

---

# Evaluation of Hash Functions for Multipoint Sampling in IP Networks

**Diplomarbeit**  
am Fraunhofer Institut Fokus  
betreut vom Fachgebiet Telekommunikationsnetze  
Technische Universität Berlin

vorgelegt von  
**Christian Henke**

am 10. 04. 2008

Betreuer Fraunhofer Fokus: Dr.-Ing. Tanja Zseby  
Dipl.-Ing. Carsten Schmoll  
Betreuer TU Berlin: Prof. Dr.-Ing. Adam Wolisz

Christian Henke  
Matrikelnummer: 205724

---



## Abstract

Network Measurements play an essential role in operating and developing today's Internet. A variety of measurement applications demand for multipoint network measurements, e.g. service providers need to validate their delay guarantees from Service Level Agreements and network engineers have incentives to track where packets are changed, reordered, lost or delayed. Multipoint measurements create an immense amount of measurement data which demands for high resource measurement infrastructure. Data selection techniques, like sampling and filtering, provide efficient solutions for reducing resource consumption while still maintaining sufficient information about the metrics of interest. But not all selection techniques are suitable for multipoint measurements; only deterministic filtering allows a synchronized selection of packets at multiple observation points. Nevertheless a filter bases its selection decision on the packet content and hence is suspect to bias, i.e the selected subset is not representative for the whole population. Hash-based selection is a filtering method that tries to emulate random selection in order to obtain a representative sample for accurate estimations of traffic characteristics.

The subject of the thesis is to assess which hash function and which packet content should be used for hash-based selection to obtain a seemingly random and unbiased selection of packets. This thesis empirically analyzes 25 hash functions and different packet content combinations on their suitability for hash-based selection. Experiments are based on a collection of 7 real traffic groups from different networks.



## Zusammenfassung

Netzwerk Messungen sind essentiell für die Verwaltung und Weiterentwicklung des heutigen Internets. Es existieren immer mehr Messapplikationen die Daten von mehreren Punkten benötigen, z.B. müssen Internet Provider ihre Vereinbarungen über maximale Übertragungsverzögerung verifizieren und Netzwerk Entwickler möchten wissen, wo im Netzwerk Internetpakete verloren oder verzögert werden. Mehrpunktmessungen haben aber den Nachteil, dass sie eine große Menge von Messdaten produzieren, deren Verwaltung und Speicherung hohe Messinfrastrukturkosten mit sich ziehen. Es existieren einige Stichprobenmethoden die es ermöglichen die Datenmenge zu reduzieren und die Messdaten zu schätzen. Nicht alle Stichprobenverfahren sind geeignet für Mehrpunktmessungen, nur deterministische Filter gewährleisten, dass an jedem Messpunkt die gleiche Stichprobe genommen wird. Allerdings ist die Selektionsentscheidung für ein Paket bei einem Filter deterministisch abhängig von dessen Inhalt und von daher ist nicht gewährleistet, dass es sich bei den selektierten Paketen um eine repräsentative Stichprobe handelt. Ein auf einer Hashfunktion basierendes Filterverfahren soll dazu verwendet werden eine Zufallstichprobe nachzubilden um möglichst genaue Schätzungen der Messdaten zu ermöglichen.

In dieser Arbeit wird analysiert welche Hash Funktion und welche Paketinhalte für dieses Verfahren verwendet sollen, um eine möglichst unverzerrte Schätzung zu erhalten. Es werden 25 Hash Funktionen und verschiedene Paketinhalte auf ihre Eignung für das Selektionsverfahren untersucht.



## **Acknowledgment**

First of all I would like to thank Tanja Zseby and Carsten Schmoll for giving me the opportunity to write this diploma thesis in a research centered topic at Fraunhofer Fokus. Their subtle supervision gave me confidence in my own ideas and helped me improve my researching ability. I am also very grateful that Prof. Dr.-Ing. Adam Wolisz pointed out critical issues as the adviser at the TU Berlin. Many thanks go to my colleagues Nikolaos Chatzis and Thomas Hirsch for their help and friendship. When I was struggling to the mist of C++ pointer or droughts of ideas they showed me the right way out of misery.

Special thanks go to Nora Eisemann who I could talk to about statistical matters. With her knowledge she helped me to tweak equations and make the calculations more worthwhile.

Last but not least I would like to thank my family. My father's joy about me doing research and my mother's support encouraged me to delve into the work. Thanks to my beloved Katja, the person behind myself, who gives me strength when no one else can.



# Contents



# List of Figures



# List of Tables

# Chapter 1

## Introduction

The intention of this introduction is to describe the scope of hash-based selection. The first section will explain the motivation for network measurements, main driving forces and why network measurement serves research and commercial demands. Multipoint measurements are a challenging area of network measurement, because an immense amount of measurement data is created. Hence one often has to assess if it is necessary and applicable to measure all packets. For a variety of scenarios it is rather useful to only select a subset of packets. Different filtering and sampling techniques will be compared on their suitability for multipoint measurements. Further, scenarios for hash-based selection will be presented. At the end of the introduction I will define the targets of the thesis.

### 1.1 Measurements in IP Networks

Network Measurements play an essential part in operating and developing today's Internet. From the operating view, Internet Service Providers (ISP) need to measure data volumes and usage duration on a per customer basis in order to account for used services. Research is asking for more measurement information in order to analyze problems in different topics. The diversity of network nodes and the variety of network hardware, topologies, links, protocols and applications make it a challenge to identify why and where packets get fragmented, changed, reordered, lost or delayed. Network measurements help to understand network behavior and can initiate ideas for improvement. The following classification of applications shows motivations for network measurements according to [?] and [?].

#### Usage-Based Accounting

As already pointed out, ISPs may charge their customers for the usage of Internet access and Internet services depending on usage duration or transfer volume. This implies that the service providers can measure the transmitted traffic in order to bill their customers.

## **Traffic Engineering**

Traffic Engineering is concerned with network performance and resource optimization. One goal of traffic engineering is efficient load balancing between network nodes to ensure that network nodes on one path do not become overutilized while other adjoining paths are underutilized. On the basis of measurements of link utilization and routing information, one can adjust load balancing policies to avoid congestion. As an example measurements in ad-hoc wireless networks are a wide research area, because engineers want to understand problems and improve routing policies.

## **Traffic Profiling**

Traffic Profiling serves multiple applications. It is a process of characterizing traffic flows by measuring duration, volume, burstiness and packet information like protocols and services. The data is used for network planning, trend analysis and business models development. For example one can measure which influence VoIP traffic has on network performance.

## **Attack/Intrusion Detection**

Network measurement is inevitable for attack and intrusion detection. Depending on the type of network attack, specific patterns can occur in the network traffic. For example an abnormal increase of small flows in a network node is suspicious of a bot net attack. In case those attack patterns are detected a network can commence counter actions to regain normal traffic status.

## **QoS Monitoring**

Service providers and customers often negotiate on different quality metrics of the provided service. These Service Level Agreements (SLA) require the measurement of Quality of Service (QoS) parameters in order to validate the contracts. For instance customers demand measurements to ensure that their TV-on-demand provider really complies with promised reachability and delay.

## **Fault management and network maintenance**

Network maintenance serves the goal of configuring and monitoring the network. Fault management as a part of network management shall detect, recover and limit the impact of failures in the network. For instance, if a network link is broken, measurements automatically detect the failure and the network can be reconfigured.

## **1.2 Active and Passive Measurements**

Network measurements can be distinguished in *active (intrusive)* and *passive (non intrusive)* measurements. Active measurements [?] [?] inject test traffic into the network in order to measure network characteristics. For example ping and traceroute send ICMP echo ping requests to a specified destination. On the basis

of the echo reply, the availability of the destination, the path and delay can be measured. Active measurements produce additional traffic load on the network which distorts measurement results. Further one cannot ensure that the active packets are equally handled as the traffic that they are intended to measure. In contrast, passive measurements only use existing traffic to measure the network characteristics. A special probe or some additional intelligence incorporated in a network node extracts relevant traffic information from the observed packet stream and reports the results. Hence passive measurement provides information about traffic already present in the network. Because of this non-intrusive property passive measurements are valuable for SLA validation and usage-based accounting.

### 1.3 Passive Multipoint Measurements

A broad spectrum of network measurement applications demand for passive multipoint measurements. Providers of interactive services, like audio and video conferencing, guarantee their customer certain delay limits which they need to validate. Multipoint measurements can also be an input for traffic engineering. Packets can be traced throughout the network which allows a detailed picture of the measured domain. Routing loops can be detected and network locations where packets get reordered or lost can be identified.

In Fig. ?? the general concept of passive multipoint measurements is shown. As a packet traverses through the measured network it passes observation points. These can be any device that can listen on the shared medium such as a network card or a router. At the observation point a copy of the packet is taken and a packet ID which identifies the packet throughout the network is generated. This can be either parts of the packet or a hash value over the packet content. The packet ID and a timestamp of the packet's arrival are transferred to a common multipoint collector. The collector can either be a dedicated device or can be co-located at an observation point. The trace of each packet and the delay between the observation points can be calculated by correlating the packet IDs from the different observation points. The delay is gained by subtracting the timestamp values of two corresponding packet IDs from two measurement

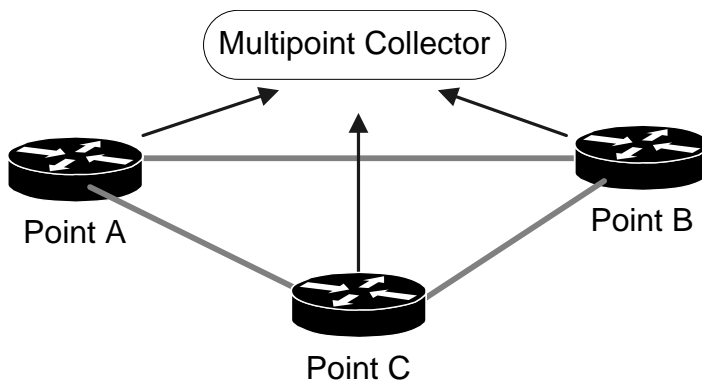


Figure 1: Multipoint Measurements

points. The arithmetic sign shows the direction of the packet.

## 1.4 Measurement Architecture and Measurement Process

In order to show a more detailed view of the measurement procedure, a partial measurement architecture and the measurement process as depicted in Fig. ?? shall be explained. The following representation is conform to the IETF drafts [?] [?] of the IETF Packet Sampling (PSAMP) working group. The measurement process comprises of packet capturing and a set of packet processing steps.

**Packet Capturing** records the packets including header information and packet data. The amount of data can be limited by a pre-configured snapsize.

**Time Stamping** is essential for multipoint delay measurements and needs to be processed as early as possible in the measurement process. For consistent time stamping between measurement points the internal clocks need to be synchronized by either one of the following technologies: network timing protocol (NTP) [?], global positioning system (GPS), radio signals or precision timing protocol (PTP) [?].

**Classification** is especially relevant for flow-based measurements. The packets are grouped according to identical properties (e.g. source address and destination address), where all packets join exactly one group (flow). Moreover stratified sampling is based on classification, whereby estimation accuracy can be improved by grouping the population [?].

**Selection** During the selection process a subset of the observed packet stream is chosen. This can be done by a deterministic filter on the packet content or by sampling which is packet content independent. Hence, the selection process either extracts relevant data or chooses a representative subset of the packet stream. In contrast to classification, a selector does not allocate each packet into a group, packets are either selected or dropped. Hash-based selection is such a selector.



Figure 2: Simplified Measurement Architecture and Measurement Process

**Aggregation** combines several information from multiple packets into an aggregated composition of values. Hence a summary of characteristics of the whole traffic and/or of each flow is exported. For multipoint measurements it is required to generate a packet ID during the aggregation step. The packet ID may be a hash value or parts of the packet content.

**Export of measurement data** The measurement data is transferred from an observation point to a collector for further post-processing. For example Netflow and sFlow are protocols that support the transmission of flow based measurement data. The PSAMP group recommends the IP Flow Information eXport (IPFIX) [?] protocol which became an IETF approved standard.

Whereas aggregation is mandatorily the last step of packet processing, classification and/or selection can be used multiple times in arbitrary order. Although it is advised to apply selection before classification as less data needs to be categorized, it may also be useful to apply classification first and sampling afterwards (e.g. stratified sampling). At the completion of a measurement interval the observation point exports the aggregated values to at least one collector. The measurement results can be transferred over a dedicated measurement network or the measured network itself. Transferring the measured data over the network itself entails similar problems as active measurements because additional load is introduced into the network. This may cause bias of the measured data. Nevertheless one can prevent bias by exporting results 1) at times when no measurement is taken (measurement-gap) or 2) over a network that is not measured.

## 1.5 Incentives for Sampling

There is a general problem about network measurements. More applications demand for more types of measurement data which the measurement infrastructure may not readily support. On the contrary, if measurement hardware is deployed a single observation point can produce an immense amount of data per hour. With the growing usage of the Internet and higher data rates, even more measurement information is created. This implies an expensive infrastructure of high-performance measurement nodes and collectors in order to cope with the amount of data. There are three constrains for the network measurement infrastructure:

1. *Processing capacities.* Higher data rates caused by high bandwidth access networks imply high processing capacities for network routers. Therefore only limited processing resources can be allocated for network measurements without influencing the normal network operation.
2. *Storing capacities.* More applications demand for flow-based measurement data which implies that flows need to be maintained in the measurement node. More elaborate measurements even ask for multiple metrics per flow which increases storing demands.

3. *Exporting capacities* At times where network load is high and more flows are present, the nodes export more measurement data which engraves bandwidth consumption. Especially if measurement results are transferred over the measured network this leads to distortions in the measurement results.

Multipoint measurements even demand for additional resources because:

- The packet ID generation (e.g. hash value calculation) and timestamping requires processing capacities at the measurement node.
- For every packet at least a packet ID and timestamp is created which need to be stored until the end of the measurement interval.
- More measurement data is exported because packet ID and timestamp cannot be aggregated and both values need to be exported for each packet.

Whenever the storage, processing and exporting resources for measurements are depleted, information will be lost. In case of buffer overflow the exported metrics may not be useful as they do not contain any flow information that occurred after the overflow. In case of bandwidth exhaustion the metrics may be exported with high delay which reduces their value if they are time critical. Service Providers therefore demand for easy to implement methods to reduce data and measurement infrastructure costs. **Packet Selection**, like sampling and filtering, provide less measurement costs by reducing the measured data. Instead of measuring all relevant data, only a subset is selected and the characteristics of the whole traffic are estimated. Service providers therefore can offer lower tariffs because of less measurement costs.

## 1.6 Selection Techniques

As pointed out in Section ?? selection is a part of the measurement process. In this packet processing step the selection of a subset is conducted, i.e. packets will be either selected or dropped. The variety of selection methods can be distinguished in **filtering** and **sampling**. A filter is a deterministic selector based solely on the packet content or a router state. A router state may be the ingress and egress interface of the packet, the origin or destination Autonomous System, the incidence that the packet violates an Access Control List or that the packet cannot be correctly routed. All other selectors that use other information

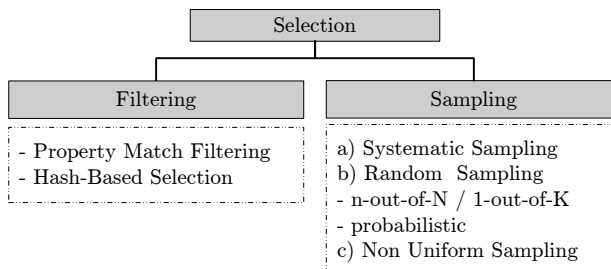


Figure 3: Packet Selection

than the packet content are considered to be sampling. A categorization of selection methods according to the PSAMP group is shown in Figure ??.

### 1.6.1 Sampling

In order to describe the different sampling techniques, the terms in the scope of sampling are presented as defined by the PSAMP working group.

<b>Population</b>	The population is a packet stream which is the base set from which packets are selected, e.g. the observed packet stream.
<b>Sample</b>	The Sample is the selected subset of the population.
<b>Population/Sample Size</b>	Amount of packets in the population / sample.
<b>Measurement Interval</b>	The measurement interval is the time or packet count-based length of a measurement.
<b>Sample fraction</b>	The sample fraction is the configured / attained ratio of sample size to population size.

#### 1.6.1.1 Systematic Sampling

Systematic Sampling is a deterministic selection at a time or count based period, i.e. either a packet is selected at the end of a time interval or every m-th packet in the observed stream is selected.

#### 1.6.1.2 Random Sampling

**n-out-of-N** A measurement interval is subdivided in blocks with the size of N packets. From each block a random subset of n packets is chosen. The attained and configured sampling fraction is therefore the same. A special case of n-out-of-N sampling is 1-out-of-K sampling.

**1-out-of-K** is a count-based stratification of n-out-of-N sampling. From each block consisting of K packets one packet is chosen randomly. 1-in-K sampling guarantees that each period of K, one packet is selected but in no fixed period.

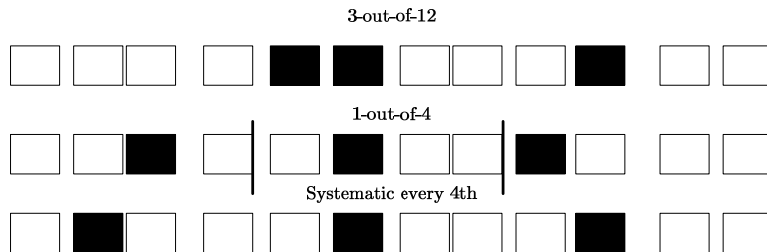


Figure 4: Sampling

**Probabilistic Sampling** Each packet in the population is selected independently with a predefined probability. This probability equals the configured sampling fraction.

**Non-Uniform Sampling** An enhanced sampling method is non-uniform random sampling, where first packets are classified and afterwards sampled with different probabilities depending on the category they are in. This is done because random sampling in IP networks has one major flaw. The estimation for small flows is very inaccurate. This is caused by the lack of packets from small flows, e.g. a flow that consists of 10 packets where each packet is selected with a probability of 10% has a chance of  $(0.9)^{10} = 35\%$  not to be even in the sample. The problem here is that actually a small number of packets introduce the most traffic on the link [?]. One can improve estimation accuracy by using non-uniform sampling.

## 1.6.2 Filtering

### 1.6.2.1 Property Match Filtering

Property match filtering selects a packet if specific fields within the packet and/or properties of the router state equal a predefined value [?]. Packet properties include header field values like netmask prefix of source and destination address or the used transport protocol. For instance Internet Service Providers may distribute IP addresses with the same netmask for customers that do not use flatrate services. In this way the Service Provider can easily filter flows that need to be measured for accounting. The selection can also be based on router states like ingress and egress interface of the router or the autonomous system where the packet is originated.

### 1.6.2.2 Hash-Based Selection

Hash-based selection is a filtering method because the selection decision is solely based on the packet content. In the researched papers the terms hash-based sampling, hash-based filtering and hash-based selection are optionally used and all refer to the same selection technique. In this paper the term hash-based selection is used because it complies with the categorization within the measurement process as defined by the PSAMP working group. Before the hash-based selection technique is explained an introduction to the terminology of hash functions and hash-based selection is given.

#### Hash Function

A hash function  $h$  maps an input value  $c$  of variable length into an output value of constant length  $h(c)$ .

#### Hash Input

The hash input  $c$  is the packet content on which the hash function is calculated. Assigned to this hash input value is only one corresponding output value. The PSAMP working group also uses the term hash domain in the scope of hash-based selection. In [?] and [?] the term hash key is used as a synonym for hash input.

#### Hash Value

The hash value is the output value of a hash function  $h(c)$ .

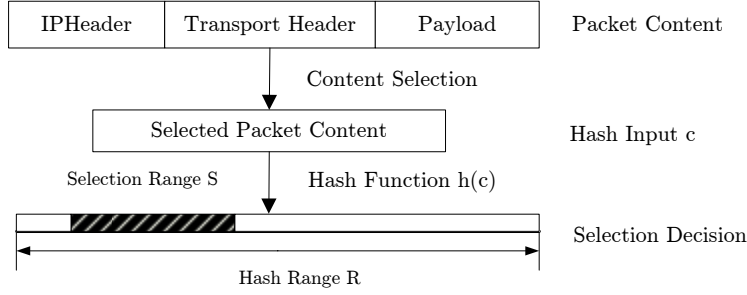


Figure 5: Hash-Based Packet Selection

<b>Digest Length</b>	The digest length represents the constant length in bits of the hash value.
<b>Hash Range</b>	A hash function of digest length $D$ maps all hash keys to an interval of $[0..2^D - 1]$ which is called the hash range.
<b>Hash Selection Range</b>	Is a subset of the hash range. Packets are selected if their hash values fall into this range. The elements in the selection range do not need to be contiguous successive values; they can be any elements of the hash range.
<b>Secret Key</b>	The secret key $k$ is only known by the measurement operator. It is appended or concatenated to the hash input $c$ before the hash value is calculated. This can also be a private parameter or an initial value for the hash function. In case an adversary gets to know the hash function and the selection range, this key prevents that an adversary can craft packages that are selected at his will. This provision of more security is explained in ??.
<b>Packet ID</b>	After the hash-based selection process, a packet ID for multipoint measurement is assigned to each packet. This might be again a hash value or parts of the packet content. The term packet ID and hash value has to be differentiated. Whereas the packet ID is required for multipoint measurement to correlate packets at the collector, the hash value for the hash-based selection technique is required to obtain a selection decision for a packet.

The technique of hash-based selection shall be explained with the use of Fig. ??. The content of an IP packet is structured; starting with the IP header, followed by the transport protocol header (e.g. TCP header) and further data which is labeled here as payload. For hash-based selection only parts of the header information and possibly some payload is cropped out from the rest of the packet. Optionally a not publicly known secret key is combined with the selected packet content to make up the hash input. The hash function  $h$  assigns to the hash input  $c$  a correspondent hash value  $h(c)$  by applying bit or bitwise operations on the input. The measurement operator priorly defines a selection range  $S \subset R$  which is part of the hash function's hash range. When the calculated hash value falls into this selection range the packet is selected. Provided that the hash input, hash function and selection range are equal at the different observation points,

this technique ensures that the selection decision for every packet is consistent throughout the network.

## 1.7 Comparison of Selection Techniques

The different selection techniques are compared in Table ?? in terms of resource consumption, security and applicability for multipoint measurements.

**Multipoint Measurement applicability** Systematic sampling is not suitable for multipoint measurements. One cannot assure to select the same packets at each observation point along the packets path because of packet reordering and packets that are coming along and leave the packet’s path. Random selection techniques are not applicable for multipoint measurements because packets are independently selected at each observation point. If the configured sampling fraction at each observation point is  $f$  ( $f < 1$ ) and one samples at  $n$  measurement points, the collector receives on average  $f^n$  traceable packets, i.e. packets which are sampled at all observation points. Property-match filtering and hash-based selection provide a consistent selection of packets because the selection decision is based on the packet content. Hence selected packets from different measurement points can be correlated.

**Bias** Random sampling acquires an unbiased estimation of the population’s characteristics, because packets are selected independently. The deterministic selection techniques bear the risk of bias because packets are not selected independently. For systematic sampling bias is introduced when packets with certain attributes occur in between the selection periods. Because the hash-based selection is based on the packet, the selection decision is suspect to bias. Depending on the traffic in the network, the underlying hash function and the configured hash input, the selection may prefer packets with certain attributes. Property match filtering is biased by definition, because packets with certain packet attributes are selected. Nevertheless there exist approaches to filter packets according to packet attributes that are uncorrelated to other packet attributes. For example Gon Jian [?] proposes a filtering based on the packet IP identification field. With non-uniform sampling packets are categorized according to their attributes

Selection method	Property Match Filtering	Hash-Based Filtering	Systematic	n-out-of-N	1-out-of-K	Probabilistic	Non-Uniform Random
<b>(R)andom/ (D)eterministic</b>	D	D	D	R	R	R	R
<b>Content Dependent</b>	Y	Y	N	N	N	N	Y
<b>Multipoint consistent</b>	Y	Y	N	N	N	N	N
<b>Risk of Bias</b>	Y	Y	Y	N	N	N	Y
<b>Selection Effort</b>	high	very high	low	medium	medium	medium	high
<b>Secure</b>	N	N	N	Y	Y	Y	Y

Table 1: Comparison of Sampling Techniques

and different selection probabilities are used for these categories, hence the selection decision is content dependent.

**Selection Effort** In terms of resource consumption systematic sampling is very lightweight because a counter with one random initial number or an internal clock is sufficient to derive the selection decision. Random sampling techniques require a random number generator or a stored pseudo random number table within each observation point. For the random sampling techniques the selection decision can already be made before the measurement interval, i.e. the random packet numbers that are going to be selected can be generated before the actual measurement. This is not possible for filtering methods because the selection decision is based on the packet content. Property match filtering requires the extraction of the filtering information from the packet. The packet content is compared with the filter expression. This method can be done at line speed, e.g. with a Berkeley packet filter (BPF). Compared to the other selection methods hash-based selection is very resource intensive. Hash-based selection involves the extraction of the hash input and a hash value calculation for each packet during the measurement.

**Security** In the scope of selection techniques the term security describes the possibilities of an adversary to intentionally influence the selection probability of packets. Random selection techniques are secure, because an adversary cannot influence the packet selection. For systematic sampling an adversary may generate malicious packets periodically and places them in the middle of the selection period. Property match filtering is very insecure, because the adversary can craft packets that are disproportionally selected very easily in case he knows the filter expression. The security of hash-based selection is discussed in [?]. Goldberg and Rexford prove that the security of hash-based selection is dependent on the underlying hash function and the use of a secret key. They prove that hash-based selection based on a linear hash function is insecure whereas an adversary cannot craft intentionally selected packets with the use of a keyed Pseudorandom Function (PRF). The security topic of hash-based selection will be more thoroughly assessed in Section ??.

## 1.8 Scenarios for Hash-Based Selection

Hash-based selection enables the correlation of packets which are selected at different measurements **without exporting the whole population**. One can follow a small subset of packets traversing through the network with only little measurement data. In case the packets are representative for the whole population, this subset can be used for estimating the behavior of the whole traffic.

### I. Traffic Engineering

**Routing Improvement** Because hash-based selection can trace a packet through the network, one can identify routing loops and routing inefficiencies on the packets path. This data can be used to improve routing policies.

Duffield, Grossglauser [?] and Niccolini et al. [?] show how paths of packets can be made traceable with the use of packet IDs.

**Network Topology Discovery** In case of a broad deployment of routers that are able of hash-based selection, it is possible to discover the network topology [?] without active measurements. One collector can gather information about where packets are traversing, with which delay and loss between each network node. Added with some more information like bandwidth utilization, the collector can draw a detailed picture of the network topology of the collector's domain. The ingress and egress points matrix can be made visible by looking for points where new packet IDs appear and disappear.

## II. QoS Monitoring

**Multipoint Metrics** Because packets can be correlated between two measurement points, the collector is able to calculate multipoint QoS metrics. Service providers can use hash-based selection to figure out if they comply with their Service Level Agreements in terms of one-way delay, one-way loss and derived metrics like delay variation. The measurement of one way delay with the use of hash-based selection is discussed in various papers [?] [?] [?].

## III. Attack/Intrusion Detection

**IP traceback** Network attacks are still a wide problem in the Internet. The nature of the IP protocol makes it difficult to identify the true source of malicious packets if the source wants to conceal it. An attacker can generate packets with spoofed addresses (i.e. forging another IP address) and hide his localization. In case hash-based selection is widely deployed the multipoint collector can trace *selected* packets back to their origin by simply following the packet IDs path [?] [?]. In case of DoS attacks where multiple packets are sent this is already sufficient.

Another applicable scenario is an adversary using a proxy to hide his network address. The adversary sends a packet to a proxy which changes the source network address in the packet and replaces it with its own. In case hash-based selection is deployed and the hash input does not comprise the source network address, the packet is still traceable beyond the proxy.

# 1.9 Problem Statement and Target of this Work

## 1.9.1 Problem Statement

Hash-based selection is a **deterministic selection** of packets and thus it is susceptible to a biased selection decision. The selected subset may include a higher fraction of packets with certain attributes than there is in the population. This biased selection decision concludes to a biased estimation of traffic characteristics. In case packets with e.g. certain lengths are preferably selected it is impossible to give a sound estimation of the packet length distribution.

## 1.9.2 Targets of this Work

### 1.9.2.1 Emulation of Random Sampling

Random Sampling provides features which are desirable for network measurements. Random sampling selects packets independently which excludes bias and ensures the selection of a representative subset. Based on the representative subset the traffic characteristics can be estimated with predictable accuracy. In case hash-based selection can emulate random sampling, it can be modeled as random probabilistic sampling for which already mathematical models for bias and accuracy are available [?].

The main target of the work is to evaluate with **which configuration** hash-based selection can emulate random sampling in terms of unbiasedness and representativeness of the selected subset. There are two factors in the hash-based selection technique that have an influence on the selection decision quality:

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

### 1.9.2.2 Packet Fields for Hash-Based Selection

The choice of the packet content (header fields and payload) to be used for the hash input is of utter importance, because this decision can introduce bias. This is shown by the example in Fig. ???. This figure shows two different packets 1 and 2 (only the IP Header) which have similar packet contents. It is assumed that only the first 8 bytes of the packet are used as the hash input. Since these fields are equal for both packets, the same hash input  $c$  is formed. Thus both have the same hash value  $h(c)$  and the same selection decision. Let's assume there are more packets like Packet 1 and 2 that are equal in the most significant 8 bytes, all having the length of 276. They all map to the same hash value and either all or none of them are selected. This means that packets with 276 bytes length are either over- or underrepresented in the sampled subset.

This example shows that the selected bytes from the packet content should be sufficient to identify unique packets in order to avoid **input collisions**, i.e. packets with the same hash input. Especially for flow-based sampling where packets are already categorized according to common properties one has to ensure that packets within a flow can be distinguished. As a restraint it has to be clear that it is not suitable to use the whole packet content as hash input because routers

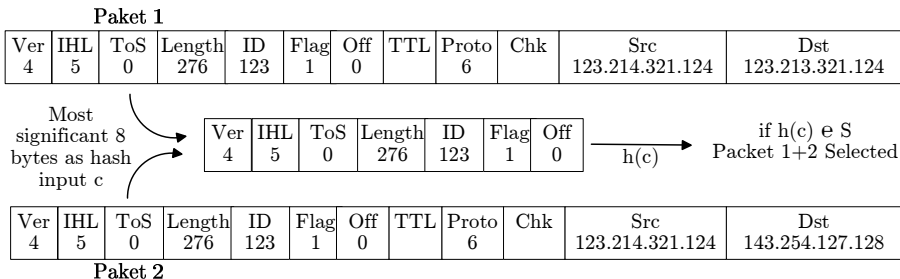


Figure 6: Input Collision of 2 Different Packets

that serve as measurement points have only a limited capacity for processing packets and measurements. The hash function has to be applied for each packet and therefore the amount of bytes that is used for hashing should be kept at a minimum, because this will reduce the calculation effort.

One target of the thesis is to find a hash input configuration that includes low input collisions with minimal input length.

### 1.9.2.3 Hash Functions for Hash-Based Selection

The choice of the hash function is of crucial importance to avoid bias. Hash functions of poor quality do not well distribute the hash input over the hash range. Hash inputs which are similar will be mapped closely together, therefore packets that are similar will have a high probability to possess the same selection decision. This implies bias, because similar packets are not independently selected.

Moreover the hash function should be secure to prevent adversaries to influence the selection decision. Contradictory to the security is that the hash function also has to be fast, because the hash value has to be calculated on every packet that is passing the observation point.

In this thesis one target is to find a hash function that is suitable for hash-based selection and is able to select an unbiased sample based on the hash input.

### 1.9.2.4 Packet ID Generation

An adjacent topic to hash-based selection is packet ID generation. Each selected packet will be assigned a packet ID and a timestamp which are exported to the multipoint collector as described in Sec.???. This packet ID is not necessarily the hash-value that is obtained by hash-based selection, because the packet ID requires different properties than the hash-based selection hash value. In particular the packet ID requires low collision probability. Hence, related work [?] [?] [?] proposes to use two different hash functions for packet ID generation and hash-based selection. A target of this work is to find hash functions and hash input configurations that are applicable for packet ID generation. Further it is evaluated under which scenarios one may use one hash value for packet ID generation and hash-based selection.

## 1.10 Document Structure

The document is structured as follows:

In Chapter 2 existing literature in the scope of hash-based selection is presented. Further this chapter shows existing approaches to evaluate the hash input and hash function for hash-based selection. Chapter 3 is dedicated to the evaluation of IPv4 and IPv6 packet content for hash input. First the investigated traces will be presented, followed by the empirical analysis. The results of the analysis will give suggestions which parts of the packet content shall be used as hash input.

In Chapter 4 quality criteria for hash functions will be defined. After some considerations about security issues, a collection of 25 hash functions will be evaluated according to these quality criteria. Each hash function will use the recommended

combination of header fields from chapter 3 as hash input.

In Chapter 5 the applicability of hash-based selection for different traffic classes will be investigated. The influence on the selection decision and reasons of duplicate packets is discussed.

In Chapter 6 it is analyzed which hash functions are suitable for packet ID generation and if hash-based selected packets need an extra packet ID hash value.

The concluding Chapter 7 will summarize the results. Still open problems will be presented for future research.



# Chapter 2

## State of Art

At first relevant work on multipoint measurements which is in the scope of hash-based selection will be presented. Later relevant work for hash-based selection is analyzed according to the targets of the work: 1) which packet content to use and 2) how to evaluate hash functions for hash-based selection and packet ID generation.

### 2.1 Passive Multipoint Measurements

Passive multipoint measurement techniques have been proposed for packet tracing and one-way delay measurements in [?] [?] [?][?]. In contrast to active measurements, passive measured packets are not marked but their occurrences recorded at the measurement point. For this purpose a packet ID is generated based on the packet content. In case different packets have the same packet ID in the collector domain, the collector needs to identify which packet ID belongs to which packet. In [?] Duffield and Grossglauser show which trajectories of colliding packet IDs are ambiguous and which can be disambiguated. In Fig.?? two exemplary trajectories of two packets with the same packet ID originating at the ingress nodes A and B are depicted. Whereas in example (a) the trajectories of both packets can be fully reconstructed, there are multiple possible packet traces in example (b). Although it is possible to disambiguate the trajectories in

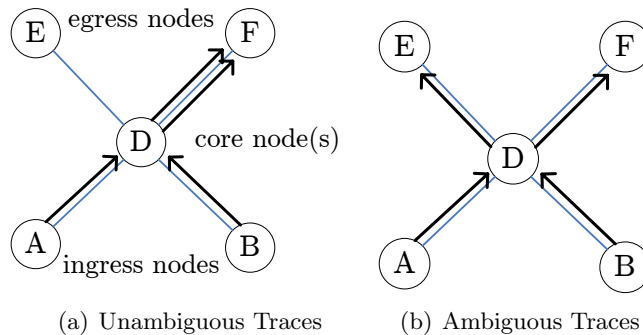


Figure 7: Trace Disambiguation of Two Packets with Same Packet ID

example (a) it is not possible to measure the delay between the nodes, because one cannot resolve in which order the packets originating at A and B arrive at C and E.

Again Duffield [?] [?] and Niccolini et al. [?] show algorithms to differentiate packets according to their timestamp. They argue that the only place to identify packet ID collisions is at the measurement domains ingress nodes, i.e. nodes where traffic is introduced into the network. If a packet ID is observed more than once at any ingress node within a given time window than all observations of this packet ID are excluded from delay measurements. The time window has the size of the measurement domains maximum delay  $t_{max}$ , starting at the first occurrence of the packet ID at an ingress node. In order to compensate for the eliminated packet samples, the sampling fraction is adjusted by the packet ID collision probability.

In [?] Duffield and Grossglauser further propose the use of a Bloom filter to check whether a packet ID is already present in an unreliable network. A Bloom filter consists of a bit array of size  $M$  and a set of  $v$  hash functions with codomains  $[1..M]$ . Starting off with an empty Bloom filter where all bits of the array are set to zero, the packet IDs is used as the hash input for each of the  $v$  hash functions. The  $v$  hash values are then coded into the bit array by setting each bit that corresponds to the hash values to one. If the same packet ID arrives again, than the bits for this packet ID are already set within the bit array and all observations of this packet ID will not be used for delay measurements. Although all duplicate packet IDs will be identified with this technique, there is a chance of false positives, i.e. packet IDs that have not been observed yet but are identified as duplicates, because the bits in the array were already set by other packet IDs. Duffield and Grossglauser [?] propose to compensate for these accidental discarded packets by adjusting the sample size.

An approach by Eriksson et al. [?] enables the reduction of the observation points by deducing the network topology based on the IP time to live value. He proposes to send additional the TTL together with the packet ID to the collector. With this approach it is for example possible to trace a packet in Fig. ?? traversing from Point A to B to C without measuring at point B, because these packets will have a reduced TTL of 2, whereas packets directly traversing from A to B only have a reduced TTL of 1.

Raspell et al. [?] proposes a method to configure passive measurements using signaling based on the NSIS Transport Layer Protocol (NTLP). NTLP packets are send from a source to a destination configuring each visiting router. In this way measurements for a specific flow can be configured, e.g. an adjustment of the selection range or defining the start of the measurement interval.

An IP traceback system as proposed by Snoeren et al. [?] [?] uses a similar technique to hash-based selection in order to trace single malicious packets to their destination by storing hash-values over invariant packet content in Bloom filters. Wang et. al [?] presents a heuristic method for traceback systems that derives the amount of required routers to identify an attack within a predefined hop-count distance.

## 2.2 Packet Header Fields useful for Hash-Based Selection

Duffield et al. [?] is the first to present the hash-based selection method for tracing packets through the network. In order to distinguish packets throughout the network, he declares that those packet fields should be used for hash-based selection and packet ID generation that are

1. **constant between nodes, but**

2. **variable between packets.**

He divides the IPv4 header fields into three categories 1) variable, 2) low entropy and 3) high entropy fields. He identifies the TTL, type of service and header checksum as variable fields which cannot be used for hash-based selection, because they change between network nodes. Low entropy fields are version, header length and protocol. High entropy fields are source and destination address, identification, fragment offset, flags and total length. The low and high entropy fields are invariant between network hops and therefore can be used for hash-based selection.

In order to find out the amount of bytes that should be used for hashing, Duffield identifies collisions between packets when variable fields are masked out. Collisions are here defined as different packets that have the same hash input value, because not the whole packet is used as hash input. He uses traces of packets that are recorded at a campus network LAN interface. His results show that the input collisions drop the more bytes are used. Increasing the hash input length beyond 40 consecutive bytes of the packet does not reduce the collisions any further.

Molina [?] uses the same approach to test for collisions but after applying the hash function. He evaluates how many hash values collide depending on the input length. With increasing hash input length the collision rate drops until a certain length after which there is no further significant decrease.

Snoeren [Sn01] uses the hash-based approach for IP traceback systems and identifies unique packets by considering different input length. All three authors (Duffield, Molina, Snoeren) conclude that an input length between 28 and 40 consecutive bytes (without variable fields) is sufficient for hash-based purposes.

Niccolini et. al [?] uses the same header fields like Duffield and Molina except the fragment offset and flags fields because they can change at network hops. Additionally he uses an arbitrary amount of 12 byte long blocks of the transport header.

The PSAMP [?] working group advises the use of IPv4 identification field, flags field, fragment offset, source IP address, destination IP address and a configurable number of bytes from the IP payload, starting at a configurable offset. Duffield, Snoeren and Molina only consider general assumptions about header fields that are suitable for hash-based selection but lack the basis of an empirical and bitwise analysis.

The entropy-based approach used in this work enables the ordering of header bytes according to their suitability for hash-based selection. This systematic approach is used in order to find a smaller subset of bytes that show similar or

better results than the recommended PSAMP bytes.

Entropy-based measurements have already been proposed in the field of selection techniques by Gong Jian [?]. He analyzes the entropy of IPv4 Header fields per bit and byte on a CERNET backbone link. The analysis includes the first 20 bytes of the IPv4 header of 10 million different packets. He shows that there are major differences of entropy and that only the identification field has an entropy efficiency of close to 1. He proposes to sample according to the IP Identification field because it is variable and does not change during measurement points. Furthermore entropy based measurements have been proposed for network anomaly and worm detection in IP networks [?] [?]. In case of network attacks the distribution of packet characteristics change which can be detected by entropy measurements. Usually the entropy of e.g. address bytes drops because the attack traffic is less variable in those bytes than the normal traffic.

## 2.3 Existing Evaluation Approaches for Hash Functions

A general performance comparison of cryptographic hash functions can be found at [?] [?]. The MD5 and SHA-1 algorithm are well known and show comparably good calculation time. Nevertheless cryptographic hash functions are generally very computation expensive. Mulvey [?] and Jenkins [?] show approaches to evaluate hash functions on their randomness and uniformity. They use the Avalanche criterion and chi-square uniformity tests to evaluate hash functions for general purposes.

**Evaluation of Hash Functions for Hash-Based Selection** Molina [?] analyzes four hash functions (CRC32, MMH, IPSX, BOB) on their suitability for hash-based selection and packet ID generation. As criteria for the suitability for hash-based selection he uses performance and the ability to uniformly distribute the hash values. He argues that the uniformity of hash values is a quality criteria for hash-based selection as the selection range can be easily configured to gain a certain sample size. Further Molina analyzes the collision probability of the hash values to evaluate their applicability for packet ID generation. He uses synthetically generated packets and real packet traces for the evaluation. The BOB hash function performs slightly better than the 2nd placed CRC32 function for hash-based selection and packet ID generation.

Zseby [?] compares 6 hash functions for packet ID generation (40 bytes of unprocessed fields, CRC-16, a 16-bit hash function, CRC-32, a folded 32bit MD5 and MD5-128bit) on 1) the size of the resulting ID, 2) probability of collision 3) processing time and 4) effort to encode additional information. The measurement results are based on a modular meter implementation. The unprocessed fields do not require any processing and include minimal collisions, but imply maximum measurement traffic. The 16 bit hash values include large amounts of collisions and the calculation time is not significantly less than for CRC 32bit hash function. CRC-32, folded 32bit MD5 and 128 bit MD5 provide low collision probability for artificial packets and for a real RTP-Flow.

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. He investigates dependence between single bits and sampling decision. There has been no statistical sign for dependence if 40 input bytes were used. Furthermore he evaluates if hash-based sampling can genuinely estimate fractions of packets with common network addresses. He reasons that *good hash functions can "randomize" sampling decisions such that the set of sampled packets are representative.*

Niccolini et. al [?] and Raspell [?] use the MMH hash function for hash-based selection and packet ID generation, because of its calculation speed and uniform hash value distribution.

The PSAMP[?] working group advises that the BOB hash function should be used for hash-based selection whereas IPSX and CRC may be used.

Molina analyzes only 4 hash functions solely on their performance and uniform hash value distribution as criteria for hash-based selection. Nevertheless, one can adjust the selection range according to any hash value distribution in order to obtain a certain sample size. In this work it is reasoned that unbiasedness and representativeness of the selected subset are more important for accurate measurements than the uniformity of hash value distribution. Duffield evaluates only one linear hash function on the independence of sampling decision and one packet attribute (network address).

In this work a collection of 25 hash functions will be analyzed on the independence of multiple packet attributes and the sampling decision. Further, the chi-square goodness-fit test is used to evaluate if hash-based selection is able to produce a representative subset of the observed packet stream and hence can emulate random sampling.



# Chapter 3

## Evaluation of Packet Content

### 3.1 Approach

#### 3.1.1 Applicable Header Fields

The IP and transport header fields upon which the hash-based selection decision is based require two properties. The header fields need to be

1. **static between network nodes.** The header fields need to be static because the sampling decision is required to be consistent throughout the network. If variable header fields are used as hash input the sampling decision based on these fields will differ from measurement point to measurement point. Header fields that are not static between network nodes are: Time to Live and IP Checksum. In case of packet fragmentation on the packets path, the packet length, fragmentation flags and fragment offset fields may change. Nevertheless it is here assumed that most fragmentation issues are negotiated between the end nodes and that fragmentation in the network is rare. Hence packet length, fragmentation offset and fragmentation flags are considered to be static.
2. **variable among packets.** In order to prevent bias caused by input collisions it is favorable that **the hash input of every unique packet is unique as well** (as shown in ??). The higher the variability in one header field the higher the probability that two packets differ in this field.

In this section the **entropy per byte** is used to evaluate the variability of each header byte. The higher the entropy in a byte, the higher the probability that packets differ in this byte and hence the byte is more suitable for hash-based selection. In the second part of the evaluation it is analyzed which influence the choice of high entropy bytes has on the probability of **hash input collisions**.

#### 3.1.2 Entropy per Byte

The entropy per byte [?] is defined as:

$$H(B) = - \sum_{i=0}^{255} p_i \log p_i \quad (3.1)$$

Where  $B$  is the discrete variant of byte values;  $p_i$  is the probability that the byte value  $i$  occurs. The maximum entropy denotes the state that all possible byte values have the same probability of occurrence. For scaling purposes the byte entropy is divided by its maximum value  $H_{max}$  to obtain the information efficiency  $E$  [?].

$$H_{max}(B) = - \sum_{i=0}^{255} \frac{1}{256} \log_2 \frac{1}{256} = 8$$

$$E = \frac{H(B)}{H_{max}(B)}$$

The information efficiency  $E$  measures the byte randomness. An information efficiency of 0 denotes no randomness (i.e only one value occurs) and a value of 1 denotes maximum entropy.

### 3.2 Traces Analyzed

Trace Name	Location	Duration	IP Snapsize	packets in millions	IP Address anonymous	IP Version	AS Class
NZIX	New Zealand	30 hours	40	~200	Yes	4	5
Ciril	France	6 hours	20	~100	No	4	1
FH Salzburg	Austria	3 days	40	~110	Yes	4	4
LEO 1		3 hours	140	~200	No	4	1
LEO 2		6 min	140	~12	No	4	1
Twente	Netherlands	10 days	40	~360	Yes	4	2
Mawi	Japan	40 days	60	~80	Yes	6	5

Table 2: Traces Used for Evaluation

When performing experiments which involve packet content it is inevitable to use real traffic traces. Since traffic differs between locations in the Internet (backbone, edge-routers, campus, hosts), geographical locations and time, one can only use many different traces in order to reflect possible scenarios.

The evaluations will be based on a set of 7 trace groups which are recorded at different observation points. All traces can be accessed via the MOME database [?]. The NZIX traces [?] were recorded by the WAND research group at a peering point among a number of major ISP's in New Zealand. FH Salzburg traces are captured at an WAN Access Network on a student campus. The Twente traces are captured at in the netherlands at an aggregated uplink of an ADSL access network with a couple of hundred ADSL customers, mostly student dorms. There are two trace groups that were recorded by a large european telecommunication operator and are noted as LEO1 and LEO2. The MAWI traces [?][?] are the only accessible IPv6 traces. The traces were captured at two samplepoints at lines connected to the 6Bone and WIDE-6Bone. The 6Bone is located on a FastEthernet segment connected to an IPv6 internet exchange point in Tokyo. All but the Ciril and IPv6 MAWI traces include parts of the transport header (max. 20 bytes).

**Anonymization** Most traffic traces are anonymized in order to protect data privacy. This anonymization implies that the IP address fields (source and destination) are irreversibly scrambled. The IP addresses of the Twente and FH Salzburg traces are anonymized with the Tcprpriv "-A50" option. This option ensures that within a trace two original addresses which equal in the most significant n bytes will have scrambled addresses that are equal in these n bytes as well. For example, the two source addresses a.b.c.1 and a.b.c.7 will be mapped to x.y.z.n and x.y.z.k. For the other anonymized traces (NZIX and MAWI) the mode is not known. The LEO 1 and LEO 2 traces are not anonymized.

**Classes of Autonomous Systems** According to [?] Autonomous Systems can be categorized in six classes.

1. Large ISPs: Large intercontinental backbone providers.
2. Small ISPs: Regional and access providers with small metropolitan or larger regional networks.
3. Customer ASes: Companies or organizations that run their own networks but do not provide Internet access.
4. Universities: University or college networks. These networks are distinguished from members of the Customer AS class, since they typically have substantially larger networks that serve thousands of end hosts.
5. Internet exchange points (IXPs): Small networks serving as interconnection points for ISPs.
6. Network information centers (NICs): Networks hosting important network infrastructure, such as root or top level domain servers.

This categorization in classes of autonomous systems is applied to the measurement points of the available traces in Table ???. The LEO1, LEO2 and Ciril traces were captured in a large ISP network but the measurement point location is unknown. The FH Salzburg traces were captured in an University network at a WAN access point. The Twente traces are not assigned in the class of University networks, because the traffic is captured at an ADSL uplink that provides Internet Access. The NZIX and MAWI traces were captured at peering points between a number of ISPs in New Zealand and Japan.

Although each trace is assigned to a specific AS class it cannot be assured that the presented traces are representative for this AS type. The traffic can differ depending on the geographic and network location and capturing time.

**Protocols and Applications** Table ?? shows that each trace group represents a different traffic scenario indicated by their different transport protocol mix and applications. Applications are assessed by the transport header port addresses. Because Ciril and MAWI traces do not include transport headers their protocol mix could not be evaluated. The FHSalzburg traces consist to 90% of http traffic. The LEO1 traffic is composed of 25% edonkey traffic and 5% http traffic; other protocols do not make up more than 1% of the traffic. The traffic captured in

	TCP	UDP	ICMP	others	Main Applications
FHSalzburg	99	1	0	0	http(90)
LEO 1	90	10	0	0	edonkey(25) http(5)
LEO 2	33	60	0	7	tunnel(60) edonkey(10)
NZIX	68	20	9	3	http(50) quake(5)
Twente	89	7	1	3	diversified

Table 3: Traces Protocols

the trace group LEO2 consists to 60% of the layer 2 tunnel protocol over UDP and 10% of edonkey traffic. The Twente traces include a variety of applications with only 2% http traffic and applications each with at most 1% of the traffic. The NZIX traces consists of about 50% http and 5% Quake packets.

### 3.3 Entropy Evaluation

The entropy per byte is calculated with a tool written in C++ which uses the libpcap library to read packet traces. The tool records the frequencies of each byte value and calculates the entropy.

#### 3.3.1 IPv4

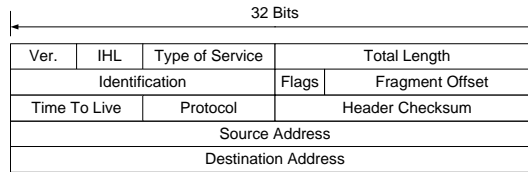


Figure 8: IPv4 Header

All six IPv4 traces were used for measuring the entropy of the IP Header (cf. Fig. ??). The IP source and destination address fields from Twente, NZIX, FHSalzburg traces are anonymized and therefore do not reveal the real entropy. The Twente and FHSalzburg traces show the real entropy in the most significant byte, because they were encoded using the tcpdpriv "-A 50" option. Figure ?? illustrates the results. The diagram does not include the Time To Live and Checksum field as these are variable from hop-to-hop.

Version, IP Header Length and the least significant byte (LSB) of Fragment Offset show an entropy very close to zero and are unsuitable for hash-based selection. The identification (ID) field has the highest entropy which shows that the values are randomly distributed. But the identification field bears a problem; many packets include an IP ID of 0 caused by differences in IP ID handling of operating systems.

Since there exist many small packets with a length below 255 bytes, the entropy for the most significant byte (MSB) of the length field is comparably low (about 0.2 information efficiency). The figure shows that there is an increase of entropy

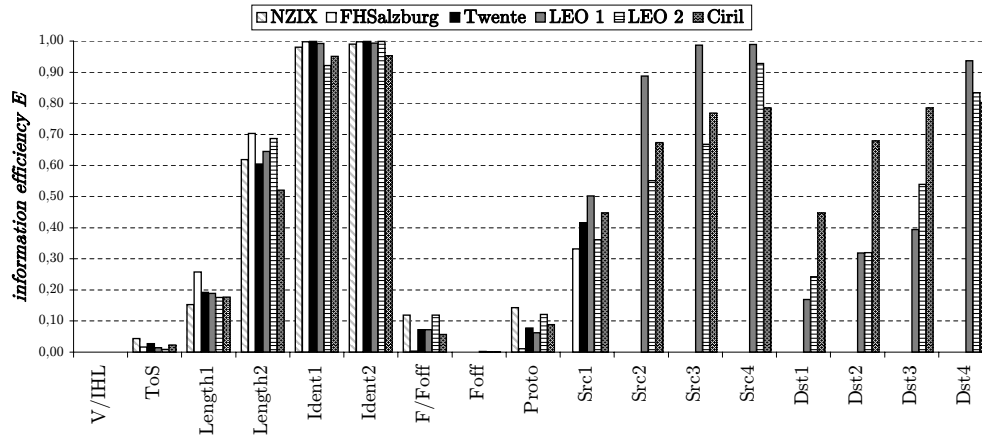


Figure 9: Information Efficiency IPv4

from MSB to LSB of the IP source and destination addresses. Nevertheless, it is expected that the information efficiency of all 8 address bytes is less than all its components, because there is a strong correlation between the address bytes. The highest information efficiencies are found in these bytes: **Identification**, **Source Address (two LSB)**, **Destination Address (two LSB)**, **Length(LSB)**.

### 3.3.2 TCP

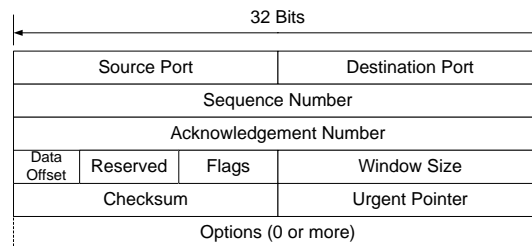


Figure 10: TCP Header

For analyzing the TCP header fields on their entropy, the Ciril and MAWI traces had to be omitted, because they do not include any transport header data. The TCP Header (Fig. ??) follows the IP header and comprises of at least 20 bytes depending if Options are present. All header fields are applicable for hash-based selection because they are invariant between network nodes. The urgent pointer field is never used in the analyzed traces and has an entropy of 0 and is therefore not depicted in Figure ?. Only the two bytes comprising offset, reserved and control bits include low entropy. Other header fields show high entropy. Because the FH Salzburg traces comprise mostly of http traffic, the source port entropy is significantly lower compared to other traces. The most suitable fields for hash-based selection with an information efficiency close to 1 are: Checksum, Sequence Number and Acknowledgment Number.

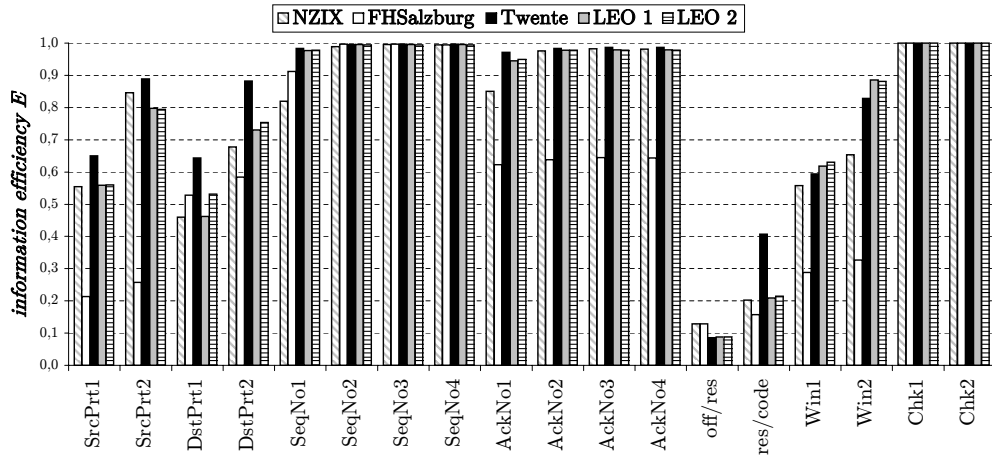


Figure 11: Information Efficiency TCP

### 3.3.3 UDP

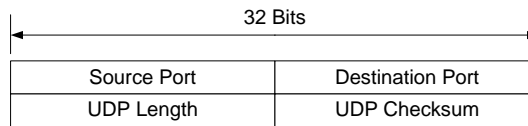


Figure 12: UDP Header

The UDP Header consists of 8 bytes which are all invariant between network nodes. Figure ?? illustrates the results of the entropy analysis. The LSB of the port addresses show higher entropy than the MSB. For the FH Salzburg traces which includes mostly http traffic, the entropy of the port addresses is lower than for the other traces. Most UDP packets are DNS requests emerging through the usage of http.

In the LEO2 traces the checksum is not used and set to zero for most packets.

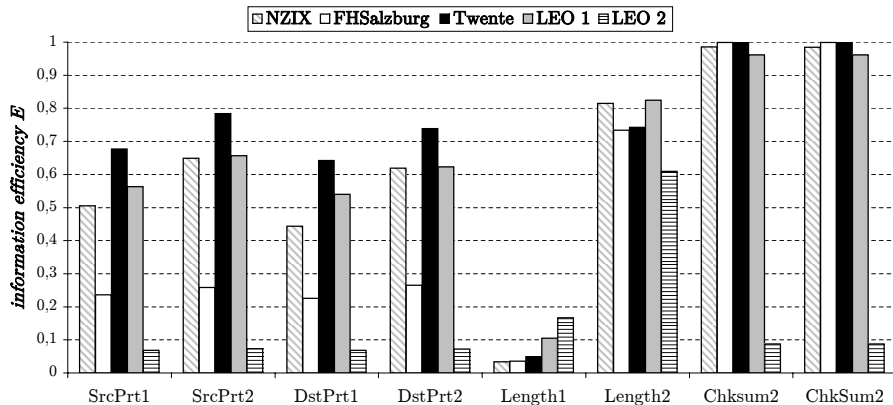


Figure 13: Information Efficiency UDP

Most of the packets with no UDP checksum include the Layer 2 Tunnel Protocol with IP and TCP in its payload. Packets with the Tunnel Protocol include the same source and destination port numbers 1701 which causes the low entropy for the LEO2 traces in port addresses as well. The Layer 2 Tunnel Protocol is applied in virtual private networks (VPN). Although the Checksum is optional it is still advisable to use the **UDP Checksum field** for hash-based packet selection as it includes very high entropy. Other fields with high entropy are: **SrcPort (LSB)**, **DstPort (LSB)**, **Length (LSB)**.

### 3.3.4 ICMP

The Internet Control Message Protocol (ICMP) differs in length and content depending on the type of message. All ICMP packets have four bytes in common: 1 Byte Type, 1 Byte Code, 2 Bytes Checksum. Following bytes can include other internet packets, an identifier, a sequence number and/or timestamps. Since there are only few ICMP packets in the analyzed traces it is refrained to distinguish each ICMP message type to analyze the entropy. Figure ?? illustrates the results. The entropy is calculated without associating any specific field to the value, hence Byte 5 denotes the fifth byte in the header. The FHSalzburg traces did not include a sufficient amount of ICMP packets, therefore the entropy is very variable and is not shown in the diagram. The following bytes have the highest information efficiency and are most suitable for hash-based selection: **Checksum, Byte 12-13, Byte 18-19**.

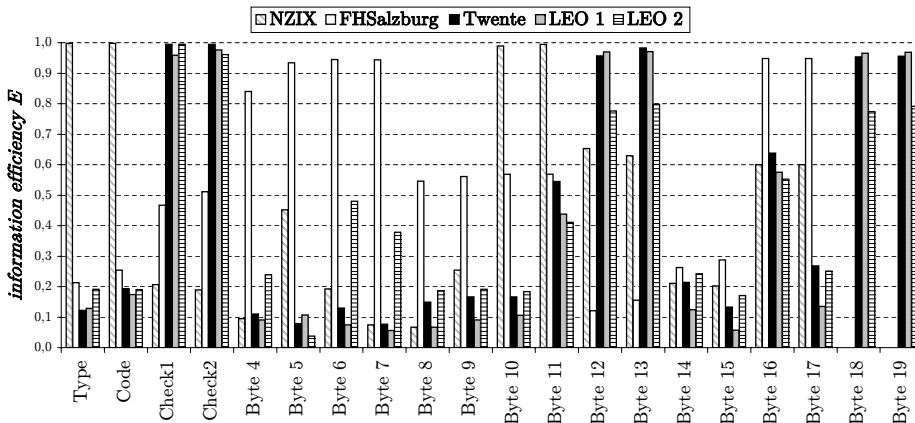


Figure 14: Information Efficiency ICMP

### 3.3.5 IPv6

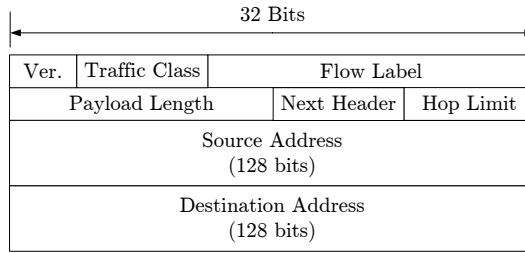


Figure 15: IPv6 Header

The MAWI traces are used to evaluate the variability of fields in the IPv6 header (cf. Fig. ??). The MAWI traces are captured at 2 different measurement points C(6Bone) and D(WIDE\_6Bone). Both sampling points are evaluated separately. The IPv6 Header is at least 40 bytes long. IPv6 allows to chain extension headers by using the next header field. Only the first header is considered for evaluation. The Hop Limit field is similar to the Time To Live field of IPv4 and is decremented each network hop and therefore cannot be used for hash-based selection. Since the source and destination addresses are anonymized, the entropy of the original fields cannot be recovered from the original addresses. The results for the remaining 7 bytes are shown in Fig. ??.

It is obvious that there is only very low variability in those bytes. The Version field (first 4 bits) is static because only IPv6 packets are considered. The only byte that includes some more entropy is the LSB of the Length field which might be useful for hash-based selection purposes.

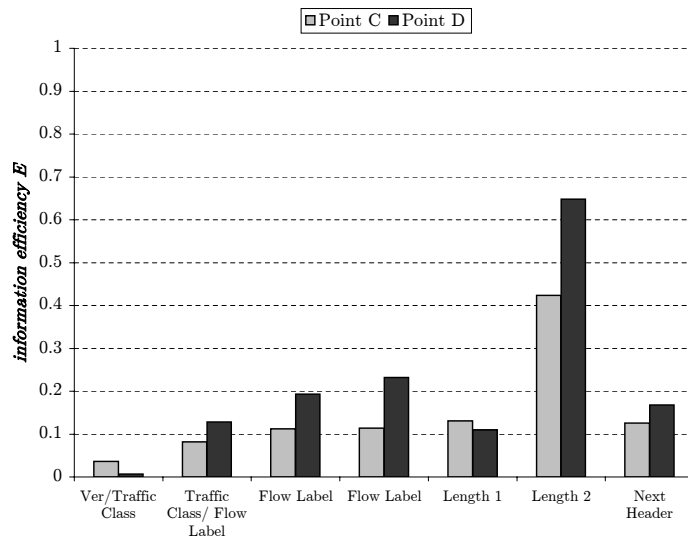


Figure 16: Information Efficiency IPv6

IPv4	IPv6	TCP	UDP	ICMP
Identification	Length LSB	Checksum	Checksum	Checksum
Destin. 2LSB		Sequence No	Length LSB	Byte 12-13
Source 2LSB		Acknowledg. No	Source Port LSB	Byte 18-19
Length LSB			Destin. Port LSB	

LSB - Least Significant Byte

Table 4: Bytes Recommended for Hash-Based Selection

### 3.3.6 Conclusion Entropy Analysis

The entropy of the packet varies from byte to byte. For certain bytes only very few values are used (e.g. protocol) or the distribution of the values is concentrated around few values (e.g. MSB of IP length). The entropy of packet bytes is strongly dependent on the applications within the traces. The LEO2 traces that have a comparably low entropy caused by tunneled packets. There are certain bytes in the IP and transport layer header that are more suitable for hash-based selection than others. The amount of bytes used for hashing should be kept at a minimum because every additional byte increases the hash calculation and processing time at the measurement node which may cause buffer overflow. The bytes with the highest entropy are most suitable for hash-based selection as there is a higher probability that packets differ in high variable bytes. A summary of fields with high entropy is shown in Table ???. Although there are some high variable fields in the IPv4 header it is shown that it is not sufficient to use Identification, Length, Source and Destination Address. In the analyzed traces there are packets whose ID field is often zero with the same source and destination address. This is caused by different handling of the identification field in operating systems. Consequently one has to include more variability by using transport header fields as well. Especially the TCP header includes many high entropy bytes, whereas UDP lacks some. It has to be assessed if the variability for UDP is sufficient or if some payload bytes should additionally be used. The ICMP packets are usually less frequent in a packet stream. Because of the similarity and the short length ICMP packets easily collide in the hash input if not enough bytes are used. As ICMP and UDP packets are strongly similar in their headers one may even consider to use an additional amount of bytes for the different transport protocols in order to enable a better differentiation of packets. These bytes may consist of further bytes of the headers or payload.

**IPv6 packets** The IPv6 address fields could not be evaluated because of anonymization of the MAWI traces. Nevertheless it is insufficient to use only the 40 byte IPv6 Header because at one observation point many packets will include the same destination and source address. Other bytes do not show high variability and cannot help to distinguish all packets. Hence it is required to use additional bytes from the transport header. In future, when IPv6 traces with transport header data become available it has to be evaluated if the transport headers used with IPv6 show similar entropy as the IPv4 transport headers.

## 3.4 Hash Input Collisions Evaluation

### 3.4.1 Input Collisions

An input collision consists of multiple packets that provide the same hash input. As pointed out in Section ??, packets with the same hash input introduce bias to the hash-based selection. All packets within an input collision have the same sampling decision and therefore the packets are either over- or underrepresented in the sampled subset. There are two reasons why packets hash inputs' collide: 1) the colliding packets are identical or 2) the packets are not all identical but the selected bytes for hash input are equal. Input collisions of non identical packets can be reduced by using some more bytes as hash input, but to deal with identical packets is difficult. In order to cope with identical packets that occur randomly, it is recommended to change the selection range periodically and synchronically at all observation points. In this way it is ensured that packets with large interarrival times have different selection decisions and bias can be reduced.

**Identical Packets** While analyzing the packet traces it became obvious that there is a high amount of identical packets that occur at an observation point. Here it is defined that identical packets are packets that are identical for the whole captured snaplength including the IP and transport header except the IP Time to Live and IP Checksum field. The IP Time To Live field and IP Checksum are excluded because identical packets can take different paths to the destination or may take detours in routing loops.

Of course it is not sure that for equal packet headers the packets are all identical. Nevertheless, the UPD, TCP and ICMP checksum is calculated over the whole packet content and hence packets with equal IP and transport header are likely to be identical. For TCP packets with equal acknowledgment and sequence number an identity is even more probable. In the presented evaluation all packets with equal headers (IP and transport) are labeled to be identical, because there is not more data available to distinguish the packets for traces where the payload is missing. This is done despite the awareness that these identical labeled packets may not be identical (e.g. for UDP packets that do not use the optional UDP checksum).

### 3.4.2 Experimental Setup

A tool written in C++ cuts a combination of fields from all packet and matches double values. As a result, one obtains all input collisions with the corresponding number of occurrences.

In order to measure the amount of collisions caused by identical packets, the collisions that appear for the whole IP and Transport header (except TTL and IP Checksum) are analyzed. Secondly, 8 high entropy bytes are used in order to compare how many additional input collision are implied by using only parts of the packet content. The amount of identical packets will be compared to collisions caused by a bad hash input configuration (only IP header without checksum and TTL), an 8 high entropy bytes hash input configuration and a 16

bytes combination used by Molina in [?].

1. Recommended 8 bytes - based on the entropy evaluation results the IP identification field and 6 Bytes depending on the transport protocol are chosen: 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)
2. Molina’s 16 bytes - Molina [?] proposes the use of these 16 bytes: Total Length, Identification, Source and Destination IP address and the 2<sup>nd</sup> 32 bit word of the transport header.

### 3.4.3 Results

#### 3.4.3.1 Comparison of Header Byte Combinations

The size of a collision is the amount of packets with the same hash input. Large collisions are of more concern than small collisions as the colliding packets in large collisions all have the same selection decision and are either over- or under-represented in the selected subset. The same amount of packets within several small collisions is less crucial, because the packets from different small collisions can have different selection decisions and there is no "all or none" decision. Hence the largest input collisions are of major concern.

Here the 20 largest input collisions in each trace are analyzed. The amount of packets contained in these input collisions were added for the whole trace group. For all traces there is a great amount of packets that occur in bursts and have different TTL and IP Checksum values but are equal in the remaining bytes. Detailed analysis showed that these packets are caused by routing loops or the lack of the IP ID field (IP ID=0). The results in Tab. ?? are explained by an example of the 36 Twente trace files that are evaluated. The 20 largest collisions of each trace include 13,120 packets because the packets are identical. Using only the IP header as hash input about 475,000 packets are observed in the largest collisions. With the use of the recommended 8 Bytes 16,004 packets collided, whereas with Molina’s 16 bytes 49,477 packets collide. For the FHSalzburg trace group, the content combinations of Molina and ours identify the same amount of identical packets. Our subset shows a similar amount of identical hash inputs for the NZIX traces whereas Molina’s subset has 3 times more collisions. The LEO2 trace group that consists of VPN tunnel traffic shows a different result: Molina’s 16 byte subset includes less collisions. This is caused by the low entropy of our recommended bytes in the LEO2 UDP header fields, especially the missing UDP

Trace Group	Colliding Packets	Largest Collision	Packets in this Trace	Largest Collision/Packets
FH Salzburg	0,05%	21	5283018	3,98E-06
Twente	0,04%	320	10000000	3,20E-05
LEO 1	0,35%	1077	10000000	1,08E-04
LEO 2	8,83%	113	10000000	1,13E-05
NZIX	8,37%	33753	10000000	3,38E-03

Table 5: Largest Clusters of Identical Packets in Traces

Trace Group	Trace Files	Packets/File in millions	Identical IP + Transport header	Identical IP Header	Recommended 8 Bytes	Molina's 16 Bytes
FH Salzburg	18	6	3547	238174	3547	3547
NZIX	19	10	484034	1564246	484405	1562066
Twente	36	10	13120	475570	16004	49477
LEO 1	12	10	61072	450273	73730	86809
LEO 2	1	10	949	8116	7919	1121

Table 6: Packets in Collisions for Different Selected Byte Combinations

checksum causes more identical packets. From the results of the input collision comparison it is concluded that it is important to use high variable fields as hash input in order to keep hash input collisions at minimum. The recommended combination is sufficient for most traces but can produce larger clusters as in the LEO1 Trace Group. In order to improve the results one can add some more high entropy bytes which are proposed in Table ??.

### 3.4.3.2 Traces Suitable for Hash Function Comparison

The evaluation of input collisions also identifies traces that are not suitable for hash-based selection. Many input collisions in a trace that are caused by identical packets lead to a biased selection. This bias is not introduced by the hash function but by input collisions. Hence, the traces that have very few input collisions are best suited for evaluating hash functions. As shown in Table ?? the trace groups from NZIX and LEO2 include more than 8% of identical packets. Although the LEO1 trace group includes only 0.35% identical packets, high amounts of packets share the same hash input in the 20 largest collisions (cf. Tab. ??). The NZIX, LEO1 and LEO2 trace groups will imply bias for the hash-based selection decision and should not be used for hash functions comparison. In Section ?? the reason for doubled packets in all traces will be discussed more thoroughly. This may enable a classification of traffic traces where hash-based selection is applicable. Because the Twente and FH Salzburg traces include only few collisions the evaluation of hash functions in the next chapter will be based on them.

# Chapter 4

## Evaluation of Hash Functions

In this chapter a set of hash functions will be evaluated on their suitability for hash-based selection. The main goal is to find a combination of hash input and hash function which selects a representative subset of packets. Quality criteria are defined upon which each function is tested. Further security issues are discussed when using hash functions for packet selection purposes.

### 4.1 Desired Properties

Following requirements a hash function intended for hash-based selection has to fulfill:

1. fast calculation time
2. nonlinearity
3. unbiasedness
4. representativeness of sampled subset

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

The second criterion is the nonlinearity of hash values, i.e that hash input are distributed randomly over the hash range. In case packets which only differ in some bits of the hash input are mapped closely together, they have the same sampling decision and the sampled subset is biased. This may even be exploited by an adversary who intentionally crafts packets in order to influence the sampled subset (more in Sect. ??). The **Avalanche Criterion** is used to assess the ability to distribute hash values randomly over the hash range.

The third requirement is that the selection decision of the hash function based on an hash input should be **unbiased**. The selection should not favor any packets or bitstreams to others in the selection decision. The **chi-square independence test** is used to analyze if the sampling decisions based on the hash input and the

hash function is independent to packet attributes.

The last requirement is that of the **representativeness of the sampled subset**. The distribution of packet attributes in the sampled subset have to be the same as in the population. The **chi-square distribution test** also known as the Pearson's goodness-of-fit test compares the proportion of packet attributes in the sampled subset and the population. It will be shown that unbiased sampling leads to representative subsets. The chi-square independence test will be derived from the chi-square distribution test in order to prove that hash functions that sample unbiased to a certain packet attribute also select a representative subset too.

Following requirements are proposed by Molina [?]. These requirements may be fulfilled for hash functions intended for hash-based selection, but should be fulfilled for hash functions creating packet IDs.

- uniform distribution
- low collisions

**Uniform distribution and low collision probability** are two concurring criteria for hash functions. In order to obtain low collisions the hash function has to distribute the has values uniformly over the hash range. For hash-based selection it is more convenient to adjust the selection range for a certain target sampling ratio if the hash values distribute uniformly although uniformity is not a prerequisite. For uniform hash values and a certain target sampling fraction  $t\%$ , the selection range has to consist of  $t\%$  values of the hash range. For non-uniform hash value distributions the selection range has to be delicately adjusted according to the hash value distribution (empirically measured) in order to obtain a certain target sample size.

Collisions are critical for cryptographic purposes like verifying the integrity of data (see section ??). The hash function should be collision resistant for packet ID generation as well. Packet IDs are intended for distinguishing different packets at the multipoint collector. In case of packet ID collisions packets may not be properly distinguished and need to be discarded which decreases the techniques efficiency. For hash-based selection collisions do not matter as long as they appear randomly and there is no dependency between any packet property of the colliding packets. Random colliding packets are still representatively and unbiasedly selected.

## 4.2 Collection of Hash Functions

The computational overhead introduced by hash-based packet selection is critical for multipoint measurements. The resource consumption at the observation point has to be minimized. From previous tests [?] it is already known that CRC32 is already very slow for hash-based selection purposes. In comparisons of cryptographic hash functions [?] [?] it is shown that most cryptographic hash functions are even slower than CRC32. Therefore the hash function collection that is going to be analyzed consists of 23 fast non-cryptographic hash functions. All hash functions have a 32 bit digest (hash value) which makes it more easy to

Hash Function	Invented By	Source Code	Hash Function	Invented By	Source Code
<b>BOB</b>	Robert Jenkins	[Jen]	<b>BRPHash</b>	Bruno Preuss	[Lov]
<b>OAAT</b>	Robert Jenkins	[Jen]	<b>CRC32</b>		[Lov]
<b>Simple</b>		[Mul]	<b>PYHash</b>		[Lov]
<b>SBOX</b>		[Mul]	<b>SDBM</b>		[Neu]
<b>TWMX</b>	Thomas Wang	[Lov]	<b>OCaml</b>		[Neu]
<b>Hsieh</b>	Paul Hsieh	[Lov]	<b>SML</b>		[Neu]
<b>RS</b>	Robert Sedgewick	[Lov]	<b>STL</b>		[Neu]
<b>JS</b>	Justin Sobel	[Lov]	<b>SHA</b>		[RFC3174]
<b>BKDR</b>	B.Kernighan	[Lov]	<b>MD5</b>		[Plu]
<b>DJB</b>	Daniel Bernstein	[Lov]	<b>MMH</b>		[HaKr97]
<b>NDJB</b>	Daniel Bernstein	[Lov]	<b>FNV32</b>	Fowler Noll	[Nol]
<b>DEK</b>	Donald E.Knuth	[Lov]	<b>CS</b>	Carsten Schmoll	[Schm]
<b>AP</b>	Arash Partow	[Lov]			

Table 7: Hash Function Collection

compare the functions. The cryptographic hash functions SHA1 and MD5 are added to the collection as well. These cryptographic functions shall be compared to the other functions in order to assess if they show better quality. The SHA-128 bit and MD5-160 bit hash values are trimmed to 32 bits by adding up each 32 bit subblock of them. The functions were available at various online sources [?][?][?][?][?][?][?][?][?][?]. An overview of the Collection is shown in Table ??.

### 4.3 Security Issues for Hash-Based Selection

A recent paper published by Goldberg and Rexford [?] deals with security vulnerabilities arising with hash-based selection. In the context of packet sampling security is broken if an adversary can influence the selection process in a way, that the adversary's packets are disproportionately included in the the sampled subset.

**Need for keyed hash function** Goldberg and Rexford demonstrate that only a keyed hash function is secure. A keyed hash functions appends some secret key to the hash input before calculating the hash value. Suppose a unkeyed hash function is used for hash-based packet selection and the hash function and selection range is known to the adversary. If the adversary does not want to be billed for his traffic he calculates the hash value prior to sending his packets and ensures by padding some additional bits that the hash value does not fall into the selection range.

Even if the selection range is secret the adversary can learn the range by looking at his bill. He crafts packets that have hash values in a small range A. If A is disjoint with the selection range  $S$  than the packets are not selected for accounting. Even if the packets are selected the adversary learns that A and  $S$  overlap and will use a different range A in the next billing period until he finds a range that is not included in  $S$ .

These examples demonstrate that secure hash-based selection requires a secret key and a private selection range at all observation points. The secret key can be 1) an initial value for the hash function 2) some bytes that are appended to the hash input or 3) some private hash function parameter. In this way the adversary cannot calculate the selection hash value before sending the packets

without knowing the secret key. Furthermore, Goldberg and Rexford suggest the use of new secret keys at all observation points after each billing period. An adversary can learn from the billing report which packets are selected and craft identical packets with the same hash input. This is also known as replay attack.

**Pseudorandom Function** Goldberg and Rexford moreover reason that a keyed Pseudorandom Function (PRF) is required for hash-based packet selection. Simple linear<sup>1</sup> hash functions are exploitable by their characteristics and reveal the secret key. They show that a CRC hash function  $h(c \cdot k) = c \bmod k$  is not secure where  $c$  is the hash input and  $k$  is the secret modulus. The function has the property that hash values with a distance of  $k$  have the same selection decision. Hence the adversary tries to find two consecutive pairs of packets where the first packet of each pair is not selected and the successor is selected:

Packet Pair 1 :  $(h(c) \ni S, h(c + 1) \in S)$

Packet Pair 2 :  $(h(c + x) \ni S, h(c + x + 1) \in S)$ .

If he finds such a pair the secret key  $k$  is the distance  $x$  between those pairs. Because of the linear property of the hash function, an adversary can craft packets with successive selection hash values by using packets with successive hash inputs. The adversary then observes the selection decision of each packet on his bill until he finds such two pairs P1 and P2. This example shows that linear hash functions have security vulnerabilities and should not be used for hash-based selection. Goldberg and Rexford propose the use of Pseudorandom Functions that take a non random input and produce output values which are undistinguishable from a random number.

These computational intensive functions introduce heavy load on each observation point. For the hash-based selection purpose it has to be assessed if it is realizable and appropriate to process a heavy function for each packet at a network node. Here it is argued that hash functions that use a secret key and fulfill the avalanche criterion are sufficient for the hash-based selection purpose. The avalanche criterion tests the randomness and non-linearity of output values (all PRFs have to pass this test). The Avalanche criterion is fulfilled if the change of one input bit introduces a change of all output bits by a probability of 50%. If there is a hash function that satisfies this criterion each change in one bit of a packet will create a totally different hash value. With secured packet exports an adversary can only learn from billing reports about the hash function and the secret key. As there is no linearity between hash key and hash value it will be difficult if not impossible to figure out the secret key or craft disproportional sampled packets by only knowing the selection decision of priorly sent packets. The information from billing even loses value if the secret key is changed every billing period as proposed by Goldberg.

**Secure Export of Packet Information** Goldberg and Rexford argue that information of selected packets should not be transferred to the collector instantaneously on the arrival of the packet, but the information should be stored at the measurement point until the end of a measurement interval. Otherwise an

---

<sup>1</sup>The hash value  $h(c + 1)$  is correlated to  $h(c)$  by:  $h(c + 1) = h(c) + 1$

adversary can observe which of his crafted packets are selected by looking at the occurrence of measurement reports at the arrival of the packets.

The PSAMP working group advises that the export data has to be encrypted on the way to the collector. There is a need of deploying algorithms that ensure 1) confidentiality 2) integrity and 3) authenticity [?]. Confidentiality is the ability to encrypt export data so that it cannot be read by someone else. Because user specific information is exported in flow aggregated data the data has to be scrambled to ensure privacy. A sniffer that is able to read exported packet IDs can trace packets from a user to the destination. Integrity secures that alterations in the exported data are detected, whereas authenticity ensures that the sender of data can be verified in order to detect forged data. An adversary may try to send bogus information to the multipoint collector in order to cover his sent traffic. Integrity and authenticity ensure that no information can be changed in the export stream and that an adversary can pretend to be an observation point.

## 4.4 Performance Measurements

### 4.4.1 Experimental Setup

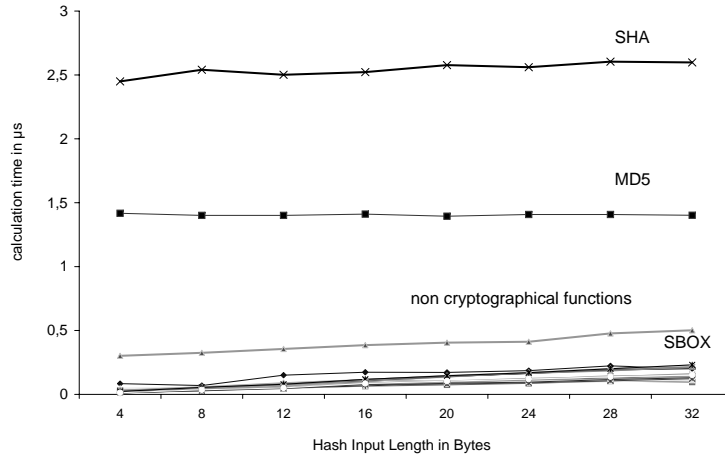
The performance of the 25 hash functions is measured by an evaluation tool written in C++. The tool generates random bytestreams of definable length upon which the hash function is applied. An average of 1 million repeated calculations is measured. The tool is run on a Dell Inspirion PC (2.8 GHz, 2048 KB cache, 1 GB RAM) with Kubuntu Feisty OS. Each hash function is analyzed for 8 different hash input lengths (4,8,12, ..., 32 bytes) which are the first 8 multiples of 4 bytes.

### 4.4.2 Results

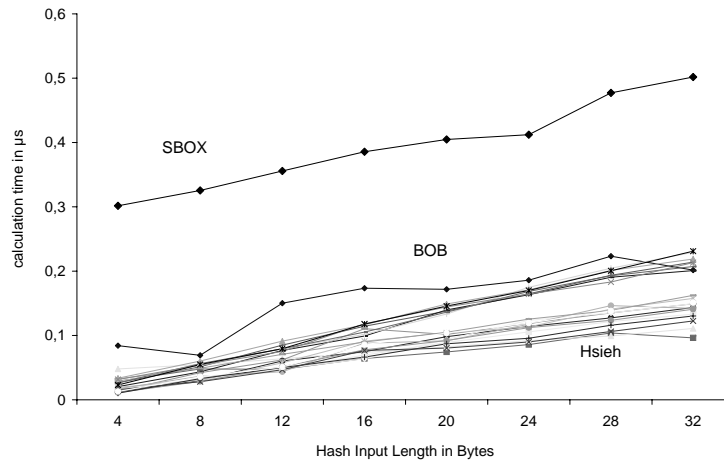
In Fig. ?? the measurements results are shown for different packet lengths. Because of the large amount of hash functions both sub figures only show a general view for hash function groups.

**Difference Cryptographic and Non-Cryptographic Hash Functions** Figure ?? shows the contrast in calculation time between the cryptographic hash functions (SHA, MD5) and all other non-cryptographic hash functions. For small hash inputs the average calculation time of all non-cryptographic hash functions (except CRC32) is compared to MD5 and SHA 33 and 57 times faster. With a 32 byte hash input length this discrepancy becomes less with an average difference of 7 (MD5) and 12 (SHA) times the calculation time. CRC32 is not depicted in these diagrams because it is with about  $200\mu s$  very slow. 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 and one may consider to use a hash function multiple times, i.e calculate the function on the the hash values again.

**Growing Calculation Time with Increasing Hash Input Length** It is observed that the hash calculation time increases with the packet length for non-cryptographic hash functions (cf. Fig. ??). The non-cryptographic hash func-



(a) Performance Cryptographic/Non-Cryptographic Functions



(b) Only Non-Cryptographic Functions

Figure 17: Performance of Hash Functions

tions 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%. For the purpose of hash-based selection these results mean that it is necessary to keep the hash input length at a minimum. The packet content that is used as hash input should be as short as possible since the network nodes only have restricted resources for measurement purposes. Nevertheless the hash input length should be sufficient to ensure proper packet selection that can closely emulate random selection.

After evaluating the hash function collection on other criteria a more distinguished diagram on the performance of the best hash functions will be shown.

## 4.5 Avalanche Criterion

### 4.5.1 Approach

The avalanche criterion tests the second requirement for the hash function intended for hash-based selection: **nonlinearity**. The Avalanche criterion is proposed by [?] [?] 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 1/2. This implies that hash keys that only differ in one bit disperse randomly. The algorithm is simple and shall be explained with a hash function with constant hash input length  $n$ . Consider that we want to measure the probability that input bit  $i$  changes the output bit  $o$ . First all  $C = 2^n$  hash input combinations are grouped in  $P_i = \frac{C}{2}$  pairs that only differ in hash input bit  $i$ . Consider to look at output bit  $o$  of the pairs corresponding hash values. Now one has to count the amount of hash input pairs that only differ in bit  $i$  and cause a change in hash value bit  $o$ . Let's denote this amount  $P_{i1}$  and pairs that do not cause a change  $P_{i0}$ . In case of avalanche the amount  $P_{i1}$  is close to half the amount of all pairs  $P_i$  that only differ in bit  $i$ . As to say  $\frac{P_{i1}}{P_i} = \frac{P_{i0}}{P_i} = 50\%$ .

### 4.5.2 Experimental Setup

A C# implementation of the test available at [?] is adopted to C++ for evaluating the hash function collection. All hash functions are analyzed on their avalanche, it is calculated with which probability the output bits change when one input bit is changed. The hash input bits are generated randomly. The avalanche test is conducted for different hash input lengths, all multiples of 2 bytes up to 16 bytes.

### 4.5.3 Results

The probabilities of changes are depicted in Fig. ?? which shows 3 exemplary avalanche results of the BKDR, BOB and CRC32 hash function with a hash input length of 2 bytes. In the figures the 32 output bits (digest length) are shown in the columns whereas the rows depict the 16 bits (=2 bytes) of input. Dark background colors and white font color represent poor avalanche whereas white background and dark fonts depict good avalanche. For the BKDR hash (cf. Fig. ??) 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 2 least significant bytes are not changed at all. With increasing hash input length those functions show better avalanche. Figure ?? shows the linear property of the CRC32 hash function. The change of an input bit causes always or never a change for an output bit. Five hash functions (CRC32, DEK, BRP, AP) have a linear dependency between hash input and hash value, these function are vulnerable to bias and security attacks (as explained in Section ??).

**Avalanche Improvement with Increasing Hash Input Length** The BKDR example also demonstrates one common shortcoming of some analyzed hash func-



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

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

Table 8: Avalanche Test Results

## 4.6 Independence of Sampling Decision and Representativeness

In the following Person’s chi-square tests are used for evaluating the hash-based selection decision on the third and fourth criterion: **biasedness** and **representativeness**. Already in 1993 Claffy [?] used the chi-square goodness of fit test to prove that different systematic and random sampling techniques select a representative subset of the original population. Duffield [?] used the chi-square independence test to analyze the selection decision of one hash function on the bias to the network prefix. Here both approaches will be combined to evaluate the biasedness and representativeness for the selected subset of packets.

### 4.6.1 Chi-Square Independence Tests

The chi-square independence test [?] statistically evaluates the dependency between two categorical variables. This test evaluates if the sampling decision  $X = \{x_1 = \text{“selected”}, x_2 = \text{“notselected”}\}$  is independent to a packet attribute  $Y = \{y_1, \dots, y_L\}$  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  $T_I$  is the sum of deviations between expected ( $\overline{h_{ij}}$ ) and realized ( $h_{ij}$ ) simultaneous occurrences of  $x_i$  and  $y_i$ .

$$T_I = \sum_{i=1}^2 \sum_{j=1}^L \frac{h_{ij} - \overline{h_{ij}}}{\overline{h_{ij}}} \quad (4.1)$$

The error  $\alpha$  to incorrectly reject H0 has to be defined prior to the test. Usually  $\alpha$  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%. Given that the selection decision and packet attribute are independent,  $T_I$  is asymptotically  $X^2$  distributed with  $(2 - 1)(L - 1) = (L - 1)$  degrees of freedom. The H0 hypothesis is rejected if  $T_I$  is above the critical  $X^2$  value  $X_{crit}((L - 1); \alpha)$ .

**Precondition for the test:** all expected frequencies  $\overline{h_{ij}}$  have to be at least 5.

### 4.6.2 Introduction to Person's Goodness-Of-Fit Test

The chi-square goodness-of-fit test evaluates if observed frequencies of a random variable follow a specified distribution. The test can be applied to arbitrary distributions, e.g. one can test if a distribution follows a normal distribution or any empirical distribution. The test statistic  $T_D$  is the normalized sum of differences between the expected frequencies assuming a certain distribution  $\overline{h}_i$  and the observed frequencies  $h_i$ , where  $i=(1..L)$  denotes the different realizations.

$$T_D = \sum_{i=1}^L \frac{h_i - \overline{h}_i}{\overline{h}_i} \quad (4.2)$$

Assuming that both distributions are equal the test statistic  $T_D$  is asymptotically  $X^2$  distributed with  $N=(L-1)$  degrees of freedom. The  $H_0$  hypothesis is rejected if  $T_D$  is above the critical  $X^2$  value  $X_{crit}((L-1); \alpha)$ .

**Precondition for the test:** all expected frequencies  $\overline{h}_{ij}$  have to be at least 5.

### 4.6.3 Derivation of Goodness-of-Fit 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. ??):

$$T_I = \underbrace{\sum_{i=1}^L \frac{h_{i1} - \overline{h}_{i1}}{\overline{h}_{i1}}}_{Term\ 1} + \underbrace{\sum_{i=1}^L \frac{h_{i2} - \overline{h}_{i1}}{\overline{h}_{i2}}}_{Term\ 2} \quad (4.3)$$

The term 1 in Eq. ?? equals the test statistic  $T_D$  from the chi-square distribution test (Eq. ??). The independence test statistic  $T_I$  is larger than  $T_D$ , because it includes the additional summand of term 2. This implies that the test statistic  $T_I$  of the independence test is more stringent than the one of the goodness-of-fit test. More independence tests are rejected than goodness-of-fit tests.

It is noted 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.

### 4.6.4 Parameters for the Chi-Square Tests

The chi-square test is used to analyze if the hash-based selection technique is able to select an unbiased and representative subset of the population. Nevertheless the test results do not solely depend on the hash function, but on different parameters. The chi-square tests will be conducted with different parameter configurations in order to show an overall applicability for these configurations. In the following these parameters will be presented:

**I. The structure and length of the hash input** As shown in the avalanche criterion evaluation some hash functions are not indifferent to the length of the hash input. Furthermore there is a chance that the dynamic structure of the hash input has an influence on the sampling decision. In the following tests two hash input configurations are used:

1. IP and Transport Header except Version/IHL, TOS, TTL, IP Checksum, Flags/Fraggment Offset and IP Options<sup>2</sup>
2. Recommended 8 Bytes - based on the entropy evaluation results from the previous chapter the IP identification field and 6 Bytes depending on the transport protocol are chosen: 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)

**II. The traffic trace** The quality of the selection decision is trace dependent. As shown in Fig. ?? traces with high amounts of identical packets evoke bias in the selection decision. Hence two trace groups with low collisions will be used for evaluating the selection decision:

1. FH Salzburg trace group
2. Twente trace group

Both traces include about the same percentage of colliding packets: FH Salzburg 0.05% and Twente 0.04%. Nevertheless the size of the collisions differ, in the FHSalzburg traces the largest collision includes only 21 packets whereas for the Twente trace group up to 231 packets are identical in all bytes except TTL and Checksum.

**III. The analyzed packet attribute** The chi-square independence tests analyze the dependency between one packet attribute and the selection decision. The chi-square independence tests will be conducted for three packet attributes:

1. packet length
2. transport protocol
3. byte value.

**IV. The hash function** The hash function has major influence on the quality of the selection decision. The following tests will be conducted with all 25 hash functions.

#### 4.6.5 Multiple Independence Tests

The chi-square independence test is applied to the hash function collection in order to test if the sampling decision and certain packet attributes are dependent. The tests are based on the FHSalzburg trace group consisting of 51 trace

---

<sup>2</sup>These fields are either variable between network nodes or almost not variable among packets as explained in section ?. They are unsuitable for hash-based selection

files and on the Twente trace group consisting of 100 trace files. For each trace file a separate independence test is conducted. The reason for using separate independence test are the following: 1) measurement intervals are simulated and 2) the influence on the bias caused by large hash input collisions is minimized. In longer measurement intervals recurring packets will have the same selection decision. In a real life scenario the selection range can be modified after a measurement interval and the selection decision for packets with the same hash value can be different. This improves the representativeness of the selected subset. The problem with multiple chi-square tests is that every test inherits an error. As priorly noted, H0 is discarded in  $\alpha$  (e.g 5%) of the individual tests although there is no dependency between the variables. One is 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 [?] [?]. Under the assumption of H0, the probability  $\alpha_{global}$  of falsely rejecting at least one out of n independent individual tests, each of level  $\alpha_{ind}$ , is  $1 - (1 - \alpha_{ind})^n$ . A global test is defined that restricts the combined error  $\alpha_{global}$  of all individual tests each with an error  $\alpha_{ind}$ . The global test hypotheses are:

**global H0:** The attributes are independent

**global H1:** The attributes are dependent.

H0 is rejected if at least one of the adjusted individual test shows dependency. The global error  $\alpha_{global}$  is set to be 5%, i.e. if none of the test shows real dependency one expects the global test to fail only with a 5% probability. In order to achieve a global error of 5% the individual test error  $\alpha_{ind}$  are adjusted according to Eq. ??.

$$\alpha_{ind} = 1 - (1 - \alpha_{global})^{\frac{1}{n}} \quad (4.4)$$

Because the individual test error is lowered only individual tests are rejected that lead to a rejection of the global H0. This implies that the critical  $X_{crit}^2$  for the global test is above the individual tests critical value.

## 4.6.6 Independence Sampling Decision and Packet Length

### 4.6.6.1 Experimental Setup

Common Parameter	Value
Dependency Analyzed	Sampling Decision And Length
Categories	7
Degrees of Freedom	$(7 - 1) \times (2 - 1) = 6$
Alpha error	5%
Selection Range	$[0.2 \times 2^{32} .. 0.3 \times 2^{32}]$ (10%)

The packets are categorized into 7 length groups: 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 the sampling decision is dependent on the packet length, i.e if packets with a length in one category have a higher sampling probability as in another category. For the FHSalzburg traces a measurement interval of 200.000 packets and for the Twente trace a measurement interval of 1 million packets is chosen. The selection range is the third 10% slice of the hash range ( $S = [\frac{2}{10} 2^{32}; \frac{3}{10} 2^{32}]$ ). For each trace file a separate Independence Test is conducted.

The test hypotheses are:

- H0:** Sampling decision and packet length are independent
- H1:** Sampling decision and packet length are dependent

The test statistic is:

$$T = \sum_{i=1}^2 \sum_{j=1}^7 \frac{h_{ij} - \bar{h}_{ij}}{\bar{h}_{ij}}$$

#### 4.6.6.2 Results

Parameter	FH Salzburg	Twente
Measurement Interval	200.000	1 Mio
Trace Files	51	100
alpha error individual	0.1%	0.05%
X <sup>2</sup> critical global	22.45	24.04

The Box-Whisker-Plots shown in Figure ?? depict the Minimum, 5% Quantil, Median, 95% Quantil and Maximum of the test statistic T for each hash function. The 95% quantil illustrates the amount of tests that are expected to reject H0 under independence. On the very left the distribution of the test statistic T for random probabilistic sampling (p=10%) is depicted as reference. The lower horizontal line represents the individual critical value  $X_{crit}^2 = 12.6$  when the multiple test correction is not used. All individual tests that have a test statistic T above this value reject H0.

**Dunn-Sidak** Even if Packet Length and Sampling decision are independent we expect 5% of the 51 tests for each hash function to reject H0. 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 three times for FH Salzburg and one time for Twente traces although dependency can be excluded by definition. Hence we need to adjust the error  $\alpha$  to obtain only tests that truly show dependency. Applying the Dunn-Sidak correction (Eq. ??) we obtain:

$$FH : \alpha_{ind} = 1 - (1 - 0.05)^{\frac{1}{51}} = 0.001 \quad Twente : \alpha_{ind} = 0.0005$$

The upper horizontal lines in Fig. ?? show the adjusted global  $X_{crit}^2$  values. Each hash function that has one rejected test (T above global  $X_{crit}^2$ ) is truly biased at least for this trace.

**IP and Transport Header used as Hash Input** The OAAT hash function has no rejected individual test for the FH Salzburg and Twente traces when the IP and Transport header is used as hash input. Hence, OAAT is the only hash function whose sampling decision is definitely independent from the packet length in case of this hash input configuration. For the other hash functions at least one individual test is rejected. It is noticeable that the majority of hash functions perform very well; their distribution of the test statistisc T is not obviously different to probabilistic sampling. Interesting are the results for the BRP and AP Hash function for the FH Salzburg traces. Although for both

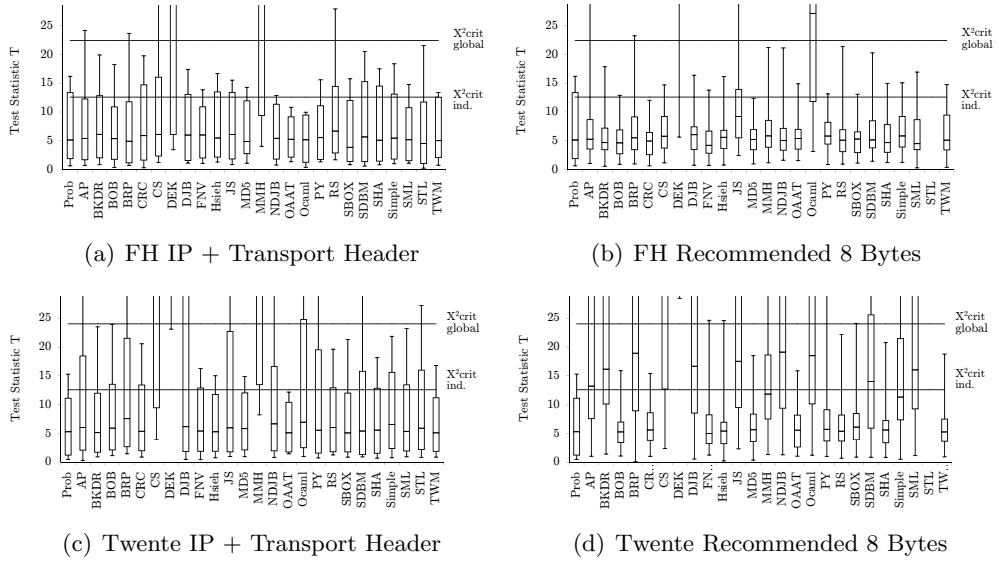


Figure 19: Distribution of Test Statistic T for Independence Test on Length

(a) IP + Transport Header

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	TWIX
Individual Tests																										
FH (51)	3	2	3	2	1	4	8	42	5	2	4	4	1	46	1	0	0	1	5	1	4	5	3	3	1	3
Twente (100)	1	14	5	10	28	6	85	100	17	6	3	18	3	96	18	0	15	18	6	4	10	6	11	7	14	4
Global Tests																										
FH (51)	0	1	0	0	1	0	1	38	0	0	0	0	0	34	0	0	0	0	1	0	0	0	0	0	0	0
Twente (100)	0	2	0	0	5	0	62	99	8	0	0	5	0	86	1	0	6	3	0	0	3	0	0	0	1	0
Overall																										
Quality		++	++	++	-	++	-	-	-	++	++	+	++	-	+	++	-	+	+	++	+	++	++	++	+	++

++ no test globally rejected + 1-2 rejected +- 3-5 rejected - 6-10 rejected -- more than 10

(b) Recommended 8 Bytes

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	TWIX
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
Overall																										
Quality		--	--	++	-	++	-	-	-	+	+	-	++	-	-	++	-	+	++	+	-	++	--	--	x	++

++ no test globally rejected + 1-2 rejected + 3-5 rejected - 6-10 rejected -- more than 10

Table 9: Individual and Global Rejections of Independence Test Length

functions less than 5% of the individual tests are rejected (their 95% Quantil is below the individual critical value), both include one test where the sampling decision is strongly correlated to the packet length. This assumption is fortified by the results of the Twente traces. Both hash functions imply a biased decision for 2 (AP) and 5 (BRP) tests. The Ocaml Hash function which did not indicate any dependence for all 51 FH Salzburg trace files includes 6 Twente tests that prove a strong correlation between length and sampling decision. The amount of rejections for individual and global tests are shown in Table ???. Hash functions with many global rejections imply bias on the sampling decision caused by the length of packets. 12 hash functions sample unbiased to packet length for the long hash input.

**8 Recommended Bytes used as Hash Input** In case the hash input consists of only the 8 recommended bytes, some hash functions' results degrade. With the 8 byte long hash input the STL hash function does not produce any hash values in the selection range and cannot be used with this configuration. For the other hash functions especially the tests based on the Twente traces are more often rejected. Hash functions (like the BKDR function) that performed very well for the long hash input configuration, have a multitude of rejected tests for the 8 byte hash input configuration. Other hash functions whose performance decreases are AP, BRP, DJB, JS, NDJB, Ocaml, SDBM and Simple hash function, marked with a gray colored background in Tab. ??. The SBOX, Hsieh and FNV32 hash function have one rejected test as well, but still perform well. In contrast, fewer independence tests are rejected for PY and RS hash functions for the short input configuration. The hash functions that produce an unbiased subset in all tests for the 8 byte hash input configuration are BOB, MD5, CRC32, OAAT, RS, SHA and TWMX.

#### 4.6.7 Independence Sampling Decision and Protocol

In the following it is evaluated if the transport protocol has an influence on the packet selection decision when hash-based selection is used.

##### 4.6.7.1 Experimental Setup

Common Parameter	Value
Dependency Analyzed	Sampling Decision On Protocol
Categories	3
Degrees of Freedom	$(2 - 1) \times (3 - 1) = 2$
Alpha error	5%
Selection Range	$[0.2 \times 2^{32} .. 0.3 \times 2^{32}]$ (10%)

From all traces the 3 main transport protocols are filtered: TCP, UDP and ICMP. Hence the degrees of freedom for the chi-square tests are  $(3 - 1) \times (2 - 1) = 2$ . The hypotheses are:

**H0:** Sampling decision and transport protocol are independent

**H1:** Sampling decision and transport protocol are dependent

The test statistic is:

$$T = \sum_{i=1}^2 \sum_{j=1}^3 \frac{h_{ij} - \bar{h}_{ij}}{\bar{h}_{ij}}$$

#### 4.6.7.2 Results

Parameter	FH Salzburg	Twente
$X^2$ critical individual	5.99	
Measurement Interval	200000	1 Mio
Trace Files	51	100
$X^2$ critical global	13.8	15.15

**IP and Transport Header used as Hash Input** The results are presented in the Box-Whiskers-Plots in Fig. ???. The lower horizontal line represents the critical  $X^2$  above which all individual tests reject  $H_0$ . The upper horizontal line represents the Dunn-Sidak adjusted global critical  $X^2$  value. All tests that have a test statistic  $T$  above this value show true dependency between sampling decision and protocol at least for this trace file. The MMH hash function which performed very poorly in the independence test between sampling decision and length has no rejected global test for the FH Salzburg traces but 16 for the Twente traces. For two tests based on the SHA function the test statistic is above the global critical bound which indicates biasedness. The number of rejections for individual and global test for all hash functions are shown in Table ???. There are 9 hash functions that did not show any sign of dependency between protocol and sampling decision.

**8 Recommended Bytes used as Hash Input** In case the 8 recommended bytes are used as hash input, it is again the same collection of hash functions whose performance degrades. Although the protocol field is not included in the hash input the AP, BKDR, BRP, JS, NDJB, OCAML, SDBM, Simple and SML hash functions show strong dependency of the sampling decision and the transport protocol for the 8 bytes input configuration. Compared to the FH Salzburg traces, the tests based on the Twente traces are more often rejected, e.g. 2% of the FH Salzburg tests are rejected for the AP hash function compared to 44% for the Twente tests. Nevertheless for 10 hash functions (BOB, CRC32, FNV, Hsieh, MD5, OAAT, RS, SBOX, SHA and TWMX) none of the tests is rejected and all these hash functions do not show any dependency between sampling decision and protocol for the short hash input.

#### 4.6.8 Independence Sampling Decision and Byte Value

Hash functions are indifferent to the conceptual 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.

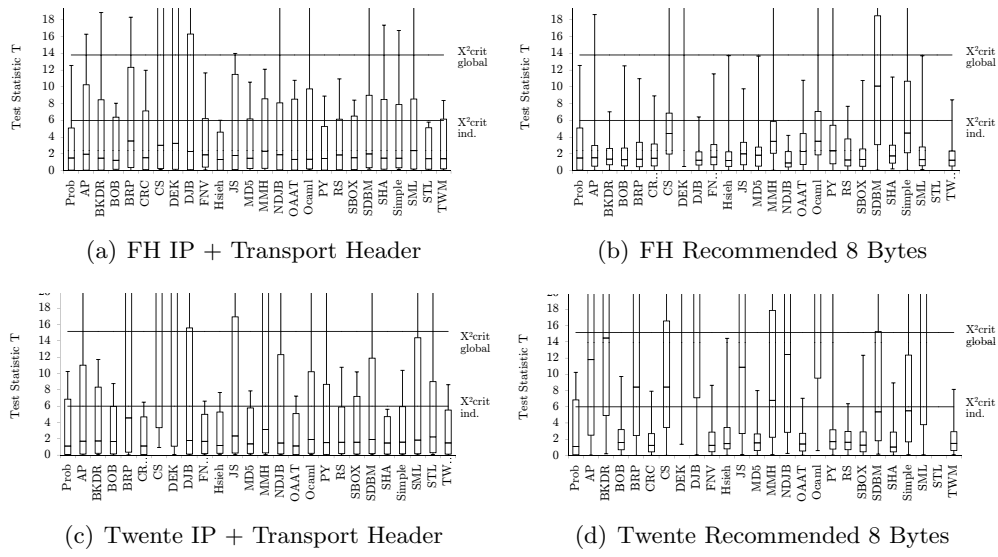


Figure 20: Distribution of Test Statistic T for Independence Test Protocol

(a) IP + Transport Header

Function	Prob	AP	BKDR	BOB	BRP	CRC32	CS	DEK	DJB	FNV32	Hsieh	JS	MD5	MMH	NDJB	OAAAT	Ocaaml	PY	RS	SBOX	SDEM	SHA	Simple	SML	STL	TWIMX			
Individual Tests																													
FH (51)	2	11	6	4	13	5	18	18	6	3	1	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	
Twente (100)	6	17	10	5	45	1	92	87	23	1	2	16	5	35	16	3	14	15	5	11	17	0	4	17	13	4	4	4	
Global Tests																													
FH (51)	0	2	2	0	1	0	5	6	4	0	0	1	0	0	2	0	2	0	0	0	1	2	1	1	0	0	0	0	
Twente (100)	0	3	0	0	13	0	76	77	7	0	0	6	0	16	1	0	2	2	0	0	3	0	0	4	1	0	0	0	
Overall																													
Quality		+-	+	++	--	++	--	--	--	++	++	-	++	--	+-	++	+-	++	++	++	+-	+	+	+-	+	++	++	++	++

++ no test globally rejected + 1-2 rejected +- 3-5 rejected - 6-10 rejected -- more than 10

(b) Recommended 8 Bytes

Function	Prob	AP	BKDR	BOB	BRP	CRC32	CS	DEK	DJB	FNV32	Hsieh	JS	MD5	MMH	NDJB	OAAAT	Ocaaml	PY	RS	SBOX	SDEM	SHA	Simple	SML	STL	TWIMX			
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	2	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	6	6	
Global Tests																													
FH (51)	0	1	0	0	0	0	3	45	0	0	0	0	5	0	0	2	2	0	0	20	0	9	0	x	0	0	0	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	0	0	
Overall																													
Quality		--	--	++	--	++	--	--	--	++	++	--	++	--	--	++	--	+	++	++	++	--	++	--	--	x	++	++	++

++ no test globally rejected + 1-2 rejected +- 3-5 rejected - 6-10 rejected -- more than 10

Table 10: Individual and Global Rejections of Independence Test Protocol

#### 4.6.8.1 Experimental Setup

For each byte of the hash input an individual chi-square independence test is conducted. Because of differences in the hash input length the tests differ for both hash input configurations.

**IP and Transport Header as Hash Input** The first input configuration includes all IP and transport header fields except Version/IHL, TOS, TTL, IP Checksum and Flags/Fragment Offset. Because the different transport headers are of different length, the hash input varies for the different transport protocols as well. The shortest keys are included in UDP packets with only 21 bytes (13 IP bytes and 8 UDP bytes). This is the reason why only the first 21 bytes of the hash input are analyzed in the independence test. Hence, 21 independence tests are conducted for each trace file on each hash function. This is a total of 1071 tests for the FH Salzburg trace group and 2100 tests for the Twente traces group for each hash function. The Dunn-Sidak correction is used (Eq. ??) to adjust the error  $\alpha$  for each individual test in order to imply the correction for multiple testing. As to satisfy the precondition for the independence tests byte values which have lower frequencies than 5 are merged with their successor value.

**8 Recommended Bytes as Hash Input** For this configuration the hash input consists of 8 bytes. As only the dependency between any bytes of the hash input and the sampling is analyzed, there are only 8 tests for each trace and hash function. This a total of 408 test for FH Salzburg and 800 for the Twente trace group. The hypotheses are:

**H0:** Sampling decision and byte value are independent

**H1:** Sampling decision and byte value are dependent

The test statistic for each individual test is:

$$T = \sum_{i=1}^2 \sum_{j=1}^N \frac{h_{ij} - \bar{h}_{ij}}{\bar{h}_{ij}}$$

Since there are bytes that have very few degrees of freedom (e.g protocol) and some that include many (e.g transport checksum) the critical individual and global  $X^2$  differ for each byte. 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 T is divided by this critical  $X^2$  value in order to compare results for different chi-square tests with different degrees of freedom. The obtained metric is the relative test statistic  $T_r$  on which the test decision can be based. If  $T_r$  is above 1 H0 is rejected, which means that at least for this byte the selection decision is biased.

$$T_r = \frac{T}{X_{critical}^2} \rightarrow \text{if } T_r \begin{cases} \leq 1 & \text{H0 not rejected} \\ > 1 & \text{H0 rejected} \end{cases}$$

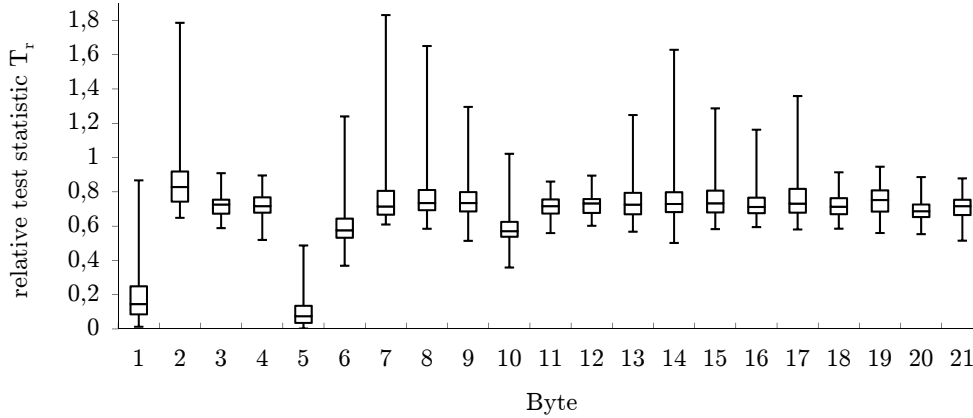


Figure 21: Independence Test on Byte for Simple Hash Function

#### 4.6.8.2 Results

**Graphical Explanation** An exemplary graphical result for the Simple hash function based on the Twente traces is shown in Fig. ???. The IP and Transport is used as hash input for this example. The tests that show a relative test statistic  $T_r$  above 1 reject  $H_0$  and statistically prove dependency between a byte value and the sampling decision. For 10 out of the 21 bytes there are test statistic above the critical  $X^2$  value, i.e. the selection decision is at least biased to these bytes.

**IP and Transport Header as Hash Input** For the long hash input configuration many independence tests are based on few degrees of freedom of the test statistic. For some bytes only 5-20 of the possible 256 byte values are occupied which indicates the low entropy of these fields. The test results for all hash functions are presented in a condensed summary in Tab. ??. This table shows the sum of the global rejected tests for all bytes for each hash function. Most hash functions perform poorly in these tests even if the long hash input configuration is used. The cryptographic hash function MD5 has 7 rejected tests for the FH Salzburg traces. The previously well performing CRC32 hash function has 13 rejections. Only BOB, SHA, SBOX and OAAT showed no dependency. Still reasonable well performed the Hsieh, RS and TWMX hash function with only 1 or 2 rejected tests.

**8 Recommended Bytes as Hash Input** For the 8 byte hash input configuration, the critical  $X^2$  values are higher because the degrees of freedom are higher. Each analyzed byte of all traces has 256 possible hash values, which indicates the high entropy of these bytes. Only 408 and 800 tests are conducted for the FH Salzburg and Twente trace group respectively, because less input bytes can be analyzed. Hence the amount of rejections for these test cannot be compared to the amount of rejections with the long input configuration. 7 hash functions (BOB, Hsieh, MD5, RS, SBOX, TWMX) show no dependency. With only 1 rejected test CRC32, FHN32 and SHA perform reasonable well.

(a) IP + Transport Header

	AP	BKDR	<b>BOB</b>	BRP	CRC32	CS	DEK	DJB	FNV32	Hsieh	JS	MD5	MMH	NDJB	<b>OAAT</b>	Ocaml	PY	RS	<b>SBOX</b>	SDBM	<b>SHA</b>	Simple	SML	STL	<b>TWMMX</b>
FH Salz (1071)	43	33	<b>0</b>	43	7	161	782	50	9	2	16	7	768	18	<b>0</b>	47	1	1	<b>0</b>	25	<b>0</b>	36	21	33	0
Twente (2100)	87	45	<b>0</b>	102	6	1814	2014	171	0	0	171	0	1906	87	<b>0</b>	109	106	1	<b>0</b>	97	<b>0</b>	60	103	100	1

(b) Recommended 8 Bytes

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

Table 11: Global Rejections of Independence Test For Byte Values

## 4.7 Conclusion Hash Function Evaluation

In this chapter each of the 25 hash functions were analyzed on their suitability for hash-based selection. The hash functions were evaluated on 4 criteria: fast calculation time, non-linearity, unbiasedness and representativeness of the sampled subset.

**Hash Function Quality depending on Hash Input Length** Whereas the calculation time for SHA and MD5 hash functions are close to constant, the calculation time of non-cryptographic hash functions decreases significantly when less bytes are used for hash input. A contrary problem shows the avalanche criterion evaluation. In case less hash input bits are used, the avalanche declines for some hash functions and the hash values are not distributed randomly. This notion is also fortified by the unbiasedness and representativeness evaluation based on the chi-square independence test. For the analyzed short 8 bytes hash input configuration several hash functions (SIMPLE, SML, BKDR) are not able to select an unbiased and representative subset of the population. The STL hash function did not even produce a hash value in the selection range. Nevertheless the quality of some hash functions is not dependent on the hash input length as could be observed with BOB, OOAT, SBOX, MD5, SHA, Hsieh, Simple.

**Traffic Trace** The traffic trace that is used for the hash input has an influence on the quality of the selection decision. The amount of rejected global tests for the Twente traces are significantly greater than for the FH Salzburg traces. This can be reasoned by the bigger size of collisions within the Twente traces and the longer measurement interval. Because the traffic trace dependency of the results

	BOB	OAAT	SBOX	TWMMX	Hsieh	RS	SHA	MD5	FNV32	CRC32	PY	Simple	SDBM	BKDR	AP	SML	NDJB	BRP	DJB	JS	Ocaml	CS	MMH	DEK	STL
Rejected Chi Square Independence Test - Long Hash Input Configuration																									
Length	0	0	0	0	0	1	0	0	0	0	3	0	3	0	3	0	1	6	8	5	6	63	120	137	1
Protocol	0	0	0	0	0	0	2	0	0	2	1	4	2	5	5	3	14	11	7	4	81	16	83	1	
Byte	0	0	0	1	2	2	0	7	9	13	107	96	122	78	130	124	105	145	221	187	156	1975	2674	2796	133
Rejected Chi Square Independence Test - Short Hash Input Configuration																									
Length	0	0	1	0	1	0	0	0	1	0	1	21	27	38	35	38	43	47	37	42	65	57	16	147	x
Protocol	0	0	0	0	0	0	0	0	0	4	30	46	49	45	59	44	39	65	40	69	33	34	132	x	
Byte	0	0	0	0	0	0	1	0	1	1	23	182	261	300	292	291	354	352	276	366	716	727	534	924	x

Table 12: Summary Independence Test Results

are of major concern the chapter ?? is devoted to the analysis of the applicability of hash-based selection for different traffic trace classes.

**Linear Hash Functions** The CRC32 is the only linear hash function that shows good results in the independence tests for packet length and transport protocol. The test for independence for all bytes show in 13 tests bias in the sampling decision. CRC32 calculation time is awfully slow in the software implementation but available hardware will narrow the discrepancy between CRC32 and the other hash functions. Nevertheless CRC32 is not suitable for hash-based packet selection because of security issues (as explained in Sect. ??). The other linear hash functions DEK, BRP and AP perform poorly in the independence tests.

**Cryptographic Hash Function** Both cryptographic hash functions SHA and MD5 show perfect avalanche, but for SHA two tests reveal that there is some dependency between sampling decision and the transport protocol which causes the selection of biased subsets. The MD5 hash function has 7 rejected independence tests on sampling decision and byte value. This slack in performance may be caused by the trimming to 32bit output values. Both cryptographic hash functions imply heavy processing time which is undesired for hash-based selection.

**Recommended Functions For Hash-Based Selection** The only functions that master the tests without a flaw are the BOB and OAAT hash function. Both hash functions show very good avalanche and not a single chi-square test is rejected for any of the two hash input configurations. These results testify that both hash functions sample an unbiased and representative subset of packets. Other functions with few rejected tests are SBOX (1), TWMX(1), Hsieh(3), RS(3), SHA(3), MD5(7). All of these hash functions passed the linearity test with good and very good avalanche results. For the long hash input configuration the SBOX hash function has no rejected test, but sampled for the 8 byte hash input configuration one subset that is biased to length. For the short hash input configuration the RS, TWMX and MD5 hash function sampled always representative and unbiased subsets.

The test results show that an 8 byte hash input configuration is sufficient for hash-based selection. Based on the performance test results, one should discourage the use of the cryptographic functions MD5 and MD5 because they are about 30 and 50 times slower than the other recommended functions for an 8 byte hash input configuration. This implies that measurement nodes that use SHA and MD5 need to provide 3000% and 5000% higher processing capacity

Function	BOB	OAAT	Hsieh	SBOX	RS	TWMX	MD5	SHA	FN32	CRC32	NDJB	PY	SDEM	SML	BRP	JS	Ocaml	STL	AP	BKDR	Simple	DJB	CS	DEK	MMH
Performance	++	++	++	+	++	++	.	+	++	+	++	++	++	++	++	++	++	++	++	++	++	++	++	++	++
Avalanche	++	++	++	++	+	+	++	++	+	+	+	+	+	+	+	+	+	+	+	+	++	+	+	+	+
Bias	++	++	+	+	+	+	++	+	+	+	.	+	+	+	+	+	+	+	+	+	+	+	+	+	+

Table 13: Summarized Test Results

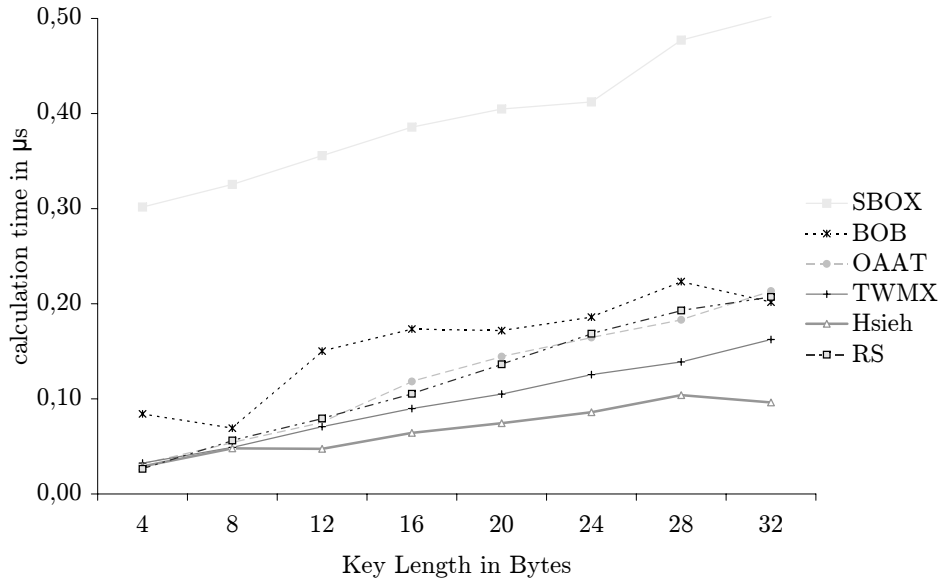


Figure 22: Performance of Recommended Hash Functions

than with the other functions. The SBOX (Substitution Box) hash function consists of a static table with 256 random generated 32 bit value entries. Consecutively each hash input byte is substituted by the random number within the byte's value table entry and the different values are concatenated with XOR and some bit multiplication. This makes reconfiguration easy as only the table of random generated values has to be replaced when too many information about the hash function leaks to an adversary. Nevertheless SBOX is about 10 times slower than BOB, OAAT, Hsieh, RS and TWMX which are recommended for low-resource and slow-performance measurement nodes. A distinguishable view on the performance of these recommended hash functions is shown in Fig. ??.

## Chapter 5

# Applicability of Hash-Based Selection on other Traces

In the previous chapters the intention of the evaluation was to gain input parameters for the hash-based selection technique. As a result a hash input and hash function combination was recommended. The evaluation based on the Twente and FH Salzburg traces proves that with the recommended configuration the hash-based selection technique can emulate random selection in terms of representativeness and unbiasedness of the selected subset. The intention in this section is to analyze other traces in order to make an assertion about the applicability of hash-based selection for different traffic trace classes.

### 5.1 Chi-Square Independence Tests for other Traces

#### 5.1.1 Traffic Traces

In this section the chi-square independence test is conducted for the LEO 1, LEO 2 and NZIX traces. Although these traces include many identical packets that have the same selection decision it is here evaluated if the selection decision may not be significantly biased. In Sect. ?? the basket of traces was classified according to their contained applications, transport protocol and the autonomous system they were captured in. In Tab. ?? this classification is provided with additional information about collision percentage and the maximum amount of

Trace Group	Autonomous System	TCP/UDP/I CMP	Applications	Colliding Packets	Largest Collision
FH Salzburg	University	99/1/0	http(90)	0,05%	21
Twente	ISP	89/7/1	diversified	0,04%	320
LEO 1	Large ISP	90/10/0	edonkey(25) http(5)	0,04%	1077
LEO 2	Large ISP	33/60/0	tunnel(60) edonkey(10)	8,83%	113
NZIX	Peering Point	68/20/9	http(50) quake(5)	8,37%	33753

Table 14: Hash Function Collection Properties

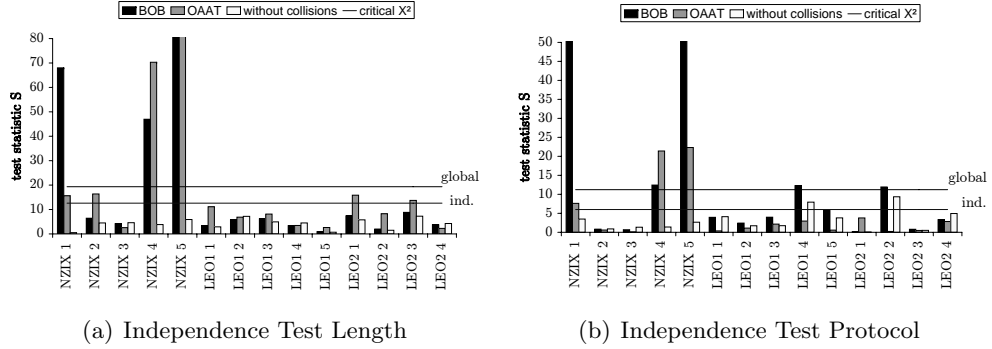


Figure 23: Independence Test Statistic T - Including and Without Collisions

packets within one collision in one trace file of 10 million packets. Because only few traces are available for each class, the evaluation results of the traces cannot be representative for their traffic class. More traces are required to obtain a more general statement.

### 5.1.2 Measurement Setup

The measurement setup is similar to the independence tests conducted in Sec. ???. It is analyzed if the selection decision based on each trace is independent to the packet attributes packet length (categorized in 7 groups) and the transport protocol. 5 NZIX traces, 5 LEO1 and 4 LEO2 traces are analyzed. The measurement interval is 2 million packets. The tests are conducted with the long hash input configuration, i.e. all packet header fields except Version/IHL, TOS, TTL, IP Checksum, Flags/Fragment Offset and IP Options. The long hash input configuration is used because the measurement results shall not be distorted by additional hash input collisions that are caused by insufficient amount of input bytes. The hash-based selection decision is based on the recommended hash functions BOB and OAAT.

### 5.1.3 Measurement Results

The measurement results of the independence tests are depicted in Fig. ??. The first and second bar in the diagram represent the test statistic T (cf. Eq. ??) for the BOB and OAAT hash function. In order to show the influence of collisions the third bar displays the test statistic T for an independence test based on the same traces but with collisions removed. These test are based on the BOB hash function. The lower horizontal line represents the critical value for the test statistic for each individual test whereas the upper horizontal line represents the Dunn-Sidak adjusted critical value (cf. Sect. ??).

It is obvious that the selection decision is dependent on the packet length and transport protocol for the NZIX traces 1, 4 and 5 if collisions are present in the traces. The fourth LEO1 trace and the second LEO 2 trace show dependency between sampling decision and protocol when the BOB hash function is used. In case colliding packets are removed prior to the test, the global test of

independence shows no dependency.

**Collisions** The tests show the influence of input collisions on the selection decision. The selection decisions based on traces including hash input collisions are significantly biased whereas the selection decision is unbiased when the collisions are removed from the traces. For all traces with high S statistics, S decreases when the hash input collisions are removed (the third bar in Fig.?? is smaller than the first bar). No global test is rejected when collisions are removed from the traces.

**Large and Small Collisions** The test results show that the selection decision is more biased for large collision. This means that the test statistic S is worse for traces that include many packets with the same hash input (one large collision) than traces with many hash input collisions consisting of few packets (many small collision). This can be seen by the different results of the LEO2 and NZIX traces. Although the LEO2 traces and NZIX traces include almost the same amount of colliding packets, the test statistic T is significantly larger for the NZIX traces, because the NZIX traces include larger collisions. This can be explained by the following argumentation: packets within one large collision all have the same packet attributes and have the same selection decision. The selection decision will be strongly biased because the colliding packets are either over or underrepresented in selected subset. Packets within small collisions are equal among themselves but different between the collisions. This implies that packets in different small collisions have different selection decisions and there is no "all or none" decision.

For scenarios where a representative subset shall be selected, traces with small collisions are better than traces with large collisions. Nevertheless it has to be kept in mind that colliding packets cannot be used for delay measurements because packet IDs cannot be correlated. A trace with all double packets may be useless for delay measurements although a representative subset is selected whereas a trace with one large collision may at least enable a delay calculation of all non-colliding packets.

**Difference OAAT and BOB Hash Function** The selection decision for traces with large collisions is biased. Nevertheless one may have the notion that either one of the hash functions may manage collisions better than the other. The hash functions have enormous differences in the test statistic T, e.g for the length attribute in the NZIX 1 trace (68.0 for BOB and 15.6 for OAAT). These differences are coincidental and depend on the proportion of large collisions that are selected. It may be possible that BOB's hash values for the 4 largest collision all fall into the selection range whereas for the OAAT hash function only one large collision is selected. As already noted prior to the tests, traces with large collisions should not be used for evaluating hash functions on their overall applicability for hash-based selection.

## 5.2 Trace Analysis

Our measurement results show that traffic traces with many and large input collisions are unsuitable for hash-based selection. Because only one or two traces are analyzed for each autonomous system class it is refrained from declaring any AS class more suspect to large input collisions than others. Instead, the reasons for double packets will be more thoroughly investigated with the intention to find applications that accumulate most collisions. For this analysis all colliding packets are extracted from the traces and the number of packets in one collision is counted. **A collision consists of packets that may only differ in the IP Time-To-Live and IP checksum field.** All collisions that include more than 20 packets are more closely investigated.

### 5.2.1 Reasons for Identical Packets

In the analysis roughly 20 groups of collision are identified. Exemplary some observations are shown in the following. Afterwards the reasons for identical packets are classified.

**NZIX NBNS protocol** [?] The largest collisions occurred in the NZIX traces between UDP packets of the NetBios Name Service (NBNS) protocol. Up to 33768 packets are identical except in TTL and Checksum. The UDP checksum and IP identification are set and non zero. There are also NBNS packets that did not collide. Whereas the source differs between different collisions, the colliding packets have the same IP destination address. There occur up to 1000 identical packets per second. Nevertheless there are also longer pauses between packets which endure minutes or hours. The packets in a collision have decreasing Time To Live values. These phenomenons lead to the conclusion that these collisions are caused by routing loops which are created by a non-reachable NBNS server.

**NZIX ICMP echo ping request** Correlated with the colliding packets of the NBNS protocol seem to be ICMP echo ping requests which have the same destination address as the NBNS packets. There occur up to 10000 identical packets in intervals of 200 packets per second, discontinued by pauses of several minutes or hours. The TTL field decreases between following observations. The IP Identification and ICMP checksum is set and non-zero. These collisions are assumed to be caused by routing loops.

**NZIX Infocrypt Shiva** Other collisions that occurred in the NZIX traces consist of UDP packets with source and destination port 2233 registered to Infocrypt which offers VPN services. The IP Identification and UDP checksum of the colliding packets are set to 0, although there are also non colliding packets with an ID different to 0. The UDP checksum might be set to 0 because of security reasons. There occur about 3 identical packets per second and the collisions include up to 80 packets. The colliding packets are not caused by routing loops, because the Time To Live field is always the same. It is assumed that these packets are different but because the IP Identification and UDP checksum is not set they are identical.

**LEO1 HSRP** In the LEO1 trace group occur colliding packets that include the UDP-based Hot Standby Routing Protocol (HSRP) [?]. The HSRP packets col-

lide in all traces. The IP identification of the colliding packets is not set and 0. The collisions include up to 300 packets per 15 min trace and occur every 3 seconds. Because for the LEO1 traces the whole packet is captured it can be assured that the packets are really identical. The packets are hello packets from the active and standby routers that verify that they are operational (heartbeat or keep-alive packets).

**LEO1 Halflife and Star Wars Galaxy** UDP packets with the source port 27015-27020 (Halflife) and 44463 (Star Wars Galaxy) collided multiple times within the LEO1 and LEO2 traces. The IP identification is set to 0, but the checksum is used. This indicates that the payload of the colliding packets is the same. The IP Time To Live field is constant which indicates normal routing operation. The packets occur about once every second. It is assumed that constant information (e.g. game server user lists) are included in these packets.

These five examples demonstrate reasons why packets collide. In the following the observed collision are categorized.

1. **Routing misconfiguration (especially routing loops)**

2. **Lack of IP identification**

- (a) Recurring packets (keep alive and hello packets, status information)
- (b) Missing transport header checksum (e.g. in VPN, IP tunneling)

3. **Coincidental colliding packets**

**1. Routing Misconfiguration** Observed packet collisions that are caused by routing configurations are ruinous for the hash-based selection technique. Large numbers of colliding packets occur in short interarrival times because packets will loop until the Time To Live value reaches 0. The short interarrival times of the colliding packets imply that most packets get the same selection decision even if the selection range is changed periodically. The NZIX traces include many 2-9 hop routing loops, where the 2 hop routing loops are the most grievous because they produce the most colliding packets. In order to ensure the representativeness of hash-based selection one has to install routing loop prevention algorithms which probably were not deployed in the NZIX measured network at that time (2000). The conducted analysis shows that most large packet collisions are caused by routing loops. Especially applications that are concerned with the initial transmission setup like name services and destination reachability (like ICMP ping) produce packets in routing loops.

**2. Lack of IP Identification** Different IP stack handling of operating systems cause packets that have an IP identification field of 0. The lack of the IP identification is especially severe when the application generates identical packets as observed with the HSRP "heartbeat" packets. These packets have the same length, include the same payload and thus the same transport header checksum. Another problem arises with packets that do not include a transport header checksum (e.g. the UDP checksum is optional). Packets from one application in a flow have the same source and destination addresses and port. Packets with

the same IP ID can only be distinguished by their packet length. This leads to packet collision at the application specific packet lengths. In the LEO1 traces this phenomenon occurred with VPN traffic where the UDP checksum was probably removed due to security reasons. In the LEO2 traces the missing UDP checksum was observed for IP tunneled packets as well, but the packets could be distinguished by their different IP ID and packet length. It is very likely that for other operating systems that do not set the IP ID field of IP tunneled packets, a high amount of collisions will be observed.

### 5.2.2 Consequences for Hash-based Selection

In order to ensure that the hash-based selection technique is applicable, the network operator should install routing loop prevention techniques at the observation points. Further it is desired that the traffic sources set their IP identification field or at least a transport header checksum. These two actions would decrease the occurrences of identical packets and ensure proper selection. In case none of these both prerequisites can be deployed in the network the measurement operator has to be aware that packets in routing loops and without IP identification are not representatively selected. The evaluation shows that the applicability may not be dependent on any traffic class but on the configuration of the observation point (routing loop prevention) and the traffic sources (IP ID handling).

Further the bias introduced by identical packets occurring with large interarrival times can be reduced by changing the selection interval periodically. The identical HSRP alive packets in the LEO1 trace occur every 3 seconds. Even if the selection range is only changed every minute only 20 identical packets have the same selection decision.

## Chapter 6

# Hash Functions for Packet ID Generation

### 6.1 Introduction

In the main part of this thesis a hash function and hash input combination was found that is suitable for hash-based packet selection. The evaluation was based on the following properties 1) fast calculation time 2) non-linearity 3) unbiasedness of the selection decision and 4) representativeness of the selected subset.

Except the first, all these properties are not necessary for packet ID generation. Nevertheless the proposed hash input and hash function configuration may prove applicable for packet ID generation as well, independent of the selection technique (e.g for random selection or hash-based selection).

Packet ID generation is necessary for multipoint measurements. Each packet traversing the observation point will be assigned a packet ID and timestamp which are exported to the multipoint collector as described in Sec. ???. If packet IDs are unique in the measured domain the packets paths can be traced according to the packet ID observations, following the timestamp of the packet. Nevertheless packet IDs are not unique; they collide either because different packets are identical or because the packet ID hash function assigns different packets the same hash value. These packet ID collisions are critical. The trajectories of duplicate packet IDs can be misinterpreted as a trace from a single packet. This leads to wrong results for delay and loss measurements. In the following it is evaluated which hash functions are suitable for packet ID generation based on the criterion of low packet ID collision probability. The goal of this analysis is to recommend a hash function for packet ID generation.

At first, a mathematical model of the collision probability will be derived and tested with a random number generator function. It is assessed if the packet ID values generated from real traffic traces and different hash functions are comparable to the random number generator in terms of packet ID collisions. Further it is discussed under which circumstances the hash-based selection hash value can be used as packet ID.

## 6.2 Approach

The quality of a hash function for packet ID generation is measured by the collision probability of the hash values. The lower the collision probability the fewer packets have to be discarded for delay measurements. In order to assess the packet ID collision probability two criteria will be evaluated 1) uniformity of hash value distribution and 2) empirical packet ID collisions. The uniformity of the hash value distribution is evaluated by the chi-square goodness-of-fit test. A mathematical model for the calculation of expected collisions is presented. Later, the empirical packet ID collision probability is compared to the model.

## 6.3 Uniformity of Hash Value Distribution

Non-uniform distributed hash values pile up at specific intervals or values and hence have a higher collision probability. The chi-square goodness-of-fit test [?] is used to analyze if the hash value distribution is significantly different from a uniform distribution. The hypotheses to test are:

**H0:** The distribution of hash values is uniform

**H1:** The distribution of hash values is not uniform

Because the amount of possible hash values  $R$  is large and only few occurrences of the same hash value are expected, the hash values are categorized in  $b$  bins that consist of  $\frac{R}{b}$  values. The test statistic  $T$  is obtained by

$$T = \sum_{i=1}^b \frac{(n_i - \bar{n})^2}{\bar{n}} \quad (6.1)$$

where  $n_i$  are the observed frequencies in bin  $i$ . By dividing the number of total observed packets  $N$  by the amount of bins  $b$ , one obtains the number of expected observations  $\bar{n}$  in each bin. The H0 hypotheses is rejected if  $T$  is above the critical  $X^2$  value with  $b-1$  degrees of freedom and an error of  $\alpha$ .

### 6.3.1 Experimental Setup

The uniform distribution evaluation is based on a single NZIX trace including  $N=10$  million unique packets, i.e. double packets are removed from the trace. The test is applied for the hash function collection summarized in Tab. ???. For the evaluation the 8 recommended bytes and long hash input configuration (IP and Transport Header except Version/IHL, TOS, TTL, IP Checksum, Flags/Fragment Offset and IP Options) are used as hash input. The hash values are categorized in  $b = 1024$  bins. Under the Null hypothesis that the hash values are uniformly distributed, the test statistic  $U$  is  $X^2$  distributed with  $b - 1$  degrees of freedom. Tolerating an error of  $\alpha = 5\%$ , H0 is rejected for a test statistics  $T$  above  $X^2(1023, 0.05) = 1098.5$ .

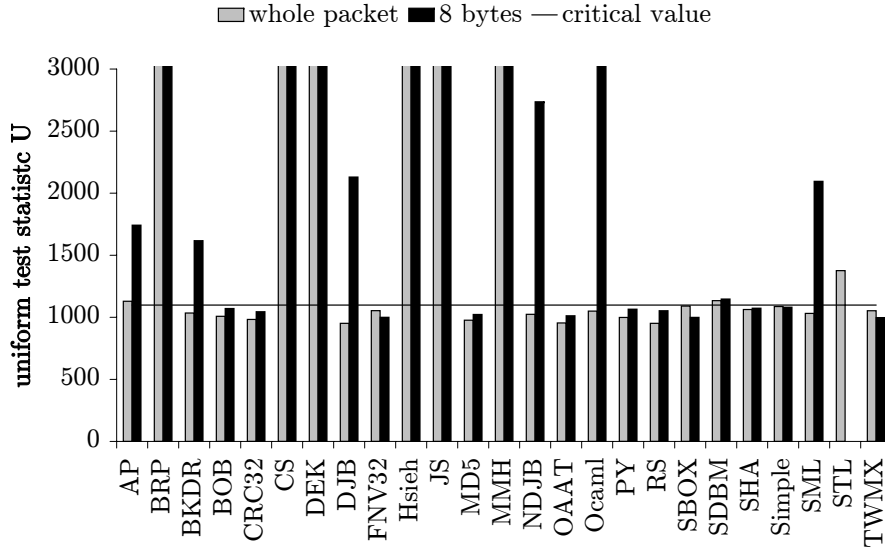


Figure 24: Uniformity Tests on 25 Hash Functions

### 6.3.2 Results

The measurement results for all 25 hash functions are depicted in Fig. ???. For every hash function two tests are conducted, based on the two different hash inputs. There are nine hash functions that did not pass the uniformity test for neither hash input configuration: AP, BRP, CS, DEK, Hsieh, JS, MMH, SDBM, STL. Five other hash functions generate uniform hash values for the long hash input configuration but not for the 8 bytes input: BKDR, DJB, NDJB, OCaml, SML. This slack in performance with decreasing hash input length was observed in the chi-square independence tests for these hash functions as well. The remaining hash functions pass the uniformity test and are very likely to include only few collisions: BOB, CRC32, FNV32, MD5, OAAT, PY, RS, SBOX, SHA, SIMPLE, TWMX. These hash functions will be further evaluated on their collision probability in Sect. ???.

## 6.4 Collision Probability

For hash-based selection it does not matter how many hash-values collide as long as the collisions are uncorrelated. In contrary, hash collisions are important for packet ID generation, because packets with colliding hash values cannot be distinguished in delay measurements and will be discarded. Here, a mathematical model is derived for the analysis of hash value collisions that are not caused by hash input collisions. Later the theoretical considerations are compared with empirical observed collisions using a set of hash functions.

## 6.5 Theoretical Probability

### 6.5.1 Initial Assumptions

In order to provide a model for the collision probability of hash values some assumptions are made:

1. The hash function provides perfect random hash values, i.e. the generated hash values are uncorrelated and independent.
2. All hash values have an equal chance  $p$  to occur, i.e. the distribution is uniform over the hash range.

In Sect. ?? it has already been shown that there are hash functions that uniformly distribute their hash values in 1024 buckets. Both assumptions are true for random number generator functions that are specially designed to provide perfect random numbers. The hash functions from the analyzed set are not specially designed for providing real random numbers. It is empirically analyzed if these hash functions comply to the model and can compete with a random number generator.

### 6.5.2 Mathematical Model for Collision Probability

#### 6.5.2.1 Parameters of the Model

A mathematical model for the amount of collisions of a perfect random hash function is derived here. The collision probability depends on:

- the number of packets  $N$  that are observed at all ingress nodes during the critical time interval  $[t_0..t_0 + t_{max}]$
- the hash range size  $R$

Duplicate packet IDs are identified at the ingress nodes of the measurement domain. Ingress nodes are incoming links from external routers or traffic sources within the measured network. In order to provide accurate delay measurements it is not necessary to discard all duplicate packet IDs, but only those occurring close together in a time interval  $[t_0..t_0 + t_{max}]$ , where  $t_0$  denotes the first occurrence of the packet ID at any ingress node. The critical time interval  $t_{max}$  is the maximum delay inside the measurement domain. Assessing the maximum delay is tedious and influences the delay measurement. If the maximum delay is chosen too large the collector 1) has to store and loop through immense amounts of packet IDs before it can purge old reports and 2) discards more packets than are necessary. If the maximum delay  $t_{max}$  is assessed too low, it is possible that wrong packet ID pairs are formed and the delay measurement becomes inaccurate.

The amount of packets  $N$  that are generated during the critical time interval at all  $g$  ingress each with a data rate  $B_i$  can be calculated using the incoming data rate  $B = \sum_{i=1}^g B_i$ , the average packet length  $\bar{l}$  and the maximum domain delay  $t_{max}$ :

$$N = \frac{B \cdot t_{max}}{\bar{l}} \quad (6.2)$$

A conservative example that uses high delay and bandwidth figures shows the order of magnitude of packets within the critical time interval. Assuming a 1 second maximum delay  $t_{max}$  in a measurement domain with four OC-48 backbone ingress links each with about 2500 Mbit/s transmission capacity and an average packet size of 500 bytes, 2.5 million packets are included in the critical collision interval.

### 6.5.2.2 Collision Model

The hash value distribution can be obtained by using a two step urn model. First the probabilities are calculated that one hash value occurs  $m$  times when  $N$  hash values are generated. Second, these probabilities for one hash value are transferred to all hash values to gain the distribution of unique hash values.

**First step:** From a number of  $R$  values,  $N$  values are drawn with repetitions. A random variable  $X$  is defined:

$X$  = amount of draws of the specific hash value  $i$

The probability distribution of  $X$  is given by the binomial distribution  $B(m|p, N)$  [?]:

$$B(m|p, N) = P(X = m) = \binom{N}{m} p^m (1 - p)^{N-m} \quad (6.3)$$

where  $p = \frac{1}{R}$  is the success probability that the specific value  $i$  is drawn in one try. Because the amount of packets  $N$  generated during the critical measurement interval is usually very large ( $N > 10^4$ ), the binomial coefficients can only be tediously calculated. As a rule of thumb the binomial distribution can be approximated by the poisson distribution for large  $N > 50$  and  $p < 0.05$  and  $Np < 9$  [?]:

$$P(X = m) = \binom{N}{m} p^m (1 - p)^{N-m} \approx \frac{\lambda^m}{m!} e^{-\lambda}, \quad (6.4)$$

where  $\lambda = Np$ . Equation ?? states the probabilities that a specific hash value  $i$  occurs  $m$  times.

**Second Step:** In the following the intention is to derive a statement about all hash values, i.e. the amount of non-colliding hash values in the measurement interval. There are two approaches to derive the amount of non-colliding hash values:

**1st Approach:** The experiment with the random variable  $X$  is repeated for all  $R$  hash values. One can define a new random variable  $Y$  which is the amount of unique hash values, i.e the amount of experiments where  $X_i = 1$ . The success probability  $p_y$  is the probability that the hash value is unique  $P(X_i = 1)$ . There are two problems with this approach:

1. The  $R$  different experiments  $X_i$  are not really independent. In case in the first  $Z$  experiments ( $X_1 \dots X_{Z < R} \dots X_R$ ) a total amount of  $N$  hash

values are generated, then the remaining hash values  $[Z..R]$  cannot possibly occur. Because the experiments are not independent one cannot use the binomial distribution to model the amount of unique hash values.

2. The model does not describe the real experiment. Although the approach gives the expected amount of collisions the variance is overvalued. In the model there is a chance that more unique hash values occur than there are in the critical measurement interval. The experiment is repeated  $R$  times each with a success probability of  $P(X = 1)$  which means that  $Y$  can obtain values from  $1..R$ . This also implies that the variance of unique hash values is distorted.

Because of these two reasons this approach should not be used.

**2nd Approach:** A better approach to gain a distribution of non-colliding hash values in the critical measurement interval  $N$  is the following. First it is assumed that already  $N - 1$  hash values are generated. All hash values in the hash range  $R$  are then binomial  $B(m|N - 1, \frac{1}{R})$  distributed. The  $N$ th hash value is generated and it is observed how many times this hash value has been already drawn. If the hash value has not been priorly drawn than a unique hash value has been generated. The probability that the hash value has not been generated before is approximately  $P(X = 0)$ , because  $N$  is large. Moreover this conclusion can be applied for all other  $N-1$  hash values. The probabilities that each of the hash values has not been drawn or will not be generated is given by  $P(X = 0)$ . This approach to model the distribution of unique hash values will be used in the following.

A new random variable  $Y$  is defined:

$Y$  = the amount of unique hash values in the critical measurement interval, i.e experiments where  $X_i = 0$  for  $i = 1..N$ . The success probability to draw a unique hash value is  $p_y = P(X = 0)$ . Because  $N \ll R$  all experiments  $X_i$  can be assumed to be independent and  $Y$  can be modeled as binomial distributed  $Y \sim B(m|N, p_y = P(X = 0))$ . The expected amount of unique hash values and the variance is:

$$\mu = Np = Ne^{-\frac{N}{R}} \tag{6.5}$$

$$\sigma^2 = Np(1 - p) = N(e^{-\frac{N}{R}} - e^{-\frac{2N}{R}}) \tag{6.6}$$

The amount of unique hash values  $U_q$  at least expected in  $q\%$  of the cases can now be estimated by either 1) calculating the  $1 - q$  quantil of the binomial distribution  $P(Y < U_q) = 1 - q$  or 2) using the lower bound of chebychev's inequality  $P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2} = 2q - 1$ . The  $1 - q$  quantil of the binomial distribution is tedious to calculate because the binomial coefficients are large. Chebychev's inequality provides an easy calculable bound for any distribution of values around the mean where the distributions variance is known.

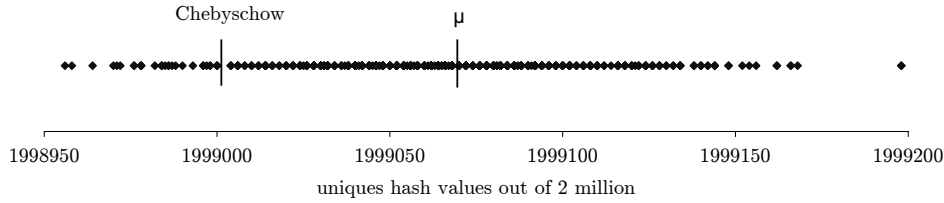


Figure 25: Unique Hash Values - 500 Tests with 2 Million Random Values

## 6.6 Empirical Collision Probability

A random number generator will be used to verify the mathematical model. Afterwards, hash functions that performed well in the uniformity tests in Sec. ?? will be analyzed on their packet ID collision probability by means of their derivation of the mathematical model. The Hsieh hash function has shown promising performance in the hash-based selection evaluation and will be analyzed as well, although Hsieh performed poor in the uniformity tests.

### 6.6.1 Measurement Setup

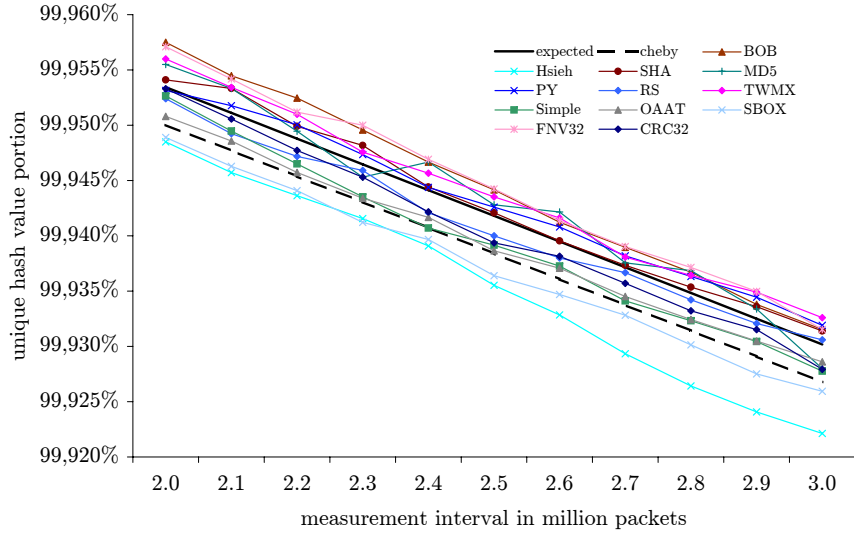
The Mersenne Twister (mt19937) random number generator from the GNU scientific library [?] is used to obtain 32bit random values. In 500 tests the amount of unique hash values from 2 million random generated values are counted.

The hash function evaluation is based on one NZIX trace (20000706-060000) from which all collisions are removed. Different measurement intervals ranging from 2 to 3 million packets are used for each hash function. As the hash input two configurations will be used 1) the IP and Transport header (except Version/IHL, TOS, TTL, IP Checksum and Flags/Fragment Offset) and 2) the recommended 8 bytes from Sect. ??.

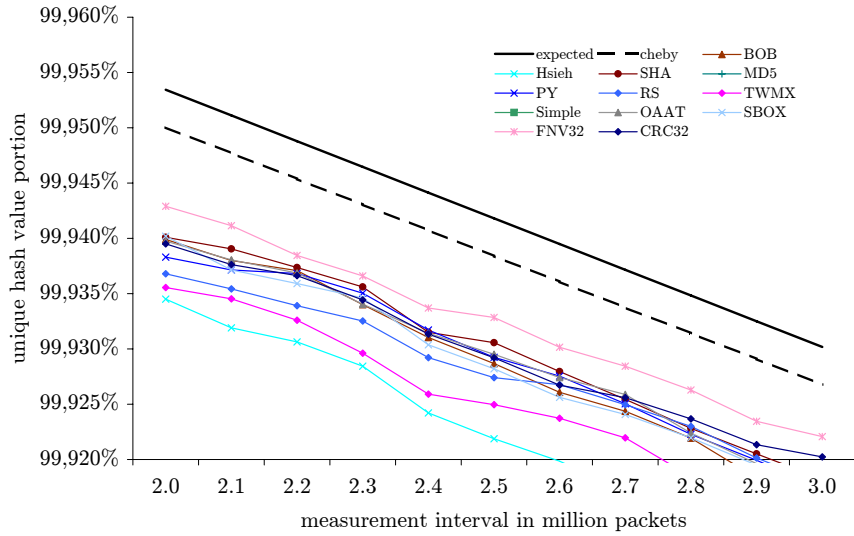
### 6.6.2 Measurement Results

**Evaluation of Model using Random Generator** In order to verify the model for hash collisions described in Sec. ?? the unique hash values obtained by a random number generator were counted. For 2 million generated 32bit values the expected amount of unique hash values is (see Eq. ??)  $\mu = 1999069$ . The mean amount of empirical measured unique hash values in the conducted 500 runs is  $\bar{U} = 1999066$ . The measurement results are depicted in Fig. ?. This figure also shows the lower bounds for the expected amount of unique hash values in  $b=90\%$  of the cases using chebyshev's inequality. 32 tests (=6.4%) show less unique hash values than the chebyshev 90% bound. The results are consistent with the mathematical model.

**Evaluation of Hash Functions for Packet ID Generation** The percentage of non-colliding packet IDs for different measurement intervals from 2-3 million packets using the 12 analyzed hash functions is depicted in Fig. ?. The expected percentage of usable packets  $P(X=0)$  is shown by the straight black linear line.



(a) IP + Transport Header Hash Input



(b) 8 Bytes Hash Input

Figure 26: Percentage of Non Colliding Packet IDs

In order to represent the variance, the expected least amount of non-colliding packets in 90% of the cases is calculated using chebyshev's inequality, shown as the black dashed line. When the IP and Transport header are used as the hash input (cf. Fig. ??) only the Hsieh and SBOX are outside the chebyshev bound implying that they generate significantly more colliding packet IDs than is expected. The lowest collision probability is measured for the BOB and FNV32 hash function.

When only the 8 recommended bytes are used as the hash input (cf. Fig. ??), more packet IDs collide than expected. The unique hash value proportion drops around 0.015%. This can be explained by the additional amount of input collisions that are caused by the insufficient amount of hash input bytes.

## 6.7 Packet ID and Selection Hash Value

When applying hash-based selection, it may or it may not be reasonable to use the same hash value which is intended for the sampling decision to use as packet ID. Using the same hash value as packet ID will relieve the measurement points processing capacities, because only one hash value will be calculated for the selected packets. On the other hand it will increase the collision probability of the packet IDs. The hash-based selected packets possess hash values which fall into the selection range S instead of the whole possible hash range R. This is also obvious by looking at Eq. ?. The amount of unique hash values from sN packets when two hash functions are used:

$$U = sNe^{\frac{sN}{R}} \quad (6.7)$$

The amount of unique hash values from sN packets using one hash function for packet ID generation and selection is:

$$U = sNe^{\frac{sN}{sR}} \quad (6.8)$$

Since the hash domain R is decreased to sR, the collision probability is increased. This entails that more packets are discarded because of identical packet IDs. The increase of collision probability is the reason why Duffield and Grossglauser [?] Molina Niccolini [?] recommend to use a different hash function for packet ID generation and packet selection.

It is here argued that it does not make sense in every case to use different hash functions. For example, in a theoretical scenario with a sampling fraction close to 1 the attained collision improvement does not justify the processing demands for a new hash value calculation for almost all packets. Instead of generating packet IDs for the sampled packets one may just select additional packets in order to compensate for the additional collisions. Of course the additionally selected packets increase the measurement traffic. A short example will show the trade-off. Assuming an amount of 2,5 millions packets within the critical measurement interval and a selection fraction of s=10% the amount of additional collisions C is:

$$C = sNe^{-\frac{sN}{R}} - sNe^{-\frac{sN}{sR}} = 131 \quad (6.9)$$

This means that instead of calculating an extra hash value on 250000 packets one can just select  $\approx 131$  packets additionally. In this example the measurement

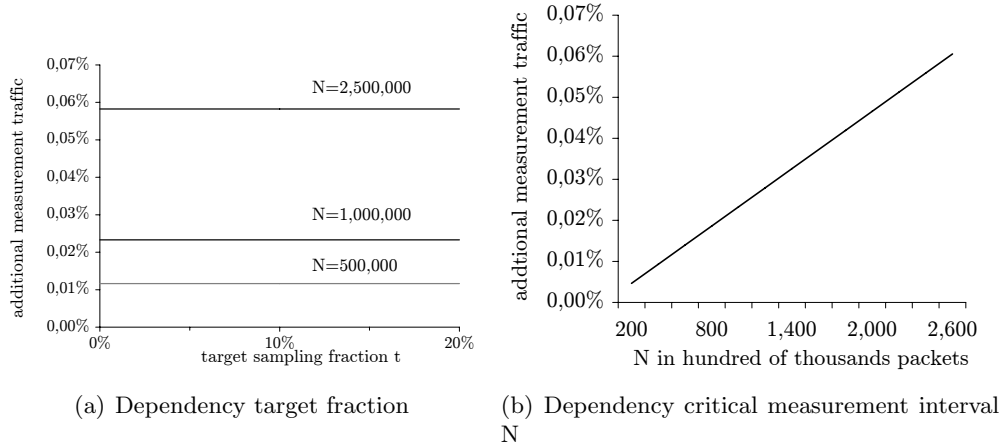


Figure 27: Additional Traffic in One Hash Function Scenario

operator has to decide between an increase of hash value calculation time of 10% or an increase of measurement traffic of  $\frac{C}{sN} = 0.052\%$  assuming both hash functions have the same calculation time. This value only gives an approximate figure of measurement increase because the additional selected packets have a chance to collide as well.

For a more accurate value one has to look at a more practical question: How does the selection range need to be configured to gain a certain target selection fraction  $t$  of unique (and usable) packets?

### 6.7.1 Selection Range Adjustment for One Hash Function

In the case that only one hash function is used for hash-based selection and packet ID generation, the selection range fraction of the hash range  $s_1$  has to be configured according to Eq. ?? in order to gain a target sampling fraction  $t$  of usable packets.

$$tN = s_1 N e^{-\frac{s_1 N}{s_1 R}} \rightarrow s_1 = t e^{\frac{N}{R}} \quad (6.10)$$

Additionally to the target sampling fraction a proportion of  $\frac{s_1 - t}{t}$  packets are required to compensate for the packet ID collisions. The additional traffic proportion is constant for all target selection fractions (cf. Fig. ??) and only depends on the amount of packets in the critical measurement interval  $N$  (cf. Fig. ??):

$$\frac{s_1 - t}{t} = e^{\frac{N}{R}} - 1 \quad (6.11)$$

### 6.7.2 Selection Range Adjustment for Two Hash Functions

In the case that two different hash functions are used for hash-based selection and packet ID generation there are still packet ID collisions. Therefore it is

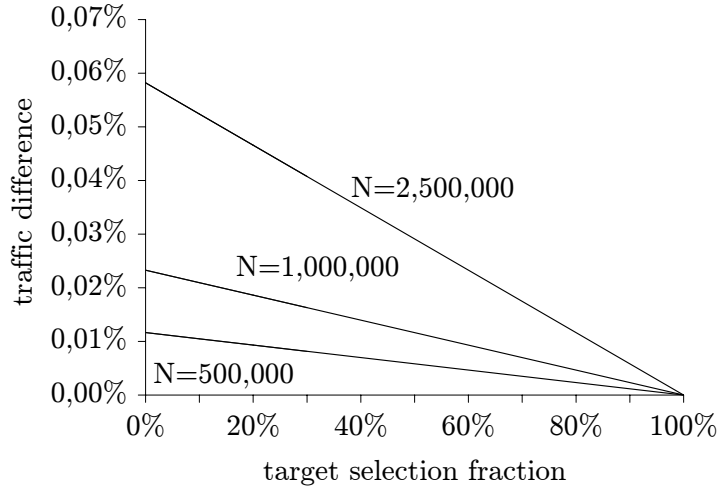


Figure 28: Measurement Traffic Increase using 1 instead of 2 Hash Functions

still required to adjust the selection range  $s_2$  in order to gain a target sampling fraction  $t$  of usable packets.

$$t = s_2 e^{-\frac{s_2 N}{R}} \quad (6.12)$$

Unfortunately Eq. ?? can only be numerically solved for  $s$ . For small target selection fractions and small measurement intervals there exist few collisions and the additional required traffic for compensating the collisions is negligible. The higher the measurement interval and sampling fraction, the higher the amount of colliding packets and the required additional packets.

### 6.7.3 Difference Between One and Two Hash Function Approach

The selection range configuration differs if either one or two hash functions are used for packet ID generation and hash-based selection. The ratio  $\frac{s_1 - s_2}{s_2}$  denotes the proportional difference of measurement traffic between both approaches. This ratio depends on the target fraction of usable packets as depicted in Fig.?. When only one hash function is used for packet ID generation and hash-based selection the measurement traffic increases. The increase is negligible for high target sampling ratios and small amounts of packets within the critical measurement interval  $N$ . Even for large amounts of packets within the critical time interval ( $N = 2.5\text{million}$ ) and small target selection fraction ( $t \rightarrow 0$ ), the measurement traffic difference is less than 0.06%.

### 6.7.4 Conclusion

**Collision Model** The evaluation results based on the random number generator have shown that the packet ID collision probability can be described with the mathematical model. A lower bound of unique packets can be calculated by the chebyshev inequality.

**Hash Functions Recommended for packet ID generation** The second part of the evaluation shows that a set of hash functions (BOB, FNV32, PY, RS, SHA, MD5, Simple, TWMX, OAAT, CRC32) can be used for packet ID generation because their packet ID collision probability is statistical not significantly different from a random number generator. In the conducted tests the BOB and FNV32 hash function performed best, but in bounds of random variance.

**Hash Input and Packet ID generation** It has to be accentuated that the hash input has to be unique in order to gain these low packet ID collision probabilities. This can be achieved by using high entropy bytes from Sec. ?? and by configuring the measured network as proposed in ?. The 8 bytes that are recommended for hash-based selection increases the amount of hash input collisions and more packets need to be discarded, i.e. the packet ID collisions cannot be assessed with the random generator model. For the analyzed trace 0.015% packets additionally collided. This small increase can be compensated by selecting some more packets.

**Packet ID and Hash-based Selection** Contrary to current approaches it has been analyzed to use only one instead of two hash function for packet ID generation and hash-based selection. The trade-off between both approaches is compared in terms of additional traffic and processing capacities. Whereas the two hash functions approach implies lower measurement traffic it requires the additional calculation of packet IDs. Using the mathematical model derived in Sec. ?? it has been shown for both approaches how the selection range has to be adjusted to gain a certain target sampling fraction of non-colliding and usable packets. With these results one can calculate the additional traffic that is required when only one hash function is used for packet ID generation and hash-based selection. Figure ?? depicts the difference in measurement traffic when only one hash function is used.

In measurement scenarios where the transmission capacities are scarce, the measurement operator will use a very small selection fraction to reduce the measurement traffic. In this scenario it is applicable to calculate another packet ID on the few selected packets in order to relieve the measurement network. In case the transmission capacities are not as scarce (e.g. with a dedicated measurement network) and the sampling fraction is higher, it is recommended to use only one hash value for the selection decision and the packet ID.

# Chapter 7

## Conclusion

In this work the hash-based selection technique has been presented as a part of the measurement process in multipoint measurements. The main advantage of passive multipoint measurements is to observe the path and delay of the measured traffic, instead of injecting artificial probe packets into the network that may distort measurements. Further passive multipoint measurements enable traceback systems in order to unveil sources of malicious packets. Nevertheless, passive multipoint measurements are a main measurement challenge because immense amounts of data are exported to a common collector. Different selection techniques like random and systematic sampling techniques are compared and discussed on their applicability for multipoint measurements in order to provide solutions to relieve the multipoint measurement infrastructure.

The advantage of hash-based selection is that the selection decision is consistent throughout the measured network, because the selection is based on a deterministic hash function on the packet content. Selected packets become traceable and one can calculate multipoint metrics like delay without storing, exporting and correlating all but only a configurable amount of packets. Because the selection technique is solely based on the packet content it is suspect to bias that is undesired for estimations of traffic characteristics. In this work it has been evaluated which combination of packet content and hash function is most suitable for selecting an unbiased and representative subset of the observed packet stream.

### 7.1 Packet Content Evaluation

The influence of the packet content, used as hash input, on the biasedness of the selection decision was analyzed. Input collisions are undesirable because the selection decision of packets within a collision is not independent and leads to bias in the selected sample. In order to find a composition of bytes that are unique for each packet a collection of IPv4 and IPv6 trace groups are analyzed on the entropy of packet bytes. The higher the entropy of the packets byte the higher the probability that the packets differ in this byte. The entropy analysis is conducted on a collection of 7 recorded traffic trace groups with different characteristics. Based on this evaluation a composition of 8 high entropy bytes has been proposed for the use in hash-based selection. This 8 byte combination was compared to a 16 byte combination proposed by Molina [?] in terms of packets

in the largest collisions. The 8 byte combination showed less input collision for almost all traces. As a result fewer bytes can be used as the hash input which will reduce the calculation effort compared to alternative methods.

These results are published in [?]. This paper is accepted for the 9th Passive and Active Measurement (PAM) Conference April, 2008 in Cleveland, Ohio.

## 7.2 Hash Functions for Hash-Based Selection

A collection of 25 hash functions were analyzed on their suitability for hash-based selection. The elaboration is based on a set of four desired properties: performance, avalanche, biasedness and representativeness of the selected subset. Concerning the calculation time, cryptographic hash functions are slower compared to non-cryptographic functions and are therefore unsuitable for low-resource environments. The calculation effort of non-cryptographic functions increases significantly for longer hash inputs which fortifies the demand for short hash inputs. In the contrary, the mixing quality, assessed by the avalanche evaluation, decreases for many hash functions when fewer hash input bytes are used. The unbiasedness and representativeness of the selected subset was evaluated on the basis of 2 hash input configurations using the chi-square independence test. The evaluation was based on two traffic traces that include few hash input collisions. The evaluation shows that for some functions the selection decision becomes dependent to the packet content for shorter hash inputs. The BOB and OAAT hash functions select a representative subset of the observed packet stream, no matter which of the two hash inputs are used. Therefore, both hash function are recommended for the use in hash-based selection.

## 7.3 Emulation of Random Sampling

The evaluation proves that hash-based selection can emulate random sampling with the recommended hash function and hash input configuration. Hash-based selection can emulate two important features of random sampling 1) independent sampling decision and 2) representativeness of the sampled subset. Nevertheless the quality of the emulation is strongly dependent on the network traffic and the analyzed metric. It was shown that many identical packets in traffic traces can lead to a biased selection. In case the variability of header fields in the network traffic is insufficient the unbiasedness and representativeness of the selected subset for a metric cannot be ensured.

## 7.4 Influence of Hash Input Collisions

Hash input collisions were identified as a source of bias. Packets within the same input collision are not independently selected. For the NZIX traces the amount of collision were so immense that no representative subset could be selected for any hash input and hash function combination. In further analysis it has been shown that most identical packets in the analyzed traces are caused by routing

misconfiguration and the non-use of the IP ID field. Those collisions can be prevented by setting the IP ID field and using smart routing loop prevention algorithms. Further a periodic change of the selection range interval may weaken the influence of identical heartbeat packets that occur in large interarrival times.

## 7.5 Hash Functions for Packet ID generation

There are different criteria for hash functions intended for hash-based selection and packet ID generation. Packet ID generating functions require low collision probability. A random number generator possesses low collision probability, but the packet ID cannot be a random generated value, because it is calculated over the packet content. Our evaluation based on real traffic traces showed that some hash functions have very similar packet ID collision probabilities like a random number generator in absence of hash input collisions. Especially the BOB and FNV32 hash functions showed good results and are recommended for packet ID generation.

It has been shown that for a variety of scenario the hash-based selection hash-value can be used for packet ID generation as well. The increased collision probability can be compensated by few additionally selected packets. In case the 8 bytes recommended for hash-based selection are used as hash input for packet ID generation, the amount of packet ID collision increase, but still proves sufficient for packet ID generation.

## 7.6 Further Work

The entropy based measurement results show that some packet content fields are more suitable for hash-based selection than others. The presented evaluation was only based on single bytes but disregarded the correlation between packet fields. Targeting a set of input bytes that includes no hash input collisions, entropy tests are planned that investigate the correlation between packet header fields in order to gain a more distinguished view on each field's variability. It is argued that byte combinations that include less entropy than their average entropy are correlated.

When IPv6 traffic traces become available that include transport headers, it should be further evaluated if the transport protocols used with IPv6 show similar entropy as with IPv4 in order to recommend an hash input configuration for IPv6. Another field of study is the practical implementation of hash-based selection. The hash-based selection technique based on the recommended hash functions has been implemented in OpenIMP [?] an open source measurement platform. A possible scenario is to use the OpenIMP library in a Wireless Mesh in order to observe packet traces, delay and points of packet loss. These measurements can help to develop ideas on how to improve routing policies and hosts reachability.



# Appendix A

## Acronyms

<b>BPF</b>	Berkeley Packet Filter
<b>DoS</b>	Denial of Service
<b>GPS</b>	Global Positioning System
<b>GSL</b>	Gnu Scientific Library
<b>HSRP</b>	Hot Standby Routing Protocol
<b>HTTP</b>	Hypertext Transfer Protocol
<b>ID</b>	Identification
<b>ICMP</b>	Internet Control Message Protocol
<b>IETF</b>	Internet Engineering Task Force
<b>IXP</b>	Internet Exchange Point
<b>IPv4 / IPv6</b>	Internet Protocol Version 4/6
<b>ISP</b>	Internet Service Provider
<b>IHL</b>	IP Header Length
<b>LSB</b>	Least Significant Byte
<b>MSB</b>	Most Significant Byte
<b>NSIS</b>	Next Steps in Signaling Working group
<b>NTLP</b>	NSIS Transport Protocol Protocol
<b>H0</b>	Nullhypothesis
<b>OS</b>	Operating System
<b>PSAMP</b>	Packet Sampling
<b>PTP</b>	Precision Timing Protocol
<b>PRF</b>	Pseudorandom Function
<b>RTP</b>	Real-Time Transport Protocol
<b>TTL</b>	Time To Live value of an IP packet
<b>TCP</b>	Transmission Control Protocol
<b>TOS</b>	Type of Service
<b>UDP</b>	User Datagram Protocol
<b>VPN</b>	Virtual Private Network
<b>WAN</b>	Wide Area Network



# Appendix B

## Symbols

$\alpha$	test error
$\alpha_{global}$	global test error
$\alpha_{ind}$	individual error that is adjusted to comply with $\alpha_{global}$
$\mu$	expected value of unique hash values
$\sigma^2$	variance of unique hash values
$c$	selected packet content used as hash input
$E$	information efficiency
$h()$	hash function that maps hash input to hash value
$H(B)$	Entropy of discrete byte variant
$H_{max}$	maximum entropy
$k$	secret key
$L$	number of possible values for one packet attribute
$N$	packets in the critical measurement interval $[t_0..t_{max}]$
$p$	probability that hash value occurs $\frac{1}{R}$
$p_y$	probability that a unique hash value occurs
$R$	hash range
$S$	selection range
$s$	selection range proportion of hash range
$s_1$	selection range when only 1 hash value is used for packet ID and selection
$s_2$	selection range when 2 hash values are used for packet ID and selection
$t$	target sampling fraction
$T$	test statistic for chi square tests
$t_0$	time of first occurrence of a packet with specific packet ID
$t_{max}$	maximum packet delay in measured domain
$T_r$	relative test statistic
$U_q$	amount of unique values at least expected in q% of the cases
$X$	random variable; amount of draws of specific hash value
$Y$	random variable; amount of unique hash values



# Erklärung der Urheberschaft

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ort, Datum

Unterschrift