# Enhancing Network Intrusion Detection by Correlation of Modularly Hashed Sketches

Martin Drašar, Tomáš Jirsík, Martin Vizváry

Institute of Computer Science, Masaryk University,
Botanická 68a, 61200 Brno, Czech Republic
{drasar,jirsik,vizvary}@ics.muni.cz

**Abstract.** The rapid development of network technologies entails an increase in traffic volume and attack count. The associated increase in computational complexity for methods of deep packet inspection has driven the development of behavioral detection methods. These methods distinguish attackers from valid users by measuring how closely their behavior resembles known anomalous behavior. In real-life deployment, an attacker is flagged only on very close resemblance to avoid false positives. However, many attacks can then go undetected. We believe that this problem can be solved by using more detection methods and then correlating their results. These methods can be set to higher sensitivity, and false positives are then reduced by accepting only attacks reported from more sources. To this end we propose a novel sketch-based method that can detect attackers using a correlation of particular anomaly detections. This is in contrast with the current use of sketch-based methods that focuses on the detection of heavy hitters and heavy changes. We illustrate the potential of our method by detecting attacks on RDP and SSH authentication by correlating four methods detecting the following anomalies: source network scan, destination network scan, abnormal connection count, and low traffic variance. We evaluate our method in terms of detection capabilities compared to other deployed detection methods, hardware requirements, and the attacker's ability to evade detection.

## 1 Introduction

With the dramatic development of network technology over last few years, the number of Internet users and traffic volume has increased. Globally, the number of Internet users is forecast to increase from 1.9 billion users in 2010 to 3 billion in 2015. The amount of the traffic per average Internet user is estimated to grow from 7.3 gigabytes per month to 61.8 gigabyte per month. Furthermore, global consumer internet traffic is about to reach nearly 60 exabytes per month in two years.[15] Along with the growth of the traffic volume, we monitor an increase in the attack count. There was a 42 % increase in the targeted attacks in 2012 and more than 5 thousands of new vulnerabilities were discovered.[16]

Given the immense volume of traffic and arising computational complexity, deep packet inspection becomes inefficient as a method for attack detection

and prevention in large-scale networks. Therefore, behavioral detection methods based on flow analysis have been deployed to detect attackers. Behavioral detection methods distinguish the attackers from valid users by comparing either how their behavior resembles known anomalous behavior or how their behavior deviates from valid behavior. Since there are many types of attacks, various methods have been invented to detect them. Simple methods monitor basic network characteristics and are based on naïve thresholds, while more sophisticated methods employ statistics or behavioral profiling based on more complex characteristics.

Real-life deployment of behavior detection methods has its problems, though. Most methods described in current papers exhibit an excessive false positive ratio and their deployment is subject to base-rate fallacy [1]. Because security administrators need to identify attackers with nearly absolute certainty, the parameters of methods are set to minimize the false positive ratio. However, this minimization results in higher false negative ratios and consequently higher number of undetected attacks. There is also an issue with limited detection capabilities. Currently available methods are usually able to detect a set of attacks that are synergistic in their nature, e.g., port scanning and malware propagation. Methods that employ some form of event correlation to reduce false positives usually focus on events with identical dimensions and are unable to take into account the inherent uncertainty of some events.

Based on the previous discussion, we identify following research questions:

1. How can we correlate diverse anomaly reports from different sources with different dimensions?
2. Can the correlation be done efficiently enough to permit deployment on high speed networks?
3. What is the impact of correlation on the false positive and negative ratio?

To answer the stated questions we apply the following approach. First we propose a correlation mechanism based on modularly hashed reversible sketches and discuss its advantages and drawbacks. Then we use our preliminary implementation and deploy it in a network with more than 15.000 machines to assess its efficiency by comparison with already deployed detection methods.

This paper is organized into five section. In Section 2 we give a brief background on base-rate fallacy, current state of statistic anomaly correlation, and the concept of sketches. In Section 3 we explain in detail the concept of modular key hashing scheme and our correlation method. In Section 4 we evaluate the method both from theoretical standpoint and by experiments on a set of attacks. In Section 5 we conclude our paper and present focus of our future research.

## 2 Background

In this section we present background for our work. First, we introduce a phenomenon of base-rate fallacy of detection methods in real-life networks. Then we briefly describe the approaches to lower false positives of detected events using statistic correlation, and conclude with description of sketches.

### 2.1 Base-rate fallacy

A base-rate fallacy is a phenomenon of Bayesian statistics. The base-rate fallacy is committed when available statistical data are ignored in favor of specific data to make a probability judgment. Base-rate fallacy in network intrusion detection is well described by Axelsson [1]. Briefly, it states that false positive ratios (FP) in units of percent are treated as a remarkable success in papers. However, in real-life and large-scaled networks, due to the large amount of examined anomalies, this FP ratio would result in an excessive number of false alarms.

A closer examination of various detection methods shows that even though authors often claim very high detection efficiency, their results are still not suitable for deployment in real networks. For example, Casas et al. [2] in their clustering method for detection of DoS attacks present a FP ratio 1-3.5 %. Francois et al. [6] who detect bots using link analysis and clustering, give a FP ratio of 2 %. This is equivalent to several thousand false alarms in real-life deployment. Even if they lowered the FP to 0,1 % by slightly reducing the true positive ratio, it still represents several hundred FPs.

### 2.2 Statistic correlation of events with the same dimensions

Statistical correlation of events from more sources is a natural approach for lowering false positives. This approach is typically used in networks where data is gathered from multitude of sources, such as sensor networks. One approach for correlating the results of methods used in sensor networks is proposed by Idé et al. [9]. However, this approach requires homogeneity of gathered data that typical networks cannot provide.

For IP networks, a number of approaches was suggested that correlate events with the same dimensions to confirm an attack. They can process flows in separate time windows [6, 7], detect unexpected changes in time series [10, 21], or measure co-occurrence of events of the same dimensions across multiple data sources. [14]

There are, however, no correlation mechanisms that we know of that can correlate temporally dependent events of different dimensions, e. g., a small increase in the communication of multiple sources and a large increase in the communication to one destination.

### 2.3 Sketches

Next approach to network management and anomaly detection are sketches. In the area of anomaly detection, sketches are mostly used for detection of heavy changes and heavy hitters, usually in the form of top percentile of traffic [3, 13, 22].

A sketch is a probabilistic data-stream aggregation technique that projects a high dimensional network events into a smaller set of dimensions, by hashing the event descriptors. In our case an event is a network flow and descriptors are fields of a flow, e. g. source and destination addresses and ports. A sketch is then

defined by one or more hash functions and a hash table with bucket count equal to the output size of a hash function. Upon each new event a hashing function is applied to event descriptors and the result is used as a key to identify a bucket in the hash table that gets its value updated. Thus information about the entire traffic can be stored in a limited and arbitrarily set space that is dictated by the output size of a hashing function. This is an immense advantage for processing of large data streams, because the hashing can be done very fast and the memory footprint is constant, regardless of the traffic volume. Sketches also have linear structure that enables easy arithmetic operations between different sketches.

Sketches have some disadvantages, though. Most notably collisions and necessity to maintain a structure for sketch reversing due to irreversibility of hashes. Collisions of hash functions could be solved using a $k$-ary sketch proposed by Krishnamurthy et al. [11]. The $k$-ary sketch consists of $H$ hash tables of size $K$. Each incoming event is then hashed by $H$ hashing functions into $H$ tables. This duplication enables to lower collisions to desired levels by modifying $H$ and $K$. An approach to hashing reversal is proposed by Schweller et al. [17, 18], who achieved reasonable reversibility while keeping low memory footprint, high throughput and detection accuracy.

Given their properties, sketches can be used beyond the detection of heavy hitters and heavy changes. Thanks to their linearity, they can be also used for correlation of anomalies and events with different dimensions.

## 3 Correlation of modularly hashed sketches

In the following section we will discuss the concept of modular hashing and explain how it can be used for the correlation of detected anomalies.

### 3.1 Modular hashing

Generally, sketching algorithms work with $k$-universal hashing functions that take the entire input and produce a sketch key in one step. This approach, however, has two drawbacks. The first one is inherent to the nature of hashing functions. Keys are not easily reversible to their original value and reconstructing the original value in naïve implementation may require checking all possible inputs. The second is related to the essence of hashing functions that aim to mangle as much bits of input as possible, thus destroying dependencies between input descriptors. In case of multidimensional events with more descriptors (e.g., source and destination IP addresses, and source and destination ports, where various anomalies can manifest), it is very problematic to reconstruct the nature of an anomaly given only a key and a value.

The reversibility problem was already addressed using numerous approaches, but we have chosen to adapt the method of modular hashing proposed by Schweller et al. [18]. They have demonstrated that by relaxing requirements on hash $k$-universality and then hashing input by separate parts (octets of IP address in their case), it is possible to efficiently reverse keys of a sketch to their

original value. We propose to extend this idea even further and hash each input descriptor separately and then concatenating the result. In this paper we use this notion of a key hashing an input event: $K(I) = K(i_1, \ldots, i_n \mid i \in I) = h(i_1)|| \ldots ||h(i_n)$. Input descriptors that are hashed to a key are referred to in this paper as key parts.

Our proposed modular hashing scheme enables easy reversibility as well as preserving dependencies between descriptors, which are two key properties for our correlation method. These will be described next.

### 3.2 Correlation

Our proposed correlation method is built to work with the uncertainty that is inherent in all anomaly detection methods. This uncertainty is usually dealt with by setting high enough detection thresholds, but this entails the increase in false negatives that we are trying to overcome. Instead of setting one threshold for each detection method that is to be correlated, we are working with uncertainty expressed as an anomaly score. This score represents the possibility that any given network traffic is in fact an anomaly. To this end we operate with two different types of sketches. The first one is the *source sketch*. This sketch accumulates input data and is subject to periodic analysis. The second one is the *analyzed sketch*. This sketch is a result of application of analytic function on a *source sketch*. The function prototype is: $f : S \to S$.

The high-level algorithm for our correlation method is presented in pseudocode in Algorithm 1. On the input it takes analyzed sketches and returns sources of identified anomalies. The algorithm works in four distinct phases: accumulation, analysis, combination, and aggregation.

**Algorithm 1:** High-level overview of the correlation algorithm

**Input**: Set of analyzed sketches

```
1 begin
2     accumulate sketches with the same key structure;
3     while there are sketches left to analyze do
4         scan sketches for anomaly scores going over threshold;
5         combine sketches that have non-empty intersection;
6         aggregate combined sketches with the same key structure;
7         discard already analyzed sketches;
8     end
9 end
```

**Accumulation Phase** In the first phase of the algorithm, anomaly scores of sketches with identical key parts are summed. This phase is ran only once at the

beginning. A simple sum of values was selected as an appropriate mechanism, because on the input, there are analyzed sketches whose values are weighted to be in the same value range. After this phase, the algorithm is left with a set of analyzed sketches that have different key parts, e. g., K(source IP address, destination port), K(destination port, destination IP address), and K(source IP address, destination IP address).

**Analysis Phase** In the second phase of the algorithm, analyzed sketches are checked for values crossing an anomaly threshold, and anomalies are eventually reported. Sketches are linearly scanned for values greater or equal than the anomaly threshold. Such values are marked in each of the arrays for each hash function in a sketch. If for a given source IP address the value is greater or equal to the anomaly threshold in each array, then the source IP address is reported as anomalous. The reversing processes of our algorithm is identical to the one proposed by Schweller et al. [18], which is available as an open-source solution [12], and not explained here for the sake of brevity.

**Combination phase** In the third phase, sketches with different key parts are combined to create new sketches that capture the influences of original sketches. Each two sketches that share at least one key part are combined together. The result of such combination is a new sketch whose key is a union of key parts of the original sketches, e.g., for two sketches with keys K(source IP address, destination port) and K(destination IP address, destination port), the resulting sketch will have a key K(source IP address, destination IP address, destination port). The principle of this combination is illustrated in the Algorithm 2. For each non-null value $v_1$ in the sketch $s_1$, all non-null values $v_i$ of the sketch $s_2$ which have the same value of shared key parts are added together. The result of this addition is stored in the new sketch $s$ with the key which is the union of keys for $v_1$ and all $v_i$.

Example of such combination is given in Table 1.

**Aggregation phase** In the fourth phase, sketches resulting from the combination phase with identical key parts are aggregated together. Sketches that were input to the combination phase do not enter the aggregation phase and are discarded, because their influence is already present in the combined sketches. The purpose of this phase is to eliminate the growing number of sketches that can result from the previous phase. This phase is almost the same as the accumulation phase, but values in buckets sharing the same key are not added together.

| SIP | DPORT | Value |
|---|---|---|
| $A_1$ | $B_1$ | $X_1$ |
| $A_1$ | $B_2$ | $X_2$ |

$\oplus$

| DPORT | DIP | Value |
|---|---|---|
| $B_1$ | $C_1$ | $Y_1$ |
| $B_1$ | $C_2$ | $Y_2$ |

$=$

| SIP | DIP | DPORT | Value |
|---|---|---|---|
| $A_1$ | $C_1$ | $B_1$ | $X_1 + Y_1$ |
| $A_1$ | $C_2$ | $B_1$ | $X_1 + Y_2$ |

Table 1: Example of sketch combination

**Algorithm 2:** Combination of sketches with different key parts

**Legend**: $s_c, s_1, s_2$: Sketches
$P_x$: Set of key parts of sketch $x$
$s(P_y)$: New sketch with key having key parts $P_y$
$h(p)$: Hash of a value of a key part $p$
$H(p)$: All possible hash values of a key part $p$
$V(s, K)$: Values of sketch $s$ at key $K$ in all sketch tables
**Input** : Two sketches $s_1$ and $s_2$
**Output**: Combined sketch $s_c$

**begin**
  $s_c \leftarrow$ *new sketch* $s(P_{s_1} \cup P_{s_2})$;
  **for** $\{p_1, \ldots, p_m \in (P_{s_1} \cap P_{s_2})\}$ **do**
    $V(s_c, K_1 \cup K_2) \leftarrow V(s_1, K_1) + V(s_2, K_2) \mid$
    $K_1 = (h(p_1) \mid\mid \ \ldots \ \mid\mid h(p_m) \mid\mid H(p_m) + 1 \mid\mid ... \mid\mid H(p_{|P_{s_c}|})) \wedge$
    $K_2 = (h(p_1) \mid\mid \ \ldots \ \mid\mid h(p_m) \mid\mid H(p_m) + 1 \mid\mid ... \mid\mid H(p_{|P_{s_c}|})) \wedge$
    $V(s_1, K_1) \neq 0 \wedge V(s_2, K_2) \neq 0$;
  **end**
**end**

Only the higher value is kept to prevent the superfluous rapid growth of values that would result from the repeated addition of anomaly scores from each input sketch.

## 4 Method evaluation

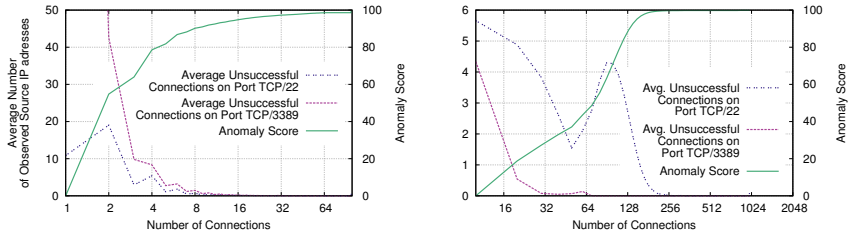
To evaluate the efficiency of the proposed method, we have decided to measure its properties on a real-life network traces containing attacks against Secure Shell (SSH) and Remote Desktop Protocol (RDP) services. We have chosen dictionary attacks because they inherently manifest themselves in several aspects. This was noted by Hellemons et al. [8] and Vykopal [19]. They discovered that port scanning often precedes dictionary attacks. They have also found flow counts typical for such attacks. Also Drašar [4] has revealed that for automated dictionary attacks low variance in flow count is symptomatical. Based on their observations we have decided to implement four sketch-based methods covering four different aspects of dictionary attacks.

### 4.1 Particular Detection Methods

The first sketch method is *source network scan detection*, which measures the number of unsuccessful connections from one IP address to a given port. This method is implemented on top of a sketch with a key K(source IP address and destination port). The second sketch method is *destination network scan detection*, which measures the number of unsuccessful connections to one IP address on a given port. The point of this method is to keep the link between attacking

and attacked computers if, for example, one computer is used for scanning and the other for attacking. This method is implemented on top of a sketch with a key K(destination IP address and destination port). The third sketch method is detection of *abnormal number of connections*, which measures the number of successful flows from one source IP address. The point of this method is to reveal computers that initiate an abnormally large number of connections which is always suspicious in case of dictionary attacks. This method is implemented on top of a sketch with a key K(source IP address and destination port). The fourth method is *low traffic variance detection*, which measures whether connections from one IP address to one port exhibit a near-constant pattern. It is implemented on top of a sketch with key K(source IP address and destination port).

In the traditional settings, methods such as the proposed ones have thresholds separating anomalies from normal traffic based on heuristic or statistical properties. However, this approach forces a binary view on an inherently multivalued one. For example, given a threshold of 30 scanning attempts per five-minute time window, is it reasonable to expect that an IP address with 29 is not anomalous? Rather, it could be argued that the IP address is not anomalous with very low probability. To quantify these probabilities in the aforementioned detection methods, we have analyzed traffic going over the perimeter of Masaryk University's network.




(a) Source Network Scans (b) Destination Network Scans

Fig. 1: Anomaly scores of detection methods

An analysis of traffic in the past year has given us an important insight into the nature of certain traffic aspects and also outlined an approach for measuring anomaly scores in our four detection methods. For *source network scan detection*, we have counted all unsuccessful connections to a given port in five-minute window frame by each communicating IP address. We have then made a histogram of the average number of IP addresses initiating such connections. The result is depicted in Figure 1a. Even a cursory look at the graph reveals that more than 90 % of IP addresses made less then five unsuccessful connections in five-minute time window. Based on this observation, we have decided to set a threshold for maximal anomaly score to a value that represented ∼1 %

of most active traffic sources. That is, in case of *source network scan detection* more than 100 scan attempts were evaluated to maximal anomaly score. Anomaly score for a lower number of scan attempts is set to be proportional to how common these attempts are in a network, i. e.: a ratio of a percentile of given number of connections to difference between minimal and maximal percentile: $\frac{percentile\_of\_measured\_connections - minimal\_percentile}{1 - minimal\_percentile}$. The same approach was used for the *destination network scan detection*, whose result can be seen in Figure 1b, and for *abnormal number of connections*, whose results can be seen in Figure 2a. Anomaly scores for *low traffic variance detection* were based on a paper by Drašar [4] and set as an inverse of relative difference in flow count.

### 4.2 Experiment

To test the efficiency of our method, we have analyzed one week of traffic over the perimeter of Masaryk University. This networks connects about 15.000 active computers every day and the data throughput ranges from about 1 Gbit/s at night to 8 Gbit/s at peak hours. Because this data is not annotated, we have used two already deployed methods for the detection of bruteforce attacks against SSH and RDP services to verify our results. These are pattern matching methods based on a research done by Vykopal [19]. Both methods work by identifying machines that were scanning the network in the previous week and then attempted to log into one or more targets. The SSH detection method uses the following pattern: more than 10 scanning attempts on port TCP/22, connects with more than 10 target machines, performs more than 3 login attempts on each connected host. The RDP detection method uses this one: more than 5 scanning attempts on port TCP/3389, connects with at least one target machine, performs more than 14 login attempts in case of one target or more than 4 in case of multiple targets. Detected attackers are blocked from accessing the network for at least a day.

Data from our experiments can be found in Table 2. The table summarizes the number of anomalies as well as unique IP addresses (in parentheses) that were marked as anomalous by *abnormal number of connection detection*, by accumulation of method with the same key parts and by combination of all methods. The remaining three particular methods are not included, because no anomaly crossed the anomaly threshold. Numbers for the two reference methods are included as well.

### 4.3 Discussion

Given the high thresholds of our reference methods, we regard all their results as true positives. This view is also reinforced by deployment experience – in three years of the SSH attack detection and two years of the RDP attack detection no identified attack was challenged as a false positive. To estimate false positive ratio of our proposed method we have then analyzed sample of identified attacking IP addresses not detected by our reference methods. In this sample, all addresses were confirmed to be attackers. Although not everything was checked,

| **Anomalies** | Connection | Accumulation | Combination | Reference methods |
|---|---|---|---|---|
| **TCP/22** | 2679 (116) | 10045 (264) | 26408 (551) | (47) |
| **TCP/3389** | 53 (20) | 2175 (1079) | 0 | (878) |

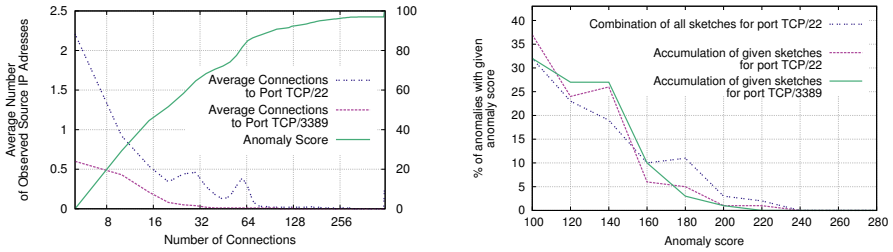
| | |
|---|---|
| SNS | *Source Network Scan Detection* |
| DNS | *Destination Network Scan Detection* |
| Connection | *Abnormal Number of Connections Detection* |
| Variance | *Low Traffic Variance Detection* |
| Accumulation | *Accumulation of* SNS, Connection, Variance |
| Combination | *Combination of* SNS, DNS, Connection, Variance |

Table 2: Detected anomalies on ports TCP/22 and TCP/3389 in one week traffic

we have a very high confidence that the rest was true positive too. Estimation of false negative ratio is even more complicated. As the reference methods are in a way subset of a detection of our proposed method, there are no measured false negatives. However, we believe that there still may be attacks evading our detection, because our thresholds are still relatively high. Complete and rigorous evaluation of positive and negative ratios would require annotated datasets that to our knowledge do not yet exist.

Despite certain gaps in the evaluation, we were able to make some key observations:

- The correlation of data leads to an increase in detection accuracy and number of detected anomalies that surpasses capabilities of particular detection methods. This observation is in accord with observation of other researchers [5, 14, 21].
- Even an unoptimized algorithm for correlation was able to process five minute long windows of traffic over the perimeter of Masaryk University in units of seconds. This observation is very promising for deployment in large networks.




(a) Abnormal Number of Connections (b) Accumulation and Combination

Fig. 2: Anomaly scores of a detection method and overall anomaly scores of accumulation and combination

- It was confirmed by overall anomaly scores for dictionary attacks on the SSH service (see Figure 2b), symptoms of network scanning, connection count, and low variance are strongly correlated.
- No results in combination of sketches for the RDP service verified our observation from production use that malware targeting this service does not precede attacks with large-scale scanning, but rather attacks the first computer available.

Our proposed correlation method can be a powerful tool for the detection of attacks that are intentionally hidden, as was described in [20], because it adds together small anomalies that would be otherwise unseen or ignored. It also enables the sharing of workloads for different detection methods. For example, our two reference methods both check for scanning IP addresses and the same computation is done twice. With correlation, this computation can be seamlessly shared, thus saving computational resources.

Sketch-based methods have proven to be usable and fast enough even on high-speed networks [18]. Our proposed method builds on this foundation and is expected to work in a two tier model. In the first tier, *source sketches* accumulate data from switches, routers, probes, etc. with fixed memory and computational constraints. In the second tier, analyses of these sketches and their correlation are done on more powerful hardware with much relaxed requirements. The correlation phase requires much more memory, because linear growth in key size requires exponential growth in sketch size. There are, however, some optimization techniques that can lower the memory requirements. One can lower key sizes by intentionally limiting the range of possible values. For example, the network space of Masaryk University is only $2^{16}$ addresses, so we could shrink the destination IP key size from 12 bits to only 6. One can also discard values over anomaly threshold from combination phase and switch in later iterations from sketches to hash tables. By omitting already reported anomalies, the number of events to process falls significantly and hash tables can be much more memory and CPU efficient, even for very large keys.

## 5 Conclusion

In this paper we presented an approach to correlation based on modularly hashed reversible sketches. We proposed an algorithm that correlates anomaly reports from different sources.

The detection algorithm, based on $k$-ary sketches, has proved to be a suitable tool for correlation of events with different dimensions. We conducted an experiment in which we deployed the proposed method into a university's network. Even though the method's algorithm was unoptimized, it performed well enough to analyze the five-minute time windows in units of seconds. Compared to other deployed methods, the algorithm was able to detect more anomalies without an increase in the number of false positives.

Our future work will concentrate on addition of new detection methods for correlation. We will also explore possibilities for optimization.

## Acknowledgments

This paper is supported by the Czech Ministry of Interior under Identification code VF2013201531.

## References

1. S. Axelsson. The Base-rate Fallacy and the Difficulty of Intrusion Detection. *ACM Trans. Inf. Syst. Secur.*, 3(3):186–205, August 2000.
2. P. Casas, J. Mazel, and P. Owezarski. Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge. *Computer Communications*, 35(7):772 – 783, 2012.
3. G. Cormode and S. Muthukrishnan. What's new: finding significant differences in network data streams. In *Proceedings of the IEEE INFOCOM*, volume 3, pages 1534–1545 vol.3, March 2004.
4. M. Drašar. *Protocol-independent Detection of Dictionary Attacks*, pages 304–309. Springer Berlin Heidelberg, Berlin, 2013.
5. R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda. MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking. In *Proceedings of the 6th International COnference*, Co-NEXT '10, pages 8:1–8:12, New York, NY, USA, 2010. ACM.
6. J. Franois, S. Wang, R. State, and T. Engel. BotTrack: Tracking Botnets Using NetFlow and PageRank. In Jordi Domingo-Pascual, Pietro Manzoni, Sergio Palazzo, Ana Pont, and Caterina Scoglio, editors, *IFIP NETWORKING*, volume 6640 of *Lecture Notes in Computer Science*, pages 1–14. Springer Berlin Heidelberg, 2011.
7. J. Goldfarb. Identifying Anomalous Network Traffic Through the Use of Client Port Distribution. In *CERT FloCon Workshop*, Vancouver, Washington, USA, 2006. <`http://www.cert.org/flocon/2006/presentations/clientport_dist1205.pdf`>. [Accessed Jan 11., 2014].
8. L. Hellemons, L. Hendriks, R. Hofstede, A. Sperotto, R. Sadre, and A. Pras. SSHCure: A Flow-Based SSH Intrusion Detection System. In R. Sadre, J. Novotný, P. Čeleda, M. Waldburger, and B. Stiller, editors, *Dependable Networks and Services*, volume 7279 of *Lecture Notes in Computer Science*, pages 86–97. Springer Berlin Heidelberg, 2012.
9. T. Idé, S. Papadimitriou, and M. Vlachos. Computing Correlation Anomaly Scores Using Stochastic Nearest Neighbors. In *Proceedings of the IEEE International Conference on Data Mining*, pages 523–528, 2007.
10. K. Ishibashi, T. Kondoh, S. Harada, T. Mori, R. Kawahara, and S. Asano. Detecting Anomalies in Interhosts Communication Graph. In *CERT FloCon Workshop*, Scottsdale, Arizona, USA, 2009. <`http://www.cert.org/flocon/2009/presentations/Ishibashi_GraphAnomalies.pdf`>. [Accessed Jan 11., 2014].
11. B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based Change Detection: Methods, Evaluation, and Applications. In *Proceedings of the 3rd ACM SIGCOMM*, IMC '03, pages 234–247, New York, NY, USA, 2003. ACM.
12. Network Systems Lab. opensketch. `https://github.com/USC-NSL/opensketch`, 2013.
13. A. Li, Y. Han, B. Zhou, W. Han, and Y. Jia. Detecting Hidden Anomalies Using Sketch for High-speed Network Data Stream Monitoring. *Applied Mathematics and Information Sciences*, 6(3):759–765, 2012.

14. A. Mahimkar, A. Lall, J. Wang, J. Xu, J. Yates, and Q. Zhao. SYNERGY: Detecting and Diagnosing Correlated Network Anomalies. <http://www.research.att.com/export/sites/att_labs/techdocs/TD-7KEJWS.pdf>. [Accessed Jan 11., 2014].
15. IEEE 802.3 Ethernet Working Group. IEEE 802.3TM Industry Connections Ethernet Bandwidth Assessment. http://www.ieee802.org/3/ad_hoc/bwa/BWA_Report.pdf, July 2012.
16. Synmatec Corporation. Internet Security Threat Report 2013. http://www.symantec.com/security_response/publications/threatreport.jsp, April 2013.
17. R. Schweller, Y. Chen, E. Parsons, A. Gupta, G. Memik, and Y. Zhang. Reverse Hashing for Sketch-based Change Detection on High-speed Networks. Technical report, Proceedings of the INFOCOM, 2004.
18. R. Schweller, A. Gupta, E. Parsons, and Y. Chen. Reversible Sketches for Efficient and Accurate Change Detection over Network Data Streams. In *Proceedings of the 4th ACM SIGCOMM*, IMC '04, pages 207–212, New York, NY, USA, 2004. ACM.
19. J. Vykopal. A Flow-Level Taxonomy and Prevalence of Brute Force Attacks. In *Advances in Computing and Communications*, pages 666–675, Berlin, 2011. Springer Berlin Heidelberg.
20. J. Vykopal, M. Drašar, and P. Winter. *Flow-based Brute-force Attack Detection*, pages 41–51. Fraunhofer Research Institution AISEC, Garching near Muenchen, 2013.
21. R. Yan and Ch. Shao. Hierarchical Method for Anomaly Detection and Attack Identification in High-speed Network. *Information Technology Journal*, 11(9):1243–1250, 2012.
22. Y. Zhang, S. Singh, S. Sen, N. Duffield, and C. Lund. Online Identification of Hierarchical Heavy Hitters: Algorithms, Evaluation, and Applications. In *Proceedings of the 4th ACM SIGCOMM*, IMC '04, pages 101–114, New York, NY, USA, 2004. ACM.