华中师范大学

硕士学位论文

概率统计在计算机密码学中的应用

姓名：刘平

申请学位级别：硕士

专业：概率论与数理统计

指导教师：李波

20080501

摘 要

密码学中 Hash 函数能够用于数据完整性和消息认证以及数字签名，新近的对于 MD5，SHA-1 等的碰撞攻击[5]表明在一个 Hash 函数的内部迭代过程中的随机统计特性对其安全性也有一定程度影响，因此对于 Hash 函数迭代过程中输出序列的随机性进行分析对于研究 Hash 函数的安全性是有重要意义的。运用概率统计的思想方法来研究 Hash 函数的一些性质必将为密码分析人员提供一些新的攻击密码方案的新方法和新思路，也必将使得将概率统计应用于密码学中成为一个新的研究热点。而 MD5 等 Hash 函数的破译也意味着研究 SHA-224、SHA-256、SHA-384 及 SHA-512 的密码系统的迫切性和重要性。

SHA-256 是使用最广泛的一种 Hash 函数。本文针对 SHA-256，用已有统计检测方法中的 χ^2 检验对其进行了随机性测试（包括频数检验，跟随检验，游程检验）及雪崩效应的测试，对每种测试都取了两种有代表性的输入：有规律的输入和随机的输入。最后对测试结果进行了分析讨论，得出结论：

1. SHA-256 在进行迭代过程中，随着迭代次数的增加，从整体上来看，所得到的输出序列的随机性是越来越好的。

2. 在随机性逐渐变好的过程中，第 27 轮，30 轮，43 轮和 59 轮与其对应的前一轮或者后一轮相比随机性很不好。

由以上结论可知，SHA-256 算法存在着一定的不足之处，这些结论也将为密码分析人员提供有用的密码攻击方案的新方法和新思路。

关键词：密码学；Hash 函数；随机序列；SHA-256；MD5；随机测试；雪崩效应；频数检验；跟随测试；游程检验

Abstract

In the cryptology ,the Hash function can be used in the data integrity and the news authentication as well as the digital signature, the latest collision attack for MD5, SHA-1 and so on showed that the random statistical properties in the internal iterative affect the security of a Hash function.So it's important to study the random output sequence in the Hash function iterative process for the security of the Hash function.Using of the probability and statistics to study the nature of the Hash function will not only provide some new methods and ideas for Password analysts, but also be a new study hot.The decipher of the Hash functions such as MD5 also means that it's urgent and important to study the SHA-224, SHA-256, SHA-384 and SHA-512.

SHA-256 is one kind of Hash functions which is used widespreadly. This article aims at SHA-256, We carry on the random test as well as the avalanche effect test to it with χ^2 text in the statistical examination method (including frequency tests, following tests, run-length test), two representative inputs were used for each test: regular input and random input. Finally, the test results were analyzed and discussed, concluded:

1. In iterative process for SHA-256, with the increase in the number of iterative, on the whole, the output of random sequence is more good.
2. In the process of gradually good randomness,it's a very bad compared to randomness with the first round or after for the round of 27, 30, 43 and 59.

There are certain deficiencies for SHA-256 algorithm from the above conclusions. These findings will also provide useful new methods and new ideas for the password analysts.

Keywords: Cryptography; Hash function; Random sequence; SHA-256; MD5; Random testing; Avalanche effect; Frequency of testing; Follow the test; Run test

华中师范大学学位论文原创性声明和使用授权说明

原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的研究成果。除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

作者签名：刘平　　日期：2008年 6月5日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中师范大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。同时授权中国科学技术信息研究所将本学位论文收录到《中国学位论文全文数据库》，并通过网络向社会公众提供信息服务。

作者签名：刘平　　导师签名：

日期：2008年6月5日　　日期：2008年6月5日

本人已经认真阅读“CALIS高校学位论文全文数据库发布章程”，同意将本人的学位论文提交“CALIS高校学位论文全文数据库”中全文发布，并可按“章程”中的规定享受相关权益。同意论文提交后滞后：□半年；□一年；□二年发布。

作者签名：刘平　　导师签名：

日期：2008年6月5日　　日期：2008年6月

引 言

在二十一世纪电子商务和电子政务时代，人们所面临的一个至关重要的问题就是信息安全问题。密码学是保障信息安全的核心技术，它以很小的代价提供很大的安全保护，其应用涉及军事、国防、商贸以及人们日常生活中的各方面。

密码学是一门古老而又年青的科学，它用于保护军事和外交通信可追溯到几千年前。在当今的信息时代，大量的敏感信息如病历、法庭记录、资金转移、私人财产等常常通过公共通信设施或计算机网络来进行交换，而这些信息的秘密性和真实性是人们迫切需要的。因此，现代密码学的应用已不再局限于军事、政治和外交，其商用价值和社会价值也已得到了充分肯定。

密码学是数学、计算机、通信等多学科的交叉。密码学的基础包括数论、概率论与数理统计、信息论、编码等等。概率统计主要用于对密码算法的统计性能进行测试和分析，从而给密码编码人员和密码分析人员提供一定的信息。

密码学中的 Hash 函数[1-4]（cryptography Hash functions）能够用于数据完整性和消息认证以及数字签名。其基本思想是把 Hash 函数值看成输入的消息摘要（message digest），当输入中的任何一个二进制位发生变化时都将引起 Hash 函数值的变化。

关于 Hash 的算法研究，一直是信息科学里面的一个前沿，尤其在网络技术普及的今天，他的重要性越来越突出，其实我们每天在网上进行的信息交流安全验证，我们在使用的操作系统密钥原理，里面都有它的身影。目前已经公开的代表性的 Hash 算法有 MD4、MD5、SHA-1、SHA-2、RIPEMD-128[7]等。为了满足数据完整性和消息认证的需要，Hash 函数必须满足特定的密码学要求，例如单向性和抗碰撞[7]等。

基于分组密码的 Hash 函数也是最近的研究热点之一。主要用于信息安全领域中的加密算法。针对 Hash 算法的一些弱点可对 Hash 算法进行攻击，可利用 Hash 算法的代数结构及其所使用的分组密码的弱点来攻击一些 Hash 方案。例如针对 DES 的一些弱点（即互补性、弱密钥、密钥碰撞等）来攻击基于 DES 的 Hash 方案。

新近的对于MD5等的碰撞攻击[6]表明在一个Hash函数的内部迭代过程中的随机统计特性对其安全性也有一定程度影响。因此对 Hash 函数迭代过程中输出序列的随机性进行分析对于研究 Hash 函数的安全性是有重要意义的。故运用概率统计的思想方法来研究 Hash 函数的一些性质必将为密码分析人员提供一些新的攻击密码方案的新方法和新思路，也必将使得将概率统计应用于密码学中成为一个新的研究热点。

本文主要是研究概率论与数理统计在密码学中的应用：对 Hash 函数迭代过程中输出序列的随机性进行分析。我们无法用数学方法对一个序列是否是真正的随机比特序列给出证明，但是我们可以通过使样本序列经过不同的统计测试来检测该序列所具有的某些缺点；统计测试通常用说明随机样本的统计量[6]。被选择的统计量通常易于有效计算，并且近似地服从 $N(0,1)$ 或 χ^2 分布。样本输出序列的统计量的值经过计算，然后与特定的随机序列的期望值进行比较，从而可以对该种 Hash 算法的安全性做出分析。本文的测试对象主要是针对 Hash 算法中的 SHA-256 进行的，通过一系列的统计测试，对 SHA-256 的安全性做出了分析，并指出了该算法的不足之处。。

章节安排：第一章介绍 Hash 函数；第二章介绍随机序列的统计测试；第三章介绍了几种密码算法的输出序列的随机性研究；第四章主要是第三章的结论及分析；第五章是需要进一步解决的问题及展望。

第一章 Hash 函数的介绍

密码学中的Hash函数主要用于数据完整性和消息认证以及数字签名。Hash函数是密码学领域内兴起的一种极其重要的通信工具，并且已成为密码学者最为关注的焦点之一。

Hash函数的研究是密码学和信息安全领域中的一个非常重要的基本组成部分，但是，自从以MD5为代表的系列Hash函数被我国学者王小云等人破译后，关于Hash函数的研究又重新回到了起步阶段，国际上对Hash函数的研究又成了一个热点。

1.1 Hash 函数的概念[12,16,20]

Hash 函数是一种将任意长度的消息 M 压缩到某一固定长度的消息摘要（message digest）函数，可以表示为 $h=H(M)$，其中 h 为输出值，长度为 l，M 为输入明文。

定义 1.1 若 Hash 函数 $H(M)$ 满足以下性质：

（1）压缩性：对于任意长度的消息 $M\in\{0,1\}^*$，输出的压缩值 $h=H(M)\in\{0,1\}^l$，并且计算 h 是容易的。

（2）抗计算原象性（Preimage resistance）：给定 h，计算 M 满足 $H(M)=h$ 是计算上困难的。

（3）抗计算第二原象性（2nd- Preimage resistance）：给定 M，要找到另一消息 M' 并满足 $H(M)=H(M')$ 是计算上困难的。

则 Hash 函数 $H(M)$ 称为弱单向 Hash 函数(weak one-way Hash function).

定义 1.2 若 $H(M)$ 为弱单向 Hash 函数，并且满足下列特性：

（4）抗碰撞性(Collision resistance or Free-collision)：寻找任何明文对 M、M'，并且满足 $H(M)=H(M')$ 是计算上困难的。

则 $H(M)$ 称为强单向 Hash 函数或者无碰撞 Hash 函数。一般也简称为单向 Hash 函数或散列函数。

1.2 Hash 函数的构造[18]

在密码学和信息安全领域里，几个比较著名的算法都是通过迭代压缩函数来实现的，因此，在用迭代方法设计函数时，最关键的是能否找到或设计出一个安全的压缩函数。通常设计压缩函数有两种方法：一、利用某些数学工具专为 Hash 目的

而设计一个压缩函数。二、利用现有的密码算法如分组密码来作为压缩函数等。

目前，设计Hash函数的基本方法有以下几种：

1）利用某些数学难题比如因子分解问题、离散对数问题等设计Hash函数。已设计出的算法有Davies-Price 平方Hash算法、CCITT 建议、Jueneman Hash算法、Damgard平方Hash算法、Damgard 背包Hash算法、Schnorr 的FFT Hash算法等。

2）利用某些私钥密码体制比如DES 等设计Hash函数。这种Hash函数的安全性与所使用的基础密码算法有关。这类Hash算法有Rabin Hash算法、Winternitz Hash算法、Quisquater-Girault Hash算法、Merkle Hash算法、N-Hash 算法等。

3）直接设计Hash函数，这类算法不基于任何假设和密码体制。这种方法受到人们的广泛关注和青睐，是当今比较流行的一种设计方法。美国的安全Hash算法（SHA）就是这类算法，此类算法还有MD4、MD5、MD2、RIPE-MD、HAVAL等。

1.3 SHA 系列 Hash 函数[1]（安全 Hash 算法）

1.3.1 基本介绍

安全 Hash 算法包括 SHA-1、SHA-256、SHA-384 和 SHA-512 四种（本文主要以介绍 SHA-256 为主）。这四种算法都是迭代的单向 Hash 函数，这些 Hash 函数可以作用于一个消息（message）从而产生一个消息摘要（message digest）。这些算法能够保证一个消息的完整性，即：消息的任何一个改变都将导致消息摘要以一个很高的概率发生改变。而这个性质在数字签名、信息证实代码和产生随机数的应用中是很有用的。

安全 Hash 算法的每个算法都可以描述成两个步骤：预处理和 Hash 值的计算。预处理包括:a、消息填充，b、将填充后的消息分成若干个块，每个块都具有 m bit，c、设置初值。这些初值将在计算 Hash 值的运算中用到。Hash 值的计算可以从填充后的消息中产生一个消息列表，使用这个表，连同函数，常量和字操作可以反复的产生一系列的 Hash 值。由 Hash 计算产生的最终的 Hash 值是用来决定消息摘要的。

这四个算法在为混乱后的数据提供的安全比特位数上是有明显的不同的。这主要与消息摘要的长度有关。当一个安全的 Hash 算法与另一个算法一起使用时，可能需要一些特定的条件：需要使用一个具有一定安全比特位数的安全 Hash 算法。例如：如果一个有正负之分的消息使用了一个提供了 128 位安全比特数的数字签名算法，那么这个数字签名算法可能就需要使用一个也提供 128 位安全比特数的安全 Hash 算法（如 SHA-256）.

另外，这四个算法在分块的大小也是不同的。下面的表描述了这四个安全 Hash

算法的一些基本性质：

表 1:安全 Hash 算法的性质

算法	消息长度 (bits)	分块长度 (bits)	字长度 (bits)	消息摘要长度 (bits)	安全位数 (bits)
SHA-1	$<2^{128}$	512	32	160	80
SHA-256	$<2^{64}$	512	32	256	128
SHA-384	$<2^{128}$	1024	64	384	192
SHA-512	$<2^{128}$	1024	64	512	256

1.3.2 安全 Hash 算法 SHA-256

SHA-256 是用来对长为l-bit（其中，$0\le l<2^{64}$）的消息M进行 Hash 计算。这个算法使用了：

- 64 个 32-bit 的字的消息表；
- 8 个 32-bit 的工作变量；
- 8 个 32-bit 字的 Hash 值。

最终，SHA-256 的输出 Hash 值为一个 256-bit 的消息摘要。

消息表中的 64 个字分别记为：W_0，W_1，…，W_{63}。8 个工作变量记为：a，b，c，d，e，f，g，h。第i轮的 8 个 32-bit 字的 Hash 值分别记为：$H_0^{(i)}$，$H_1^{(i)}$，$H_2^{(i)}$，$H_3^{(i)}$，$H_4^{(i)}$，$H_5^{(i)}$，$H_6^{(i)}$，$H_7^{(i)}$，其中初值为$H^{(0)}$，即$H_0^{(0)}$，$H_1^{(0)}$，$H_2^{(0)}$，$H_3^{(0)}$，$H_4^{(0)}$，$H_5^{(0)}$，$H_6^{(0)}$，$H_7^{(0)}$；最终的 Hash 值为$H^{(N)}$。SHA-256 使用了两个临时变量T_1，T_2。SHA-256 使用了 6 个逻辑函数，每个函数都是作用于 3 个 32bit 的字x，y，z，并且输出结果也是一个 32 bit 的字。

$$Ch(x,y,z)=(x\wedge y)\oplus(\neg x\wedge z)$$

$$Maj(x,y,z)=(x\wedge y)\oplus(x\wedge z)\oplus(y\wedge z)$$

$$\sum_0^{\{256\}}(x)=ROTR^2(x)\oplus ROTR^{13}(x)\oplus ROTR^{22}(x)$$

$$\sum_1^{\{256\}}(x)=ROTR^6(x)\oplus ROTR^{11}(x)\oplus ROTR^{25}(x)$$

$$\sigma_0^{\{256\}}(x)=ROTR^7(x)\oplus ROTR^{18}(x)\oplus SHR^3(x)$$

$$\sigma_1^{\{256\}}(x)=ROTR^{17}(x)\oplus ROTR^{19}(x)\oplus SHR^{10}(x)$$

其中，$ROTR^n(x)=(x>>n)\vee(x<<w-n)$，$SHR^n(x)=x>>n$。

1. SHA-256 的预处理

- 填充消息 M；
- 将填充后的消息分解成 N 个 512-bit 的消息块：$M^{(1)}$，$M^{(2)}$，…，$M^{(N)}$。
- 给 Hash 值附初值 $H^{(0)}$。

2. SHA-256 的 Hash 计算

SHA-256 的 Hash 计算分别用到了前面提到的函数和常量。这里的加法（+）是对于模 2^{32} 而言的。

在预处理结束后，每个消息 $M^{(1)}$，$M^{(2)}$，…，$M^{(N)}$ 依次按照下面的步骤进行处理：

For i=1 to N

{

1. 准备消息表 $\{W_t\}$：

$$W_t=\begin{cases} M_t^{(i)} & 0\le t\le 15 \\ \sigma_1^{\{256\}}(W_{t-2})+W_{t-7}+\sigma_0^{\{256\}}(W_{t-15})+W_{t-16} & 16\le t\le 63 \end{cases}$$

2. 将第（$i-1$）轮的 Hash 值分别附给五个工作变量 a，b，c，d，e，f，g，h：

$$a=H_0^{(i-1)}$$
$$b=H_1^{(i-1)}$$
$$c=H_2^{(i-1)}$$
$$d=H_3^{(i-1)}$$
$$e=H_4^{(i-1)}$$
$$f=H_5^{(i-1)}$$
$$g=H_6^{(i-1)}$$
$$h=H_7^{(i-1)}$$

3. For t=0 to 63:

{

$$T_1 = h + \sum_1^{\{256\}}(e) + Ch(e,f,g) + K_t^{\{256\}} + W_t$$
$$T_2 = \sum_0^{\{256\}}(a) + Maj(a,b,c)$$
$$h = g$$
$$g = f$$
$$f = e$$
$$e = d + T_1$$
$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

}

4. 计算第i轮的Hash值$H^{(i)}$：

$$H_0^{(i)} = a + H_0^{(i-1)}$$
$$H_1^{(i)} = b + H_1^{(i-1)}$$
$$H_2^{(i)} = c + H_2^{(i-1)}$$
$$H_3^{(i)} = d + H_3^{(i-1)}$$
$$H_4^{(i)} = e + H_4^{(i-1)}$$
$$H_5^{(i)} = f + H_5^{(i-1)}$$
$$H_6^{(i)} = g + H_6^{(i-1)}$$
$$H_7^{(i)} = h + H_7^{(i-1)}$$

}

在重复1-4步N次后（即处理$M^{(N)}$后），就会产生一个256-bit的消息摘要M，即：$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(N)}$。

1.3.3 安全Hash算法的安全性

MD-5、SHA-1 是当前国际上最常用的单向散列函数。近期的研究发现，SHA-1算法和MD-4、MD-5 等单向散列函数的安全性已经受到了质疑。山东大学信息安全所所长王小云教授在2004年宣布了她的研究成果：对MD-5、HAVAL-128、MD-4和RIPEMD 等4个著名安全散列函数的破译结果。2005 年她又宣布了破译SHA-1的研究成果。

NIST 针对王小云教授的研究发现表示，为配合先进的计算机技术，美国政府5年内将不再使用SHA-1，并计划在2010年改用先进的SHA-256、SHA-384及SHA-512的

密码系统。这也就意味着研究SHA-224、SHA-256、SHA-384 及SHA-512的密码系统的迫切性和重要性。本文的研究主要是针对SHA-256的。

第二章 随机序列的统计测试

对一个密码算法来说,其输出序列的随机性是其安全性的很重要的一方面。对基于分组密码算法的Hash函数来说,其是否可以用作伪随机数产生器是评价这个算法的重要指标之一(可参看 AES 的评选报告)。所以随机性测试在密码学中占有很重要的作用。

本章将分以下几部分来介绍:2.1 随机序列的介绍;2.2 随机性测试的重要性;2.3 随机序列的统计测试。这里,我们将以第三部分为主,主要介绍随机序列统计测试的定义、统计量及其相关证明。

2.1 随机序列

在许多密码机制中,随机数生成都是一个重要的要素。例如,加密变换的密钥必须以一种不可被敌手预测的方式产生。而生成一个随机密钥通常涉及随机数或随机比特序列的选择。

随机序列常被用做密钥管理,其主要功能是产生真随机的序列来用做密钥。产生随机序列的方法有很多种,既有数学方法产生的,也有物理方法产生的,从而随机数产生器有真随机和伪随机之分。

2.1.1 真随机数产生器

随机比特生成器:一种能够输出统计上独立且无偏的二进制数字序列的设备或算法。真随机数产生器产生的随机数是不可重现的,绝大多数真随机序列源都来自物理方法,产生它们的代价大,速度慢。为此,使用伪随机序列:按一种确定的方式从一个称为种子的较短序列来构造的序列。通常生成算法是公开的,但种子除了生成序列的实体外不被人所知。

2.1.2 伪随机数产生器

伪随机比特生成器(PRBG):一种确定性算法(给定了相同的初始种子时,生成器将产生相同的输出序列),即在给定长度为k的真随机二进制序列时,能够输出一个长度为l($l>>k$)且看上去"随机"的二进制序列。PRBG 的输入称为种子,输出称为伪随机比特序列。实际上,伪随机数产生器产生的随机数并不是真的随机,其具有周期性,也就是说,其产生的随机数序列总会产生重复,不过如果产生器的周期足够长(至少要远远大于可能采集的随机数的长度),那么这个随机数产生器产生

的局部的随机序列也就和真随机序列看起来没有什么区别了。如线性同余发生器、线性反馈移位寄存器、附加式发生器等[17]。

2.2 随机性测试的重要性

不管是真随机数产生器还是伪随机数产生器,都必须通过随机性测试。对一个密码算法来说,其输出序列的随机性是其安全性的很重要的一方面。对分组密码算法来说,其是否可以看作伪随机数产生器是评价这个算法的重要指标之一(可参看AES 的评选报告[1]) 。所以随机性测试在密码学中占有很重要的作用。密码学意义上安全的随机数比其它大多数应用对随机性的要求更严格，要求满足以下3个基本的特性:

(1) 不可预测性: 也就是说,即使给出产生序列的算法或者硬件设计和以前已经产生的序列的所有知识,你也不可能通过计算来预测下一个比特是什么。

(2) 不能重复产生: 也就是说,即使在完全相同的操作条件下(同样的地点、同样的温度等) 用完全相同的输入对序列产生器操作两次,你也将得到两个完全不同的、毫不相关的位序列。

(3) 能通过随机性统计检验。即能通过我们所能找到的所有的正确的随机性检验,这也是最起码的要求。目前存在的随机性检验约有200 多个,而且根据参数的改变一个检验有时可以变换为很多个检验,要让一个算法通过所有的随机性检验是件很困难的事,但是有些基本的检验是必须得通过的,比如频数检验。算法无法通过某个随机性检验就表明算法的设计在某些方面有缺陷。这无疑也给密码分析人员或密码攻击人员提供了一定的信息。本文主要是针对SHA-256的每一轮输出序列进行了一系列的随机性统计检验，考察了SHA-256的安全性问题。

序列的随机性检测一直是信息安全领域的一个重要研究方向。Hash函数输出序列的随机性如何直接影响到整个系统的安全性[8]。所以,对Hash函数的输出序列进行随机性研究是有重大意义的。

2.3 随机序列的统计测试

随机序列的统计测试主要是用随机数产生器或者密码算法产生一串二进制序列，设计一系列的统计测试来检验其是否是真随机的或与真随机的差距。

我们无法用数学方法对一个序列是否是真正的随机比特序列给出证明，但是我们可以通过使样本序列经过不同的统计测试来检测该序列所具有的某些缺点；统计测试通常用于说明随机样本的统计量。被选择的统计量通常易于有效计算，并且近似地服从 N(0, 1)或 χ^2 分布。样本输出序列的统计量的值经过计算，然后与特定的

随机序列的期望值进行比较。在统计分析中，常常用χ^2检验来对算法的扩散性能进行检测。本文在已有的χ^2检验的理论基础上，对密码算法中的 SHA-256 的每一轮输出进行了频数检验、跟随测试、游程检验以及雪崩测试，给出了测试结果，并对测试结果进行了分析讨论，指出了该算法的一些不足之处。

2.3.1 统计检验的基础知识

定义 2.3.1：（二进制对称源）称源S为二进制对称源，若S产生的二进制随机数为R_i，$R_i \in B$，其中$B=\{0,1\}$，R_i的概率分布满足：

R_i	0	1
P	1/2	1/2

统计检验是用来发现一个随机比特生成器的一个可能的统计缺陷，也就是说，用统计模型来描述的当一个随机比特生成器的输出序列明显偏离一个二进制对称源时，可以用统计检验来发现这种情况。

现要检验测试随机比特生成器的一个长度为N的样本序列，且当样本序列的某种性质隐含着一种可能的不随机性时，就拒绝这个随机比特生成器，即认为这个随机比特生成器所输出的序列不是伪随机序列。

定义统计量T如下：

$$T：\ B^N \rightarrow \{\text{接受，拒绝}\}$$

函数T将长度为N的二进制序列集合B^N映射到一个较小的集合S_T，集合（$B^N - S_T$）去除掉集合B^N中随机性不好的序列，而保留随机性好的序列。即：

$$S_T = \{s^N : T(s^N) = \text{拒绝}\} \subseteq B^N$$

统计量T主要有两个参数：样本序列的长度N和拒绝概率（即一个随机比特生成器被拒绝的概率，在实际进行统计检测时，ρ应越小越好。）$\rho = \frac{\#S_T}{2^N}$。其中，$\#S_T$表示集合S_T中的元素的个数。

记：$f_T : B^N \rightarrow R$：$s^N \rightarrow f_T(s^N)$，则有：

$$P(f_T(R^N) \le t_1) + P(f_T(R^N) \ge t_2) = \rho$$

在实际应用中，我们一般取$P(f_T(R^N) \le t_1) \approx P(f_T(R^N) \ge t_2) \approx \frac{\rho}{2}$。势为$\#S_T = \rho 2^N$的

随机性不好的序列的集合S_T就可以定义如下：

$$S_T=\{s^N\in B^N: f_T(s^N)\le t_1 或 f_T(s^N)\ge t_2\}$$

f_T通常会选为一些比较常用的概率分布，大多情况下会选正态分布或自由度为d（d为正整数）的χ^2。主要是因为这些分布的数值表都是可查的，这样，在给定拒绝概率ρ和序列样本长度N的情况下，可以通过查表确定上限t_1和下限t_2的值。

2.3.2 频数检验

频数检验是最基本的检验，用来检验一个随机比特生成器是否具有无偏性。在进行随机性测试时，应该首先选择进行频数检验，频数检验通过后再选择进行其它检验，否则就不必选择进行其它检验了。因为频数检验不通过，其它检验肯定也不会通过，有一些检验比如线性复杂度检验会耗费大量的时间，而频数检验是速度最快的，为了节省时间，我们有必要在进行其它检验以前，先选择进行频数检验。频数检验是用来检验一个位序列中0 和1 的个数是否近似相等，这正是随机序列所应具备的[6]。

设频数检验所测试的是长度为N的样本序列为：$s^N=s_1$，s_2，…，s_N。定义：

$$f_T(s^N)=\frac{2}{\sqrt{N}}(\sum_{i=1}^{N}s_i-\frac{N}{2})$$

定理 1：当$N\to\infty$时，$f_T(s^N)=\frac{2}{\sqrt{N}}(\sum_{i=1}^{N}s_i-\frac{N}{2})\to N(0,1)$

证明：长度为N的随机序列中 1 的个数$R^N=R_1+R_2+...+R_N$，其中$R_i=\begin{cases}0,当s_i=0\\1,当s_i=1\end{cases}$（$i=1,2,...,N$）。

因为当$1\le i\le N$时：

$$E(R_i)=\frac{1}{2}，\ Var(R_i)=\frac{1}{4}$$

故R^N可以近似的认为是均值为$\frac{N}{2}$，方差为$\frac{N}{4}$的正态分布的随机变量。

所以，据中心极限定理可知，

当$N\to\infty$时，$f_T(s^N)=\frac{2}{\sqrt{N}}(\sum_{i=1}^{N}s_i-\frac{N}{2})\to N(0,1)$　证毕！

即当N足够大时，$f_T(R^N)$可以近似地认为是均值为 0，方差为 1 的正态分布，且

$t_2=-t_1\approx 2.5\cdots 3$。

定理2[6]：令n_0和n_1分别表示长度为N待测位序列s^N中0和1的个数。这里我们取频数检验所使用的统计量为：$\chi_1=\frac{(n_0-n_1)^2}{N}$，若$N$不小于10，则统计量$\chi_1=\frac{(n_0-n_1)^2}{N}$可以看作是近似地服从自由度为1的$\chi^2$分布。

注：在实际使用中建议样本数量$N>>10000$。本文采用的是定理2中的χ^2统计量χ_1。

2.3.3 跟随测试 (又称序列测试或双比特测试)

该测试的目的是判定位序列s^N的子序列00，01，10，11所出现的次数是否近似相等，这也是一个随机序列所应具备的特性[6]。

定理3[6]：令n_0和n_1分别表示s^N中0和1的个数，且n_{00}，n_{01}，n_{10}，n_{11}分别表示S中子序列00，01，10，11出现的次数。（注意$n_{00}+n_{01}+n_{10}+n_{11}=(N-1)$，因为这些子序列允许相交）所使用的统计量为：

$$\chi_2=\frac{4}{N-1}(n_{00}^2+n_{01}^2+n_{10}^2+n_{11}^2)-\frac{2}{N}(n_0^2+n_1^2)+1$$

若N不小于21，则该统计量近似地服从自由度为2的χ^2分布。

2.3.4 游程检验

游程是序列的一个子串，由连续的0或者1组成，并且其前导和后继元素都与其本身的元素不同。游程检验主要检验待检序列s^N中游程总数是否符合随机性要求。游程测试可用来判定序列s^N中不同长度游程的个数是否与随机序列中所期待的一样。

定理4[6]：令B_i，G_i分别为s^N中长度为i的1游程和0游程的个数，所使用的统计量为：

$$\chi_3=\sum_{i=1}^{k}\frac{(B_i-e_i)^2}{e_i}+\sum_{i=1}^{k}\frac{(G_i-e_i)^2}{e_i}$$

其中，$e_i=(N-i+3)/2^{i+2}$，k是满足$e_i\geq 5$的最大的i.则统计量近似地服从自由度为$2k-2$的χ^2分布。

2.3.5 雪崩测试

（1）雪崩效应及严格雪崩准则的统计量

雪崩测试是用来测试密码算法的输入与输出的独立性：测试算法的输出是否有

不依赖于输入的统计特性。主要考虑两方面的测试：

- 如果输入具有某种明显的统计特征，算法具有较好的输入输出独立性，则输入与其对应的输出的距离应是随机的：改变输入分组的任一比特，用导致输出分组中大约一半比特的变化。
- 如果输入是随机的，则输入与其对应的输出的距离也应是随机的：改变密钥的任一比特，应导致输出分组中大约一半比特的变化。

雪崩效应是指输入任一比特的改变都应造成输出平均半数比特的改变；所谓严格雪崩准则是指输入任一比特的改变都应造成输出每一比特以1/2 的概率发生改变[6]。

$\Phi(z)=P(Z<z)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{z}e^{-\frac{u^2}{2}}du$ 表示标准正态分布的分布函数，约定下列符号：

$(.)_j$	向量的第 j 比特
$W_H(.)$	向量的汉明重量（即向量中1的个数）
$\#\{.\}$	集合的势
$P(.)$	事件发生的概率
$E(T)$	随机变量 T 的数学期望
$Var(T)$	随机变量 T 的方差
$H(.)$	一个 n 比特输入 m 比特输出的变换函数
$x=(x_1,x_2,...,x_n)$	$H(.)$ 的输入向量，
x_k	输入向量 x 的第 k 比特，$x_k\in\{0,1\}$，k=1, 2, …，n
$x^{(i)}$	仅改变 x 的第 i 比特后的输入向量，i=1, 2, …，n
$H(x)$	输入向量 x 对应的二进制输出向量
$H(x^{(i)})$	输入向量 $x^{(i)}$ 对应的二进制输出向量
X	函数 $H(.)$ 的输入变量的样本子集，且 $X\subset Z_2^n$

设函数 H 的输入变量取自样本子集 $X\subset Z_2^n$，记

$a_{ij}=\#\{x\in X\,|\,(H(x))_j\neq(H(x^{(i)}))_j\}$，$i$=1，2，…，$n$；$j$=1，2，…，$m$。

表示 X 中的输入向量 x 和 $x^{(i)}$ 对应的输出向量之间第 j 比特不同的个数；

$b_{ij}=\#\{x\in X\,|\,W_H(H(x)\oplus H(x^{(i)}))=j\}$，$i$=1，2，…，$n$；$j$=1，2，…，$m$。

表示 X 中的输入向量 x 和 $x^{(i)}$ 对应的输出向量之间的差分汉明重量为 j 的 x 的个数

（即 X 中输入向量 x 和 $x^{(i)}$ 对应的输出向量之间不同的比特数为 j 的 x 的个数）；

定理5：雪崩效应的度量可以有以下几种[2, 10, 11]：

1. $d=\sum_{i=1}^{n}\sum_{x\in X}(H(x)\oplus H(x^{(i)}))$

d 即表示分别改变 X 中每个 x 的1，2，…，n 比特后，导致输出改变的比特总数。若算法能满足雪崩效应的准则，则 $d\approx\frac{\#X\times m}{2}$。

2. $d_1=\frac{1}{n}\sum_{i=1}^{n}(\frac{1}{\#X}\sum_{x\in X}W_H(H(x)\oplus H(x^{(i)})))$

d_1 即表示分别改变 X 中每个 x 的1，2，…，n 比特后，导致输出改变的平均比特数。若算法能满足雪崩效应的准则，则 $d_1\approx\frac{m}{2}$。

3. $d_2=1-\frac{1}{n}\sum_{i=1}^{n}|\frac{1}{\#X\times m}\sum_{j=1}^{m}2jb_{ij}-1|$

$=1-|2\cdot\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\#X\times m}\sum_{j=1}^{m}2jb_{ij}-1|$

其中，$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\#X\times m}\sum_{j=1}^{m}2jb_{ij}$ 表示了分别改变 x 的1，2，…，n 比特后，导致输出改变的平均比特数，若算法能满足雪崩效应的准则，$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\#X\times m}\sum_{j=1}^{m}2jb_{ij}\approx\frac{1}{2}$，故 $d_2\approx 1$。

严格雪崩效应的度量为：

4. $d_3=1-\frac{\sum_{i=1}^{n}\sum_{j=1}^{m}|\frac{2a_{ij}}{\#X}-1|}{nm}$

$\frac{a_{ij}}{\#X}$ 即表示改变 x 的第 i 比特后，导致输出第 j 比特发生改变的概率，若算法能满足严格雪崩效应的准则，则 $d_3\approx 1$。

（2）统计量 d_1、d_2 和 d_3 的分布[2]

1. 基于 a_{ij} 的有关分布

为了简化推导，不妨设 $\#X$ 为偶数。

定理6：若 $H(.)$ 输出随机，记 $P_z=\frac{C_{\#X}^{z}}{2^{\#X}}$，则

- $a_{ij} \sim B(\#X, \frac{1}{2})$
- 令 $T = |a_{ij} - E(a_{ij})|$，T 表示 a_{ij} 的偏差，则 T 的分布如下：

$$P(T=z)=\begin{cases} 2P_{\#X/2-z} & 0<T\le\frac{\#X}{2} \\ P_{\#X/2-z} & T=0 \end{cases}$$

且当 $\#X$ 足够大时，$\frac{2T}{\sqrt{\#X}}$ 近似服从单边正态分布。

证明：

- $\because\ a_{ij}=\#\{x\in X \mid (H(x))_j \neq (H(x^{(i)}))_j\}$，

  i=1，2，…，n；j=1，2，…，m。

$\therefore a_{ij}$ 表示 X 中的输入向量 x 和 $x^{(i)}$ 对应的输出向量之间第 j 比特不同的个数；

又 $P((H(x))_j=(H(x^{(i)}))_j)$

$=P((H(x))_j\neq(H(x^{(i)}))_j)$

$=\frac{1}{2}$

$$\therefore\ P(a_{ij}=z)=P_z=\binom{\#X}{z}\left(\frac{1}{2}\right)^z\left(\frac{1}{2}\right)^{\#X-z}$$

$$=\binom{\#X}{z}\left(\frac{1}{2}\right)^{\#X}$$

$$=\frac{C_{\#X}^z}{2^{\#X}}$$

即 $a_{ij} \sim B(\#X, \frac{1}{2})$，也就是说 a_{ij} 是服从二项分布的随机变量。

- $\because T=|a_{ij}-E(a_{ij})|$ 表示 a_{ij} 的偏差

  $\therefore T\in\left\{0,1,...,\frac{\#X}{2}\right\}$　由前面的证明可知，$E(a_{ij})=\frac{\#X}{2}$

  当 $z=0$ 时，$P(T=z)=P(T=0)=P(a_{ij}=\frac{\#X}{2})=P_{\#X/2}=P_{\#X/2-z}$；

  当 $z\neq 0$ 时，$P(T=z)=P(a_{ij}=E(a_{ij})\pm z)=P(a_{ij}=\frac{\#X}{2}\pm z)$

又因为 $a_{ij} \sim B(\#X, \frac{1}{2})$，所以 a_{ij} 的概率分布应关于其均值 $\frac{\#X}{2}$ 对称

所以 $P(a_{ij} = \frac{\#X}{2} \pm z) = 2P(a_{ij} = \frac{\#X}{2} - z) = 2P_{\#X/2-z}$

即 当 $z \neq 0$ 时，$P(T = z) = 2P_{\#X/2-z}$

综合知，T 的分布如下：

$$P(T=z) = \begin{cases} 2P_{\#X/2-z} & 0 < T \le \frac{\#X}{2} \\ P_{\#X/2-z} & T = 0 \end{cases}$$

下面来证明：当 $\#X$ 足够大时，$\frac{2T}{\sqrt{\#X}}$ 近似服从单边正态分布。

（注：单边正态分布的密度函数为：$f(u) = \sqrt{\frac{2}{\pi}} e^{-\frac{u^2}{2}}$，$u > 0$）

$\because E(a_{ij}) = \frac{\#X}{2}$，$Var(a_{ij}) = \frac{\#X}{4}$

$\therefore \frac{|a_{ij} - E(a_{ij})|}{\sqrt{Var(a_{ij})}} = \frac{2T}{\sqrt{\#X}}$，而当 $\#X$ 足够大时，由中心极限定理可知，

$\frac{|a_{ij} - E(a_{ij})|}{\sqrt{Var(a_{ij})}}$ 可近似看作是标准正态分布，

$$\therefore P(\frac{2T}{\sqrt{\#X}} < u) = P(\frac{|a_{ij} - E(a_{ij})|}{\sqrt{Var(a_{ij})}} < u)$$

$$= P(-u < \frac{a_{ij} - E(a_{ij})}{\sqrt{Var(a_{ij})}} < u)$$

$$= P(\frac{a_{ij} - E(a_{ij})}{\sqrt{Var(a_{ij})}} < u) + P(\frac{a_{ij} - E(a_{ij})}{\sqrt{Var(a_{ij})}} < -u)$$

$$= \Phi(u) + \Phi(-u)$$

$$= 2\Phi(u) - 1$$

故 $\frac{2T}{\sqrt{\#X}}$ 的密度函数为：$f(u) = (2\Phi(u) - 1)' = 2\Phi'(u) = \sqrt{\frac{2}{\pi}} e^{-\frac{u^2}{2}}$，

所以，当 $\#X$ 足够大时，$\frac{2T}{\sqrt{\#X}}$ 近似服从单边正态分布。证毕！

设u服从单边正态分布，记$V=\frac{2T}{\#X}$，则

$$P(V=z)=P(\frac{2T}{\#X}=z)=P(T=\frac{\#X}{2}\bullet z)；$$

$$Var(V)=Var(\frac{2T}{\#X})=Var(\frac{2T}{\sqrt{\#X}}\bullet\frac{1}{\sqrt{\#X}})=\frac{Var(u)}{\#X}；$$

$$E(V)=E(\frac{2T}{\#X})=E(\frac{2T}{\sqrt{\#X}}\bullet\frac{1}{\sqrt{\#X}})=\frac{E(u)}{\sqrt{\#X}}。$$

下面来计算$E(u)$和$Var(u)$。

$$E(u)=\int_0^{+\infty}\sqrt{\frac{2}{\pi}}xe^{-\frac{x^2}{2}}dx=-\sqrt{\frac{2}{\pi}}\int_0^{+\infty}e^{-\frac{x^2}{2}}d(-\frac{x^2}{2})=-\sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}}\Big|_0^{+\infty}=\sqrt{\frac{2}{\pi}}$$

$$E(u^2)=\int_0^{+\infty}x^2\sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}}dx=\sqrt{\frac{2}{\pi}}\int_0^{+\infty}x^2e^{-\frac{x^2}{2}}dx=\sqrt{\frac{2}{\pi}}\int_0^{+\infty}x^2e^{-\frac{x^2}{2}}dx$$

$$=-\sqrt{\frac{2}{\pi}}\int_0^{+\infty}xde^{-\frac{x^2}{2}}=-\sqrt{\frac{2}{\pi}}(xe^{-\frac{x^2}{2}}\Big|_0^{+\infty}-\int_0^{+\infty}e^{-\frac{x^2}{2}}dx)=\sqrt{\frac{2}{\pi}}\int_0^{+\infty}e^{-\frac{x^2}{2}}dx$$

$$=\sqrt{\frac{2}{\pi}}\times\sqrt{2\pi}=1$$

$$\because Var(u)=E(u^2)-(E(u))^2$$

$$\therefore Var(u)=1-(\sqrt{\frac{2}{\pi}})^2=\frac{\pi-2}{\pi}$$

现设有N个独立样本V_1，V_2，…，V_N，N足够大，记$VN=\frac{\sum_{i=1}^{N}V_i}{N}$，则由中心极限定理知，$VN$的分布为：

$$P(VN<z)=P(\frac{\sum_{i=1}^{N}V_i}{N}<z)=P(\sum_{i=1}^{N}V_i<Nz)$$

$$=P(\frac{\sum_{i=1}^{N}V_i-NE(V)}{\sqrt{N\times Var(V)}}<\frac{Nz-NE(V)}{\sqrt{N\times Var(V)}})$$

$$=\Phi(\frac{N(z-E(V))}{\sqrt{N\times Var(V)}})$$

$$=\Phi(\frac{N(z-\frac{E(u)}{\sqrt{\#X}})}{\sqrt{N\times\frac{Var(u)}{\#X}}})$$

$$=\Phi(\sqrt{\frac{N}{Var(u)}}(z\times\sqrt{\#X}-E(u)))$$

由前面的介绍知，统计量$V=\frac{2T}{\#X}$反映了随机变量的偏差，而统计量VN反映的是N个独立同分布样本的统计平均。

2. 基于b_{ij}的有关分布

定理7：若$H(.)$输出随机，令$U=\sum_{j=1}^{m}jb_{ij}$，记$P(U=z)=P_z=\frac{C_{\#X\times m}^{z}}{2^{\#X\times m}}$，则

- $U\sim B(\#X\times m,\frac{1}{2})$
- 令$W=|\frac{2U}{\#X\times m}-1|$，则$W$的分布为：

$$P(W=z)=\begin{cases}2P_{(1-z)\times z\times \#X/2} & 0<W\le 1\\ P_{(1-z)\times z\times \#X/2} & W=0\end{cases}$$

且当$\#X$足够大时，$\sqrt{\#X\times m}W$可近似看作服从单边正态分布。

证明：

- $U=\sum_{j=1}^{m}jb_{ij}$表示改变X中的所有输入向量x的第i比特所导致的输出向量的总的改变的比特数。

  若$H(.)$输出随机，则改变输入向量x的第i比特导致输出向量的每一比特发生改变的概率都是$\frac{1}{2}$，所以

$$P(U=z)=P_z=\binom{\#X\times m}{z}(\frac{1}{2})^z(\frac{1}{2})^{\#X\times m-z}$$
$$=\binom{\#X\times m}{z}(\frac{1}{2})^{\#X\times m}=\frac{C^z_{\#X\times m}}{2^{\#X\times m}}$$

即$U\sim B(\#X\times m,\frac{1}{2})$。

- $W=|\frac{2U}{\#X\times m}-1|$表示了随机变量$W$的归一化偏差，则

当$z=0$时，

$P(W=z)=P(W=0)=P(\frac{2U}{\#X\times m}=1)=P(U=\frac{\#X\times m}{2})=P_{\#X\times m/2}=P_{(1-z)\times\#X\times m/2}$；

当$z\neq 0$时，

$P(W=z)=P(|\frac{2U}{\#X\times m}-1|=z)=P(\frac{2U}{\#X\times m}-1=\pm z)=P(U=(1\pm z)\times\frac{\#X\times m}{2})$

又因为$U\sim B(\#X\times m,\frac{1}{2})$，所以$U$的概率分布应关于其均值$\frac{\#X\times m}{2}$对称

所以 $P(U=(1\pm z)\times\frac{\#X\times m}{2})=2P(U=(1-z)\times\frac{\#X\times m}{2})=2P_{(1-z)\times\#X\times m/2}$

即 当$z\neq 0$时，$P(W=z)=2P_{(1-z)\times\#X\times m/2}$

综合知，W的分布如下：

$$P(W=z)=\begin{cases}2P_{(1-z)\times z\times\#X/2} & 0<W\leq 1\\ P_{(1-z)\times z\times\#X/2} & W=0\end{cases}$$

下面证明，且当$\#X$足够大时，$\sqrt{\#X\times m}W$可近似看作服从单边正态分布。

因为 $\sqrt{\#X\times m}W=|\frac{U-\frac{1}{2}(\#X\times m)}{\frac{1}{2}\sqrt{\#X\times m}}|=|\frac{U-E(U)}{\sqrt{Var(U)}}|$，接下来就是证明当$\#X$足够大时，$|\frac{U-E(U)}{\sqrt{Var(U)}}|$可近似看作服从单边正态分布，这与定理6中证明当$\#X$足够大时，$\frac{2T}{\sqrt{\#X}}$近似服从单边正态分布相似，略。证毕！

由定理7中，当“$\#X$足够大时，$\sqrt{\#X\times m}W$可近似看作服从单边正态分布”可知：

$$E(W)=E(\frac{\sqrt{\#X\times m}W}{\sqrt{\#X\times m}})=E(\frac{u}{\sqrt{\#X\times m}})=\frac{1}{\sqrt{\#X\times m}}E(u)$$

$$Var(W)=Var(\frac{\sqrt{\#X\times m}W}{\sqrt{\#X\times m}})=Var(\frac{u}{\sqrt{\#X\times m}})=\frac{1}{\#X\times m}Var(u)$$

现设有 N 个独立同分布样本 W_1，W_2，…，W_N，N 足够大，记 $WN=\frac{\sum_{i=1}^{N}W_i}{N}$，则由中心极限定理有，$WN$ 的概率分布为：

$$P(WN<z)=P(\frac{\sum_{i=1}^{N}W_i}{N}<z)=P(\sum_{i=1}^{N}W_i<Nz)$$

$$=P(\frac{\sum_{i=1}^{N}W_i-NE(W)}{\sqrt{N\times Var(W)}}<\frac{Nz-NE(W)}{\sqrt{N\times Var(W)}})$$

$$=\Phi(\frac{N(z-E(W))}{\sqrt{N\times Var(W)}})$$

$$=\Phi(\sqrt{\frac{N}{Var(u)}}(\sqrt{\#X\times m}\times z-E(u)))$$

4. 统计量 d_1、d_2、d_3 的分布

如果 $F(\bullet)$ 是随机变换，则 $F(\bullet)$ 变换就具有很好的雪崩效应，满足严格雪崩准则，可以近似的认为 a_{ij}，b_{ij} 是独立同分布的。

- $\because d_2=1-\frac{1}{n}\sum_{i=1}^{n}|\frac{1}{\#X\times m}\sum_{j=1}^{m}2jb_{ij}-1|$

  $\therefore 1-d_2=\frac{1}{n}\sum_{i=1}^{n}|\frac{1}{\#X\times m}\sum_{j=1}^{m}2jb_{ij}-1|=\frac{1}{n}\sum_{i=1}^{n}W_i$，故 $1-d_2$ 是与 W 同分布的 n 个独立随机变量的统计平均。

故：$E(d_2)=1-E(W)=1-\frac{E(u)}{\sqrt{\#X}}$，$Var(d_2)=Var(\frac{1}{n}\sum_{i=1}^{n}W_i)=\frac{1}{n}Var(W)=\frac{1}{n}\frac{Var(u)}{m\times\#X}$。

- $\because d_3=1-\frac{\sum_{i=1}^{n}\sum_{j=1}^{m}|\frac{2a_{ij}}{\#X}-1|}{nm}$

  $\therefore 1-d_3=\frac{\sum_{i=1}^{n}\sum_{j=1}^{m}|\frac{2a_{ij}}{\#X}-1|}{nm}$，

$$又V=\frac{2T}{\#X}=\frac{2|a_{ij}-E(a_{ij})|}{\#X}=|\frac{2a_{ij}}{\#X}-\frac{2\times\frac{1}{2}\times\#X}{\#X}|=|\frac{2a_{ij}}{\#X}-1|$$

$$\therefore 1-d_3=\frac{\sum_{i=1}^{n}\sum_{j=1}^{m}|\frac{2a_{ij}}{\#X}-1|}{nm}=\frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}V_{ij}=\frac{1}{nm}\sum_{i=1}^{nm}V_i$$，故 $1-d_3$ 是与 V 同分布的 nm 个独立随机变量的统计平均。故：

$$E(d_3)=1-E(V)=1-\frac{E(u)}{\sqrt{\#X}},\ Var(d_3)=Var(\frac{1}{nm}\sum_{i=1}^{nm}V_i)=\frac{1}{nm}Var(V)=\frac{1}{nm}\frac{Var(u)}{\#X},$$

下面来求 d_1、d_2、d_3 的置信区间。

定理8：给定置信水平 α，$z_{1-\alpha/2}$ 表示标准正态分布的 $1-\alpha/2$ 分位点，则有：

- $E(d_1)=\frac{m}{2}$，d_1 的置信区间为：$d_1\in[\frac{m}{2}-\frac{z_{1-\alpha/2}}{2}\sqrt{\frac{m}{\#X}},\frac{m}{2}+\frac{z_{1-\alpha/2}}{2}\sqrt{\frac{m}{\#X}}]$；
- d_2 的置信区间为：$d_2\in[E(d_2)-z_{1-\alpha/2}\sqrt{\frac{1}{n}\sqrt{\frac{Var(u)}{m\times\#X}},E(d_2)+z_{1-\alpha/2}\sqrt{\frac{1}{n}\sqrt{\frac{Var(u)}{m\times\#X}}}}]$
- d_3 的置信区间为：$d_3\in[E(d_3)-z_{1-\alpha/2}\sqrt{\frac{1}{nm}\sqrt{\frac{Var(u)}{\#X}},E(d_3)+z_{1-\alpha/2}\sqrt{\frac{1}{nm}\sqrt{\frac{Var(u)}{\#X}}}}]$

其中，$E(u)=\sqrt{\frac{2}{\pi}}$，$Var(u)=\frac{\pi-2}{\pi}$。

2.3.6 完全性

（1）完全性及完全性的统计量

所谓完全性是指加密函数输出的每一比特都与输入的所有比特有关[6]。完全性程度的度量可以用下面的统计量来表示：

$$d_4=1-\frac{\#\{(i,j)|a_{ij}=0,i=1,...,n;j=1,...,m\}}{mn}$$

可知，$\#\{(i,j)|a_{ij}=0,i=1,...,n;j=1,...,m\}$ 表示分别改变第1，2，…，n比特后，输出不变的比特数。故当一个算法输出随机时，则有

$\#\{(i,j)|a_{ij}=0,i=1,...,n;j=1,...,m\}=0$，即 $d_4=1$。

（2）d_4 的分布

由2.3.5中关于 a_{ij} 的分布可以知：$P(d_4=1)=1-(\frac{1}{2})^{\#X}\approx 1.000000$

第三章 SHA-256 输出序列的随机性研究

新近的对于MD5等的碰撞攻击表明在一个Hash函数的内部迭代过程中的随机统计特性对其安全性也有一定程度影响。因此对于Hash函数迭代过程中输出序列的随机性进行分析对于研究Hash函数的安全性是有意义的。

我们无法用数学方法对一个序列是否是真正的随机比特序列给出证明，但是我们可以通过使样本序列经过不同的统计测试来检测该序列所具有的某些缺点；统计测试通常用于说明随机样本的统计量。被选择的统计量通常易于有效计算，并且近似地服从 N(0,1)或 χ^2 分布。样本输出序列的统计量的值经过计算，然后与特定的随机序列的期望值进行比较。在统计分析中，常常用 χ^2 检验来对算法的扩散性能进行检测。

在文献[3]中，给出了对 Hash 算法扩散性能的偏差分析，并给出了 MD5 算法的运用实例。偏差分析是对整个输入/输出关系做统一的非线性度量分析，这也是其局限性所在：不能较为细致地刻画算法输入/输出间的依赖关系[3]。本章在已有的 χ^2 检验的理论基础上，对密码算法中的 SHA-256 的每一轮输出进行了频数检验、跟随测试、游程检验以及雪崩测试，给出了测试结果。而测试结果分析讨论将放在第四章中进行。

3.1 SHA-256 的频数检验

我们测试100组输入数据，并把100组输入相应的每轮输出串成一串，这样就串成了一个长度为25600的位序列，然后对该序列进行频数检验。

当输入为随机（用Visual C++中的rand()函数）的时候，测试结果为表2（如下）：

表2：频数检验观测值

迭代轮数	频数检验观测值X_i	X_i的置信区间（给定置信水平 α=0.05）
0	39.690000	（0.00393，3.841）
1	0.202500	（0.00393，3.841）
2	0.005625	（0.00393，3.841）
5	0.507656	（0.00393，3.841）
16	0.810000	（0.00393，3.841）

20	0.787656	(0.00393，3.841)
21	0.832656	(0.00393，3.841)
22	0.068906	(0.00393，3.841)
23	0.438906	(0.00393，3.841)
26	0.581406	(0.00393，3.841)
27	2.847656	(0.00393，3.841)
28	0.113906	(0.00393，3.841)
29	0.001406	(0.00393，3.841)
30	8.122500	(0.00393，3.841)
31	0.022500	(0.00393，3.841)
32	4.840000	(0.00393，3.841)
33	0.015625	(0.00393，3.841)
34	1.722656	(0.00393，3.841)
60	0.026406	(0.00393，3.841)
61	1.440000	(0.00393，3.841)
62	0.010000	(0.00393，3.841)
63	0.302500	(0.00393，3.841)

3.2 SHA-256 的跟随测试

与 3.1 中的频数检验的测试方法一样，也是测试 100 组随机的输入数据，输入为随机（用 Visual C++中的 rand()函数）时，测试结果如下（表 3）：

表3：跟随测试观测值

迭代轮数	跟随测试观测值 X_2	X_2的置信区间（给定置信水平 α=0.05）
0	46.393792	(0.103，5.991)
1	4.756248	(0.103，5.991)
2	0.872964	(0.103，5.991)
5	0.550315	(0.103，5.991)
16	1.363327	(0.103，5.991)
20	0.973780	(0.103，5.991)

21	0.919092	(0.103, 5.991)
22	0.187158	(0.103, 5.991)
23	2.312568	(0.103, 5.991)
26	0.960959	(0.103, 5.991)
27	3.419893	(0.103, 5.991)
28	1.318767	(0.103, 5.991)
29	0.233095	(0.103, 5.991)
30	8.793590	(0.103, 5.991)
31	1.899880	(0.103, 5.991)
32	6.908389	(0.103, 5.991)
33	1.858315	(0.103, 5.991)
34	1.841975	(0.103, 5.991)
60	0.511857	(0.103, 5.991)
61	2.956226	(0.103, 5.991)
62	1.492988	(0.103, 5.991)
63	0.331079	(0.103, 5.991)

3.3 SHA-256 的游程检验

测试100组输入数据，并把100组输入相应的每轮输出串成一串，这样就串成了一个长度为25600的位序列，本文对SHA-256的游程检验所测试的是这输出序列25600比特中的前20000比特，并把长度大于6的游程全部看成长度为6的游程。这样序列中各种长度的0游程个数和1游程个数的界要满足下列条件（其中n0[i]表示序列中长度为i的0游程的个数； n1[i] 表示序列中长度为i的1游程的个数。i=1, 2, 3, 4, 5, 6）：

n0[1]，n1[1]：2267-2733；n0[2]，n1[2]：1079-1421；

n0[3]，n1[3]： 502-748；n0[4]，n1[4]： 223-402；

n0[5]，n1[5]： 90-223；n0[6]，n1[6]： 90-223；

我们测试了两类输入数据：

a. 有规律的输入，输入消息具有特殊规律，如x0=0, x1=1, …, x99=99(即每个输入较前一个输入增加1)时，测试结果为表4及表5：

表4（0游程的测试结果）：

跌代拍数	n0[1]	n0[2]	n0[3]	n0[4]	n0[5]	n0[6]
	2267-2733	1079-1421	502-748	223-402	90-223	90-223
0	n0[1]=2107	n0[2]=1249	n0[3]=782	n0[4]=235	n0[5]=312	n0[6]=79
1	n0[1]=2190	n0[2]=1250	n0[3]=937	n0[4]=391	n0[5]=156	n0[6]=78
2	n0[1]=2188	n0[2]=1171	n0[3]=858	n0[4]=391	n0[5]=78	n0[6]=78
9	n0[1]=2266	n0[2]=1563	n0[3]=546	n0[4]=313	n0[5]=156	n0[6]=78
10	n0[1]=2267	n0[2]=1482	n0[3]=470	n0[4]=312	n0[5]=313	n0[6]=0
11	n0[1]=2266	n0[2]=1560	n0[3]=390	n0[4]=156	n0[5]=234	n0[6]=158
12	n0[1]=2813	n0[2]=1094	n0[3]=313	n0[4]=234	n0[5]=78	n0[6]=235
26	n0[1]=2520	n0[2]=1250	n0[3]=643	n0[4]=300	n0[5]=151	n0[6]=158
27	n0[1]=2142	n0[2]=1691	n0[3]=572	n0[4]=267	n0[5]=134	n0[6]=142
28	n0[1]=2379	n0[2]=1260	n0[3]=657	n0[4]=316	n0[5]=149	n0[6]=164
30	n0[1]=2077	n0[2]=1072	n0[3]=542	n0[4]=260	n0[5]=140	n0[6]=214
31	n0[1]=2455	n0[2]=1217	n0[3]=633	n0[4]=282	n0[5]=171	n0[6]=169
41	n0[1]=2579	n0[2]=1259	n0[3]=611	n0[4]=291	n0[5]=155	n0[6]=145
42	n0[1]=2548	n0[2]=1246	n0[3]=581	n0[4]=298	n0[5]=157	n0[6]=163
43	n0[1]=2206	n0[2]=1636	n0[3]=530	n0[4]=254	n0[5]=153	n0[6]=154
44	n0[1]=2534	n0[2]=1245	n0[3]=589	n0[4]=278	n0[5]=159	n0[6]=177
58	n0[1]=2450	n0[2]=1196	n0[3]=642	n0[4]=333	n0[5]=171	n0[6]=155
59	n0[1]=2144	n0[2]=1636	n0[3]=551	n0[4]=323	n0[5]=153	n0[6]=138
60	n0[1]=2347	n0[2]=1240	n0[3]=624	n0[4]=361	n0[5]=181	n0[6]=157
63	n0[1]=2456	n0[2]=1226	n0[3]=607	n0[4]=324	n0[5]=166	n0[6]=165

表5（1游程的测试结果）：

跌代拍数	n1[1]	n1[2]	n1[3]	n1[4]	n1[5]	n1[6]
	2267-2733	1079-1421	502-748	223-402	90-223	90-223
0	n1[1]=2345	n1[2]=1249	n1[3]=390	n1[4]=468	n1[5]=0	n1[6]=313
1	n1[1]=2579	n1[2]=1485	n1[3]=312	n1[4]=313	n1[5]=78	n1[6]=234
2	n1[1]=2341	n1[2]=1250	n1[3]=313	n1[4]=314	n1[5]=234	n1[6]=313
9	n1[1]=2188	n1[2]=1015	n1[3]=937	n1[4]=469	n1[5]=313	n1[6]=0
10	n1[1]=2266	n1[2]=1016	n1[3]=703	n1[4]=390	n1[5]=391	n1[6]=78
11	n1[1]=1717	n1[2]=1562	n1[3]=860	n1[4]=235	n1[5]=234	n1[6]=156

12	n1[1]=1799	n1[2]=1173	n1[3]=859	n1[4]=390	n1[5]=468	n1[6]=78
26	n1[1]=2531	n1[2]=1262	n1[3]=599	n1[4]=333	n1[5]=154	n1[6]=142
27	n1[1]=2146	n1[2]=1676	n1[3]=550	n1[4]=311	n1[5]=135	n1[6]=131
28	n1[1]=2438	n1[2]=1249	n1[3]=612	n1[4]=308	n1[5]=152	n1[6]=167
30	n1[1]=2068	n1[2]=1116	n1[3]=566	n1[4]=283	n1[5]=126	n1[6]=146
31	n1[1]=2369	n1[2]=1290	n1[3]=638	n1[4]=310	n1[5]=147	n1[6]=172
41	n1[1]=2506	n1[2]=1281	n1[3]=614	n1[4]=331	n1[5]=139	n1[6]=168
42	n1[1]=2458	n1[2]=1258	n1[3]=637	n1[4]=316	n1[5]=160	n1[6]=163
43	n1[1]=2135	n1[2]=1652	n1[3]=562	n1[4]=278	n1[5]=144	n1[6]=162
44	n1[1]=2456	n1[2]=1257	n1[3]=618	n1[4]=334	n1[5]=165	n1[6]=151
58	n1[1]=2496	n1[2]=1197	n1[3]=610	n1[4]=318	n1[5]=161	n1[6]=164
59	n1[1]=2194	n1[2]=1650	n1[3]=533	n1[4]=294	n1[5]=147	n1[6]=127
60	n1[1]=2475	n1[2]=1198	n1[3]=614	n1[4]=293	n1[5]=163	n1[6]=167
63	n1[1]=2492	n1[2]=1178	n1[3]=616	n1[4]=317	n1[5]=171	n1[6]=170

b. 随机的输入，输入消息随机产生（用Visual C++中的rand()函数）时，测试结果为表6及表7：

表6（0游程的测试结果）：

跌代拍数	n0[1]	n0[2]	n0[3]	n0[4]	n0[5]	n0[6]
	2267-2733	1079-1421	502-748	223-402	90-223	90-223
0	n0[1]=2458	n0[2]=1357	n0[3]=474	n0[4]=249	n0[5]=344	n0[6]=44
1	n0[1]=2481	n0[2]=1377	n0[3]=594	n0[4]=308	n0[5]=236	n0[6]=71
2	n0[1]=2255	n0[2]=1538	n0[3]=580	n0[4]=300	n0[5]=192	n0[6]=106
9	n0[1]=2463	n0[2]=1213	n0[3]=640	n0[4]=323	n0[5]=154	n0[6]=167
10	n0[1]=2537	n0[2]=1254	n0[3]=631	n0[4]=284	n0[5]=146	n0[6]=170
11	n0[1]=2142	n0[2]=1694	n0[3]=548	n0[4]=270	n0[5]=134	n0[6]=147
12	n0[1]=2560	n0[2]=1195	n0[3]=653	n0[4]=321	n0[5]=171	n0[6]=126
26	n0[1]=2497	n0[2]=1313	n0[3]=579	n0[4]=317	n0[5]=164	n0[6]=149
27	n0[1]=2206	n0[2]=1648	n0[3]=545	n0[4]=289	n0[5]=140	n0[6]=155
28	n0[1]=2439	n0[2]=1257	n0[3]=629	n0[4]=302	n0[5]=154	n0[6]=174
30	n0[1]=2580	n0[2]=1267	n0[3]=592	n0[4]=289	n0[5]=156	n0[6]=149
31	n0[1]=2425	n0[2]=1248	n0[3]=625	n0[4]=320	n0[5]=182	n0[6]=137

41	n0[1]=2472	n0[2]=1203	n0[3]=632	n0[4]=332	n0[5]=154	n0[6]=167
42	n0[1]=2565	n0[2]=1202	n0[3]=647	n0[4]=304	n0[5]=160	n0[6]=156
43	n0[1]=2218	n0[2]=1662	n0[3]=544	n0[4]=282	n0[5]=125	n0[6]=156
44	n0[1]=2593	n0[2]=1196	n0[3]=625	n0[4]=337	n0[5]=159	n0[6]=142
58	n0[1]=2473	n0[2]=1224	n0[3]=646	n0[4]=346	n0[5]=146	n0[6]=152
59	n0[1]=2173	n0[2]=1654	n0[3]=578	n0[4]=260	n0[5]=148	n0[6]=145
60	n0[1]=2486	n0[2]=1227	n0[3]=631	n0[4]=319	n0[5]=142	n0[6]=167
63	n0[1]=2516	n0[2]=1200	n0[3]=655	n0[4]=298	n0[5]=147	n0[6]=161

表7（1游程的测试结果）：

跌代拍数	n1[1]	n1[2]	n1[3]	n1[4]	n1[5]	n1[6]
	2267-2733	1079-1421	502-748	223-402	90-223	90-223
0	n1[1]=2208	n1[2]=1375	n1[3]=563	n1[4]=548	n1[5]=41	n1[6]=191
1	n1[1]=2529	n1[2]=1360	n1[3]=546	n1[4]=395	n1[5]=81	n1[6]=156
2	n1[1]=2575	n1[2]=1190	n1[3]=566	n1[4]=330	n1[5]=104	n1[6]=206
9	n1[1]=2482	n1[2]=1232	n1[3]=589	n1[4]=325	n1[5]=189	n1[6]=143
10	n1[1]=2512	n1[2]=1227	n1[3]=655	n1[4]=340	n1[5]=138	n1[6]=149
11	n1[1]=2154	n1[2]=1670	n1[3]=545	n1[4]=265	n1[5]=153	n1[6]=148
12	n1[1]=2522	n1[2]=1266	n1[3]=601	n1[4]=314	n1[5]=150	n1[6]=173
26	n1[1]=2528	n1[2]=1224	n1[3]=636	n1[4]=321	n1[5]=163	n1[6]=148
27	n1[1]=2212	n1[2]=1677	n1[3]=559	n1[4]=270	n1[5]=143	n1[6]=123
28	n1[1]=2478	n1[2]=1203	n1[3]=658	n1[4]=291	n1[5]=160	n1[6]=166
30	n1[1]=2448	n1[2]=1303	n1[3]=638	n1[4]=338	n1[5]=162	n1[6]=145
31	n1[1]=2399	n1[2]=1248	n1[3]=646	n1[4]=323	n1[5]=157	n1[6]=164
41	n1[1]=2469	n1[2]=1232	n1[3]=662	n1[4]=297	n1[5]=119	n1[6]=180
42	n1[1]=2511	n1[2]=1274	n1[3]=650	n1[4]=313	n1[5]=142	n1[6]=143
43	n1[1]=2154	n1[2]=1706	n1[3]=593	n1[4]=286	n1[5]=128	n1[6]=119
44	n1[1]=2512	n1[2]=1310	n1[3]=614	n1[4]=316	n1[5]=163	n1[6]=138
58	n1[1]=2531	n1[2]=1197	n1[3]=625	n1[4]=315	n1[5]=158	n1[6]=161
59	n1[1]=2197	n1[2]=1657	n1[3]=553	n1[4]=273	n1[5]=135	n1[6]=143
60	n1[1]=2484	n1[2]=1236	n1[3]=602	n1[4]=312	n1[5]=186	n1[6]=151
63	n1[1]=2459	n1[2]=1248	n1[3]=649	n1[4]=297	n1[5]=168	n1[6]=156

测试结果表明，在迭代轮数较少的情况下，SHA-256中间过程产生的输出序列

无法通过游程测试，具体的总结分析我们在下章给出。

3.4 SHA-256 的雪崩测试

测试64组输入数据，每次只改变输入的最后6位中的一位（在程序中表现为只改变data[15]中的后6位），并把这64组输入相应的每轮输出串成一串，这样就串成了一个长度为16384（=64*8*32）的位序列；把改变后的64组输入相应的每轮输出串成另一串，再将改变前的输出和改变后的输出进行异或，异或结果就可以用来衡量雪崩效应的好坏。如果满足雪崩效应的话，那么异或结果应在8192（=16384/2）左右变动；若与8192相差很大（如相差约为1000左右甚至更大的话），则可以认为不满足雪崩效应。

我们测试了两类输入数据：

a. 有规律的输入

在程序中表现为data[i]=0或oxaaaaaaaa或ffffffff等等，data[15]=x，其中i=0, 1, …, 15; x=0, 1, …63(即每个输入较前一个输入增加1，这样就改变了data[15]的后6位。如原来输入中data[15]=0x00000000，增加1就变为了data[15]=0x00000001；再增加1就变为了data[15]=0x00000002，以此类推)；为节省篇幅，下面只列出其中的一种有规律的输入的测试结果；

当输入为data[i]=0，data[15]=x时，测试结果为表8中第二列数据m1。

b. 随机的输入

当输入为随机（用Visual C++中的rand()函数）时，测试结果为表8中第三列数据m2。

表8： SHA-256的雪崩测试

迭代步数	不同的比特数 m1	不同的比特数 m2
0	0	0
1	0	0
2	0	0
3	0	0
11	0	0
12	0	0
13	0	0

14	0	0
15	450	2415
16	2381	4427
17	4416	6547
18	6437	8506
19	8014	8110
20	8148	8128
23	8069	8205
24	8047	8225
25	8053	8231
26	8066	8168
27	7067	7200
28	8031	8211
29	8120	8159
30	7080	7164
31	8142	8200
39	8035	8205
40	7994	8186
41	7974	8175
42	7967	8207
43	6999	7198
44	7994	8265
45	8058	8169
57	8146	8249
58	8154	8256
59	7100	7226
60	8084	8262
61	8055	8254
63	8158	8293

第四章 测试结果的分析讨论

本章主要是对第三章中得到的测试结果进行分析讨论。

4.1 分析讨论

（1） 在3.1和3.2部分中，从频数检验和跟随检验的测试结果可以发现：当输入为有规律的数时（为了节省篇幅没有列出测试结果），随着迭代次数的增加，整体上看，输出序列的随机性都比较好，但是第30轮的迭代效果不是很好，得到的χ^2值比相应的临界值大了很多；当输入为随机时（表2和表3），由图1和图2可知，随着迭代次数的增加，整体上看，输出序列的随机性都比较好。但是从测试结果可以看到，迭代到第30轮以及第32轮，所得到的χ^2值都不在置信区间内，而且比相应的临界值也大了很多。这说明了SHA-256这种Hash函数的算法在进行迭代过程中，输出序列的第30轮和32轮的随机性测试中不能通过频数检验和跟随检验。

图1：

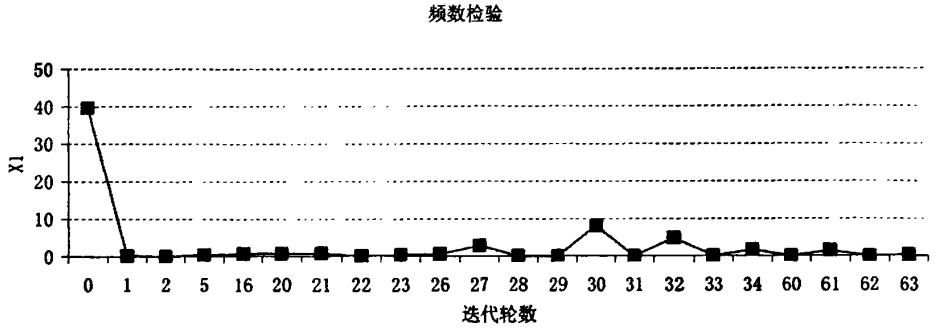
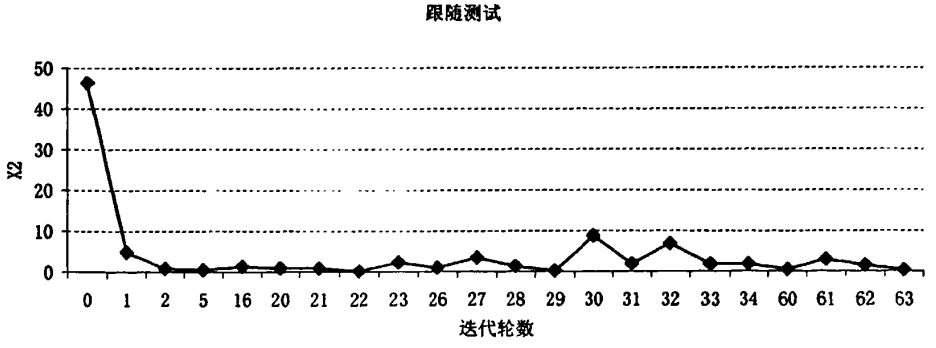


图2：



（2） 在3.3部分中，从游程检验的测试结果可以发现：当输入为有规律的数时（表4和表5），刚开始的几轮输出序列都不能通过游程测试，但随着迭代次数的增加，除了第27轮，30轮，43轮和59轮所得到的0游程和1游程的个数不在所规定的范围以内，从第18轮开始基本上能通过游程测试；当输入为随机时（表6和表7），除了第27轮，43轮和59轮所得到的输出序列不能通过游程测试外，其余的所得到的0游程和1游程的个数都在所规定的范围以内（图3给出了输入有规律和输入随机时0游程为1的游程测试情况，图4给出了输入有规律和输入随机时1游程为1的游程测试情况）。由此，我们可以看到，SHA-256这种Hash函数的算法在进行迭代过程中，输出序列的第27轮，43轮和59轮的随机性测试中不能通过游程检验。

图3：

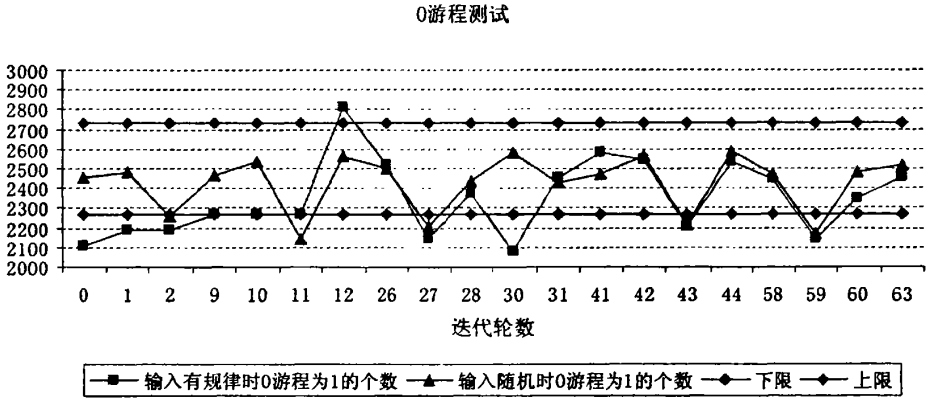
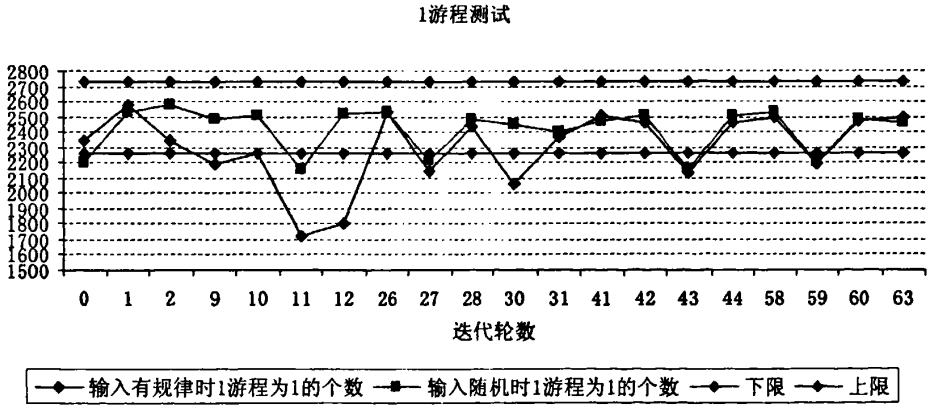
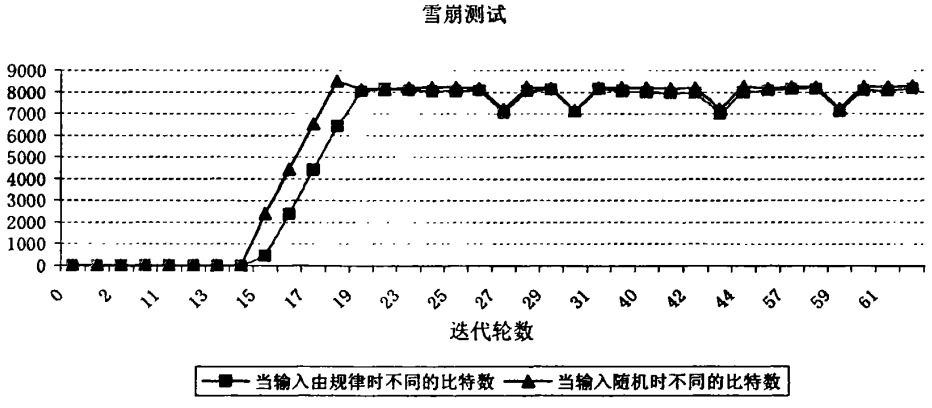


图4：



（3） 在3.4部分中，我们对SHA-256进行了雪崩效应的测试，从测试结果（表8）中可以看到：无论输入是有规律的数还是随机的数，由于测试时只改变了输入的最后6位，也就是只改变了程序中的data[15]，故输入改变前和改变后的前15轮所得到的输出是一样的，异或结果均为0；但随着迭代次数的增加，改变前的输入所得到的输出与改变后的输入所得到的输出的不同的位数应越来越多，直到在8192左右变化。从测试结果可以看到（图5），前15轮的异或结果均为0，从第16轮开始增加，从第18轮开始在8192左右变动。但我们可以发现18轮以后的测试结果中，第27轮，30轮，43轮和59轮的结果均与其前一轮的测试结果相差1000左右，雪崩效应很不好。由此可知，SHA-256这种Hash函数的算法在进行迭代过程中，输出序列的第27轮，30轮，43轮和59轮的雪崩效应不好，不能通过雪崩测试。

图5：



4.2 小结

从4.1节中的试验分析可知， Hash函数SHA-256的算法在进行迭代过程中，随着迭代次数的增加，从整体上来看，所得到的输出序列的随机性是越来越好的。但是在随机性逐渐变好的过程中，第27轮，30轮，43轮和59轮与其对应的前一轮或者后一轮相比随机性不好，这也说明了Hash函数SHA-256的算法在某些方面存在着一些缺陷。这无疑是SHA-256的算法也存在着安全性问题隐患的一个暗示。

第五章　需要进一步解决的问题及展望

本文主要针对SHA-256，用已有统计检测方法中的χ^2检验对其进行了随机性测试以及雪崩效应的测试，并对测试结果进行了分析讨论，指出了该算法中的一些不足之处：由图1至图5，可以看出，Hash函数SHA-256的算法在进行迭代过程中，随着迭代次数的增加，从整体上来看，所得到的输出序列的随机性是越来越好的。但是在随机性逐渐变好的过程中，第27轮，30轮，43轮和59轮与其对应的前一轮或者后一轮相比随机性不好，这也说明了Hash函数SHA-256的算法在某些方面存在着一些缺陷。

需要进一步解决的问题有：

1)分析上述统计结果产生的原因和SHA-256的改进方案。

2)提出判定序列随机性的新的有效方法。例如可以提出新的判断序列随机性的统计量，给出其概率分布，与其他的统计量相比较等等。

参考文献

[1] PRENEEL B, BOSSELAERS A, RIJMEN V. Comments by the NESSIE project on the AES finalists[EB/OL]. AES Round 2 public comment, http://www.nist.gov /aes, 2000-05

[2] 朱明富， 张宝东，吕述望，分组密码算法扩散特性的一种统计分析。通信学报，2002.10:142～149

[3] 王安胜，朱明富，吕述望，Hash 算法扩散性能的偏差分析。通信保密，2003.1:94～96

[4] 孙淑玲 编著，应用密码学，清华大学出版社，2004

[5] X.Y.Wang, H.B.Yu, How to break MD5 and other Hash functions. Advances in Cryptology-Eurocrypt'05, Springer-Verlag,2005, LNCS, 3494,19～35

[6] A. J. Menezes, P.C. Van Oorschot, S. A. Vanstone 著，胡磊，王鹏 译，应用密码学手册，电子工业出版社，2005

[7] 王张宜，李波，张焕国，Hash 函数的安全性研究，计算机工程与应用,2005

[8] B Schneier. Secrets & lies: digital security in networked world. Jone Wiley & Sons, 2000: 85~101

[9] Sandhu R, Coyne, E, Feinstein H, et al. Role-based Access ControlModels[J]. IEEE Computer, 1996, 29(2):38-47

[10] 薛英花，吕述望，郭圣权，随机数发生器分析及其在安全信息系统中的应用，计算机工程，2003

[11] 陈旭,吕述望，一种随机序列的读取及其随机性测试的解决方法，计算机应用，2002

[12] 徐金福，Hash 函数 HAS_160 和 MD5 潜在威胁的分析：[硕士学位论文].山东：山东大学，2007

[13] 李刚，分组密码 ARIA 的低轮差分分析：[硕士学位论文]。西安：西安电子科技大学，2007

[14] 刘景伟，分组密码中关键问题的研究：[硕士学位论文]。西安：西安电子科技大学，2004

[15] 李信然，有关密钥流生成器的概率模型及逻辑函数的性质研究：[硕士学位论文]。西安：信息工程大学，2006

[16] 王美琴，分组密码算法 Serpent_256 的差分代数分析：[博士学位论文].山东：山东大学，2007

[17] 肖国镇，梁传甲，王育民。伪随机序列及其应用。国防工业出版社，1985
[18] 陈勤，江虹，Hash 函数的设计与分析，杭州大学学报(自然科学版),第 26 卷第 1 期,1999 年 1 月
[19] 李国明，关于密码学上的 Hash 函数研究:[硕士学位论文].四川：西南交通大学，2005
[20] 梁杰，MD5_Hash 函数的安全性分析：[硕士学位论文].上海：上海交通大学，2007
[21] 祝捷,陈剑清,刘文芬,李世取，一种基于后验概率判决的 one-by-one 快速相关攻击算法，应用数学，2004，17：99～109
[22] 刑育森，杨义先，分组密码安全性研究的新进展，通信保密，1996 年第 3 期
[23] NBS. F IPS PUB 46 Date Encryp tion standard[S]. Washington: [S. L.] , 1977.（DES）
[24] 赵剑，赵钦生，王冰冰，分组密码发展现状，长春大学学报，2006
[25] 林德敬，林柏钢，林德清，国内外分组密码理论与技术的研究现状及发展趋势，天津通信技术，2002
[26] 崔健双，李铁克， 网络信息系统安全研究现状及热点分析，计算机工程于应用，2003

硕士期间的研究成果

(1) 李波,刘平,王张宜. SHA-256 输出序列的随机性研究。计算机工程与应用,2007,43(9):142-144

致 谢

本课题在选题及研究过程中得到李波老师的悉心指导。李老师多次询问研究进程，并为我指点迷津，帮助我开拓研究思路，精心点拨、热忱鼓励。 李老师一丝不苟的作风，严谨求实的态度，踏踏实实的精神，不仅授我以文，而且教我做人，虽历时仅三载，却给以终生受益无穷之道。对李老师的感激之情是无法用言语表达的。

感谢谢民育老师、陈应保老师对我的教育培养。他们细心指导我的学习与研究，在此，我要向诸位老师深深地鞠上一躬。我还要感谢数统学院的各位老师在这三年时间里对我的帮助，使我得以顺利地完成学业。感谢帮助、支持过我的同学，谢谢你们给了我一个和谐融洽的学习氛围。

另外我还要特别感谢一个人，王张宜老师，在本课题的研究过程中，他在计算机密码学方面给了我很大的帮助！

最后，向我的父母亲和所有关心我的人致谢，感谢他们对我的理解与支持。

刘平

2008 年 5 月于武昌桂子山

概率统计在计算机密码学中的应用

作者：刘平
学位授予单位：华中师范大学