Empirical Evaluation of Hash Functions for Multipoint Measurements

Christian Henke, Carsten Schmoll, Tanja Zseby
Fraunhofer Institute FOKUS
Kaiserin-Augusta-Alle 31
10589 Berlin, Germany
{christian.henke,carsten.schmoll,tanja.zseby}@fokus.fraunhofer.de

ABSTRACT

A broad spectrum of network measurement applications demand passive multipoint measurements in which data from multiple observation points has to be correlated. Examples are the passive measurement of one-way delay or the observation of the path that a packet takes through a network. Nevertheless, due to high data rates and the need for fine granular measurements, the resource consumption for passive measurements can be immense. Furthermore, the resource consumption depends on the traffic in the network, which usually is highly dynamic. Packet and flow-selection methods provide a solution to reduce and control the resource consumption for passive measurements. In order to apply such techniques to multipoint measurements the selection processes need to be synchronized. Hash-based selection is a deterministic packet selection based on a hash function computed on selected parts of the packet content. This selection decision is consistent throughout the network and enables packet tracing and the measurement of delay between network nodes. Because the selection is based on deterministic function it can introduce bias which leads to wrong estimation of traffic characteristics. In this paper we define a set of quality criteria and select methods to investigate which hash function is most suitable for hash-based packet selection. We analyze 23 non-cryptographic and 2 cryptographic hash functions. Experiments are performed with real traffic traces from different networks. Based on the results we recommend 2 fast hash functions which show low bias and sample a representative subset of the population.

Categories and Subject Descriptors

C.2.3 [**Network Operations**]: Network Monitoring; C.4 [**Performance of Systems**]: Measurement techniques

General Terms

Network Measurement

Keywords

sampling, hash function, multipoint measurements, packet selection

1. INTRODUCTION

Network measurements play a vital role in operating and developing today's Internet. Beyond measuring of bandwidth utilization and packet count at a single measurement point there exists a variety of applications that require multipoint measurements ([20], [34], [22], [5]). Service Providers need to validate their delay guarantees from Service Level Agreements. Network research has incentives to track where packets are changed, reordered, lost, or delayed, for instance, in error-prone environments like mobile adhoc networks.

Hash-based selection is a passive measurement technique that enables multipoint measurements and packet tracing ([8], [29]). In contrast to active one-way delay measurements ([23], [25], [16]) no additional traffic needs to be introduced into the measured network and quality statements can be made directly about the customer traffic. Hash-based selection reduces the calculation effort because in contrast to other passive multipoint measurements it does not correlate all packets from the measurement points. Instead, with hash-based selection only a subset of packets at every measurement point is selected and the real traffic characteristics can be estimated.

Hash-based selection is realized by the following technique. Parts of the packet content (header and payload) that are invariant between measurement nodes are extracted and used as the hash input for a hash function. The hash function with a digest length of N bits maps the hash input to a value in the hash range R = $[0..2^N - 1]$. The packet itself is selected if the hash value falls into a predefined selection range $S \subset R$. In order to obtain a certain sample size one can adjust the selection range. The advantage of this technique is that the selection decision for each packet along its path is the same, provided that the selected packet content (hash input), hash function and selection range are the same at the different measurement points. Because of this quality, packets are consistently sampled along their path and can be correlated for delay and loss measurements at a common collector.

2. PROBLEM STATEMENT

Random selection techniques have the advantage to select an unbiased and representative subset of the population. This is inevitable for proper traffic estimation and for providing an accuracy statement from statistics [32]. Hash-based selection is a deterministic selection based on the packet content. Therefore it is by definition very likely that the selection is biased, i.e. packets with certain attributes may be preferred selected than others. Bias results in incorrect estimations about the real traffic properties and prevents the application of standard statistics. For instance: if packets with certain lengths are preferred in the selection, a sound estimation of the packet size distribution is impossible. The same applies to all packet properties that might have correlation with the packet length. The target quality is to emulate random packet selection with hash-based packet selection, but it has to be evaluated if this is possible. There are two settings in hash-based selection that can be configured and that have an influence on the selection decision quality:

1. the packet content used as hash input
2. the hash function

2.1 Hash Input

In accordance with Duffield [8] we define the following required properties for the hash input.

1. Invariant between network nodes. Fields that change between network nodes like Time To Live and IP Checksum cannot be used for hash-based selection, because the selection decision would differ between the network nodes.
2. Highly variable between packets. In order to avoid hash input collision (packets that are unequal but have the same hash input) one should use highly variable packet header fields. The higher the variability of the field, the higher the probability that packets differ in this field. In case the choice of the hash input results in too many collisions, bias is introduced to the selection. Because packets with the same hash input have the same hash value and selection decision, they are not selected independently. All packets that have the same hash input (and hence have common properties) will either be under- or overrepresented in the selected subset. Large collisions (large amount of packets with same hash input) are of more concern than the same amount of packets within multiple small collisions, because packets from different small collisions can have different selection decisions. In [13] we investigate header fields from real traffic traces in order to find a suitable hash input combination (see Sect. 3).

2.2 Hash Function

The choice of the hash function is of crucial importance to avoid bias. Although the selection decision is based on the hash function and the hash input, there are certain properties that a hash function should fulfill in order to be applicable for hash-based selection. At first, the hash function has to be fast, because a hash value is calculated for every packet. Second, the hash functions should disperse hash values randomly over the hash range. Otherwise, hash inputs which are similar will be mapped closely together, therefore packets that are similar will have a higher probability to have the same selection decision which introduces bias to the selection. The third criterion is that the selection decision based on the hash function should be unbiased to any packet attribute. As we will show later this relates to the fourth criteria: a representative subset of the population should be selected. In this paper we evaluate 25 hash functions on their suitability for hash-based selection. The evaluation is based on these four quality criteria upon which we develop and implement a set of tests in order to assess if hash-based selection can emulate random sampling.

3. STATE OF ART

Duffield and Grossglauser [8] were the first to introduce the hash-based selection technique with the purpose of packet tracing. Duffield evaluates a simple modulus hash function using four different traces from a campus network. He uses the chi-square independence test in order to analyze if there is dependence between network address prefix and sampling decision. Further, he investigates dependence between single bits of the network address and sampling decision. There has been no statistical sign for dependence if 40 input bytes were used.
Molina [17] analyzes four hash functions (CRC32, MMH, IPSX, BOB) for the purpose of hash-based selection. He investigates the hash functions' performances, hash value collision probability and the ability to uniformly distribute the hash values. He concludes that from the 4 evaluated hash functions, BOB performs slightly better than the 2nd placed CRC32 function in terms of performance and uniformity. Niccolini [20] and Raspall [26] use the MMH hash function for hash-based selection because of its calculation speed and uniform hash value distribution. The PSAMP [33] working group advices the use of the BOB hash function for hash-based selection; alternatively IPSX or CRC.
A general performance comparison of cryptographic hash functions can be found at [7] and [31]. MD5 and SHA are two of the fastest cryptographic hash functions. Nevertheless cryptographic hash functions are generally very computation expensive.
In [13] an entropy based approach is used to measure the variability of each header field byte. An 8 byte hash input configuration consisting of highly variable bytes is recommended for the use with hash-based packet selection. It could be shown that this configuration provides a comparable amount of hash input collisions to a configuration where the whole IP and transport header (except TTL and Checksum) are used as hash input and shows less collisions than a 16 byte hash input configuration proposed by Molina [17].
A recent paper from Goldberg and Rexford [11] points out security issues caused by hash-based selection. They prove that non-cryptographic hash functions are vulnerable to attacks because an adversary is able to craft packets that are disproportionally selected. They propose the use of a keyed pseudo random hash function, e.g. MD5 with a secret key that is appended to the hash input.

4. APPROACH

We investigate a collection of 25 hash functions on their suitability for hash-based selection. We define following quality criteria for a hash function intended for hash-based selection and how they can be tested:

1. Performance The hash value has to be calculated on each packet which is captured at an observation point. Hence, performance is very critical because an observation point like a network router has a limited amount of processing capacity. Usually only a small contingent of the nodes resources are allocated for measurements and measurements shall not influence the normal routing operation.

2. Non-linearity It is required that hash values are distributed randomly over the hash range. A linear hash function h has a linear dependency between hash input values x and x+1 to hash values, e.g. $h(x) + 1 = h(x + 1)$. This may be exploited by an adversary who intentionally crafts packets in order to influence the selected subset as shown by Goldberg and Rexford [11]. Linearity is also undesired because packets which only differ in some bits of the hash input are mapped closely together and have the same selection

Table 1: Hash Function Collection

Hash Fct.	Invented By	Source Code	Hash Fct.	Invented By	Source Code
BOB	Robert Jenkins	[14]	CRC32		[15]
OAAT	Robert Jenkins	[14]	BRPHash	Bruno Preuss	[15]
Simple		[18]	PYHash		[15]
SBOX		[18]	SDBM		[19]
TWMX	Thomas Wang	[15]	OCaml		[19]
Hsieh	Paul Hsieh	[15]	SML		[19]
RSHash	Robert Sedgewick	[15]	STL		[19]
JSHash	Justin Sobel	[15]	SHA		[9]
BKDR	B.Kernighan	[15]	MD5		[24]
DJBHash	Daniel Bernstein	[15]	MMH		[12]
NDJBHash	Daniel BErnstein	[15]	FNV32	Fowler Noll	[21]
DEKHash	Donald E.Knuth	[15]	CS	Carsten Schmoll	[28]
APHash	Arash Partow	[15]			

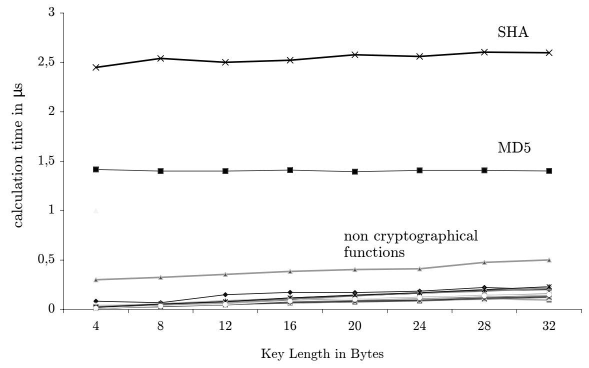
decision which introduces bias. The *avalanche criterion* is used to assess the ability to distribute hash values randomly over the hash range. Avalanche is the property that with the change of one input bit all output bits change with a probability of 50%. The closer this avalanche criterion is fulfilled the more random the hash values. Note that non-linearity does not necessarily mean that the hash function is secure in an adversarial setting, see [11] for details.

3. Unbiasedness Bias is introduced into the estimation if the hash function favors any packets or bitstreams to others in the selection decision. Duffield [8] uses the chi-square independence test to analyze if the sampling decision caused by the hash function is dependent to the network prefix for one linear hash function. We follow this approach and also use the chi-square test.

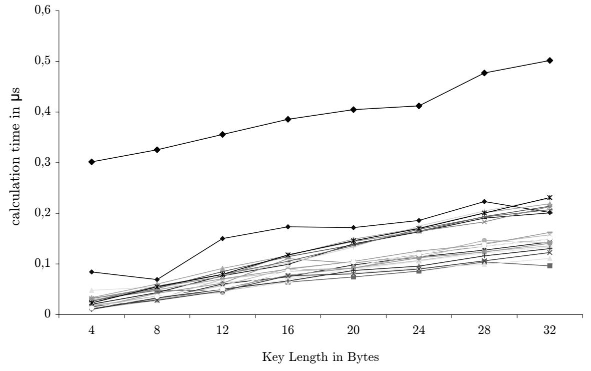
4. Representativeness of the selected subset The distribution of packet attributes in the sampled subset have to be the same as in the population in order to ensure good traffic characteristic estimation. In order to compare the proportions of packet attributes in the sampled subset and the population we considered the use of the *goodness-of-fit test*. Claffy [6] used the goodness-of-fit test to prove that random selection techniques sample a representative subset. Nevertheless as we later point in our case the chi-square independence test is stronger than the goodness-of-fit test and does not only evaluate biasedness but also representativeness of the selected subset.

5. COLLECTION OF HASH FUNCTIONS

We selected a collection of 23 non-cryptographic hash functions. All hash functions have a 32 bit digest (hash value) and a variable hash input length. Exemplary two cryptographic hash functions - SHA1 and MD5 - were added to the collection as well. The hash values of SHA (128 bits) and MD5 (160 bits) were trimmed to 32 bits by adding each 32 bit subblock. The functions were available at various online sources. An overview of the Collection is shown in Tab. 1.
In our work we focus on the non-cryptographic hash functions. As already shown in [11] non-cryptographic hash functions imply security impairments in an adversarial setting, where an adversary tries to influence the selection decision. The cryptographic MD5 and SHA hash function are proven to be robust for the adversarial setting and shall be compared to the other functions in order to assess if they show better quality for other criteria as well.



(a) Cryptographic/Non-Cryptographic Functions



(b) Only Non-Cryptographic Functions

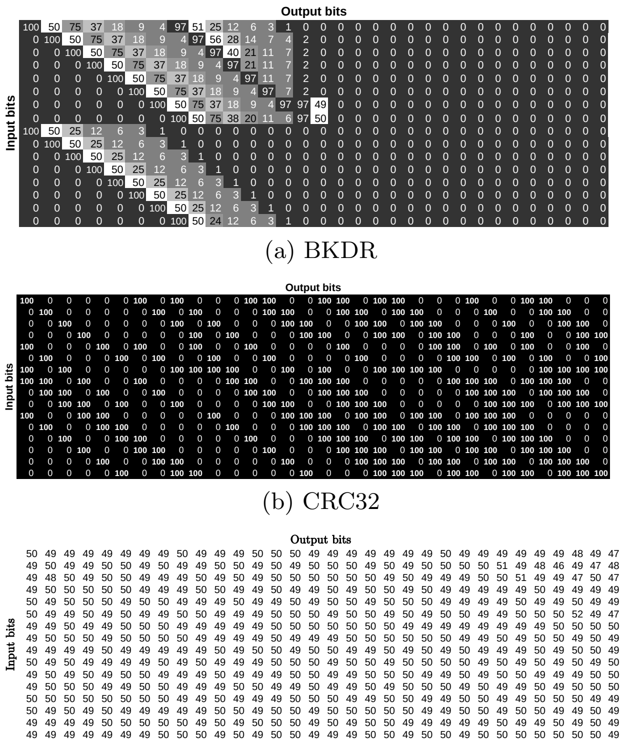
Figure 1: Hash Functions Performance

6. EVALUATION OF HASH FUNCTIONS

In this section we compare the hash functions in accordance to the quality criteria defined in Sect. 4.

6.1 Performance

The performance of the 25 hash functions is measured by an evaluation tool written in C++. The tool generates random artificial packets of definable length upon which all hash functions are applied. An average of 1 million repeated calculations is measured. The tool is run on a Dell Inspirion PC (2.8 GHz, 2048 KB cache, 1 GB RAM) with Kubuntu Feisty OS. Fig. 1 shows the measurement results for different packet lengths. Because of the amount of hash functions both subfigures only show a condensed view on the performance of both hash function groups. From Fig. 1(b) one can observe that the non-cryptographic hash functions have a linear increase of computation time in the range of 66% (SBOX) to 1260% (SML) when the hash input length is increased from 4 to 32 bytes. The computation time of SHA and MD5 are more or less constant with an increase of 0% and 5%. Fig. 1(a) shows the contrast in performance between the cryptographic hash functions (SHA, MD5) and non-cryptographic hash functions. For small packet lengths the average calculation time of all non-cryptographic hash functions (excluding CRC32) is 33 (MD5) and 55 (SHA) times faster than the cryptographic functions. With a 32 byte hash input length this discrepancy narrows with an average difference of 7 (MD5) and 12 (SHA) times the calcula-



(a) BKDR

(b) CRC32

(c) BOB

Figure 2: Avalanche with a 2 byte Hash Input

tion time. CRC32 is not depicted in these diagrams because it is very slow with about $200\mu s$. The majority of hash functions have a calculation time in the range of $0.05 - 0.12\mu s$ for a 12 byte hash input length. These results show that non-cryptographic hash functions are faster than SHA and MD5. For non-cryptographic hash functions one may consider to calculate the function on the hash values again for this may improve the hash value dispersion.
The evaluation of the different algorithm's performance is software based on a standard PC platform. Implementation on network card processors or FPGAs may show a different ordering of the performance results, since the algorithms have different potential for improvement by hardware parallelism on these platforms. Especially the CRC32 function can be efficiently designed with bit shifting registers.

6.2 Non-Linearity

The non-linearity of a hash function is with the avalanche criterion. The avalanche criterion is proposed by [10] and [4] to measure randomness of hash output values. Avalanche is the property that with the change of one input bit all output bits change with a probability of 50%. This implies that hash inputs that only differ in one bit disperse randomly. A C# implementation of the test is available at [18] which we adopted to C++ for the hash function collection. We conducted the avalanche test with different hash input lengths (2-16 bytes). Fig.2 illustrates three exemplary avalanche results of the BKDR, BOB and CRC32 hash functions with hash input length of two bytes. In the figures the 32 output bits are shown in the columns whereas the rows depict the two bytes of input. The numbers represent the probability of a change in that certain output bit caused by the input bit. Dark background colors and white font color represent poor avalanche whereas white background and dark numbers represent good avalanche. For the BKDR hash with a 2 byte hash input in Fig. 2(a) one can observe that the change of the second input bit always leads to a change of the second output bit, whereas the first output bit is not affected. The BKDR 2-byte example also demonstrates one common shortcoming of some analyzed hash functions: because the hash input is only 2 bytes long, the input information is not sufficient to affect all 32 output bits. The 2 least significant bytes are not changed at all. With increasing hash input length those functions show better avalanche. Fig. 2(b) shows the linear property of the CRC32 Hash Function. The BOB hash function has nearly perfect avalanche. All single input bits cause a change of all output bits with a probability close to 50%. Categorized results for all 25 hash functions are shown in Table 2. Four hash functions (CRC32, DEKHash, BRPHash, APHash) have a linear dependency between hash input and hash value. These function are very vulnerable to bias and security attacks.

Table 2: Avalanche Test Results

AP	BKDR	BOB	BRP	CRC32	CS	DEK	DJB	FNV32	Hsieh	JS	MD5	MMH	NDJB	OAAT	Ocaml	PY	RS	SBOX	SDBM	SHA	Simple	SML	STL	TWMX
--	+-	++	--	--	+-	--	-	+-	++	+-	++	-	+-	++	-	+-	+	++	+-	++	++	-	-	+

++	all bits are affected with 40-60% probability	-	not all bits are affected
+	all bits are affected	--	linear dependeny
+-	with increasing input length all bits are affected		

6.3 Unbiasedness

The unbiasedness of the selection decision can be tested by the chi-square independence test. The chi-square independence test [3] statistically evaluates the dependency between two categorical variables. We will use this test to evaluate if the sampling decision $X = \{x_1 = "selected", x_2 = "notselected"\}$ is independent to a packet attribute (e.g. packet length) Y= $\{y_1, .., y_L\}$ with L possible values, where y_i is a realization of the packet attribute (e.g. packet length=5). The hypotheses to test are:

H0: Selection decision (X) and packet attribute (Y) are independent
H1: Selection decision (X) and packet attribute (Y) are dependent

The test statistic S is the sum of deviations between expected ($\overline{h_{ij}}$) and observed (h_{ij}) simultaneous frequencies of x_i and y_i.

$$S_I = \sum_{i=1}^{2} \sum_{j=1}^{L} \frac{h_{ij} - \overline{h_{ij}}}{\overline{h_{ij}}} \qquad (1)$$

The error α to incorrectly reject H0 has to be defined prior to the test. Usually α is set to be 5%, i.e. although the variables X and Y are truly independent the test rejects H0 with a probability of 5%. Assuming that selection decision and packet attribute are independent S is asymptotically X^2 distributed with $(2-1)(L-1) = (L-1)$ degrees of freedom. The H0 hypotheses is rejected if S is above the critical X^2 value $X_{crit}((L-1); \alpha)$.

6.4 Representativeness

The representativeness of the selected subset is given if the subset includes the same relative frequencies of packet attributes as there are in the population. The chi-square goodness-of-fit test [3] is used to evaluate if observed frequencies follow a specified distribution. The hypotheses to test are:

H0: The packet attribute distribution in the selected subset and the population are equal
H1: The packet attribute distribution in the selected subset and the population are not equal

The test statistic S_D is the sum of deviations between the fraction of packet attributes $\overline{h_i}$ in the population and the observed fraction h_i in the sampled subset, where i=(1..L) denotes the different realizations of the attribute.

$$S_D = \sum_{i=1}^{L} \frac{h_i - \overline{h_i}}{\overline{h_i}} \tag{2}$$

The test statistic S_D is X^2 distributed with L-1 degrees of freedom under the assumption that both attribute distributions are equal. If the test statistic S_D is above $X_{crit}(L-1;\alpha)$ the distributions in the sampled subset and the population are different.

Derivation of Chi-Square Goodness-of-Fit Test from Independence Test.

One has the notion that the test statements of the independence and goodness-of-fit test are causally related. If the sampling decision is unbiased to a packet attribute, the selected subset should be representative as well. This shall be derived in short here.
The test statistic S_I of the chi-square independence test between any packet attribute and the sampling decision can also be written as (cf. Eq. 1):

$$S_I = \underbrace{\sum_{i=1}^{L} \frac{h_{i1} - \overline{h_{i1}}}{\overline{h_{i1}}}}_{Term\ 1} + \underbrace{\sum_{i=1}^{L} \frac{h_{i2} - \overline{h_{i1}}}{\overline{h_{i2}}}}_{Term\ 2} \tag{3}$$

Hence term 1 in Eq. 3 equals the test statistic S_D from the chi-square distribution test (Eq. 2). One can note that the independence test statistic S_I is larger than S_D because it includes the additional summands of term 2. Because both tests have equal critical X^2 values, more independence tests are rejected than goodness-of-fit tests.
For our evaluation one can note that an unbiased test result will always lead to a representative subset, but a passed goodness-of-fit test does not mean that the unbiasedness test passes as well. As for this work only hash functions that comply with both criteria are of interest, only the independence test is used to verify unbiasedness and representativeness.

6.5 Independence Test

6.5.1 Test Setup

I. Analyzed Traces.

In order to assess the quality of the hash functions we conduct multiple chi-square independence tests on real traffic traces. For the tests we use two (in [13] prior screened) real traffic trace groups. The Twente and FH Salzburg traces can be found at the MOME database [2]. We use 51 FH Salzburg traces that are captured at an WAN Access Network on a student campus and 100 Twente traces that are captured at an aggregated uplink of an ADSL access network. The FH Salzburg traces comprise of mostly http traffic whereas the Twente traces include a variety of applications with only 2% http traffic and applications each not composing more than 1% of the traffic. Both traces include the IP and Transport Layer Header. We have specially chosen these two trace groups because they include very few identical packets (both about 0.05%). Nevertheless, in [13] we have shown that the collisions for the Twente traces are more condensed, i.e. more packets are included in large collisions, whereas for the FH Salzburg traces more smaller collisions are observed. As pointed out in Sect. 2 identical packets cause bias that cannot be reduced by the underlying hash function.

II. Multiple Independence Tests.

The independence test is based on 51 FH Salzburg and 100 Twente traces. We will trim each of the FH Salzburg trace files to 200.000 packets and the Twente traces to 1 Mio packets. For each trace file a separate independence tests is conducted which leads to multiple independence tests for each trace group. The reason for using separate independence test are the following: 1) measurement intervals are simulated and 2) the effect on the bias caused by packets with the same hash input is mitigated as the hash input collisions are kept small.
The problem with multiple chi-square tests is that every test inherits an error. As earlier noted, we will discard H0 in α (e.g. 5%) of the individual tests although there is no dependency between the variables. We are not interested in individual tests that falsely reject H0, but in tests that show a true dependency. A common approach when multiple statistical tests are performed is the Dunn-Sidak correction [27] [30]. Under the assumption of H0, the probability of falsely rejecting at least one out of n independent individual tests, each of level α_{ind}, is $1-(1-\alpha_{ind})^n$. For the correction we have to define a global test, that restricts the combined error of all individual tests

global H0: There is no dependency for all test.
global H1: There is dependency for at least one test.

We set the global error α_{global} to be 5%, i.e. if there is no dependency global H0 is only rejected with a 5% probability. In order to achieve a global error of 5% we have to adjust the individual test error α_{ind}.

$$\alpha_{ind} = 1 - (1 - \alpha_{global})^{\frac{1}{n}} \tag{4}$$

Because the individual test error α_{ind} is lowered, the critical X^2_{crit} for the global test is above the individual tests critical values.

III. Hash Input Configuration.

As the hash input we will use the 8 high entropy bytes that we proposed in [13]. These 8 bytes are the IP identification field and 6 bytes depending on the transport protocol: TCP (Checksum, 2 LSB of Sequence and Acknowledgment Number) UDP (Checksum, Source Port, LSB Destination Port, LSB Length) ICMP (Checksum, Bytes 12,13,18,19). The selection range includes a 10% slice of the hash range. We do not use a secret key in the evaluation, therefore all hash functions are evaluated in an unkeyed version. It is part of future work which type of concatenation (append, XOR, etc.) of the secret key is most useful.

IV. Tested Attributes.

The chi square independence tests are conducted for three different packet attributes:

I. packet length
II. transport protocol
III. byte value

6.5.2 Test Results

I. Independence of Sampling Decision and Packet Length.

The packets are categorized into 7 length groups as in [1]: 0-44 Bytes, 45-90 Bytes, 91-180 Bytes, 181-260 Bytes, 261-576 Bytes, 577-1120 Bytes and longer than 1120 Bytes. It is evaluated if packets within a certain length category have a higher sampling probability as in another category.

The Box-Whisker-Plots for the hash function collection is shown in Fig. 3. The Box-Whiskers-Plots depict the minimum, 5% quantil, median, 95% quantil and maximum of the test statistic S for the 51 FH Salzburg and 100 Twente traces. The 95% quantil is chosen to depict the amount of tests that are expected to accept H0 under independence. On the very left the distribution of the test statistic S for random probabilistic sampling (p=10%) is depicted as reference. The lower horizontal line represents the critical value $X^2_{crit} = 12.6$. All tests that have a test statistic S above this value reject H0. It is noticeable that the majority of hash functions perform very well; their distribution of S is not obviously different to probabilistic sampling.

Dunn-Sidak The 95% quantil of the Box-Whisker-Plot shows if more or less than 5% of the test reject H0. The tests based on random probabilistic sampling reject H0 3 times for FH Salzburg and one time for Twente traces although dependency can be excluded by definition. Hence we need to adjust the error α to obtain only tests that show true dependency. Applying the Dunn-Sidak correction (Eq. 4) we obtain a global critical values for FH Salzburg (51 tests) $X^2_{crit} = 22.4$ and for Twente (100 tests) $X^2_{crit} = 24.0$. The upper horizontal lines in Fig. 3 show the adjusted global X^2_{crit} values. Each hash function that has one rejected test (S above global X^2_{crit}) is truly biased at least for this trace. For example there is only one test of the AP hash function based on the FH Salzburg traces that is above the individual critical value. Nevertheless this test shows true dependency because its test statistic is even above the global critical value.
It is noticeable that the results for the Twente traces are worse than the one of FH Salzburg. Especially hash functions that indicate biasedness for the FH Salzburg traces show significant dependency for the Twente traces (e.g. AP, BKDR, BRP). There are hash functions that did not show any dependency for the FH Salzburg traces but show strong dependency for the Twente traces (SDBM, Simple, SML). There are 3 causes that may explain this difference. 1) The traffic traces include different traffic 2) the measurement interval for the Twente traces is larger and hence more hash input collisions are included 3) the hash input collisions are larger for Twente than for FH Salzburg.
The number of rejections for individual and global tests are shown in Table 3. In the table all hash functions that produce an unbiased subset in all tests for the 8 byte hash input configuration are marked: BOB, MD5, CRC32, OAAT, RS, SHA and TWMX. The SBOX, Hsieh and FNV32 hash function have one global rejected test.

II. Independence of Sampling Decision and Protocol.

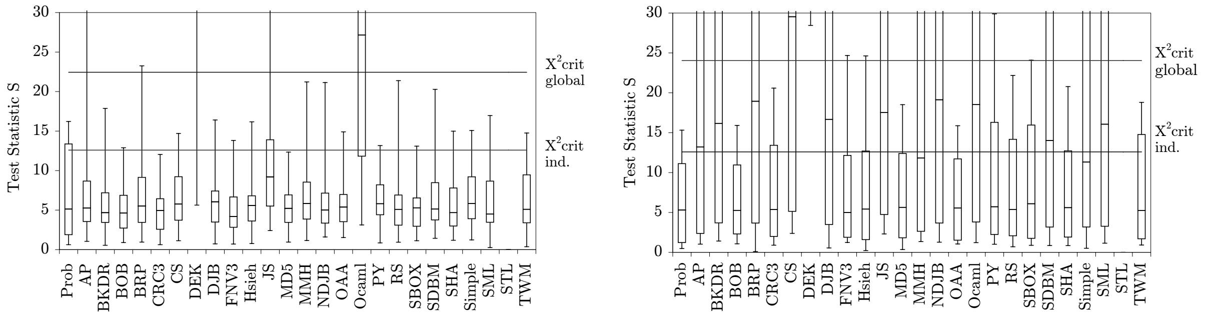
Only three transport protocols are considered for the chi-square independence test; others are filtered out: TCP, UDP and ICMP. We will conduct the independence test in order to evaluate if there is any dependency between the sampling decision and these 3 transport protocols. The results are presented in the Box-Whisker-Plots in Fig. 4. The lower horizontal line represents the critical X^2 above which all individual tests reject H0. The upper horizontal line represents the Dunn-Sidak adjusted global critical X^2 value. All tests that have a test statistic S above this value show true dependency between sampling decision and protocol for this trace file. The number of rejections for individual and global tests for all hash functions is shown in Tab. 4. Again it is obvious that there is a difference of the selection decision based on the Twente and FH Salzburg traces for the hash functions that already failed in the length independence test. Nevertheless there are 10 hash functions that do not indicate any dependency between sampling decision and transport protocol: BOB, CRC32, FNV, Hsieh, MD5, OAAT, RS, SBOX, SHA and TWMX.

III. Independence of Sampling Decision and Byte Value.

Hash functions are indifferent to the conceptional model of packet header fields, they calculate on bits and bytes. Therefore it is evaluated if any byte that is used as hash input has an influence on the sampling decision. For instance if a certain value in the hash input byte number 2 is favored to other values in byte number 2.
For each byte of the hash input an individual chi-square independence test is conducted. Hence, there are 8 tests for each trace and hash function. This is a total of 408 tests for FH Salzburg and 800 for the Twente trace group. Because for some bytes not all possible 256 values are occupied, the degrees of freedom for the tests differ. The critical X^2 values are calculated with as many degrees of freedom as there are in the specific trace for this byte. The test statistic S is divided by this critical X^2 value in order to be able to compare results for different chi-square tests with different degrees of freedom. The obtained metric is the relative test statistic R on which the test decision can be based. If R is above 1 the tests rejects H0:

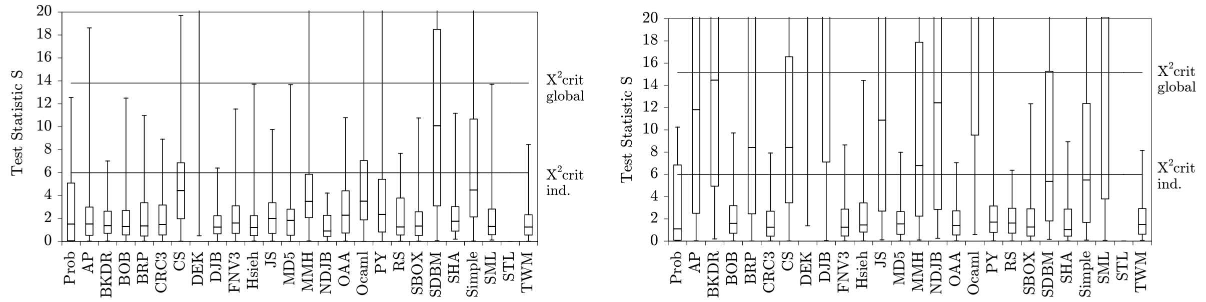
$$R = \frac{S}{X^2_{critical}} \rightarrow \; if \; R \begin{cases} \leq 1 & \text{H0 not rejected} \\ > 1 & \text{H0 rejected} \end{cases}$$

An exemplary graphical result for the DJB hash function based on the FH Salzburg traces is shown in Fig. 5. The



(a) FH Salzburg (b) Twente

Figure 3: Distribution of Test Statistic S for Independence Test on Length



(a) FH Salzburg (b) Twente

Figure 4: Distribution of Test Statistic S for Independence Test Protocol

Table 3: Individual and Global Rejections of Independence Test Packet Length

Function	Prob	AP	BKDR	**BOB**	BRP	**CRC32**	CS	DEK	DJB	FNV32	Hsieh	JS	**MD5**	MMH	NDJB	**OAAT**	Ocaml	PY	**RS**	SBOX	SDBM	**SHA**	Simple	SML	STL	**TWMX**
Individual Tests																										
FH (51)	3	4	3	**1**	9	**0**	4	49	3	3	1	16	**0**	4	5	**2**	37	1	**4**	1	6	**2**	2	2	x	**3**
Twente (100)	1	51	67	**1**	64	**6**	75	100	57	4	6	69	**5**	44	63	**4**	69	9	**8**	10	55	**6**	43	63	x	**6**
Global Tests																										
FH (51)	0	1	0	**0**	1	**0**	0	47	0	0	0	3	**0**	0	0	**0**	28	0	**0**	0	0	**0**	0	0	x	**0**
Twente (100)	0	34	38	**0**	46	**0**	57	100	37	1	1	39	**0**	16	43	**0**	37	1	**0**	1	27	**0**	21	38	x	**0**

Table 4: Individual and Global Rejections of Independence Test Protocol

Function	Prob	AP	BKDR	**BOB**	BRP	**CRC32**	CS	DEK	DJB	**FNV32**	**Hsieh**	JS	**MD5**	MMH	NDJB	**OAAT**	Ocaml	PY	**RS**	**SBOX**	SDBM	**SHA**	Simple	SML	STL	**TWMX**
Individual Tests																										
FH (51)	2	4	3	**4**	3	**2**	20	47	1	**6**	**3**	3	**7**	13	0	**7**	15	10	**3**	**5**	30	**5**	23	5	x	**2**
Twente (100)	6	64	74	**5**	60	**2**	60	94	76	**7**	**6**	57	**2**	54	62	**4**	80	12	**2**	**7**	47	**8**	46	72	x	**6**
Global Tests																										
FH (51)	0	1	0	**0**	0	**0**	3	45	0	**0**	**0**	0	**0**	5	0	**0**	2	2	**0**	**0**	20	**0**	9	0	x	**0**
Twente (100)	0	44	49	**0**	39	**0**	30	87	65	**0**	**0**	40	**0**	29	44	**0**	67	2	**0**	**0**	26	**0**	21	59	x	**0**

Table 5: Global Rejections of Independence Test For Byte Values

	AP	BKDR	**BOB**	BRP	CRC32	CS	DEK	DJB	FNV32	**Hsieh**	JS	**MD5**	MMH	NDJB	**OAAT**	Ocaml	PY	**RS**	**SBOX**	SDBM	SHA	Simple	SML	STL	**TWMX**
FH Salz (408)	6	2	**0**	0	1	211	203	21	1	**0**	5	**0**	34	3	**0**	216	8	**0**	**0**	43	1	30	20	x	**0**
Twente (800)	286	298	**0**	352	0	516	721	255	0	**0**	361	**0**	500	351	**0**	500	15	**0**	**0**	218	0	152	271	x	**0**

Table 6: Summarized Test Result

Function	BOB	OAAT	TWMX	RS	Hsieh	SBOX	MD5	SHA	FNV32	NDJB	PY	SDBM	SML	BRP	JS	Ocaml	STL	AP	BKDR	Simple	DJB	CS	DEK	CRC32	MMH
Performance	++	++	++	++	++	+	-	--	++	++	++	++	++	++	++	++	++	++	++	++	++	++	++	--	++
Avalanche	++	++	+	+	++	++	++	++	+-	+-	+-	+-	-	--	+-	-	-	--	+-	++	-	+-	--	--	-
Bias	++	++	++	++	+	+	++	+	+-	-	+-	--	--	--	--	--	--	--	--	--	--	--	--	+	--

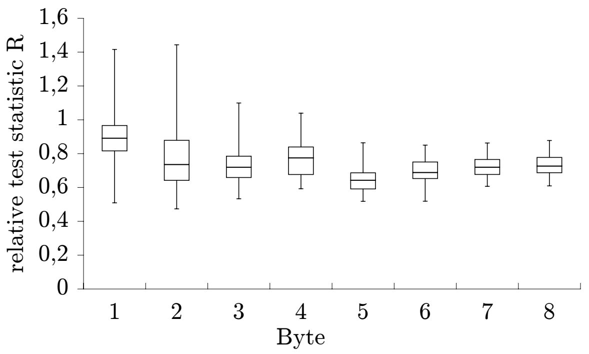


Figure 5: Independence Test on Byte for DJB Hash Function for FH Salzburg Group

tests that show a relative test statistic R above 1 reject the global H0 and statistically prove dependency between a byte value and the sampling decision. It is observable that the sampling decision of the DJB hash function based is dependent on the first 4 bytes of the hash input, because at least one of the 51 tests for these bytes is rejected. For the last 4 bytes a dependence cannot be proven.
7 hash functions (BOB, Hsieh, MD5, RS, SBOX, TWMX) show no dependency between any hash input byte and the sampling decision. With 1 rejected test for one byte the CRC32, FNV32 and SHA perform reasonably well.

7. CONCLUSION

We analyzed a collection of 25 hash functions on their suitability for hash-based selection. The hash functions were evaluated on 4 criteria: performance, non-linearity, unbiasedness and representativeness. The evaluation of unbiasedness and representativeness were based on real traffic traces. Categorial Results are shown in Tab. 6. Hash funcions on the left are most suitable for hash-based packet selection because they satisfy the quality criteria best.

Cryptographic Hash Function Both cryptographic hash functions SHA and MD5 show perfect avalanche. The MD5 hash function has no rejected independence test whereas SHA shows dependency between sampling decision and the byte value because of one rejected individual test. This may be caused by the trimming to 32bit output values. As shown in the performance evaluation both cryptographic hash functions imply heavy processing time compared to their non-cryptographic counterparts. For the short 8 byte hash input configuration SHA is 35 and MD5 20 times slower than BOB or OAAT. Especially for low-resource measurement nodes, MD5 and SHA cannot be recommended because the hash needs to be calculated over each hash packet. It has to be evaluated if hardware based approaches may narrow the performance differences to the non-cryptographic hash functions. Nevertheless it has to be clearly stated, that in an adversarial setting only a cryptographic hash function can be applied to ensure proper sampling.

Emulation of Random Selection Two properties of random selection are important for accurate measurements: unbiasedness and representativeness. We have shown that the tests for biasedness of the selection decision and the representativeness of the selected subset is causal related and can be analyzed with the chi-square independence test. We have shown that BOB, MD5, OAAT, RS and TWMX hash function are able to select an unbiased and representative sample based on our recommended 8 byte configuration. Nevertheless hash-based selection is not a random selection, because it is deterministically based on the packet content. Hence results are trace dependent and we have to be aware that the selection decision of identical packets will be cause bias.

Measurement Interval and Input Collision The results have also shown that there is a difference between the results for the two trace groups. As already pointed out there are two reasons for this difference. We used larger measurement intervals for the Twente traces and the input collision are more condensed (and larger) for the Twente traces. In order to diminish this effect one has to keep measurement intervals short and periodically change the selection range or secret key in order to ensure different sampling decisions between packets with the same hash input. This reconfiguration needs to be synchronized between the measurement nodes.

Recommended Hash Functions The BOB and OAAT hash function master all the tests (performance, avalanche, chi-square tests) and are strongly recommended for hash-based selection. Other well performing hash-functions are SBOX, Hsieh, TWMX, RS, SHA and MD5. The Hsieh function ($0.05 - 0.10 \mu s$) is the fastest hash function of these 7, whereas SHA is the slowest ($2.5 \mu s$). BOB, OAAT are recommended for low-resource measurement nodes in a non-adversarial setting. In order to provide some additional security it advised to use a secret key with the non-cryptographic hash functions as well, although this does not make the selection secure.
Our analysis confirms the recommendation by the PSAMP group to use BOB and provides further alternatives.

8. FUTURE WORK

In order to validate the results it is necessary to use more traces. Especially IPv6 traces are interesting in order to verify the selection decision quality based on the hash input of IPv6 traces. Unfortunately it is difficult to obtain IPv6 traces with more packet information than the IP header.

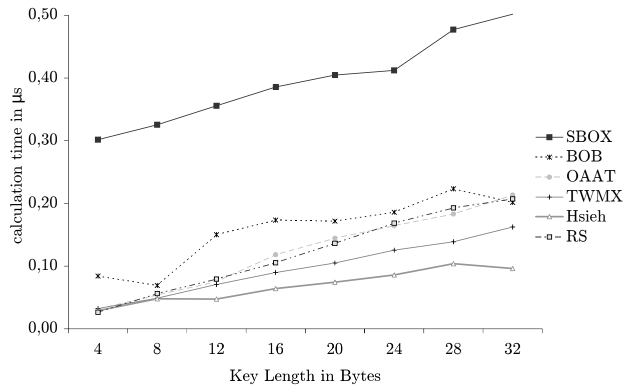


Figure 6: Performance of Recommended Hash Functions

In our paper we evaluated mostly non-cryptographic hash functions. It has to be assessed if there are cryptographic hash functions. A thorough analysis of which characteristics of cryptographic hash functions are required for hash-based packet selection can lead to cryptographic hash functions that are faster than MD5 and SHA but still provide efficient security.
Packet ID generation is an adjacent field to hash-based selection in multipoint measurements. Each packet traversing a measurement point will be assigned a packet ID and timestamp which are exported to a multipoint collector. According to this packet ID the packets observations can be correlated for packet tracing and delay measurements. This packet ID is assigned to each packet after the selection process (e.g. after hash-based selection). In [8] [17] [20] it is proposed to use different hash functions (or a different key respectively initial value) for packet ID generation and hash-based selection. A hash function for packet ID generation requires different qualities than the one used for hash-based selection, essentially only hash functions with a low collision probability are useful. Packet ID collisions are critical, because the trajectories of duplicate packet IDs can be misinterpreted as a trace from single packets which leads to wrong delay measurement results. In future we will analyze our hash function collection on their suitability for packet ID generation in terms of collision probability. Further we will evaluate under which circumstances the hash-based selection hash value can be used as packet ID.

9. REFERENCES

[1] *The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone*, Geneva, Switzerland, July 1998.

[2] MOME traffic measurement database. http://www.ist-mome.org/, March 2007.

[3] K. Bosch. *Statistik Taschenbuch.* R.Oldenbourg Verlag, 1998.

[4] J. C. H. Castro, J. M. Sierra, and A. Seznec. The strict avalanche criterion randomness test. *Math. Comput. Simul.*, 68(1):1–7, 2005.

[5] B.-Y. Choi, S. Moon, R. Cruz, Z.-L. Zhang, and C. Diot. Practical delay monitoring for ISPs. In *CoNEXT'05: Proceedings of the 2005 ACM conference on Emerging network experiment and technology.*

[6] K. C. Claffy, G. C. Polyzos, and H.-W. Braun. Application of sampling methodologies to network traffic characterization. In *SIGCOMM '93: Conference proceedings on Communications architectures, protocols and applications*, pages 194–203, New York, NY, USA, 1993. ACM Press.

[7] W. Dai. Crypto++ 5.5 benchmarks. http://www.cryptopp.com/benchmarks.html, February 2007.

[8] N. Duffield and M. Grossglauser. Trajectory sampling for direct traffic observation. *IEEE/ACM Trans. Netw.*, 9(3):280–292, 2001.

[9] D. E. Eastlake and P. E. Jones. US secure hash algorithm 1 (SHA1). *accessible under http://www.faqs.org/rfcs/rfc3174.html*, September 2001.

[10] R. Forré. The strict avalanche criterion: spectral properties of boolean functions and an extended definition. In *CRYPTO '88: Proceedings on Advances in cryptology*, pages 450–468, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[11] S. Goldberg and J. Rexford. Security vulnerabilities and solutions for packet sampling. *IEEE Sarnoff Symposium*, May 2007.

[12] S. Halevi and H. Krawczyk. MMH: Software message authentication in the gbit/second rates. In *FSE '97: Proceedings of the 4th International Workshop on Fast Software Encryption*, pages 172–189, London, UK, 1997. Springer-Verlag.

[13] C. Henke, C. Schmoll, and T. Zseby. Entropy based evaluation of packet headers for hash-based multipoint sampling. In *PAM '08: Proceeding of the 9th Passive and Active Measurement Conference 2008*, pages 82–91. Springer-Verlag.

[14] R. Jenkins. http://www.burtleburtle.net/bob/hash/doobs.html, September 1997.

[15] K. Lovett. Miscellaneous hash functions. http://www.call-with-current-continuation.org/eggs/hashes.html, February 2007.

[16] M. J. Luckie, A. J. McGregor, and H.-W. Braun. Towards improving packet probing techniques. In *IMW '01: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pages 145–150, New York, NY, USA, 2001. ACM Press.

[17] M. Molina, S. Niccolini, and N. G. Duffield. Comparative experimental study of hash functions applied to packet sampling. *International Teletraffic Congress (ITC-19)*, August 2005.

[18] B. Mulvej. Hash functions. http://bretm.home.comcast.net/hash/, February 2007.

[19] M. Neumann. Guugelhupf - design und implementation. http://www.fantasy-coders.de/projects/gh/html/x435.html, February 2007.

[20] S. Niccolini, M. Molina, F. Raspall, and S. Tartarelli. Design and implementation of a one way delay passive measurement system. *Network Operations and Management Symposium*, pages 469–482, April 2004.

[21] L. C. Noll. Fowler/Noll/Vo hash. http://www.isthe.com/chongo/tech/comp/fnv/FNV-source.

[22] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, F. Tobagi, and C. Diot. Analysis of measured single-hop delay from an operational back bone network. In *IEEE Infocom*, New York, June 2002.

[23] V. E. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, EECS Department, University of California, Berkeley, 1997.

[24] C. Plumb. Md5 message-digest algorithm. http://archiv.tu-chemnitz.de/pub/1999/0004/data/md5/md5.c, February 2007.

[25] H. Pucha, Y. Zhang, Z. M. Mao, and Y. C. Hu. Understanding network delay changes caused by routing events. *SIGMETRICS Perform. Eval. Rev.*, 35(1), 2007.

[26] F. Raspall, J. Quittek, M. Brunner, and M. Martin. Path-coupled configuration of passive measurements. *Inter-domain Performance and Simulation Workshop*, March 2004.

[27] J. Rupert G. Miller. *Simultaneous Statistical Interference*, pages 5–8,15–16. Springer-Verlag, 1981.

[28] C. Schmoll. Simple hash. http://net.fokus.fraunhofer.de/pub/hash_cs.txt, 2007.

[29] A. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer. Single-packet ip traceback. *IEEE/ACM Trans. Netw.*, 10(6):721–734, 2002.

[30] B. Walsh. Multiple comparisons: Bonferroni corrections and false discovery rates. Lecture Notes. http://nitro.biosci.arizona.edu/courses/EEB581-2006/handouts/Multiple.pdf.

[31] W.Erhardt. CRChash. http://home.netsurf.de/wolfgang.ehrhardt/crchash_de.html, February 2007.

[32] T. Zseby. *Statistical Sampling for Non-Intrusive Measurements in IP Networks*. PhD thesis, Technical University Berlin, 2005.

[33] T. Zseby, M. Molina, and et al. Sampling and filtering techniques for ip packet selection. *IETF Internet Draft, accessible under http://www.ietf.org/internet-drafts/draft-ietf-psamp-sample-tech-10.txt*, 2007.

[34] T. Zseby, S. Zander, and G. Carle. Evaluation of building blocks for passive one-way-delay measurements. In *Workshop on Passive and Active Measurements*, Amsterdam, Netherlands, April 2001.