Anonymization of Network Traces Data through Condensation-based Differential Privacy

AHMED ALEROUD, Yarmouk University, Jordan, University of Maryland, Baltimore County, USA
FAN YANG, SAI CHAITHANYA PALLAPROLU, ZHIYUAN CHEN, and GEORGE KARABATIS,
University of Maryland, Baltimore County, USA

Network traces are considered a primary source of information to researchers, who use them to investigate research problems such as identifying user behavior, analyzing network hierarchy, maintaining network security, classifying packet flows, and much more. However, most organizations are reluctant to share their data with a third party or the public due to privacy concerns. Therefore, data anonymization prior to sharing becomes a convenient solution to both organizations and researchers. Although several anonymization algorithms are available, few of them allow sufficient privacy (organization need), acceptable data utility (researcher need), and efficient data analysis at the same time. This article introduces a condensation-based differential privacy anonymization approach that achieves an improved tradeoff between privacy and utility compared to existing techniques and produces anonymized network trace data that can be shared publicly without lowering its utility value. Our solution also does not incur extra computation overhead for the data analyzer. A prototype system has been implemented, and experiments have shown that the proposed approach preserves privacy and allows data analysis without revealing the original data even when injection attacks are launched against it. When anonymized datasets are given as input to graph-based intrusion detection techniques, they yield almost identical intrusion detection rates as the original datasets with only a negligible impact.

CCS Concepts: • **Security and privacy → Intrusion/anomaly detection and malware mitigation**; **Privacy-preserving protocols**; **Pseudonymity, anonymity and untraceability**; **Intrusion detection systems**;

Additional Key Words and Phrases: Data Injection attacks, information security, netflow, intrusion detection, semantic link network, differential privacy, trace anonymization

ACM Reference format:
Ahmed Aleroud, Fan Yang, Sai Chaithanya Pallaprolu, Zhiyuan Chen, and George Karabatis. 2021. Anonymization of Network Traces Data through Condensation-based Differential Privacy. *Digit. Threat.: Res. Pract.* 2, 4, Article 30 (October 2021), 23 pages.
https://doi.org/10.1145/3425401

This work was partially supported by MITRE-USM FFRDC under grant # 11183.
Authors' addresses: A. Aleroud, School of Computer and Cyber Sciences, Augusta University, 2500 Walton Way, Augusta, GA 30904; email: aaleroud@augusta.edu; F. Yang, S. C. Pallaprolu, Z. Chen, and G. Karabatis, Department of Information Systems, University of Maryland, Baltimore, MD 21250; emails: {fyang, p39, zhchen, georgek}@umbc.edu.
Updated author affiliation: Ahmed Aleroud, Augusta University, Georgia, USA

2576-5337/2021/10-ART30 $15.00
https://doi.org/10.1145/3425401

1 INTRODUCTION

Network security is one of the most significant issues for any organization, and network trace data is a primary asset that needs to be protected. It can be used in several tasks, such as network management, packet classification, traffic engineering, and tracking user behavior [18]. However, these tasks are routinely performed by external organizations. Releasing network trace data to external entities is a very sensitive issue for any organization, and it is often prohibited because sharing such data exposes critical information of the organization, such as host addresses, emails, personal webpages, and even authentication keys. To protect such sensitive information (e.g., IP addresses, user IDs, passwords) the traces must first be anonymized before they are released to external entities or even to the public [28]. In addition, the anonymization process needs to be carried out in a manner that maintains both the research value of the trace and the owner's privacy. Several network traffic data sources such as RIPE (European IP Networks) [23], Route Views [30], and the **Center for Applied Internet Data Analysis (CAIDA)** [1] provide anonymized collected traffic traces to the research community. Most of the attributes in network traffic logs are considered sensitive information and must be anonymized. An example is mapping each IP into a random 32-bit address, a technique called one-to-one random mapping [18]. However, such anonymization approaches result in losing the prefix relationship between IP addresses. In several data mining algorithms, such a relationship is significant; therefore, it is highly desirable to anonymize IPs using prefix-preserving approaches, that is, the original and the corresponding anonymized IP addresses have the same k bit prefixes. Another problem with existing techniques is not handling data injection attacks. This is the act of an adversary injecting specific information to be logged as part of the dataset that will be anonymized so it may be recognized after the anonymization process and reveal additional information about the source. Burkhart et al. [9] refer to this attack as a privileged one, because the adversary has more insight than an external observer. Injection attacks are executed by sending specific IP packets to appear in anonymized traffic [48]. This form of attack has been discussed by Gattani [21] and Coull et al. [11]. There are several existing techniques for anonymization of traces, including the very well-known CryptoPAn approach, which has been shown to be vulnerable to fingerprinting and injection attack [6, 8, 32, 47, 49]. In those attacks, adversaries who have some knowledge about the original flows from relevant sources such as DNS and WHOIS may inject some fake flows to the original traces and discover those flows in the resulting anonymized data using some matching algorithms. The adversaries can then escalate their fingerprinting activities to discover other sensitive attributes such as the shared prefixes of IP addresses in the network. Let us demonstrate such an attacking technique in the following adversary model, which is adopted and modified from Reference [32].

Adversary Model: Tables 1 and 2 show examples on original and anonymized traces. IP addresses were anonymized using CryptoPAn tool. Other features were anonymized using a random permutation function. Anonymization applied using CryptoPAn preserved relationships between similar prefixes of IP addresses after anonymization. Suppose that an adversary has injected the first three flows with specific timestamps and feature values (Table 1), where each feature is denoted by f.

Using the Euclidian distance function d= $((f_2^{iO} - f_1^{iA})^2 + (f_2^{jO} - f_1^{jA})^2)^{\frac{1}{2}} | O$: is the original feature, A is the anonymized feature, the flows injected in Table 1 can be discovered in Table 2 by finding their anonymized **Nearest Neighbors(NNs)** (i.e., the underlined flows in Table 2). The attacker objective is to recover most of the injected flows; this happens when many of the recovered anonymized NNs are the flows injected by the adversary. By discovering the injected flows in the anonymized traffic, it is easy to discover the shared prefixes in the anonymized traces. For example, now the adversary can realize that the first and last flows share the same prefix. As such, he can expand his knowledge to even discover many subnets of the attacked network. In this work, the assumptions about attacker knowledge are similar to the works in References [6, 8, 47, 49], that is, the anonymization function is publicly known, but the key is not known by the adversary. The objective of the adversary becomes to recover all injected flows after anonymization to ultimately discover sensitive features such as IP addresses.

Table 1. Original Trace

Start time	Src Port	Dst Port	P	Pkts	Src_IP
11:51:45:52	**902**	**600**	**UDP**	**6**	**10.1.1.0**
11:51:45:55	**901**	**2,000**	**UDP**	**8**	**150.10.10.1**
11:51:45:57	**900**	**63**	**UDP**	**10**	**128.10.10.1**
11:51:45:59	800	1,900	TCP	2	10.1.1.0
11:51:45:61	750	2,330	TCP	1	150.10.1.0
11:51:45:66	220	591	TCP	1	150.10.20.0
11:51:45:68	22	2,600	TCP	1	10.1.1.0

Table 2. Output of CryptoPAn and a Random Permutation Function

Start time	Src Port	Dst Port	P	Pkts	Src_IP
10:53:42:54	930	608	UDP	7	117.14.242.125
10:53:42:57	916	2,010	UDP	10	159.61.5.252
10:53:42:59	917	65	UDP	10	135.243.4.124
10:53:42:63	806	1,910	TCP	3	117.14.242.125
10:53:42:64	739	2,339	TCP	2	159.61.13.126
10:53:42:67	230	599	TCP	5	159.61.20.124
10:53:42:69	29	2,614	TCP	2	117.14.242.125

Recenty Mohammady et al. [32] proposed a multi-view method to address the problem of injection attacks. The proposed method divides data into multiple partitions and applies prefix-preserving encryption on IP addresses in each partition different number of times. This prevents adversaries from using injection attacks to recover the same IP across partitions. The data analyzer generates N views of the anonymized dataset and one of them will match the original anonymized dataset (when prefix-preserving encryption is only applied once). The key is that the data owner knows which view is correct, but the data analyzer does not. The data analyzer needs to analyze all N different views and generate N reports. The data owner can use oblivious query to retrieve the correct report. This approach provides protection against injection attacks and does not compromise utility. The problem is that now the data analyzer needs to spend N times of computational effort to analyze the data. In experiments presented in that paper, N ranges from 20 to 160. Given the large volume of network data, this is a major burden on data analyzer, making this solution often impractical.

In this work, we address two shortcomings in existing work on anonymizing network trace data [27, 36–38, 40]: (1) lack of a formal and strong privacy protection model and (2) exposure to injection attacks [9, 25]. The contributions of this work are the following:

1. We propose a condensation-based differential privacy anonymization model for network data, which provides a privacy guarantee when sharing such data, thus generating a dataset that can be used for security analysis (i.e., retaining the utility of the original dataset)
2. We show that our anonymization technique is robust to data injection attacks
3. We prove that the anonymized data can be used to detect cyber-attacks not only using attack classification models but also using other graph-based intrusion detection techniques.
4. Our method does not add any burden on data analyser. Data analysis can be done, as it is without any additional steps.

The remainder of this article is organized as follows: Section 2 presents a background on anonymization and data injection attacks. Section 3 presents our proposed anonymization approach. Section 4 discusses the experimental results. Section 5 presents our findings and discusses future work.

2 RELATED WORK

There have been surveys on anonymizing network trace data such as in Reference [43]. We conducted a comprehensive survey on this topic, including most recent developments in privacy protection techniques such as differential privacy [15] and secure multi-party computation [44]. In the following, we discuss research relevant to network trace anonymization.

Anonymization is a process of excluding sensitive identifiers from a dataset but keeping its statistical characteristics so it can be still useful for analysis and scientific research by external entities [45]. A network trace is the flow of packets between a sender and a receiver. It contains attributes such as source and destination addresses, source and destination port numbers, MAC address, timestamp, packet length, protocol, payload, and so on. Adversaries may use some of these attributes to either identify end points (e.g., addresses), reveal user behavior (e.g., payload), or inject certain information (e.g., timestamp) that can be easily tracked.

2.1 Anonymization Algorithms

Anonymization algorithms vary based on the level of anonymization performed on the data and the type of features sanitized. For instance, enumeration can only be applied to numeric attributes in sequential order (e.g., IP address). Enumeration sorts the values of an attribute in ascending order and adds a value that is larger than the original one. This technique can be applied to all attributes [18]. In Reference [39] Slagell et al. proposed a scheme to anonymize network traces by shifting each value with a random offset to replace the original value. A random permutation scheme can be used with timestamps. It is a one to one mapping process and is mostly applied to IP and MAC addresses.

Permutation requires two tables, one for maintaining the mapping from non-anonymized to anonymized IP addresses and another to store anonymized IP addresses. Prefix-preserving pseudonymization works in the same manner. It concentrates on satisfying the following rule: If the original IP address has a k bit prefix, then the anonymized version must share the same prefix. Xu et al. in Reference [48] released a trace dataset using a prefix-preserving technique. Overall, prefix preservation has a greater utility than that of Black Marker anonymization scheme, since the latter replaces IP values with constants in a way that significantly degrades the statistical characteristics of a network trace [18, 34, 39]. It has the same effect as simply printing the log and blacking out all IP addresses. This method loses all IP address information and is completely irreversible. While this method is simple, it is quite undesirable, because it does not allow correlation of events perpetrated against a single host. Sequential numbering could be used by adding sequence numbers to distinguish the attribute values according to the order, but it requires large storage to maintain consistency of data.

Hashing addresses this limitation by using a cryptographic hashing function. In general, the generated hash value is smaller than the original attribute value, so this may facilitate dictionary attacks, which can be avoided by combining the hash function with a secret key using **Hash Message Authentication Codes (HMAC)** algorithm. Kuenning and Miller [28] used the two previous methods for anonymizing URLs and filenames. The sequence numbering works faster and results in shorter trace than keyed hashing, except that it can be executed on a single system compared to keyed hashing, which is proven and used in distributed systems. Partitioning creates equivalence relations and canonical examples for a set of values, and then assigns an anonymized value whose range is within the corresponding partition. Truncation anonymizes the IP and the MAC addresses by deleting the least and keeping the most significant bits. This technique can make an end-point non-identifiable [7]. For string attributes, constant substitution can be used. The original data is replaced by a constant to add confidentiality to sensitive attributes [34]. Applying it to the identity attributes results in an undistinguishable

data. Shuffling re-arranges pieces of data (e.g., within an attribute). Generalization approaches replace a data value by a more general data value [39]. In the k-anonymity model by Sweeny [42], the attributes are divided into sensitive, non-sensitive and quasi-identifier attributes. Several equivalence classes are created by hiding the values of quasi-identifiers, such that the quasi-identifier attributes of any record would be similar to at least k-1 quasi-identifier attributes. The k-anonymity model has some limitations regarding the diversity of sensitive attributes. Therefore, the l-diversity model requires the equivalence classes to have at least l unique values for sensitive attributes [29]. Both k-anonymity and l-diversity show good privacy protection on categorical attributes, but they lead to information loss when numerical attributes are anonymized. Micro aggregation techniques are comparable to k-anonymity techniques, as they work mainly on numerical attributes. The records are clustered such that each cluster includes at least k records. However, the features are replaced with values that represent information about the cluster itself [14]. Micro-aggregation techniques cluster the records in the dataset so the similarity among data points inside a cluster is minimized, while the similarity among data points in different clusters is maximized. The quasi-identifier values are masked in a way that is relevant to the cluster itself, e.g., they can be replaced with the cluster average. Data generalization approaches are applied to network traces by "partitioning" information (also called "grouping" or "binning"), e.g., grouping TCP/UDP port numbers by assigning a fixed value to each group [40].

A recent work by Mohammady et al. [33] proposed a multi-view solution to address the problem of injection attacks. Their solution is based on the following method: After an initial application of prefix-preserving encryption on IP addresses, the data owner divides data into partitions and generates a size d (d as number of partitions) vector V of randomly generated integers in the range of 1 to N. A random number α is multiplied to V to get a vector V'. The data owner then applies prefix-preserving encryption techniques $-V'$ of times on each partition, where a negative sign means prefix-preserving encryption is applied in reverse fashion. The data owner sends this anonymized dataset along with boundary of partitions as well as vector V to data analyzer. Since the random number α is withheld, the data analyzer cannot recover original IP addresses. However, the data analyzer can apply prefix-preserving encryption V, $2V \ldots$ NV number of times on each partition. One of these views (when αV) is applied and recovers original anonymized data (before applying V' encryptions). The data analyzer then analyzes all N versions of anonymized data and data owner can obliviously retrieve the correct version of results. The basic method is then extended to provide more protection, however, still N views will be generated, and the data analyzer must spend N times more effort to analyze data. Given that network trace data is often quite large, and N is also quite large (from 20 to 160 in the paper), this method is often impractical.

In our previous work, we proposed a modified condensation based anonymization algorithm for network trace data [4]. This algorithm optimizes the tradeoff between privacy protection and utility preservation, and it achieves much better privacy protection and utility preservation than existing anonymization techniques. There have been some research on anonymizing system logs [12, 22]. However, there are shortcomings in anonymizing logs similar to traces: (1) they only remove identifying information such as IP addresses or user names but are still vulnerable to injection attacks based on other attributes or add significant computational burden to data analyzer; (2) they do not optimize the tradeoff between privacy protection and utility preservation.

2.2 Requirements for Anonymization and Existing Tools

Anonymization tools should satisfy a few requirements to maintain the value of traces. The first is pseudonym consistency requirement, which means that it is necessary to maintain consistency among anonymization for each distinct IP address or hardware addresses within a trace or between different traces that belong to the same organization. The second requirement is to perform a systematic sanitization of the transport, network, and data link layer header information in a trace while eliminating payload. Different tools provide different tradeoffs between privacy and information loss. Few tools work only on network layer information, while others work on cross layer packet anonymization [34]. Tcpdpriv removes private information from network traces using a

prefix-preserving anonymization technique [31]. Xu et al. [48] improved tcpdpriv by using a cryptography-based prefix-preserving anonymization technique. CryptoPAn can be used with parallel and distributed processing of traces [16]. It also meets the pseudonym consistency requirement. Fan et al. [17] evaluated the tool and found that attacks are still possible based on trace type. Slagell et al. in Reference [37] suggested an improved version of CryptoPAn that performs prefix-preserving IP address pseudonymization. Slagell also proposed FLAIM [39], a tool that is not tied to the specific log being anonymized and supports multi-level anonymization. Gamer et al. introduced Pktanon [20], a tool that achieves flexibility, extensibility, and privacy; it allows arbitrary anonymization for every protocol field and uses a defensive transformation technique to prevent privacy violations. Ipsumdump is a tool that translates tcpdump files into ASCII format to be easily readable by programs. It relies on prefix-preserving pseudonymization techniques [26]. Koukis et al. developed an **Anonymization Application Programming Interface (AAPI)** tool, so users can write their own anonymization function and choose the appropriate policy for each attribute [27]. Foukarakis et al. [19] developed ANONTOOL based on AAPI, a command line tool that can generate synthetic data for both online and offline traces. However, all these tools are still vulnerable to injection attacks on anonymized data, which will be described in next section.

2.3 Attacks on Anonymized Data

Several attacks are initiated on anonymized data to reveal or infer sensitive sanitized information, such as identifying network topology [10] or discovering user behavior [11, 27, 46].

In general, there are two types of attacks: inspection attacks and injection attacks. In inspection attacks the attacker is not authorized and only has information from trace and observation, while in injection attack the attacker is authorized and has knowledge about the injected pattern that no one else has [9]. In injection attacks, the attacker injects specific data into traffic. When the dataset is anonymized and released to the public, the attacker's goal is to identify the injected pattern and therefore, easily discover the binding between the original and anonymized data. For example, an attacker may inject a sequence of packets with certain patterns (e.g., specific source or destination port numbers, specific delay between packets, or specific packet sizes. The attacker may recognize these patterns in the anonymized data and through reverse engineering may uncover original data. Gattani in Reference [21] showed that injection attacks are only possible when sufficient knowledge is available on when, how, and where the trace is collected. So, the best countermeasure against this attack is to keep such information private [48]. However, Pang et al. [34] recommended eliminating the data generated by scanners that probe active hosts prior to anonymization, a good approach to protect against active attacks. Burkhart et al. [9] demonstrated that injection attacks are performed either by injecting complex patterns within short time or injecting simple patterns over long time periods. They injected five different types of patterns with different complexity and concluded that it is difficult to protect data from injection attacks using traditional anonymization approaches. Brekne et al. [7] empirically demonstrated the effectiveness of this attack against prefix-preserving anonymization and suggested remedies that might limit its damaging capability. However, they concluded that it is quite difficult to stop such an attack without continuous human investigation. They also found that anonymizing IP addresses by assigning unique and static values to IP address via pseudonymization does not guarantee adequate privacy and immunity against packet injection attacks.

In structure recognition attacks, the objective is to exploit the structure among objects to infer their identities. For example, traces of Internet traffic will often include sequential address scans made by attackers probing for vulnerable hosts [35]. There are attacks that aim to recover IP addresses anonymized using prefix-preserving anonymization techniques [35]. Those attacks exploit shared-text matching for cascading effects, with the shared text being the prefix.

Next, we describe our novel approach that generates an anonymization model with strong privacy protection. In particular, we adapt an improved version of the k-anonymity [42] that incorporates a differential privacy approach in it.

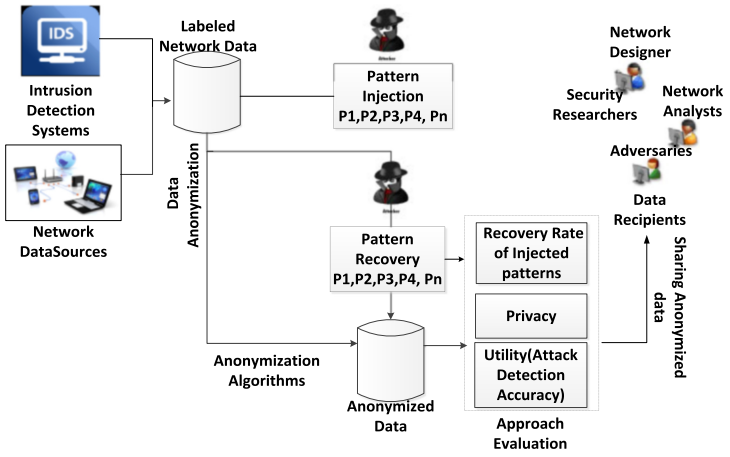


Fig. 1. Research approach and overall system architecture.

3 APPROACH

Most existing anonymization techniques only encrypt the IP addresses in the dataset, but they are vulnerable to injection attacks as shown by Burkhart et al. in Reference [9], where a large fraction of injected packet patterns can be recovered even after permutation of IP addresses, bucketization of ports, adding random noise to time, to number of packets, and to packet size.

Figure 1 depicts our system architecture and flow of information. Network data is collected from network data sources. Adversaries may have injected certain traffic patterns as well. Our anonymization algorithms will be applied to anonymize the collected dataset and send the anonymized data to data recipients. We evaluate our approach based on privacy, utility (measured as accuracy of attack detection), and whether injected patterns can be recovered.

Next, we describe the sensitive attributes we need to anonymize in Section 3.1. Section 3.2 describes pre-processing steps to collect labeled network flow data. Section 3.3 presents our anonymization method for IP addresses. Section 3.4 reviews an existing anonymization method called condensation. Section 3.5 presents an enhanced condensation method to anonymize non-IP features. This method supports k-anonymity. Section 3.6 proposes a condensation-based differential-privacy anonymization method. Section 3.7 describes how to test injection attacks, and Section 3.8 discusses how to test our approach on graph-based intrusion detection methods.

3.1 Sensitive Attributes

Data features in network data sources, when shared, may reveal the network architecture, user identity, and user information; therefore, it is essential to identify sensitive information. Based on our review, we identified several sensitive attributes that need to be protected.

IP addresses: They are considered one of the most important attributes to be anonymized. An attacker relies mainly on discovering the mapping of IP addresses to detect the host and network. For example, a source IP address indicates a user's IP, which may reveal the user identity. The destination IP address may be used by attackers to launch attacks. In addition, IP addresses may be used in intrusion detection algorithms as well. For example, if we know attacks often originate from specific IP addresses, then we can be suspicious of possible attacks from these IP addresses. Similarly, if we know there have been attacks against a certain host (destination IP), then we can pay more attention to packets having that host as a destination IP. Thus, we need to carefully balance the need for privacy protection and intrusion detection when anonymizing IP addresses.

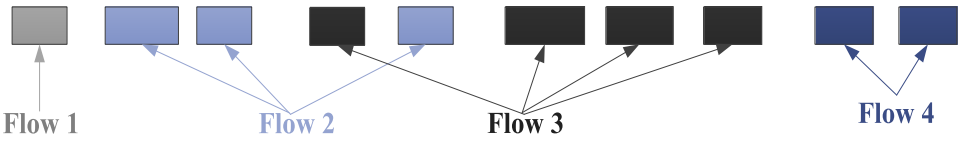


Fig. 2. Packet aggregation to create flows.

Timestamps: They do not indicate any user information; however, they could be used in data injection attacks through discovering the mapping of information that is already known prior to anonymization. In addition, a timestamp may denote specific action with respect to response delay and inter-flow time that could be matched with already known values.

Port Numbers: They partially characterize the applications that are used to create the trace and may be used in fingerprinting attacks to reveal that a certain application with suspected vulnerabilities is running on a network where the trace is collected from.

Trace Counters: They indicate the number of packets per flow. This attribute may be used for fingerprinting and injection attacks.

3.2 Labeled Network Flow Data

In this article, we focus on anonymizing network flows with header information. Each flow contains aggregated information from network packets that have common features, e.g., same source and destination IPs, same protocol, and so on. Figure 2 shows an example of packets that are aggregated into four flows based on common features.

Packets that are close to each other in time and destined to same location are aggregated into a single flow. Data about alerts raised by **Intrusion Detection Systems (IDSs)** are extracted and correlated with the corresponding flows. For example, if a flow contains information about packets associated with alerts, then the flow is automatically labeled as suspicious, otherwise it is labeled as normal (more details on this correlation approach is discussed in our previous work in Reference [13]). The result of this process is a labeled dataset of network flows.

3.3 Anonymization of IP Addresses

The following two stages describe anonymization of IP addresses: (1) First, encrypt/permute the leading digits of the IP addresses (network number). Intrusion detection methods can still use the leading portion of the IP addresses. Attackers may discover the subnet, but the next stage prevents identifying the host. (2) Then, for the remaining digits of the IP (host number part), we cluster these addresses and randomize addresses in the same cluster (exact IP address cannot be located).

Algorithm 1 summarizes the steps needed to anonymize the IP address. The dataset D is divided into n datasets, such that D_i contains flows with label L_i where each label can be an attack or a benign activity. Then permute the leading digits of the IP addresses (network number) using a prefix-preserving permutation function. The IP addresses are then clustered into k clusters based on their least significant digits (host number). The average for the least significant digits of IPs in the same cluster (host number) is calculated. Then the least-significant digits of IPs (the last three digits) in each cluster are replaced using the computed average. Figure 3 shows an example of IP anonymization into two clusters.

3.4 Data Anonymization Using Condensation

A heuristic condensation algorithm by Aggarwal and Yu uses the statistical characteristics of the original data to generate synthetic data while preserving its statistical characteristics [2]. Other anonymization algorithms

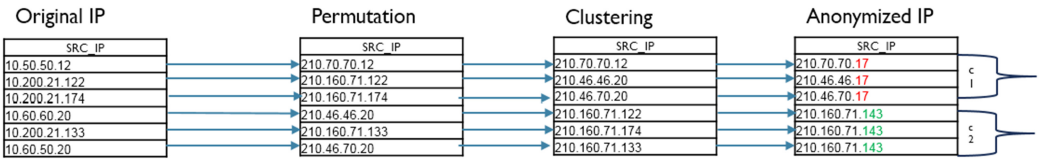


Fig. 3. Anonymization of IP addresses.

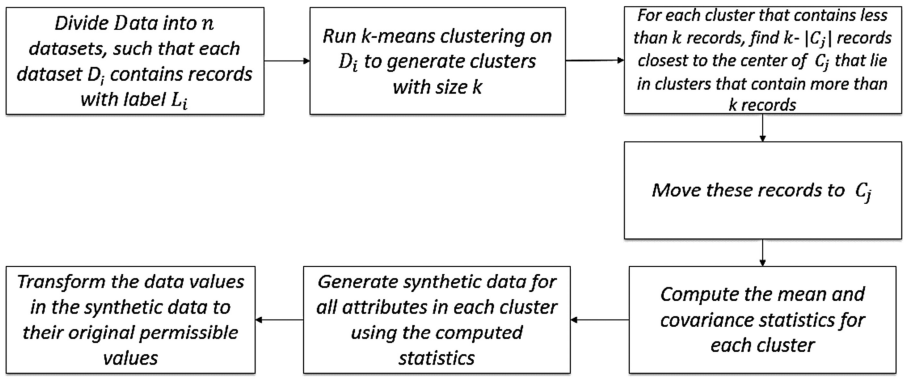


Fig. 4. Anonymization of non-IP features using an enhanced condensation approach.

are limited to noise addition to the data, which may lead to insufficient privacy level. The original condensation algorithm clusters records in the dataset such that the similarity among data points inside the clusters is minimized and the similarity among data points in different clusters is maximized. Each cluster contains at least k records, and the quasi-identifier values are masked in a way that is relevant to the cluster; for example, they can be replaced with the cluster averages.

ALGORITHM 1: IP addresses Anonymization ($\boldsymbol{D}$, $\boldsymbol{k}$)

Divide dataset D into n datasets, such that D_i contains records with label L_i
  For i=0 to n do
    For j=1 to $|IP|$ do
     Permute the leading digits of the IP addresses (network number) using prefix-preserving permutation function
    End for
  Cluster IP addresses into k clusters based on their least significant digits (host number)
  Sort the clusters in ascending order of cluster size. Let them be C_1, C_2, C_k
  For each cluster C_j that contains less than k records,
    Find$k$- $|C_j|$ records closest to the center of C_j that lies in clusters that contain more than k records
    Move these records to C_j
  For each cluster C_j do
    Compute the mean for the least significant digits of IPs in the same cluster (host number)
    Replace the least significant digits of IPs in each cluster using the computed statistics
   End for
  End for

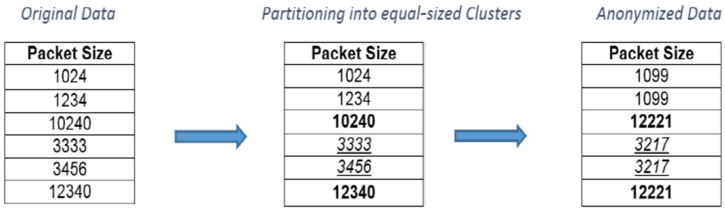


Fig. 5. Cluster-based differential privacy anonymization example.

3.5 Anonymizing non-IP Features Using Modified Condensation

We utilize a condensation-based approach to perform anonymization on non-IP features. We apply two modifications to the original condensation algorithm.

- —First, we implement a per-class condensation mechanism on network traces. The original condensation algorithm does not consider the differences between classes to perform the de-identification. In general, there is a significant difference between the behavior of network attackers and other users and such differences need to be captured in the anonymized data.
- —Second, the original condensation algorithm picks cluster centers randomly, which may lead to inferior clusters. Instead, we utilize k-means clustering algorithm, which is relatively efficient in terms of within-class variance [28].

Figure 4 shows the steps to anonymize non-IP features. The clusters are sorted in ascending order of cluster size. For each cluster C_j that contains less than k records, k - C_j records are selected if they are the closest to the center of C_i that lies in clusters that contain more than k records. The selected records are then moved to C_j. For each cluster C_j the data is shifted into a new space using PCA. In the new space $Z_1, Z_2, \ Z_p$ are independent components. Then, a random data $Z_i^{'}$ with similar statistical features of Z_i is generated. $Z_1^{'}, Z_2^{'}, \ Z_3^{'}$ are combined into one dataset. Finally, Z' is shifted back to the original data space using reverse PCA.

3.6 Condensation-based Differential-privacy Anonymization Method

The differential privacy methods introduced by Dwork [15] provide stronger privacy protection than k-anonymity. To protect the sensitive information, differential privacy methods systematically add a random number generated from a special distribution centered at zero to the results of all data queries. Differential privacy mechanisms ensure that the addition or removal of single database record has no significant effect on the outcome of any analysis performed on the database. The idea of preserving the privacy of network traffic using a noisy version of the true answer is not new, however, the way of noise addition is different in the case of differential privacy.

Our differential privacy approach works as follows: First, we implement a prefix-preserving technique to anonymize IP addresses. We permute the ***n*** leading digits (network part). For the remaining digits, cluster these addresses into K clusters (host part). Then, we randomize addresses within the same cluster. Second, we implement a per-class differential privacy mechanism. Third, we utilize the differential privacy in our condensation method. Our method clusters records based on the features of network trace data. Now each cluster has packets or flows with similar features. We then compute the mean of these features and add Laplace noise to the mean. The perturbed mean replaces the original values. Figure 5 illustrates our cluster-based differential privacy anonymization method. At first, we partition data into three clusters that are displayed by different font type in the table. Then, we compute the mean of each cluster and add Laplace noise to the mean. The final step is to replace every original value with this perturbed mean.

Table 3. Anonymization Policies Used for Testing Data Injection Attacks

	IP Addr.	Ports	Time [S]	Packets	Octets
A1	Permutation	-	-	-	-
A2	Permutation	-	-	O(5)	O(50)
A3	Permutation	B(8)	O(30)	-	-
A4	Permutation	B(2)	O(60)	-	-
A5	Permutation	B(8)	O(30)	O(5)	O(50)
A6	Permutation	B(2)	O(120)	O(10)	O(200)

- B(x): bucketized in x buckets,
- O(x): Added a uniform random offset between −x and +x.

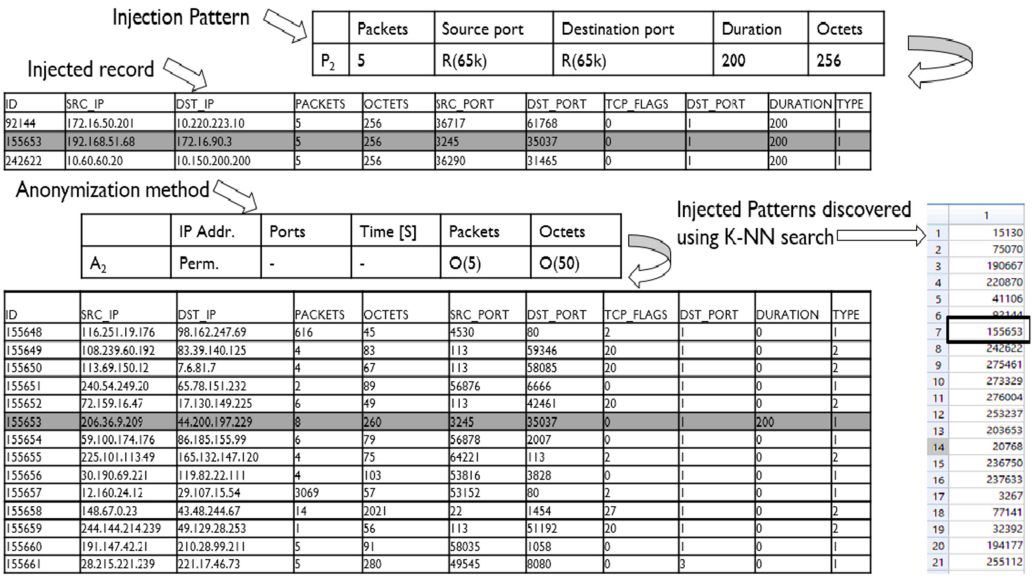
ALGORITHM 2: Differential Private-Condensation of network data (Dataset D, k)

Divide D into n datasets, such that D_i contains records with label L_i
Let n_i be the size of D_i
For i=0 to n do
  Run k-means clustering on D_i to generate $\lfloor n_i/k \rfloor$
  Sort the clusters in ascending order of cluster size. Let them be C_1, C_2, . . ., C_k
  For each cluster C_j that contains less than k records,
    Find k - $|C_j|$ records closest to the center of C_j that lies in clusters that contain more than k records
    Move these records to C_j
  End For
  For each cluster C_j
    Synthetic Data generation: compute the mean of each features and corresponding Laplace noise.
    Replace the data values with the perturbed mean.
  End for
End for

The added noise follows Laplace distribution with mean zero and standard deviation = sensitivity/ε, where sensitivity = (max value in cluster − min value in cluster)/cluster size. Epsilon is a small constant (usually around 0.01). According to the definition, we can see that the larger the cluster size, the smaller the noise, so this method works better for large volumes of data. Algorithm 2 shows the steps to achieve differential privacy on all features except IP addresses, which are anonymized based on an IP prefix-preserving method. First the dataset is divided into subsets and each subset contains instances with an identical class label. Then, we utilize K-means clustering method to generate clusters per subset, since we may end up with some clusters having fewer than k- points and some may have more. Some points are moved from large clusters to small clusters. Then, we compute mean of these features and adds Laplace noise to the mean. The perturbed mean will replace the original values.

3.7 Testing Injection Attacks on Data Anonymized by Our Algorithms

We want to investigate whether datasets that have been anonymized with differential privacy and other approaches are robust enough to withstand injection attacks. Table 3 shows some example patterns used to prepare the injected data. We inject similar patterns to those proposed by Burkhart et al. in Reference [9] and they are injected in the data before anonymization. The data is anonymized using several permutation-based anonymization policies including our proposed differential privacy method. Table 3 shows the anonymization approaches listed in Reference [9]. We then try to identify the injected patterns. We use K-NN Search to recover the injected patterns from the anonymized data. Formally, $knnsearch(p_i, A_i) \forall p_i$ finds the nearest neighbor in



	Packets	Source port	Destination port	Duration	Octets
P_2	5	R(65k)	R(65k)	200	256

ID	SRC_IP	DST_IP	PACKETS	OCTETS	SRC_PORT	DST_PORT	TCP_FLAGS	DST_PORT	DURATION	TYPE
92144	172.16.50.201	10.220.223.10	5	256	36717	61768	0	1	200	1
155653	192.168.51.68	172.16.90.3	5	256	3245	35037	0	1	200	1
242622	10.60.60.20	10.150.200.200	5	256	36290	31465	0	1	200	1

	IP Addr.	Ports	Time [S]	Packets	Octets
A_2	Perm.	-	-	O(5)	O(50)

ID	SRC_IP	DST_IP	PACKETS	OCTETS	SRC_PORT	DST_PORT	TCP_FLAGS	DST_PORT	DURATION	TYPE
155648	116.251.19.176	98.162.247.69	616	45	4530	80	2	1	0	1
155649	108.239.60.192	83.39.140.125	4	83	113	59346	20	1	0	2
155650	113.69.150.12	7.6.81.7	4	67	113	58085	20	1	0	2
155651	240.54.249.20	65.78.151.232	2	89	56876	6666	0	1	0	1
155652	72.159.16.47	17.130.149.225	6	49	113	42461	20	1	0	2
155653	206.36.9.209	44.200.197.229	8	260	3245	35037	0	1	200	1
155654	59.100.174.176	86.185.155.99	6	79	56878	2007	0	1	0	1
155655	225.101.113.49	165.132.147.120	4	75	64221	113	2	1	0	2
155656	30.190.69.221	119.82.22.111	4	103	53816	3828	0	1	0	1
155657	12.160.24.12	29.107.15.54	3069	57	53152	80	2	1	0	1
155658	148.67.0.23	43.48.244.67	14	2021	22	1454	27	1	0	2
155659	244.144.214.239	49.129.28.253	1	56	113	51192	20	1	0	2
155660	191.147.42.21	210.28.99.211	5	91	58035	1058	0	1	0	1
155661	28.215.221.239	221.17.46.73	5	280	49545	8080	0	3	0	1

	1
1	15130
2	75070
3	190667
4	220870
5	41106
6	92144
7	155653
8	242622
9	275461
10	273329
11	276004
12	253237
13	203653
14	20768
15	236750
16	237633
17	3267
18	77141
19	32392
20	194177
21	255112

Fig. 6. A scenario on injection attack and its recovery.

the anonymized data A_i for each record that represents the pattern p_i. The result of K-NN search is a column vector where each record contains the index of nearest neighbor in the anonymized flows for the corresponding record in the injected flows. If there is a match between the injected pattern and the nearest neighbor, then the attack is considered successful. The number of recovered patterns using each anonymization policy is reported. An example on an injection attack and how it is recovered is shown in Figure 6.

3.8 Graph-based Intrusion Detection Methods

An important question comes up with any anonymization method: How well do existing techniques work when they are applied on anonymized data versus the original data? In this section, we provide an answer to this question by using intrusion detection methods applied on anonymized and original data. Our previous work [27] showed that simple classification-based intrusion detection methods generate accurate results on anonymized datasets compared with original non-anonymized datasets. We now test our anonymization method on graph-based intrusion detection methods such as those that rely on semantic networks [5]. Semantic networks are graphs with nodes representing attacks or benign activities and edges representing the semantic links between them, and they are called **semantic linked networks (SLN)**.

The stronger the relationship between nodes, the higher the possibility they co-occur under a particular context. Consequently, observing one suspicious node can help proactively avoid another. We generate two types of SLNs: The first one was based on the original trace data, while the second one on the anonymized data. Logged labeled flows are anonymized, then classification techniques are applied on the anonymized flows. Then, the SLN generated from anonymized data is used to identify intrusions. We finally compare the SLN generated with anonymized data, against the SLN built over original data. More details applying SLNs for intrusion detection can be found in our previous work [5]. The objective of creating SLNs using the original data is not to evaluate its results in the intrusion detection area, but to use it as a ground truth to compare with the SLNs created using the anonymized traces.

4 EXPERIMENTS AND EVALUATION

In this section, we present the results of the experiments that have been conducted to test the effectiveness of our anonymization approach. Two sets of experiments are described:

—The first set of experiments evaluates our approach by measuring privacy and accuracy. Two different datasets are anonymized, and privacy is measured on the resulting datasets. In addition, accuracy is measured on a testing data with non-anonymized flows using a training data before and after anonymization. Furthermore, we measure the robustness of the approach when the anonymized data is used in graph-based intrusion detection techniques, versus when the original non-anonymized data was used to generate such techniques;
—The second set of experiments measures the immunity of the proposed techniques against data injection attacks. We measure the recovery rate of several patterns that are injected in the datasets before anonymization.

Objectives of experiments: In the experimental evaluation, we prove that our model:

—works reliably and is accurate enough while preserving privacy, when compared with other approaches;
—is immune against Data Injection Attacks;
—works very well when applied on graph-based intrusion detection techniques.

4.1 Datasets

PREDICT Repository: PREDICT (A Protected REpository for Defense of Infrastructure against Cyber Threats) has shared real-world datasets for cyber security research to advance the state-of-the-art network security research and development [24]. In our experiments, we used packet captures from the 2013 National Collegiate Cyber Defense Competition (nccdc.org). We created a software application to generate flows from packet captures and correlate the created flows with alerts generated by the Snort Intrusion Detection System [13]. We generated a total of 400,893 benign and suspicious flows to use in our experiments. University of Twente Dataset: The second dataset provided by Sperotto et al. was created at the University of Twente by monitoring a honeypot for HTTP, SSH, and FTP traffic [41]. We selected 401,732 suspicious flows from this dataset with the corresponding alerts. Since the PREDICT dataset contains mostly normal flows and the Twente dataset mostly attack flows, we draw a random sample from each dataset and combine them to create two new mixed datasets. The combined datasets are:

—Dataset 1: 70% PREDICT dataset + 30% Twente dataset
—Dataset 2: 50% PREDICT dataset + 50% Twente dataset

These two datasets are further partitioned into training (70%) and evaluation (30%) parts. The evaluation part was not anonymized

4.2 Evaluation Measures

Accuracy: We employ several accuracy measures to validate the effectiveness of our anonymization algorithms such as TP Rate, FP Rate, Precision, Recall, F-Measure, and **ROC (Receiver Operating Characteristic)** area.

Privacy: We use conditional privacy to measure the privacy of anonymized traffic data [3]. This measure depends on mutual information between the raw and anonymized records at a certain confidence level; while information loss is related to the amount of mismatch among the records before and after, conditional privacy

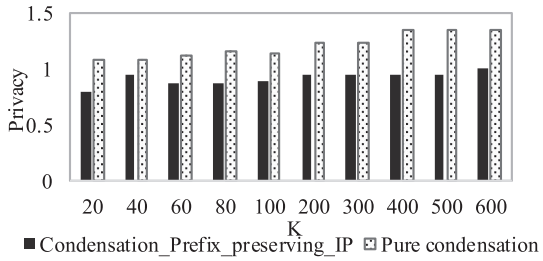


Fig. 7. Privacy results of condensation-based anonymization techniques using Dataset 1.

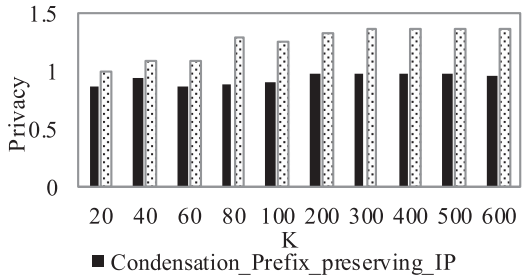


Fig. 8. Privacy results of condensation-based anonymization techniques using Dataset 2.

is based on differential entropy of a random variable. The differential entropy of A given $B = b$ is

$$h(A|B) = \int_{\Omega_{A,B}} f_{A,B}(a,b) log_2 f_{A|B=b}(a) da\, db \tag{1}$$

where A is a random variable that describes the data, and B is a variable that gives information on A. $\Omega_{A,B}$ identifies the domain of A and B. Therefore, the average conditional privacy of A given B is

$$\prod(A|B) = 2^{h(A|B)} \tag{2}$$

if D_i is the attribute value of the original data and $D_i^{'}$ is the value after anonymization. The conditional privacy for anonymizing that attribute is $2^{h(D_i|D_i^{'})}$, where $h(D_i D_i^{'})$ is the conditional entropy of the original data given the anonymized data. Conditional privacy is calculated and averaged over all attributes.

4.3 Results

4.3.1 Privacy Results. Figures 7 and 8 show the conditional privacy results for the anonymized datasets generated by the original condensation technique (described in Section 3.5) and our modified one (Algorithm 2). This experiment measures the effect of increasing the cluster size (k) on the values of conditional privacy when using generalization approaches such as condensation. We utilized the pure condensation but without preserving the Prefix of IP. Then, we utilized condensation, and we preserved the prefix of IP addresses.

Two main conclusions can be drawn from those figures. First, the values of conditional privacy get higher when we increase k. Pure condensation attains higher privacy values than condensation with prefix-preservation. Source and destination IP address have a significant contribution to the higher privacy values in the case of pure condensation. However, the prefixes of IP addresses are not preserved using pure condensation, which leads to more information loss and higher values of conditional privacy. The second set of privacy experiments compares different anonymization techniques, including our differential privacy approach. Figure 9

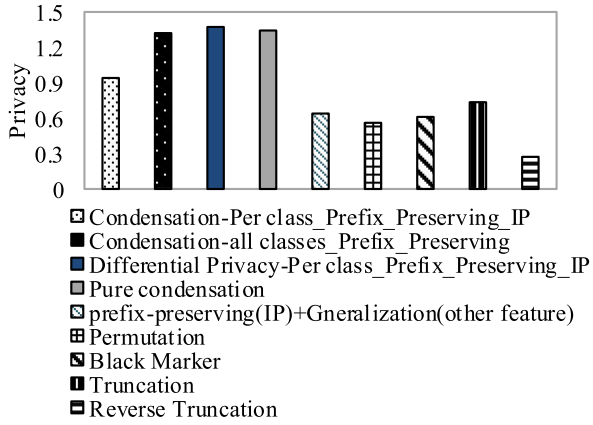


Fig. 9. Privacy results of different anonymization methods with Dataset 1.

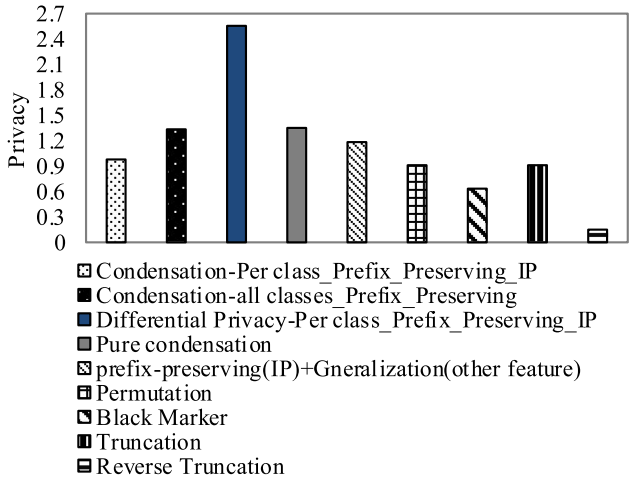


Fig. 10. Privacy results of different anonymization methods with Dataset 2.

shows the privacy measures for the Dataset 1 using different techniques. Based on this measure, our technique (Algorithm 2) performed better than most of existing techniques. We utilized three types of condensation approaches: First, we performed typical condensation but without preserving the prefix of IP. Then, we performed typical condensation, but we performed prefix-preserving anonymization on IP addresses. Finally, we performed our per-class condensation method with IP prefix-preserving and differential private per-class condensation with IP prefix preservation. It is observed that pure condensation attains higher privacy values than prefix-preserving condensation. The per-class condensation method with differential privacy approach outperformed all other methods. The experiment results using Dataset 2, shown in Figure 10, are similar, but our approach reveals much higher levels of privacy compared to the other approaches.

4.3.2 Accuracy Results. We ran several experiments to measure and compare accuracy on anonymized vs. original data. We utilize **K-Nearest Neighbors (KNN)** classifier to run our experiments. Tables 4 and 5 show the KNN classification results on Dataset 1 and Dataset 2, respectively. In terms of accuracy, our approach

Table 4. Dataset 1 Experiment-KNN Classification on Anonymized and Original data

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.98	0.013	0.981	0.98	0.98	0.984	Attack
	0.987	0.02	0.987	0.987	0.987	0.984	Normal
Original	**0.984**	**0.017**	**0.984**	**0.984**	**0.984**	**0.984**	**Avg**
	0.941	0.059	0.961	0.941	0.951	0.941	Attack
	0.941	0.059	0.913	0.941	0.927	0.941	Normal
Condensation-per -Class_Prefix-Preserving_IP	**0.941**	**0.059**	**0.942**	**0.941**	**0.941**	**0.941**	**Avg**
	0.628	0.582	0.62	0.628	0.624	0.523	Attack
	0.418	0.372	0.426	0.418	0.422	0.523	Normal
Condensation-All Classes_Prefix-Preserving_IP	**0.545**	**0.498**	**0.543**	**0.545**	**0.544**	**0.523**	**Avg**
	0.941	0.059	0.96	0.941	0.95	0.94	Attack
	0.941	0.059	0.913	0.941	0.927	0.94	Normal
Differential Privacy-per-Class_Prefix-Preserving_IP	**0.941**	**0.059**	**0.941**	**0.941**	**0.941**	**0.94**	**Avg**
	0.691	0.612	0.631	0.691	0.66	0.54	Attack
	0.388	0.309	0.454	0.388	0.418	0.54	Normal
Pure Condensation	0.571	0.491	0.56	0.571	0.564	0.54	Avg
	1	1	0.602	1	0.752	0.5	Attack
	0	0	0	0	0	0.5	Normal
Prefix-Preserving (IP)+ Generalization (Other_Features)	**0.602**	**0.602**	**0.362**	**0.602**	**0.452**	**0.5**	**Avg**
	0.999	1	0.602	0.999	0.751	0.5	Attack
	0	0.001	0.048	0	0	0.5	Normal
Permutation	**0.602**	**0.602**	**0.381**	**0.602**	**0.452**	**0.5**	**Avg**
	1	1	0.602	1	0.752	0.5	Attack
	0	0	0	0	0	0.5	Normal
Black Marker	0.602	0.602	0.362	0.602	0.452	0.5	Avg
	0.578	0.506	0.633	0.578	0.604	0.577	Attack
	0.494	0.422	0.436	0.494	0.463	0.577	Normal
Truncation	**0.544**	**0.473**	**0.555**	**0.544**	**0.548**	**0.577**	**Avg**
	0.082	0.163	0.432	0.082	0.137	0.46	Attack
	0.837	0.918	0.376	0.837	0.519	0.46	Normal
Reverse Truncation	**0.382**	**0.463**	**0.41**	**0.382**	**0.289**	**0.46**	**Avg**

when applied with prefix-preserving approach to anonymize data (Condensation-per-Class, differential private-Condensation-per-Class) achieves the highest accuracy compared to other techniques. It is evident that while some techniques such as Black Marker achieve acceptable privacy levels, they lead to high information loss, as demonstrated by our privacy results. Therefore, the results when such approaches are used are low compared to other techniques. It is also noticed that there is a significant difference between approaches that belong to the same category. For instance, truncation attains higher accuracy compared to reverse-truncation. Reverse truncation sets the most significant bits to zero, therefore, the predictability of features is significantly affected after anonymization.

The results clearly indicate the importance of prefix-preserving approach to decrease the amount of information loss. Consequently, all approaches that apply our prefix-preserving algorithm attain higher accuracy values. In addition, the prefix-preserving differential privacy algorithm achieves the best results in terms of accuracy. Contrary to approaches such as Black Marker and Truncation, the results of the differential privacy algorithm are consistent across both datasets when comparing the results shown in Tables 4 and 5.

Table 5. Dataset 2 Experiment-KNN Classification on Anonymized and Original Data

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.991	0.013	0.991	0.991	0.991	0.989	Attack
	0.987	0.009	0.987	0.987	0.987	0.989	Normal
Original	**0.984**	**0.011**	**0.989**	**0.989**	**0.989**	**0.989**	**Avg**
	0.954	0.118	0.917	0.954	0.935	0.918	Attack
	0.882	0.046	0.934	0.882	0.907	0.918	Normal
Condensation-per-Class_Prefix-Preserving_IP	**0.924**	**0.088**	**0.924**	**0.924**	**0.923**	**0.918**	**Avg**
	0.553	0.562	0.575	0.553	0.564	0.495	Attack
	0.438	0.447	0.416	0.438	0.427	0.495	Normal
Condensation-All Classes_Prefix-Preserving_IP	**0.504**	**0.514**	**0.508**	**0.504**	**0.506**	**0.495**	**Avg**
	0.975	0.125	0.915	0.975	0.944	0.945	Attack
	0.875	0.025	0.962	0.875	0.916	0.945	Normal
Differential Privacy-per-Class_Prefix-Preserving_IP	**0.933**	**0.083**	**0.935**	**0.933**	**0.932**	**0.945**	**Avg**
	0.662	0.597	0.603	0.662	0.631	0.532	Attack
	0.403	0.338	0.464	0.403	0.431	0.532	Normal
Pure condensation	0.553	0.488	0.545	0.553	0.547	0.532	Avg
	1	1	0.579	1	0.733	0.67	Attack
	0	0	0	0	0	0.67	Normal
Prefix-Preserving (IP)+ Generalization (Other_Features)	0.579	0.579	0.335	0.579	0.424	0.67	Avg
	0.083	0.31	0.27	0.083	0.127	0.387	Attack
	0.69	0.917	0.354	0.69	0.468	0.387	Normal
Permutation	0.339	0.566	0.305	0.339	0.271	0.387	Avg
	0	0	0	0	0	0.5	Attack
	1	1	0.421	1	0.593	0.5	Normal
Black Marker	0.421	0.421	0.178	0.421	0.25	0.5	Avg
	0.499	0.396	0.634	0.499	0.559	0.595	Attack
	0.604	0.501	0.468	0.604	0.527	0.595	Normal
Truncation	0.544	0.44	0.564	0.544	0.546	0.595	Avg
	0.906	0.9	0.58	0.906	0.708	0.503	Attack
	0.1	0.094	0.437	0.1	0.163	0.503	Normal
Reverse Truncation	0.567	0.56	0.52	0.567	0.478	0.503	Avg

We ran another statistical analysis on the original and de-identified datasets and compared the correlation coefficients resulted by analysis run on the original datasets to the coefficients resulted by analysis run on the anonymized datasets. For this measure, coefficients that changed direction before CFTP anonymization compared to after anonymization affects utility. The smaller number of coefficients that changed direction means better utility preservation. Preserving correlation between features is important to decide if applying diverse anonymization techniques to different features affects associations between features. We used Pearson Correlation in this experiment. It measures the linear association between two nodes f_i and f_j and is defined as $cov(\sigma_{f_i}, \sigma_{f_j}) / \sigma_{f_i} \times \sigma_{f_j}$, where cov is the covariance and σ is the standard deviation. The direction of coefficients is preserved between most features after anonymization, as shown in Table 7 (after data anonymization), compared to Table 6 (before data anonymization). There are very few features that lost correlations with other features such as time features. Results are reported for the first dataset, but there are no significant differences when correlation analysis is applied to the second dataset.

Table 6. Dataset 1. Correlation Coefficients before Anonymization

	SRC IP	DST IP	PACKETS	OCTETS	START TIME	END TIME	SRC PORT	DST PORT	TCP FLAGS	DURATION
SRC_IP	1	−0.04	0	0	0.41	0.41	0.28	−0.12	−0.25	0.05
DST_IP	−0.04	1	0.01	0	0.33	0.33	−0.1	0.22	−0.22	0.04
PACKETS	0	0.01	1	0	−0.02	−0.02	−0.01	−0.02	−0.01	0
OCTETS	0	0	0	1	0.01	0.01	0	0	0	0
START_TIME	0.41	0.33	−0.02	0.01	1	0.4	0.33	0.1	−0.61	0.15
END_TIME	0.41	0.33	−0.02	0.01	0.4	1	0.33	0.1	−0.61	0.15
SRC_PORT	0.28	−0.1	−0.01	0	0.33	0.33	1	−0.47	−0.61	−0.01
DST_PORT	−0.12	0.22	−0.02	0	0.1	0.1	−0.47	1	0.14	−0.03
TCP_FLAGS	−0.25	−0.22	−0.01	0	−0.61	−0.61	−0.61	0.14	1	−0.15
DURATION	0.05	0.04	0	0	0.15	0.15	−0.01	−0.03	−0.15	1

Table 7. Dataset 1. Correlation Coefficients after Anonymization

	SRC IP	DST IP	PACKETS	OCTETS	START TIME	END TIME	SRC PORT	DST PORT	TCP FLAGS	DURATION
SRC_IP	1	−0.01	0.06	0.04	0.14	0.17	0.24	−0.08	−0.17	0.1
DST_IP	−0.01	1	0.03	0	0.15	0.05	−0.04	0.16	−0.19	−0.08
PACKETS	0.06	0.03	1	0	0.05	−0.05	−0.03	−0.03	−0.02	0.2
OCTETS	0.04	0	0	1	−0.05	0.01	0.01	0.01	0	0.02
START_TIME	0.14	0.15	0.05	−0.05	1	−0.2	0.36	−0.11	−0.9	0.03
END_TIME	0.17	0.05	−0.05	0.01	−0.2	1	0.33	−0.19	−0.09	−0.04
SRC_PORT	0.24	−0.04	−0.03	−0.01	0.36	0.33	1	−0.53	−0.53	−0.02
DST_PORT	−0.08	0.16	−0.03	0.01	−0.11	−0.19	−0.53	1	0.18	−0.03
TCP_FLAGS	−0.17	−0.19	−0.02	0	−0.09	−0.09	−0.53	0.18	1	−0.13
DURATION	0.1	0.08	0.2	0.02	0.03	−0.04	−0.02	−0.03	−0.13	1

4.3.3 Results on Graph-based Intrusion Detection Techniques. In this set of experiments, we evaluate the two types of SLNs created using the original and the anonymized datasets. Different Intrusion detection evaluation metrics are used to measure the success rate of identifying attacks using SLNs when applied on top of K-NN classification techniques. The two types of initial SLNs created before and after anonymization are shown in Figures 11 and 12. The two SLNs have the exact same structure. In addition, the strengths of relationships between attacks (values on the graph edges) are very close on the SLNs before and after anonymization. When SLNs are used for attack detection purposes, typically they increase recall values. We experimented with SLNs using different values of a threshold t, which specifies the minimum cutoff value to include relevant nodes to the starting one initially predicted by the K-NN classifier. We compare the accuracy of identifying attacks using SLNs on anonymized and original datasets in terms of Precision, Recall, F-measure, and **Receiver Operating Characteristic (ROC)**. The ROC is a popular measure that has been used to compare intrusion detection techniques and plot TP and FP rates associated with various operating points when different intrusion detection techniques are used. The values of TP and FP rates (TPR and FPR) are calculated as:

$$TPR = \frac{TP}{TP + FN} \tag{3}$$

$$FPR = \frac{FP}{FP + TN} \tag{4}$$

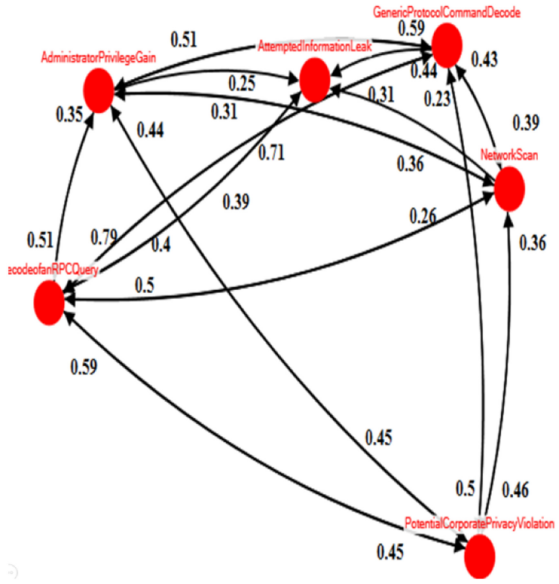


Fig. 11. SLN before anonymization.

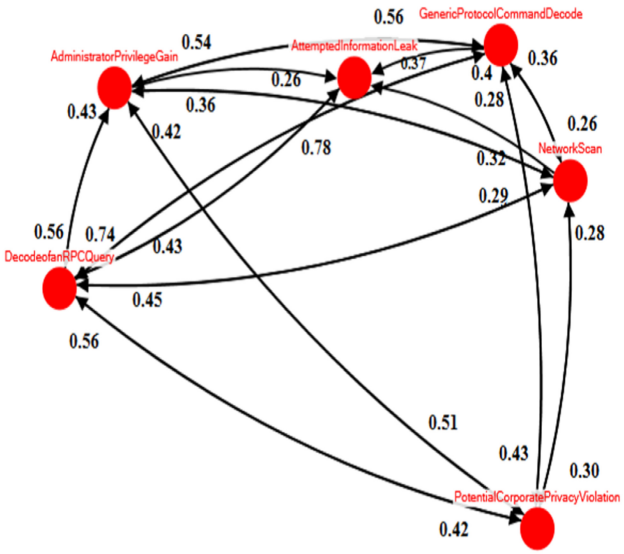


Fig. 12. SLN after anonymization.

The results of this experiment are shown in Figures 13(a)–13(e). The results of the experiments on Dataset 1 show that there are no significant differences of Precision, Recall, and F-measure values before and after anonymization. The ROC curves for each dataset are shown in Figure 13(e) and are clearly close, and the differences between original and anonymized datasets are very small in terms of TPR and FPR. Results on the second dataset are very similar to the first dataset, so they are omitted.

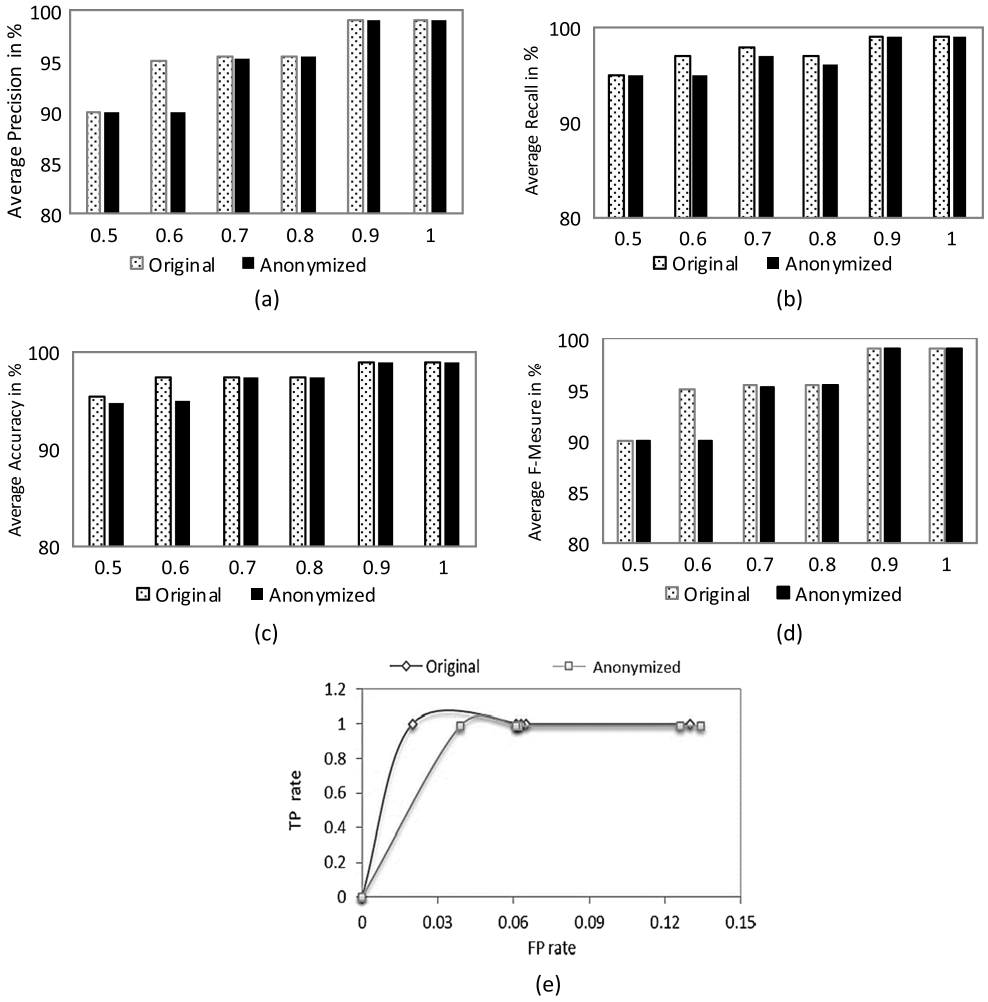


Fig. 13. (a) Precision values for SLNs—original vs. anonymized Dataset 1. (b). Recall values for SLNs—original vs. anonymized Dataset 1. (c). Accuracy values for SLNs—original vs. anonymized Dataset 1. (d). F-measure for SLNs—original vs. anonymized Dataset 1. (e). ROC curve for SLNs—original vs. anonymized Dataset 1.

4.3.4 Results on Injection Attacks. We simulate injection attacks by adding records with specific patterns to the original datasets. Then, we run the anonymization algorithms on the two datasets and try to identify the injected records. We then compare the **Injected Pattern Recovery Rate (IPRR)** on various anonymization policies using the following formula:

$$IPRR = \frac{Recovered\ injected\ pattern}{Total\ number\ of\ injected\ patterns} \tag{5}$$

We applied five anonymization policies ($A_1 - A_5$) in addition to our differential privacy approach on both datasets. Those policies are described in Section 3.7. In addition, the five patterns described in Table 8 are injected in both datasets.

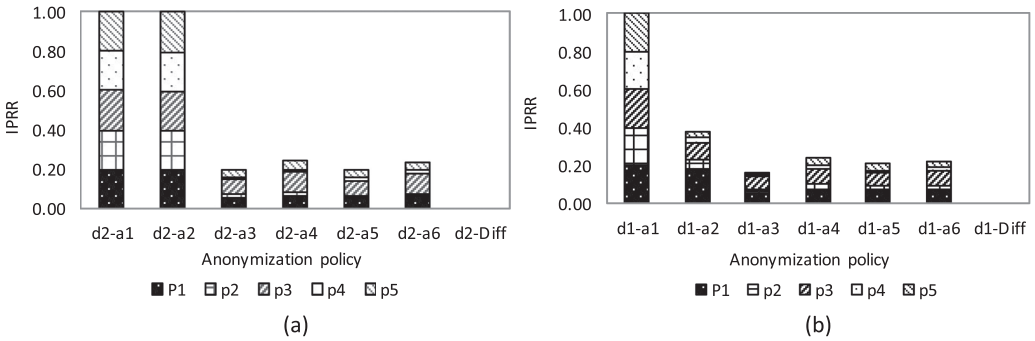


Fig. 14. (a) Testing data injection attacks using various anonymization policies on Dataset 1. (b). Testing data injection attacks using various anonymization policies on Dataset 2.

Table 8. Patterns Injected in the Trace Data

	Packets	Source port	Destination port	Duration	Octets
P1	1	Fixed	80	–	160
P2	5	R(65k)	R(65k)	200	256
P3	110	Fixed	80	200	480[+32]
P4	10	R(65k)	R(65k)	200	832[+32]
P5	50	R(65k)	R(65k)	150+R(300)	1,208[+R(8)]

- Values in square brackets denote the attribute evolution between flows.
- R(x): random number between 1 and x.
- Total number of injected flows is 650 (130) flows from each pattern.

Patterns 1, 2, and 3 are simpler than patterns 4 and 5. Each pattern consists of 130 records, with a total of 650 injection attempts. Those patterns work as a fingerprint in the original data to be discovered later after anonymization. The objective of this process is to discover the immunity of the anonymization algorithms against injection attacks [9]. The results of this experiment on both datasets are shown in Figures 14(a) and 14(b). Permutation-based anonymization policies lead to the highest recovery ratio compared to other approaches; KNN search discovers most records for patterns 1 and 3. As the complexity of the permutation function used in the anonymization policy increases, IPRR values decreases. However, KNN search still discovers a significant percentage of the injected patterns. Compared to these anonymization policies, when our differential privacy-based anonymization policy is used, zero records are recovered, testifying to the robustness of our approach against injection attacks.

5 CONCLUSIONS AND FUTURE WORK

This article proposes a method that utilizes differential privacy to anonymize network traces, and it has the following characteristics: It has a very strong privacy guarantee; it is robust when used in generating attack prediction models even when sophisticated intrusion detection techniques such as graph-based approaches are used; it does not add any burden to data analyzer. Data analyzer can analyze data as it is. Our experiments show that using differential privacy for anonymization produces superior results compared to existing techniques in terms of privacy-utility tradeoff. Our work so far only focuses on network-trace data. A plethora of sensitive information is often contained in a variety of formats, and security experts often need to analyze multiple types of security data. For example, to detect an **APT (Advanced Persistent Threat)**, experts often need to analyze network traces, system logs, and web logs. Like network trace data, other types of cyber security data often contain sensitive information, such as username, IP address, and node name, which can be found in event/system

call logs, authorization/connection logs, web logs, malware execution traces, and so on. It is important to extend our technique to anonymize such additional security data in the near future and expand the spectrum of our approach.

REFERENCES

[1] Center for Applied Internet Data Analysis (CAIDA). 2015. Retrieved from https://www.caida.org/tools/taxonomy/anonymization.xml.

[2] C. C. Aggarwal and S. Y. Philip. 2004. A condensation approach to privacy preserving data mining. In *Advances in Database Technology-EDBT 2004*. Springer, 183–199.

[3] R. Agrawal and R. Srikant. 2000. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*.

[4] A. Aleroud, Z. Chen, and G. Karabatis. 2016. Network trace anonymization using a prefix-preserving condensation-based technique (short paper). In *Proceedings of the International Symposium on Secure Virtual Infrastructures Cloud and Trusted Computing (OTM'16)* Springer International Publishing, 934–942.

[5] A. F. Aleroud and G. Karabatis. 2018. Queryable semantics to detect cyber-attacks: A flow-based detection approach. *IEEE Trans. Syst., Man,Cyber.: Syst.* 48, 2 (2018), 207–223.

[6] T. Brekne and A. Årnes. 2005. Circumventing IP-address' pseudonymization. In *Communications and Computer Networks*. ACTA Press, 43–48.

[7] T. Brekne, A. Arnes, and A. Øslebø. 2006. Anonymization of IP traffic monitoring data: Attacks on two prefix-preserving anonymization schemes and some proposed remedies. In *Proceedings of the 5th International Conference on Privacy Enhancing Technologies (PET'05)*. Springer, 179–196.

[8] T. Brekne, A. Årnes, and A. Øslebø. 2005. Anonymization of ip traffic monitoring data: Attacks on two prefix-preserving anonymization schemes and some proposed remedies. In *Proceedings of the International Workshop on Privacy Enhancing Technologies*. Springer, 179–196.

[9] M. Burkhart, D. Schatzmann, B. Trammell, E. Boschi, and B. Plattner. 2010. The role of network trace anonymization under attack. *ACM SIGCOMM Comput. Commun. Rev.* 40, 1 (2010), 5–11.

[10] S. E. Coull, C. V. Wright, A. D. Keromytis, F. Monrose, and M. K. Reiter. 2008. Taming the devil: Techniques for evaluating anonymized network data. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'08)*.

[11] S. E. Coull, C. V. Wright, F. Monrose, M. P. Collins, and M. K. Reiter. 2007. Playing devil's advocate: Inferring sensitive information from anonymized network traces. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'07)*. 35–47.

[12] R. Dahlberg and T. Pulls. 2016. Standardized Syslog Processing: Revisiting Secure Reliable Data Transfer and Message Compression. Retrieved from Karlstads universitet website: http://urn.kb.se/resolve?urn=urn:nbn:se:kau:diva-45392.

[13] T. Ding, A. Aleroud, and G. Karabatis. 2015. Multi-granular aggregation of network flows for security analysis. In *Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI'15)*. IEEE, 173–175.

[14] J. Domingo-Ferrer and V. Torra. 2005. Ordinal, continuous and heterogeneous k-anonymity through microaggregation. *Data Mining Knowl. Discov.* 11, 2 (2005), 195–212.

[15] C. Dwork. 2008. Differential privacy: A survey of results. In *Proceedings of the International Conference on Theory and Applications of Models of Computation*. Springer, 1–19.

[16] J. Fan, J. Xu, and M. H. Ammar. 2004. Crypto-PAn: Cryptography-based prefix-preserving anonymization. *Comput. Netw.* 46, 2 (2004).

[17] J. Fan, J. Xu, M. H. Ammar, and S. B. Moon. 2004. Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. *Comput. Netw.* 46, 2 (2004), 253–272.

[18] T. Farah. 2013. *Algorithms and Tools for Anonymization of the Internet Traffic*. Ph.D. Dissertation, Simon Fraser University Applied Sciences: School of Engineering Science.

[19] M. Foukarakis, D. Antoniades, and M. Polychronakis. 2009. Deep packet anonymization. In *Proceedings of the 2nd European Workshop on System Security*. ACM, 16–21.

[20] T. Gamer, C. Mayer, and M. Schöller. 2008. PktAnon–A generic framework for profile-based traffic anonymization. *PIK-Praxis der Informationsverarbeitung und Kommunikation* 31, 2 (2008), 76–81.

[21] S. Gattani. 2008. *Reference Models for Network Trace Anonymization*. Master's Thesis. Iowa State University.

[22] S. Ghiasvand and F. M. Ciorba. 2017. Anonymization of system logs for privacy and storage benefits. *arXiv preprint arXiv:1706.04337*.

[23] H. Holen. 2020. RIPE (European IP Networks). Retrieved from https://www.ripe.net/.

[24] E. Kenneally. 2020. Information marketplace for policy and analysis of cyber-risk & trust. Retrieved from https://www.dhs.gov/csd-impact.

[25] J. King, K. Lakkaraju, and A. Slagell. 2009. A taxonomy and adversarial model for attacks against network log anonymization. In *Proceedings of the ACM Symposium on Applied Computing. ACM*, 1286–1293.

[26] E. Kohler. 2017. IPSUMDUMP and IPAGGCREATE. Retrieved from http://www.read.seas.harvard.edu/~kohler/ipsumdump.

[27] D. Koukis, S. Antonatos, D. Antoniades, E. P. Markatos, and P. Trimintzios. 2006. A generic anonymization framework for network traffic. In *Proceedings of the IEEE International Conference on Communications (ICC'06)* IEEE, 2302–2309.

[28] G. Kuenning and E. L. Miller. 2003. *Anonymization Techniques for URLs and Filenames.* Technical Report, TR UCSC-CRL-03-05, University of California at Santa Cruz.
[29] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. 2007. l-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data* 1, 1 (2007), 3.
[30] D. Meyer. 2013. University of Oregon route views project. Retrieved from http://www.routeviews.org/.
[31] G. Minshall. 1997. Tcpdpriv. Retreived from http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html.
[32] M. Mohammady, L. Wang, Y. Hong, H. Louafi, M. Pourzandi, and M. Debbabi. 2018. Preserving both privacy and utility in network trace anonymization. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security.* 459–474.
[33] M. Mohammady, L. Wang, Y. Hong, H. Louafi, M. Pourzandi, and M. Debbabi. 2018. Preserving both privacy and utility in network trace anonymization. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security.* ACM, 459–474.
[34] R. Pang, M. Allman, V. Paxson, and J. Lee. 2006. The devil and packet trace anonymization. *ACM SIGCOMM Comput. Commun. Rev.* 36, 1 (2006), 29–38.
[35] R. Pang and V. Paxson. 2003. A high-level programming environment for packet trace anonymization and transformation. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications.* ACM, 339–351.
[36] B. Shebaro and J. R. Crandall. 2011. Privacy-preserving network flow recording. *Dig. Investig.* 8 (2011), S90–S100.
[37] A. Slagell, J. Wang, and W. Yurcik. 2004. Network log anonymization: Application of crypto-PAn to Cisco netflows. In *Proceedings of the Workshop on Secure Knowledge Management.*
[38] A. Slagell and W. Yurcik. 2005. Sharing computer network logs for security and privacy: A motivation for new methodologies of anonymization. In *Proceedings of the Workshop of the 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks.* IEEE, 80–89.
[39] A. J. Slagell, K. Lakkaraju, and K. Luo. 2006. FLAIM: A multi-level anonymization framework for computer and network logs. In *Proceedings of the 20th Large Installation System Administration Conference (LISA'18).* 3–8.
[40] A. J. Slagell, Y. Li, and K. Luo. 2005. Sharing network logs for computer forensics: A new tool for the anonymization of netflow records. In *Proceedings of the Workshop of the 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks.* IEEE, 37–42.
[41] A. Sperotto, R. Sadre, F. Vliet, and A. Pras. 2009. A labeled dataset for flow-based intrusion detection. In *Proceedings of the 9th IEEE International Workshop on IP Operations and Management((IPOM'09)*, G. Nunzi, C. Scoglio and X. Li Eds., 39–50. DOI : http://dx.doi.org/10.1007/978-3-642-04968-2_4.
[42] L. Sweeney. 2002. k-anonymity: A model for protecting privacy. *Int. J. Uncert., Fuzz. Knowl.-based Syst.* 10, 05 (2002), 557–570.
[43] K. Tan, J. Yeo, M. E. Locasto, and D. Kotz. 2011. Catch, Clean, and Release: A Survey of Obstacles and Opportunities for Network Trace Sanitization. Dartmouth Scholarship. 3162. Retrieved from https://digitalcommons.dartmouth.edu/facoa/3162.
[44] J. Vaidya, Y. M. Zhu, and C. W. Clifton. 2005. *Privacy Preserving Data Mining (Advances in Information Security).* Springer-Verlag New York, Inc.
[45] N. Van Dijkhuizen and J. Van Der Ham. 2015. Online event registration with minimal privacy violation. *A study into Privacy Enhanced Filtering Techniques.* University of Amsterdam. Retrieved from https://homepages.staff.os3.nl/~delaat/rp/2014-2015/p95/report.pdf.
[46] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel. 2010. A practical attack to de-anonymize social network users. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'10)* IEEE, 223–238.
[47] J. Xu, J. Fan, M. H. Ammar, and S. B. Moon. 2002. Prefix-preserving ip address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In *Proceedings of the 10th IEEE International Conference on Network Protocols.* IEEE, 280–289.
[48] J. Xu, J. Fan, M. H. Ammar, and S. B. Moon. 2002. Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In *Proceedings of the 10th IEEE International Conference on Network Protocols.* IEEE, 280–289.
[49] T.-F. Yen, X. Huang, F. Monrose, and M. K. Reiter. 2009. Browser fingerprinting from coarse traffic summaries: Techniques and implications. In *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment.* Springer, 157–175.

Received April 2020; accepted September 2020