别对这一问题进行了研究。

5. 1. 1 Bellare的分析

Bellare提出了一个度量hash函数规则性的度量指标－平衡度，并分析了以此为标准hash函数的不规则性对生日攻击所构成的影响。平衡度的定义为：

定义5.1： 设hash函数 h: D→R 的值域R中有r > 2个点 $R_1, R_2, ...R_r$。对于i = 1, 2...r，设 $h^{-1}(R_i)$ 为在h下，R_i 的原象，即所有 $x \in D, h(x) = y_i$ 所构成的集合。令 $d_i = \left|h^{-1}(R_i)\right|$，作为$R_i$前像集合的大小，$d = |D|$ 作为值域的大小。记平衡

度为：

$$\mu(h)=\log_r\left[\frac{d^2}{d_1^2+\ldots+d_r^2}\right]$$

由定义可知，平衡度是一个从 0 到 1 的实数，当对于任意 i 和 j，有 $d_i=d_j$，即 hash 函数是完全规则时，平衡度取最大值 1。当 $d_i=1,d_j=0(j\neq i)$，即 hash 函数为常数函数时，平衡度取最小值 0。

我们更为感兴趣的是一个函数平衡度对生日攻击复杂度可能带来的影响。Bellare 在文章中得出结论为：$Q=r^{\mu(h)/2}$，其中 Q 为攻击成功所需要的试验次数。这个等式说明，生日攻击的复杂度可以简单而准确的表达为平衡度的函数。并且由这个等式可以得到如下结论：当 $\mu(h)=1$ 时，$Q=r^{1/2}$ 即通常认为的生日攻击的复杂度。当 $\mu(h)=0$ 时，Q=1，即当函数为常数函数时，只需要一次试验即可找到碰撞对。当 $\mu(h)=1/2,Q=r^{1/4}$，这远远低于 $r^{1/2}$。由此可以看出，hash 函数的平衡度对生日攻击复杂度的影响是不可忽视的。特别的，当函数平衡度很低时，攻击的复杂度将远远低于设计者所期望的值，即 hash 函数其实并不能达到设计者所设想的安全强度。

5．1．2　G．Lacceti，G.Schmid的分析

G. Laccetti, G. Schmid的文章中并没有明确的提出平衡度的概念，而是用函数输出的概率分布来刻画函数的规则性，并引入了majorazation的概念对hash函数的规则性进行比较。Majorization是n维实数数组的一种前序关系，即数组之间通过数组内元素的差别程度来进行排序。最大的数组为元素间没有差别，即含有完全相同元素的数组。输出概率分布及majorization的定义分别为：

定义5.2： 设h：X->Y是一个（M，N）hash函数，则离散分布

$$D_h:y_n\in\{y_1,y_2,\ldots y_N\}\rightarrow p_n=\left|h^{-1}(y_n)\right|/M$$

为h的输出概率分布。其中，M为h的定义域，N为h的值域。

定义5.3： 令$x=(x_1,..x_n), y=(y_1,...y_n)$是$R_n$的两个元素，设

$\left(x_{[1]},x_{[2]},...x_{[n]}\right)x_{[i]} \geq x_{[i+1]}$ i =1, 2, …n为X的一个降序排列，如果X，Y满足以下的关系

$$\sum_{i=1}^{k} x_{[i]} \leq \sum_{i=1}^{k} y_{[i]} \qquad (k = 1, \ldots, n-1)$$

$$\sum_{i=1}^{n} x_{[i]} = \sum_{i=1}^{n} y_{[i]}$$

我们就说$x < y$。其中X，y分别为降序排列。

文章中证明，相对于这种前序关系，抗碰撞性是其输出概率分布数组的递增函数。即在majorization关系中越大的hash函数具有越高的抗碰撞性。但同时他们也发现可逆性则是输出概率分布的递减函数，即在majorization中越大的hash函数其抗可逆性却越低。这样看起来，hash函数的规则性对其抗碰撞性和抗可逆性的影响似乎构成了矛盾，提高一个则会降低另一个。但是，后来作者证明对抗碰撞性的攻击复杂度通常情况下都会远低于对抗可逆性的攻击，所以在上面矛盾情况下，应该优先考虑抗碰撞性，即应该提高hash函数的规则性以增强其安全性。

由上面的结果知道，Hash 函数的平衡度应该作为 Hash 函数的一个设计标准，即为了达到 Hash 函数预想的安全性，应该保证其有较高的平衡度。现在 Hash 函数的主要设计方式是 MD 方式。MD 方式是一种基于压缩函数的迭代形式，由此构造的 Hash 函数其安全性可归结为压缩函数的安全性。MD 方式的具体形式为：

Function H（M）

将消息分成长度为 b 的块：$x_1,..x_t$，消息长度不是 m 的倍数则填充；并将消息长度填充到消息末尾。

$H_0 = IV$

for I= 1 to t do $H_i = f(X_i, H_{i-1})$

$h(x) = g(H_t)$

return h（x）

如果 MD 方式可以保持压缩函数的平衡度，那么设计者就可以把精力集中在设计具有高平衡度的压缩函数上。但是，Bellare 定义的压缩函数的平衡度并不能保证 Hash 函数的平衡度。在文献[21]中，作者证明只有当压缩函数的平衡度等于 1 时，Hash 函数才保持这个平衡度。但当压缩函数偏离 1 甚至只是微小的偏离就可能引起 Hash 函数的平衡度的急剧降低。这样，我们就没有办法将平衡度的指标用于压缩函数的设计。

但是，经过研究我们发现，如果重新定义压缩函数的平衡度，使压缩函数满足更高一点的要求，那么 MD 方式就可以保持这个平衡度，从而在压缩函数的设计中，就可以把这个新定义的平衡度作为一个指标。下面，我们引出局部平衡度的概念，并证明 MD 方式构造的 Hash 函数的平衡度将大于等于其压缩函数的局部平衡度。

5. 2 局部平衡度

5. 2. 1 局部平衡度的概念

我们这样定义函数局部平衡度：

定义 5.4： 设 $f:B\times C\to R$ 为 $\{0,1\}^b\times\{0,1\}^c\to\{0,1\}^r$ 的函数，其中 $b\geq r$。设 d 为定义域 B 的大小，$R_i\ (i=1,2...2^r)$ 为值域 R 中的一点。对于任意 $c\in C$，设 $d_{i,c}=|\{b:f(b,c)=R_i,b\in B\}|$，即在取定 c 情况下，$R_i$ 在定义域 B 中原象集合的大小。

对于任意 $c\in C$，定义函数 f 相对于 c 在定义域 B 上的局部平衡度为：

$$PB_c(f)=\log_r\left(\frac{d^2}{d_{1,c}^2+d_{2,c}^2+...+d_{r,c}^2}\right)$$

另外，定义函数 f 在定义域 B 上的局部平衡度为：

$$PB(f,B)=\min\{PB_c(f):c\in C\}$$

由定义可知，函数的局部平衡性反映的是定义域中某一部分在映射时的分布情况。局部平衡性的要求比整个函数平衡性的要求更强一些。下面是两者之间的关系。

整体平衡的函数并不能保证其局部平衡，例如：

取函数 $H:\{0,1\}^b\times\{0,1\}^c\to\{0,1\}^c$ 为 $H(m,n)=n$,其中m $\in B,n\in C$ 是集合内的任意元素，显然函数 H 的平衡度是 1，因为值域中每个点都正好有 2^b 个原象。但是，对于定义域 B，其局部平衡性为 0，因为取定任意一个 c，值域中 c 点有 2^b 个原象，而其它点的原象个数为零。

反之，如果函数在任意部分定义域上的局部平衡性为 1 的话，那么可以证明该函数的整体的平衡性也为 1。证明如下：

设函数 $f:B\times C\to R$为$\{0,1\}^b\times\{0,1\}^c\to\{0,1\}^r$的函数，如果函数 f 在定义域 B 上的局部平衡性为 1 的话，可以知道，对于任意 $c\in C$，值域中每个点在 B 上有相同的原象数目，假设为 n。则对于值域中任一点 R_i，它的原象的数目为 $\sum_{i=1}^{2^c} n$，从而可知值域上每一点的原象数目仍是相等的，其平衡度为 1。

5．2．3　压缩函数的局部平衡度与 Hash 函数平衡度

将局部平衡度的概念用于压缩函数。设压缩函数为 $f:B\times C\to C$，其中 B 为消息块，取值为 $\{0,1\}^b$，C 为中间变量，取值为 $\{0,1\}^c$。可以证明，如果压缩函数在定义域 B 上，即消息块空间上是局部平衡的，或者有较好的局部平衡性，则用此压缩函数以 MD 方式构造的 Hash 函数也会具有较好的平衡性。事实上，Hash 函数的平衡度应该大于等于其局部平衡度。

定理 5.1：设 $f:B\times C\to C$ 为压缩函数，$H:\{0,1\}^{nb}\times\{0,1\}^c\to\{0,1\}^c$ 为以 f 为压缩函数并且以 MD 方式构造的 Hash 函数，$PB(f,B)$ 为 f 在定义域 B 上的局部平衡度，$Balance(H)$ 为 H 的平衡度，则有 $Balance(H)\geqslant PB(f,B)$。

证明：用数学归纳法进行证明。

首先，来证明只有两轮的 MD 构造方式 H_2。

设初始向量为 c_0，$c_{\min}$ 为局部平衡度最小的中间变量。则有 $PB_{c_0}(f) \geq PB_{c\min}(f)$ 。$c_1 = f(c_0, b), b \in B$，显然，对于任意 $b \in B$，仍然有 $PB_{c_1}(f) \geq \mathrm{PB}_{c\min}(f)$。设 H_2' 为这样一个函数，对于任意的 c，取 f 的分布率等于 C= $c_{\min}$ 时的分布率，则 H_2' 的平衡度可以表示为：

$$Balance(H_2') = \log_r(\frac{(2^{2b})^2}{(2^b d_1)^2 + (2^b d_2)^2 + \ldots + (2^b d_r)^2})$$

$$= \log_r(\frac{(2^{2b})^2}{2^{2b}\sum_{i=1}^{r} d_i^2})$$

$$= \log_r(\frac{2^{2b}}{d_1^2 + d_2^2 + \ldots + d_r^2})$$

$$= PB_{c_{\min}}(f) = PB(f, B)$$

显然有，$Balance(H_2) \geq Balance(H'_2)$，从而有 $Balance(H_2) \geq PB(f, B))$。

对于 n＋1 轮的 hash 函数，假设 n 轮 Hash 满足上面的不等式，即有 $Balance(H_n) \geq PB(f, B)$，同样，设 H'_{n+1} 为这样一个函数，取 H'_n 的分布率等于 $C = c_{\min}$ 时 f 的分布率，对于任意的 c，取 f_{n+1} 的分布率等于 $C = c_{\min}$ 时的分布率，则 H'_n 的平衡度可以表示为：

$$Balance(H_{n+1}') = \log_r(\frac{(2^{(n+1)b})^2}{(2^{nb} d_1)^2 + (2^{nb} d_2)^2 + \ldots + (2^{nb} d_r)^2})$$

$$= \log_r(\frac{(2^{(n+1)b})^2}{2^{nb}\sum_{i=1}^{r} d_i^2})$$

$$= \log_r(\frac{2^{2b}}{d_1^2 + d_2^2 + ... + d_r^2})$$

$$= PB_{c_{\min}}(f)$$

$$= PB(f, B)$$

因为 $Balance(H_n) \geq PB(f,B) = PB_{c\min}(f)$，$PB_c(f) \geq \mathrm{PB}_{c\min}(f)$，所以有 $Balance(H_{n+1}) \geq Balance(H'_{n+1})$，从而 $Balance(H_{n+1}) \geq PB(f,B)$。

证毕

由该定理可知，在压缩函数的设计中，如果使压缩函数具有较高的相对于消息块的局部平衡度，那么由此压缩函数以 MD 方式构造的 Hash 函数也将有较高的平衡度。从而，局部平衡度可以作为压缩函数设计的一个重要指标。

本章提出了函数局部平衡度的概念，并用此概念解决了压缩函数和 Hash 函数平衡度之间的关系问题。现有的 Hash 函数的设计准则是随机性，而随机性并不能保证规则性。对于一般的随机函数，研究结果表明生日攻击的复杂度大约是完全规则函数的 2/3。所以，虽然 Hash 函数的随机性可以使其抵抗差分、线性攻击等密码分析，但是对于生日攻击不具备最好的性质。怎样来设计压缩函数，使得其既满足随机性，又有较高的局部平衡性，是下一步考虑的问题。

第六章 Hash 函数的攻击及预防

Hash 函数本身的历史并不长，但是其攻击方法在近些年，特别这近两年内发展迅速。在原有的差分攻击的基础上密码学者又提出了很多新的更有效的攻击方式，如针对 MD 构造方式的多碰撞攻击、多块碰撞攻击以及近似碰撞攻击等等。本章将介绍这些攻击方法，以及针对这些攻击方法对原有构造方式的改进办法。

本章内容分为两个部分，第一部分是对 MD 构造方式的攻击及预防方法，第二部分是针对压缩函数和实际应用的 Hash 函数的攻击和预防方法。

6.1 MD 构造方式的攻击

6.1.1 MD 构造方式的多碰撞攻击

MD 构造方式是实际应用中采用最多的一种构造方式，在前面章节做过详细的介绍。我们知道，如果压缩函数具有抗自由碰撞性，那么 MD 构造方式构造的 Hash 函数就具有抗碰撞性。一直以来，MD 方式都被被认为是安全的构造模式。但是，多碰撞攻击的出现打破了这一局面。下面首先介绍多碰撞攻击的概念，然后说明这一攻击对 MD 构造方式安全性的影响。

多碰撞攻击是 Juox 在 04 年密码学会议上提出的一种对迭代 hash 函数的一般性攻击[27]。多碰撞的定义如下：

定义 6.1 设函数 $g:D\to R$，g 的 r 维碰撞或者多碰撞是指定义域 D 的一个子集 $\{x_1,....x_r\}$ 且有 $g(x_1)=g(x_2)=...=g(x_r)=z$。其中，z 是指多碰撞子集的 Hash 结果值。

寻找 Hash 函数多碰撞子集的攻击就称为多碰撞攻击。在随机模型下，对于多碰撞攻击的复杂度有如下定理：

定理 6.1 设 $g:D\to R$ 是随机函数，那么寻找 r 维多碰撞子集的复杂度是 $\Omega(|R|^{\frac{r-1}{r}})$。生日攻击的复杂度是 $\Omega(|R|^{\frac{r-1}{r}})$。

由定理 6.1 可知，在随机模型下，生日攻击是最好的多碰撞攻击方法。但是

迭代构造方式并不满足这一条件。MD 方式是一种典型的迭代式结构，下面介绍 Joux 对 MD 方式的多碰撞攻击方法，这种攻击方法的复杂度远远低于生日攻击。

设 f 是压缩函数，其输出长度为 m 比特，H 是 f 以 MD 方式构造的 Hash 函数。设算法 Q 是压缩函数 f 的一个碰撞攻击，假设 f 是抗自由碰撞的，那么算法 Q 的复杂度应为 $2^{m/2}$。构造算法 A 来寻找 H 的多碰撞。

算法 A:

设 h_0 是 H 的初始向量

For i = 1 to t do:

调用算法 Q 并得到 B_i, B'_i，使得 $f(h_{i-1}, B_i) = f(h_{i-1}, B'_i)$

$h_i = f(h_{i-1}, B_i)$

End for

填充并输出 2^t 个有形式 $b_1, ..., b_t, b_{len}$ 的消息，其中 b_i 是 B_i 或者 B'_i 中的一个;

显然，算法 A 输出的 2^t 个消息都具有同一 Hash 值。算法 A 的复杂度为 $t * 2^{m/2}$，远低于生日攻击的复杂度为 $2^{\frac{n(2^r-1)}{2^r}}$。可见，MD 方式并不能保持压缩函数的抗多碰撞性。该攻击算法对于所有的迭代结构都是有效的，因此，任何迭代结构都不能抵抗多碰撞攻击。

下面来看多碰撞攻击可能的应用场景。显然，对于普通 Hash 函数，这种攻击并没有实际的意义。因为该攻击是基于碰撞对攻击的，而对于普通 Hash 函数来说，根据其抗碰撞性的要求，搜索压缩函数的计算复杂度已经是不可行的了。但是，对于理论分析，多碰撞攻击具有很重要的价值。下面来看两个理论分析的例子。

用于分析两个 MD 结构 Hash 函数并联的构造方式。设 $G, H : D \to \{0,1\}^n$，则 $G(.) \| H(.)$ 称为 G 和 H 的并联。在多碰撞攻击出现之前，这种构造方式的安全性一直没有恰当的分析方法。表面上看，并联方式似乎能够达到完全安全性。多碰

撞攻击成功的解决了这一问题。我们来看怎样用多碰撞攻击来对并联方式构造的函数进行攻击。

首先构造一个函数 G 的大小为 2^n 的多碰撞子集。由多碰撞攻击的复杂度可知，这个过程的复杂度为 $n2^{n/2}$。而由生日攻击原理可知，这个子集中以很大概率存在一个 H 的碰撞对。这样，该攻击总的复杂度是 $n2^{n/2}$，而不是 2^n。因此，这种构造方式是达不到完全抗碰撞安全的。

用于寻找二次碰撞[28,29]。这里将原多碰撞攻击做稍微改动。在每次寻找碰撞对的时候，不再是找两个单块的碰撞，而是在第 i 次寻找长度为 1 和 2^i 块的碰撞对。这样在当 i=k 时，就可以构造出长度在（k，$k+2^k-1$）之间任意长度的碰撞消息集合，我们称这个消息集合为可扩展消息集合。设原消息为 M，设 M 的长度大于等于 $k+2^k+1$。我们利用可扩展消息集合来构造 M 的二次碰撞。

1. 构造（k，$k+2^k-1$）可扩展消息集合，记其最终计算结果为h。
2. 计算消息M的Hash值，并记录消息M计算过程中的每一个中间状态（$h_1,...,h_{k+2^k+1}$）。
3. 寻找一个连接块 m，使得 $f(h,m)$ 等于（$h_1,...,h_{k+2^k+1}$）中的某个 hj。那么有 $M'=M^*\|m_{link}\|m[j+1]...m[2^k+k],H(M)=H(M')$。其中，M'的长度等于M的长度，M*是可扩展碰撞集合中的某个符合长度的消息。

整个攻击过程的复杂度为 $k*2^{n/2+1}+2^{n-k+1}$，显然这个复杂度低于强力攻击的复杂度2^n。但这个攻击存在的问题是，所求的消息的长度必须足够长，要大于等于 $k+2^k+1$，并且第二步中要保存大量的中间变量，实际上是以空间换时间的攻击方法。这种攻击只有理论分析的意义。用这种方法对SHA-1的二次前像攻击的复杂度可以达到2^{105}。

6．1．2　　抵抗多碰撞攻击的改进方式

由上面的分析可以知道，多碰撞攻击实际上利用的是迭代结构的特性。任何具有迭代性质的构造方式都会存在这个问题。为了避免这种攻击，可以从两个方

面考虑。一是考虑改变MD结构的迭代性质，二是扩展中间变量的长度，使得中间变量的长度大于最终结果长度。

6．1．2．1　改造MD结构方式

我们提出了一种对 MD 结构简单改动的方式来抵抗多碰撞攻击。

设消息 $M = M_1,...,M_k$，定义

$$H^*(M) = H(M \| M_{k+1}),\ \text{其中} M_{k+1} = M_1 \oplus M_2,...,\oplus M_3$$

可以在随机模型下对这个结构抗多碰撞的安全性进行证明。设 H 的压缩函数是随机函数，或者只假设压缩函数是抗自由碰撞函数。另外假设算法 Q 输出的碰撞消息对是随机的。这个假设是合理的，因为算法 Q 只是生日攻击。可以证明算法 A 的多碰撞攻击在这种情况下不能成功。

对于算法 A 得到的碰撞对集合，每个消息都要附加对 $M_{k+1} = M_1 \oplus M_2,...,\oplus M_3$ 的计算。由于算法 Q 的输出都是随机的，因此有每一个 $M_{k+1} = M_1 \oplus M_2,...,\oplus M_3$ 也都是随机的，设 h_k 为原碰撞消息块的计算结果，那么 $f(h_k, M_{k+1}^i) = f(h_k, M_{k+1}^j)$ 的概率为 $2^{n/2}$。也就是说原碰撞对集合里任意两个消息仍是碰撞对的概率为 $2^{n/2}$。

这种结构的改动并不会改变原结构的抗碰撞性和抗前像性，这里不做证明。

6．1．2．2　增长中间变量方式

Wide-Pipe Hash 函数是一种典型的增长中间变量方式[30]。Wide-Pipe 的出发点非常简单。从一个输出长度较长的压缩函数构造安全的输出长度较短的 Hash 函数显然要容易的多。设压缩函数 $f': \{0,1\}^w \times \{0,1\}^m \to \{0,1\}^w$，$f'': \{0,1\}^w \to \{0,1\}^n$，$H_0 \in \{0,1\}^w$ 作为初始变量。其中，$w > n$。按如下方式计算 Wide-Pipe 函数 H:

For i = 1 to L : $H_i = f'(H_{i-1}, M_i)$;

$H(M) = f''(H_L)$;

在对 Wide-Pipe 做多碰撞攻击时，如果假设 f'' 是完全安全的，那么攻击复

杂度至少为 $t*2^{w/2}$，如果取 w=2n，那么攻击的复杂度将相当于生日攻击的复杂度。也就是说，当 w=2n 时的 Wide-Pipe 函数是可以抵抗多碰撞攻击的。同样可以证明这个函数满足其它的安全性要求。但是，这种构造方式最大的缺点是效率低，并且构造输出长度更大的压缩函数并不是一件容易的事情。

另外一个例子是 Double-Pipe 函数。此函数同样增加了中间变量的长度，与 Wide-Pipe 不同的是采用了两个压缩函数并联的办法来增加长度。其安全性的证明与前面类似。同样也存在效率低的问题。

6.2 基于具体算法的攻击

具体算法的攻击是指针对具体的压缩函数的结构或者代数特性进行攻击。目前，最有效的攻击方法是差分攻击，特别是对于直接构造的 Hash 函数，如 MD4、MD5、SHA0 和 SHA1 等。最早的 Hash 函数差分攻击是 Dobbertin 对 MD4 的攻击[31]。随后，随着差分攻击方法本身的成熟和其它攻击方法的补充，直接构造的 Hash 函数的安全性受到了巨大的威胁[32-40]，MD 家族全军覆没，SHA 系列也面临着严重的威胁。

差分攻击最初被用来分析分组密码，我们来看差分攻击的概念。

设函数为 f 的输入输出分别为 X 和 Y。其中，$X=\{x_1,....x_n\}$和$y=\{y_1,....y_n\}$。设对应于 f 的两个不同的输入 X 和 X' 的输出分别为 Y 和 Y'。令 $\Delta X=\{\Delta x_1,....\Delta x_n\},\Delta y=\{\Delta y_1,....\Delta y_n\}$，其中 $\Delta x_i=x_i\oplus x'_i,\Delta y_i=y_i\oplus y'_i$。如果函数 f 是随机函数，那么给定 ΔX 得到指定某个 ΔY 的概率为 $2^{1/n}$，其中 n 是 X 的比特长度。差分攻击就是寻找这样一对（ΔX，ΔY），给定 ΔX 函数将以很高的概率得到 ΔY。我们称（ΔX，ΔY）为差分。为了从 ΔX 以较高的概率到 ΔY，需要对函数 f 的每一轮的输入输出差分进行分析。称所有这些轮的差分为差分特征或差分路径 ΔP。

对于 Hash 函数，由于我们的目的是寻找碰撞对，因此希望 $\Delta Y=0$。显然，对 Hash 函数的差分攻击可以归结为寻找高概率的 ΔX 和 ΔP。下面来具体分析几个差分攻击的例子。

6．2．1　王小云对 MD4 的差分攻击

此攻击采用了整数加差分。采用整数加差分的目的是能够使寻找差分路径时有更多的灵活性。攻击分为三个步骤：

第一步：　选取高概率的差分和差分路径

消息差分为：

$$\Delta H_0 = 0 \overset{M,M'}{\rightarrow} \Delta H = 0$$

$$\Delta M = M - M' = (\Delta m_0, ...\Delta m_{15})$$

$$\Delta m_1 = 2^{31}, \Delta m_2 = 2^{31} - 2^{28}, \Delta m_{12} = -2^{16}, \Delta m_i = 0, 0 \le i \le 15, i \ne 1,2,12.$$

具体的差分路径参见文献[34]；

寻找差分路径时要注意和充分利用一下几点：

a) 整数加差分的特性。对于某个整数加差值，可以扩展为多种异或比特差值。比如 $\Delta X = +2^6$ 与 $\Delta X = +2^7 - 2^6$ 就有相同的整数加差值。

b) 非线性函数的特性。比如，对于 XOR 函数，如果两个元素同时取反那么函数值仍然不变，取反一个元素则函数值取反。其它函数同样也有其特性。

c) 循环移位对差值的影响。

第二步：　获取为了得到上述的差分，函数的每步的中间变量需要满足的充分条件。

这一步主要利用的是非线性函数的特性以及轮函数的运算性质。1 中的差分路径要满足的条件共有 122 个。

第三步：　采用消息修正技术，使得第二步中尽量多的条件得到满足。

消息修正技术可以有单块修正和多块修正方式。单块修正是比较简单的修正方式，只通过对单个消息字的修改来使得中间变量满足条件。通常单块修改用于第一轮，这是因为第一轮的消息修改有很大的自由度。多块修改比较复杂，通常是通过同时对连续的几个消息字的修改来达到目的。多块修改通常用于第一轮之后的修改。采用多块消息的原因是在第二轮对消息的修改会影响到第一轮的计算结果，因此要通过多个消息字来消除这种影响。

经过修正之后，不能满足的条件只有 6 个。也就是说这个攻击可以以 2^6 的

复杂度找到碰撞对。消息修正是一种非常有效的技术，它能够非常有效的降低差分攻击的复杂度。在王小云之后，很多人对消息修正技术做了更深入的研究，提出了很多更好的修正技术，这里不再赘述。

6．2．2　王小云对 MD5 的差分攻击

MD5 的攻击采用了与 MD4 相同的攻击技术，但攻击过程有所不同。MD5 压缩函数的轮数更多，并且比 MD4 更复杂，因此并不容易找到高概率的差分和差分路径。为了解决这个问题，王小云采用了近似碰撞和多块攻击方法。

近似碰撞是指消息 M 和 M'的 Hash 值大部分相同，只存在很少的差分。近似碰撞和多块攻击是由 Biham 等人提出的[30,31]，其结构如下所示：

$$\Delta H_0 \overset{M_0,M_0'}{\rightarrow} \Delta H_1 \overset{M_1,M_1'}{\rightarrow} \Delta H_2 \overset{M_2,M_2'}{\rightarrow} \ldots\ldots \Delta H_{k-1} \overset{M_{k-1},M_{k-1}'}{\rightarrow} \Delta H$$

其中，$\Delta H = 0$，ΔH_i是第 i-1 步的输出差分并且满足近似碰撞的定义，同时ΔH_i又是第 i 步的输入差分。这样，通过多块消息相连接最终构造出碰撞对。

MD5 的攻击采用了两块消息，其消息差分为：

$$\Delta H_0 = 0 \overset{M_0,M_0'}{\rightarrow} \Delta H_1 \overset{M1,M_1'}{\rightarrow} \Delta H = 0$$

其中，

$$\Delta M_0 = M'_0 - M_0 = (0,0,0,0,2^{31},0,0,0,0,0,0,2^{15},0,0,2^{31},0)$$
$$\Delta M_1 = M'_1 - M_1 = (0,0,0,0,2^{31},0,0,0,0,0,0,-2^{15},0,0,2^{31},0)$$
$$\Delta H_1 = (2^{31},2^{31}+2^{25},2^{31}+2^{25},2^{31}+2^{25})$$

搜索碰撞算法如下：

1) 随机生成 512 比特的第一块消息M_0；
2) 使用消息修正算法修正M_0，使得中间变量的条件尽量得到满足；
3) 如果所有条件都得到了满足，那么以 MD5 的初始向量为初始值，计算这个修正过的消息的 Hash 值。否则，随机的选取m_{14}, m_{15}并回到第二步；
4) 随机生成 512 比特的第二块消息M_1；
5) 采用与第一块消息相似的步骤，不同的是初始变量使用第一块的输出值；
6) 计算$M'_0 = M_0 + \Delta M, M'_1 = M_1 + \Delta M$；

7) $(M_0, M_1), (M'_0, M'_1)$ 即是碰撞消息对。

搜索第一块消息的复杂度为 2^{39}，第二块消息的复杂度为 2^{32}。这种攻击算法目前并不能获得有意义的碰撞对，但是可以设计某些应用场景，使得本来无意义的碰撞对发挥作用。比如，有人将其用在数字证书中的随机串部分从而构造出具有相同 Hash 值的证书[41]。因此，这种攻击对实际应用来说也是很危险的。在王小云的攻击公布之后，又有很多人对其做了改进，详见参考文献[42 - 46]。

6．2．3　王小云对 SHA0 和 SHA1 的差分攻击

SHA0 和 SHA1 差分攻击仍是采用了消息修正、近似碰撞和多块碰撞等技术。与 MD4 和 MD5 攻击不同的是差分和差分路径的寻找方法。SHA0，SHA1 的消息编排与 MD4 和 MD5 不同，它们采用了函数的方式。

SHA0 的消息编排公式为：$m_i = m_{i-3} \oplus m_{i-8} \oplus m_{i-14} \oplus m_{i-16}, i = 16,...,79$；

SHA1 的消息编排公式为：$m_i = (m_{i-3} \oplus m_{i-8} \oplus m_{i-14} \oplus m_{i-16}) << 1, i = 16,...,79$；

由消息编排公式可以知道，编排后的消息字之间是存在着线性关系的。同样如果引入消息差分，那么后面的差分是可直接计算的。另外，SHA 的轮函数有如下性质，当在某一轮引入一个差分之后，可以通过在随后的五轮中引入其它差分将该差分的影响消除。因此，对 SHA 攻击的主要的问题是怎样找到有较少消息差分的差分路线。

这里不再详细介绍攻击的过程。SHA0 的攻击复杂度约为 2^{39}，SHA1 为 2^{69}。SHA1 的攻击复杂度虽然要低于生日攻击的复杂度，但是 2^{69} 对于实际攻击来说还是不可行的。因此，这个结果目前还只是有理论上的意义。但是，随着攻击技术的不断进步，这个攻击可能会得到进一步的改进。

6．2．4　抵抗差分攻击的模式

由攻击过程可知，王小云的差分攻击主要基于两点：一是找到高概率的差分和差分路径，二是通过消息进行修正来满足条件。如果能采用某种方式使得这两点之一不能成功，那么就可以抵抗这种差分攻击。另外，由于较复杂的算法需要多块差分消息连接，因此如果能抵抗多块攻击那么也可以削弱差分攻击的效力。下面，介绍几种按这种思路对 Hash 函数的改造方式。

6．2．4．1　消息预处理方式

消息预处理方式是 Michael Szydl 等人提出的一种改进方案[47]。该方案基于第一种思想，即使得差分路径的寻找和消息修正更加困难。消息预处理方式是指在原算法不变的基础上，通过对消息进行预处理来增强其抗差分攻击能力。其形式可以表示为：

$$H^*(M) = H(\phi(M))$$

其中，$\phi(M)$ 是一个简单函数。

为了使算法能够做到在线处理，我们希望 $\phi(M)$ 的处理能控制在某个局部范围内。下面介绍两种满足局部性要求的 $\phi(M)$。

a) 消息白化

$m = (m_0, m_1, ..., m_{16-t})$ 扩展成 $\phi(m) = (m_0, m_1, ..., m_{16-t}, 0, ..., 0)$

b) 消息交叉

$m = (m_0, m_1, ..., m_k)$ 扩展成 $\phi(m) = (m_0, m_0, m_1, m_1, ..., m_k, m_k)$

这两种方法都通过增加消息的结构化，降低了消息字选取的灵活性。可以使用编码理论对这两种处理方式对差分路径搜索难度的影响做出分析，这里不再详述。对于消息修正，显然消息白化方式中被取值为 0 的消息字是不能进行修改的。对于消息交叉方式，由于相邻两个消息要相同，因此也很大的提高了修改的难度。

基于消息处理的思想，我们也提出了一个新的处理方式。该方式比以上两种方式更简单易行。具体做法是将上一个消息块的结果值作为下一个消息块的前几个字，比如，MD5 可以是前四个字，SHA-1 则是前五个字。第一块消息的前几个字可以采用白化的方式。该方法除了可以提高差分路径搜索和消息修正的难度之外，还可以增加多块碰撞攻击的难度。

消息预处理方式的优势是不需要对原算法做改动，非常方便实际应用实现的升级。但是，这种方案同时也有效率较低的问题。比如，消息交叉方式的处理效率只有原来的一半，消息白化方式的效率取决于取 0 值的消息字的个数。

6．2．4．2　加标记方式

加标记方式是 Neil Kauer 等人提出的方式[48]。该方式也是通过增加消息整体

的结构化来提高对差分攻击的抵抗性。其基本思想是在对消息做 hash 之前，先做一遍 MAC，然后将 MAC 的结果 tag 分别加在原消息的头部和尾部，最终对处理过的消息做 hash。其改造过程如下：

长度较短的消息的处理过程：

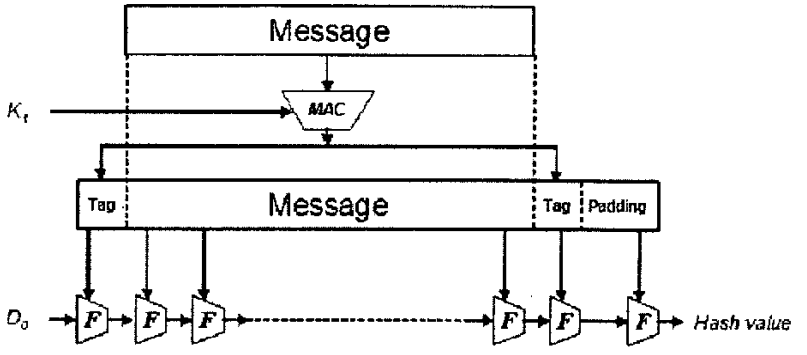

图 6.1

其中，D_0 部分即使传统的 MD 模式。

对于长度较长的消息，考虑到缓冲区容量的问题，将消息分部分来做，过程与上面类似：

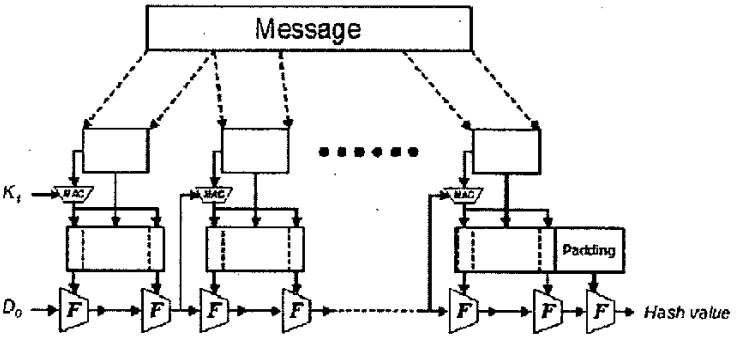

图 6.2

其中的 MAC 可以考虑采用泛 Hash 函数。显然，这种结构使得消息块差分不仅要满足本块的碰撞，还要使得整体的 MAC 值碰撞或者有多块更复杂的差分。

这两种结构都不具有在线计算功能，并且消息要被扫描两遍。

可以考虑去掉添加的前一个 MAC 来改进这个结构。去掉前一个 MAC 似乎不会对整体的安全性构成很大的影响。如果可以构造某个 F 的碰撞对并且这个碰撞对的 MAC 值也恰好相等，那么加两个 MAC 与加一个的效果是一样的。这样

改造之后，该结构就可以实现在线计算，并且 Hash 和 MAC 可以实现并行计算，这样就可以提高效率。这种方式的安全性有待于继续研究。

6．2．4．3　3C Hash

3C Hash 主要是为了抵抗多块碰撞攻击[49]。其结构如下：

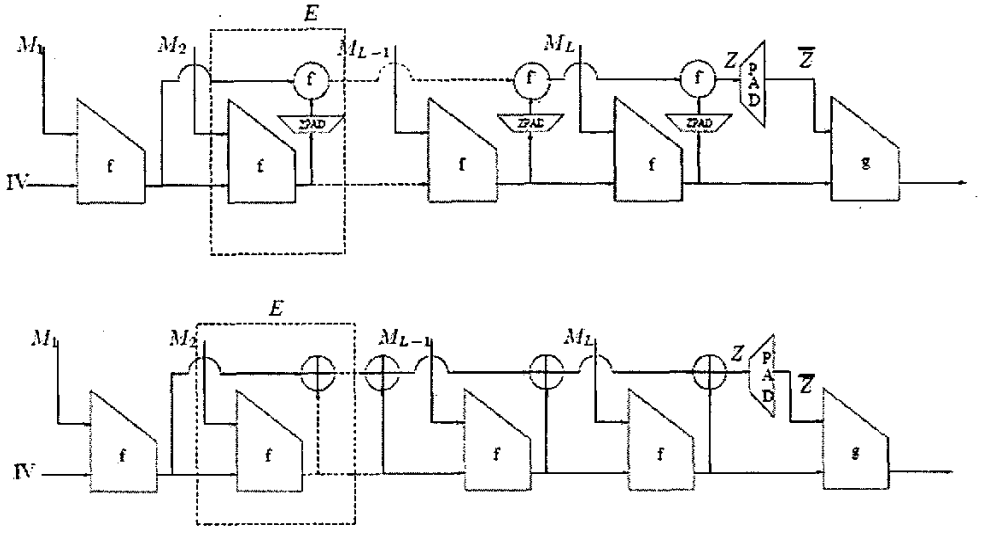

图　6.3

其中，f 指压缩函数，g 是某个单向函数。

3C Hash 有如下几个特点：

- 不管处理多长的消息至少要调用 3 次压缩函数。
- 对原有的 hash 函数进行了高效的改进。
- 使用多块碰撞攻击的方法必须同时找到在链变量和累加链中的碰撞。这样就增加了攻击的难度。
- 3C-hash 能抵抗扩展攻击。

3C Hash 是目前改造方式中比较好的结构，其安全性在随机模型下可以证明，并且也具有高效的特点。但是，3C Hash 的证明过程对压缩函数的安全性要求比较高，并且不能抵抗多碰撞攻击。因此，这一结构仍有待进一步改进。

这些预防手段都是从便于实际应用的角度考虑的。如果不考虑时间应用的方便，也可以采用其它的方法来增加原函数的安全性。比如增加轮数，替换非线性函数等等。但是，更长远的解决办法还是设计能够抵抗这些攻击的新算法。

结束语

本文主要讨论了 Hash 函数的安全性分析。前三章对 Hash 函数的基本概念和安全性质做了介绍，并比较和分析了 Hash 函数的各种设计方法。第四章的主要内容是在 MD 加强模式下压缩函数与 Hash 函数安全性质关系的一些分析研究。通常为了保证 Hash 函数的抗碰撞性需要压缩函数具有抗自由碰撞性，但是在本文中我们在黑盒子模型下证明在压缩函数不能抵抗某些自由碰撞攻击的情况下，由此构造的 Hash 函数仍然可以具有抗碰撞性。因此，在 MD 加强模式下，可以不要求压缩函数具有抗自由碰撞性。这个条件的放宽可以更有利于压缩函数的设计。第五章是 Hash 函数平衡度的分析，其中重点是压缩函数平衡度在 MD 加强模式下的保持问题。在原定义模型下，MD 方式不能保持压缩函数的平衡性，从而给压缩函数的设计要求带来了困难。我们提出了局部平衡度的概念解决了这一问题。可以证明，当压缩函数满足局部平衡性要求时，MD 方式构造的 Hash 函数的平衡度要大于等于压缩函数的平衡度。第六章介绍了目前对 Hash 函数的各种攻击方法和攻击成果，并着重讨论了预防这些攻击方法的措施。本文中讨论的措施主要从实际应用角度出发，通过对算法简单升级的办法来增加其抵抗攻击的能力。结合已有的升级方案，在文章中我们也提出了新的解决方案。

对于 Hash 函数未来的研究方向，我们认为主要集中在两个方向。

一是对 Hash 函数本身性质的研究，Hash 函数作为密码学的基础模块，一直没有完整成熟的理论，这有待于继续完善。

二是新 Hash 函数的设计。由于 MD5，SHA-1 等相继被攻破，新的安全的 Hash 函数的设计成为急需解决的问题。事实证明，原有的直接构造 Hash 函数的方式并不能抵抗差分攻击，MD5,SHA-1 的失败就是例子。新 Hash 函数的设计可以从两种途径考虑。一是基于分组密码构造 Hash。AES 等更新更安全的分组密码的出现为这一方向奠定了基础，这个方向的重点应该是双倍长度并且高效的构造方式；二是采用新的方式直接设计新算法。在新算法的设计中应该尽量引入分组密码的设计理念和准则。比如 S 盒，线性变换，变化的循环移位，快速有效的混和等等。另外，对于分组密码的安全分析也应该用于 Hash 函数的分析，比如对线性攻击，代数攻击的抵抗能力等。又因为 Hash 函数有其自身的特点，还要注意某些对于分组密码并不构成足够威胁的攻击方法，比如相关密钥攻击等等。

参考文献：

[1] B.Preneel, Analysis and design of cryptographic hash functions, Doctoral dissertation, Katholieke University Leuven, 1993

[2] B.Preneel, The state of cryptographic hash functions. Lecture Notes in Computer Science,1561 (1999), 158–182 (Lectures on Data Security: Modern Cryptology in Theory and Practice)

[3] L.R.Knudsen, X.Lai, B.Preneel, Attacks on fast double block length hash functions,Journal of Cryptology, Vol.11,No.1,1998,pp.59-72

[4] D.R.Stinson. Some observations on the theory of cryptographic hash functions. Designs, Codes and Cryptography,. 35 (2005), 119-152.

[5] R.L.Rivest, The MD5 message-digest algorithm, RFC1321,Internet Activity Board, Internet Privacy Task Force, 1992

[6] NIST/NSA, "FIPS 180-2: Secure Hash Standard (SHS)", August 2002 (change notice: February 2004)

[7] Damgard. A design principle for hash functions. In Advances in Cryptology–Crypto'89, volume 435 of Lecture Notes in Computer Science, pages 416–427.Springer–Verlag, 1989

[8] Bok-Min Goi, M.U. Siddiqi, and Hean-Teik Chuah. Incremental Hash Function Based on Pair Chaining & Modular Arithmetic Combining.INDOCRYPT 2001, LNCS 2247, pp. 50–61, 2001.Springer-Verlag Berlin Heidelberg 2001

[9] B.Preneel, R.Govaerts, J.Vandewalle, Hash functions based on block ciphers:a synthetic approach, Advances in Cryptology, Proceedings Crypto'93, LNCS773, Springer-Verlag, 1994,pp.368-378.

[10]Black, Rogaway,and Shrimpton,T..Black-box analysis of the block-cipher-based hash function constructions from PGV. In Advances in Cryptology, CRYPTO '02 (2002), vol. 2442 of Lecture Notes in Computer Science, Springer-Verlag.

[11]X.Lai, J.Lmassey, Hash functions based on block ciphers, Advances in Cryptology,ProceedingsEurocrypt'92,LNCS658,Springer-Verlag,1993,pp.55-70

[12]L. Knudsen, X. Lai, and B. Preneel, "Attacks on fast double block length hash functions," Journal of Cryptology, vol. 11, no. 1, pp.59–72, 1998.

[13]M.Hattori, S.Hirose and S.yoshida, Analysis of double block length hash functions. Lecrure Note in Computer Science 2898(2003), 290-302(Cryptography and Coding 2003).

[14]L. Knudsen and B. Preneel. Hash functions based on block ciphers and quaternary codes. In ASIACRYPT'96 Proceedings, pages 77 - 90, 1996. Lecture Notes in Computer Science 1163.

[15] L. Knudsen and B. Preneel. Construction of secure and fast hash functions usingnonbinary error-correcting codes. IEEE Transactions on Information Theory,48(9):2524 - 2539, 2002.

[16]Mridul Nandi1, Wonil Lee, Kouichi Sakurai, and Sangjin Lee. Security Analysis of a 2/3-Rate Double Length Compression Function in the Black-Box Model。 FSE 2005, LNCS 3557, pp. 243–254, 2005. International Association for Cryptologic Research 2005

[17]M. Hattori, S. Hirose and S. Yoshida. Analysis of Double Block Lengh Hash

Functions. Cryptographi and Coding 2003, LNCS 2898.

[18]B .O. B rachtl, D. Coppersmith, M.M. Hyden, S.M. Matyas, C.H. Meyer, J. Oseas, S. Pilpel, M. Schilling, "Data Authentication Using Modification Detection Codes. Based on a Public One Way Encryption Function," U.S. Patent Number 4,908,861, March 13, 1990. 170

[19]C.H. Meyer, M. Schilling, Secure program load with Manipulation Detection Code, Proc. Securicom 1988, pp. 111 - 130.

[20]J. Black, M. Cochran, and T. Shrimpton, On the Impossibility of Highly-Efficient Blockcipher-Based Hash Functions, EUROCRYPT 2005, pp526-541

[21]R. Anderson and E. Biham, Tiger: A Fast New Hash Function, *Fast Software Encryption - FSE'96*, LNCS 1039, Springer-Verlag (1996), pp. 89--97.

[22]P. S. L. M. Barreto and V. Rijmen, The Whirlpool Hashing Function, First open NESSIE Workshop, Leuven, Belgium, 13--14 November 2000.

[23]D. Boneh, M. Franklin, Efficient generation of shared RSA keys, Advances in Cryptology, Proceedings Crypto' 97, LNCS 1294, Springer-Verlag, 1997, pp. 425 - 439. 173

[24]Y. Frankel, P. D. MacKenzie, M. Yung, Robust efficient distributed RSA-key generation, Proc. 30th ACM Symposium on the Theory of Computing, 1998. 173

[25]M.Bellare, T.Kohno, Hash function balance and its inpact on birthday attacks,Advances in Cryptology,Proceedings Eurocrypt'04, LNCS3027, Springer-Verlag, 2004,pp.401-418

[26]G. Laccetti and G. Schmid. On a Probabilistic Approach to the Security Analysis of Cryptographic Hash Functions. http://eprint.iacr.org/2004/324

[27]A.Joux, Multicillisions in iterated hash fuctions.applications ton cascaded constructions, Advances in Cryptology, Proceedings Crypto'04, LNCS3152, Springer-Verlag, 2004, pp.306-316

[28]7J. Kelsey, B. Schneier, Second preimages on n-bit hash functions for much less than2n work, eprint archive 2004/304, http://eprint.iacr.org/2004/304;

[29]John Kelsey, Tadayoshi Kohno, Herding Hash Functions and the Nostradamus Attack. eprint.iacr.org/2005/281.pdf

[30]Stefan Lucks, A Failure-Friendly Design Principle for Hash Functions. ASIACRYPT 2005, LNCS 3788, pp. 474--494, 2005. International Association for Cryptologic Research 2005

[31]H. Dobbertin, Cryptanalysis of MD4, Journal of Cryptology 11:4 (1998), pp. 253—271

[32]John Kelsey. Some Current Thinking on Hash Functions Within NIST

[33]H. Dobbertin, Cryptanalysis of MD5 Rump Session of Eurocrypt 96, May. http//www.iacr.org/conferences/ec96/rump/index.html.

[34]Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the Hash Functions.MD4 and RIPEMD.EUROCRYPT 2005, LNCS 3494, pp. 1--18, 2005

[35]Xiaoyun Wang and Hongbo Yu, How to Break MD5 and Other Hash Functions, published on the web. http://www.infosec.sdu.edu.cn/paper/md5-attack.pdf

[36]F.Chabaud, A.Joux, Differential collisions in SHA-0, Advances in Cryptology, Proceedings of Crypto'98, LNCS1462,Springer-Verlag, 1999, pp. 56-71

[37]E.Biham, Near-collisions of SHA-0,Advances in Cryptology,Proceedings of Crypto'04, LNCS3152, Springer-Verlag, 2004,pp.290-305

[38]E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet, and W. Jalby. Collisions of SHA-0 and Reduced SHA-1. In R. Cramer, editor, Advances in Cryptology – EURO- CRYPT 2005, volume 3494 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, 2005.

[39]Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1 V. Shoup (Ed.): Crypto 2005, LNCS 3621, pp. 17–36, 2005.

[40]Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient Collision Search Attacks on SHA-0. Crypto 2005, LNCS 3621, pp. 1 - 16, 2005.International Association for Cryptologic Research 2005

[41]A. Lenstra, X. Wang, and B. de Weger, Colliding X.509 certificates. Cryptology ePrint Archive, Report 2005/067, 2005. Available online at http://eprint.iacr.org/

[42]V. Klima. Finding MD5 collisions on a notebook PC using multi-message modifications. Cryptology ePrint Archive, Report 2005/102, 2005.

[43] Vlastimil Klima. Finding MD5 Collisions – a Toy For a Notebook, http://eprint.iacr.org/2005/075.pdf, March 5, 2005.

[44]Zhang-yi Wang, Huan-guo Zhang, Zhong-ping Qin, Qing-shu Meng, A Fast Attack on the MD5 Hash Function, paper submitted to The 2nd Information Security Practice and Experience Conference.

[45]Jun Yajima, Takeshi Shimoyama, Wang's sufficient conditions of MD5 are not sufficient, http://eprint.iacr.org/2005/263.pdf, 2005.

[46]Yu Sasaki, Yusuke Naito, Noboru Kunihiro, Kazuo Ohta, Improved Collision Attack on MD5, http://eprint.iacr.org/2005/400.pdf, 2005.

[47]Michael Szydlo1 and Yiqun Lisa Yin2. Collision-Resistant usage of MD5 and SHA-1 via Message Preprocessing. http://eprint.iacr.org

[48]Neil Kauer, Tony Suarez, Yuliang Zheng y Enhancing the MD-Strengthening and Designing Scalable Families of One-Way Hash Algorithms. http://eprint.iacr.org

[49]3C -Praveen Gauravaram1, William Millan1, Juanma Gonzalez Neito1, Edward Dawson .A Provably Secure Pseudorandom Function and Message Authentication Code.A New mode of operation for Cryptographic Hash Function. http://eprint.iacr.org/2005/390

硕士期间完成的论文：

（1） 奚青，吴文玲，压缩函数局部平衡度与 Hash 函数平衡度的关系研究，《中国科学院研究生院学报》2006 年第三期。

（2） Qing Xi，Wenling Wu， Study on the Relation between the Security of Compression Function and Hash Function。

致谢

三年的硕士生活转瞬既逝。在硕士学位论文即将完成之际，我想向曾经给我帮助和支持的人们表示衷心的感谢。

感谢我的导师吴文玲研究员，她在学习和科研方面给了我大量的指导，并为我们提供了良好的科研环境，让我学到了知识，掌握了科研的方法，也获得了实践锻炼的机会。她严谨的治学态度、对我的严格要求以及为人处世的坦荡将使我终身受益。除此之外，她对我生活的关心和照顾也使得我得以顺利完成研究生的学业。在此祝愿她身体健康，全家幸福！

感谢武传坤、林东岱两位研究员对我的帮助和教导。

感谢两年来给了我很多帮助的王鹏、张文涛、陈华、林品、王大印等各位师兄师姐们，以及实验室里与我共同学习生活的各位同学，是你们的关心和帮助让我度过了两年的快乐时光。

感谢我的父母，一直以来他们都是那样默默的支持我。十九年的学生生活，有挫折有快乐，是他们一直与我分担忧愁，分享快乐。

感谢我的朋友们，是你们的友爱让我的生活充满阳光，你们的鼓励让我一直勇敢进取。

感谢老师们的辛勤培育，感谢朋友们的一路相伴，感谢父母家人的支持，感谢一切幼稚的，成熟的，忧伤的，开心的，所有的一切。

愿我们都有一个更美好的明天！