Unique File Identification in the National Software Reference Library

Steve Mead
National Institute of Standards & Technology
100 Bureau Drive, Stop 8970
Gaithersburg, MD 20899
smead@nist.gov

Abstract:

The National Software Reference Library (NSRL) provides a repository of known software, file profiles, and file signatures for use by law enforcement and other organizations involved with computer forensic investigations. The NSRL is comprised of three major elements:

1. A physical library of commercial software packages.
2. A database of information about each file within each software package.
3. A smaller database of the most widely used information that is updated and released quarterly. This database is called the NSRL Reference Data Set (RDS) and is NIST Special Database #28 [18].

During a forensic investigation, hundreds of thousands of files may be encountered. The NSRL is used to identify known files. This can reduce the amount of time spent examining a computer. Matches for common operating systems and applications do not need to be searched, either manually or electronically, for evidence. Additionally, the NSRL is used to determine which software applications are present on a system. This may suggest how the computer was being used and provide information on how and where to search for evidence.

This paper examines whether the techniques used to create file signatures in the NSRL produce unique results—a core characteristic that the NSRL depends on for the majority of its uses. The uniqueness of the file identification is analyzed via two methods: an empirical analysis of the file signatures within the NSRL and research into the recent attacks on the hash algorithms used to generate the file signatures within the NSRL.

The research addresses the following questions:

1. Are the file signatures in the NSRL unique? The NSRL was examined for distinct files that generated the same signature (i.e., a collision).

2. How likely is it that collisions will occur in the future? The probability of future collisions depends directly on the randomness of the file signatures. We ran statistical tests to answer the following questions:
    - Do file signatures appear to be random?

- Do files bias the randomness of the file signatures in any detectable way?

3. Do the recent attacks on MD5 and SHA-1 pose any specific threats to the NSRL?

The conclusions of this paper are:

- There are no file signature collisions in the NSRL for either MD5 or SHA-1.
- There was no detectable bias introduced by hashing files, and so the probability of future collisions is negligible.[1]
- Although there are methods to attack the underlying hash algorithms, they are not relevant to the NSRL.

[1] The probability of a collision between hashes in either MD5 or SHA1 is so small that it is effectively zero. Even if the size of the NSRL doubles each year in size, it would take more than 50 years before there would be enough SHA-1 file signatures to encounter a collision just by chance.

1.0 Introduction

The National Software Reference Library (NSRL) provides a repository of known software, file profiles, and file signatures for use by law enforcement and other organizations with computer forensic investigations. The NSRL is comprised of three major elements:

1. A physical library of commercial software packages.
2. A database of information about each file within each software package.
3. A smaller database of the most widely used information that is updated and released quarterly. This database is called the NSRL Reference Data Set (RDS) and is NIST Special Database #28 [18].

The NSRL project was initiated in 1999 at the request of the FBI, the DoD Cyber Crime Center and the National Institute of Justice. The project is a part of the forensics sciences program at NIST's Office of Law Enforcement Standards.[2] The first release of the NSRL RDS[3] was in 2001.

As of February 2006, the NSRL consists of over 7083 software application packages that include over 34 million files. Since many of the files are used within multiple applications, there are many duplicate files within the NSRL. Currently, there are over 10 million unique files. [4]

During a forensic investigation, hundreds of thousands of files may be encountered. The NSRL is used to identify known files. This can reduce the amount of time spent examining a computer. Matches for common operating system files or applications do not need to be searched, either manually or electronically, for evidence. For example, if the forensic examiner was searching images on a Microsoft Windows™ 2000 system, a comparison against the NSRL would identify over 4000 image files that come with the standard Windows™ installation[5]. These specific files could be excluded from further examination, as the content is known, and would not contain evidence.

Additionally, some NSRL matches are used to determine what software applications were used on a system. This may provide information for the investigator to determine how and where to search for evidence. For example, if a computer system contains applications to crack passwords, keyboard loggers, or rootkit[6] packages, this may lead to

[2] Additional information on the Office of Law Enforcement Standards (OLES) is at http://www.eeel.nist.gov/oles/index.html.
[3] The RDS is available at http://www.nsrl.nist.gov.
[4] The most recent version of the NSRL, 2.11 contains 34,994,666 files, and 10,793,831 unique SHA-1 file signatures.
[5] Certain commercial software, equipment, instruments, or materials are identified in this paper to foster understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.
[6] A rootkit commonly refers to software installed after an attacker has gained access to a system to allow continued access, and to actively hide traces of the attacker's activity.

further investigation to determine if the system was used to hack into other computer systems. This type of matching could also be used to resolve an intellectual property question if a system contained proprietary software for which the system owner had no license.

2.0 NSRL File Signatures

The NSRL uses a mathematical technique called hashing to produce file signatures. Currently, the NSRL uses three methods to create signatures: two are cryptographic hash algorithms and one is an error checking technique called a cyclic redundancy check (CRC) [3]. It is important to note that there are known limitations in using the CRC to generate file signatures, and that the CRC signatures within NSRL are not unique, given that the same signature may be associated with more than one file.

Hash algorithms work by taking an input, in this case a file, and use advanced mathematics to compress it to a fixed length string of zeros and ones. The specific cryptographic hash algorithms used to generate file signatures are one-way functions: a given input will always produce the same output and the process cannot be reversed. The output is called a hash and, in the context of the NSRL, is referred to as a file signature.

One important property exhibited by these types of hash algorithms is that their output is randomly distributed across the entire range of possible outputs.[7] This property explains why one-way hash functions are frequently used as building blocks in random number generators [17,27].

The specific hash algorithms used by the NSRL to generate the file signatures are MD5 and SHA-1. MD5 is an older hash algorithm and is defined by the Internet Engineering Task Force, Request for Comment 1321 [23]. The SHA-1 is a Federal Information Processing Standard (FIPS) promulgated by NIST as FIPS PUB 180-2 [19]. The third technique is a 32 bit version of the CRC error checking method, as defined by [8,32]. Both the MD5 and SHA-1 are cryptographic hash algorithms.

NIST is planning to add additional file signatures generated by other hash algorithms in the future, including those identified in FIPS PUB 180-2 (SHA-256, SHA-384, SHA-512) [19].

3.0 Uniqueness

An important issue is whether the methods used to create file signatures in the NSRL produce unique results. This core characteristic is the basis for the use of the NSRL within the forensic community. Since it is used for file identification, this analysis examines the uniqueness of file identification employing both an empirical analysis of the file signatures within the NSRL and an analysis of current research relating to the underlying algorithms used to generate the file signatures.

[7] For a detailed look at hash algorithms, see *Applied Cryptography* by Bruce Schneier, see Ref. [27].

Specifically, the research addresses the following three questions:

1. Do collisions occur within the NSRL?

A collision occurs if two different files generate the same hash file signature. Since the NSRL uses file signatures to identify known files and applications, a collision could cause files to be incorrectly identified. For each hash algorithm used within the NSRL, the file signatures were examined to identify any collisions.

2. How likely is it that a collision will occur in the future?

In order to address this question, it is necessary to analyze aspects randomness in relation to the hash algorithms used to generate the file signatures. Two key properties that cryptographic hash algorithms exhibit are the randomness of their output, and the large number of potential outputs that can be generated. If the file signatures conform to these properties, the standard methods for evaluating the chances for a collision are well known. We examine the overall randomness of the file signatures and whether hashing files bias the resulting signatures.

- **Are the file signatures within the NSRL random?**
  The algorithms used to generate the NSRL file signatures have been thoroughly tested throughout the cryptographic community. However, an empirical analysis of such a large set of hashes generated from files has not been publicly disseminated. If the NSRL file signatures adhere to the properties of the underlying hash algorithms, they should appear to be random.[8]

- **Do files used as input bias the randomness of hash signatures within the NSRL?**
  The NSRL file signatures are generated directly from files which range in size from a few bytes to gigabytes in size. Files and applications typically tend to have some forms of structure, repetition, and other modulated patterns, which may subtly affect the distribution of values produced by the hash algorithm.

3. Are the hash algorithms used to generate the file signatures able to be manipulated to create collisions relevant to the NSRL?

Recently, methods for finding collisions in MD5 and SHA1, two hash functions included in the NSRL, have been discovered. Currently, there are multiple examples of MD5 collisions [14,30], but as of yet no documented SHA-1 collisions generated. These specific types of attacks reduce the usefulness of MD5 and SHA-1 for some, but not all, applications [20]. These types of attacks were examined here to see if they could have any impact on the NSRL within typical forensic settings.

[8] Cryptographic hash algorithms such as MD5 and SHA-1 produce output that should be indistinguishable from a random sequence of numbers, given that the input is not known.

4.0 Approach

The three primary questions, although all related in nature, require different approaches. The first two questions, examining the NSRL for collisions, and addressing the likelihood of future collisions was accomplished through an empirical analysis of the file signatures within the NSRL. The last question, evaluating the impact of attacks on hash algorithms used within the NSRL relies on research into the specific attacks and analyzing them within the context of how the NSRL is used within the field of computer forensics.

4.1 Examining the NSRL for Collisions

A collision is defined as two different streams of input that, when hashed, produce the same output. In terms of the NSRL, a collision would occur if two different files produce the same file signature. If the content between two files is different even by a single bit, the files should generate distinct file signatures. It is important to keep in mind that only the file content is used to calculate the file signature—not file metadata such as file name.

Identifying collisions is a relatively straightforward process. The NSRL is searched for all file signatures that appear more than once. If those signatures are generated by different files, then a collision has occurred.

Many of the files within the NSRL have been included multiple times. For example, certain graphic files used in the Windows™ operating system are included in multiple releases of Windows™. Files that had the same file size and the same file signature for each CRC-32, MD5, and SHA-1 were considered duplicates and removed from further consideration[9]. When looking for collisions, it is important to differentiate between an actual collision (where the file content is different) and cases where the file signatures are the same because the files are duplicates.

After prescreening the data, we compared all of the hashes in search of any collisions. Each set of file signatures generated from a specific algorithm (CRC-32, MD5, SHA-1) was examined and any collisions recorded.

In addition to identifying collisions, we sorted the signatures and recorded information about the spacing between successive hashes. Each file signature is essentially a very large number, and the maximum, minimum, and average distances between sorted hash pairs can be measured. The distance should be related to the specific hash algorithm, as well as the total number of file signatures. If there is substantial grouping or clustering, this could affect the probabilities of a collision, and would need to be investigated further.

[9] Due to the logistics and time involved to compare each file byte for byte for duplicates, a combination of all file hashes and file size was used to identify files with identical content.

4.2 Likelihood for Future Collisions

The possibility of future collisions in the NSRL is more difficult to determine given that the NSRL grows in size. However, there are primarily two factors to consider: the total number of hashes an algorithm can generate, and how those values are distributed.

For any hash algorithm, the total number of unique values that can be generated is a function of the specific algorithm. As the total number of unique values increase for an algorithm, the chances that there will be a collision decrease. Currently, the NSRL uses three primary algorithms to generate file signatures, and each of them can generate a different maximum number of hashes. Since the quantity of hashes can vary greatly between the algorithms, in table 1 we use drops of water as an analogy to understand the relative difference between the algorithms.

File Signature Algorithm Capacity Comparison			
Algorithm	**Max # of Hashes (Base 2)**	**Max # of Hashes (Base 10)**	**Relative Size Analogy[10] (Drops of Water)**
CRC-32	2^{32}	4.29×10^{9}	All the water in a small pond
MD5	2^{128}	3.40×10^{38}	All the water in our galaxy
SHA-1	2^{160}	1.46×10^{48}	All the water in our universe

Table 1: File Signature Algorithm Capacity Comparison

The number of potential values a given algorithm can generate is only part of calculating the probability for collision. The other critical aspect to this calculation is how likely any particular hash value is to be used or generated. If all values for a given hash algorithm have an equal probability[11] of being used, the odds of a collision occurring can be calculated. The NSRL uses two cryptographic algorithms to generate file signatures, MD5 and the SHA-1. Both of these algorithms generate output that appears to be random. In fact, both of these algorithms have been used as building blocks commonly used in random number generators.

One area of research that has not been extensively studied is the impact of files as input to hash algorithms. Files can vary greatly in size, ranging from a couple of bytes to gigabytes. Additionally, files tend to have internal structures and repeated patterns, which may not be present in other forms of data. We examined if files as input to hash algorithms could bias how the algorithms operate, and change the probability of generating a collision.

[10] These values are for comparisons of relative size and assume the following: there are approximately 20 drops of water per ml, the volume of water on earth is 1.3×10^{9} km^3, and that each star in our galaxy contains an earth-like volume of water. Additionally, that there are approximately 4×10^{11} stars in our galaxy, and 8×10^{10} total galaxies in our universe.

[11] This would be described as uniformly random—where all values have an equal chance of being generated.

In relation to future collisions within the NSRL, if files do not bias the generation of file signatures, then the probabilities of a collision are known. However, if we are able to detect bias in file signatures, then the probabilities for a collision may not fit the current models and would need to be reexamined.

In order to identify bias in the NSRL file signatures, our approach focused on two general areas: what tests would need to be performed to identify significant bias and how to organize and structure the NSRL data for analysis.

Randomness is a property that is straightforward to conceptualize, but difficult to define exactly and empirically measure. There is no single statistical or deterministic test that can conclusively identify if a given sequence of numbers is random. Furthermore, there are different aspects to randomness that cannot be captured by one type of test. To have the greatest amount of confidence in the results from any randomness testing, multiple tests need to be performed to detect various types of deviation.

There are two well known sets of tests, the Diehard tests by Marsaglia [15] and those defined by Knuth in his *Seminumerical Algorithms* [13]. Each identify a variety of tests that cover a range of random behavior from the simple distribution of 0's and 1's (binary frequency tests) to more complex methods that look at various interactions between bits in a sequence.

NIST developed a set of statistical tests specifically to address the need to evaluate random number generators for cryptographic applications. They include variations of those by Marsaglia [15] and Knuth [13]. However, new tests were introduced such as the Approximate Entropy Test [22], Berlekamp-Massey Linear Complexity Test [16], and the Random Excursion Test [24]. This suite was released as the NIST Statistical Test Suite (STS) and the latest release is version 1.8 [25].

The STS provides a framework to run multiple types of tests specifically designed to evaluate binary random number generators. The specific tests used, as well as the defects they are designed to detect are listed in Table 2 [25,29]. Detailed explanations on each test, examples, and configuration guidelines, may be found on the STS website at http://csrc.nist.gov/rng.

#	Statistical Test	Defect Detected
1	Frequency (Mono-bit) Test	Too many zeros or ones.
2	Block Frequency Test	Too many zeros or ones in discrete m-sized blocks.
3	Cumulative Sums Test	Too many zeroes or ones at the beginning of a sequence.
4	Runs Test	Large (small) total number of runs indicates that the oscillation in the bit stream is too fast (too slow).
5	Longest Run of Ones in a Block Test	Deviation of the distribution of long runs of ones.
6	Random Binary Matrix Rank Test	Deviation of the rank distribution from a corresponding random sequence, due to periodicity.
7	Discrete Fourier Transform (Spectral) Test	Periodic features in the bit stream.
8	Overlapping (Periodic) Templates Matching Test	Too many occurrences of m-bit runs of ones.
9	Maurer's Universal Statistic Test	Data can be compressed more than expected for a random sequence.
10	Approximate Entropy (Apen) Test	Non uniform distribution of m-length words.
11	Serial Test	Non-uniform distribution of m-length words.
12	Linear Complexity Test	Deviation from the distribution of the linear complexity for finite length (sub) strings.
13	Random Excursions Test	Deviation from the distribution of the number of visits of a random walk to a certain state.

Table 2: Statistical Tests and Defects Detected

An additional set of tests was designed to present the file signatures visually. We wanted to assess the potential to detect non-random behavior in the file signatures through visual analysis of the data.

The second aspect to our analysis was how to organize and structure the data for analysis, as there are multiple versions of the NSRL, and the data is not simply a stream of bits created from a random number generator.

1. Although there are multiple versions of the NSRL, each version includes all of the file signatures and profile information from the previous versions, as well as any additional signatures added for the must current version. We chose the latest version of the NSRL, 2.9, for analysis.

2. To create a stream of bits for analysis, we extracted the file signatures for each algorithm from the NSRL and concatenated them.[12]

3. The majority of the tests examined the whole stream of data. However, some tests looked at blocks of data. We configured the block sizes in those tests to align with the specific size of the file signature being examined. For example, some of the SHA-1 tests analyzed each discrete file signature—since the SHA-1 file signatures are 160 bits, any block testing explicitly analyzed blocks of 160 bits. Additionally, we were careful to ensure that any manipulations of the bit streams were aligned to multiples of the block size.

[12] If the file signatures are extracted in sort order, the distribution of zeros and ones in the concatenated string would be biased.

4.3 The Impact of the Recent Attacks on MD5 and SHA-1

The last part of our analysis examines the current research in the area of hash algorithms, especially the recent attacks on the MD5 and the SHA-1, and their impact on the NSRL.

We examined the types of attacks and their impact on file signatures. We paid particular attention to the specific context in which the NSRL is used and its role within computer forensics.

5.0 Results: Collisions in the NSRL

Examining the NSRL for collisions was based on two separate tests: identifying collisions between different files, and looking to see how close file signatures were to each another.

5.1 Collisions Detection

No collisions were found within the MD5 or SHA-1 file signatures. However, a large number of collisions were detected within the set of CRC-32 file signatures.[13]

Collisions in the NSRL, Version 2.9			
Total Files Signatures Examined: 10,533,771			
	SHA-1	MD5	CRC-32
Collisions Detected	0	0	15,364

Table 3: Collisions with NSRL 2.9 CRC-32, MD5, and SHA-1 File Signatures

5.2 Results: Distribution of Hash Values

Each set of file signatures was examined for the smallest, largest, and average distances between signatures. This was accomplished by sorting the CRC-32, MD5, and SHA-1 signatures, and measuring the distance between each successive pair. For each algorithm, the minimum and maximum separation distance between pairs of signatures was recorded. Additionally, all of the distance measurements were used to calculate an average separation distance.

For comparison, the expected average distance between hash values was calculated. For example, with the SHA-1, if you have 10.5 million signatures, the expected average spacing (given random distribution) should be around $(2^{160})/(10.5 \text{ million})$, or approximately $1.3 \text{ X } 10^{41}$.

NSRL 2.9, Distance between File Signatures (10,533,771)			
	SHA-1	MD5	CRC-32
Average (Expected)	$1.387443 \text{ X } 10^{41}$	$3.230395 \text{ X } 10^{31}$	$4.077331 \text{ X } 10^{2}$
Minimum (Observed)	$7.088510 \text{ X } 10^{33}$	$4.218731 \text{ X } 10^{23}$	0
Maximum (Observed)	$2.313508 \text{ X } 10^{42}$	$5.398685 \text{ X } 10^{32}$	$6.47000 \text{ X } 10^{2}$
Average (Observed)	$1.387444 \text{ X } 10^{41}$	$3.230393 \text{ X } 10^{31}$	$4.07000 \text{ X } 10^{2}$

Table 4: NSRL 2.9, Distance Between Hash Values

[13] On further analysis, many of the CRC-32 collisions were due to files that had the same hash value of "FFFFFFFF." This was identified as a technique that can exploit how the CRC-32 algorithm functions so that files can verify their content without needing to store any CRC values.

5.3 Analysis

The distances between the hashes did not identify any specific problems. The average spacing reflected the approximate size of the hash space for each algorithm, as well as the number of file signatures distributed within that space.

In continuing with our previous analogy of comparing hashes to drops of water, the average distance between hashes is approximately the following:

Algorithm	Average Distance Between Hash Signatures	Relative Density Analogy (Drops of Water)
CRC-32	4.070000×10^{2}	Two drops separated by 2 tablespoons of water.
MD5	3.230393×10^{31}	Two drops of water separated by all the water contained in 1 million Earth-like planets.
SHA-1	1.387444×10^{41}	Two drops of water separated by all the water contained in 10,000 galaxies.

Table 5: Distance Between File Signatures Analogy

6.0 Future Collisions—File Bias on Signature Generation

As covered earlier, estimating the probability for collisions in a given hash algorithm depends on a fundamental assumption—that the output is random. By using a variety of statistical tests, we can identify if the NSRL file signatures uphold this assumption, or if they are biased (i.e., not randomly distributed).

6.1 Statistical Test Suite Overview

The STS 1.8 was designed to run a battery of statistical tests against a stream of bits to identify the degree of randomness exhibited [25]. It was specifically developed to test the output from cryptographic random number generators. The STS uses formal statistical hypothesis testing as a framework for analyzing sequences for randomness.

6.1.1 Statistical Hypothesis Testing

The file signatures within the NSRL must appear to be random. This attribute can be characterized probabilistically. This can also allow us to identify the degree of adherence we expect to see, and identify if the result falls outside our expectations. Specifically, we can examine the file signatures looking for deviations from the expected randomness.

To identify bias with file signatures we used statistical hypothesis testing [1,21]. In its simplest form, statistical hypothesis testing, in this context, identifies sequences that reject the null hypothesis. In this case, the null hypothesis is defined as 'the sequence is random,' and any result that appears to be non-random is rejected. This is accomplished by pre-selecting a significance level (the Greek symbol alpha, α) and comparing the results of a statistical test (P-value) to the significance level (α). If the P-value is less than α, then the null hypothesis (H_0) is rejected—evidence that the sequence is not random.

6.1.2 Significance Level, α

The significance level determines the strength of a given result in rejecting the null hypothesis. If the α is set at .01, then the likelihood of rejecting the null hypothesis in error is 1%. Stated in a different way, if the null hypothesis is rejected at the .01 level, then there is a 1 in 100 chance that the result could be in error. Although common α values are .05 and .01, due to the critical nature of the NSRL we chose a more restrictive value of .001. This gives the likelihood or rejecting a given sequence in error due to chance at 1 in 1000.

6.1.3 Probability Value (P-value)

The P-value is a means to describe the results of a given test in terms of probability. It is the likelihood that a test result would be more extreme than the observed result. For example, if the P-value from a given test is .01, there is only a 1% chance of observing a

result more extreme. In terms of the null hypothesis, the smaller the value, the stronger the evidence is in the data rejecting the null hypothesis.

The file signatures within the NSRL are generated from hash algorithms and should be analogous to a series of randomly generated large binary numbers. The results from any test should show results that support random behavior. However, as the deviation between the results and the expected behavior become more extreme, the likelihood that it occurred through chance becomes less likely.

6.1.4 STS Tests

The following tests from the STS 1.8 were used [25]:

Test Name	Test Description and Focus
Frequency (Mono-bit) Test	The proportion of zeroes and ones for the entire sequence.
Frequency within a Block Test	The proportion of zeroes and ones within M-bit blocks.
Cumulative Sums Test	The maximal excursion (from zero) of the random walk defined by the cumulative sum of adjusted (-1, +1) digits in the sequence.
Runs Test	The total number of zero and one runs in the entire sequence, where a run is an uninterrupted sequence of identical bits. A run of length k means that a run consists of exactly k identical bits and is bounded before and after with a bit of the opposite value.
Longest Run of Ones in a Block Test	The longest run of ones within M-bit blocks.
Random Binary Matrix Rank Test	The rank of disjoint sub-matrices of the entire sequence.
Discrete Fourier Transform (Spectral) Test	The peak heights in the discrete Fast Fourier Transform.
Overlapping (Periodic) Template Matching Test	The number of pre-defined target substrings.
Maurer's Universal Statistical Test	The number of bits between matching patterns.
Approximate Entropy (Apen) Test	The frequency of each and every overlapping m-bit pattern.
Serial Test	The frequency of each and every overlapping m-bit pattern across the entire sequence.
Linear Complexity Test	The length of a generating feedback register.
Random Excursions Test	The number of cycles having exactly K visits in a cumulative sum random walk. The cumulative sum random walk is found if partial sums of the (0,1) sequence are adjusted to (-1, +1). A random excursion of a random walk consists of a sequence of n steps of unit length taken at random that begin at and return to the origin.

Table 6: STS Tests Performed against NSRL File Signatures

6.1.5 Methodology (STS)

In order to use the STS to analyze the NSRL file signatures, the following process was followed.

First, the file signatures for a given algorithm (SHA-1, MD5, CRC-32) were extracted from the NSRL. This data was not sorted, and was converted from the native hexadecimal format of the NSRL file signatures into a binary representation of the data.

All binary file signatures were concatenated together creating a stream of file signatures. For NSRL 2.9, this represented 10,533,771 file signatures and created stream lengths of approximately 330 MB for CRC-32, 1.3GB for MD5 signatures, and 1.6MB for SHA-1 file signatures.

Second, the STS software was configured to run specific tests, with all relevant parameters selected. For the block frequency test, the size of the blocks aligned with the size of the file signature being examined. The CRC-32 data was tested using a block size of 32, the MD5 had a block size of 128, and the SHA-1 of 160. Additionally, if there was any other data stream manipulation (like subdividing the bit stream), it was configured to align with the appropriate algorithm block size. The number of bit streams generated was 1000 and the significance level was set at .001. The length of the bit stream varied depending on the algorithm tested.[14]

Third, the tests were run on the appropriate NSRL bit streams, and the results were captured and analyzed. Summary results are included within this document. Detailed results are posted at the NSRL website, including test configuration, and specific parameters for each test.[15]

[14] Each SHA-1 bit stream contained 1,685,280 bits, and each MD5 bit stream contained 1,348,224 bits.
[15] http://www.nsrl.nist.gov/STS/NSRL2_9.

6.2 Results Summary

The statistical tests indicated that the two cryptographic algorithms (MD5, SHA-1) performed as expected and did not identify any specific bias due to hashed files. The CRC-32 algorithm exhibited non-random behavior.

Statistical Test[16]	SHA-1	MD5	CRC-32
Frequency (Mono-bit) Test	Pass	Pass	Pass
Frequency with a Block Test	Pass	Pass	**Fail**
Cumulative-Sums Tests {1,2}	Pass	Pass	Pass
Runs Test	Pass	Pass	Pass
Longest Run of Ones in a Block Test	Pass	Pass	**Fail**
Random Binary Matrix Rank Test	Pass	Pass	Pass
Discrete Fourier Transform (Spectral) Test	Pass	Pass	Pass
Overlapping (Periodic) Template Matching Test	Pass	Pass	Pass
Maurer's Universal Statistical Test	Pass	Pass	Pass
Approximate Entropy (Apen) Test	Pass	Pass	**Fail**
Serial Test {1,2}	Pass	Pass	**Fail**
Linear Complexity Test	Pass	Pass	Pass
Random Excursions Tests {1-8}	Pass	Pass	Pass

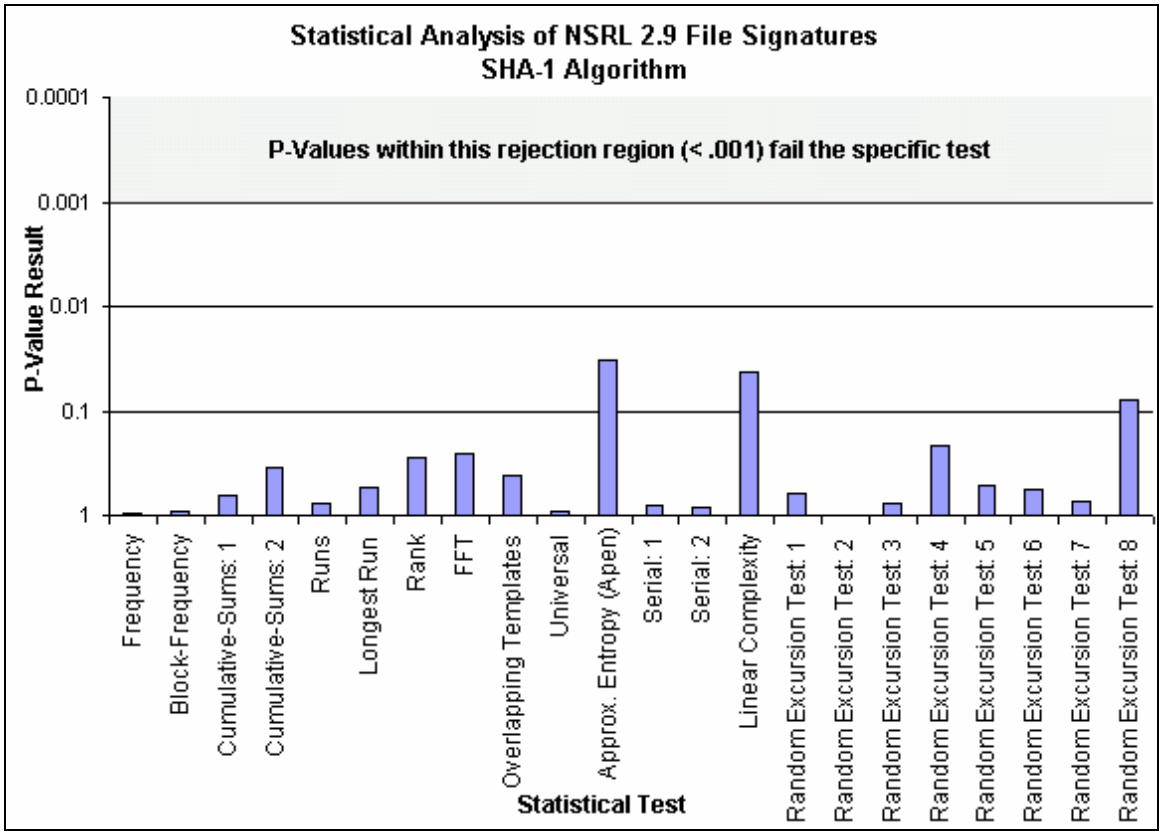
Table 7: Summary of Statistical Tests

The summary of the tests ran for each set of file signatures (SHA-1, MD5, and CRC-32) follows. To provide a better comparison between the resulting P-values, we used a log scale and identified the region where the null hypothesis is rejected (i.e., fails a specific test).

[16] Tests that have multiple versions or variants will have the test number beside the test (e.g., Serial {1,2}). If any of the tests fail, it is marked as "Fail" in the results summary table. In the following section, each test and variant will be shown in the graphs.

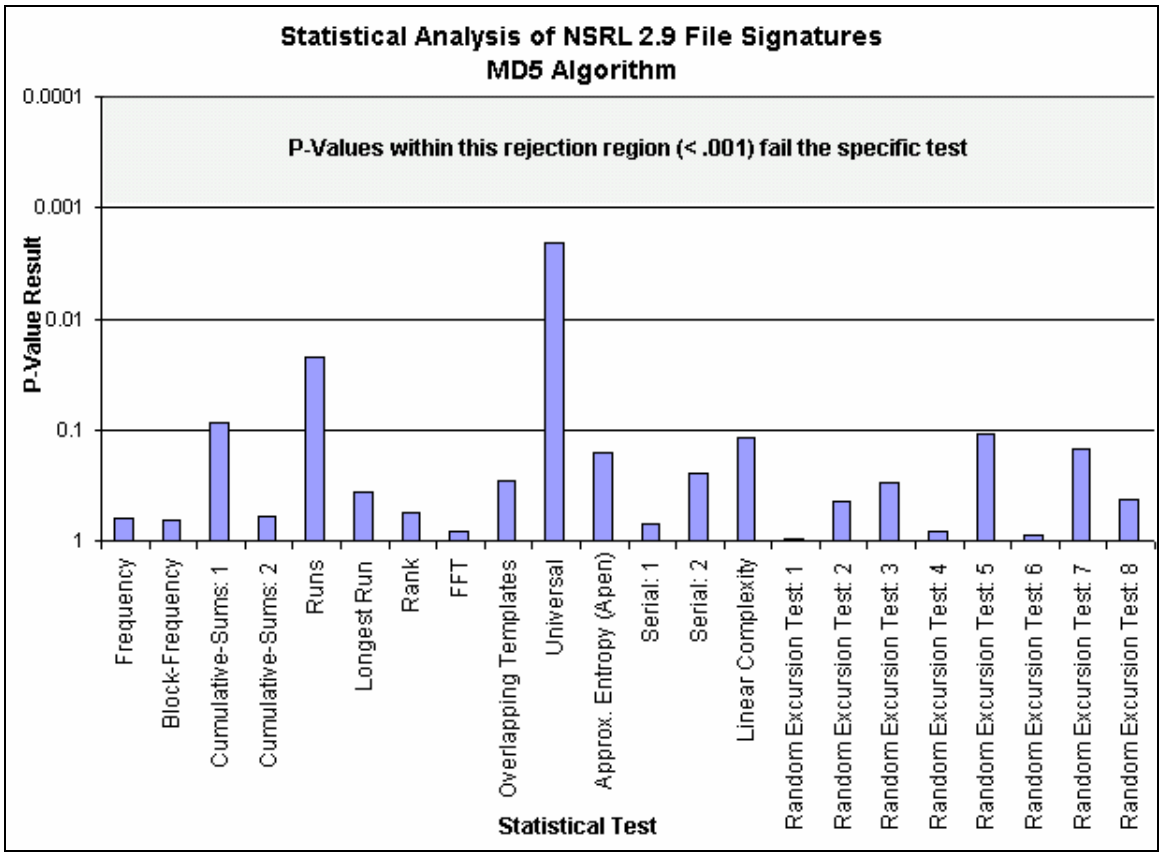
6.2.1 Results: SHA-1 File Signatures

- None of the SHA-1 tests failed with a significance level of .001.



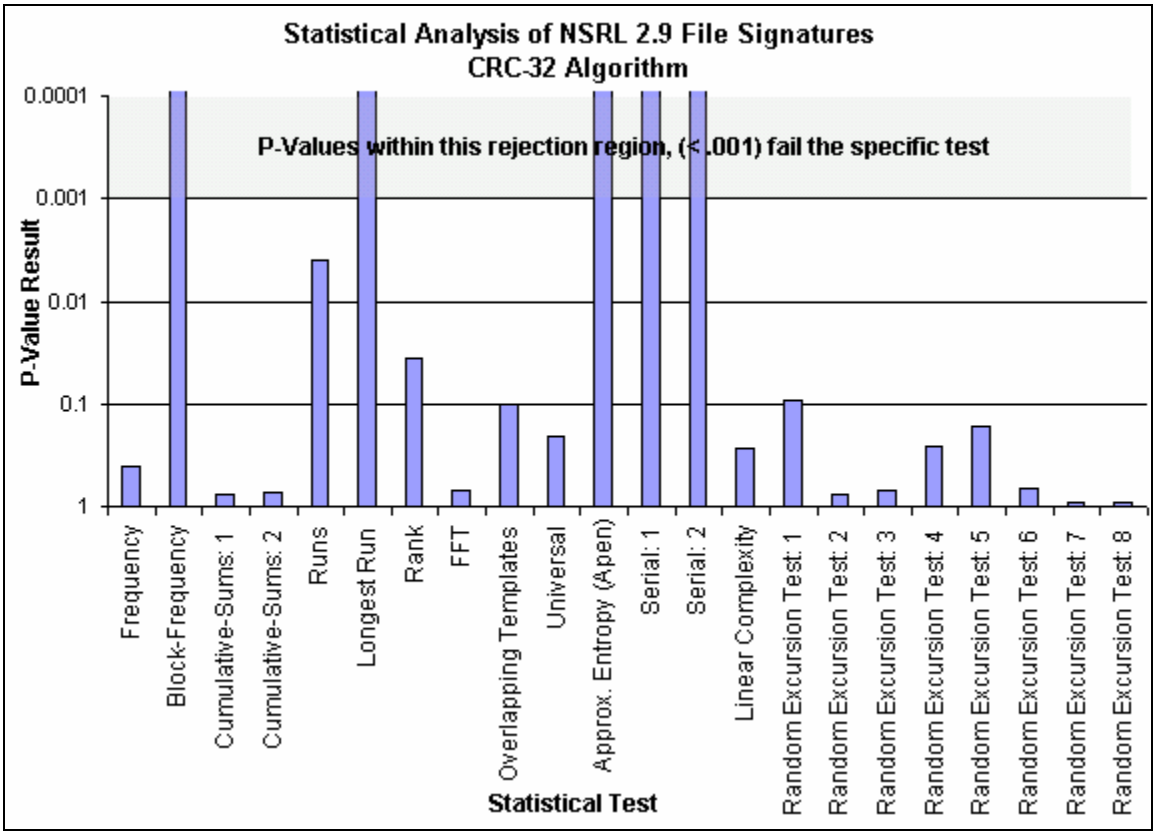
6.2.2 Results: MD5 File Signatures

- None of the MD5 tests failed with a significance level of .001.



6.2.3 Results: CRC-32 File Signatures

- Five of the CRC-32 test failed with a significance level of .001.



7.0 Results: Future Collisions—Detecting File Bias through Data Visualization

Some researchers and analysts have a well developed ability to detect patterns in data that is presented visually. Each set of file signatures was transformed into a condensed 3-D representation and then graphed for analysis. It is important to note that this is not a mathematically rigorous approach. However, graphical representation should be considered as a method for data exploration, looking for explicit patterns, groupings, or clear skewing of the data.

7.1 Methodology

We processed each set of file signatures using the following four steps.
First, the individual hashes were extracted from the NSRL. In contrast to the STS tests, the order of the data was not relevant, and so no data transformation was needed.

Second, each file signature was broken into three parts (at the binary level), an X, Y, and Z component. The size of each component depended on the algorithm used. For example, with the CRC-32 algorithm, the first 10 bits were considered the X component, while the X component for the SHA-1 algorithm consisted of the first 52 bits.

Algorithm	X (Bits)	Y (Bits)	Z (Bits)
SHA-1	0-52	53-106	107-160
MD5	0-42	43-85	86-128
CRC-32	0-10	11-21	22-31

Table 8: SHA-1, MD5, CRC-32, X, Y, Z Components

Third, the Z component was averaged so that the density of the data could be analyzed. If the file signatures were graphed directly (x,y,z), the result is simply a mass of 10 million points—very difficult to draw any conclusions from. The averaging was done by gridding the data and averaging all Z values within that grid.

Fourth, each file signature was then graphed. The X and Y components were used directly, and the Z average was plotted. The density of the data is shaded to highlight differences between areas of data. Additionally, contour lines were projected onto the floor of the graph to further assist in identifying data patterns.

Essentially, we ended up with a graph that maps out each algorithm, displaying where values cluster, and where there are similar patterns in density.

Key points to look for in the graphs.

- Specific groupings of data,
- Consistent and systematic data bias, and
- Any clearly distinguishable patterns.

7.2 Analysis Summary

The graphical display of the NSRL data demonstrated some correlation with the other analyses techniques.

The MD5 and SHA-1 algorithms do not present any clear or systematic patterns that can be identified using this method.

The CRC-32 data also displays a relatively even distribution across the majority of the data. However, there are some distinctive peaks at the corners of the graph. Because this pattern is so clear and pronounced, visually it suggests there is some systematic bias in the CRC-32 file hashes.

7.2.1 Distribution: Visual Analysis of SHA-1

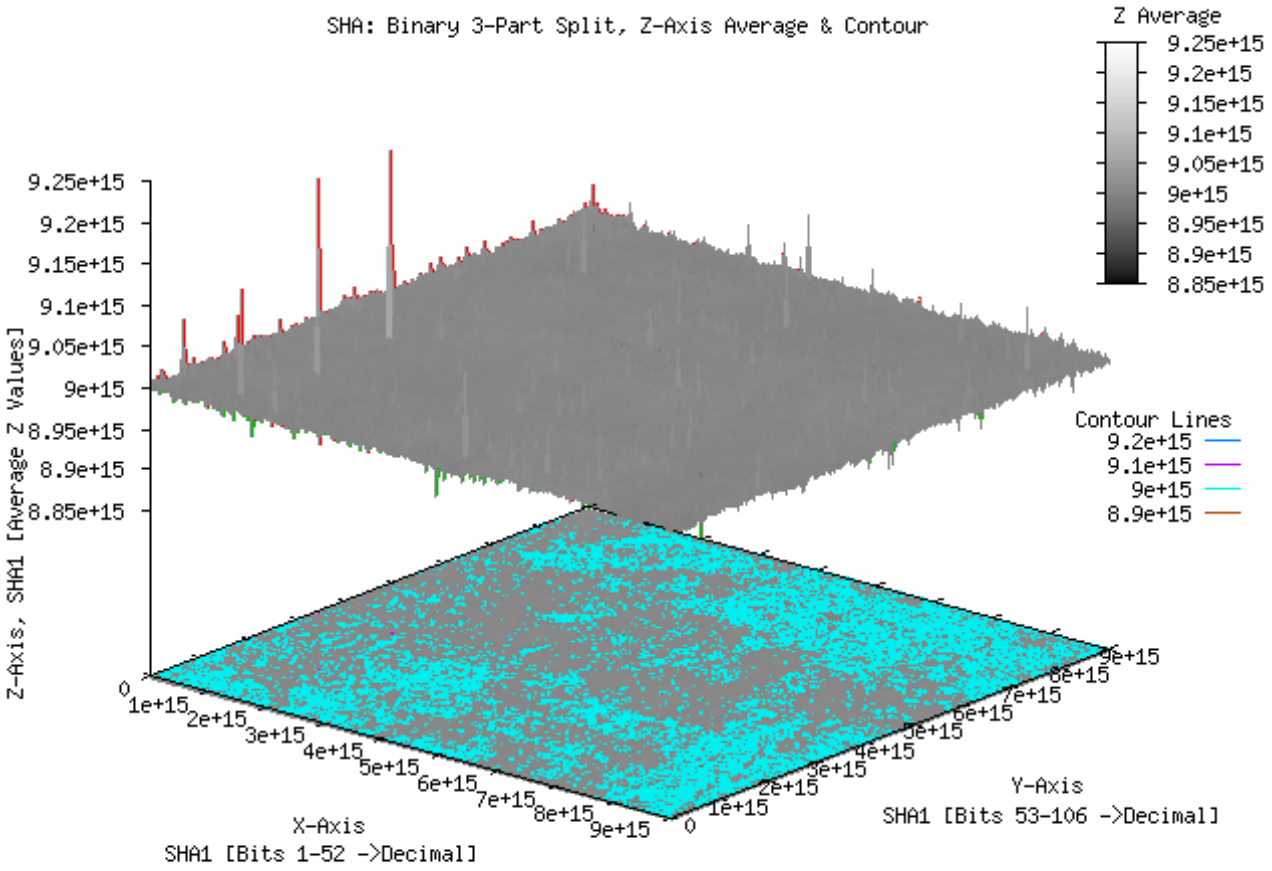


Figure 1: SHA-1 Binary Z-Axis Average & Contour

7.2.2 Distribution: Visual Analysis of MD5

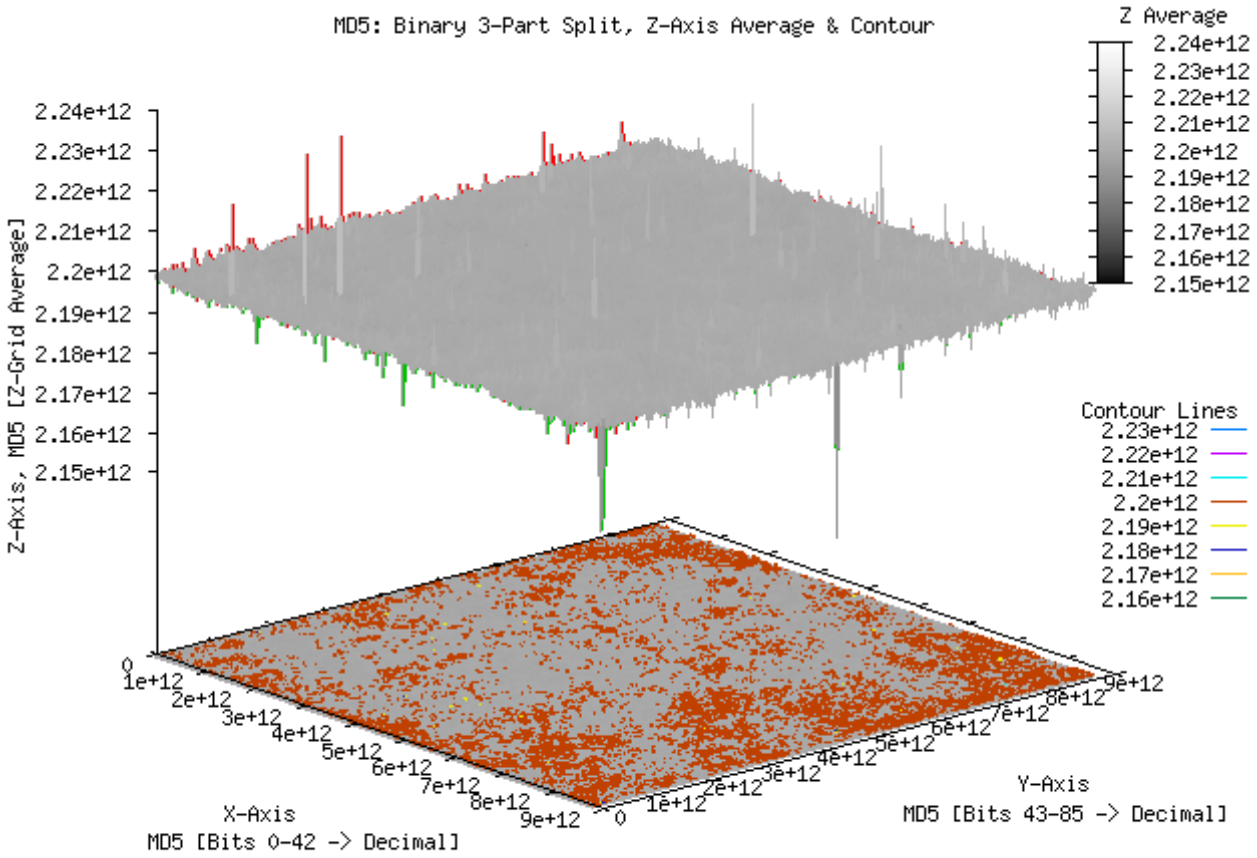


Figure 2: MD5 Binary Z-Axis & Contour

7.2.3 Distribution: Visual Analysis of CRC-32

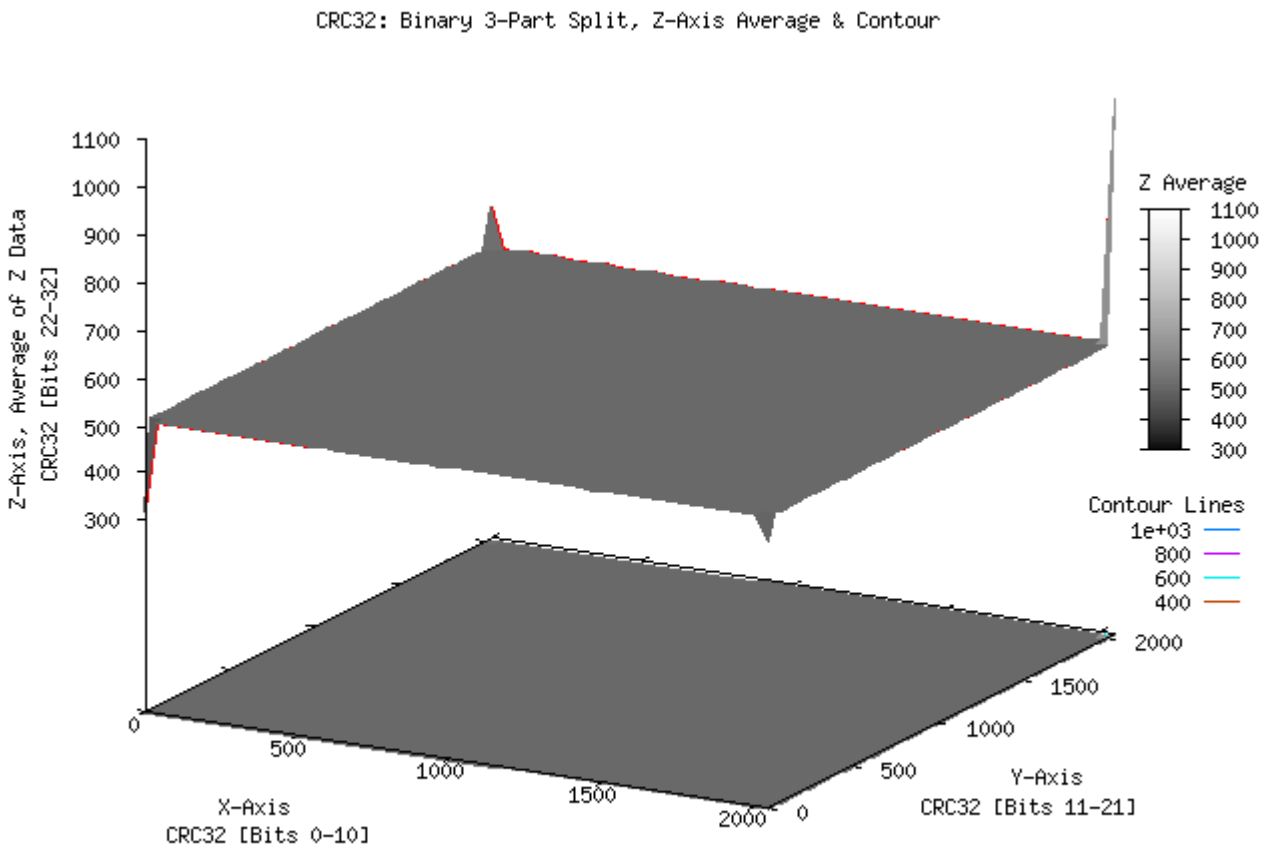


Figure 3: CRC-32 Binary Z-Axis Average & Contour

8.0 Implications of MD5 and SHA-1 Attacks on Forensic Use of Hashes for File Identification

It has been reported in the scientific press that several of the commonly used one-way hashing functions have been broken. Research presented by Wang [31], Biham and Chen [2], Joux [9] at Crypto 2004 prompted well known cryptologist, Bruce Schneier, to publish an opinion piece in *Computer World* titled, "Cryptanalysis of MD5 and SHA: Time for a New Standard." [26] More recently, in early 2005, Schneier stated on his web log, "SHA-1 has been broken." [28] The MD5 and the SHA-1 are the primary algorithms used by the NSRL. While this research is significant, it does not affect the use of hashes within the NSRL. To clarify our position, the research must be examined in the context of how the NSRL is used.

The NSRL uses hashes to identify known files. However, hashes are used for a variety of other purposes as well.

- Digital signatures,
- Random number generation,
- Cryptographic key update and derivation, and
- Integrity protection.

Two fundamental properties that cryptographic hash algorithms rely on are collision resistance and being a one-way function (i.e., being pre-image resistant). The first, being collision resistant, means that two different messages should not hash to the same value. Normally, the efficacy of a hash is measured by the number of possible combinations that need to be tried before a collision occurs (i.e., brute force collision generation).

The second property that good hash algorithms have is that they are pre-image resistant, i.e., it is computationally infeasible for a message to be constructed that matches a given hash. Additionally, this is extended even further by requiring that the algorithm should be second pre-image resistant, i.e., infeasible to construct a second file that generates the identical hash signature as the original file.

MD5, the older of the two hash functions provided by the NSRL, is susceptible to collision attacks. MD5 was "almost broken" in 1995 [5] and a full collision attack was found in 2004 [30]. By 1996, computer security experts were warning the community not to use MD5 for applications that required collision resistance [6]. Furthermore, work by Vlastimil Klima [12] has shown that the MD5 vulnerabilities can be exploited by creating collisions within hours and specialized attacks to create collisions in minutes [11]. Daum and Lucks demonstrate how to use collisions to create two documents with the same MD5 fingerprint that appear to have completely different content[17] [4].

[17] They use the MD5 collision technique to generate two blocks of data that collide (i.e., the blocks are different, but don't affect the MD5 calculation). They then create a multi-part document that is sensitive to the collisions and use it to display different parts of the document. If colliding block #1 is present, then the first part of the document is displayed; if colliding block #2 is present, the second part of the document is shown.

As of April 2005, there are no known collisions in the SHA-1 algorithm. However, researchers have identified methods to attack the SHA-1 with far less effort than was originally expected [10,30]. Simpler versions of the SHA have been successfully attacked [2,9].

These attacks and demonstrated collision weaknesses of hash algorithms are significant for cryptographic applications. However, on closer examination of the specific attacks, they do not affect the use of the NSRL within forensic applications.

Currently, there are no known pre-image or second pre-image attacks on the MD5 or SHA-1.

However, since this research does represent potential attacks, it is important to put them in the context of how the NSRL is used and where collision attacks might be relevant. Essentially, the NSRL is used in two ways. First, and most common, the NSRL is used to filter out known files. This allows an investigator to examine a hard drive and remove a large number of benign files that belong to the OS, applications, and common components. Second, the NSRL can be used to identify if applications relevant to an investigation are installed on a given hard drive.

In the first case, known files are filtered from an examination. If an adversary were able to change contraband materials to match hashes within the NSRL, such materials could incorrectly be excluded from an examination. However, this would mean that for each file to be modified for this purpose, it would have to be changed in such a way to match a given hash signature within the NSRL. This would constitute a second pre-image attack (as a file would need to be modified to create a specific signature). Second pre-image attacks have not yet been shown to be feasible in MD5 or SHA-1.

The second common usage for the NSRL is to identify specific files, applications, or software that are relevant to an investigation. A common example of this would be in piracy cases, where investigators may be looking for installed versions of unlicensed software. For this use, if there are no hash signatures within the NSRL that match a particular file, the files are not identified. To defeat the NSRL in this case does not require any specialized mathematics or techniques—simply add a single byte to each file[7].

Overall, collision attacks will not affect file identification using the NSRL. In order to use a collision for misidentification, an attacker must get the hash of the "malicious" file into the NSRL. This would need to happen by either supplying the "malicious" file to the NSRL to be hashed or to get the hash directly inserted into the NSRL. The NSRL does not accept donated hashes. All of the hashes produced at NIST are from software traceable to the software manufacturer or distributors. The NSRL is itself hashed and the correct hashes are posted on the NIST website. This enables anyone to verify that their version of the NSRL is the correct version – even if this version was made by

downloading a website or copied from another sources. (The NSRL can be freely copied and re-distributed.)

Although attacks on the various algorithms used with the NSRL are significant for cryptographic applications, in reality they have little impact on how the NSRL is used within the forensics community. If there is a point in time where a given algorithm is deemed unacceptable within the NSRL context, multiple hash algorithms are already included within the NSRL, and NIST has processes in place to easily add additional algorithms as they are needed.

9.0 Conclusions

The NSRL was created to assist law enforcement in filtering the vast number of files they have to process during a case. It helps them with identifying and screening files that are known, such as applications that are part of a given operating system, files related to a given application, or files that are commonly used in a variety of hacking tools.

Using cryptographic hash algorithms such as the MD5 and the SHA-1, the hash signatures are shown to be statistically random. There were no collisions identified within the NSRL for those algorithms, and no significant bias in the hash signatures in either data randomness or data distribution.

The specific results from our study were as follows:

Collision Tests

- For NSRL version 2.9, the MD5 and SHA-1 file signatures did not have any collisions and the average distance between the file signatures matched the expected results.

- For NSRL version 2.9, the CRC-32 file signatures had multiple collisions.[18]

Randomness Tests

- For the NSRL version 2.9, the MD5 and SHA-1 hash signatures did not show any deviation from randomness in the any of the 13 statistical tests performed on the data.

- For the NSRL version 2.9, the CRC-32 file signatures failed in several of the tests, indicating that its output is not always random.

Data Visualization

- The visual analysis of the MD5 and SHA-1 file signatures did not identify any systematic patterns or bias of the data.

- The visual analysis of the CRC-32 file signatures suggested systematic bias in the signatures that correlated with the results of other analysis techniques.

Implications of Recent MD5 and SHA-1 Attacks on the NSRL

Lastly, our research on the recent attacks on the MD5 and SHA-1 algorithms suggests that these current collision attacks do not affect the NSRL for two reasons.

[18] It is important to keep in mind that the CRC-32 is not a cryptographic hash function, and collisions were expected for this algorithm.

- First, none of the attacks are pre-image attacks. In a pre-image attack, an attacker must manufacture a file with a previously known hash. All of the current attacks are only able to produce two files with the same hash, but it is a random hash.

- Second, the NSRL provides traceability, as it comes from a known and trusted source. All of the hashes within the NSRL have been generated by us, directly from vendor's media—we do not accept file signatures or hashes produced by third-parties. An attacker has no opportunity to insert a modified file signature within the NSRL.

Overall, the MD5 and SHA-1 algorithms used to generate the hash signatures clearly support their use within the forensic community for the identification of known content. While the work of Wang and others reduces the usefulness of the MD5 and SHA-1 for cryptographic applications such as digital signatures, at the current time there is no significant impact on forensic applications of file identification for inclusion or exclusion from an investigation.

10.0 Acknowledgments

I am grateful for suggestions, comments, assistance, and contributions from the following: J. Lyle, D. White, B. Livelsberger, K. Rider, S. Ghosh, S. Nottham, J. Sheng, W. Ashley, B. Guttman, J. Soto, L. Gallagher, S. Leigh, and J. Kelsey.

11.0 References

[1] I. Bass, Hypotheses Testing, URL:http://www.sixsigmafirst.com/hyptest.htm , (2003) [Last Accessed: August 2005].

[2] E. Biham and R. Chen, Near-Collisions of SHA, Advances in Cryptology--Crypto'04 Vol.3152, Springer-Verlag (2004)

[3] T. Boland and G. Fisher, Selection of Hashing Algorithms, URL:http://www.nsrl.nist.gov/Technical_papers.htm , (June 2000) [Last Accessed: January 2005].

[4] M. Daum and S. Lucks, Attacking Hash Functions by Poisoned Messages: The Story of Alice and Her Boss, URL:http://www.cits.rub.de/MD5Collisions/ , (November 2005) [Last Accessed: August 2005].

[5] H. Dobbertin, Cryptanalysis of MD5 compress, URL:http://citeseer.ist.psu.edu/dobbertin96cryptanalysis.html , (1996) [Last Accessed: May 2005].

[6] H. Dobbertin, The Status of MD5 After Recent Attack, CryptoBytes, RSA Laboratories Vol.2 (2), (1996).

[7] J. Foster and V. Liu, Catch Me If You Can, Black Hat Briefings, 2005, URL:http://www.metasploit.com/projects/antiforensics/36 , (2005) [Last Accessed: January 2006].

[8] International Standards Organization, Data Communications: High-level Data Link Control Procedures--Frame Structure (ISO 3309), IS 3009 (3rd Edition), (10-1-1984).

[9] A. Joux, Multicollisions in Iterated Hash Functions: Application to Cascaded Constructions, Advances in Cryptology--Crypto 2004 Vol.3152, Springer (2004) 306-316.

[10] J. Kelsey and B. Schneier, Second Preimages on n-bit Hash Functions for Much Less than 2n Work, URL:http://eprint.iacr.org/2004/304 , (November 2004) [Last Accessed: December 2005].

[11] V. Klima, Tunnels in Hash Functions: MD5 Collisions Within a Minute, URL:http://eprint.iacr.org , (January 2006) [Last Accessed: March 2006].

[12] V. Klima, Finding MD5 Collisions--A Toy for a Notebook, Cryptology ePrint Archive: Report 2005/075, URL:http://eprint.iacr.org/2005/075 , (March 2005) [Last Accessed: September 2005].

[13] D. Knuth, The Art of Computer Programming, Seminumerical Algorithms, Addison Wesley, Reading MA, (1993).

[14] V. Liu and P. Stack, MD5 Collision Generation Program, C Code, URL:http://www.stachliu.com/md5coll.c , (2005) [Last Accessed: January 2006].

[15] G. Marsaglia, The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness, URL:http://www.stat.fsu.edu/pub/diehard , (July 2005) [Last Accessed: November 2005].

[16] A. Menezes, P. Oorschot and others, Handbook of Applied Cryptography, CRC Press, Boca Raton, FL. (1997).

[17] National Institute of Standards and Technology, Digital Signature Standard (DSS), FIPS 186-2, URL:http://csrc.nist.gov/publications/ fips/fips186-2/fips186-2-change1.pdf , (January 2000) [Last Accessed: November 2005].

[18] National Institute of Standards and Technology, NIST Special Database #28, National Software Reference Library, URL:http://www.nist.gov/srd/nistsd28.htm , (February 2000) [Last Accessed: February 2005].

[19] National Institute of Standards and Technology, Secure Hash Standard, FIPS PUB 180-2, URL:http://www.itl.nist.gov/fipspubs/fip180-2.htm , (August 2002) [Last Accessed: November 2005].

[20] National Institute of Standards and Technology, NIST Brief Comments on Recent Cryptanalytic Attacks on SHA-1, URL:http://csrc.nist.gov/news-highlights/NIST-Brief-Comments-on-SHA1-attack.pdf , (2005) [Last Accessed: February 2006].

[21] National Institute of Standards and Technology, NIST/SEMATECH e-Handbook of Statistical Methods, URL:http://www.itl.nist.gov/div898/handbook , (September 2005) [Last Accessed: December 2005].

[22] S. Pincus and B. Singer, Randomness and Degrees of Irregularity, Proceedings of the National Academy of Science Vol.93 (5), (3-5-1996).

[23] R. Rivest, RFC 1321: The MD5 Message-Digest Algorithm, URL:http://ietf.org/rfc/rfc1321.txt , (April 1992) [Last Accessed: August 2005].

[24] A. Rukhin, Approximate Entropy for Testing Randomness, Journal of Applied Probability Vol.37 (2000).

[25] A. Rukhin, J. Soto and others, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, NIST Special Publication 800-22, Revised May 2001, (2001).

[26] Schneier, Bruce, Cryptanalysis of MD5 and SHA: Time for a New Standard, Computer World, (August, 2004)

[27] B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, John Wiley and Sons, New York (1996).

[28] B. Schneier, Schneier on Security: SHA-1 Broken, URL:http://www.schneier.com/blog/archives/2005/02/sha1_broken.html , (February 2005) [Last Accessed: May 2005].

[29] J. Soto, Statistical Testing of Random Number Generators, URL:http://www.csrc.nist.gov , (June 2000) [Last Accessed: June 2005].

[30] X. Wang, D. Feng, X. Lai, and Y. Hongbo, Collisions for Hash Functions MD4, MD5, Haval-128 and RIPEMD, URL:http://eprint.iacr.org/2004/199.pdf , (August 2004) [Last Accessed: December 2005].

[31] X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu, Cryptanalysis of the Hash Functions MD4 and RIPEMD., Advances in Cryptology--Eurocrypt 2005 (8-1-2004).

[32] R. Williams, A Painless Guide to CRC Error Detection Algorithms, URL:ftp.adelaide.edu.au/pub/rocksoft/crc_v3.txt , (August 1993) [Last Accessed: September 2005].

12.0 About the Author

Steve Mead is a Computer Scientist at NIST, working on the Computer Forensic Tool Testing (CFTT) project. He has 17 years of experience in system administration, security, and application development. Mead holds a B.A. in Psychology, and graduate degrees in Administration of Justice, Computer Science, and is currently completing a M.S. in Security Informatics at Johns Hopkins.