Distributed Monitoring of Conditional Entropy for Network Anomaly Detection[⋆]

Chrisil Arackaparambil, Sergey Bratus, Joshua Brody, Anna Shubina

Department of Computer Science, Dartmouth College
Hanover, NH 03755, USA
{cja, sergey, jbrody, ashubina}@cs.dartmouth.edu

Dartmouth Computer Science Technical Report TR2009-653

Abstract. Monitoring the empirical Shannon entropy of a feature in a network packet stream has previously been shown to be useful in detecting anomalies in the network traffic. Entropy is an information-theoretic statistic that measures the variability of the feature under consideration. Anomalous activity in network traffic can be captured by detecting changes in this variability.

There are several challenges, however, in monitoring this statistic. Computing the statistic efficiently is non-trivial. Further, when monitoring multiple features, the streaming algorithms proposed previously would likely fail to keep up with the ever-increasing channel bandwidth of network traffic streams. There is also the concern that an adversary could attempt to mask the effect of his attacks on variability by a mimicry attack disguising his traffic to mimic the distribution of normal traffic in the network, thus avoiding detection by an entropy monitoring sensor. Also, the high rate of false positives is a big problem with Intrusion Detection Systems, and the case of entropy monitoring is no different.

In this work we propose a way to address the above challenges. First, we leverage recent progress in sketching algorithms to develop a *distributed* approach for computing entropic statistics accurately, at reasonable memory costs. Secondly, we propose monitoring not only regular entropy, but the related statistic of conditional entropy, as a more reliable measure in detecting anomalies. Lastly, we implement our approach and evaluate it with real data collected at the link layer of an 802.11 wireless network. To our knowledge, this is the first time entropy-based approaches have been considered for this kind of traffic.

[⋆] This research program is a part of the Institute for Security Technology Studies, supported by Intel Corporation, NSF grants CCF-0448277 and EIA-98-02068, and by Award number NBCH2050002 from the U.S. Department of Homeland Security, Science and Technology Directorate. Points of view in this document are those of the authors and do not necessarily represent the official position of the U.S. Department of Homeland Security, Intel Corporation, or any other sponsor.

1 Introduction

Reliably detecting attack-related anomalies in network traffic is known to be a hard problem. There are many kinds of attacks that are likely to manifest themselves as traffic anomalies. Some examples of such attacks are: a denial of service (DoS) attack, interference or jamming by an attacker, "scanning" activity by reconnaissance tools or botnet nodes, and covert channels that make unintended use of (unused) features of network protocols to communicate data out of a network. Several approaches in recent research have focused on monitoring various statistics of the traffic stream [15, 17, 22]. The property desired of such a statistic is that it should capture a characteristic of the traffic distribution that would reveal anomalies. The statistic value is monitored for deviations from "normal" values as ascertained by training with prior observed data and continuous adjustment by personnel at the Network Operations Center (NOC). Several features of the observed stream may be monitored in conjunction to improve reliability of detection.

The information-theoretic statistic of empirical Shannon entropy (or simply entropy) has received a lot of attention in this respect [4, 10, 16–19, 21, 22]. Computing entropy in the straightforward manner, by maintaining counters to keep track of the distribution histogram, is expensive memory-wise and computationally. Lall et al [16] showed that computing entropy either deterministically or precisely requires linear space, therefore in these cases no non-trivial savings can be achieved over naïve computation. Thus, one has to consider randomized approximation algorithms to have any hope of computing entropy efficiently. This problem of monitoring entropy efficiently has received widespread interest in the theory community, and several data streaming algorithms are now known [2, 3, 6, 7, 11, 12, 16] that utilize significantly lower memory than straightforward computation. However, even the proposed streaming algorithms fail to keep up with ever-increasing channel bandwidths observed at a typical NOC of a large organization, especially considering that monitoring of several stream features is desirable for accurate detection.

Our contributions. We refer the reader to the abovementioned previous research for the strong justification of entropy's usefulness in network anomaly detection. In this paper, we concentrate on making such entropy estimations on real traffic scalable and efficient, and extending them to conditional entropy. In particular, we

- implement a distributed sketching algorithm, and compare it with a previously studied streaming estimation algorithm;
- show that it computes both pairwise-conditional entropy with reasonable accuracy;
- make a case study of conditional entropy successfully detecting an attack-related anomaly more efficiently than single-feature entropies.

We now discuss these contributions in more detail.

1.1 Sketching algorithms

In this work, we take advantage of a useful feature of some of the streaming algorithms mentioned previously. The algorithms of Bhuvanagiri and Ganguly [3] and Harvey et al [12] are in a class of algorithms called *sketching algorithms*. A sketching algorithm is a streaming algorithm that maintains a data structure called a *sketch*. A sketch can be combined with other sketch instances corresponding to other streams to obtain a "global" sketch for the union of the streams under consideration. As we shall see in the next section, this property is useful when trying to make our monitoring algorithms distributed, scalable and hence practically usable.

1.2 The case for conditional entropy

Even given algorithms for monitoring entropy efficiently and scalably, there are two challenges while deploying anomaly detection systems. First is the high rate of false positives. Monitoring entropy is subject to the activity of the users in a network, and it is difficult to have a reasonable model of its "normal" range even with continuous adjustment by dedicated personnel. For instance, at a university, when students in a sizable class are required to download a large software package, the high bandwidth usage pattern might well trigger a false alarm. Second, it is conceivable that an attacker would attempt to mask his effect network traffic by trying to mimic the normal distribution of features in the packets he introduces or even train the intrusion detection system gradually to accept attack traffic as normal [9]. Both of these problems somewhat take away from the benefit of monitoring entropy of traffic distributions.

To overcome these obstacles, we propose monitoring the related statistic of conditional entropy, which tracks the average dependence of one feature on another. The value of conditional entropy indicates the extent to which the first feature can be predicted given the value of the second feature, or, more precisely, how much uncertainty about the value of the second feature remains once the value of the first one is known. Monitoring this relationship between features is a promising direction in network anomaly detection. A change in the "predictability relationship" raises the possibility of suspicious activity. This dependence tends to persist in the course of normal traffic even through varied activities of network users, such as in the university example mentioned previously, so that the rate of false positives is minimized. Monitoring conditional entropy also has the added benefit in that it makes the job of masking by the attacker harder. Maintaining dependencies between features while at the same time carrying out an attack would be more difficult than just maintaining the distribution of features independently. Also, the attacker would have a harder time (quadratically) guessing which pairs of features the defender has chosen to monitor.

Need for distributed, scalable algorithms. With pairs of distributions for conditional entropy, the amount of processing and memory required is effectively increased quadratically. This further motivates the requirement of a scalable, distributed monitoring scheme. In response to this challenge, we produced and tested the first implementation of the sketching algorithm that naturally lends itself to a distributed implementation, namely, the Hierarchical Sample Sketch algorithm of Bhuvanigiri and Ganguly [3]. We also compared its accuracy and performance with that of our implementation of the Lall et al. algorithm.

1.3 Evaluations with 802.11 link layer

We evaluate the strength of our approaches by experimenting with link layer headers collected in real-time from an 802.11 network.

The 802.11a/b/g link layer is feature-rich and complex, and therefore allows a range of interesting attacks with corresponding statistical distribution anomalies. In particular, it includes large (2346–2358 bytes) management frames that can contain an entire ring 0 exploit in their L2 payload alone[1]. Besides the frame type and subtype fields, the link layer header may contain 1–4 MAC address fields, 8 bit flags (two of which affect interpretation of the address fields), and 2 16-bit fields, frame sequence number and duration (the distribution of which has been shown[2] to identify wireless chipset–driver combination as a distinctive fingerprint). The earlier WEP encryption implementations could be leveraged to leak encryption key bits by responding to specially crafted injected frames (e.g., the so-called KoreK attack, which we chose as an example to showcase our methods).

To the best of our knowledge, our experiments with monitoring (conditional) entropy are the first application of streaming entropy estimation methods to 802.11 wireless headers. Our experiments are also unique in that we test the first implementation of a sketching algorithm, the Hierarchical Sample Sketch algorithm of Bhuvanigiri and Ganguly [3].

The remainder of the paper is organized as follows. In Section 2, we detail how conditional entropy can be monitored to analyze network traffic for anomalies. Then, in Section 3, we detail our evaluations of the proposed approach by testing our implementation with real data from a wireless network. Finally, we discuss prior research work related to entropy-based anomaly detection methods and conclude with ideas for further work.

2 Estimating and Monitoring Conditional Entropy

Before we describe our methodology for monitoring of conditional entropy, we need some definitions. A *feature* is any property of a network packet/frame computable without additional context. It may be simply a field of the packet, or it may be an overall property such as the length of the frame. The values of

[1] Demonstrated by Ellch and Maynor at BlackHat 2006 for a vulnerability in MacBook's Wi-Fi kernel driver

[2] J.Cache, “Fingerprinting 802.11 Implementations via Statistical Analysis of the Duration Field”, `http://uninformed.org/?v=5&a=1`

the feature over all observed packets in the network form the data stream. We denote the universe of a feature under consideration to be $[n] = \{1, 2, \ldots, n\}$. Items in the stream are drawn from this universe. For instance, if the feature under consideration is the source IP address, then $n = 2^{32}$. The length of the stream is denoted by m. We first define the kinds of error in the estimates produced by streaming algorithms.

Definition 1 (Error types). *If $\hat{Y}$ denotes the estimate produced by a randomized streaming algorithm computing a statistic with the exact value Y, then the error is said to be a* multiplicative ε-error *with probability δ if*

$$\Pr[|\hat{Y} - Y| > \varepsilon \cdot Y] \leq \delta,$$

and it is said to be an additive ε-error *with probability δ if*

$$\Pr[|\hat{Y} - Y| > \varepsilon] \leq \delta,$$

where the probabilities are over the random bits used by the algorithm.

Thus for at least δ fraction of random bit sequences supplied to the algorithm, it will produce an accurate estimate depending on the parameter ε. The space and (update and estimation) time required by the algorithm typically depend on these parameters, increasing as ε and δ are made smaller. See Table 1 for actual space bounds of the algorithms we consider.

We now formalize the notion of sketching algorithms that we had described earlier.

Definition 2 (Sketching algorithm). *A* sketching algorithm *is a streaming algorithm that, on observing stream S, maintains a data structure $\mathcal{D}(S)$ called a sketch, with the following property. If $\mathcal{D}(S_1)$ and $\mathcal{D}(S_2)$ are the sketches corresponding to two independent instances of the algorithm observing streams S_1 and S_2 respectively, then given $\mathcal{D}(S_1)$ and $\mathcal{D}(S_2)$, it is possible to obtain $\mathcal{D}(S_1 \cup S_2)$ efficiently (time linear in the size of the sketches).*

Next, we define the entropies that we consider for monitoring.

Definition 3 (Entropy). *If we let f_i denote the count of item i ($1 \leq i \leq n$) in a feature stream and let X denote the random variable corresponding to the empirical distribution of the feature so that*

$$\Pr[X = i] = f_i/m, \quad \forall i \in [n],$$

then the empirical Shannon entropy *of X is*

$$H(X) = -\sum_{i \in n} (f_i/m) \log(f_i/m).$$

Definition 4 (Conditional entropy). *Let X and Y be random variables corresponding to two features in the observed stream drawn from universes $[n_1]$ and $[n_2]$ respectively, and let $Z = (X, Y)$. Then the* empirical conditional entropy *of X given Y is*

$$H(X|Y) = H(Z) - H(Y) = H(X, Y) - H(Y).$$

The quantity $H(Z) = H(X, Y)$ is known as the joint entropy of random variables X and Y. Figure 1 shows the relationship between these entropies. The quantity $I(X; Y)$ in the figure is the statistic of mutual information between X and Y and is defined as $I(X; Y) = H(X, Y) - H(X) - H(Y)$. For an excellent treatise on information theory and more information on these statistics we refer the reader to Cover and Thomas [8].

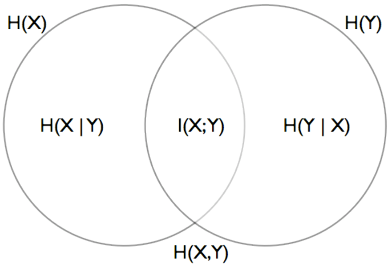


Fig. 1: Relationship between various entropies. Intuitively, values of these entropies are given by the respective areas.

2.1 Estimating conditional entropy

As we noted in the introduction, monitoring conditional entropy has several benefits towards the goal of anomaly detection. In this section we describe how this monitoring is achieved. Although several streaming algorithms are available (see Table 1) for estimating entropy, the case for conditional entropy turns out to be different. Indyk and McGregor [13] showed that estimating conditional entropy with ε-multiplicative error requires $\Omega(m)$ space, so that no non-trivial savings are possible over naïve computation even with randomized, approximation algorithms. However, it is possible to obtain ε-*additive* error estimates of conditional entropy using streaming algorithms that estimate entropy multiplicatively, at the expense of some additional factor increase in space. To get an ε-additive error approximation of conditional entropy $H(X|Y)$, entropy estimation algorithms are run to get accurate estimates of $H(X, Y)$ and $H(Y)$, with multiplicative

Algorithm	Space requirement
Naïve	$O(m \log n)$
Lall et al[3] [16]	$O((1/\varepsilon)^2 \log(1/\delta) \log m \log n)$
Bhuvanagiri & Ganguly [3]	$O((1/\varepsilon^3) \log(1/\delta) \log^5 m)$
Harvey et al [12]	$O((1/\varepsilon)^2 \log(1/\delta) \log m \log n \log(mn))$

Table 1: Comparison of theoretical space bounds of entropy estimation algorithms.

error parameter $\varepsilon' = \varepsilon/(2\log m)$. The estimate $\hat{H}(X|Y) = \hat{H}(X,Y) - \hat{H}(Y)$ is then returned. We have,

$$\begin{aligned}
|\hat{H}(X|Y) - H(X|Y)| &= |(\hat{H}(X,Y) - H(X,Y)) - (\hat{H}(Y) - H(Y))| \\
&\leq |\hat{H}(X,Y) - H(X,Y)| + |\hat{H}(Y) - H(Y)| \\
&\leq \varepsilon H(X,Y)/(2\log m) + \varepsilon H(Y)/(2\log m) \\
&\leq \varepsilon.
\end{aligned}$$

The last bound follows because both entropies are at most $\log m$. This shows that our error is indeed an ε-additive error.

To estimate the entropies $H(X,Y)$ and $H(X)$ we could use any of the algorithms listed in Table 1. In Section 3 we evaluate and compare implementations of both the Lall et al algorithm and the Hierarchical Sample Sketching (HSS) algorithm of Bhuvanagiri and Ganguly, but we focus our attention on the latter since it is in addition a sketching algorithm. For the sake of completeness, we next give a short outline of this algorithm.

2.2 A brief overview of the HSS algorithm

At a very basic level, the HSS algorithm maintains a set of counters organized hierarchically into levels. Each time an item is observed in the stream, the algorithm employs a clever sampling scheme that propagates the item to some of those levels, and a counter at each of the chosen levels is incremented. Pairwise-independent universal hash function families [5] are used at each level to decide which counter at this level is to be incremented. The final estimate is computed using the values of all counters grouped in a special manner. Figure 2 gives an idea of the organization of counters in the HSS sketch and of how an item appearing in the stream updates the sketch data structure. For more specific details on the algorithm we refer the reader to [3].

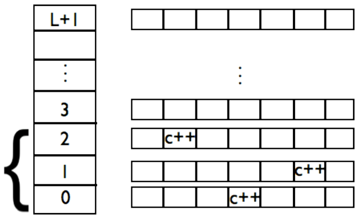


Fig. 2: The HSS sketch. The levels a certain item is propagated to are marked by the brace. The corresponding counters incremented are marked "c++".

2.3 Distributed monitoring

As already mentioned, since the computational time available per item is extremely limited in high bandwidth channels, it is highly desirable to run a monitoring system in a distributed fashion. In fact, it is likely that this is the only practical way of building actual streaming estimation based intrusion detection systems.

We now describe how the *sketching property* of the HSS algorithm can be utilized towards this goal. It is easy to see why the data structure maintained by the algorithm (see Figure 2) is indeed a sketch. For if two such data structures corresponding to two streams are provided, the new structure obtained by simply adding the counters in the two structures is the data structure corresponding to the union of the two streams under consideration. This is so because the set of counters incremented when a value appears in one stream is the same as the set of counters incremented when the same value appears in the other stream.

To utilize this property in a distributed monitoring scheme, we have a load balancer at the NOC that splits the observed stream of packets evenly over a set of sensors and forwards the packets to those sensors. Each sensor has the job of monitoring the (conditional) entropy of the forwarded stream. In order to get global estimates of the entropy, sensors forward their counter values (sketch) to an accumulator node that is able to use the sketching property to return entropy estimates. This scenario is shown in Figure 3. In an actual network, the accumulator may be co-located with the load balancer or with one of the sensors.

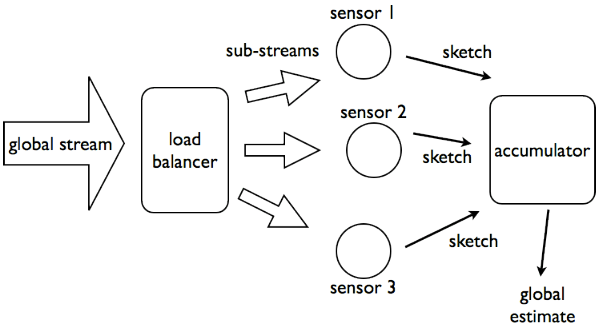


Fig. 3: Distributed monitoring of (conditional) entropy using sketching.

3 Evaluations and Observations

In this section we present the results of our evaluations of the techniques mentioned in previous sections. We first describe the setup behind our evaluations.

Datasets. Our datasets consist of 802.11 headers collected by Air Monitors (AMs) distributed across the Computer Science department at Dartmouth College. For a technical description of our infrastructure see [20]. The AMs are sniffers that monitor several 802.11 channels for frames and forward the link layer headers from the sniffed frames to a server. The headers arriving at the server are anonymized to protect MAC addresses by applying a pre-generated random one-to-one mapping so that distinct MACs never collide.

In our evaluations we use two traces collected in this manner, denoted further as Trace 1 and Trace 2. For both traces four AMs were used to collect data. Trace1 was collected over 56 continuous hours of normal network activity. Trace2 was collected over 41 hours, and apart from normal traffic this trace contained the data sniffed by an AM while we conducted the KoreK attack [14] on a setup in the same vicinity. More details on the attack are given in Section 3.2. Data in both traces was collected at the average rate of about 130,000 frames every 10 minutes. We bin the data in each trace into disjoint windows of 100,000 frames each. On average each window represents a time frame of about 7 minutes.

Distributions. In our evaluations we consider three features for entropy monitoring: *source MAC address* (last 2 bytes), *frame length*, and *duration/ID.* This gives a total of 6 combinations for computing the conditional entropy $H(X|Y)$ when X and Y are taken to be random variables representing different feature distributions.

Whenever values of (conditional) entropy are presented, they are normalized by a factor of $1/\log m = 1/\log(100000)$ in order to allow for comparisons of these values between different pairs of features.

In all evaluations, our algorithms were tested on a server with four Intel x86_64 Xeon processors at 3.0 GHz, 4GB of RAM, running Linux kernel 2.6.22.14.

3.1 Performance of algorithms in conditional entropy estimation

We now detail the performance in conditional entropy estimation of the two algorithms we consider: the HSS algorithm and Algorithm 1 of Lall et al [16] (henceforth referred to as Algorithm 1). Our tests in this section were performed on Trace 1 and follow along the lines of the evaluations in [16]. The performance of the algorithms was evaluated with respect to the accuracy of the algorithms and the memory used in the algorithms. We evaluate the accuracy by determining the relative error in computing conditional entropy over each window. The relative error in computing $H(X|Y)$ is $|H(X|Y) - \hat{H}(X|Y)|/H(X|Y)$, where $\hat{H}(X|Y)$ is the estimate returned by the algorithm. The memory usage of the algorithms is a function of the parameters ε and δ, since these determine the number of counters used in the algorithms. When comparing the relative errors between the two algorithms, we choose ε and δ in the two cases so that the number of counters is the same for both algorithms.

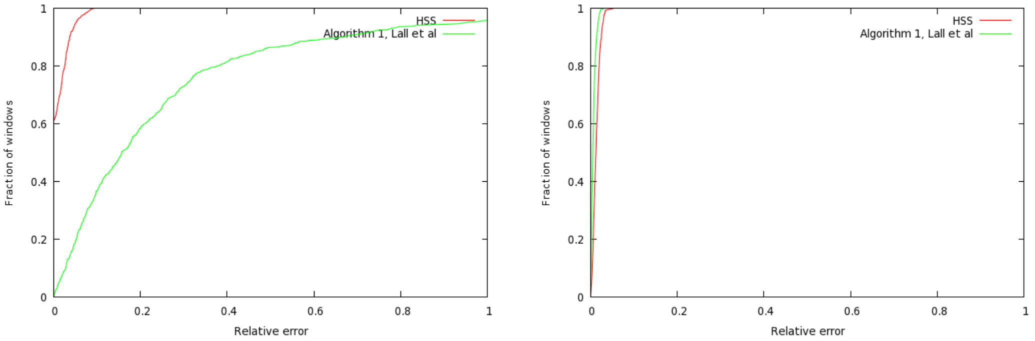
Before detailing the results of our performance evaluations it is useful to note what the values of conditional entropy look like for the different feature distributions over the windows in the trace. As we will see, the relative error in estimates produced by the algorithms depend on the magnitude of conditional entropy. This is not surprising, since the allowed relative error grows as the value of estimates gets smaller, when the algorithm only guarantees bounds on additive error. Table 2 lists the mean values of conditional entropy over the windows, for each combination of features. We now show how the relative errors of the estimates vary with different features and different number of counters used in the algorithms. In Figures 4, 5, and 6 we plot the cumulative distribution function (CDF) of the relative error, showing in what fraction of windows the algorithms produce small relative error. Doing this for different numbers of counters shows what minimum number of counters is necessary in each case. For the sake of brevity, we include only plots where the median relative error is at most 10%. *The first observation we can draw from these figures is that we can estimate conditional entropy accurately using the HSS algorithm, since in each case the relative error is at most 10% for a majority of windows and reaches a maximum of 20% in a small fraction of cases.* We also observe that the HSS algorithm does consistently better in producing accurate estimates as compared to Algorithm 1. We attribute this to the fact that the error guarantees of Algorithm 1 require assumption of certain bounds on the actual values of entropy. In our cases where the normalized conditional entropy is very small, these assumptions may not hold, and indeed we observe that for the cases of Duration/ID given Src Mac, Frame length given Src MAC, and Duration/ID given Frame length where the conditional entropy was at most 0.01 (see Table 2), the difference in accuracy between the two algorithms particularly stands out. In these cases, while the maximum relative error of the HSS algorithm is about 30%, the error due to Algorithm 1 is > 100%. Lastly, we observe that the number of counters required to compute estimates with reasonable error varies by the choice of features. To demonstrate this better we vary the number of counters used in estimating each feature pair and plot the mean relative error for the two algorithms. Figures 7, 8, and 9 show these plots. It is easily observed that the number of counters required to obtain the same level of accuracy is much higher in the three cases mentioned above where the true conditional entropy values are low. *Also the HSS algorithm is able to achieve a mean relative error of 10% in all cases using 25,000 counters, while Algorithm 1 is not able to achieve this kind of accuracy at all in the cases when conditional entropy is low.*

	Frame length	Duration/ID	Source MAC
Frame length	–	0.115	0.0105
Duration/ID	0.00351	–	0.00621
Source MAC	0.107	0.214	–

Table 2: Mean values of actual conditional entropy $H(X|Y)$. Rows represent variable X, and columns represent variable Y.

3.2 Detecting attacks with conditional entropy

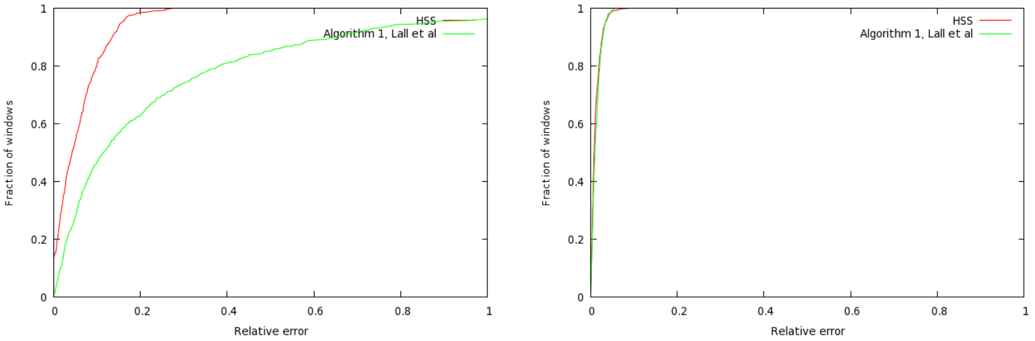
We now look at how our methodology for detecting anomalies fares in the presence of a known attack.



(a) $H(X|Y)$, 40,000 counters

(b) $H(Y|X)$, 20,000 counters

Fig. 4: Relative error of estimates, $X \equiv$ Duration/ID, $Y \equiv$ Src MAC



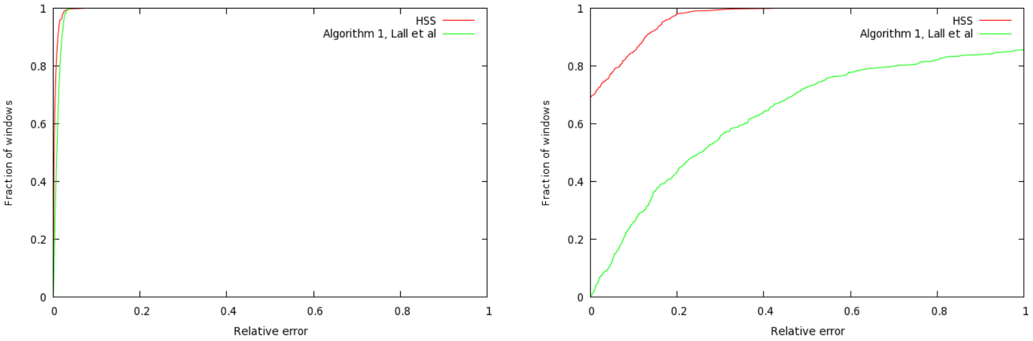
(a) $H(X|Y)$, 40,000 counters

(b) $H(Y|X)$, 25,000 counters

Fig. 5: Relative error of estimates, $X \equiv$ Frame length, $Y \equiv$ Src MAC

For the purpose of demonstrating our methods, we chose the so-called KoreK attack on WEP. Although WEP should be considered antiquitated as a security measure, in our opinion this attack has a number of exemplary properties. In particular, it attacks a cryptographic system via a weakness in its particular implementation specific to a class of devices (a significant practical concern for 802.11 equipment, as successful attacks have shown); it relies on the attacker's capability to inject custom-built 802.11 frames including malformed frames; it does not recover the encryption key, but rather gives the attacker access to sensitive information about the network. Thus we regard this attack as a poster boy for possible future attacks to come.

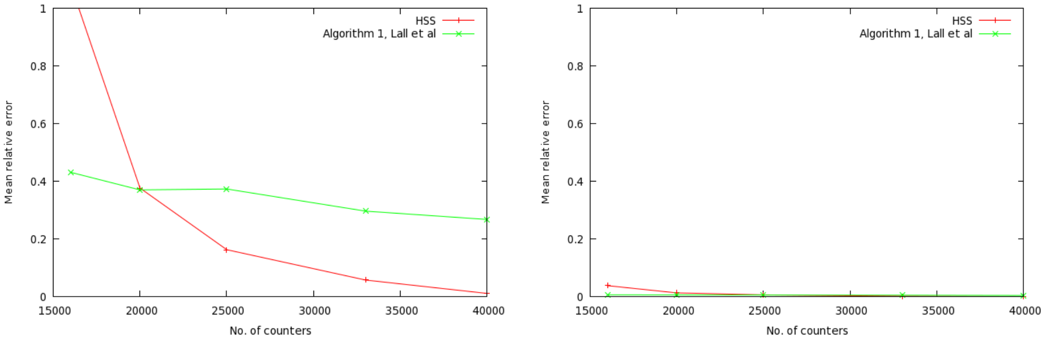
The KoreK attack. The KoreK attack attempts to decrypt a WEP data packet by guessing at the plaintext byte by byte, starting from the end of the packet [14] and making up to 256 guesses at each step. Attempting a guess, the attack produces a packet, which is a valid encrypted packet if the guess is correct,



(a) $H(X|Y)$, 16,000 counters

(b) $H(Y|X)$, 40,000 counters

Fig. 6: Relative error of estimates, $X \equiv$ Frame length, $Y \equiv$ Duration/ID



(a) $H(X|Y)$

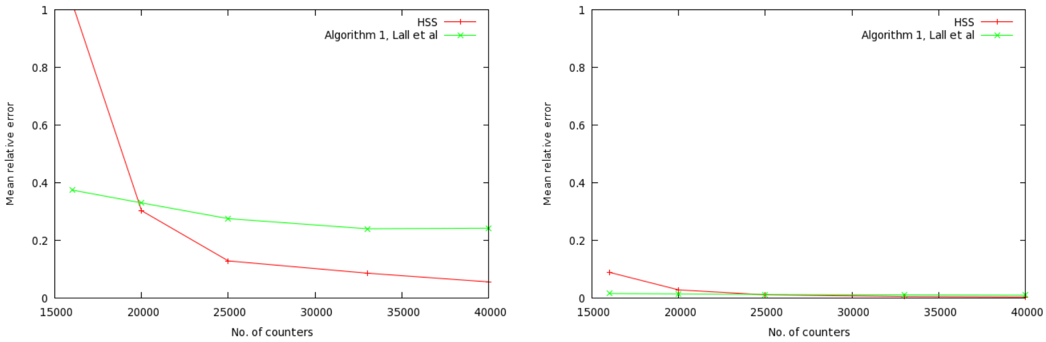
(b) $H(Y|X)$

Fig. 7: Mean relative error with varying numbers of counters, $X \equiv$ Duration/ID, $Y \equiv$ Src MAC

and transmits this packet onto the wireless network. If the packet is valid, the AP will resend it onto the network; if it is invalid, the AP will drop it, thus allowing to find out the last plaintext byte. As soon as the packet is accepted and resent by the AP, and thus the last plaintext byte is found, the attack chops it off and continues working with the packet 1 byte shorter. Thus the KoreK attack generates different-sized clusters of packets of the same size, the size of the packets reduced by 1 byte from cluster to cluster.

We used the KoreK functionality of the *aireplay-ng* tool [1] to decrypt packets on our encrypted wireless network.

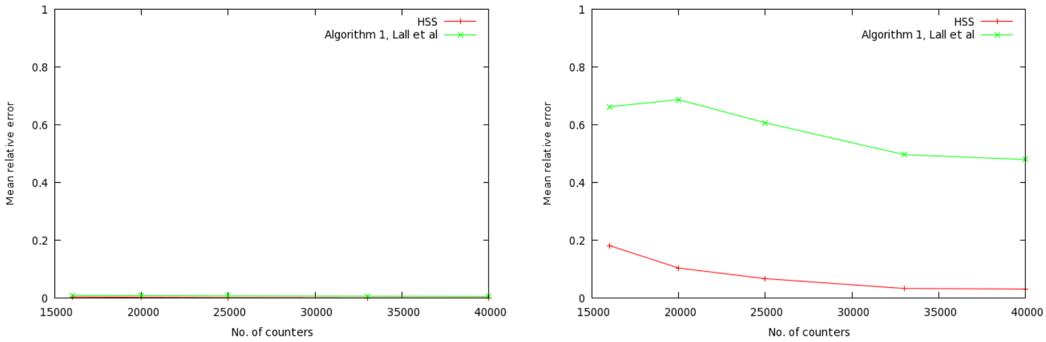
The resulting traffic was captured in Trace 2 and we now present the results of running the HSS estimation algorithm on that trace. Figure 10 shows the relative deviation in the conditional entropies from the median value over windows of Trace 2 when the features of Frame length and Src MAC are considered. On the other hand Figure 11 shows the relative deviation of the separate *entropies* of these features. We chose the median as the measure of central ten-



(a) $H(X|Y)$

(b) $H(Y|X)$

Fig. 8: Mean relative error with varying numbers of counters, $X \equiv$ Frame length, $Y \equiv$ Src MAC



(a) $H(X|Y)$

(b) $H(Y|X)$, 40,000 counters

Fig. 9: Mean relative error with varying numbers of counters, $X \equiv$ Frame length, $Y \equiv$ Duration/ID

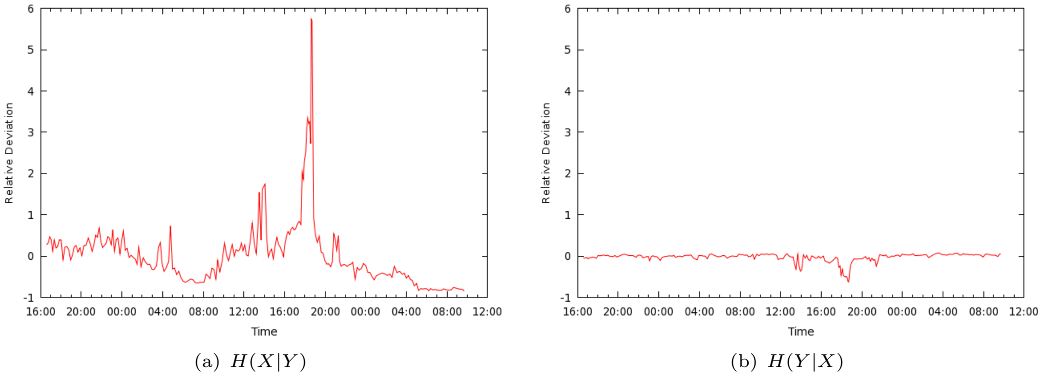
dency to ignore the effect of the KoreK attack. The spikes due to the KoreK attack are visible by observation around the window at 18:30 hrs on the second day in all cases. However, the conditional entropies are affected the most, as can be observed from the prominence of the spikes. Table 3 shows that the attack disturbs the conditional entropy of Frame length given the Src MAC the most —570%— whereas the individual entropies are affected only modestly by up to 50%. *This confirms our hypothesis that conditional entropy, capturing the dependence between feature pairs, could be a more useful statistic in detecting for anomalies.*

4 Related work

Information-theoretic statistics applied to monitoring of network traffic have been shown to be very useful in detecting changes in its character, as first sug-

Feature(s)	Statistic median	For KoreK attack	Relative change
Frame length, given Src MAC	0.0192	0.129	+572%
Src MAC, given Frame length	0.211	0.0821	-61%
Src MAC	0.374	0.240	-36%
Frame length	0.187	0.287	+53%

Table 3: Relative change in (conditional) entropy in the KoreK attack



(a) $H(X|Y)$

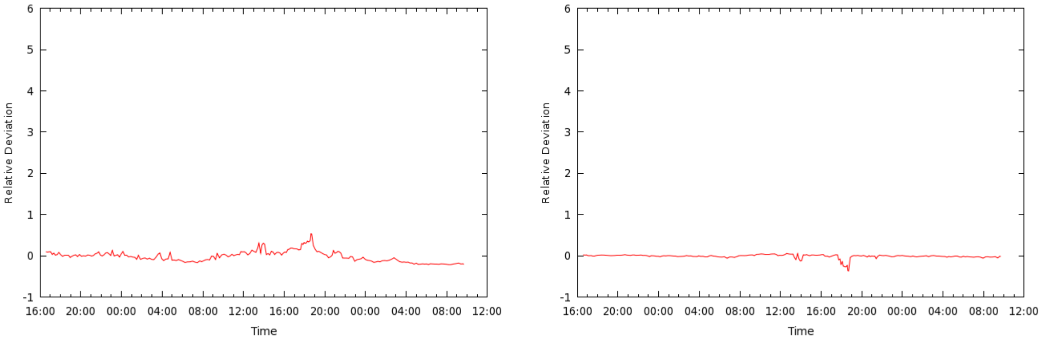
(b) $H(Y|X)$

Fig. 10: Relative deviation of conditional entropy estimates from median, $X \equiv$ Frame length, $Y \equiv$ Src MAC

gested in [18] and then developed in [17] and other works (e.g., [4, 10, 16, 19, 21, 22]).

While possessing the significant advantage of needing few modeling assumptions about what constitutes normal and abnormal traffic, information-theoretic methods incur the substantial computational cost of having to compute entropy and related statistics, which presented an obstacle to their practical adoption. Recent advances in streaming estimation of entropy [3, 6, 12, 16] pave the way to overcoming this obstacle.

Reducing the computation cost of entropy estimation opens the way not only for single-variable entropy-based anomaly detection, but also for analysis of pairwise feature dependencies, which can be an efficient anomaly detection tool. Nychis et al [19] commented that computing entropy of several features over time epochs, and tracking the correlation between the entropy of pairs of features over a window of time epochs might be a useful approach for anomaly detection. Our approach is different, because we do not monitor the correlation between entropy, but rather track the conditional entropy. This is a direct statistic of the concerned feature distributions, and not one of entropies of the feature distributions. Changes in the nature of network traffic are likely to manifest directly in the conditional entropy. The computation of statistics of pairs of features is computationally heavy and we also provide an approach for managing it in a distributed manner.



(a) $H(X)$

(b) $H(Y)$

Fig. 11: Relative deviation of entropy estimates from median, $X \equiv$ Frame length, $Y \equiv$ Src MAC

5 Future Work

As mentioned in the introductory section, monitoring conditional entropy for all pairs of features requires a quadratic blowup in memory. In this paper, we avoid this issue by selecting a limited set of feature pairs to analyze. Our results show that monitoring conditional entropy is a promising approach, even when selecting only a small number of feature pairs.

A more general approach would be to model the entropy of the joint distribution of *all* features simultaneously by monitoring the “top n” pairs that are most likely candidates for a Chow-Liu type spanning tree model of the joint distribution. However, it is not yet clear to us that such a problem can be handled efficiently by a sketching algorithm. We intend to explore this in future work.

6 Conclusion

A strong case for using Shannon entropy of various characteristics of network traffic for anomaly detection has been made by a number of researchers. However, computational challenges of keeping up with high-bandwidth network links are still significant, even after the recent theoretical advances in streaming estimation algorithms.

We show that robust estimation of entropy in network streams can be done in distributed fashion, thus dramaticially increasing its scalability in practical implementation. We produced an implementation of the Bhuvanagiri-Ganguly HSS sketching algorithm (which, to the best of our knowledge, is the first practical implementation of this algorithm) and studied its accuracy, and memory requirements. We applied our implementation to estimating entropies of features of the 802.11 link layer, with real network data collected by our experimental wireless infrastructure at Dartmouth.

With the extra scalability advantages of the distrubuted approach to entropy estimation, the use of pairwise *conditional* entropies for anomaly detection now comes within reach. We study the application of HSS to estimating conditional entropies of 802.11 link layer features, and make a case study of the KoreK attack with both single-feature and pairwise-conditional entropies. We demonstrate both feasibility and superior sensitivity of the conditional approach.

We hope that the distributed streaming estimation algorithms will open a wide range of new possibilities in the practice of anomaly-based intrusion detection and network monitoring.

References

1. KoreK chopchop. `http://www.aircrack-ng.org/doku.php?id=korek_chopchop`.
2. C. Arackaparambil, J. Brody, and A. Chakrabarti. Functional monitoring without monotonicity. In *Proc. of ICALP*, 2009. To appear.
3. L. Bhuvanagiri and S. Ganguly. Estimating entropy over data streams. In *Proc. of ESA*, 2006.
4. D. Brauckhoff, B. Tellenbach, A. Wagner, A. Lakhina, and M. May. Impact of traffic sampling on anomaly detection metrics. In *Proc. of ACM/USENIX IMC*, 2006.
5. J. L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143154, 1979.
6. A. Chakrabarti, G. Cormode, and A. McGregor. A near-optimal algorithm for computing entropy of a stream. In *Proc. of SODA*, 2007.
7. A. Chakrabarti, K. Do Ba, and S. Muthukrishnan. Estimating entropy and entropy norm on data streams. In *Proc. of STACS*, 2006.
8. T. M. Cover and J. A. Thomas. Elements of Information Theory. *John Wiley & Sons, Inc.*, 1991.
9. Dorothy E. Denning. An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, 13:222-232, 1987.
10. L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred. Statistical approaches to DDoS attack detection and response In *In Proc. of DARPA Information Survivability Conference and Exposition*, 2003.
11. S. Guha, A. McGregor, and S. Venkatasubramanian. Streaming and sublinear approximation of entropy and information distances. In *Proc. of SODA* 2006.
12. N. J. A. Harvey, J. Nelson, and K. Onak. Sketching and streaming entropy via approximation theory. In *Proc. of FOCS*, 2008.
13. P. Indyk and A. McGregor. Declaring independence via the sketching of sketches. In *Proc. of SODA*, 2008.
14. informIT. Byte-Sized Decryption of WEP with Chopchop. `http://www.informit.com/guides/content.asp?g=security&seqNum=196`, 2009.
15. V. Karamcheti, D. Geiger, Z. Kedem, and S. Muthukrishnan. Detecting malacious network traffic using inverse distributions of packet contents. In *Proc. of ACM SIGCOMM Workshop on Mining Network Data (MineNet)*, 2005.
16. A. Lall, V. Sekar, M. Ogihara, J. Xu, H. Zhang. Data streaming algorithms for estimating entropy of network traffic. In *Proc. of ACM SIGMETRICS*, 2006.
17. A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *Proc. of ACM SIGCOMM*, 2005.
18. W. Lee and D. Xiang. Information-theoretic measures for anomaly detection. In *Proc. of IEEE Symposium on Security and Privacy*, 2001.

19. G. Nychis, V. Sekar, D. G. Andersen, H. Kim, H. Zhang. An empirical evaluation of entropy-based traffic anomaly detection. *In Proc. of ACM IMC*, 2008.
20. Y. Sheng, G. Chen, H. Yin, K. Tan, U. Deshpande, B. Vance, D. Kotz, A. Campbell, C. McDonald, T. Henderson, and Joshua Wright. MAP: A scalable monitoring system for dependable 802.11 wireless networks. *IEEE Wireless Communications*, 15(5):10-18, October, 2008.
21. A. Wagner and B. Plattner. Entropy based worm and anomaly detection in fast IP networks. In *Proc. of IEEE WET ICE*, 2005.
22. K. Xu, Z.-L. Zhang, and S. Bhattacharya. Profiling internet backbone traffic: Behavior models and applications. In *Proc. of ACM SIGCOMM*, 2005.